;

מערכת לניהול פרוייקט באמצעות מתודולגיה Agile

אורן טביב
027154780

;

1 תקציר........................ 4

2 מבוא........................ 4

3 Software Requirement Specification..7

3.1 מבוא..7

3.1.1 ***מטרת המסמך***..*7*

3.1.2 ***תיחום המוצר*** *7*

3.1.3 ***הגדרות וקיצורים*** *8*

3.1.4 ***הפניות למסמכים נוספים***..*8*

3.1.5 ***סקירה כללית של המסמך*** *8*

3.2 תיאור כללי...10

3.2.1 ***מבט-על על המערכת***..*10*

3.2.2 ***. פונקציות המערכת*** *11*

3.2.3 ***.אפייני המשתמש במערכת*** *13*

3.2.4 ***אילוצים כללים***...*16*

3.2.5 ***הנחות ותלויות*** *16*

3.3 מפרט הדרישות מהמערכת 16

3.3.1 ***דרישות פונקציונליות***..*16*

3.3.1.1 תיעוד דרישות המערכת בפרוייקט...17

3.3.1.2 תיעוד Backlog..21

3.3.1.3 תיעוד משימות...27

3.3.1.4 תיעוד Sprint..31

3.3.1.5 ניהולהרשאות...38

3.3.1.6 ניהול, Login 44

3.3.1.7 ביצוע גיבוי לבסיסי הנתונים 47

3.3.1.8 תיעוד Dialy Meeting..47

3.3.1.9 הצג דוחות..50

3.3.2 ***דרישות ממשק חיצוני*** *51*

3.3.2.1 ממשק משתמש...51

3.3.2.2 ממשק חומרה 51

3.3.2.3 ממשקי תוכנה 52

3.3.2.4 ממשקי תקשורת...52

3.3.3 ***דרישות ביצועים*** *52*

;

*3.3.4 **מגבלות תכנון**...52*

*3.3.5 **מאפיינים** 53*

3.3.5.1 זמינות 53

3.3.5.2 אבטחה..53

3.3.5.3 תחזוקה 53

*3.3.6 **דרישות נוספות** 54*

3.4 פיתוחים עתידיים 55

3.5 נספח...56

4 Software System Design 59

5 Software Detail Design 67

6 Operational Requirement Specification……. 75

7 סיכום והצעות להמשך מחקר ופיתוח........................ 85

8 נספח... 86

8.1 קוד הפרוייקט 86

8.2 הצעת הפרוייקט.. 222

9. מקורות... 227

;

1 תקציר

הפרוייקט אמור לספק לראש צוות /מנהל פיתוח / מפתח ,אפלקצייה לניהול הפרוייקט במתודלגיה Agile
האפלקצייה אמורה לעזור לתעד את כל שלביי הפיתוח ע"פ מתודולוגיית Agile כולל מעקב על הפרוייקט.
האפלקצייה אמורה להיות מותאמת לכל שלביי הפיתוח של המתודלגיה בשלב ראשון מותאמת לפרוייקט אחד ובהמשך לכמה פרוייקטים.
האפלקצייה אמורה לתת אפשרות לתעד את דרישות המערכת את הישבות שנערכות ואת התקדמות הפרוייקט .
כל משימה תכיל את סטטוס הפיתוח ותוכל להשתנות מעת לעת.
הפיתוח יחולק לאטרציות (Sprint) וכל אטרציה תכיל את הדרישות שהיא מכילה וכל דרישה תכיל את המשימה שאמורה לכסות את הדרישה.
האפלקציה אמורה לתת אופציה של הפקת דוחות שתנו תמונת מצב כללית על המשימות במערכת.

2 מבוא

משמעות התואר Agile היא זריז, גמיש, והוא מתייחס ליכולתו של גוף להתאים את עצמו כראוי לסביבה משתנה. הניסיון המצטבר ביחס לניהול פרויקטי תוכנה מלמד שרצוי שגם תהליכי פיתוח תוכנה יהיו בעלי תכונות אלה. בפרט, התפיסה האג'ילית לפיתוח תוכנה מבוססת על הנחת העבודה כי תהליכי פיתוח תוכנה מאופיינים בשינויים רבים, ולכן, יש לבנות עבורם מנגנון ניהול המתמודד בהצלחה עם מאפיין זה. בנוסף, התפיסה האג'ילית שמה את הדגש על פיתוח תוכנה איכותית, הן מבחינת קיום דרישות הלקוחות והן מבחינת העדר באגים. רעיונותיה של התפיסה האג'ילית מיושמים באמצעות מספר מתודולוגיות פיתוח תוכנה אג'יליות, כמו למשל, Extreme Programming, Lean Software Development, Crystal, Scrum.

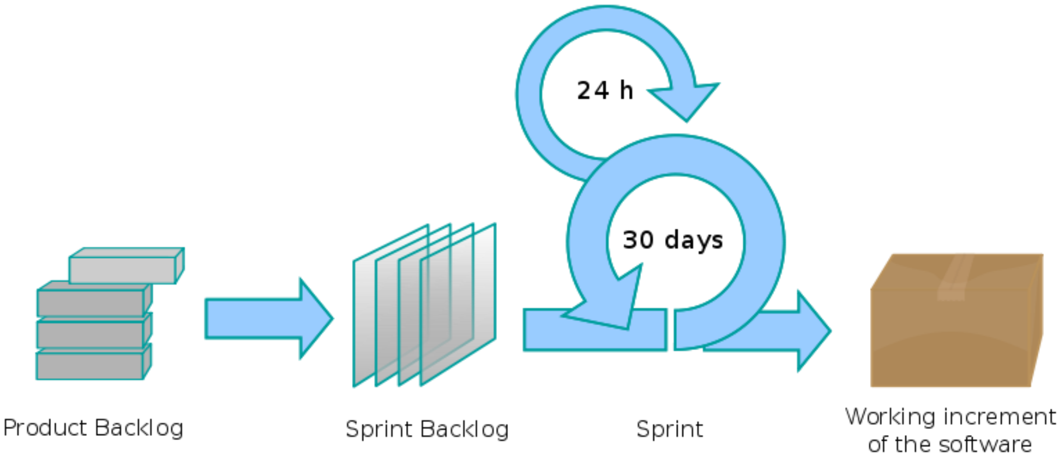
המאפיינים העיקריים של מתודלוגיה :

המאפיינים העיקריים של מתודלוגיה :
א) שביעות רצון הלקוחות
ב) קבוצה טובה עם אחריות גבוה יותר
ג) דרישות קל משקל, גמישות לשינויים מהירים
ד) מחזורים קטנים איטרטיבי
ה) רמה גבוהה של תקשורת
תהליך זריז מאפשר ללקוח לקבל תצוגה מהירה של המוצר וכמו כן מספקת מוקדם ולעיתים קרובות דבר זה נותן יתרון גדול שכן המשוב נעשה בזמן קצר ולא מחכה לסוף הפרוייקט כמו מודל מפל המים .
כמו כן דבר זה נותן ללקוח אופצייה לשינוי שכן ללקוחות יש סיכוי לבצע שינויים בדרישה אם הם חשים כי מה שהם רואים בכלל? לא תואם עם מה שהם מדמיינים. שינויים דרישות דינמי יתרון המקנה גמישות בשלביי הפיתוח.

מתודולוגיית Agile מספקת משלוח מהיר בסביבה דינאמית. זה הטבע מצטבר מספקת התנהגות חסכונית עבור פיתוח המוצר. מתודולוגיה זו לא רק מאפשר לספק אב טיפוס חסכוני, אלא גם מספק הפקטורינג גמישות מחדש. בסביבה תחרותית, העסק תמיד מתחרה עם עסקים אחרים על ידי שחרור מוצרים חדשניים בעולם מהיר דינמי, שיכול להיות נתמך לחלוטין מתודולוגיה זריזה.

;

תהליך פיתוח אג'ילי



1. תחזוקה של רשימת פריטי העבודה לביצוע, מסודרים לפי קדימויות, המכונה עתודת המוצר (Product Backlog).

2. השלמת מנה קבועה של פריטי עתודה בסדרה של איטרציות קצרות המכונות Sprint. משך כל Sprint הוא 4 שבועות, ובסיומו מסופקת תוכנה עובדת למשתמשים.

3. פגישת צוות יומית קצרצרה (עד 15 דקות) המכונה 'Stand up Meeting'. בפגישה מציג כל אחד מחברי הצוות את ההתקדמות, העבודה המתוכננת וקשיים אפשריים. הפגישה מתקיימת לרוב בעמידה.

4. פגישת תכנון ספרינט (Sprint Planning) קצרה שבה מוגדרים פריטי העתודה לאותו Sprint.

5. פגישת ניתוח ספרינט (Sprint Retrospective) קצרה להפקת לקחים מה- Sprint הקודם.

6. השיטה מיושמת בהנחיית Scrum Master שתפקידו העיקרי לסלק מכשולים המפריעים לצוות לעמוד ביעדי הSprint . ה-Scrum Master אינו ראש הצוות (מאחר שמדובר בצוות בהכוונה עצמית), אלא חוצץ בין הצוות לבין השפעות העלולות להפריע לו.

;

7. כדי לאפשר את יצירתם של צוותים בהכוונה עצמית, השיטה מעודדת ריכוז של כל חברי הצוות במיקום אחד, וכן תקשורת מילולית בין חברי הצוות ועם צוותים תומכים.

8. אחד מעקרונות המפתח של Scrum הוא הכרה בכך שאתגרים אמפיריים ביסודם אינם ניתנים לפתרון בשיטות מסורתיות המתבססות על חיזוי או תכנון. מכיוון שכך, שיטת Scrum מאמצת גישה אמפירית, המניחה שלא ניתן להבין או להגדיר את הבעיה במלואה מראש. במקום זאת, השיטה מתמקדת בשיפור יכולתו של הצוות לספק תוצרים במהירות ולהגיב לדרישות העולות תוך כדי התהליך.

;

3 Software Requirement Specification

3.1 מבוא

3.1.1 מטרת המסמך

מסמך זה מתאר את המפרט של מערכת לניהול פיתוח במתודלוגיה Agile. המסמך מציג את הדרישות מהמערכת פונקציונליות ולא פונקציונליות. המסמך מספק את הדברים הבאים:

- הגדרת סביבת העבודה של המערכת.
- הגדרת יכולות המערכת.
- ספסיפיקציה של דרישות המערכת הפונקציונליות והלא פונקציונליות.
- דרישות ממשק משתמש.
- אפשרויות הרחבה עתידיות.

קהל היעד של המסמך הם:

- מתכנתי המערכת שישתמשו באינפורמציה הכתובה כבסיס ליצירת application design.
- מתחזקי המערכת.
- בודקי תוכנה.
- מנהל הפרויקט.
- מזמין המערכת.

3.1.2 תיחום המוצר

מטרת הפרויקט זה לפתח אפליקציה לניהול פיתוח באמצעות מתודולגיות Agile" .

הפרוייקט אמור לספק לראש צוות /מנהל פיתוח / מפתח/ללקוח/ מנהל הפרוייקט,אפלקצייה לניהול הפרוייקט במתודלגיה Agile
האפלקצייה אמורה לעזור לתעד את כל שלביי הפיתוח ע"פ מתודולוגיית Agile כולל מעקב על הפרוייקט.
האפלקצייה אמורה להיות מותאמת לכל שלביי הפיתוח של המתודלגיה בשלב ראשון מותאמת לפרוייקט אחד ובהמשך לכמה פרוייקטים.

המערכת תתמוך בפעולות הבאות:

- תיעוד דרישות המערכת בפרוייקט.
- יצרת Backlog חדש.
- הוספה משימות ל Backlog
- יצרת Sprint חדש.
- הקצאת פעיליות ל Sprint.

;

- עריכת Sprint כולל הפעולות הוספה והורדה פעולות שינוי Status של Sprint.
- שיוך Sprint ל Backlog.
- שיוך משימה ל Sprint
- ניהול הרשאות
- יצירת משתמשים
- קישור בין דרישה .
- הפקת דוחות הנותנות אפשרות לראות את מצב הפרוייקט.

<u>המערכת לא תתמוך בנושאים הבאים למרות קיומם במערכת:</u>

- עקיבות בין דרישות למשימות.
- בדיקת כיסוי של משימות .

<u>אפשרויות הרחבה עתידיות:</u>

- טיפול במספר פרוייקטים בו זמנית.
- טיפול בניעלות רשימות שמספר Client מחוברים בו זמנית.
- אפשרות מידול הפרוייקט באפלקציה (Use Case ,Activity Diagram, Sequence Diagram)
- ביצוע עידכון דרך מערכת חיצונית כגון אינטרנט.

3.1.3 הגדרות וקיצורים

3.1.4 הפניות למסמכים נוספים

- IEEE Std 830-1993:IEE Recommended Parctice for Software Requirements Specification .SH 17038
- מסמך הצעת פרוייקט .
- תרשים ERD של המערכת.

3.1.5 סקירה כללית של המסמך

- <u>תיאור כללי:</u> בחלק זה ינתן תיאור של הסביבה הנדרשת לשם הפעלת האפליקציה – חומרה ותוכנה. תיאור כללי של הפונקציות אותן תמלא המערכת. תיאור של המשתמש במערכת – נסיון, הכשרה ומיומניות נדרשים והשפעתם על המערכת. וכן, תיאורם של אילוצים והגבלות נוספים שמשפיעים על האפליקציה. כמו כן תפורט מסגרת הנחות ותלויות במסגרתן ניתן הניתוח של המערכת.

;

- <u>דרישות מפורטות:</u> בחלק זה ינתן תיאור מפורט של הדרישות הפונקצינליות של המערכת, ממשק המשתמש, ממשק החומרה, התוכנות הנדרשות לשם הפעלת האפליקציה, דרישות ביצועים מהמערכת, אילוצי תכנון, אבטחת איכות המערכת ואבטחת המידע בה וכן תחזוקה עתידית של המערכת.
- <u>פיתוחים עתידיים</u>: בחלק זה ינתנו המלצות המתכננים לאפשרויות פיתוח עתידיות ולמאפיניים שרצוי להוסיף לאפליקציה הנוכחית.

;

3.2 תיאור כללי

בפרק זה ניתן תיאור כללי של המערכת, תיאור מפורט ינתן בפרק 3.

3.2.1 **מבט-על על המערכת**

המערכת נכון לשלב הבסיסי של פיתוחה הינה מערכת עצמאית, אולם המערכת תתוכנן כך שתהיה אפשרות עתידית לחבר באמצעותה ברשת של החומרה הנדרשת:

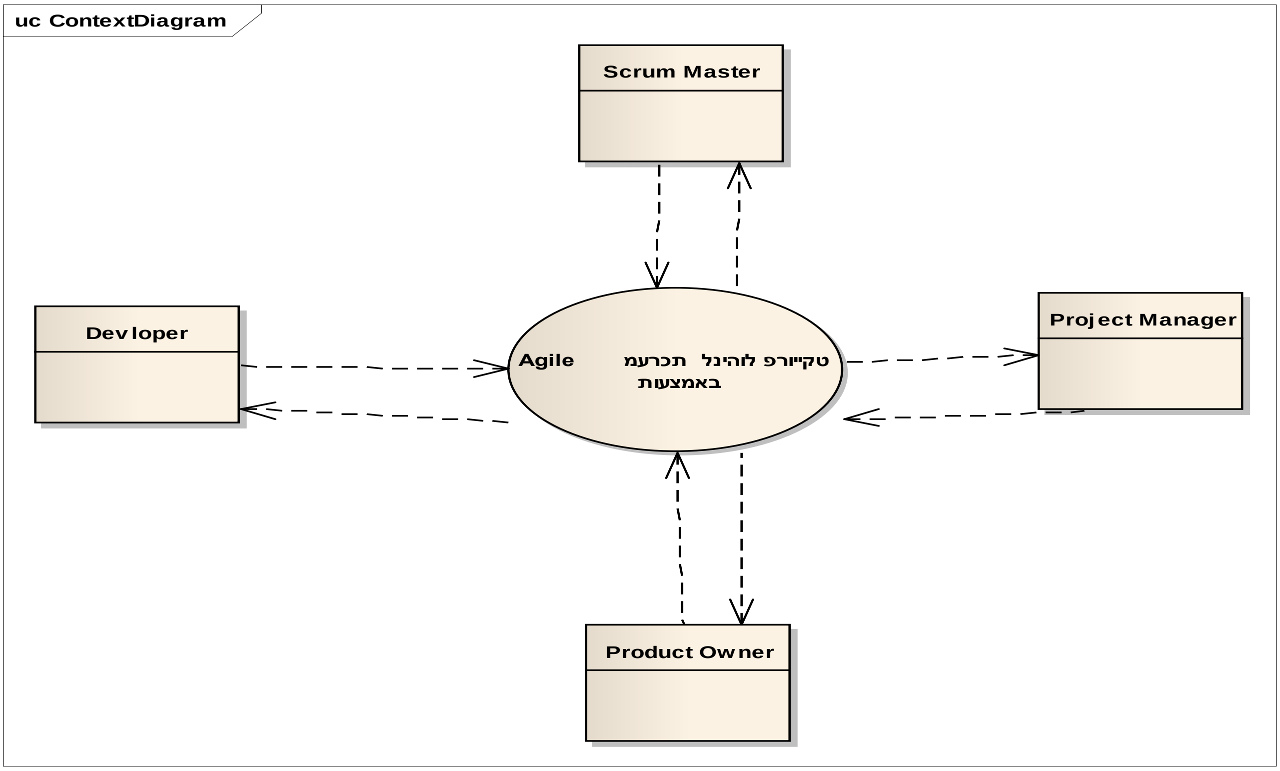
- מחשבי PC כמספר העמדות המבוקשות.
- מחשבי PC דיסק קשיח:
  - עם קיבולת של לפחות 4GB זכרון חיצוני ,
  - זיכרון פנימי 1GB.
  - MHZ1500 CPU .
  - כרטיס רשת .
- מחשב מרכזי עליו ישמרו הנתונים שהוכנסו בעמדות השונות – בעל דיסק קשיח גדול במיוחד ומעבד החזק ביותר שקיים.
- מחשב המרכזי יצאות USB או דיסק על מנת שנוכל ליצא את Data למחשב אחר.
- מחשב גיבוי.
- Router לחיבור המחשבים.
- חומרה נדרשת לדורך חיווט וחיבור מחשבי המסופים למחשב המרכזי.
- Mail server

תוכנה נדרשת:

- מערכת הפעלה Windows XP/Vista / Windows 7.
- תוכנת דואר שתותקן על השרת המרכזי.
- תוכנה לניהול בסיסי הנתונים.
- Microsoft Frame Work 4.0
- SQL Server 2008

;

Context Diagram



Scrum Master: מוודא שתהליך ה Scrum-מתבצע כנדרש.

Devloper: מפתח בצוות.

Project Manager:מנהל ומפקח על תהליכי הפיתוח

Product Owner : עובד מייצג את האינטרס של הלקוח כדי להבטיח שצוות ה Scrum-עושה את הדברים הנכונים מנקודת המבט העסקית.

.

3.2.2 <u>**. פונקציות המערכת**</u>

מכלול הפונקציות מספקות לניהול הפרויקט את כל שלביי הפיתוח דרישות הפרויקט הפונקציות העיקריות שקיימות במערכת הן:

1. **תיעוד דרישות המערכת בפרוייקט**
   - הכנסת דרישות חדשות.
   - עריכת דרישות קיימות.
   - מחיקה דרישה .

2. **תיעוד Backlog.**

;

- יצירת Backlog חדש .
- ערכת Backlog קיים.
- מחיקת Backlog.
- שיוך משימה ל Backlog.
- העברת משימה מ Backlog ל Backlog אחר.

3. . תיעוד משימות:

- יצרת משימות חדשות.
- ערכת משימות חדשות
- מחיקת משימה .

4. תיעוד Sprint:

- יצירת Sprint חדש .
- ערכת Sprint קיים.
- מחיקת Sprint.
- שיוך משימה ל Sprint.
- העברת משימה מ Sprint ל Sprint אחר.

5. ניהול הרשאות:

- הוספת משתמש.
- מחיקת משתמש.
- עריכת משתמש.
- שיוך משתמש לקבוצה קיימת.
- העברת משתמש מקבוצה לקבוצה.

6. ניהול Login

- ניהול כניסה למערכת לפי שם המשתמש הנכנס למערכת.
- הצגת UI לפי ההרשאה.

7. ביצוע גיבוי לבסיסי הנתונים

- גיבוי יומי ל בסיס נתונים

8. תיעוד Dialy meeting

;

- הכנסת סכומיי דיון
- ערכת דיון קיים

9. הפקת דוחות

- הפקת דוחות לפי נושא

3.2.3 **אפייני המשתמש במערכת.**

המשתמשים במערכת נחלקים ארבע קבוצות: Developer, Scrum master, Project_ Manager, Project Owner

.

Scrum Master: מוודא שתהליך ה-Scrum מתבצע כנדרש מוביל את התהליך ,בעל הרשאות לכל הפונקציונליות של האפלקצייה.

Developer: צוות קטן המונה 5 עד 9 אנשים בעלי מיומנויות רב-תחומיות המבצעים בפועל את העבודה,בעל הרשאות ל תיעוד המשימות ,תיעוד הדרישות ,תיעוד Backlog,.

.

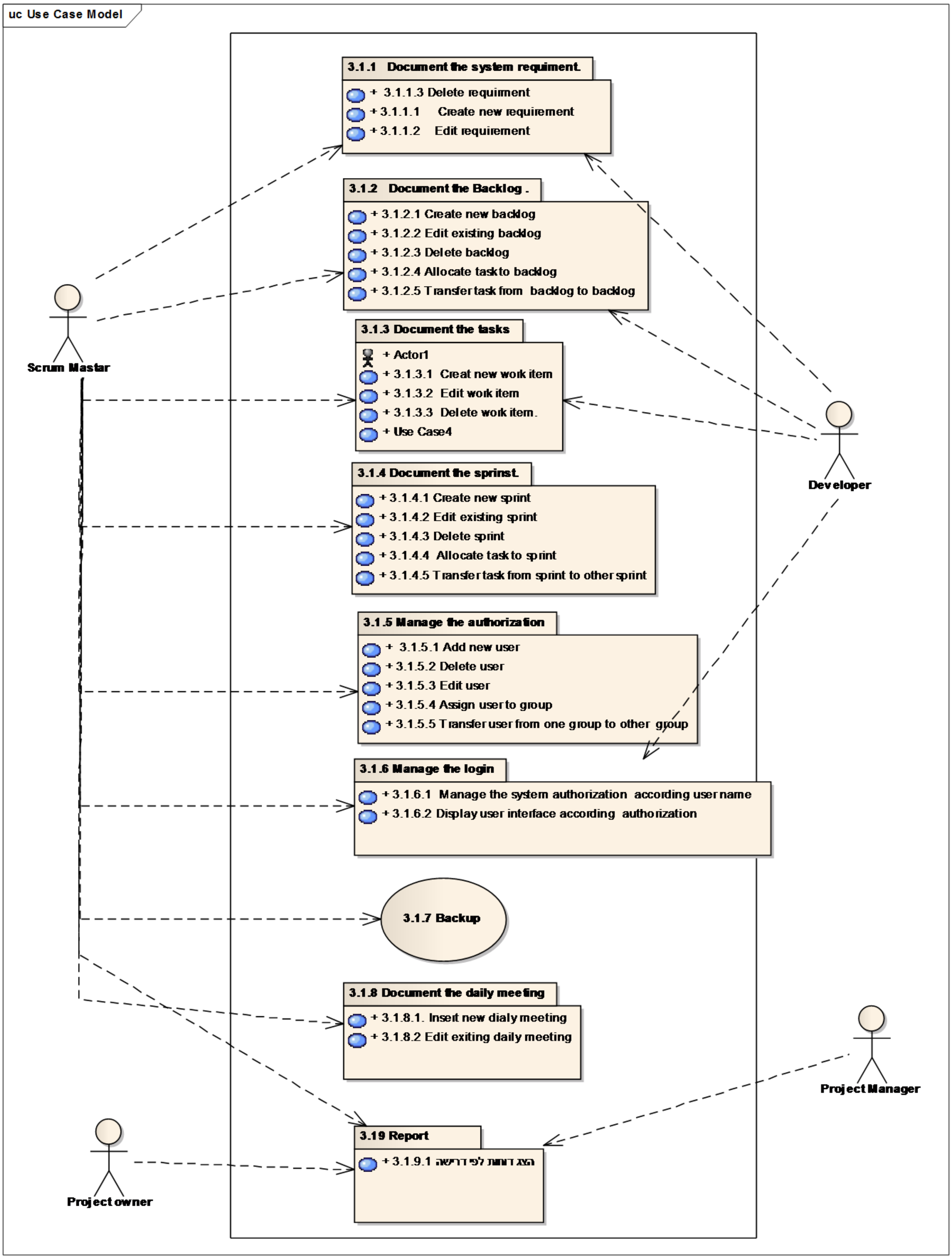
Project Manager : מנהל הפרוייקט אחראי על זמניי הפרוייקט ועל משאביי הפרוייקט בעל ההרשאות של קריאה לכל אפלקצייה .למשתמש אין זכות לבצע שינויים של עריכה אלא רק לראות וליצור דוחות .

Project Owner : מייצג את האינטרס של הלקוח כדי להבטיח שצוות ה-Scrum עושה את הדברים הנכונים מנקודת המבט העסקית.למשתמש אין זכות לבצע שינויים אלה רק לצפות בדוחות.

;

	Scrum Master	**Developer**	**Project owner**	**Project Manager**
תיעוד דרישות המערכת	v	x	x	x
תיעוד BackLog	v	v	x	x
תיעוד משימות	v	v	x	x
תיעוד Sprint	v	x	x	x
ניהול הרשאות	v	x	x	x
Dialy_meeting	v	x	x	x
הפקת דוחות	v	x	v	v

;


uc Use Case Model
3.1.1 Document the system requiment.
+ 3.1.1.3 Delete requirment
+ 3.1.1.1 Create new requirement
+ 3.1.1.2 Edit requirement
3.1.2 Document the Backlog .
+ 3.1.2.1 Create new backlog
+ 3.1.2.2 Edit existing backlog
+ 3.1.2.3 Delete backlog
+ 3.1.2.4 Allocate task to backlog
+ 3.1.2.5 Transfer task from backlog to backlog
3.1.3 Document the tasks
+ Actor1
+ 3.1.3.1 Creat new work item
+ 3.1.3.2 Edit work item
+ 3.1.3.3 Delete work item.
+ Use Case4
3.1.4 Document the sprinst.
+ 3.1.4.1 Create new sprint
+ 3.1.4.2 Edit existing sprint
+ 3.1.4.3 Delete sprint
+ 3.1.4.4 Allocate task to sprint
+ 3.1.4.5 Transfer task from sprint to other sprint
3.1.5 Manage the authorization
+ 3.1.5.1 Add new user
+ 3.1.5.2 Delete user
+ 3.1.5.3 Edit user
+ 3.1.5.4 Assign user to group
+ 3.1.5.5 Transfer user from one group to other group
3.1.6 Manage the login
+ 3.1.6.1 Manage the system authorization according user name
+ 3.1.6.2 Display user interface according authorization
3.1.7 Backup
3.1.8 Document the daily meeting
+ 3.1.8.1. Insert new dialy meeting
+ 3.1.8.2 Edit exiting daily meeting
3.19 Report
+ 3.1.9.1 הצג דוח לפי דרישה
Scrum Mastar
Developer
Project Manager
Project owner

;

3.2.4 אילוצים כללים

אילוצי נוספים שיש להתחשב בהם:

- כיוון שהמערכת מכילה נתונים חשובים יש לבצע גיבוי יומי.
- בסיס הנתונים עלול לתפוח למימדים גדולים ולכן יש לעבוד עם בסיס נתונים שיכול להכיל נתונים רבים תוך שמירה על אמינותו.
- מכיוון שכמה משתמשים יכולים הו זמנית להתחבר למערכת יש לדאוג לנעלות נתונים שנפתחו לעריכה אצל משתמש אחר(לא ימומש אופצייה עתידית).
- יש לאפשר לשמור מסמכים בבסיס נתונים (לא יוממש אופצייה עתידית)
- כיוון שהמערכת אמורה לפעול עוד שנים רבות יש לאפשר את אחזקתה ושידרוגה בקלות.

3.2.5 הנחות ותלויות

- מערכת ההפעלה .Windows XP/ VISTA/Windows 7.
- בסיס הנתונים SQL SERVER 2008.
- Microsoft Framework 4.0
- קיימת תקשורת בין כל העמדות השירות לבין השרת.
- יתבצע גיבוי יומי של בסיס נתונים ללא הפרעה למהלך העובדה בזמן הגיבוי (גיבוי חם).
- המערכת תתן בו זמנית תמיכה לכמה משתמשים הדורשים שרות כל שהוא .
- משתמש לא יוכל להיות שייך לשתי קבוצות בו זמנית.
- Scrum master יחיד(המערכת תאכוף אופציה זו)

3.3 מפרט הדרישות מהמערכת

3.3.1 דרישות פונקציונליות

הדרישות הפונקציונליות של המערכת הן:

3.1.1 תיעוד דרישות המערכת בפרוייקט.

3.1.2 תיעוד Backlog.

3.1.3 תיעוד משימות.

3.1.4 תיעוד Sprint.

;

3.1.5 ניהול הרשאות.

3.1.6 ניהול Login .

3.1.7 ביצוע גיבוי

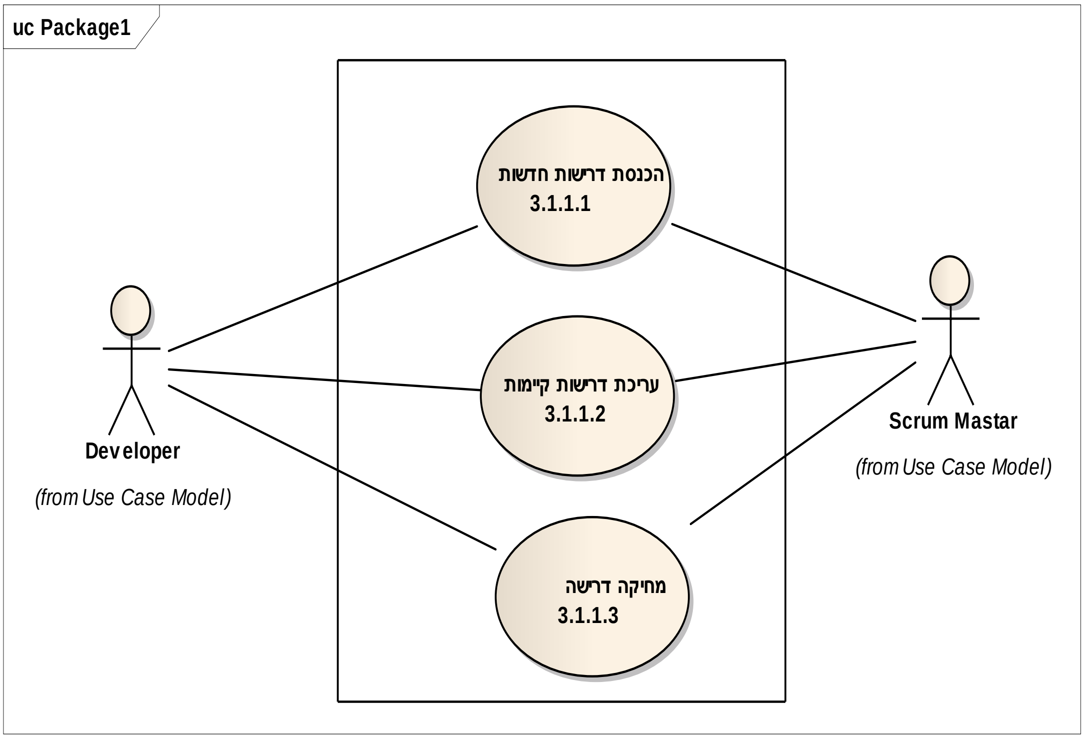
3.1.8 תיעוד Dialy Meeting .

3.1.9 הפקת דוחות .

3.3.1.1 תיעוד דרישות המערכת בפרוייקט

תיעוד הדרישות הלקוח ,בדרך כלל נלקחות ממסמך SRS או מ SSDD, הדרישות בעצם מתארות מה הפרוייקט צריך להכיל כאשר במסמך SSDD ההיבט יותר מערכתי ו SRS יותר תוכנתי.

בפרוייקט Agile נציג הלקוח Project owner יכול להוסיף או ולהוריד דרישות בין Sprint



3.3.1.1.1 הכנסת דרישות חדשות

מטרה:

לאחר שמהנדס המערכת כתב מסמך SSDD יש לתעד את הדרישות באפלקצייה.

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer

תנאי הצלחה:

הדרישות שעודכנו במערכת נשמרו בבסיס נתונים.

;

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן לשמור דרישות חדשות בבסיס הנתונים

<u>טריגר</u>:

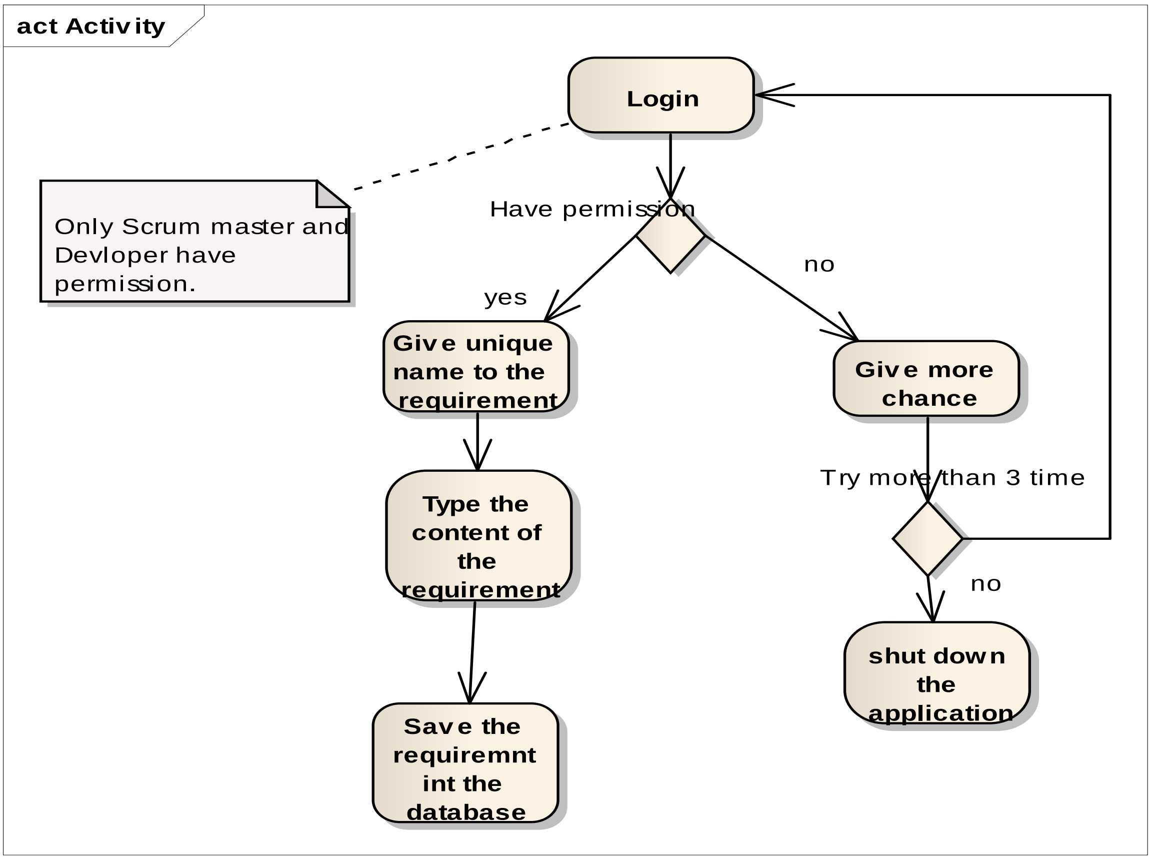
המשתמש מבקש מהמערכת להכניס דרישות חדשות.

<u>שחקנים</u>:

Scrum Master

Developer

<u>התרחיש</u>



3.3.1.1.2 עריכת דרישות קיימות

<u>מטרה</u>:

לאחר תיעוד דרישה במערכת המערכת צריכה לתת יכולת לערוך דרישה הקיימת במערכת.

<u>תנאי איתחול</u>:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

הדרישה קיימת בבסיס הנתונים .

;

<u>תנאי הצלחה</u>:

לאחר הערכה השינויים שבצעו נשמרו בבסיס נתונים.

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן לשמור דרישות בבסיס הנתונים

<u>טריגר</u>:

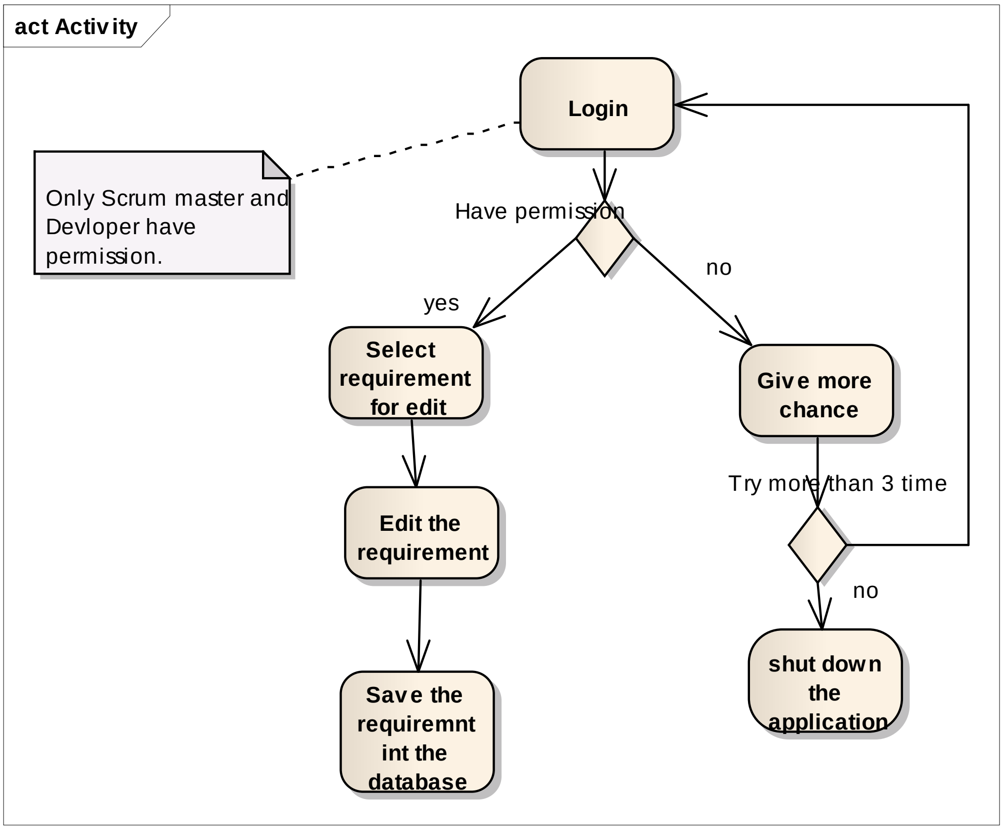
המשתמש מבקש לערוך דרישה קיימת בבסיס נתונים.

<u>שחקנים</u>:

Scrum Master

Developer

<u>התרחיש</u>



<u>3.3.1.1.3</u>   מחיקת דרישה

<u>מטרה</u>:

המערכת צריכה לתת אפשרות של מחיקה דרישה הקיימת במערכת.

<u>תנאי איתחול</u>:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

הדרישה קיימת בבסיס הנתונים .

<u>תנאי הצלחה</u>:

;

הדרישה נמחקה מהבסיס הנתונים.

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן למחוק דרישה מבסיס הנתונים

<u>טריגר</u>:

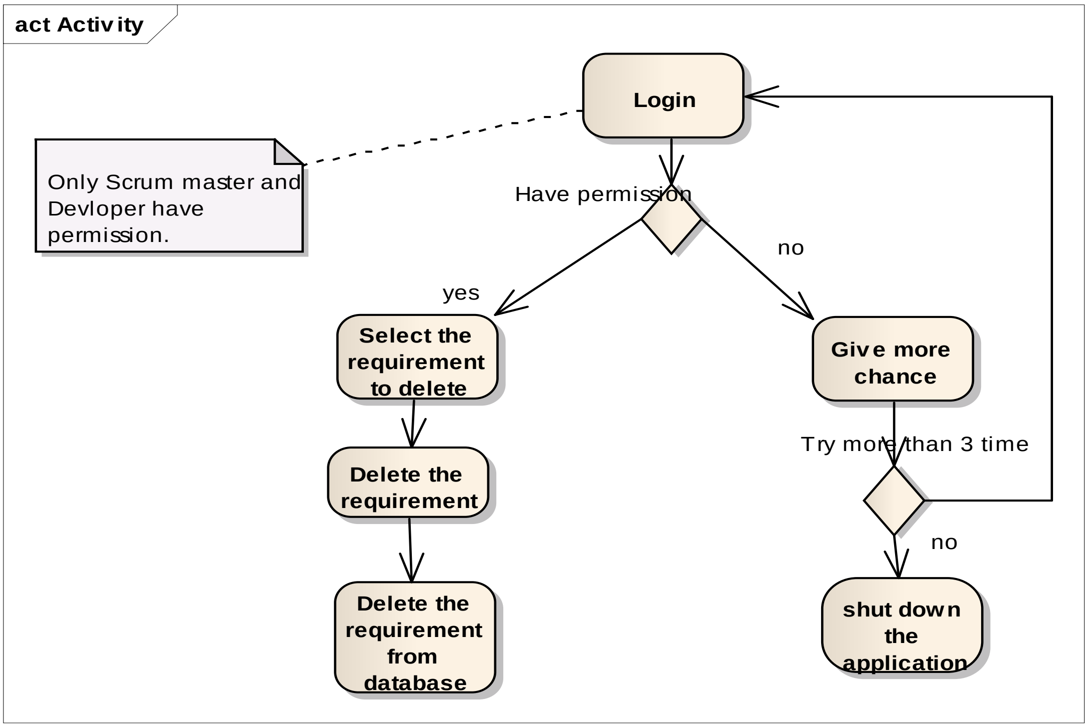
המשתמש מבקש למחוק דרישה הקיימת בבסיס נתונים.

<u>שחקנים</u>:

Scrum Master

Developer

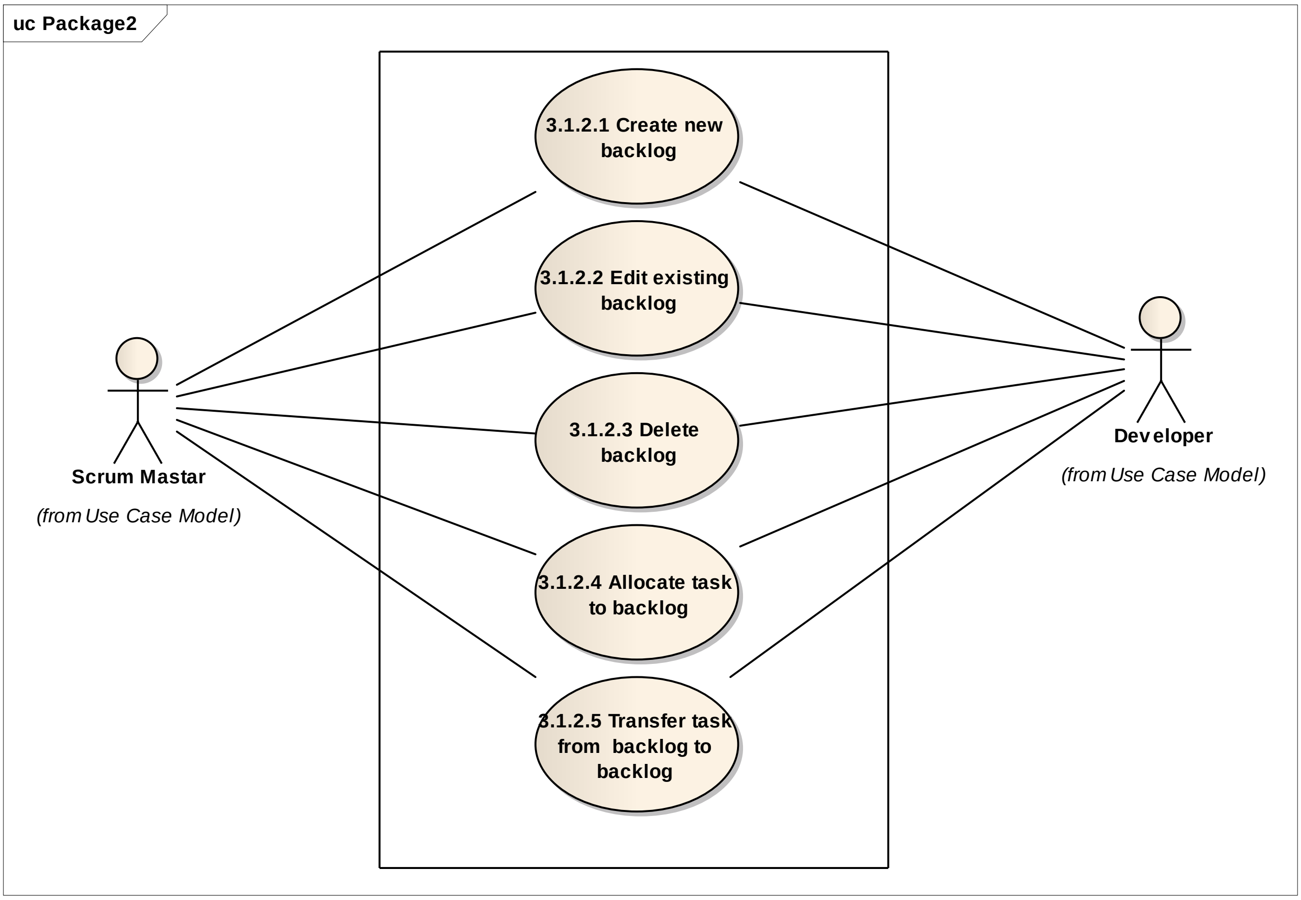
<u>התרחיש</u>



;

3.3.1.2 תיעוד Backlog

תחזוקה של רשימת פריטי העבודה לביצוע, מסודרים לפי קדימויות, המכונה עתודת המוצר(Product Backlog). למעשה Backlog מכיל את המשימות שהפרוייקט צריך לבצע על פי קידמיות .



<u>3.3.1.2.1</u> יצרת Backlog חדש

;

<u>מטרה:</u>

המערכת צריכה לתת אפשרות לצור Backlog חדש אשר יכיל את כל משימות הפרוייקט.

<u>תנאי איתחול:</u>

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

<u>תנאי הצלחה:</u>

ה Backlog נשמר בבסיס הנתונים.

<u>תנאי כשלון:</u>

תקלה בבסיס הנתונים – לא ניתן לשמור Backlog

המשתמש נתן שם ל Backlog שם של Backlog שכבר קיים במערכת לכן המערכת חסמה את המשתמש מלשמור את Backlog .

<u>טריגר:</u>

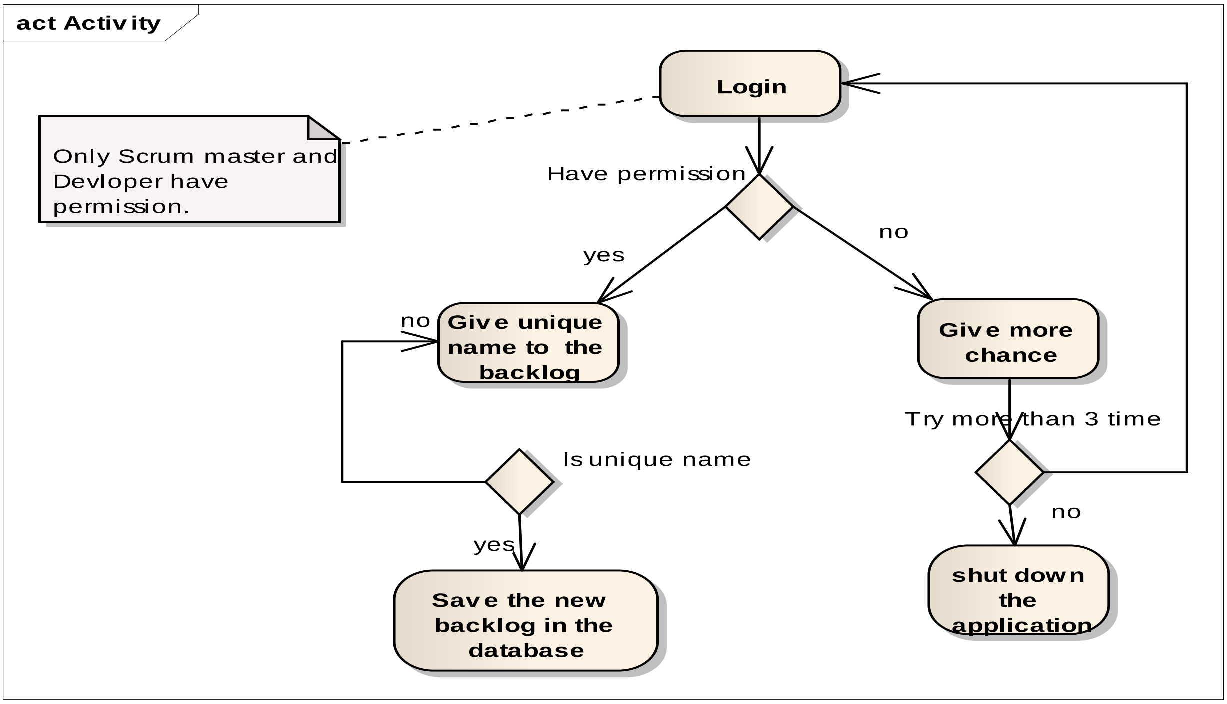
המשתמש מבקש ליצור Backlog חדש.

<u>שחקנים:</u>

Scrum Master

Developer

<u>התרחיש</u>



;

3.3.1.2.2 עריכת Backlog קיים

מטרה:

המערכת צריכה לתת אפשרות לערוך Backlog הקיים בבסיס הנתונים

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

Backlog קיים בבסיס הנתונים.

תנאי הצלחה:

השינויים Backlog נשמרו בבסיס הנתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן לשמור שינויים ב Backlog

המשתמש נתן שם ל Backlog שם של Backlog שכבר קיים במערכת לכן המערכת חסמה את המשתמש מלשמור את Backlog .

טריגר:

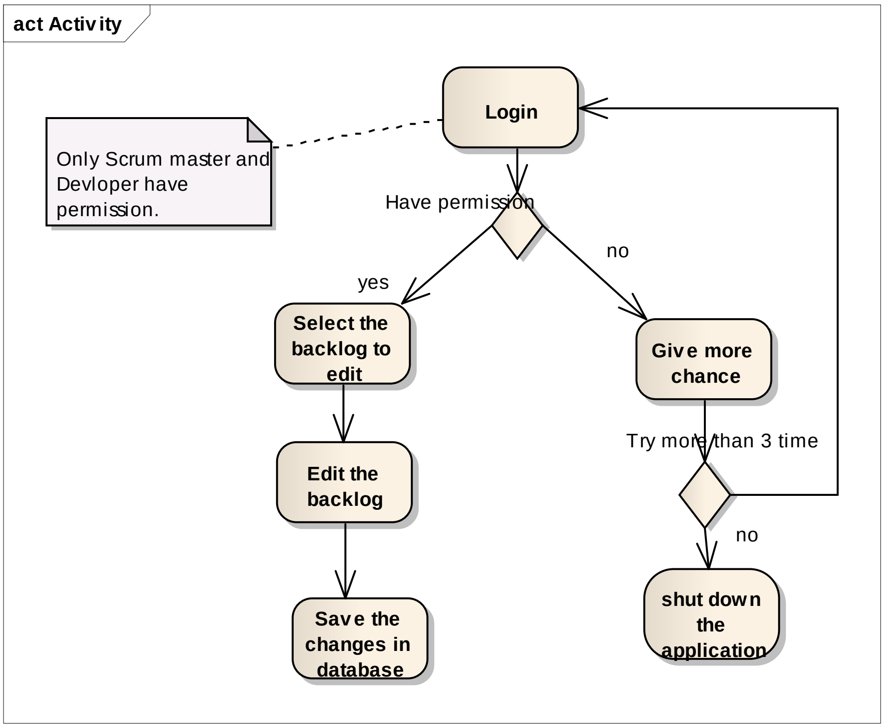
המשתמש מבקש לערוך Backlog הקיים במערכת.

שחקנים:

Scrum Master

Developer

התרחיש



;

3.3.1.2.3 מחיקת Backlog קיים

מטרה:

המערכת צריכה לתת אפשרות למחוק Backlog הקיים בבסיס הנתונים

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

Backlog קיים בבסיס הנתונים.

ל Backlog לא מכיל משימות (יש למחוק את הקישור של המשימות המשיוכת לו)

תנאי הצלחה:

השינויים Backlog נשמרו בבסיס הנתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן למחוק Backlog

המשתמש מנסה למחוק Backlog שמשימות משוייכת אליו לכן המערכת תחסם אותו.

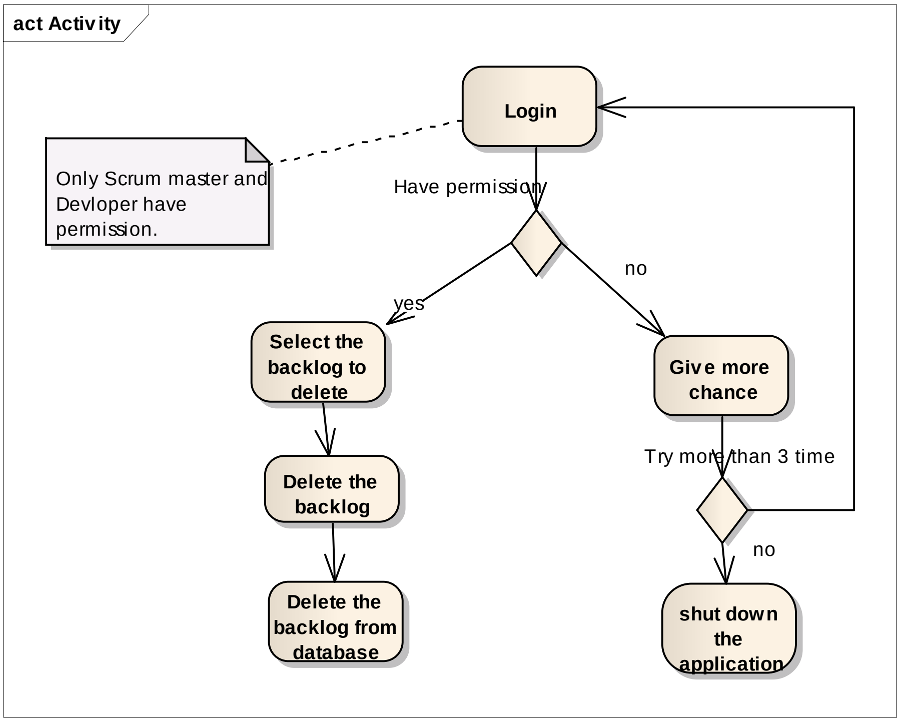
טריגר:

המשתמש מבקש למחוק Backlog הקיים במערכת.

שחקנים:

Scrum Master

התרחיש



;

3.3.1.2.4 הקצאת משימה ל Backlog

מטרה:

המערכת צריכה לתת אפשרות לשייך משימות ל Backlog

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

Backlog קיים בבסיס הנתונים.

ל Backlog לא מכיל משימות (יש למחוק את הקישור של המשימות המשיוכת לו)

תנאי הצלחה:

השינויים Backlog נשמרו בבסיס הנתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן למחוק Backlog

המשתמש מנסה למחוק Backlog שמשימות משוייכת אליו לכן המערכת תחסם אותו.

טריגר:

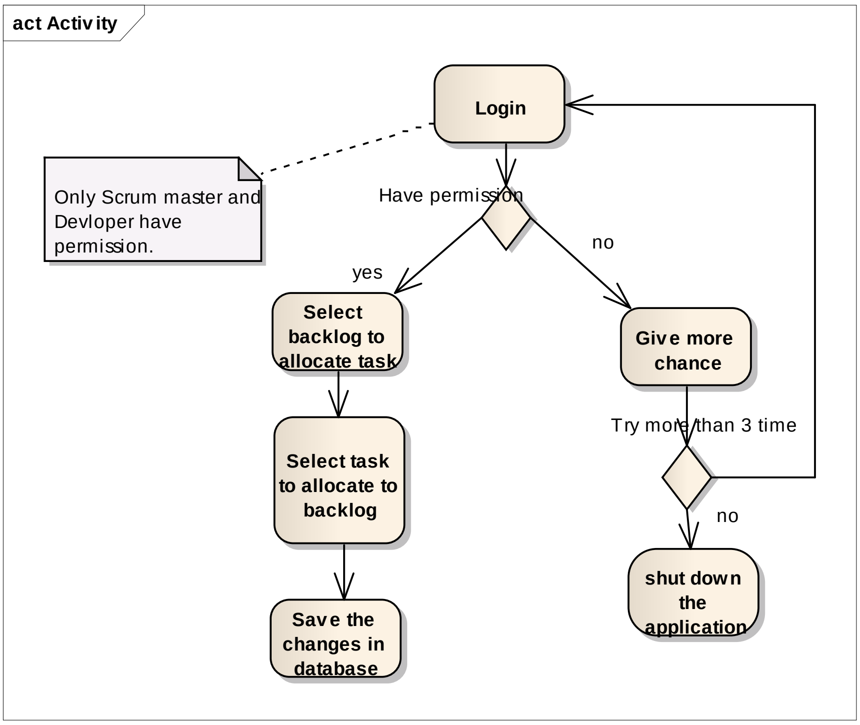
המשתמש מבקש למחוק Backlog הקיים במערכת.

שחקנים:

Scrum Master

Developer

התרחיש



;

3.3.1.2.5 העברת משימה מ Backlog ל Backlog אחר

מטרה:

המערכת צריכה לתת אפשרות להעביר משימות מ Backlog ל Backlog אחר.

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

קיימים שני Backlog בבבסיס הנתונים.

תנאי הצלחה:

העברת המשימה נשמרה בבסיס הנתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן להעביר את המשימה מ Backlog ל Backlog אחר.

טריגר:

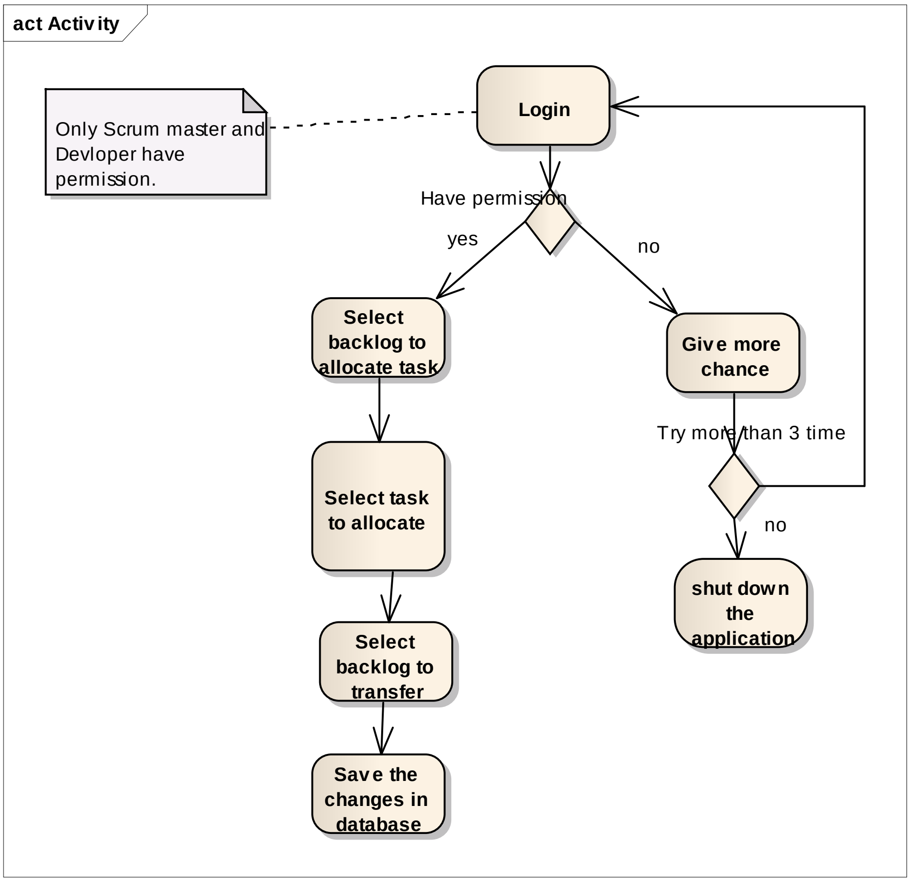
המשתמש מבקש להעביר משימה מ Backlog ל Backlog אחר .

שחקנים:

Scrum Master

Developer

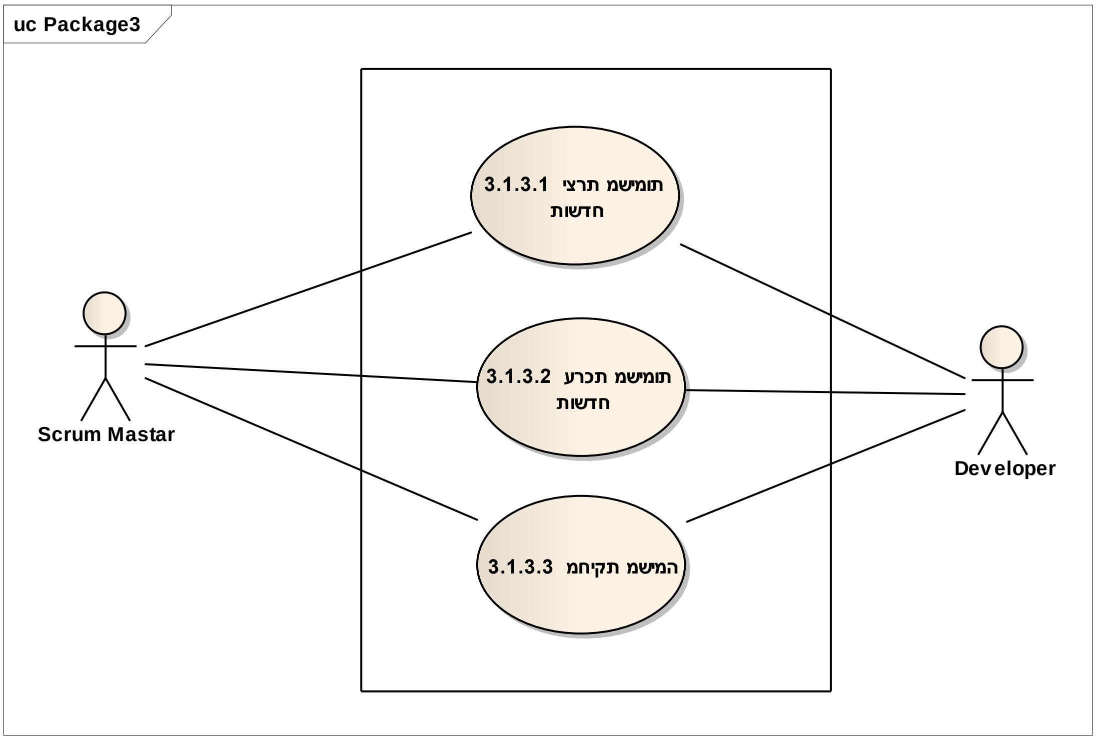
התרחיש



;

3.3.1.3 תיעוד משימות

משימות הפרוייקט בעצם נגזרות מדרישת הפרוייקט. כל דרישה יכולה להכיל פעולה אחת או יותר .כלומר המשימות צריכות לכסות את הדרישה .



<u>3.3.1.3.1</u> יצרת משימה חדש

<u>מטרה</u>:

המערכת צריכה לתת אפשרות ליצור משימה חדשה .

<u>תנאי איתחול</u>:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

<u>תנאי הצלחה</u>:

המשימה החדשה נשמרה בבסיס הנתונים.

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן לשמור משימה חדשה

המשתמש נתן שם ל משימה שם של משימה שכבר קיימת במערכת לכן המערכת חסמה את המשתמש מלשמור את המשימה .

<u>טריגר</u>:

המשתמש מבקש ליצור משימה חדשה.

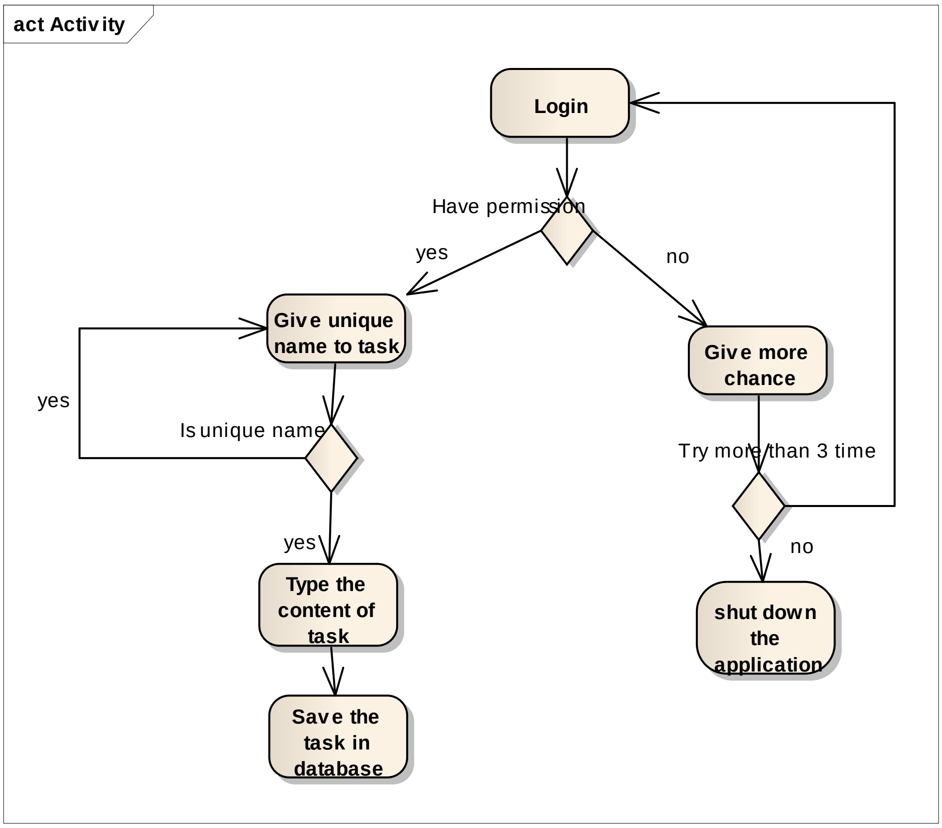
<u>שחקנים</u>:

Scrum Master

;

Developer

התרחיש



;

<u>3.3.1.3.2</u> עריכה משימה קיימת

<u>מטרה</u>:

המערכת צריכה לתת אפשרות לערוך משימה שקיימת בבסיס נתונים .

<u>תנאי איתחול</u>:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

<u>תנאי הצלחה</u>:

השינוי שבוצע במשימה נשמר בבסיס הנתונים.

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן לשמור שינויים במשימה

המשתמש נתן שם ל משימה שם של משימה שכבר קיימת במערכת לכן המערכת חסמה

את המשתמש מלשמור את המשימה בשם הנתון .

<u>טריגר</u>:

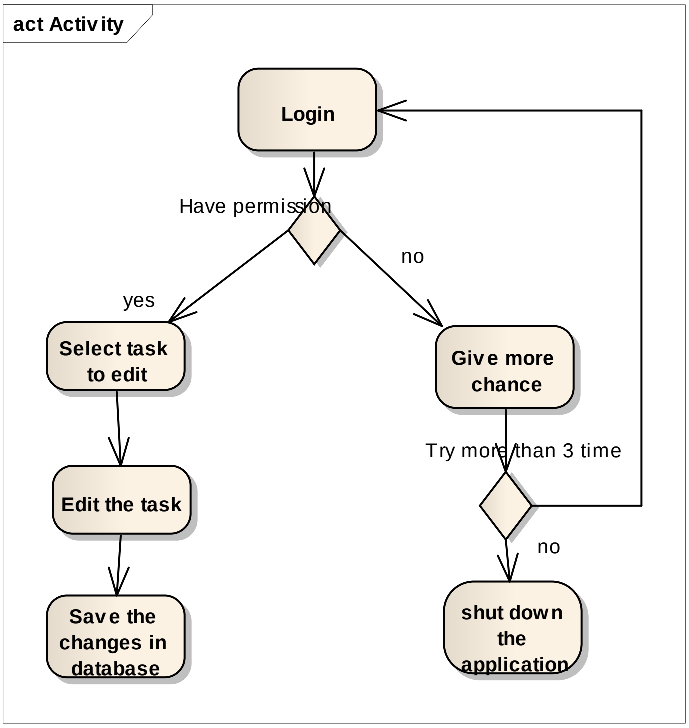
המשתמש מבקש לערוך משימה קיימת.

<u>שחקנים</u>:

Scrum Master

Developer

<u>התרחיש</u>



;

<u>3.3.1.3.3</u> מחיקת משימה

<u>מטרה</u>:

המערכת צריכה לתת אפשרות למחוק משימה שקיימת בבסיס נתונים .

<u>תנאי איתחול</u>:

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

<u>תנאי הצלחה</u>:

המשימה נמחקה מהבסיס הנתונים.

<u>תנאי כשלון</u>:

תקלה בבסיס הנתונים – המשימה לא נמחקה מהבסיס נתונים.

<u>טריגר</u>:

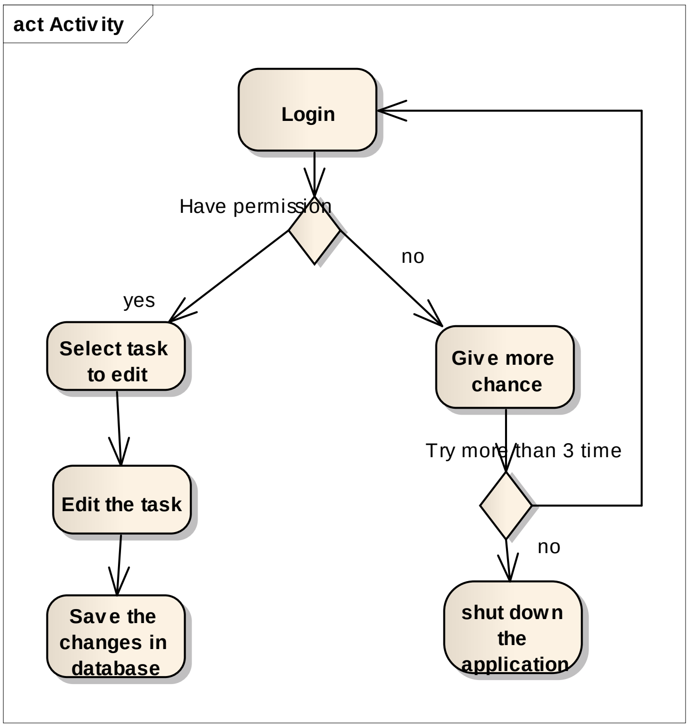
המשתמש מבקש למחוק משימה הקיימת בבסיס נתונים.

<u>שחקנים</u>:

Scrum Master

Developer

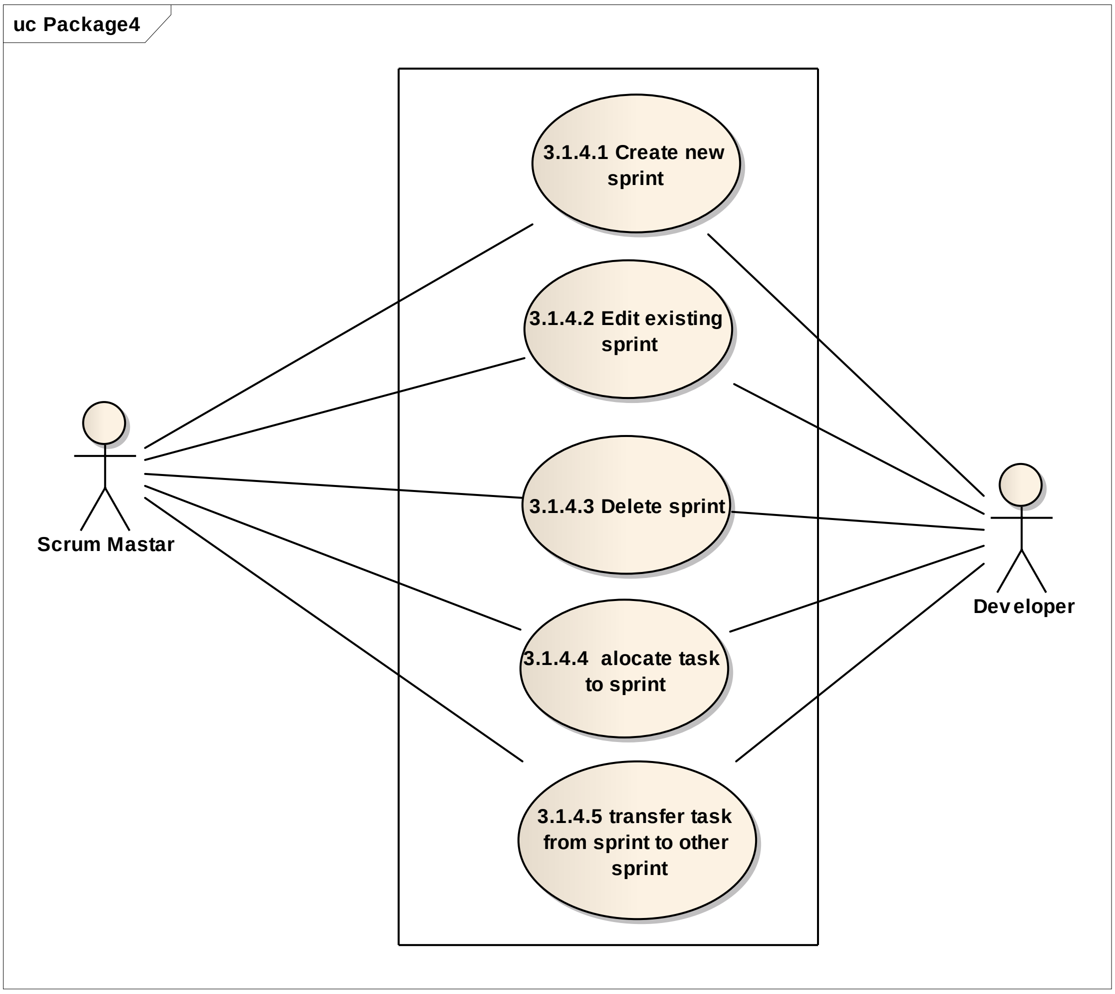
<u>התרחיש</u>



;

3.3.1.4 תיעוד Sprint

הפיתוח יחולק לאטרציות (Sprint) וכל אטרציה תכיל את הדרישות שהיא מכילה וכל דרישה תכיל את המשימה שאמורה לכסות את הדרישה



<u>3.3.1.4.1</u> יצרת Sprint חדש

<u>מטרה:</u>

המערכת צריכה לתת אפשרות ליצור Sprint חדש .

<u>תנאי איתחול:</u>

למשתמש יש הרשאות מתאימות Scrum Master /Developer.

<u>תנאי הצלחה:</u>

המשימה החדשה נשמרה בבסיס הנתונים.

<u>תנאי כשלון:</u>

;

תקלה בסיס הנתונים – לא ניתן לשמור Sprint חדש

המשתמש נתן שם ל Sprint שם של Sprint שכבר קיים במערכת לכן המערכת חסמה את המשתמש מלשמור את Sprint .

<u>טריגר</u>:

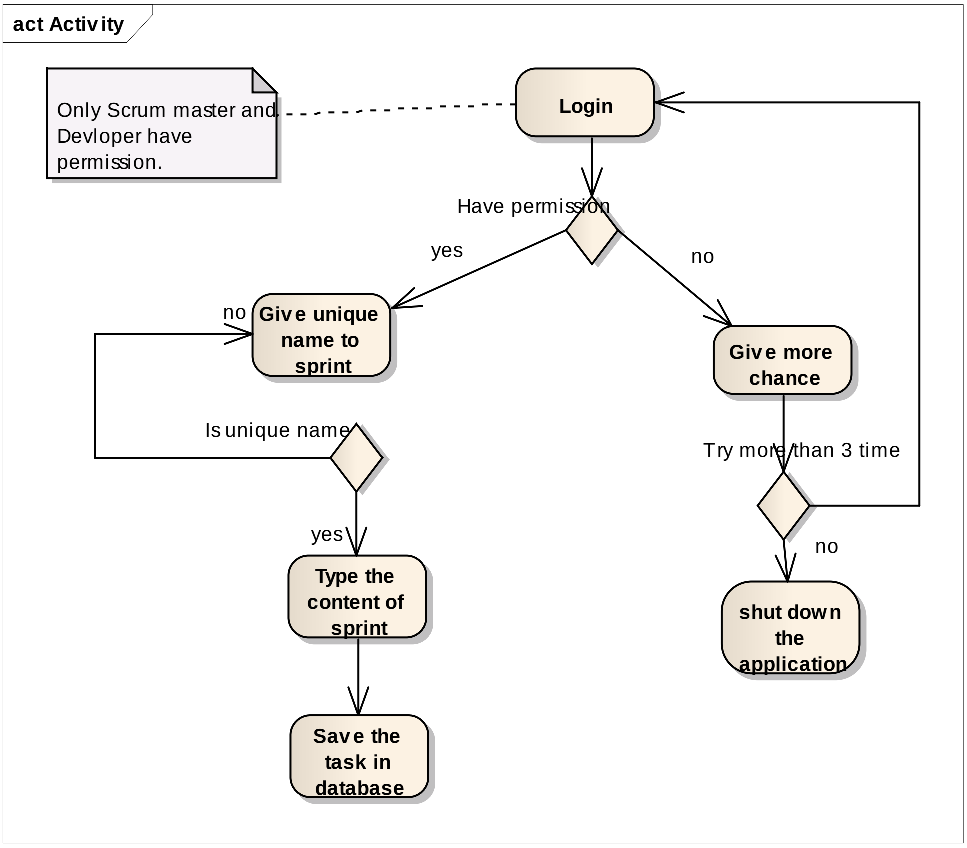
המשתמש מבקש ליצור Sprint חדש.

<u>שחקנים</u>:

Scrum Master

Developer

<u>התרחיש</u>



;

3.3.1.4.2 עריכת Sprint קיים

מטרה:

המערכת צריכה לתת אפשרות לערוך Sprint הקיים במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer

קיים כבר Sprintבמערכת .

תנאי הצלחה:

השינויים שבוצעו ב Sprint נשמרו בבסיס נתונים.

תנאי כשלון:

תקלה בבסיס הנתונים – לא ניתן לשמור את שינויי ה Sprint בבסיס הנתונים

המשתמש נתן שם ל Sprint שכבר קיים במערכת לכן המערכת חסמה את המשתמש מלשמור את Sprint .

טריגר:

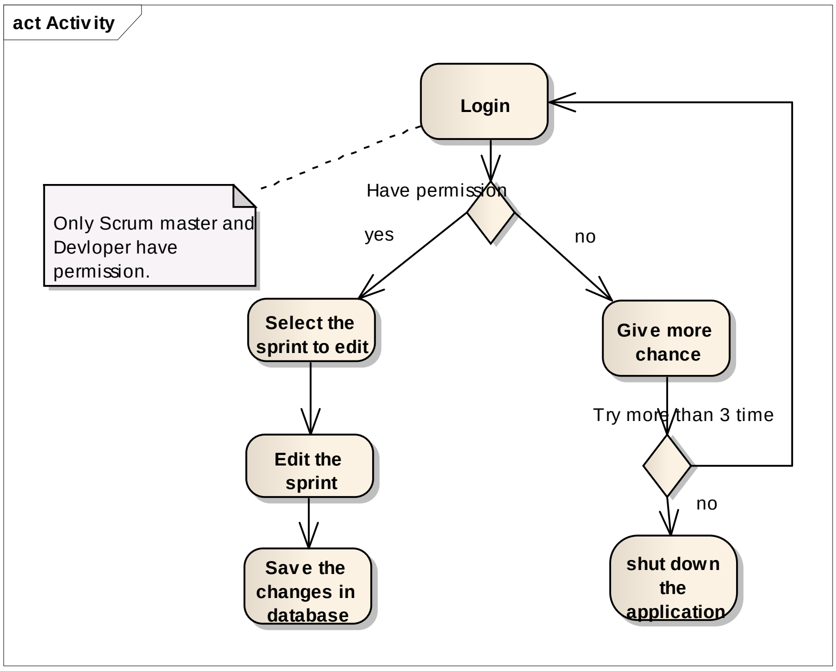
המשתמש מבקש לערוך Sprint שכבר קיים בבסיס הנתונים.

שחקנים:

Scrum Master

Developer

התרחיש



;

3.3.1.4.3 מחיקת Sprint קיים

מטרה:

המערכת צריכה לתת אפשרות למחוק Sprint הקיים במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer

קיים כבר Sprintבמערכת .

תנאי הצלחה:

השינויים שבוצעו ב Sprint נשמרו בבסיס נתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן למחוק את ה Sprint מהבסיס הנתונים

טריגר:

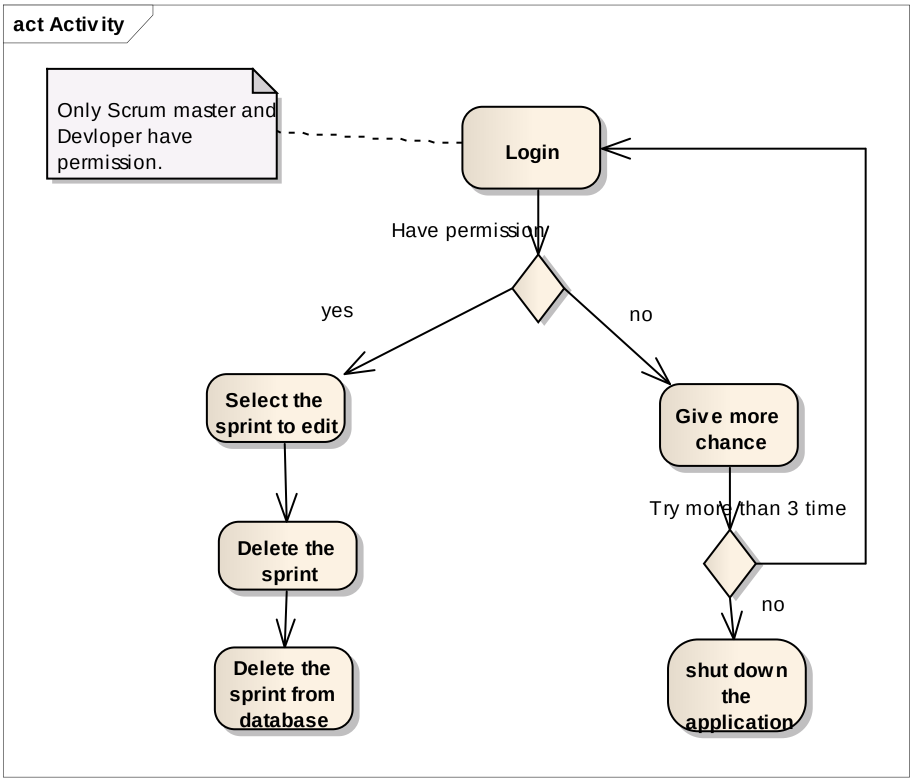
המשתמש מבקש למחוק Sprint הקיים בבסיס הנתונים.

שחקנים:

Scrum Master

Developer

התרחיש



;

3.3.1.4.4 הקצאת Sprint לבסיס נתונים

מטרה:

המערכת צריכה לתת אפשרות להקצאות משימות ל Sprint .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer

קיים כבר Sprintבמערכת .

קיים משימה במערכת.

תנאי הצלחה:

הקצאת משימה ל Sprint נשמרה בבסיס נתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן להקצאות את ה Sprint בבסיס הנתונים

טריגר:

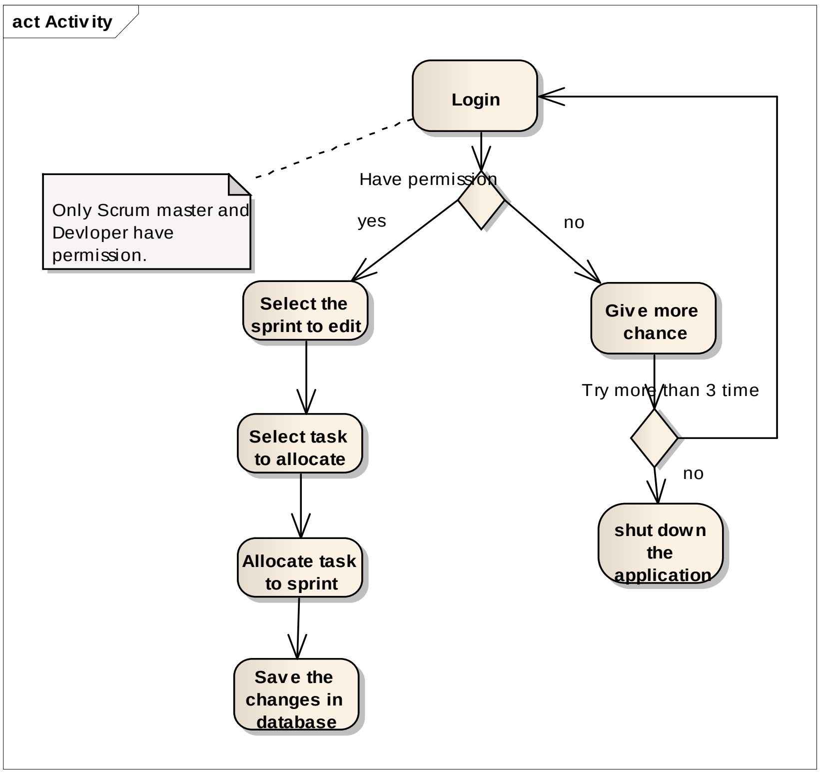
המשתמש מבקש להקצאות משימה לSprint הקיים בבסיס הנתונים.

שחקנים:

Scrum Master

Developer

התרחיש



;

3.3.1.4.5 העברת משימה מ Sprint לאחר

מטרה:

המערכת צריכה לתת אפשרות להעביר משימה Sprint אחד לאחר .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master /Developer

קיים כבר Sprintבמערכת .

קיים משימה במערכת.

תנאי הצלחה:

העברת המשימה מ Sprint אחד לשני נשמרה בבסיס הנתונים.

תנאי כשלון:

תקלה בסיס הנתונים – לא נשמרה ההעברה של המשימה מ Sprint אחד לשני.

טריגר:

המשתמש מבקש להעביר משימה מSprint אחד לשני.

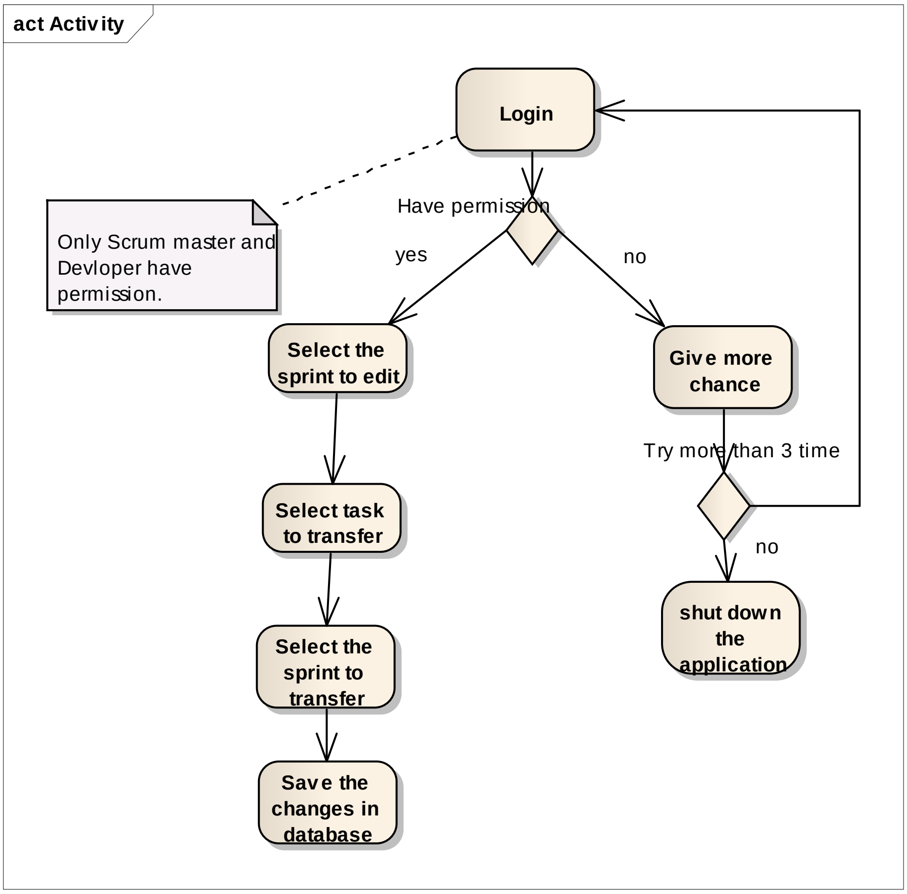
שחקנים:

Scrum Master

Developer

התרחיש

;

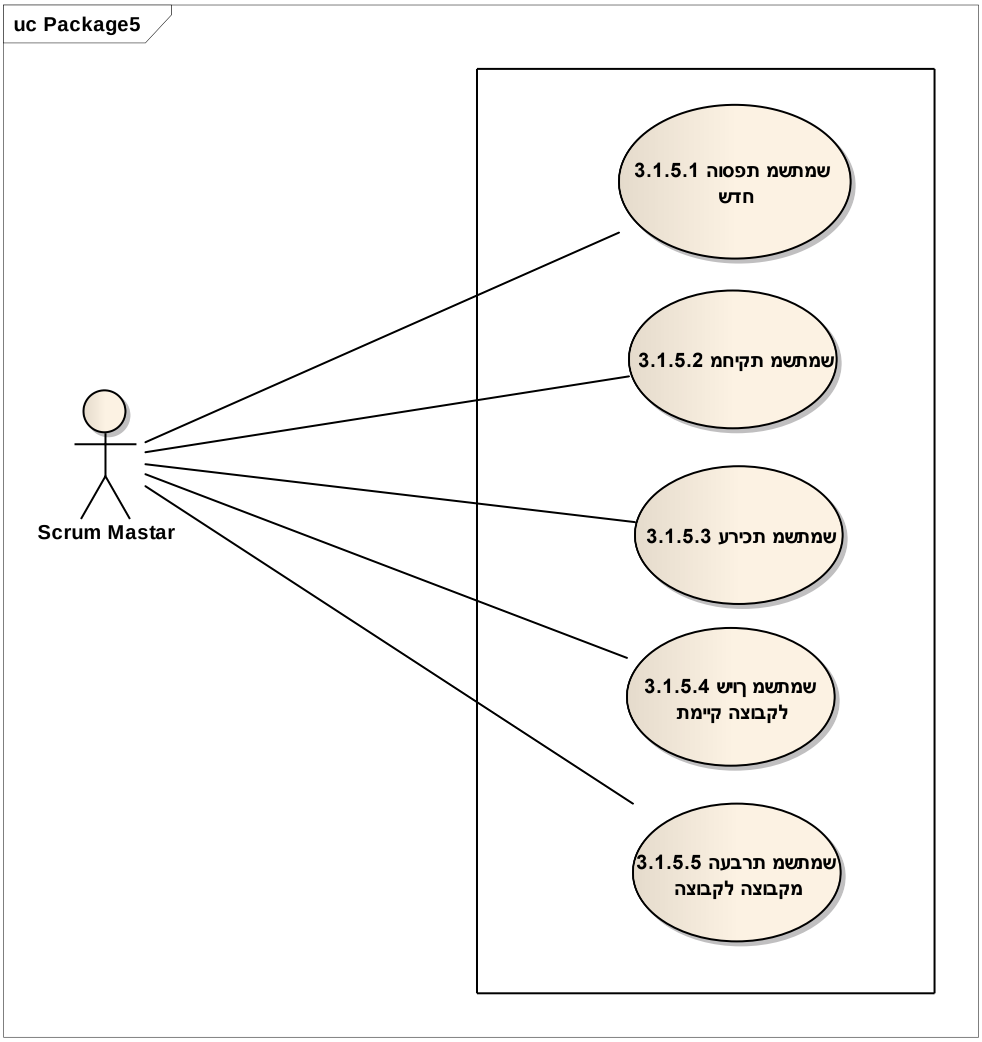

act Activity
Login
Only Scrum master and Devloper have permission.
Have permission
yes
no
Select the sprint to edit
Give more chance
Try more than 3 time
Select task to transfer
no
shut down the application
Select the sprint to transfer
Save the changes in database

;

3.3.1.5 ניהולהרשאות

ניהול ההרשאות בו ניתן לקבוע לאיזה קבוצה המשתמש שייך כאשר הקבוצות יודועות מראש

Scrum master ,Product owner ,Product manager ,Developer,



<u>3.3.1.5.1</u> הוספת משתמש חדש

<u>מטרה:</u>

המערכת צריכה לתת אפשרות להוסיף משתמשים חדשים למערכת .

<u>תנאי איתחול:</u>

למשתמש יש הרשאות מתאימות /Scrum Master.

<u>תנאי הצלחה:</u>

פרטי המשתמש החדש נשמר בבסיס הנתונים

;

<u>תנאי כשלון</u>:

תקלה בסיס הנתונים – לא ניתן לשמור משתמש חדש בבסיס הנתונים

ניסיון להוסיף שם של משתמש שכבר קיים במערכת לכן המערכת תחסום אפשרות זו.

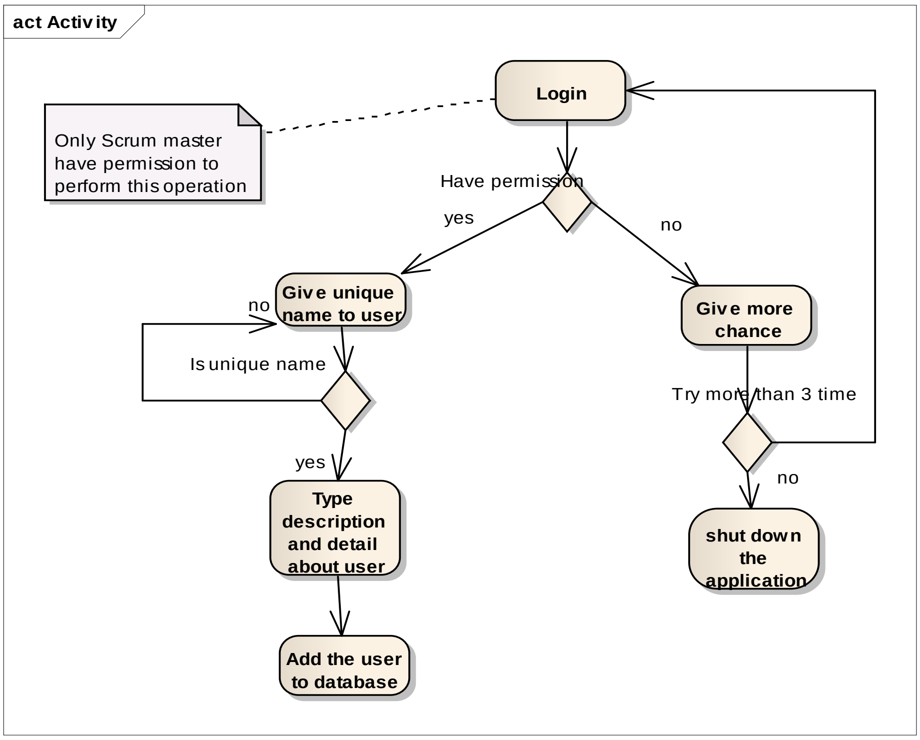
<u>טריגר</u>:

המשתמש מבקש להוסיף משתמש חדש לבסיס הנתונים.

<u>שחקנים</u>:

Scrum Master

<u>התרחיש</u>



;

3.3.1.5.2 מחיקת משתמש קיים

מטרה:

המערכת צריכה לתת אפשרות למחוק משתמש הקיים במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master.

תנאי הצלחה:

המשתמש נמחק מהבסיס הנתונים

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן למחוק משתמש מהבסיס הנתונים.

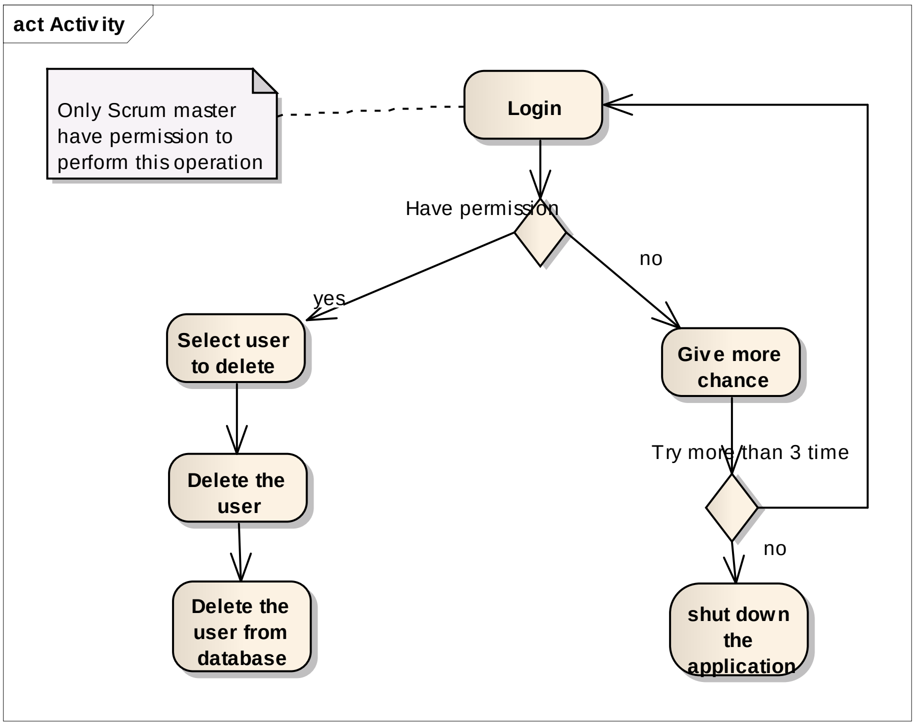
טריגר:

המשתמש מבקש למחוק משתמש הקיים במערכת.

שחקנים:

Scrum Master

התרחיש



;

3.3.1.5.3 עריכת משתמש קיים

מטרה:

המערכת צריכה לתת אפשרות לערוך משתמש הקיים במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master.

תנאי הצלחה:

השינויים נשמרו מהבסיס הנתונים

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן לערוך את פרטי המשתמש מהבסיס הנתונים.

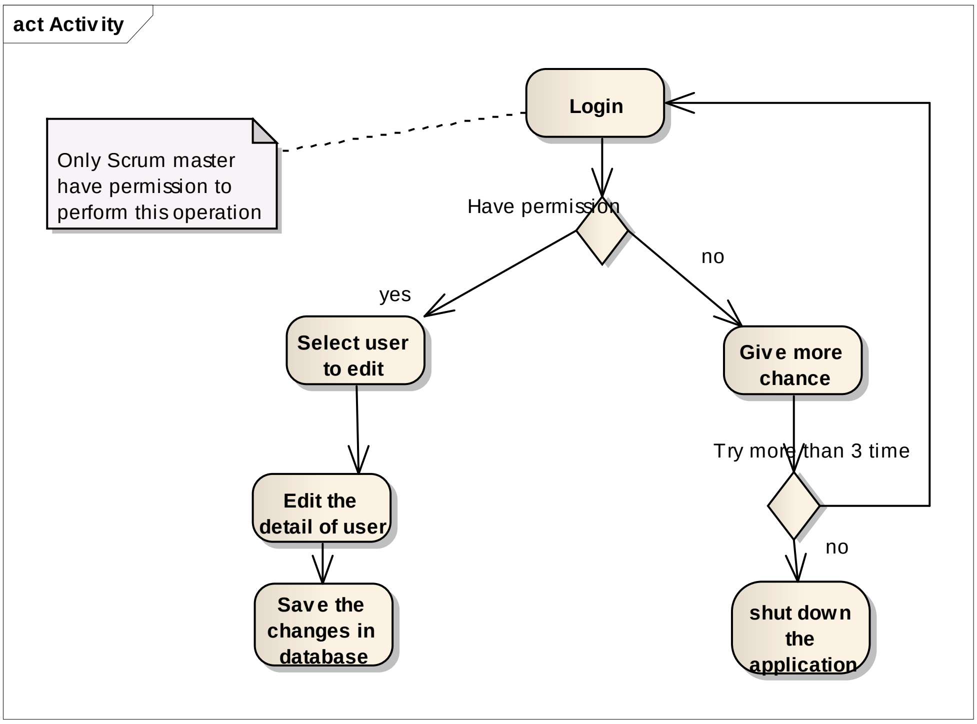
טריגר:

המשתמש מבקש לערוך את פרטי המשתמש הקיים במערכת.

שחקנים:

Scrum Master

התרחיש



;

3.3.1.5.4 שיוך משתמש לקבוצה

מטרה:

המערכת צריכה לתת אפשרות לשייך משתמש לקבוצה מסיומת אחת במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master.

תנאי הצלחה:

שיוך המשתמש לקבוצה הצליחה והשינויים נשמרו מהבסיס הנתונים

תנאי כשלון:

תקלה בסיס הנתונים – לא ניתן לערוך את פרטי המשתמש מהבסיס הנתונים.

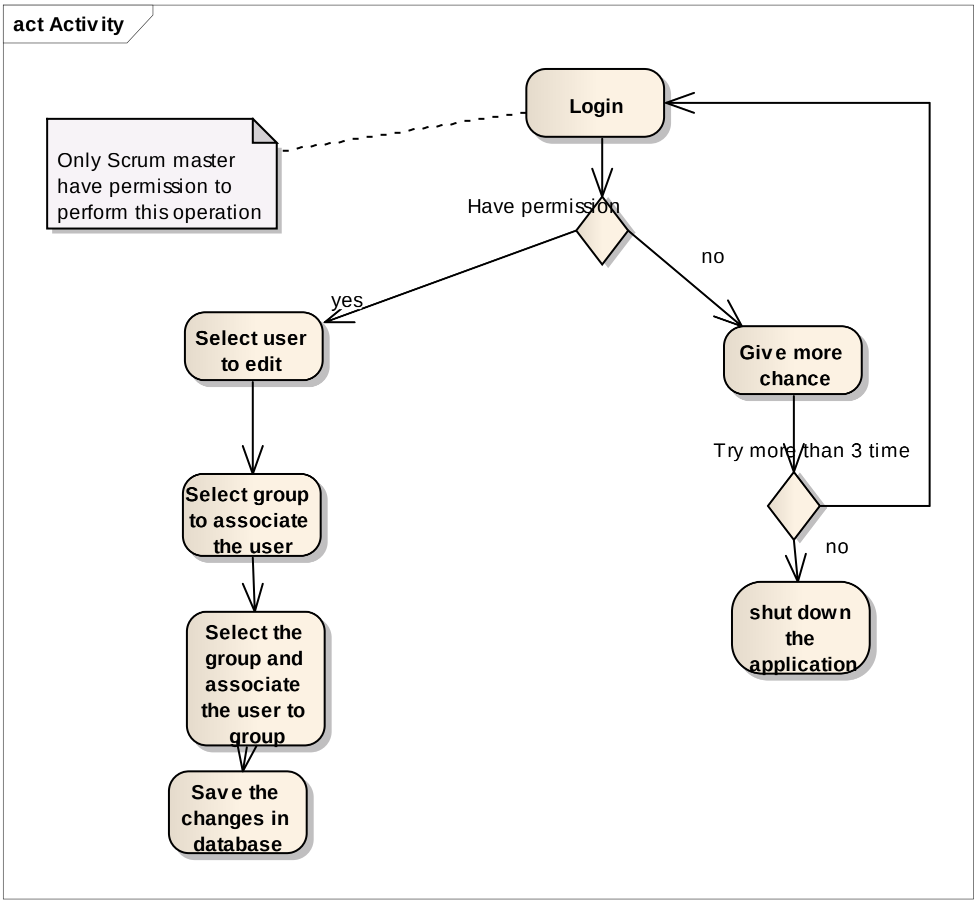
טריגר:

Scrum Master מבקש לשייך משתמש לקבוצה .

שחקנים:

Scrum Master

התרחיש



;

3.3.1.5.5 שיוך משתמש לקבוצה

מטרה:

המערכת צריכה לתת אפשרות להעבירמשתמש מקבוצה לקבוצה אחר במערכת .

תנאי איתחול:

למשתמש יש הרשאות מתאימות Scrum Master.

תנאי הצלחה:

העברת המשתמש לקבוצה נשמרה בבסיס הנתונים.

תנאי כשלון:

תקלה בבסיס הנתונים – לא ניתן להעביר משתמש מקבוצה לקבוצה ולכן השינוי לא נשמר בבסיס הנתונים.

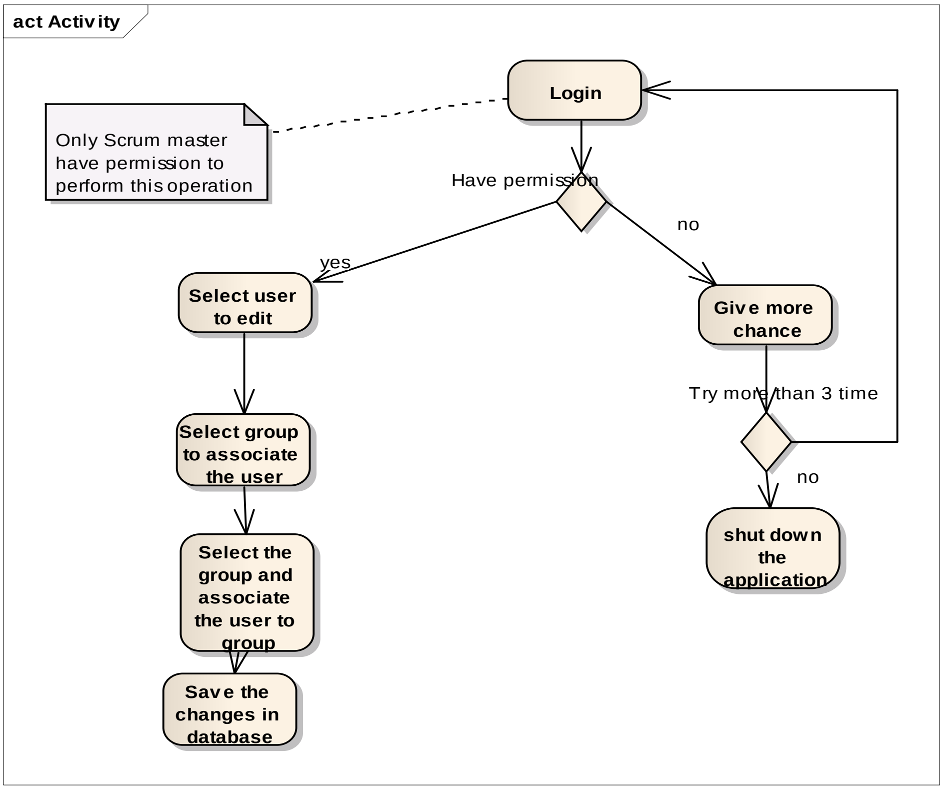
טריגר:

Scrum Master מבקש להעביר משתמש מקבוצה לקבוצה .

שחקנים:

Scrum Master

התרחיש



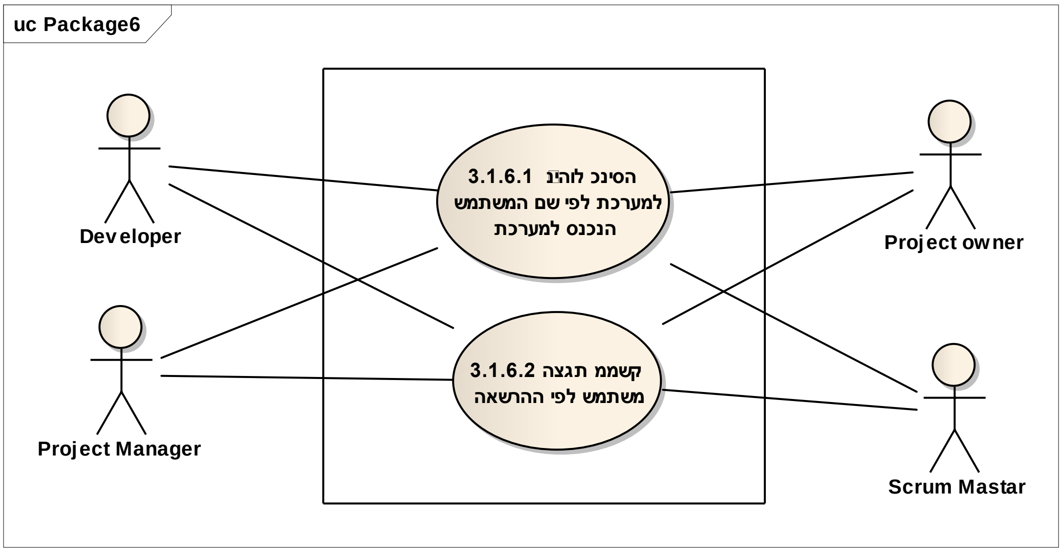
;

3.3.1.6 ניהול, Login

בתהליך ה Login מתבצע Authorization ו Authentication

כלומר מתבצע אימות אם בכלל למשתמש יש הרשאות, ולאחר מכן מתבצע תהליך לבדיקה אילו

פעולות המשתמש יכול לבצע.



<u>3.3.1.6.1</u> ניהול כניסה למערכת לפי שם המשתמש הנכנס למערכת

<u>מטרה</u>:

המערכת צריכה לבצע אימות בכניסה למערכת,ולחסם יכולות שהמשתמש אין לו הרשאה .

<u>תנאי איתחול</u>:

המשתמש מנסה להיכנס למערכת.

<u>תנאי הצלחה</u>:

המשתמש נכנס למערכת וחלק מיכולות חסומות לפי ההרשאה שניתנה לו .

המשתמש נחסם להיכנס למערכת בשל כך שאין לו הרשאות כניסה למערכת.

<u>תנאי כשלון</u>:

המערכת איפשרה להיכנס למשתמש למרות שאין למשתמש הרשאת כניסה.

המשתמש נכנס למערכת ומבצע פעולות שאין לו הרשאה .

<u>טריגר</u>:

המשתמש מבקש להיכנס למערכת.

;

<u>שחקנים</u>:

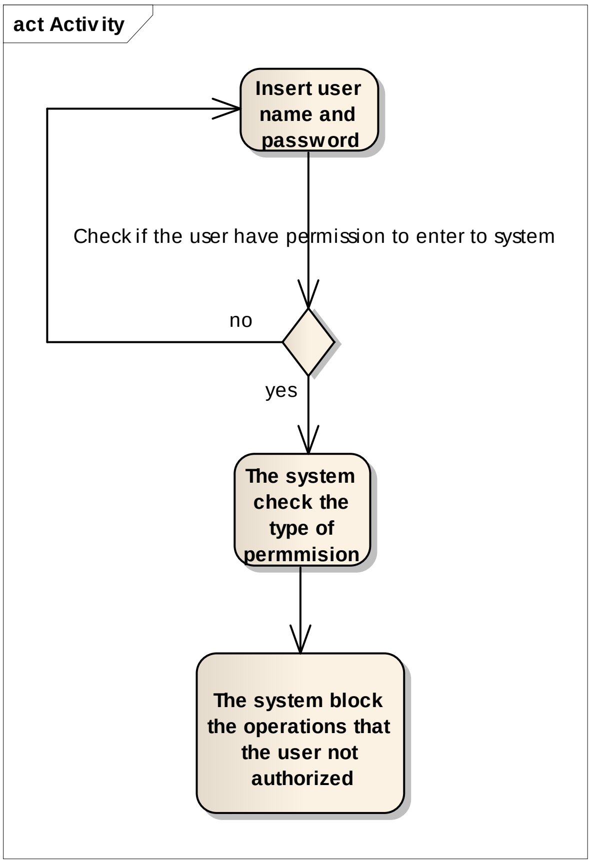
Project Manager

Project Owner

Developer

Scrum master

<u>התרחיש</u>



;

3.3.1.6.2 הצגת ממשק משתמש לפי ההרשאה

מטרה:

המערכת צריכה לבצע אימות בכניסה למערכת,ולחסם תצוגה של חלק מיכולות לפי הרשאת המשתמש .

תנאי איתחול:

המשתמש נכנס למערכת.

תנאי הצלחה:

המשתמש נכנס למערכת וחלק מיכולות לא מוצוגות על לפי ההרשאה .

תנאי כשלון:

המערכת איפשרה למשתמש להיכנס למערכת למרות שאין למשתמש הרשאת כניסה ומציגה יכולות שאין למתשתמש הרשאה אליהם.

טריגר:

המשתמש מבקש להיכנס למערכת.

שחקנים:

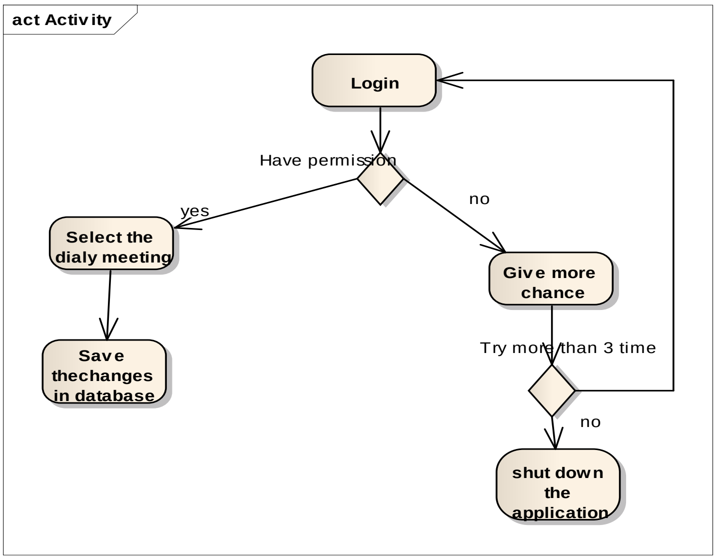
Project Manager

Project Owner

Developer

Scrum master

התרחיש



;

3.3.1.7 ביצוע גיבוי לבסיסי הנתונים

פונקציה זו תופעל על כל בסיסי הנתונים והרשימות הקיימים במערכת.

מטרה: גיבוי בסיסי הנתונים במחשב המרכזי למחשב הגיבוי לשם הגנה עליהם.

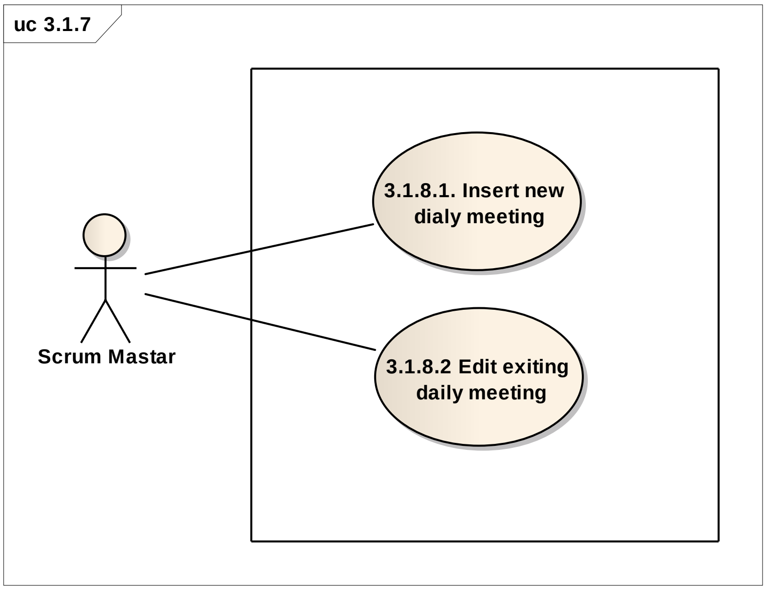
קלטים: אין.

עיבוד: הפונקציה תופעל באופן שוטף (כל 15 דקות) ותעביר את כל העידכונים מאז העידכון האחרון למחשב הגיבוי.

פלטים: המערכת תנהל log בו ירשם כל גיבוי. במידה והגיבוי נכשל תתקבל הודעה על כל מסכי המשתמשים במערכת שיש תקלה. המערכת תשלח דואר אלקטרוני לאחראי על תחזוקת המערכת ותשלח הודעה לביפר לכונן.

3.3.1.8 תיעוד Dialy Meeting

פגישת צוות יומית קצרצרה (עד 15 דקות) המכונה 'Stand up Meeting'. בפגישה מציג כל אחד מחברי הצוות את ההתקדמות, העבודה המתוכננת וקשיים אפשריים. הפגישה מתקיימת לרוב בעמידה, האפלקצייה תאפשר לסכם את עיקריי הפגישה.



<u>3.3.1.8.1</u> הכנס פגישה חדשה

<u>מטרה</u>:

המערכת מאפשרת להכניס פגישה חדשה.

<u>תנאי איתחול</u>:

המשתמש מנסה להיכנס למערכת.

;

למשתמש יש הרשאה להכניס פגישה חדשה.

<u>תנאי הצלחה</u>:

המשתמש תעד את הפגישה ,והפגישה נשמרה בבסיס נתונים .

<u>תנאי כשלון</u>:

הפגישה לא נשמרה בבסיס נתונים.

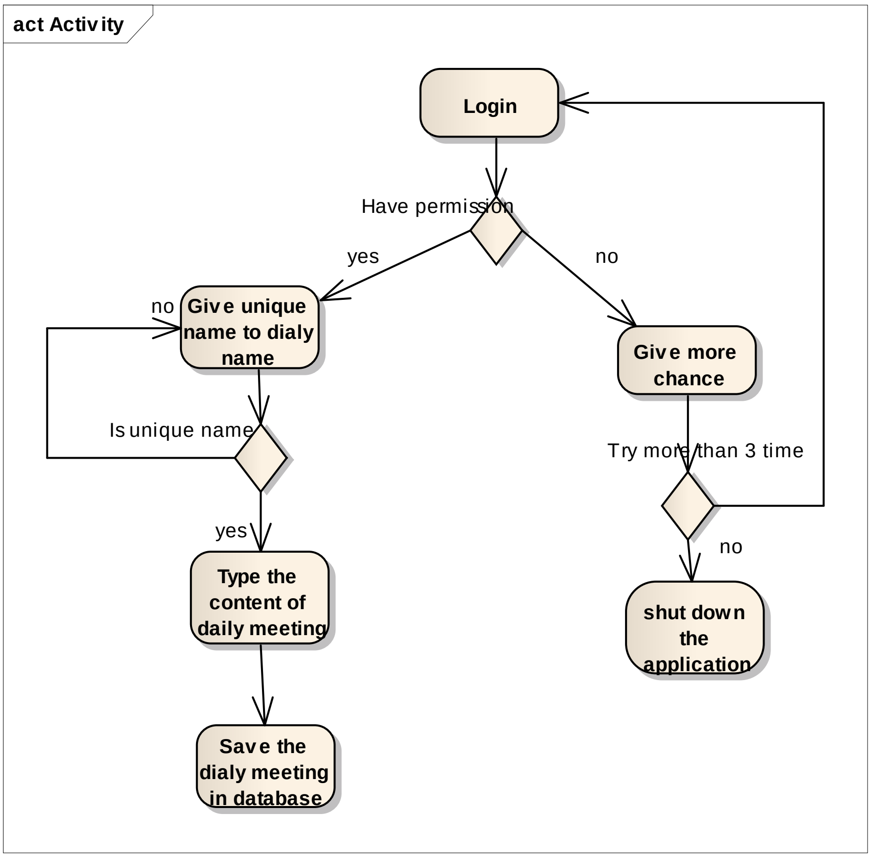
<u>טריגר</u>:

המשתמש מבקש להיכנס פגישה יומית חדשה .

<u>שחקנים</u>:

Scrum master

<u>התרחיש</u>



3.3.1.8.2 ערוך פגישה קיימת

<u>מטרה</u>:

המערכת מאפשרת לערוך פגישה הקיימת בבסיס נתונים.

;

<u>תנאי איתחול</u>:

המשתמש מנסה להיכנס למערכת.

למשתמש יש הרשאה לערוך פגישה חדשה.

<u>תנאי הצלחה</u>:

המשתמש ערך את הפגישה ,והשינויים נשמרו בבסיס נתונים .

<u>תנאי כשלון</u>:

השינויים בפגישה לא נשמרו בבסיס נתונים.

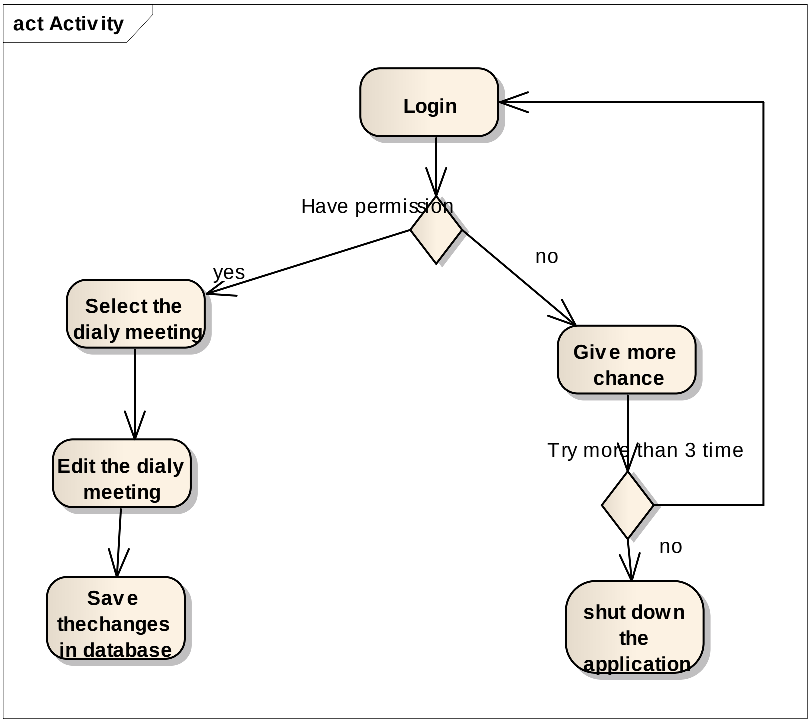
<u>טריגר</u>:

המשתמש מבקש לערוך פגישה יומית.

<u>שחקנים</u>:

Scrum master

<u>התרחיש</u>



;

3.3.1.9 הצג דוחות

האפלקצייה תיתן אפשרות לראות את מצב הפרוייקט על ידיי בחירה של הדו"ח הרצוי הדוחות יתנו תמונת מצב של הפרוייקט.

3.3.1.9.1 הצג דוחות לפי נושא

מטרה:

המערכת מאפשרת להציג דוחות.

תנאי איתחול:

למשתמש יש הרשאה לראות דוחות.

תנאי הצלחה:

הדוחות מוצגים למשתמש .

תנאי כשלון:

הפגישה לא נשמרה בבסיס נתונים.

טריגר:

המשתמש מבקש להיכנס פגישה יומית חדשה .

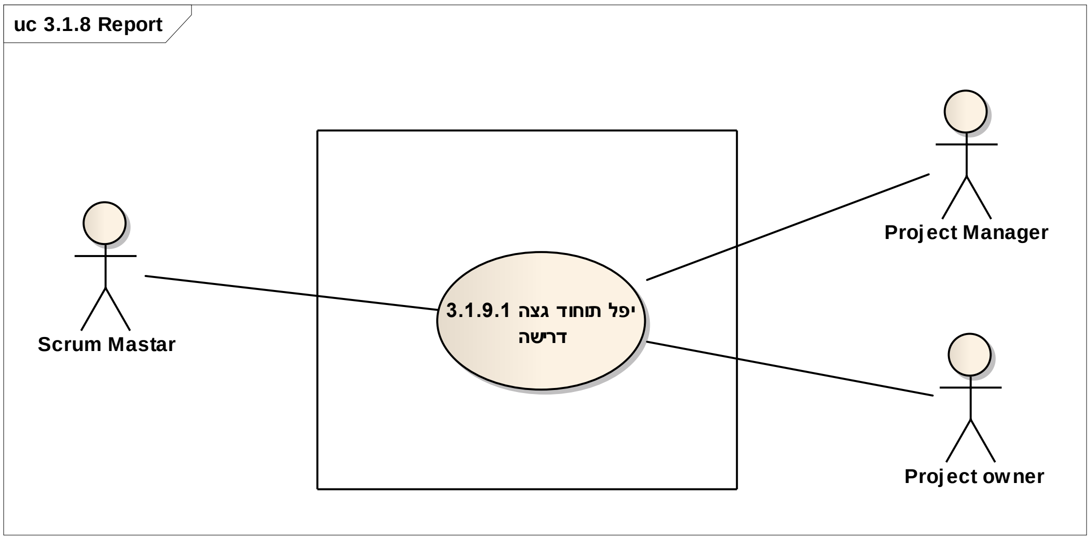
שחקנים:

Project owner

Project Manager

Scrum Master

התרחיש



;

3.3.2 **דרישות ממשק חיצוני**

3.3.2.1 ממשק משתמש

- ממשק המשתמש יהיה ממשק גרפי חלונאי.
- כל ממשק המשתמש יהיה בשפה אנגלית ותהייה אפשרות לתעד בשדות המאפשרים לתעד בשפה עברית.
- האפליקציה תקבל את הגדרות צבעים ופונטים מהגדרות המשתמש במערכת ההפעלה.
- כל הפונקציות שנקראות ע"י המשתמש (כפי העולה מפירוט הפונקציונליות) יופיעו בתפריטים וב tool bar (סטנדרט windows).
- בכל נקודה יוכל המשתמש להפעיל אופצית עזרה (סטנדרט windows) דרך תפריט או באמצעות מקש F1 שיעלה את מסך העזרה הרלוונטי.
- תהייה אפשרות להשתמש במערכת באמצעות עכבר והקלדה אולם תהייה גם אפשרות למשתמשים מיומנים לעבור בין השדות הדרושים להזנה באמצעות קיצורי דרך ו tab-ים.
- תהייה אפשרות להפעיל בממשק המשתמש mode עבור משתמשים מיומנים ו mode עבור משתמשים לא מיומנים. משתמשים מיומנים לא יהיו מעוניינים בהערות והצעות עזרה מהמערכת לגבי שדות בעייתים בעוד משתמשים בלתי מיומנים ישמחו לכך.
- לכל כפתור יהיה tool tip.
- לתהליכים בעלי הרשאות גבהות ובעלי הרשאות שלמשתמש אין הרשאה ,המשתמש לא יוכל לבצע פעולות אלו ,הפעולות יאפררו או יוסתורו כך שהמשתמש לא יוכל לבצע פעולות אלו (פעולות אסורות למשתמש)
- טרם ביצוע פעולות קריטיות המערכת תזהיר את המשתמש. אופציה זו תהיה ניתנת לביטול.

3.3.2.2 ממשק חומרה

- במחשב המרכזי יהיה מודם דרכו יהיה ניתן להתקשר עם כל עמדות הדורשות שירות.
- מכשר חיוב כרטיס אשראי על מנת לוודא פרטי כרטיס אשראי לבטחון תקפים ,ועל מנת לגבות תשלום בגמר החשבון.
- רשת ethernet של GB 1 וחיווט מסובך.
- רשת internet של 2 GB המחוברת לספק אינטרנט.

;

3.3.2.3 ממשקי תוכנה

- מערכת Windows 7 / Windows XP/Vista ו- Windows Server 2008 עבור המחשב המרכזי על כל המחשבים יותקן ה servise pack או האחרונה.
- תוכנת דואר שתשב על השרת המרכזי בכל עמדה יהיה ניתן למשוך את הדואר של המשתמש שנמצא ב login כרגע.
- תוכנה שמבצעת גיבוי אוטומטי של הנתונים בשרת אל מחשב הגיבוי תוך כיווץ האיפורמציה.
- תוכנה לניהול בסיסי הנתונים הקיימים במערכת. תוכנה זו תהיה מותקנת אך ורק במחשב המרכזי בו ממוקמים בסיסי הנתונים.

3.3.2.4 ממשקי תקשורת

- המחשבים (מסופים ו server) יהיו מחוברים ביניהם ברשת ethernet .
- כל המחשבים היו מחוברים ל רוטר.

3.3.3 <u>דרישות ביצועים</u>

ניתן לחלק את הפעולות אותן מבצעת המערכת לשני סוגים עיקריים:

- פעולות דחופות: פעולות הדורשות פלט מיידי למשתמש. בעיקר פעולות של הוספת או עריכת שדות אשר אמורות לעדכן את בסיס הנתונים.
- פעולות רגילות: פעולות אשר במידה ותהיה תקלה ,או זמן ביצועים הוא לא קריטי והזמן יכול להתארך ללא פגיעה במשתמש . לדוגמא: ביצוע גיבוי לבסיסי הנתונים, הפקת דוחות.

נדרוש ביצועים גבוהים מפעולות דחופות. זאת נשיג ע"י שימוש באלגוריתמים מהירים לביצוע חיפושים ומיונים, שימוש בציוד מחשב מתקדמים, ומערכות בסיסי נתונים מתקדמות ומהירות.

בכל מקרה, נדרוש שזמן חיפוש או מיון מידע במאגרים בעת פעולות דחופות לא יעלה על 2 שניה . כמו-כן, נדרוש שבעת פעולה רגילה זמן זה לא יעלה על 10 שניות.

.

3.3.4 <u>מגבלות תכנון</u>

- מגבלת החומרה הבעייתית ביותר הינה גודלם של בסיסי הנתונים. הנתונים במערכת יכולים להגיע לכמת גדולה של מידע גודל הזיכרון הפיזי ונפח הדיסק במחשב מוגבל

;

. כתוצאה ממגבלה זו יכתב שרות שיתרע במידה וקיבולת הדיסק מתקרבת לתפוסה מלאה.

- על מנת לשפר את ביצועי החיפוש במחשב המרכזי יש לשים בו מעבד חזק ככל האפשר, בניגוד לעמדות הקצה בהם ניתן לשים מעבדים חלשים יותר.
- לא יוזנו כל נתונים על מחשב כל שהוא אלא רק על המחשב המרכזי על מנת למנוע כפילות וחוסר תאימות לכן, בהתחשב בעובדה שהרשת מעבירה כמויות אינפורמציה רבות יש לצמצם עד כמה שניתן את הזמן שלוקח לרשת להעבור את הנתונים ולשם כך כדאי לשים רשת ethernet מהירה וחיווט מסוכך, כמו כן יש לבנות את הרשת כך שתאפשר שדרוג עתידי.

3.3.5 **מאפיינים**

3.3.5.1 **זמינות**

- על המערכת להיות זמינה כל הזמן. לא יבוצע לה אתחול אלא במקרה של תקלה. הזמינות תתאפשר עקב הגיבוי החשמלי שפורט.
- במידת האפשר תחזור המערכת לתמונת המצב האחרונה גם אם לא בוצע גיבוי.
- כאמור במקרה של תקלה יהיו גיבויים לחזור מהם אל המצב השמור האחרון.

3.3.5.2 **אבטחה**

- לכל משתמש במערכת תהיה סיסמא אישית. .
- המערכת תצא אוטומטית מ login במידה ולא השתמשו באפליקציה 10 דקות (פיתוח עתידי).

.

3.3.5.3 **תחזוקה**

- המערכת תבנה בצורה מודולרית כך שתאפשר הוספה ושדרוג קלים של מרכיביה.
- המערכת תתוחזק במידה של תקלות ובאגים הבאגים יתועדו ויתוקנו .
- בסיס הנתונים ינוהל ע"י תוכנת מנהל בסיס נתונים.
- גיבוי המערכת יתבצע כאמור באופן שוטף. הגיבוי יבוצע באופן כזה שלא תידרש התערבות חיצונית. במקרה של תקלה הבעיה תיפתר ידנית ע"י הגוף המתחזק.

;

3.3.6 **<u>דרישות נוספות</u>**

מטעמים של שמירה על סודיות יש לשמור את המחשב המרכזי ומחשב הגיבוי בחדר מבודד ונעול שהגישה אליו תהיה רק למנהל המערכת.

על מנת למנוע את נפילת הרשת על ידי חבלה מכוונת או לא מכוונת מחד ולאפשר גישה מהירה לתיקון מאידך יש להטמין את החיווט בקיר אולם לאפשר גישה לתיקון מבלי לשבור קירות. כמו כן יש להעביר חיווט כפול לשם גיבוי.

הגיבוי צריך להישמר במקום חיצוני מעת לעת לשם מניעת מצב בו כל נתונים יושמדו בשרפה או במפגע אחר.

;

3.4 פיתוחים עתידיים

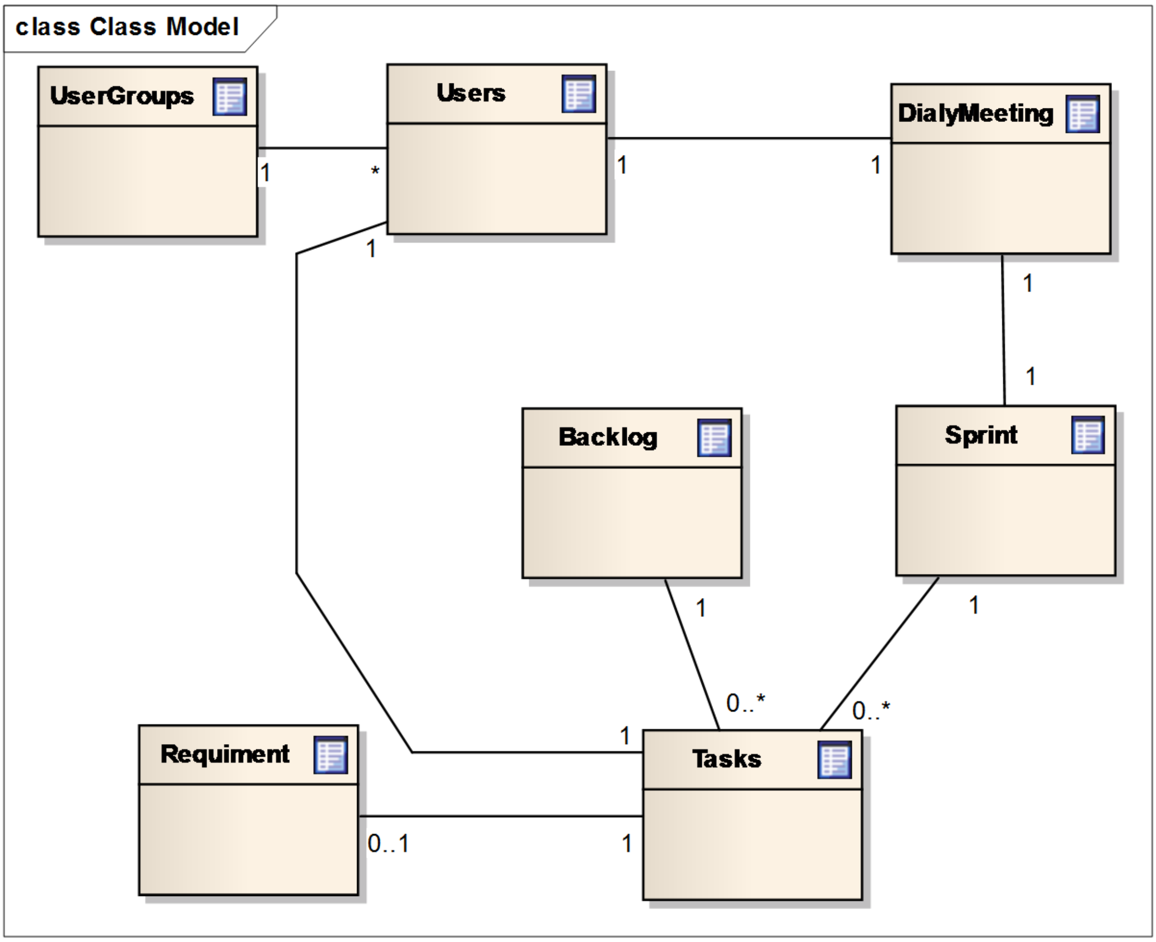
בפרק זה מצוינות הצעות לשיפור והרחבת המערכת, אשר יכולות להתווסף בעתיד לפרוייקט.

- הוספת אפשרות לטיפול בכמה פרוייקטים במקביל.
- הוספת נעילות בין שתי .Client במידה ומשתמש אחד התחיל לעדכן שהשני לא יוכל במקביל לדרוס לו את השינויים.
- אפשרות ל tractability בין דרישות למשימות שלהם.
- התוכנה תאפשר לשלוח למשתמש הודעה באמצעות דואר אלקטרוני במידה ומשימות לא הושלמו בזמן ,או ש Scrum master רוצה להודיע הודעה כל שהיא למשתמשים.
- המערכת תצא אוטומטית מ login במידה ולא השתמשו באפליקציה 10 דקות.

;

3.5 נספח

ERD


class Class Model
UserGroups
Users
DialyMeeting
Backlog
Sprint
Requiment
Tasks
1
*
1
1
1
1
1
1
1
0..*
0..*
1
0..1
1

;

- UserGroups

טבלה המכילה את קבוצות ההרשאות במערכת כל משתמש יכול להיות שייך לקבוצה אחת בלבד

  - Name: שם של הקבוצה.
  - Description:תאור של הקבוצה
  - UserGroupsID: מספר יחודי המזהה את הקבוצה.

- Users

טבלה המכילה את נתוני המשתמשים במערכת לטבלה זו יש קישור לטבלת UserGroups שלמעשה כל משתמש מצביע על קבוצה אליה הוא שייך.

  - Name: שם של המשתמש.
  - Description:תאור של המשתמש.
  - UserGroupName: שם הקבוצה שאליו שייך המשתמש.
  - Password: סיסמת המשתמש.

- DialyMeeting

טבלה המכילה את נתוני הישיבות שנערכות בכל יום ,טבלה זו משייכת לטבלת Sprint שכן כל ישיבה שייכת ל Sprint כל שהוא.

  - Name: שם של הישיבה.
  - Description:תאור של הישיבה היומית.
  - SprintID: שם ה Sprint שאליו שייכת הישיבה.
  - Owner: מזהה מי יצר את הישיבה .

- Sprint

טבלה המכילה את נתוני האטרציית הפיתוח.

  - Name: שם של Sprint.
  - Description:תאור של ה Sprint.
  - SprintID: מספר יחודי המזהה את ה Sprint.
  -

- Backlog

טבלה המכילה את מחסן המשימות שעדיין לא שויכו ל Sprint מסויים

  - Name: שם של Backlog.
  - Description:תאור של ה Backlog.
  - BacklogID: מספר יחודי המזהה את ה Backlog.

- WorkItem

טבלה המכילה את מחסן המשימות שעדיין לא שויכו ל Sprint מסויים

  - WorkItemID : מספר יחודי המזהה את WorkItem.
  - Name : שם של WorkItem.
  - Description: תאור של ה WorkItem.
  - AssignTo:למי המשימה מופנת.

;

- Owner:מי יצר את משימה זו.
- EstimateTImeLow:שערוך תחתון בשעות כמה המשימה תיקח.
- EstimateTImeHigh:שערוך עליון כמה המשימה תיקח
- Status: מצב המשימה.
- SprintID: מציין לאיזה Sprint המשימה שייכת.
- BackLogID:מציין לאיזה Backlog המשימה שייכת

- Requirement

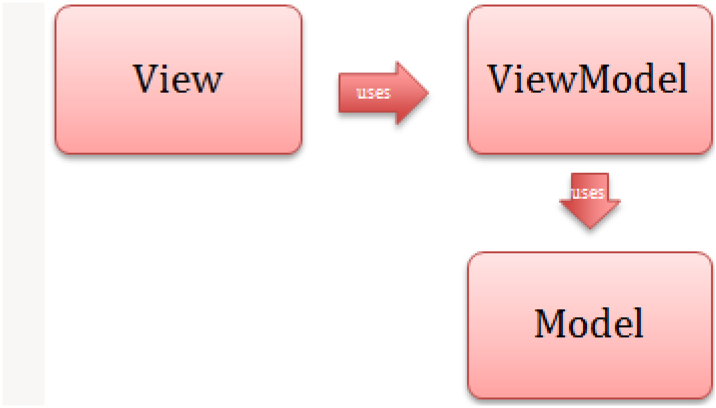
טבלה המכילה את דרישות הפרוייקט.

- RequirementID :   מספר יחודי המזהה את Requirement.
- Name : שם של Requirement.
- Description: תאור של ה Requirement.

;

4 Software System Design

פיתוח האפליקצייה נעשה באמצעות שימוש ב Design Pattern MVVM



Model

מייצג את הנתונים שאני מחזיק השייכים לעולם הבעיה של הפרוייקט, נתונים אלו יכולים למשל נתונים משכבת הפונה לבסיס הנתונים(Data Access layer) שכבה זו לא מודעת לשכבת ה UI.

View

שכבת התצוגה שכבה שבה יש את כל האלמנטים של התצוגה למשל מיקום כפתורים צבעים וטיפול במאפיינים השונים של הפקדים . שכבה זה לא אמורה להיות מודעת לשכבת ה Model

מי שאמור להיות מתווך בין שכבה זו לשכבת המודל זו שכבת Model VIEW.

ViewModel

שכבה זו למעשה מחברת בין שכבת ה UI לשכבת ה Model למעשה כל הפקודות המגיעות מה UI אמורות להגיע לשכבה זו ובמידה וצריך הנתונים שכבה זו דואגת לפנות לשכבת ה Model

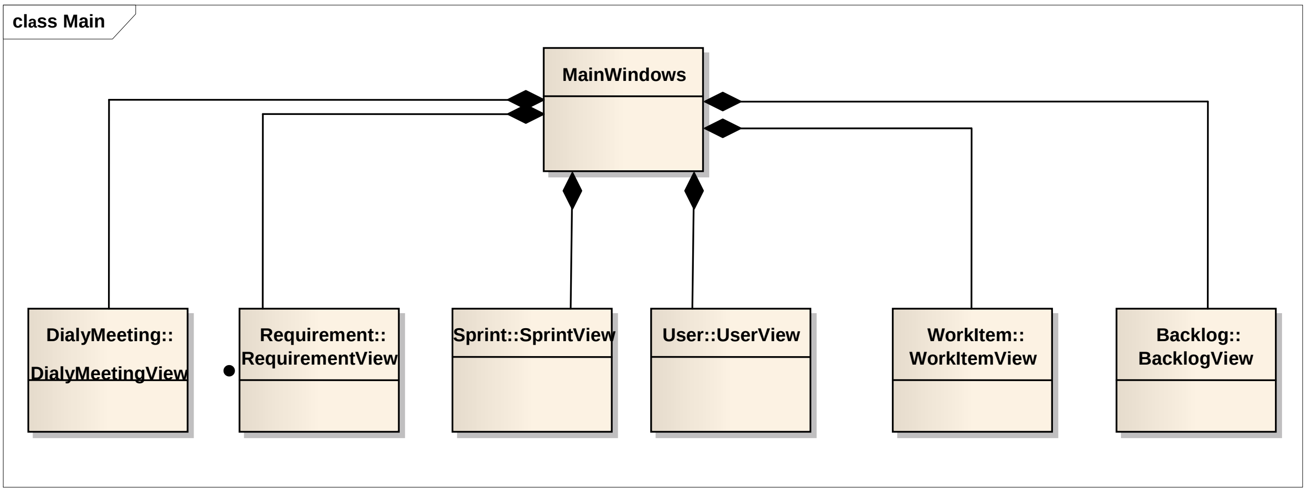
ולהביא אותם.

;

MainWindows

Class שלמעשה משמש כ Façade, מכיל את כל View class באפליקציה ומציג את class לפי בחירת המשתמש.

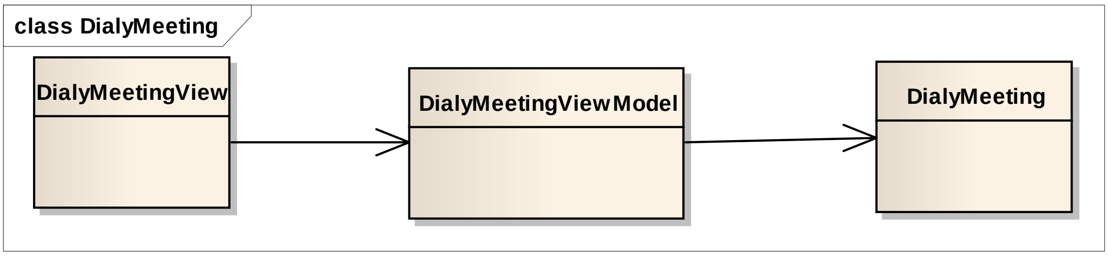
לדוגמא במידה והמשתמש בחר לראות נתוני Sprint , MainWindows מסתיר את שאר View ומציג למשתמש את נתוני ה Sprint (מציג את נתוניי class SprintView)



;

DialyMeeting

אחראי על הצגה ותיעוד של הישיבות הימיות



- DialyMeetingView

אחראי על הצגת רכיבי ה UI של DialyMeeting

- DialyMeetingViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class DialyMeeting) על מנת לעדכן נתונים בבסיס נתונים. הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ולבטל שינויים שהתבצעו.

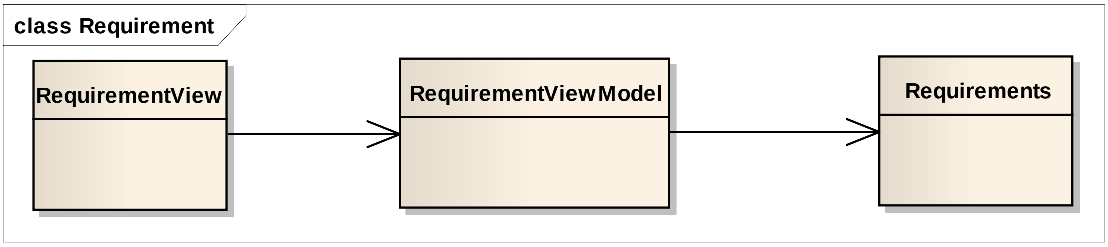
- DialyMeeting

בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של DialyMeeting בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים DialyMeeting חייב לפנות ל class זה.

;

Requirement

אחראי על הצגה ותיעוד של דרישות המערכת



- RequirementView

אחראי על הצגת רכיבי ה UI של Requirement

- Requirement ViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Requirement) על מנת לעדכן נתונים בבסיס נתונים. הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ולבטל שינויים שהתבצעו.

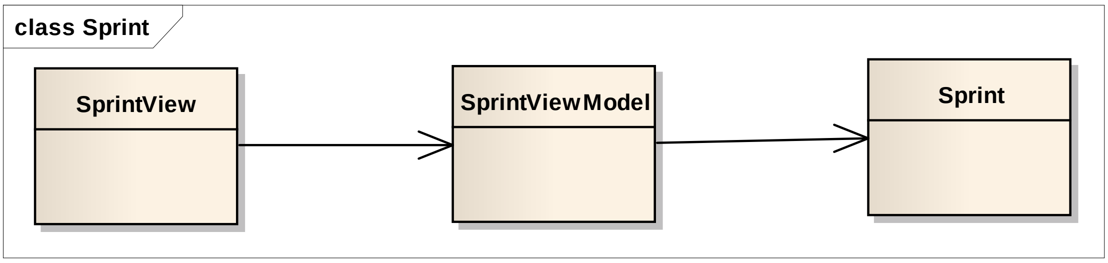
- Requirement

בעצם חלק משכבת המודל ,מחזיק את כל נתונים הנחוצים של Requirement בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל Requirement חייב לפנות ל class זה.

;

Sprint

אחראי על הצגה ועדכון של נתוניי ה Sprint.



- SprintView

אחראי על הצגת רכיבי ה UI של Sprint

- Sprint ViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Sprint) על מנת לעדכן נתונים בבסיס נתונים.

הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,שיוך WorkItem ל Sprint.

- Sprint

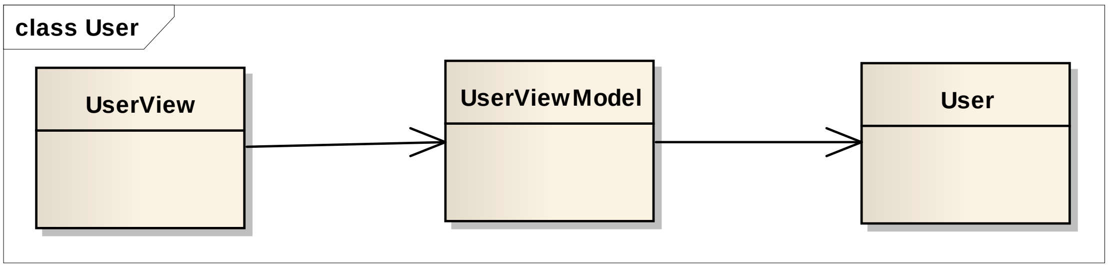
בעצם חלק משכבת המודל ,מחזיק את כל נתונים הנחוצים של Sprint

בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל   Sprint חייב לפנות ל   class זה.

;

User

אחראי על הצגה ועדכון של משתמשיי המערכת



- UserView
  אחראי על הצגת רכיבי ה UI של User
- UserViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class User) על מנת לעדכן נתונים בבסיס נתונים.
הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,שיוך.

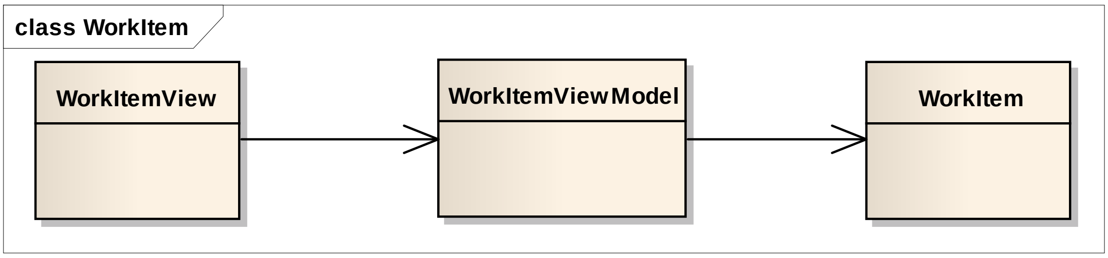
- User

בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של User
בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל User חייב לפנות ל class זה.

;

WorkItem

אחראי על הצגה ועדכון של משימות.



- WorkItemView
  אחראי על הצגת רכיבי ה UI של WorkItem
- WorkItemViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class WorkItem) על מנת לעדכן נתונים בבסיס נתונים.
הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,ויצוא נתונים לקובץ אקסל.

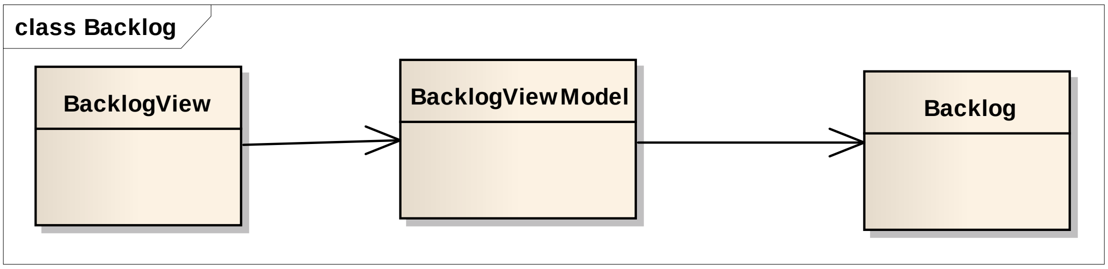
- WorkItem

 בעצם חלק משכבת המודל ,מחזיק את כל נתונים הנחוצים של WorkItem
 בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל WorkItem חייב לפנות ל   class זה.

;

<u>Backlog</u>

אחראי על הצגה ועדכון של צובר המשימות במערכת.



- BacklogView

  אחראי על הצגת רכיבי ה UI של Backlog
- BacklogViewModel

מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Backlog) על מנת לעדכן נתונים בבסיס נתונים.
הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,הקצאת משימה ל Backlog.

- Backlog

  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של Backlog

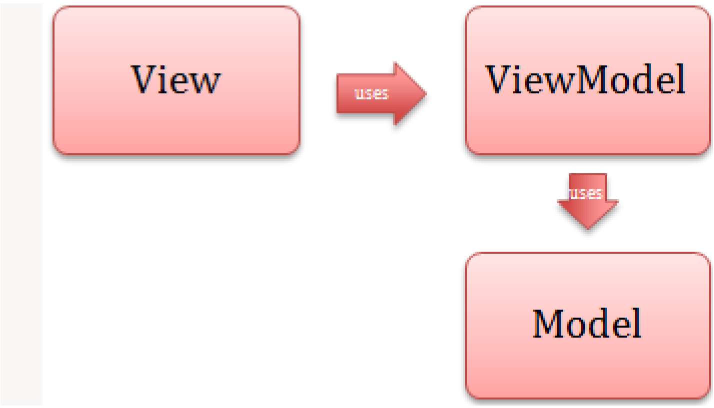
  בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל Backlog חייב לפנות ל class זה.

;

5 Software Detail Design

פיתוח האפלקצייה נעשה באמצעות שימוש ב Design Pattern MVVM

MVVM מפריד בין שכבת (View) UI לשכבת שניגשת לבסיס הנתונים(Model)



<u>Model</u>

מייצג את הנתונים שאני מחזיק השייכים לעולם הבעיה של הפרוייקט ,נתונים אלו יכולים למשל נתונים משכבת הפונה לבסיס הנתונים(Data Access layer) שכבה זו לא מודעת לשכבת ה UI.

<u>View</u>

שכבת התצוגה שכבה שבה יש את כל האלמנטים של התצוגה למשל מיקום כפתורים צבעים וטיפול במאפיינים השונים של הפקדים . שכבה זה לא אמורה להיות מודעת לשכבת ה Model

מי שאמור להיות מתווך בין שכבה זו לשכבת המודל זו שכבת Model VIEW.

<u>ViewModel</u>

שכבה זו למעשה מחברת בין שכבת ה UI לשכבת ה Model למעשה כל הפקודות המגיעות מה UI אמורות להגיע לשכבה זו ובמידה וצריך הנתונים שכבה זו דואגת לפנות לשכבת ה Model

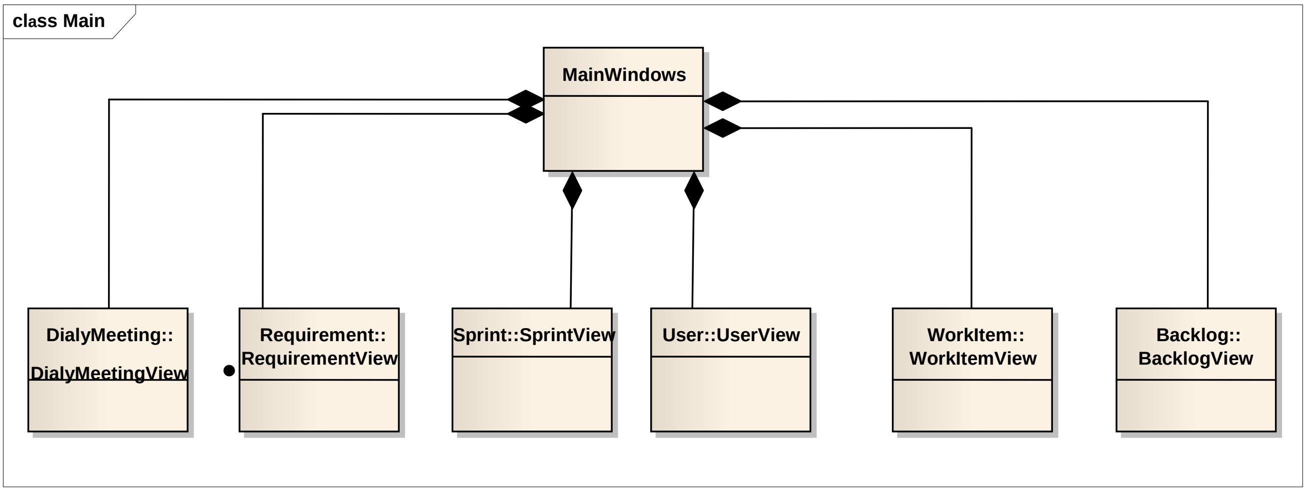
ולהביא אותם.

;

MainWindows

Class שלמעשה משמש כ Façade, מכיל את כל View class באפליקציה ומציג את class לפי בחירת המשתמש.

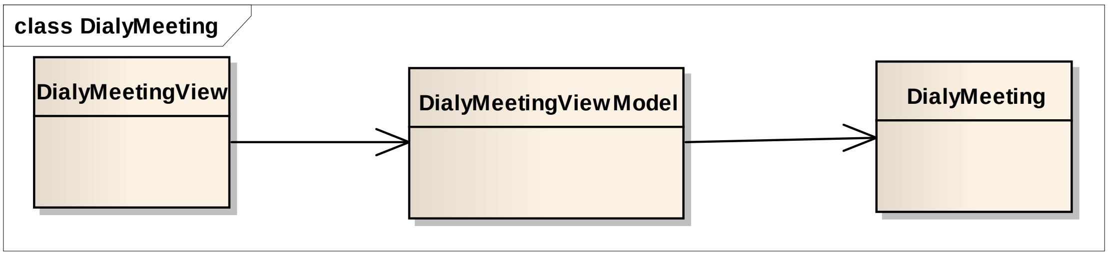
לדוגמא במידה והמשתמש בחר לראות נתוני Sprint , MainWindows מסתיר את שאר View ומציג למשתמש את נתוני ה Sprint (מציג את נתוניי class SprintView)



;

DialyMeeting

אחראי על הצגה ותיעוד של הישיבות הימיות

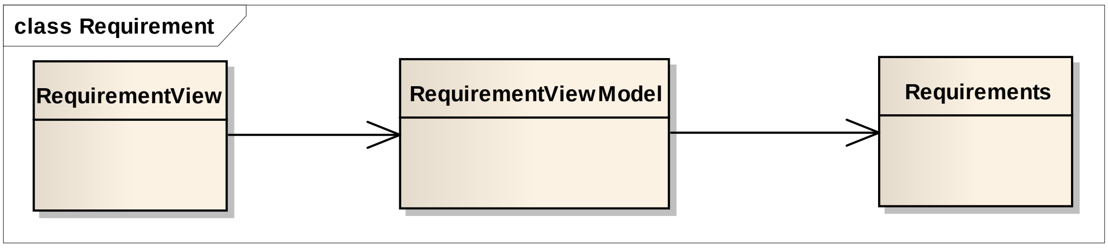


- DialyMeetingView
  אחראי על הצגת רכיבי ה UI של DialyMeeting
- DialyMeetingViewModel
  מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class DialyMeeting) על מנת לעדכן נתונים בבסיס נתונים.
  הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ולבטל שינויים שהתבצעו.
- DialyMeeting
  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של DialyMeeting בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים DialyMeeting חייב לפנות ל class זה.

;

Requirement

אחראי על הצגה ותיעוד של דרישות המערכת

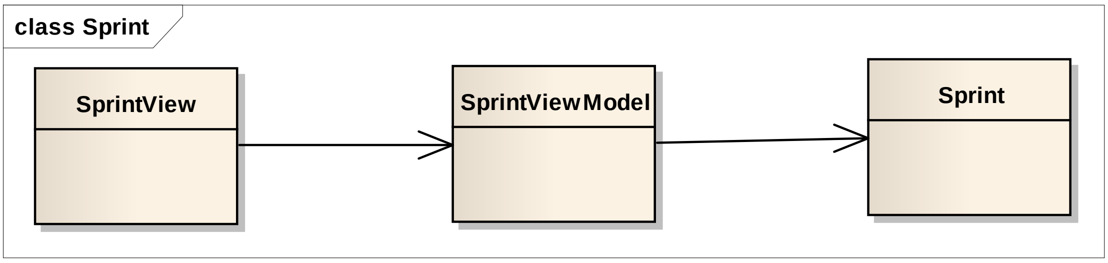


- RequirementView
אחראי על הצגת רכיבי ה UI של Requirement
- Requirement ViewModel
מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Requirement) על מנת לעדכן נתונים בבסיס נתונים.
הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ולבטל שינויים שהתבצעו.
- Requirement
בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של Requirement בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל Requirement חייב לפנות ל class זה.

;

Sprint

אחראי על הצגה ועדכון של נתוניי ה Sprint.

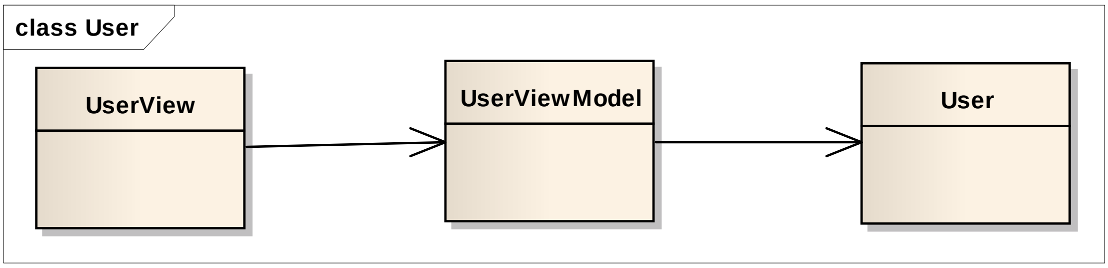


- SprintView
  אחראי על הצגת רכיבי ה UI של Sprint
- Sprint ViewModel
  מהווה מתווך בין שכבת ה   UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Sprint) על מנת לעדכן נתונים בבסיס נתונים.
  הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,שיוך WorkItem ל Sprint.
- Sprint
  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של Sprint
  בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל   Sprint חייב לפנות ל   class זה.

;

User

אחראי על הצגה ועדכון של משתמשיי המערכת

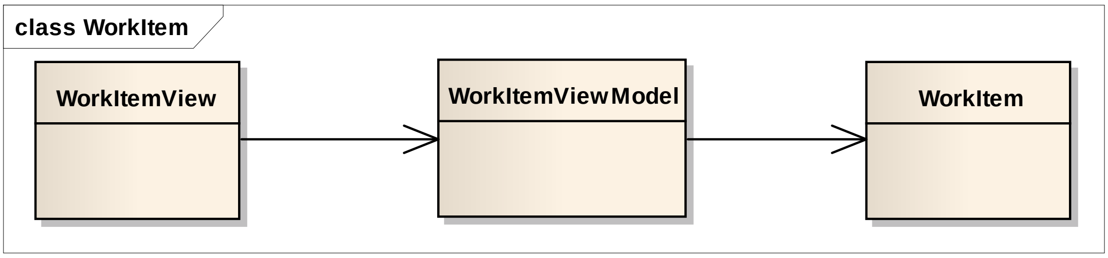


- UserView
  אחראי על הצגת רכיבי ה UI של User
- UserViewModel
  מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class User) על מנת לעדכן נתונים בבסיס נתונים.
  הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,שיוך.
- User
  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של User
  בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל User חייב לפנות ל class זה.

;

WorkItem

אחראי על הצגה ועדכון של משימות.

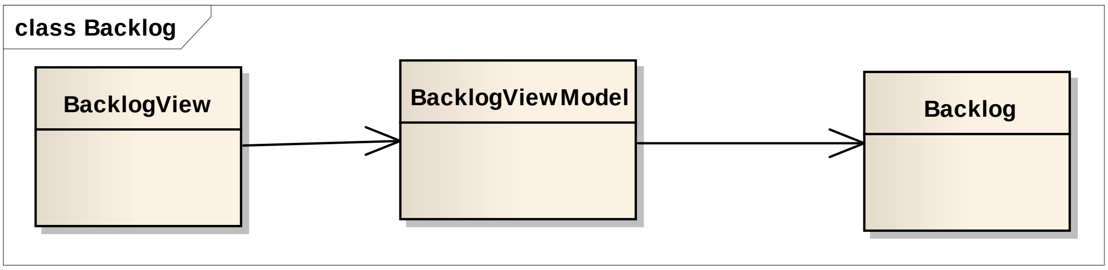


- WorkItemView
  אחראי על הצגת רכיבי ה UI של WorkItem
- WorkItemViewModel
  מהווה מתווך בין שכבת ה UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class WorkItem) על מנת לעדכן נתונים בבסיס נתונים.
  הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,ויצוא נתונים לקובץ אקסל.
- WorkItem
  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של WorkItem
  בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל WorkItem חייב לפנות ל class זה.

;

Backlog

אחראי על הצגה ועדכון של צובר המשימות במערכת.



- BacklogView
  אחראי על הצגת רכיבי ה UI של Backlog
- BacklogViewModel
  מהווה מתווך בין שכבת ה   UI לשכבת ה Model לוכד את בקשות של המשתמש ובמידת הצורך פונה ל שכבת המודל (ל class Backlog) על מנת לעדכן נתונים בבסיס נתונים.
  הפעולות ש class לוכד מ המשתמש עדכון הוספה מחיקה שמירה ביטול שינויים שהתבצעו,הקצאת משימה ל Backlog.
- Backlog
  בעצם חלק משכבת המודל, מחזיק את כל נתונים הנחוצים של Backlog
  בעל גישה ישירה לבסיס נתונים כל מי שצריך לעדכן להוסיף או למחוק נתונים הקשורים ל   Backlog חייב לפנות ל   class זה.

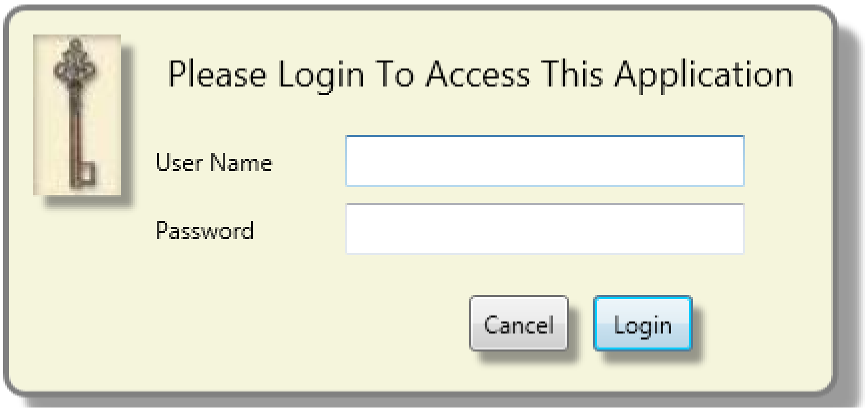
;

6 Operational Requirement Specification

מסך Login

מסך כניסה המאפשר לבצע אימות על פי שם משתמש וסיסמא

במידה והמשתמש ישגה 3 פעמים רצוף האפלקציה תסגר.

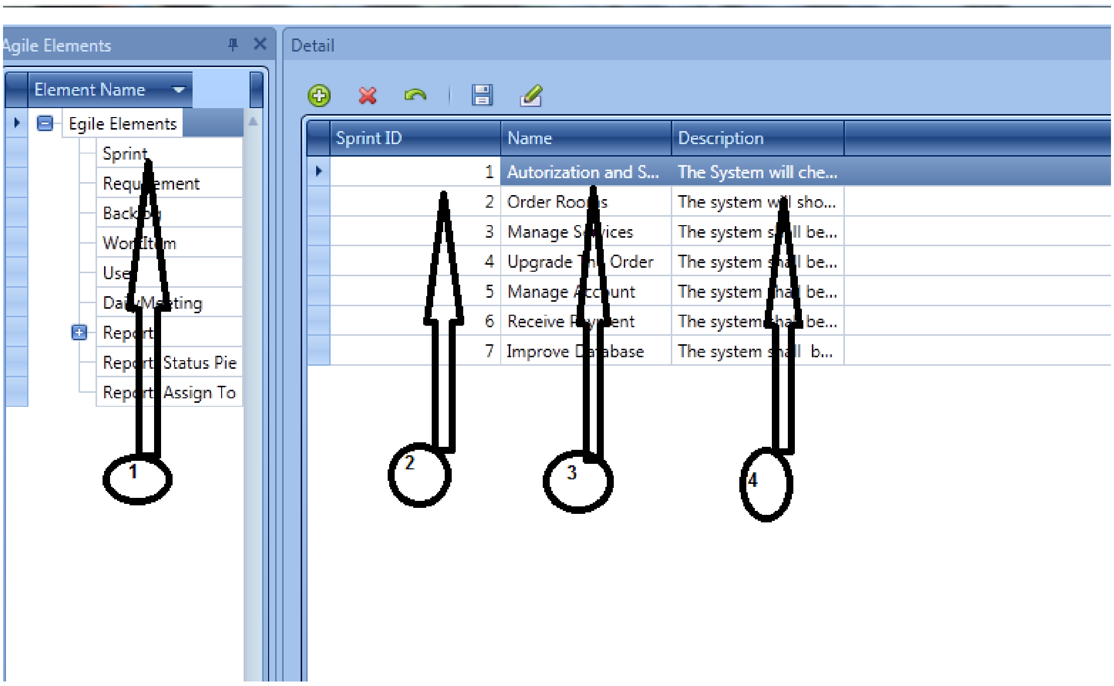


במידה ומשתמש עבר את שלב האימות הוא ימשל לתהליך של בדיקה איזה הרשאה יש לו ומה מותר להציג לו על פי הרשאתו .

;

מסך עריכת Sprint

במסך זה ניתן לערוך ולהוסיף **Sprint**

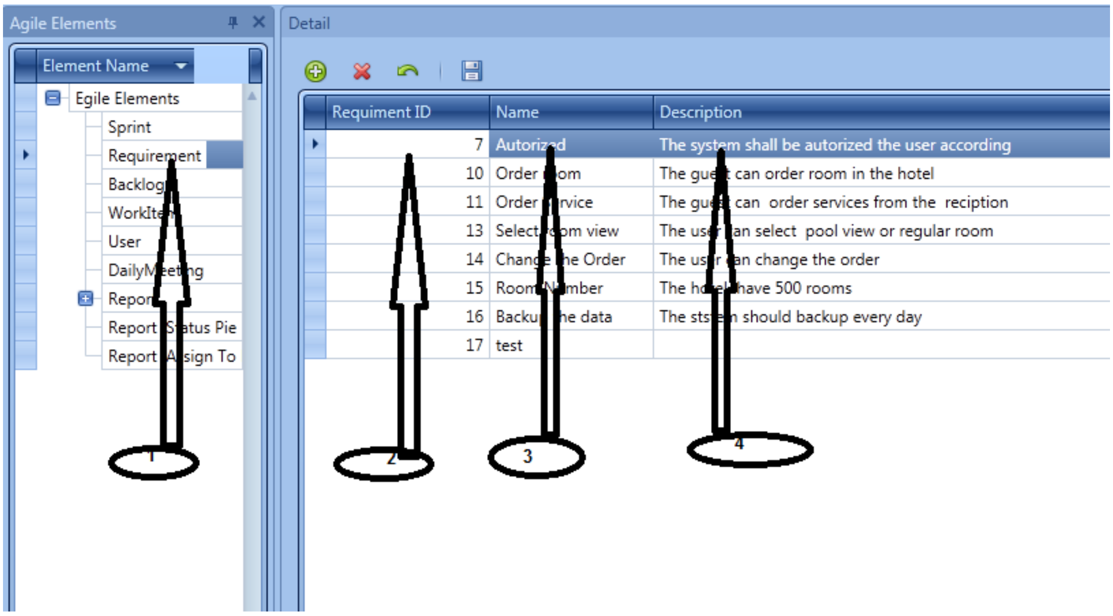


1. לחיצה בעץ האלמנטים על Sprint יביא אותנו למסך עריכת Sprint
2. שדה המציין מספר יחודי של Sprint
3. שדה המציין את שם ה Sprint
4. שדה המציין תאור של ה Sprint
5. הוסף Sprint חדש
6. מוחק Sprint קיים
7. חוזר צעד לפני השמירה האחרונה
8. שמור את השינויים
9. חבר משימה ל Sprint

;

<u>מסך עריכת Requirement</u>

במסך זה ניתן לערוך ולהוסיף דרישות

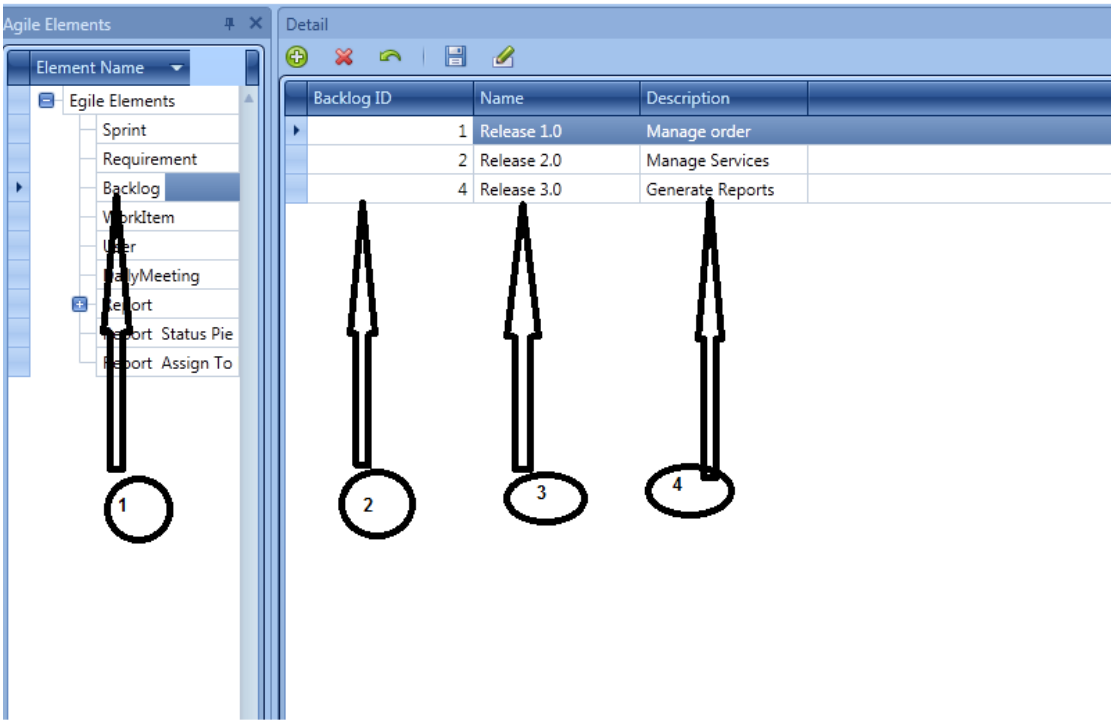


1. לחיצה בעץ האלמנטים על Requirement יביא אותנו למסך עריכת Requirement
2. שדה המציין מספר יחודי של Requirement
3. שדה המציין את שם ה Requirement
4. שדה המציין תאור של ה Requirement
5. הוסף Requirement חדש
6. מוחק Requirement קיים
7. חוזר לפני השמירה האחרונה
8. שמור את השינויים

;

מסך עריכת Backlog

במסך זה ניתן לערוך ולהוסיף Backlog

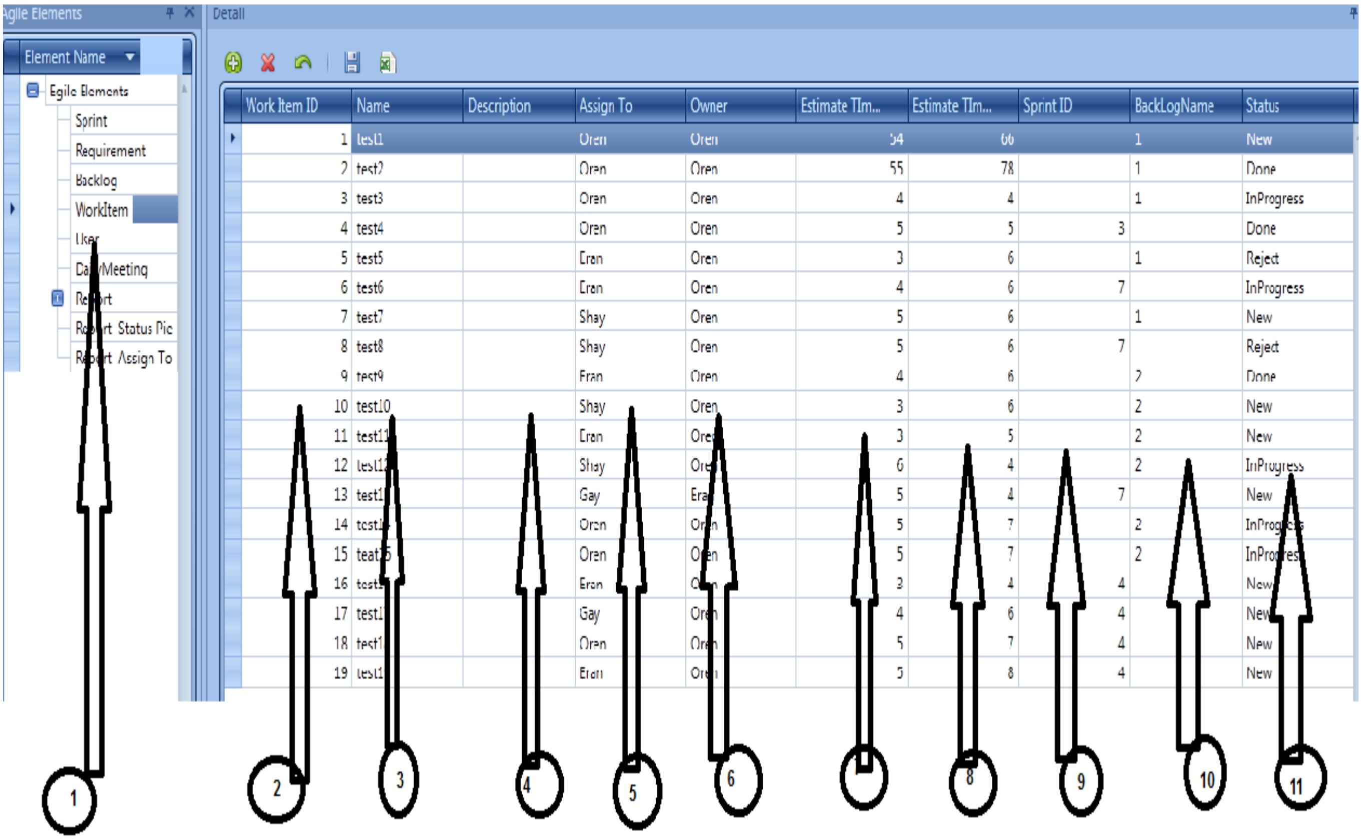


1. לחיצה בעץ האלמנטים על Backlog יביא אותנו למסך עריכת Backlog
2. שדה המציין מספר יחודי של Backlog
3. שדה המציין את שם ה Backlog
4. שדה המציין תאור של ה Backlog
5. הוסף Backlog חדש
6. מוחק Backlog קיים
7. חוזר צעד לפני השמירה האחרונה
8. שמור את השינויים

;

מסך עריכת WorkItem

במסך זה ניתן לערוך ולהוסיף משימות



1. לחיצה בעץ האלמנטים עלאק WorkItem יביא אותנו למסך עריכת WorkItem
2. שדה המציין מספר יחודי של WorkItem
3. שדה המציין את שם ה WorkItem
4. שדה המציין תאור של ה WorkItem
5. Assign To שדה המציין למי מיועדת המשימה
6. Owner שדה המציין מי יצר את המשימה
7. Estimate Time Low שדה המציין הערכה תחתונה
8. Estimate Time High שדה המציין הערכה עליונה
9. SprintID שדה המציין באיזה Sprint מבצעים את המשימה
10. BacklogName שדה המציין באיזה Backlog המשימה שייכת.
11. Status שדה המציין את מצב המשימה
12. הוסף WorkItem חדש
13. מוחק WorkItem קיים

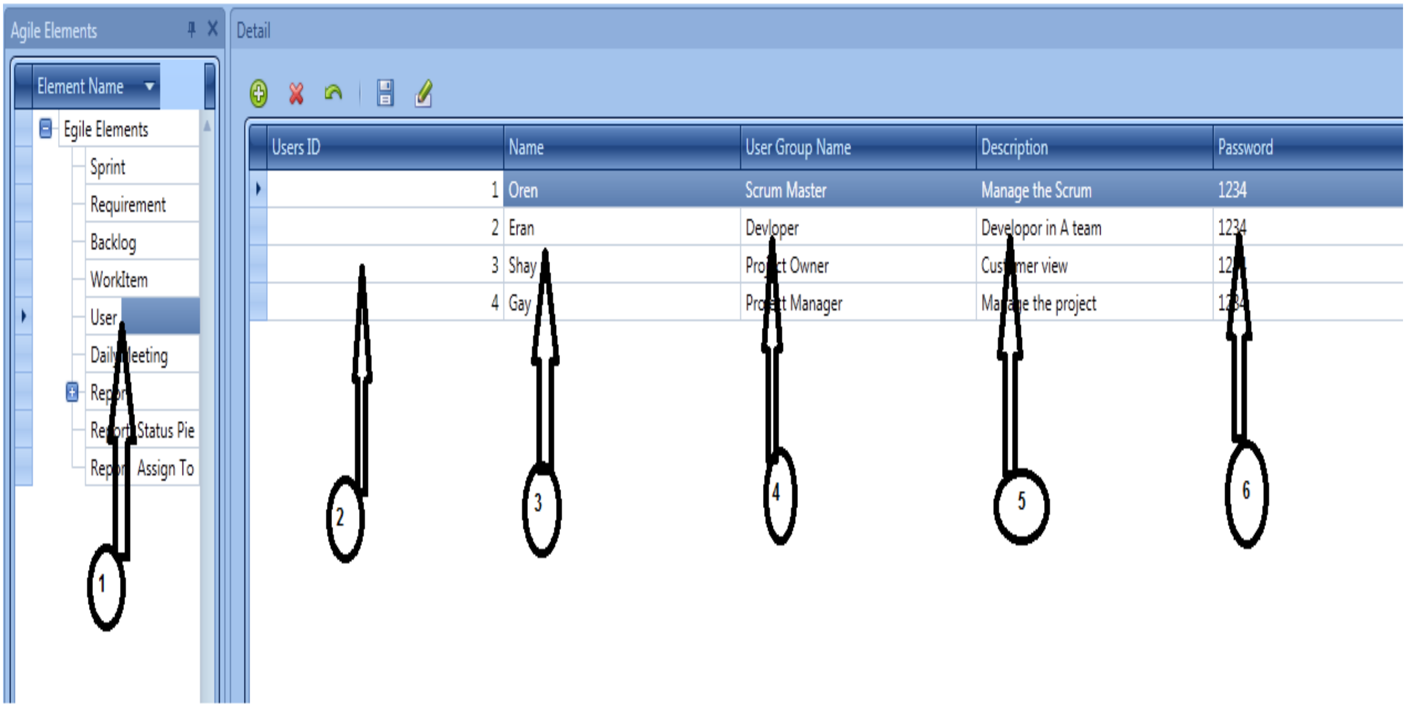
;

14. חוזר צעד לפני השמירה האחרונה
15. שמור את השינויים
16. ייצא את הנתונים לקובץ אקסל

;

<u>מסך עריכת Users</u>

במסך זה ניתן לערוך ולהוסיף משתמשים במערכת

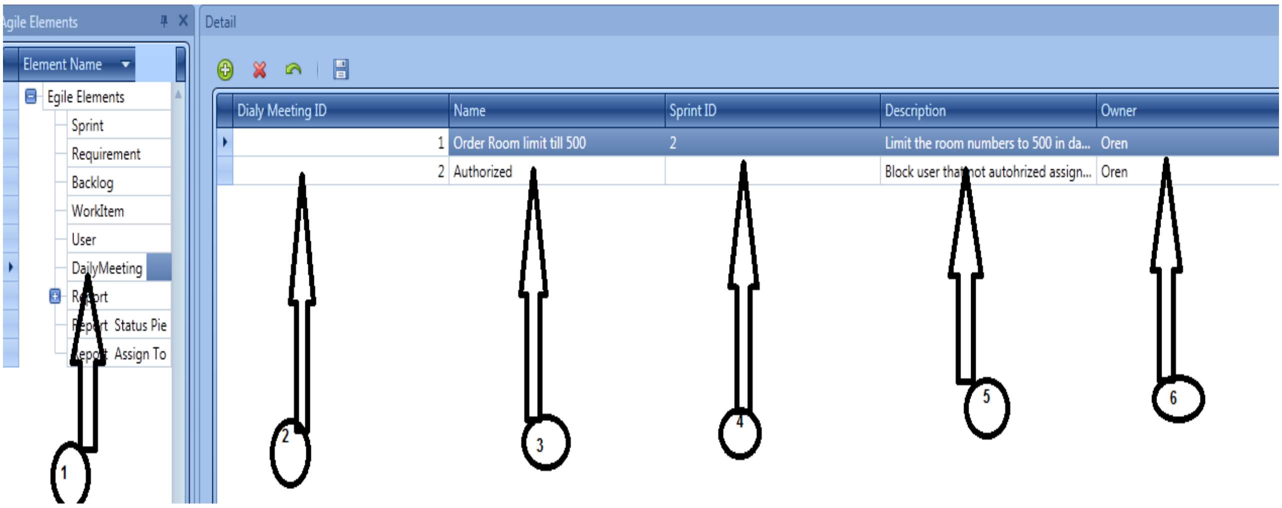


1. לחיצה בעץ האלמנטים על User יביא אותנו למסך עריכת User
2. שדה המציין מספר יחודי של User
3. שדה המציין את שם ה User
4. שדה המציין את שם ה קבוצה ש User שייך אליו
5. שדה המציין תאור של ה User
6. שדה המציין סיסמת המשתמש
7. הוסף User חדש
8. מוחק User קיים
9. חוזר צעד לפני השמירה האחרונה
10. שמור את השינויים

;

<u>מסך עריכת Daily Meeting</u>

במסך זה ניתן לערוך ולהוסיף פגישות יומיות



1. לחיצה בעץ האלמנטים על Daily Meeting יביא אותנו למסך עריכת Daily Meeting
2. שדה המציין מספר יחודי של Daily Meeting
3. שדה המציין את שם ה Daily Meeting
4. שדה המציין את שם ה Sprint של Daily Meeting שייך אליו
5. שדה המציין תאור של ה Daily Meeting
6. שדה המציין מי יצר את Daily Meeting
7. הוסף Daily Meeting חדש
8. מוחק Daily Meeting קיים
9. חוזר צעד לפני השמירה האחרונה
10. שמור את השינויים

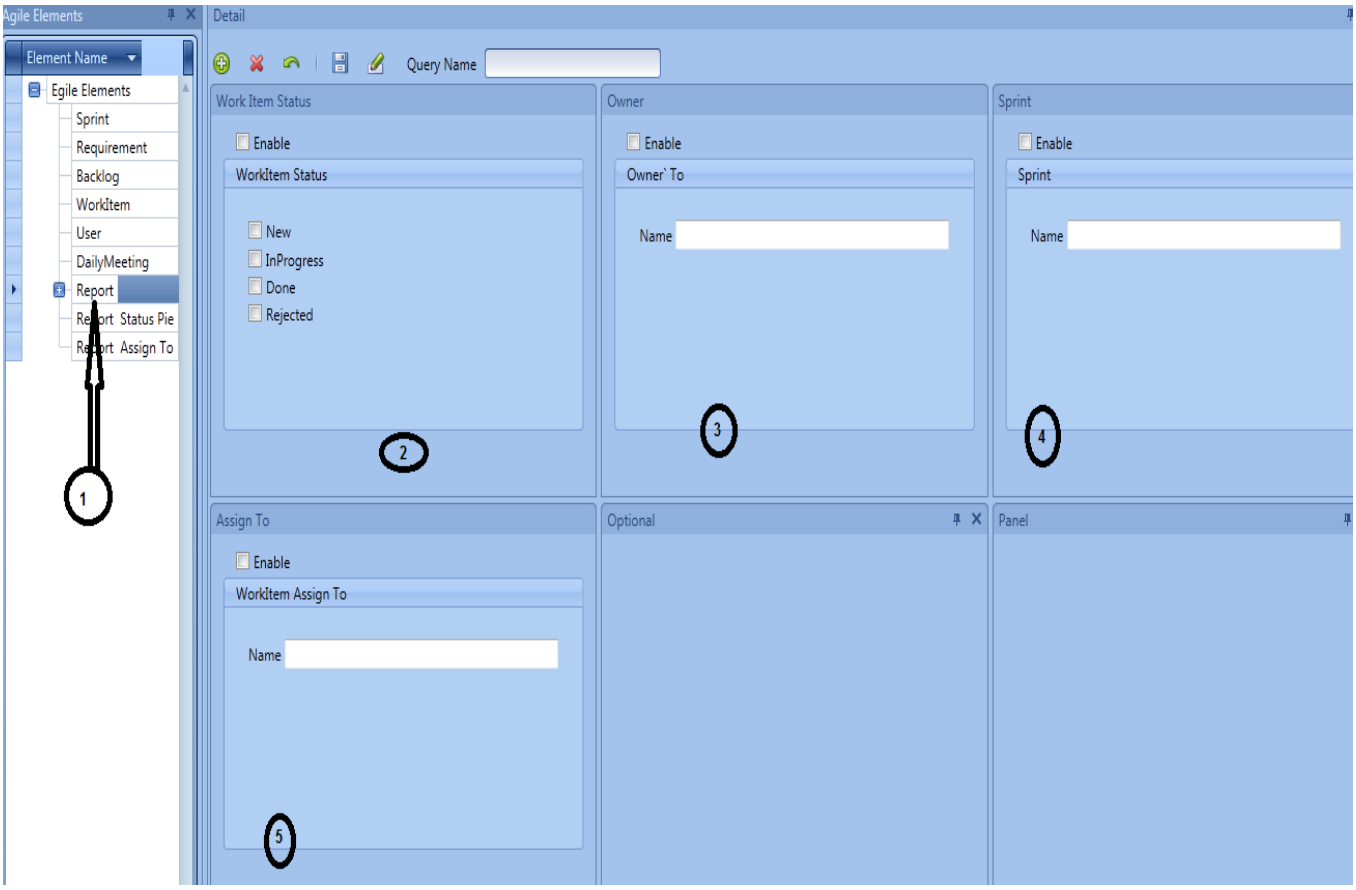
;

מסך יצירת Report

במסך זה ניתן ליצור שאילתות דינמיות

בניית שאילתא לפי החוקים הבאים:

- הקשר בין החלקים הוא And
- אם חלק ריק זאת אומרת שהוא לא משתתף בשאילתא



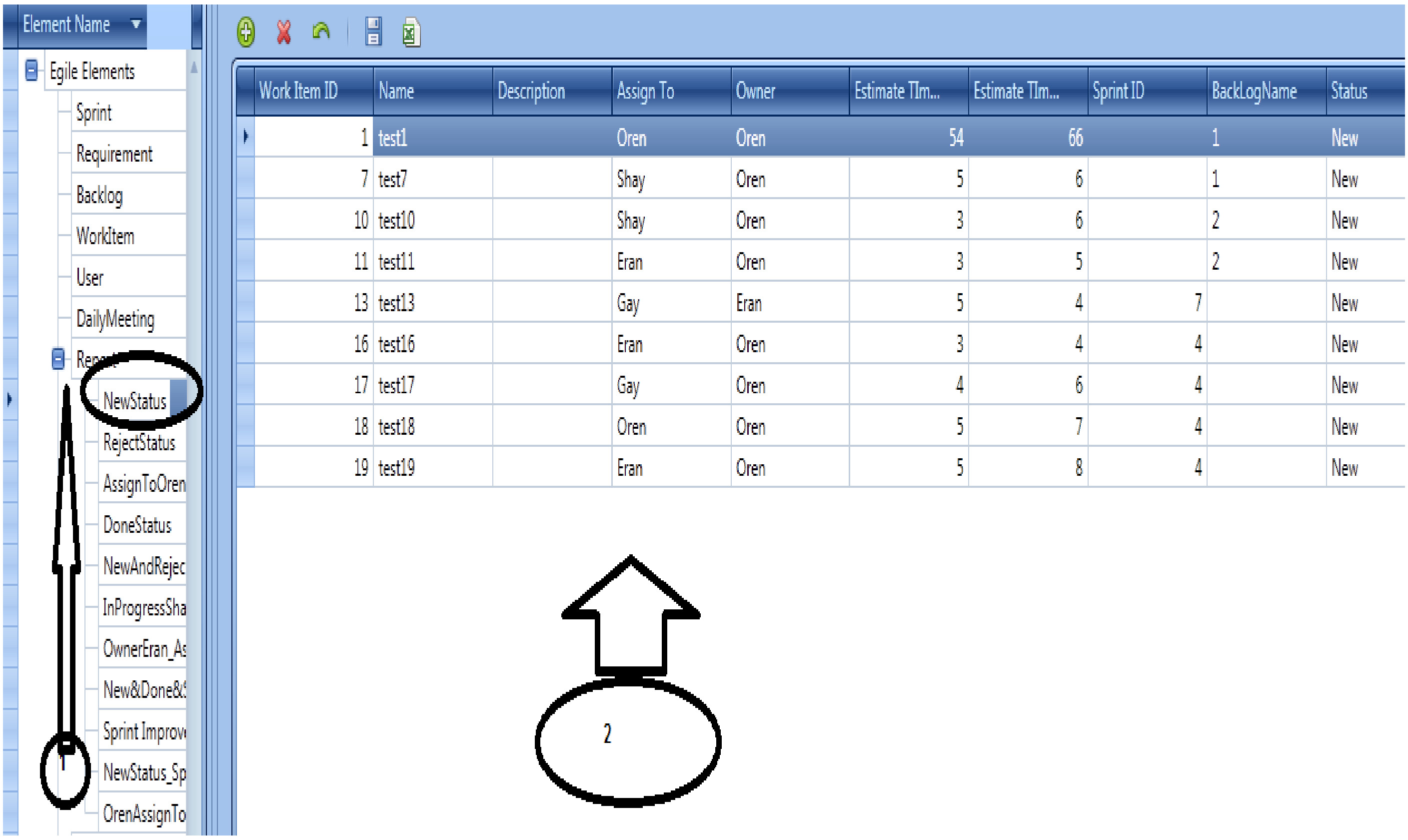
1. לחיצה בעץ האלמנטים על Report יביא אותנו למסך יצירת Report
2. Panel שבו ניתן לבחור איזה Status יכלול ה Report
3. Panel שבו ניתן לבחור מי יצר את המשימות .
4. Panel שבו ניתן לבחור איזה Sprint יכלול ה Report
5. Panel שבו ניתן לראות למי המשימות מוקצות .
6. שמור את השינויים
7. Query Name שם של השאילתא

;

לאחר יצירת ה Report יתוסף עוד דו"ח לעץ , לדוגמא לאחר יצירת שאילתא של כל Task שלהם הם New

יתוסף דוח ראה NewStatus (ראה 1).

לחיצה על דוח זה בעץ יציג לנו את כל WorkItem שבסטטוס New (ראה 2).



Work Item ID	Name	Description	Assign To	Owner	Estimate TIm...	Estimate TIm...	Sprint ID	BackLogName	Status
1	test1		Oren	Oren	54	66		1	New
7	test7		Shay	Oren	5	6		1	New
10	test10		Shay	Oren	3	6		2	New
11	test11		Eran	Oren	3	5		2	New
13	test13		Gay	Eran	5	4	7		New
16	test16		Eran	Oren	3	4	4		New
17	test17		Gay	Oren	4	6	4		New
18	test18		Oren	Oren	5	7	4		New
19	test19		Eran	Oren	5	8	4		New

;

7 סיכום והצעות להמשך מחקר ופיתוח

הפרוייקט נותן מענה לראש צוות /מנהל פיתוח / מפתח , המעונינים באפלקצייה לניהול הפרוייקט במתודלגיה Agile
האפלקצייה מאפשרת לתעד את כל שלביי הפיתוח ע"פ מתודולוגיית Agile כולל מעקב על הפרוייקט.
כמו כן האפלקצייה מותאמת לכל שלביי הפיתוח של המתודלגיה.
האפלקצייה מאפשרת לתעד את דרישות המערכת את הישבות שנערכות ואת התקדמות הפרוייקט .

נושאים לפיתוח עתידי

- דוחות על מצב הכיסוי של הדרישות .
- הוספת אפשרות לטיפול בכמה פרוייקטים במקביל.
- הוספת נעילות בין שתי .Client במידה ומשתמש אחד התחיל לעדכן שהשני לא יוכל במקביל לדרוס לו את השינויים.
- אפשרות ל tractability בין דרישות למשימות שלהם.
- התוכנה תאפשר לשלוח למשתמש הודעה באמצעות דואר אלקטרוני במידה ומשימות לא הושלמו בזמן ,או ש Scrum master רוצה להודיע הודעה כל שהיא למשתמשים.
- המערכת תצא אוטומטית מ login במידה ולא השתמשו באפליקציה 10 דקות.

;

8 נספח

8.1 קוד הפרוייקט

```
//------------------------------------------------------------------------------
-
  Model1.Designer.cs
This file is layer of data acces layer.
//------------------------------------------------------------------------------
-

using System;
using System.Data.Objects;
using System.Data.Objects.DataClasses;
using System.Data.EntityClient;
using System.ComponentModel;
using System.Xml.Serialization;
using System.Runtime.Serialization;

[assembly: EdmSchemaAttribute()]

namespace AgileManagement.DAL
{
    #region Contexts

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    public partial class AgileMngEntities2 : ObjectContext
    {
        #region Constructors

        /// <summary>
        /// Initializes a new AgileMngEntities2 object using the connection
string found in the 'AgileMngEntities2' section of the application
configuration file.
        /// </summary>
        public AgileMngEntities2() : base("name=AgileMngEntities2",
"AgileMngEntities2")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        /// <summary>
        /// Initialize a new AgileMngEntities2 object.
        /// </summary>
        public AgileMngEntities2(string connectionString) :
base(connectionString, "AgileMngEntities2")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        /// <summary>
        /// Initialize a new AgileMngEntities2 object.
        /// </summary>
        public AgileMngEntities2(EntityConnection connection) :
base(connection, "AgileMngEntities2")
        {
```

;

```
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        #endregion

        #region Partial Methods

        partial void OnContextCreated();

        #endregion

        #region ObjectSet Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<Backlog> Backlog
        {
            get
            {
                if ((_Backlog == null))
                {
                    _Backlog = base.CreateObjectSet<Backlog>("Backlog");
                }
                return _Backlog;
            }
        }
        private ObjectSet<Backlog> _Backlog;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<DialyMeeting> DialyMeeting
        {
            get
            {
                if ((_DialyMeeting == null))
                {
                    _DialyMeeting =
base.CreateObjectSet<DialyMeeting>("DialyMeeting");
                }
                return _DialyMeeting;
            }
        }
        private ObjectSet<DialyMeeting> _DialyMeeting;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<Requiments> Requiments
        {
            get
            {
                if ((_Requiments == null))
                {
                    _Requiments =
base.CreateObjectSet<Requiments>("Requiments");
                }
                return _Requiments;
            }
        }
```

;

```
        private ObjectSet<Requiments> _Requiments;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<Sprint> Sprint
        {
            get
            {
                if ((_Sprint == null))
                {
                    _Sprint = base.CreateObjectSet<Sprint>("Sprint");
                }
                return _Sprint;
            }
        }
        private ObjectSet<Sprint> _Sprint;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<UserGroups> UserGroups
        {
            get
            {
                if ((_UserGroups == null))
                {
                    _UserGroups =
base.CreateObjectSet<UserGroups>("UserGroups");
                }
                return _UserGroups;
            }
        }
        private ObjectSet<UserGroups> _UserGroups;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<Users> Users
        {
            get
            {
                if ((_Users == null))
                {
                    _Users = base.CreateObjectSet<Users>("Users");
                }
                return _Users;
            }
        }
        private ObjectSet<Users> _Users;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<WorkItem> WorkItem
        {
            get
            {
                if ((_WorkItem == null))
                {
                    _WorkItem = base.CreateObjectSet<WorkItem>("WorkItem");
                }
```

;

```
                return _WorkItem;
            }
        }
        private ObjectSet<WorkItem> _WorkItem;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<QueryWorkItem> QueryWorkItem
        {
            get
            {
                if ((_QueryWorkItem == null))
                {
                    _QueryWorkItem =
base.CreateObjectSet<QueryWorkItem>("QueryWorkItem");
                }
                return _QueryWorkItem;
            }
        }
        private ObjectSet<QueryWorkItem> _QueryWorkItem;

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        public ObjectSet<sysdiagrams> sysdiagrams
        {
            get
            {
                if ((_sysdiagrams == null))
                {
                    _sysdiagrams =
base.CreateObjectSet<sysdiagrams>("sysdiagrams");
                }
                return _sysdiagrams;
            }
        }
        private ObjectSet<sysdiagrams> _sysdiagrams;

        #endregion
        #region AddTo Methods

        /// <summary>
        /// Deprecated Method for adding a new object to the Backlog EntitySet.
Consider using the .Add method of the associated ObjectSet&lt;T&gt; property
instead.
        /// </summary>
        public void AddToBacklog(Backlog backlog)
        {
            base.AddObject("Backlog", backlog);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the DialyMeeting
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddToDialyMeeting(DialyMeeting dialyMeeting)
        {
            base.AddObject("DialyMeeting", dialyMeeting);
        }
```

;

```
        /// <summary>
        /// Deprecated Method for adding a new object to the Requiments
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddToRequiments(Requiments requiments)
        {
            base.AddObject("Requiments", requiments);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the Sprint EntitySet.
Consider using the .Add method of the associated ObjectSet&lt;T&gt; property
instead.
        /// </summary>
        public void AddToSprint(Sprint sprint)
        {
            base.AddObject("Sprint", sprint);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the UserGroups
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddToUserGroups(UserGroups userGroups)
        {
            base.AddObject("UserGroups", userGroups);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the Users EntitySet.
Consider using the .Add method of the associated ObjectSet&lt;T&gt; property
instead.
        /// </summary>
        public void AddToUsers(Users users)
        {
            base.AddObject("Users", users);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the WorkItem
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddToWorkItem(WorkItem workItem)
        {
            base.AddObject("WorkItem", workItem);
        }

        /// <summary>
        /// Deprecated Method for adding a new object to the QueryWorkItem
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddToQueryWorkItem(QueryWorkItem queryWorkItem)
        {
            base.AddObject("QueryWorkItem", queryWorkItem);
        }

        /// <summary>
```

```
;
        /// Deprecated Method for adding a new object to the sysdiagrams
EntitySet. Consider using the .Add method of the associated ObjectSet&lt;T&gt;
property instead.
        /// </summary>
        public void AddTosysdiagrams(sysdiagrams sysdiagrams)
        {
            base.AddObject("sysdiagrams", sysdiagrams);
        }

        #endregion
    }


    #endregion

    #region Entities

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="Backlog")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Backlog : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new Backlog object.
        /// </summary>
        /// <param name="backlogID">Initial value of the BacklogID
property.</param>
        public static Backlog CreateBacklog(global::System.Int32 backlogID)
        {
            Backlog backlog = new Backlog();
            backlog.BacklogID = backlogID;
            return backlog;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 BacklogID
        {
            get
            {
                return _BacklogID;
            }
            set
            {
                if (_BacklogID != value)
                {
                    OnBacklogIDChanging(value);
                    ReportPropertyChanging("BacklogID");
                    _BacklogID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("BacklogID");
                    OnBacklogIDChanged();
```

```
;
        }
    }
}
private global::System.Int32 _BacklogID;
partial void OnBacklogIDChanging(global::System.Int32 value);
partial void OnBacklogIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> TaskID
{
    get
    {
        return _TaskID;
    }
    set
    {
        OnTaskIDChanging(value);
        ReportPropertyChanging("TaskID");
        _TaskID = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("TaskID");
        OnTaskIDChanged();
    }
}
private Nullable<global::System.Int32> _TaskID;
partial void OnTaskIDChanging(Nullable<global::System.Int32> value);
partial void OnTaskIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> RequimentID
{
    get
    {
        return _RequimentID;
    }
    set
    {
        OnRequimentIDChanging(value);
        ReportPropertyChanging("RequimentID");
        _RequimentID = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("RequimentID");
        OnRequimentIDChanged();
    }
}
private Nullable<global::System.Int32> _RequimentID;
partial void OnRequimentIDChanging(Nullable<global::System.Int32>
value);
partial void OnRequimentIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Name
```

;

```
        {
            get
            {
                return _Name;
            }
            set
            {
                OnNameChanging(value);
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Description
        {
            get
            {
                return _Description;
            }
            set
            {
                OnDescriptionChanging(value);
                ReportPropertyChanging("Description");
                _Description = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Description");
                OnDescriptionChanged();
            }
        }
        private global::System.String _Description;
        partial void OnDescriptionChanging(global::System.String value);
        partial void OnDescriptionChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel",
Name="DialyMeeting")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class DialyMeeting : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new DialyMeeting object.
        /// </summary>
        /// <param name="dialyMeetingID">Initial value of the DialyMeetingID
property.</param>
```

;

```
        public static DialyMeeting CreateDialyMeeting(global::System.Int32
dialyMeetingID)
        {
            DialyMeeting dialyMeeting = new DialyMeeting();
            dialyMeeting.DialyMeetingID = dialyMeetingID;
            return dialyMeeting;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 DialyMeetingID
        {
            get
            {
                return _DialyMeetingID;
            }
            set
            {
                if (_DialyMeetingID != value)
                {
                    OnDialyMeetingIDChanging(value);
                    ReportPropertyChanging("DialyMeetingID");
                    _DialyMeetingID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("DialyMeetingID");
                    OnDialyMeetingIDChanged();
                }
            }
        }
        private global::System.Int32 _DialyMeetingID;
        partial void OnDialyMeetingIDChanging(global::System.Int32 value);
        partial void OnDialyMeetingIDChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Name
        {
            get
            {
                return _Name;
            }
            set
            {
                OnNameChanging(value);
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();
```

;

```
/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Description
{
    get
    {
        return _Description;
    }
    set
    {
        OnDescriptionChanging(value);
        ReportPropertyChanging("Description");
        _Description = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("Description");
        OnDescriptionChanged();
    }
}
private global::System.String _Description;
partial void OnDescriptionChanging(global::System.String value);
partial void OnDescriptionChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> SprintID
{
    get
    {
        return _SprintID;
    }
    set
    {
        OnSprintIDChanging(value);
        ReportPropertyChanging("SprintID");
        _SprintID = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("SprintID");
        OnSprintIDChanged();
    }
}
private Nullable<global::System.Int32> _SprintID;
partial void OnSprintIDChanging(Nullable<global::System.Int32> value);
partial void OnSprintIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Owner
{
    get
    {
        return _Owner;
    }
    set
    {
        OnOwnerChanging(value);
```

;

```
                ReportPropertyChanging("Owner");
                _Owner = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Owner");
                OnOwnerChanged();
            }
        }
        private global::System.String _Owner;
        partial void OnOwnerChanging(global::System.String value);
        partial void OnOwnerChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel",
Name="QueryWorkItem")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class QueryWorkItem : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new QueryWorkItem object.
        /// </summary>
        /// <param name="queryWorkItemID">Initial value of the QueryWorkItemID
property.</param>
        public static QueryWorkItem CreateQueryWorkItem(global::System.Int32
queryWorkItemID)
        {
            QueryWorkItem queryWorkItem = new QueryWorkItem();
            queryWorkItem.QueryWorkItemID = queryWorkItemID;
            return queryWorkItem;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 QueryWorkItemID
        {
            get
            {
                return _QueryWorkItemID;
            }
            set
            {
                if (_QueryWorkItemID != value)
                {
                    OnQueryWorkItemIDChanging(value);
                    ReportPropertyChanging("QueryWorkItemID");
                    _QueryWorkItemID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("QueryWorkItemID");
                    OnQueryWorkItemIDChanged();
                }
```

;

```
        }
    }
    private global::System.Int32 _QueryWorkItemID;
    partial void OnQueryWorkItemIDChanging(global::System.Int32 value);
    partial void OnQueryWorkItemIDChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public Nullable<global::System.Int32> WorkItemID
    {
        get
        {
            return _WorkItemID;
        }
        set
        {
            OnWorkItemIDChanging(value);
            ReportPropertyChanging("WorkItemID");
            _WorkItemID = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("WorkItemID");
            OnWorkItemIDChanged();
        }
    }
    private Nullable<global::System.Int32> _WorkItemID;
    partial void OnWorkItemIDChanging(Nullable<global::System.Int32>
value);
    partial void OnWorkItemIDChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String QueryName
    {
        get
        {
            return _QueryName;
        }
        set
        {
            OnQueryNameChanging(value);
            ReportPropertyChanging("QueryName");
            _QueryName = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("QueryName");
            OnQueryNameChanged();
        }
    }
    private global::System.String _QueryName;
    partial void OnQueryNameChanging(global::System.String value);
    partial void OnQueryNameChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String Name
    {
```

;

```
    get
    {
        return _Name;
    }
    set
    {
        OnNameChanging(value);
        ReportPropertyChanging("Name");
        _Name = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("Name");
        OnNameChanged();
    }
}
private global::System.String _Name;
partial void OnNameChanging(global::System.String value);
partial void OnNameChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Description
{
    get
    {
        return _Description;
    }
    set
    {
        OnDescriptionChanging(value);
        ReportPropertyChanging("Description");
        _Description = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("Description");
        OnDescriptionChanged();
    }
}
private global::System.String _Description;
partial void OnDescriptionChanging(global::System.String value);
partial void OnDescriptionChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String AssignTo
{
    get
    {
        return _AssignTo;
    }
    set
    {
        OnAssignToChanging(value);
        ReportPropertyChanging("AssignTo");
        _AssignTo = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("AssignTo");
        OnAssignToChanged();
    }
}
private global::System.String _AssignTo;
```

```
;
        partial void OnAssignToChanging(global::System.String value);
        partial void OnAssignToChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Owner
        {
            get
            {
                return _Owner;
            }
            set
            {
                OnOwnerChanging(value);
                ReportPropertyChanging("Owner");
                _Owner = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Owner");
                OnOwnerChanged();
            }
        }
        private global::System.String _Owner;
        partial void OnOwnerChanging(global::System.String value);
        partial void OnOwnerChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public Nullable<global::System.Double> EstimateTImeLow
        {
            get
            {
                return _EstimateTImeLow;
            }
            set
            {
                OnEstimateTImeLowChanging(value);
                ReportPropertyChanging("EstimateTImeLow");
                _EstimateTImeLow = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("EstimateTImeLow");
                OnEstimateTImeLowChanged();
            }
        }
        private Nullable<global::System.Double> _EstimateTImeLow;
        partial void OnEstimateTImeLowChanging(Nullable<global::System.Double>
value);
        partial void OnEstimateTImeLowChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public Nullable<global::System.Double> EstimateTImeHigh
        {
            get
            {
                return _EstimateTImeHigh;
```

```
                                                                        ;
            }
            set
            {
                OnEstimateTImeHighChanging(value);
                ReportPropertyChanging("EstimateTImeHigh");
                _EstimateTImeHigh = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("EstimateTImeHigh");
                OnEstimateTImeHighChanged();
            }
        }
        private Nullable<global::System.Double> _EstimateTImeHigh;
        partial void OnEstimateTImeHighChanging(Nullable<global::System.Double>
value);
        partial void OnEstimateTImeHighChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public Nullable<global::System.Int32> Status
        {
            get
            {
                return _Status;
            }
            set
            {
                OnStatusChanging(value);
                ReportPropertyChanging("Status");
                _Status = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("Status");
                OnStatusChanged();
            }
        }
        private Nullable<global::System.Int32> _Status;
        partial void OnStatusChanging(Nullable<global::System.Int32> value);
        partial void OnStatusChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public Nullable<global::System.Int32> Project
        {
            get
            {
                return _Project;
            }
            set
            {
                OnProjectChanging(value);
                ReportPropertyChanging("Project");
                _Project = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("Project");
                OnProjectChanged();
            }
        }
        private Nullable<global::System.Int32> _Project;
        partial void OnProjectChanging(Nullable<global::System.Int32> value);
        partial void OnProjectChanged();
```

;

```
        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public Nullable<global::System.Int32> SprintID
        {
            get
            {
                return _SprintID;
            }
            set
            {
                OnSprintIDChanging(value);
                ReportPropertyChanging("SprintID");
                _SprintID = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("SprintID");
                OnSprintIDChanged();
            }
        }
        private Nullable<global::System.Int32> _SprintID;
        partial void OnSprintIDChanging(Nullable<global::System.Int32> value);
        partial void OnSprintIDChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String BackLogID
        {
            get
            {
                return _BackLogID;
            }
            set
            {
                OnBackLogIDChanging(value);
                ReportPropertyChanging("BackLogID");
                _BackLogID = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("BackLogID");
                OnBackLogIDChanged();
            }
        }
        private global::System.String _BackLogID;
        partial void OnBackLogIDChanging(global::System.String value);
        partial void OnBackLogIDChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="Requiments")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Requiments : EntityObject
    {
        #region Factory Method
```

;

```
        /// <summary>
        /// Create a new Requiments object.
        /// </summary>
        /// <param name="requimentID">Initial value of the RequimentID
property.</param>
        public static Requiments CreateRequiments(global::System.Int32
requimentID)
        {
            Requiments requiments = new Requiments();
            requiments.RequimentID = requimentID;
            return requiments;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 RequimentID
        {
            get
            {
                return _RequimentID;
            }
            set
            {
                if (_RequimentID != value)
                {
                    OnRequimentIDChanging(value);
                    ReportPropertyChanging("RequimentID");
                    _RequimentID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("RequimentID");
                    OnRequimentIDChanged();
                }
            }
        }
        private global::System.Int32 _RequimentID;
        partial void OnRequimentIDChanging(global::System.Int32 value);
        partial void OnRequimentIDChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Name
        {
            get
            {
                return _Name;
            }
            set
            {
                OnNameChanging(value);
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
```

```
;
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Description
        {
            get
            {
                return _Description;
            }
            set
            {
                OnDescriptionChanging(value);
                ReportPropertyChanging("Description");
                _Description = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Description");
                OnDescriptionChanged();
            }
        }
        private global::System.String _Description;
        partial void OnDescriptionChanging(global::System.String value);
        partial void OnDescriptionChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="Sprint")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Sprint : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new Sprint object.
        /// </summary>
        /// <param name="sprintID">Initial value of the SprintID
property.</param>
        public static Sprint CreateSprint(global::System.Int32 sprintID)
        {
            Sprint sprint = new Sprint();
            sprint.SprintID = sprintID;
            return sprint;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
```

;

```
[EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
[DataMemberAttribute()]
public global::System.Int32 SprintID
{
    get
    {
        return _SprintID;
    }
    set
    {
        if (_SprintID != value)
        {
            OnSprintIDChanging(value);
            ReportPropertyChanging("SprintID");
            _SprintID = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("SprintID");
            OnSprintIDChanged();
        }
    }
}
private global::System.Int32 _SprintID;
partial void OnSprintIDChanging(global::System.Int32 value);
partial void OnSprintIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Description
{
    get
    {
        return _Description;
    }
    set
    {
        OnDescriptionChanging(value);
        ReportPropertyChanging("Description");
        _Description = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("Description");
        OnDescriptionChanged();
    }
}
private global::System.String _Description;
partial void OnDescriptionChanging(global::System.String value);
partial void OnDescriptionChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String Name
{
    get
    {
        return _Name;
    }
    set
    {
        OnNameChanging(value);
```

;

```
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="sysdiagrams")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class sysdiagrams : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new sysdiagrams object.
        /// </summary>
        /// <param name="name">Initial value of the name property.</param>
        /// <param name="principal_id">Initial value of the principal_id
property.</param>
        /// <param name="diagram_id">Initial value of the diagram_id
property.</param>
        public static sysdiagrams Createsysdiagrams(global::System.String name,
global::System.Int32 principal_id, global::System.Int32 diagram_id)
        {
            sysdiagrams sysdiagrams = new sysdiagrams();
            sysdiagrams.name = name;
            sysdiagrams.principal_id = principal_id;
            sysdiagrams.diagram_id = diagram_id;
            return sysdiagrams;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.String name
        {
            get
            {
                return _name;
            }
            set
            {
                OnnameChanging(value);
                ReportPropertyChanging("name");
                _name = StructuralObject.SetValidValue(value, false);
                ReportPropertyChanged("name");
```

;

```
        OnnameChanged();
    }
}
private global::System.String _name;
partial void OnnameChanging(global::System.String value);
partial void OnnameChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=false)]
[DataMemberAttribute()]
public global::System.Int32 principal_id
{
    get
    {
        return _principal_id;
    }
    set
    {
        Onprincipal_idChanging(value);
        ReportPropertyChanging("principal_id");
        _principal_id = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("principal_id");
        Onprincipal_idChanged();
    }
}
private global::System.Int32 _principal_id;
partial void Onprincipal_idChanging(global::System.Int32 value);
partial void Onprincipal_idChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
[DataMemberAttribute()]
public global::System.Int32 diagram_id
{
    get
    {
        return _diagram_id;
    }
    set
    {
        if (_diagram_id != value)
        {
            Ondiagram_idChanging(value);
            ReportPropertyChanging("diagram_id");
            _diagram_id = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("diagram_id");
            Ondiagram_idChanged();
        }
    }
}
private global::System.Int32 _diagram_id;
partial void Ondiagram_idChanging(global::System.Int32 value);
partial void Ondiagram_idChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
```

;

```
        [DataMemberAttribute()]
        public Nullable<global::System.Int32> version
        {
            get
            {
                return _version;
            }
            set
            {
                OnversionChanging(value);
                ReportPropertyChanging("version");
                _version = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("version");
                OnversionChanged();
            }
        }
        private Nullable<global::System.Int32> _version;
        partial void OnversionChanging(Nullable<global::System.Int32> value);
        partial void OnversionChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.Byte[] definition
        {
            get
            {
                return StructuralObject.GetValidValue(_definition);
            }
            set
            {
                OndefinitionChanging(value);
                ReportPropertyChanging("definition");
                _definition = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("definition");
                OndefinitionChanged();
            }
        }
        private global::System.Byte[] _definition;
        partial void OndefinitionChanging(global::System.Byte[] value);
        partial void OndefinitionChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="UserGroups")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class UserGroups : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new UserGroups object.
        /// </summary>
```

;

```
        /// <param name="userGroupsID">Initial value of the UserGroupsID
property.</param>
        public static UserGroups CreateUserGroups(global::System.Int32
userGroupsID)
        {
            UserGroups userGroups = new UserGroups();
            userGroups.UserGroupsID = userGroupsID;
            return userGroups;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Name
        {
            get
            {
                return _Name;
            }
            set
            {
                OnNameChanging(value);
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Description
        {
            get
            {
                return _Description;
            }
            set
            {
                OnDescriptionChanging(value);
                ReportPropertyChanging("Description");
                _Description = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Description");
                OnDescriptionChanged();
            }
        }
        private global::System.String _Description;
        partial void OnDescriptionChanging(global::System.String value);
        partial void OnDescriptionChanged();

        /// <summary>
```

;

```
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 UserGroupsID
        {
            get
            {
                return _UserGroupsID;
            }
            set
            {
                if (_UserGroupsID != value)
                {
                    OnUserGroupsIDChanging(value);
                    ReportPropertyChanging("UserGroupsID");
                    _UserGroupsID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("UserGroupsID");
                    OnUserGroupsIDChanged();
                }
            }
        }
        private global::System.Int32 _UserGroupsID;
        partial void OnUserGroupsIDChanging(global::System.Int32 value);
        partial void OnUserGroupsIDChanged();

        #endregion

    }

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="Users")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Users : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new Users object.
        /// </summary>
        /// <param name="usersID">Initial value of the UsersID
property.</param>
        public static Users CreateUsers(global::System.Int32 usersID)
        {
            Users users = new Users();
            users.UsersID = usersID;
            return users;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 UsersID
        {
```

;

```
        get
        {
            return _UsersID;
        }
        set
        {
            if (_UsersID != value)
            {
                OnUsersIDChanging(value);
                ReportPropertyChanging("UsersID");
                _UsersID = StructuralObject.SetValidValue(value);
                ReportPropertyChanged("UsersID");
                OnUsersIDChanged();
            }
        }
    }
    private global::System.Int32 _UsersID;
    partial void OnUsersIDChanging(global::System.Int32 value);
    partial void OnUsersIDChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String Name
    {
        get
        {
            return _Name;
        }
        set
        {
            OnNameChanging(value);
            ReportPropertyChanging("Name");
            _Name = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("Name");
            OnNameChanged();
        }
    }
    private global::System.String _Name;
    partial void OnNameChanging(global::System.String value);
    partial void OnNameChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String UserGroupName
    {
        get
        {
            return _UserGroupName;
        }
        set
        {
            OnUserGroupNameChanging(value);
            ReportPropertyChanging("UserGroupName");
            _UserGroupName = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("UserGroupName");
            OnUserGroupNameChanged();
```

;

```
        }
    }
    private global::System.String _UserGroupName;
    partial void OnUserGroupNameChanging(global::System.String value);
    partial void OnUserGroupNameChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String Description
    {
        get
        {
            return _Description;
        }
        set
        {
            OnDescriptionChanging(value);
            ReportPropertyChanging("Description");
            _Description = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("Description");
            OnDescriptionChanged();
        }
    }
    private global::System.String _Description;
    partial void OnDescriptionChanging(global::System.String value);
    partial void OnDescriptionChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public global::System.String Password
    {
        get
        {
            return _Password;
        }
        set
        {
            OnPasswordChanging(value);
            ReportPropertyChanging("Password");
            _Password = StructuralObject.SetValidValue(value, true);
            ReportPropertyChanged("Password");
            OnPasswordChanged();
        }
    }
    private global::System.String _Password;
    partial void OnPasswordChanging(global::System.String value);
    partial void OnPasswordChanged();

    #endregion

}

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmEntityTypeAttribute(NamespaceName="AgileMngModel", Name="WorkItem")]
```

;

```
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class WorkItem : EntityObject
    {
        #region Factory Method

        /// <summary>
        /// Create a new WorkItem object.
        /// </summary>
        /// <param name="workItemID">Initial value of the WorkItemID
property.</param>
        public static WorkItem CreateWorkItem(global::System.Int32 workItemID)
        {
            WorkItem workItem = new WorkItem();
            workItem.WorkItemID = workItemID;
            return workItem;
        }

        #endregion
        #region Primitive Properties

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 WorkItemID
        {
            get
            {
                return _WorkItemID;
            }
            set
            {
                if (_WorkItemID != value)
                {
                    OnWorkItemIDChanging(value);
                    ReportPropertyChanging("WorkItemID");
                    _WorkItemID = StructuralObject.SetValidValue(value);
                    ReportPropertyChanged("WorkItemID");
                    OnWorkItemIDChanged();
                }
            }
        }
        private global::System.Int32 _WorkItemID;
        partial void OnWorkItemIDChanging(global::System.Int32 value);
        partial void OnWorkItemIDChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Name
        {
            get
            {
                return _Name;
            }
            set
            {
                OnNameChanging(value);
```

;

```
                ReportPropertyChanging("Name");
                _Name = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Name");
                OnNameChanged();
            }
        }
        private global::System.String _Name;
        partial void OnNameChanging(global::System.String value);
        partial void OnNameChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String Description
        {
            get
            {
                return _Description;
            }
            set
            {
                OnDescriptionChanging(value);
                ReportPropertyChanging("Description");
                _Description = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("Description");
                OnDescriptionChanged();
            }
        }
        private global::System.String _Description;
        partial void OnDescriptionChanging(global::System.String value);
        partial void OnDescriptionChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
        [DataMemberAttribute()]
        public global::System.String AssignTo
        {
            get
            {
                return _AssignTo;
            }
            set
            {
                OnAssignToChanging(value);
                ReportPropertyChanging("AssignTo");
                _AssignTo = StructuralObject.SetValidValue(value, true);
                ReportPropertyChanged("AssignTo");
                OnAssignToChanged();
            }
        }
        private global::System.String _AssignTo;
        partial void OnAssignToChanging(global::System.String value);
        partial void OnAssignToChanged();

        /// <summary>
        /// No Metadata Documentation available.
        /// </summary>
        [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
```

;

```
[DataMemberAttribute()]
public global::System.String Owner
{
    get
    {
        return _Owner;
    }
    set
    {
        OnOwnerChanging(value);
        ReportPropertyChanging("Owner");
        _Owner = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("Owner");
        OnOwnerChanged();
    }
}
private global::System.String _Owner;
partial void OnOwnerChanging(global::System.String value);
partial void OnOwnerChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Double> EstimateTImeLow
{
    get
    {
        return _EstimateTImeLow;
    }
    set
    {
        OnEstimateTImeLowChanging(value);
        ReportPropertyChanging("EstimateTImeLow");
        _EstimateTImeLow = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("EstimateTImeLow");
        OnEstimateTImeLowChanged();
    }
}
private Nullable<global::System.Double> _EstimateTImeLow;
partial void OnEstimateTImeLowChanging(Nullable<global::System.Double>
value);
partial void OnEstimateTImeLowChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Double> EstimateTImeHigh
{
    get
    {
        return _EstimateTImeHigh;
    }
    set
    {
        OnEstimateTImeHighChanging(value);
        ReportPropertyChanging("EstimateTImeHigh");
        _EstimateTImeHigh = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("EstimateTImeHigh");
```

```
                                                                    ;
            OnEstimateTImeHighChanged();
        }
    }
    private Nullable<global::System.Double> _EstimateTImeHigh;
    partial void OnEstimateTImeHighChanging(Nullable<global::System.Double>
value);
    partial void OnEstimateTImeHighChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public Nullable<global::System.Int32> Status
    {
        get
        {
            return _Status;
        }
        set
        {
            OnStatusChanging(value);
            ReportPropertyChanging("Status");
            _Status = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("Status");
            OnStatusChanged();
        }
    }
    private Nullable<global::System.Int32> _Status;
    partial void OnStatusChanging(Nullable<global::System.Int32> value);
    partial void OnStatusChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public Nullable<global::System.Int32> Project
    {
        get
        {
            return _Project;
        }
        set
        {
            OnProjectChanging(value);
            ReportPropertyChanging("Project");
            _Project = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("Project");
            OnProjectChanged();
        }
    }
    private Nullable<global::System.Int32> _Project;
    partial void OnProjectChanging(Nullable<global::System.Int32> value);
    partial void OnProjectChanged();

    /// <summary>
    /// No Metadata Documentation available.
    /// </summary>
    [EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
    [DataMemberAttribute()]
    public Nullable<global::System.Int32> SprintID
```

;

```
{
    get
    {
        return _SprintID;
    }
    set
    {
        OnSprintIDChanging(value);
        ReportPropertyChanging("SprintID");
        _SprintID = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("SprintID");
        OnSprintIDChanged();
    }
}
private Nullable<global::System.Int32> _SprintID;
partial void OnSprintIDChanging(Nullable<global::System.Int32> value);
partial void OnSprintIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public global::System.String BackLogID
{
    get
    {
        return _BackLogID;
    }
    set
    {
        OnBackLogIDChanging(value);
        ReportPropertyChanging("BackLogID");
        _BackLogID = StructuralObject.SetValidValue(value, true);
        ReportPropertyChanged("BackLogID");
        OnBackLogIDChanged();
    }
}
private global::System.String _BackLogID;
partial void OnBackLogIDChanging(global::System.String value);
partial void OnBackLogIDChanged();

/// <summary>
/// No Metadata Documentation available.
/// </summary>
[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=true)]
[DataMemberAttribute()]
public Nullable<global::System.Int32> RuquirementID
{
    get
    {
        return _RuquirementID;
    }
    set
    {
        OnRuquirementIDChanging(value);
        ReportPropertyChanging("RuquirementID");
        _RuquirementID = StructuralObject.SetValidValue(value);
        ReportPropertyChanged("RuquirementID");
        OnRuquirementIDChanged();
    }
}
```

;

```
        private Nullable<global::System.Int32> _RuquirementID;
        partial void OnRuquirementIDChanging(Nullable<global::System.Int32> 
value);
        partial void OnRuquirementIDChanged();

        #endregion

    }

    #endregion

}
```

;

```
//----------------------------------------------------------------------------
-
  frmLogin.xaml.cs
This file display the Login screen
//----------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement;
using AgileManagement.DAL;

namespace WPFLoginWindowCS
{
  /// <summary>
  /// Interaction logic for frmLogin.xaml
  /// </summary>
  public partial class frmLogin : Window
  {
      static int _counter = 0;
      static bool _bCancel = false;

      public static bool BCancel
      {
          get { return frmLogin._bCancel; }
          set { frmLogin._bCancel = value; }
      }

      public static int Counter
      {
          get { return frmLogin._counter; }
          set { frmLogin._counter = value; }
      }

     public frmLogin()
    {
      InitializeComponent();
    }

    private void btnCancel_Click(object sender, RoutedEventArgs e)
    {
      DialogResult = false;
      frmLogin.BCancel = true;
      this.Close();
    }

    private void btnLogin_Click(object sender, RoutedEventArgs e)
    {
      // Write code here to authenticate user
      // If authenticated, then set DialogResult=true
        _counter++;
```

;

```
         bool result = this.AuthenticateUserPermmision();
        if (result == true)
        {
            DialogResult = true;
        }
        else
        {
            DialogResult = false;
            if (_counter == 3)
            {
                MessageBox.Show("The user try 3 time therefor the application
will close","Login Error",MessageBoxButton.OK,MessageBoxImage.Error);
                this.Close();
            }
        }
    }

      private bool AuthenticateUserPermmision()
      {
          bool result = false;


          List<Users> userLsts = UserModel.Users;
          Users user=   userLsts.FirstOrDefault(u => u.Name ==
txtUserName.Text);
          if (user != null)
          {
              int resultC = String.Compare(user.Name, txtUserName.Text);
              if (resultC == 0)
              {
                  resultC = String.Compare(user.Password,
this.txtPassword.Password);
                  if (resultC == 0)
                  {
                      UserModel.CurrentUser = user;
                      result = true;

                  }

              }

          }
          else
          {
              MessageBox.Show("Please insert valid user name.", "Error Login ",
MessageBoxButton.OK, MessageBoxImage.Error);
          }

          return result;
      }

   }
}
```

;

```
//-------------------------------------------------------------------------------
-
  frmLogin.xaml
This file contain the UI  definition of Login screen
//-------------------------------------------------------------------------------
-


<Window x:Class="WPFLoginWindowCS.frmLogin"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        WindowStartupLocation="CenterScreen"
        AllowsTransparency="True"
        Background="Transparent"
        WindowStyle="None"
        ShowInTaskbar="False"
        SizeToContent="WidthAndHeight"
        FocusManager.FocusedElement="{Binding ElementName=txtUserName}"
Title="&quot;&quot;">
  <Window.Resources>
    <Style TargetType="Label">
      <Setter Property="Margin"
              Value="4"></Setter>
    </Style>
    <Style TargetType="TextBox">
      <Setter Property="Margin"
              Value="4"></Setter>
      <Setter Property="MinWidth"
              Value="200"></Setter>
      <Setter Property="HorizontalAlignment"
              Value="Left"></Setter>
    </Style>
    <Style TargetType="PasswordBox">
      <Setter Property="Margin"
              Value="4"></Setter>
      <Setter Property="MinWidth"
              Value="200"></Setter>
      <Setter Property="HorizontalAlignment"
              Value="Left"></Setter>
    </Style>
    <Style TargetType="Button">
      <Setter Property="Margin"
              Value="6"></Setter>
      <Setter Property="Padding"
              Value="4"></Setter>
      <Setter Property="MinWidth"
              Value="50"></Setter>
    </Style>
  </Window.Resources>
  <Border CornerRadius="10"
          BorderBrush="Gray"
          BorderThickness="3"
          Background="Beige"
          Margin="24"
          Padding="4">
    <Border.Effect>
      <DropShadowEffect Color="Gray"
                        Opacity=".50"
                        ShadowDepth="16" />
    </Border.Effect>
    <Grid>
      <Grid.ColumnDefinitions>
```

;

```
    <ColumnDefinition Width="60" />
    <ColumnDefinition Width="100" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>
  <Grid.RowDefinitions>
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
    <RowDefinition />
  </Grid.RowDefinitions>
  <StackPanel Grid.Column="0"
              Grid.Row="0"
              Grid.RowSpan="3">
    <Image Name="imgKey"
           Margin="8"
           Source="/Images/Key.jpg">
      <Image.Effect>
        <DropShadowEffect Color="Gray"
                          Opacity=".50"
                          ShadowDepth="8" />
      </Image.Effect>
    </Image>
  </StackPanel>
  <Label Grid.Column="1"
         Grid.Row="0"
         Grid.ColumnSpan="2"
         FontSize="18"
         Margin="10">Please Login To Access This Application</Label>
  <Label Grid.Column="1"
         Grid.Row="1">User Name</Label>
  <TextBox Grid.Column="2"
           Grid.Row="1"
           ToolTip="Enter Your User Name"
           Name="txtUserName" />
  <Label Grid.Column="1"
         Grid.Row="2">Password</Label>
  <PasswordBox Grid.Column="2"
               Grid.Row="2"
               ToolTip="Enter Your Password"
               Name="txtPassword" />
  <StackPanel Grid.Column="2"
              Grid.Row="3"
              Margin="10"
              HorizontalAlignment="Center"
              Orientation="Horizontal">
    <Button Name="btnCancel"
            IsCancel="True"
            Content="Cancel"
            Click="btnCancel_Click">
      <Button.Effect>
        <DropShadowEffect Color="Gray"
                          Opacity=".50"
                          ShadowDepth="8" />
      </Button.Effect>
    </Button>
    <Button Name="btnLogin"
            IsDefault="True"
            Content="Login"
            Click="btnLogin_Click">
      <Button.Effect>
        <DropShadowEffect Color="Gray"
                          Opacity=".50"
```

;

```
                                        ShadowDepth="8" />
          </Button.Effect>
        </Button>
      </StackPanel>
    </Grid>
  </Border>
</Window>
```

;

```
//--------------------------------------------------------------------------------
-
PieStatus.xaml.cs
This class contain the capablity to  display pie report.
//--------------------------------------------------------------------------------
-


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using System.Data.Objects;
using Microsoft.Win32;

using System.ComponentModel;
using System.IO;
namespace PieDonut2DChart
{

    public partial class UserControlPie : UserControl
    {
        public UserControlPie()
        {
            InitializeComponent();
        }

        public void BuildSeriesByStatus()
        {
            AgileMngEntities2 agileMngEntities = new AgileMngEntities2();
           // DevExpress.Xpf.Charts.PieSeries2D item = new
DevExpress.Xpf.Charts.PieSeries2D();
           // item.Name = "Status";
           // this.chartControl1.Diagram.Series.Remove(MypieSeries);

            this.chartControl1.Diagram.Series[0].Points.Clear();


            List<WorkItem> workItemList = agileMngEntities.WorkItem.ToList();

          /*  var categoryCounts = from workItem in workItemList
                        group workItem by workItem.Status into idGroup
                        select new
                        {
                            CustomerID = idGroup.Key,
                            OrderCount = idGroup.Count(),
                            Sales = idGroup
                        };
            */

          /* var categoryCounts = from p in
                                     (from c in workItemList
```

```
                                                                          ;
                                    group c by c.Status)
                               select new { p.Key, ProductCount = p.Count()};

            */


                var query =
                    from address in workItemList
                    group address by address.Status into addressGroup
                    select new
                    {
                        PostalCode = addressGroup.Key,
                        ProductCount = addressGroup.Count(),
                        AddressLine = addressGroup
                    };

                foreach (var addressGroup in query)
                {
                    Console.WriteLine("Postal Code: {0}",
addressGroup.PostalCode);
                    foreach (var address in addressGroup.AddressLine)
                    {
                        Console.WriteLine("\t" + address.StatusName +
                            address.Name);
                        this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(address.StatusName,
addressGroup.ProductCount));
                        break;
                    }
                }


/*
            foreach(var status in  categoryCounts)
            {
               // WorkItem item =workItemList.SingleOrDefault(u => u. ==
status.Key);
             //   status.Sales.
                if (item != null)
                {
                    this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(item.StatusName, status.ProductCount));
                }

            }
 */

          //  item.Points.Add(new DevExpress.Xpf.Charts.SeriesPoint("dddd",
4));



           // this.chartControl1.Diagram.Series.Add(item);
           // this.chartControl1.Diagram.Series.Remove(MypieSeries);
        }


        public void BuildSeriesByUsers()
        {
            AgileMngEntities2 agileMngEntities = new AgileMngEntities2();
            this.chartControl1.Diagram.Series[0].Points.Clear();
```

;

```
            List<WorkItem> workItemList = agileMngEntities.WorkItem.ToList();


            var query =
                from address in workItemList
                group address by address.AssignTo into addressGroup
                select new
                {
                    PostalCode = addressGroup.Key,
                    ProductCount = addressGroup.Count(),
                    AddressLine = addressGroup
                };

            foreach (var addressGroup in query)
            {
                Console.WriteLine("Postal Code: {0}", addressGroup.PostalCode);
                foreach (var address in addressGroup.AddressLine)
                {
                    Console.WriteLine("\t" + address.StatusName +
                        address.Name);
                    this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(address.AssignTo,
addressGroup.ProductCount));
                    break;
                }
            }

        }

    }
}
```

;

```
//-------------------------------------------------------------------------------
-
PieStatus.xaml.cs
This class contain the logic to  display pie report.
//-------------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using System.Data.Objects;
using Microsoft.Win32;

using System.ComponentModel;
using System.IO;
namespace PieDonut2DChart
{

    public partial class UserControlPie : UserControl
    {
        public UserControlPie()
        {
            InitializeComponent();
        }

        public void BuildSeriesByStatus()
        {
            AgileMngEntities2 agileMngEntities = new AgileMngEntities2();
           // DevExpress.Xpf.Charts.PieSeries2D item = new
DevExpress.Xpf.Charts.PieSeries2D();
           // item.Name = "Status";
           // this.chartControl1.Diagram.Series.Remove(MypieSeries);

            this.chartControl1.Diagram.Series[0].Points.Clear();


            List<WorkItem> workItemList = agileMngEntities.WorkItem.ToList();

          /*  var categoryCounts = from workItem in workItemList
                         group workItem by workItem.Status into idGroup
                         select new
                         {
                             CustomerID = idGroup.Key,
                             OrderCount = idGroup.Count(),
                             Sales = idGroup
                         };
            */

          /* var categoryCounts = from p in
                                (from c in workItemList
                                 group c by c.Status)
```

```
                                                                          ;
                             select new { p.Key, ProductCount = p.Count()};

            */


                var query =
                    from address in workItemList
                    group address by address.Status into addressGroup
                    select new
                    {
                        PostalCode = addressGroup.Key,
                        ProductCount = addressGroup.Count(),
                        AddressLine = addressGroup
                    };

                foreach (var addressGroup in query)
                {
                    Console.WriteLine("Postal Code: {0}",
addressGroup.PostalCode);
                    foreach (var address in addressGroup.AddressLine)
                    {
                        Console.WriteLine("\t" + address.StatusName +
                            address.Name);
                        this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(address.StatusName,
addressGroup.ProductCount));
                        break;
                    }
                }


/*
            foreach(var status in  categoryCounts)
            {
               // WorkItem item =workItemList.SingleOrDefault(u => u. ==
status.Key);
             //   status.Sales.
                if (item != null)
                {
                    this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(item.StatusName, status.ProductCount));
                }

            }
 */

          //  item.Points.Add(new DevExpress.Xpf.Charts.SeriesPoint("dddd",
4));



           // this.chartControl1.Diagram.Series.Add(item);
           // this.chartControl1.Diagram.Series.Remove(MypieSeries);
        }


        public void BuildSeriesByUsers()
        {
            AgileMngEntities2 agileMngEntities = new AgileMngEntities2();
            this.chartControl1.Diagram.Series[0].Points.Clear();
```

;

```
            List<WorkItem> workItemList = agileMngEntities.WorkItem.ToList();



            var query =
                from address in workItemList
                group address by address.AssignTo into addressGroup
                select new
                {
                    PostalCode = addressGroup.Key,
                    ProductCount = addressGroup.Count(),
                    AddressLine = addressGroup
                };

            foreach (var addressGroup in query)
            {
                Console.WriteLine("Postal Code: {0}", addressGroup.PostalCode);
                foreach (var address in addressGroup.AddressLine)
                {
                    Console.WriteLine("\t" + address.StatusName +
                        address.Name);
                    this.chartControl1.Diagram.Series[0].Points.Add(new
DevExpress.Xpf.Charts.SeriesPoint(address.AssignTo,
addressGroup.ProductCount));
                    break;
                }
            }

        }

    }
}
```

;

```
//-------------------------------------------------------------------------------
-
BacklogModel.cs
This class access to database  in order to give information on Backlog
//-------------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;

namespace AgileManagement
{
    class BacklogModel

    {

        public static List<Backlog> Get()
        {
            using (AgileMngEntities2 AgileMngEntities2 = new
AgileMngEntities2())
            {
                try
                {
                    return AgileMngEntities2.Backlog.ToList();
                }
                catch (Exception ex)
                {
                    throw;
                }
            }

        }
    }
}
```

```
;
//------------------------------------------------------------------------------
-
BacklogView.xaml.cs
This class contain logic for display  the  BacklogView
//------------------------------------------------------------------------------
-



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class BacklogView : UserControl
    {

        BacklogViewModel _barmodel;
      //  AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<Backlog> _backlogList;

        public BacklogView()
        {
            InitializeComponent();
         //   _requirementList= RequirementModel.Get();
            _barmodel = new BacklogViewModel(ref gridControl1,ref tableView1);
         //   _backlogList = AgileMngEntities2.Backlog.ToList();
          // gridControl1.ItemsSource = AgileMngEntities2.Requiments;
            itemUndo.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_barmodel.itemUndo_ItemClick);
           itemAdd.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_barmodel.itemAdd_ItemClick);
           itemDelete.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_barmodel.itemDelete_ItemClick);

           barButtonItemSave.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_barmodel.barButtonItemSave_ItemClick
);
           barItemAssignWorkItem.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_barmodel.barItemAssignWorkItem_ItemC
lick);

            gridControl1.ItemsSource = _backlogList;
            gridControl1.RefreshData();

        }
```

;

```
        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }



        public void Refresh()
        {
            _barmodel.Refresh();
        }
    }
}
```

;

```
//---------------------------------------------------------------------------
-
BacklogView.xaml
This class contain the definition of UI for BacklogView
//---------------------------------------------------------------------------
-




<UserControl x:Class="AgileManagement.BacklogView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="302" d:DesignWidth="364"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded">
    <Grid>

        <!--<Grid.ColumnDefinitions>
            <ColumnDefinition Width="182*" />
            <ColumnDefinition Width="250*" />
        </Grid.ColumnDefinitions>-->
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True"  Name="barManager1"
Grid.ColumnSpan="2">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png"  />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"  />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png"  />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItemSave"
Glyph="/AgileManagement;component/Images/Save_16x161.png"  />
                <dxb:BarButtonItem Content="Assign workItem"
Name="barItemAssignWorkItem"
Glyph="/AgileManagement;component/Images/Edit_16x161.png"  />
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
```

;

```
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItemSave"
/>
                        <dxb:BarButtonItemLink
BarItemName="barItemAssignWorkItem" />
                    </dxb:Bar.ItemLinks>
                </dxb:Bar>

            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
                    <dxg:GridColumn FieldName="BacklogID"  Name="colID"/>
                    <dxg:GridColumn FieldName="Name" Name="colReq" />
                    <dxg:GridColumn FieldName="Description"
Name="colDescription" />

                </dxg:GridControl.Columns>
                <dxg:GridControl.View>
                    <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
/>
                </dxg:GridControl.View>
            </dxg:GridControl>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</UserControl>
```

;

```
//------------------------------------------------------------------------------
-
BacklogViewModel.cs
This class contain logic for display the backlog  and function as mediator to
database.
//------------------------------------------------------------------------------

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;
using DevExpress.Xpf.Grid;
using System.Windows;

namespace AgileManagement
{
    public class BacklogViewModel
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<Backlog> _backlogList;
        GridControl gridControl1;
        TableView tableView1;
        public BacklogViewModel(ref GridControl gridControl,ref  TableView
tableView)
        {
            _backlogList = AgileMngEntities2.Backlog.ToList();

            gridControl1 = gridControl;
            tableView1 = tableView;
            gridControl1.ItemsSource = _backlogList;
            gridControl1.RefreshData();
        }
        //undo
        public void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _backlogList = BacklogModel.Get();

        }
        public void Refresh()
        {
            AgileMngEntities2 = new AgileMngEntities2();
            this._backlogList = AgileMngEntities2.Backlog.ToList();
            gridControl1.ItemsSource = AgileMngEntities2.Backlog;
            this.gridControl1.RefreshData();
        }

        public void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;
```

;

```
            Backlog backlog = new Backlog();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            backlog.BacklogID = 0;
            if (_backlogList.Count != 0)
            {
                backlog.BacklogID = _backlogList.Max(u => u.BacklogID);
            }

            backlog.BacklogID += 1;
            _backlogList.Add(backlog);
            AgileMngEntities2.AddToBacklog(backlog);
            gridControl1.ItemsSource = _backlogList;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        public void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.Backlog.DeleteObject((Backlog)gridControl1.GetFocusedRow());
            _backlogList.Remove((Backlog)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        public void barItemAssignWorkItem_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            WorkItemSelectView workItemSelectView = new WorkItemSelectView();

            WorkItem workItem = workItemSelectView.ShowDialogw(5);

            int[] listSelect = this.tableView1.GetSelectedRowHandles();
            Backlog backlog = (Backlog)this.gridControl1.GetRow(listSelect[0]);
            workItem.BackLogID = backlog.Name;
            workItem.SprintID = null;

            //Update the WorkItem

            Backlog backlogtTemp = _backlogList.First(u => u.BacklogID ==
backlog.BacklogID);
            WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
            workItemTemp.SprintID = null;
            workItemTemp.BackLogID = backlogtTemp.Name;
        }

        //save
        public void barButtonItemSave_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);
```

```
;
            foreach (Requiments req in AgileMngEntities2.Requiments.ToArray())
            {
                string currentName = req.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in AgileMngEntities2.Requiments.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }
    }
}
```

;

```
//------------------------------------------------------------------------------
-
DialyMeetingView.xaml.cs
This class contain  defintion of  UI and logic to dislay dialy meeting.
//------------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class DialyMeetingView : UserControl
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<DialyMeeting> _dialyMeeting;
        DialyMeetingViewModel _dialyMeetingViewModel;
        public DialyMeetingView()
        {
            InitializeComponent();
            //    _requirementList= RequirementModel.Get();
            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();
            //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
            gridControl1.ItemsSource = _dialyMeeting;

            _dialyMeetingViewModel = new DialyMeetingViewModel(ref
this.gridControl1, ref this.tableView1, ref AgileMngEntities2);
            this.itemAdd.ItemClick+=new
DevExpress.Xpf.Bars.ItemClickEventHandler(itemAdd_ItemClick);
            this.itemDelete.ItemClick+=new
DevExpress.Xpf.Bars.ItemClickEventHandler(itemDelete_ItemClick);
            this.barButtonItem2.ItemClick+=new
DevExpress.Xpf.Bars.ItemClickEventHandler(barButtonItem2_ItemClick);
            this.itemUndo.ItemClick+=new
DevExpress.Xpf.Bars.ItemClickEventHandler(itemUndo_ItemClick);

        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }
```

;

```
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e
            )
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        private void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            DialyMeeting dialyMeeting = new DialyMeeting();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            dialyMeeting.DialyMeetingID = 0;
            if (_dialyMeeting.Count != 0)
            {
                dialyMeeting.DialyMeetingID = _dialyMeeting.Max(u =>
u.DialyMeetingID);
            }

            dialyMeeting.DialyMeetingID += 1;
            dialyMeeting.Owner = UserModel.CurrentUser.Name;
            _dialyMeeting.Add(dialyMeeting);
            AgileMngEntities2.AddToDialyMeeting(dialyMeeting);
            gridControl1.ItemsSource = _dialyMeeting;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.DialyMeeting.DeleteObject((DialyMeeting)gridControl1.GetFocus
edRow());
            _dialyMeeting.Remove((DialyMeeting)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);

            foreach (DialyMeeting dl in
AgileMngEntities2.DialyMeeting.ToArray())
```

;

```
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in
AgileMngEntities2.DialyMeeting.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();

            gridControl1.ItemsSource = AgileMngEntities2.DialyMeeting;

        }

        //Assign task to sprint
        private void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
        //    WorkItemSelectView workItemSelectView = new WorkItemSelectView();
        //    WorkItem workItem = workItemSelectView.ShowDialogw(3);

        //    int[] listSelect = this.tableView1.GetSelectedRowHandles();
        //    Users user = (Sprint)this.gridControl1.GetRow(listSelect[0]);
        //    workItem.SprintID = user.UsersID;


        //    //Update the WorkItem
        //    Sprint sprintTemp = _usersList.First(u => u.UsersID ==
user.SprintID);
        //    WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
        //    workItemTemp.SprintID = user.SprintID;

            //AgileMngEntities2.SaveChanges();

        }

        public void Refresh()
        {
            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();
            this.gridControl1.RefreshData();
        }
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
DialyMeetingView.xaml
This class contain the defintion of UI to dislay dialy meeting.
//--------------------------------------------------------------------------------




<UserControl x:Class="AgileManagement.DialyMeetingView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:AgileManagement"
             mc:Ignorable="d"
             d:DesignHeight="322" d:DesignWidth="432"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxe="http://schemas.devexpress.com/winfx/2008/xaml/editors"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded">
    <Grid>
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True" Margin="12"
Name="barManager1" ItemClick="barManager1_ItemClick">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png"  />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"  />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png" />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"  />
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                       </dxb:Bar.ItemLinks>
                </dxb:Bar>
```

;

```
            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
                    <dxg:GridColumn FieldName="DialyMeetingID"
Name="colDialyMeetingID"/>
                    <dxg:GridColumn FieldName="Name" Name="colName" />
                    <dxg:GridColumn FieldName="SprintID" Name="colSprintID"
                                    EditSettings="{dxe:ComboBoxSettings
DisplayMember= SprintID, ValueMember=SprintID, ItemsSource={x:Static
local:SprintModel.Sprints}}" />
                    <dxg:GridColumn FieldName="Description"
Name="colDescription" />
                    <dxg:GridColumn FieldName="Owner" Name="colOwner"
ReadOnly="True"/>
                </dxg:GridControl.Columns>
                <dxg:GridControl.View>
                    <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
 AutoWidth="True"/>
                </dxg:GridControl.View>
            </dxg:GridControl>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</UserControl>
```

;

```
//-----------------------------------------------------------------------------
-
DialyMeetingViewModel.cs
This class contain logic for display the dialy meeting  and function as
mediator to database.
//-----------------------------------------------------------------------------

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using DevExpress.Xpf.Grid;


namespace AgileManagement
{
    public class DialyMeetingViewModel
    {


        UserViewModel _userViewModel;
        GridControl gridControl1;
        TableView tableView1;
        AgileMngEntities2 AgileMngEntities2;
        private List<DialyMeeting> _dialyMeeting;

        public DialyMeetingViewModel(ref GridControl gridControl, ref
TableView tableView, ref AgileMngEntities2 agileMngEntities)
        {
            AgileMngEntities2 = agileMngEntities;
            gridControl1 = gridControl;
            tableView1 = tableView;

            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();

            gridControl1.ItemsSource = _dialyMeeting;
            gridControl1.RefreshData();
        }

        public void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
```

```
;
            //gridControl1.ItemsSource = _requirementList;

            DialyMeeting dialyMeeting = new DialyMeeting();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            dialyMeeting.DialyMeetingID = 0;
            if (_dialyMeeting.Count != 0)
            {
                dialyMeeting.DialyMeetingID = _dialyMeeting.Max(u =>
u.DialyMeetingID);
            }

            dialyMeeting.DialyMeetingID += 1;
            dialyMeeting.Owner = UserModel.CurrentUser.Name;
            _dialyMeeting.Add(dialyMeeting);
            AgileMngEntities2.AddToDialyMeeting(dialyMeeting);
            gridControl1.ItemsSource = _dialyMeeting;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        public void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.DialyMeeting.DeleteObject((DialyMeeting)gridControl1.GetFocus
edRow());
            _dialyMeeting.Remove((DialyMeeting)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        public  void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);

            foreach (DialyMeeting dl in
AgileMngEntities2.DialyMeeting.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in
AgileMngEntities2.DialyMeeting.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;
```

;

```
                }
            }
            AgileMngEntities2.SaveChanges();
        }

        //undo
        public void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();

            gridControl1.ItemsSource = AgileMngEntities2.DialyMeeting;

        }

        //Assign task to sprint
        public  void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //    WorkItemSelectView workItemSelectView = new
WorkItemSelectView();
            //    WorkItem workItem = workItemSelectView.ShowDialogw(3);

            //    int[] listSelect = this.tableView1.GetSelectedRowHandles();
            //    Users user = (Sprint)this.gridControl1.GetRow(listSelect[0]);
            //    workItem.SprintID = user.UsersID;


            //    //Update the WorkItem
            //    Sprint sprintTemp = _usersList.First(u => u.UsersID ==
user.SprintID);
            //    WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
            //    workItemTemp.SprintID = user.SprintID;

            //AgileMngEntities2.SaveChanges();

        }

        public void Refresh()
        {
            _dialyMeeting = AgileMngEntities2.DialyMeeting.ToList();
            this.gridControl1.RefreshData();
        }
    }
}
```

;

```
//------------------------------------------------------------------------------
-
MainWindows.xaml.cs
This class contain all object of view and function as facad responsible to
transfer  the screen according the user request.

//------------------------------------------------------------------------------

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using DevExpress.Xpf.Grid;
using TreeListControlViewModel;
using WPFLoginWindowCS;
using AgileManagement.DAL;
using PieDonut2DChart;

namespace AgileManagement {
    /// <summary>
    /// Interaction logic for MainWindows.xaml
    /// </summary>
    public partial class MainWindows : Window {

        ListCollectionView _view;
        SprintView _sprintView = new SprintView();
        RequirementView _requirementView = new RequirementView();
        BacklogView _backlogView = new BacklogView();
        WorkItemView _workItemView = new WorkItemView();
        UserView _userView = new UserView();
        DialyMeetingView _dalyMeetingView = new DialyMeetingView();
        ReportView _reportiew;
        UserControlPie _pieControlStat = new UserControlPie();
        UserControlPie _pieControlAssignTo = new UserControlPie();

        TreeListNode _rootNode;
        TreeListNode _childNodeSprint ;
        TreeListNode _childNodeReq;
        TreeListNode _childNodeBacklog;
        TreeListNode _childNodeWorkItem ;
        TreeListNode _childNodeUser;
        TreeListNode _childNodeDialyMeeting;
        TreeListNode _childNodeReport;
        TreeListNode _childNodeReportStatusPie;
        TreeListNode _childNodeReportAssignToPie;

        Dictionary<string, TreeListNode> _treeListReportNode =new
Dictionary<string,TreeListNode>();
        Dictionary<string, QueryWorkItem> _treeListQueuery = new
Dictionary<string, QueryWorkItem>();
```

;

```
public MainWindows()
{
   InitializeComponent();
    List<ProjectObject> list = ProjectObject.CreateTestData();
    _view = new ListCollectionView(list);

    //this.paneToolbox2 = new DevExpress.Xpf.Docking.LayoutPanel();
    TreeListControlDataModel obj = new TreeListControlDataModel(this);
    obj.PopulateDetailPanel(paneToolbox2);
    PopulateControl();

    obj.AttachToTreeListControl(treeListContol);
    BuildTree();
    _reportiew = new ReportView(this);
}


void PopulateControl()
{
    this.paneToolbox2.Content = _sprintView;

}
public void PopulateControl(string elementName)
{
    if (elementName == "Sprint")
    {
        this.paneToolbox2.Content = _sprintView;
    }
    else if (elementName == "Requirement")
    {
        this.paneToolbox2.Content = _requirementView;

    }
    else if (elementName == "Backlog")
    {
        ((BacklogView)_backlogView).Refresh();
        this.paneToolbox2.Content = _backlogView;

    }
    else  if (elementName == "WorkItem")
    {
        ((WorkItemView)_workItemView).Refresh();
        this.paneToolbox2.Content = _workItemView;

    }

    else  if (elementName == "User")
    {
        ((UserView)_userView).Refresh();
        this.paneToolbox2.Content = _userView;

    }
    else  if (elementName == "DailyMeeting")
    {
        ((DialyMeetingView)_dalyMeetingView).Refresh();
        this.paneToolbox2.Content = this._dalyMeetingView;

    }
    else if (elementName == "Report")
```

;

```
{
    ((ReportView)_reportiew).Refresh();
    this.paneToolbox2.Content = this._reportiew;

}
else if(elementName =="Report  Status Pie")
{
    ((UserControlPie)_pieControlStat).BuildSeriesByStatus();
    this.paneToolbox2.Content = this._pieControlStat;
}
else if (elementName == "Report  Assign To Pie")
{
    ((UserControlPie)_pieControlAssignTo).BuildSeriesByUsers();
    this.paneToolbox2.Content = this._pieControlAssignTo;
}
else
{
    if (this._treeListReportNode.ContainsKey(elementName) == true)
    {
        WorkItemView _workItemView = new WorkItemView();
        _workItemView.Queuery(this._treeListQueuery[elementName]);
        this.paneToolbox2.Content = _workItemView;
    }
}


        }



          /*   private void BuildTree()
             {
                 TreeListNode rootNode = CreateRootNode(new ProjectObject() {
ElementName = "Roots" });
                 TreeListNode childNode1 = CreateChildNode(rootNode,  new
ProjectObject() { ElementName = "Child 1" });
                 TreeListNode childNode2 = CreateChildNode(rootNode, new
ProjectObject() { ElementName = "Child 2" });
                 TreeListNode childNode3 = CreateChildNode(rootNode, new
ProjectObject() { ElementName = "Child 3" });
                 TreeListNode childNode4= CreateChildNode(rootNode, new
ProjectObject() { ElementName = "Child 4" });
                 TreeListNode childNode5= CreateChildNode(rootNode, new
ProjectObject() { ElementName = "Child 5" });
                 TreeListNode childNode6 = CreateChildNode(rootNode, new
ProjectObject() { ElementName = "Child 6" });

             }*/

          private void BuildTree()
           {

               _rootNode = CreateRootNode(new ProjectObject() { ElementName =
"Egile Elements" });
               _childNodeSprint = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "Sprint" });
               _childNodeReq = CreateChildNode(_rootNode, new ProjectObject() {
ElementName = "Requirement" });
               _childNodeBacklog = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "Backlog" });
```

;

```
                _childNodeWorkItem = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "WorkItem" });
                _childNodeUser = CreateChildNode(_rootNode, new ProjectObject()
{ ElementName = "User" });
                _childNodeDialyMeeting = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "DailyMeeting" });
                _childNodeReport = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "Report" });
                _childNodeReportStatusPie = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "Report  Status Pie" });
               _childNodeReportStatusPie = CreateChildNode(_rootNode, new
ProjectObject() { ElementName = "Report  Assign To Pie" });

                AgileMngEntities2 agileEnt = new AgileMngEntities2();
                List<QueryWorkItem> queryworkItemLst =
agileEnt.QueryWorkItem.ToList();
                foreach (QueryWorkItem queryWorkItem in queryworkItemLst)
                {
                    TreeListNode treelistNode = new TreeListNode();
                    if (queryWorkItem.Name == "")
                    {
                        treelistNode = CreateChildNode(_childNodeReport, new
ProjectObject() { ElementName = "No Name" });

                    }
                    else
                    {

                        treelistNode = CreateChildNode(_childNodeReport, new
ProjectObject() { ElementName = queryWorkItem.Name });
                        _treeListReportNode.Add(queryWorkItem.Name,
treelistNode);
                        _treeListQueuery.Add(queryWorkItem.Name, queryWorkItem);
                    }
                }

            }

             public void AddQuery(string elementName, QueryWorkItem queryWorkItem)
             {

               TreeListNode treelistNode = CreateChildNode(_childNodeReport, new
ProjectObject() { ElementName = queryWorkItem.Name });
               _treeListReportNode.Add(elementName, treelistNode);
               _treeListQueuery.Add(elementName, queryWorkItem);
            }

            public void Authorized()
             {

               switch (UserModel.CurrentUser.UserGroupName)
               {
                   case "Scrum Master":
                       {


                           break;
                       }

                   case "Devloper":
                       {
                           _rootNode.Nodes.Remove(_childNodeSprint);
```

```
                                                                                          ;
                        _rootNode.Nodes.Remove(_childNodeUser);
                        _rootNode.Nodes.Remove(_childNodeDialyMeeting);
                        _rootNode.Nodes.Remove(_childNodeReport);
                        break;
                    }

                case "Project Owner":
                    {

                        _rootNode.Nodes.Remove(_childNodeSprint);
                        _rootNode.Nodes.Remove(_childNodeUser);
                        _rootNode.Nodes.Remove(_childNodeDialyMeeting);

                        _rootNode.Nodes.Remove(_childNodeBacklog);
                        _rootNode.Nodes.Remove(_childNodeWorkItem);
                        _rootNode.Nodes.Remove(_childNodeReq);

                        break;
                    }


                case "Project Manager":
                    {

                        _rootNode.Nodes.Remove(_childNodeSprint);
                        _rootNode.Nodes.Remove(_childNodeUser);
                        _rootNode.Nodes.Remove(_childNodeDialyMeeting);

                        _rootNode.Nodes.Remove(_childNodeBacklog);
                        _rootNode.Nodes.Remove(_childNodeWorkItem);
                        _rootNode.Nodes.Remove(_childNodeReq);
                        break;
                    }

            }
        }

        private TreeListNode CreateRootNode(object dataObject)
        {
            TreeListNode rootNode = new TreeListNode(dataObject);
            treeListView.Nodes.Add(rootNode);
            return rootNode;
        }
        private TreeListNode CreateChildNode(TreeListNode parentNode,
object dataObject)
        {
            TreeListNode childNode = new TreeListNode(dataObject);
            parentNode.Nodes.Add(childNode);
            return childNode;
        }

        private void paneToolbox2_Initialized(object sender, EventArgs e)
        {

        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            this.Visibility = System.Windows.Visibility.Hidden;
            this.DisplayLoginScreen();
        }
```

;

```
        private void DisplayLoginScreen()
        {
            frmLogin frm = new frmLogin();

            //frm.Owner = this;
            frm.ShowDialog();
            if (frm.DialogResult.HasValue && frm.DialogResult.Value)
            {
                MessageBox.Show("User Logged In");
                this.Visibility = System.Windows.Visibility.Visible;
                Authorized();
            }
            else
            {
                if (frmLogin.Counter == 3|| frmLogin.BCancel ==true)
                {
                    this.Close();
                    frmLogin.Counter = 0;
                    frmLogin.BCancel = false;
                }
                else
                {
                    DisplayLoginScreen();
                }
            }

        }
}


public class StageObject
{
    public String Name { get; set; }
  //  public string Executor { get; set; }
}

public class ProjectObject
{
    public String ElementName { get; set; }
    public string Executor { get; set; }


    public static List<ProjectObject> CreateTestData()
    {
        List<ProjectObject> list = new List<ProjectObject>();
        list.Add(new ProjectObject() { ElementName = "Sprint" });
        list.Add(new ProjectObject() { ElementName = "Requirement" });
        list.Add(new ProjectObject() { ElementName = "WorkItem" });
        list.Add(new ProjectObject() { ElementName = "Child4" });
        list.Add(new ProjectObject() { ElementName = "Child5" });

        return list;

    }

}

public class TaskObject
{
    public String Name { get; set; }
    public string Executor { get; set; }
    public string State { get; set; }
```

```
                                                            ;
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
MainWindows.xaml

This class contain the definition of UI for main scrren.
//--------------------------------------------------------------------------------
-


<!--region #1-->
<Window x:Class="AgileManagement.MainWindows"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="Agile
Management" Height="377" Width="775"
        xmlns:dxdo="http://schemas.devexpress.com/winfx/2008/xaml/docking"
        xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
        xmlns:local="clr-namespace:TreeListControlViewModel"
ShowActivated="True" Loaded="Window_Loaded" WindowState="Maximized"
WindowStartupLocation="CenterScreen">

    <Grid>
        <dxdo:DockLayoutManager Margin="12" Name="dockManager1">
            <dxdo:LayoutGroup Orientation="Horizontal">
                <dxdo:LayoutPanel Caption="Agile Elements" Name="paneToolbox"
Width="200" AllowFloat="False" AllowMinimize="False" AllowMaximize="False">
                    <dxg:TreeListControl Margin="4" Name="treeListContol"
Width="Auto"  >
                        <dxg:TreeListControl.Columns>
                          <dxg:TreeListColumn FieldName="ElementName"
SortIndex="0" SortOrder="Descending" MinWidth="Auto" />
                        </dxg:TreeListControl.Columns>
                        <dxg:TreeListControl.View>
                            <dxg:TreeListView x:Name="treeListView"
AutoWidth="True" AllowEditing="False" KeyFieldName="Id"
ParentFieldName="ParentId"/>
                        </dxg:TreeListControl.View>
                    </dxg:TreeListControl>
                </dxdo:LayoutPanel>
                <dxdo:LayoutPanel Caption="Detail" Name="paneToolbox2"
Initialized="paneToolbox2_Initialized" />

            </dxdo:LayoutGroup>
            <!--Create an auto-hidden pane-->

            <!--Create a floating pane-->
        </dxdo:DockLayoutManager>
    </Grid>
</Window>
<!--endregion #1-->
```

;

```
//--------------------------------------------------------------------------------
-
ReportView.xaml.cs
This class contain logic  and UI for diasplay  report
//--------------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class ReportView : UserControl
    {

        AgileMngEntities2 _agileMngEntities2 = new AgileMngEntities2();
        private List<Backlog> _backlogList;
        private MainWindows _mainWindows;
        private int _status = 0;
        public ReportView(MainWindows mainWondows)
        {
            InitializeComponent();
         //   _requirementList= RequirementModel.Get();
            _backlogList = _agileMngEntities2.Backlog.ToList();
            _mainWindows = mainWondows;
          //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
          //  gridControl1.ItemsSource = _backlogList;

        }
        public void Refresh()
        {
          //  _sprintList = AgileMngEntities2.Sprint.ToList();
          //  this.gridControl1.RefreshData();
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }
        //assign workitem
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e )
        {
```

;

```
        }

        private void itemAdd_ItemClick(object sender, 
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
          //  MessageBox.Show("Item " + e.Item.Content + " has been clicked 
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Backlog backlog = new Backlog();

            // AgileMngEntities2.Requiments.AddObject(req);
           //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            backlog.BacklogID = 0;
            if (_backlogList.Count != 0)
            {
                backlog.BacklogID = _backlogList.Max(u => u.BacklogID);
            }

            backlog.BacklogID += 1;
              _backlogList.Add(backlog);
              _agileMngEntities2.AddToBacklog(backlog);
           //   gridControl1.ItemsSource = _backlogList;
           //  gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
             //AgileMngEntities2.SaveChanges();

        }

        //Delete
        private void itemDelete_ItemClick(object sender, 
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked 
yyyyy.");
          // 
AgileMngEntities2.Backlog.DeleteObject((Backlog)gridControl1.GetFocusedRow());
        //     _backlogList.Remove((Backlog)gridControl1.GetFocusedRow());
          //  gridControl1.RefreshData();
        }

        //save
        private void barButtonItem2_ItemClick(object sender, 
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            CheckWorkItemStatus();
            QueryWorkItem queryWorkItem = new QueryWorkItem();
            queryWorkItem.Status = _status;
            queryWorkItem.AssignTo = textBoxAssignTo.Text;
            queryWorkItem.Owner = textBoxOwner.Text;
            string sprintName=textBoxSprintName.Text;
            if(sprintName==null||sprintName =="")
            {
                queryWorkItem.SprintID = -1;
            }
            else
            {
```

```
;
                Sprint sprint = _agileMngEntities2.Sprint.FirstOrDefault(u =>
u.Name == sprintName);
                if (sprint == null)
                {
                    MessageBox.Show("There is no sprint with name.\n Please Fix
the sprint name,");
                    return;
                }
                else
                {
                    queryWorkItem.SprintID = sprint.SprintID;
                }


            }

            if (this.barEditItemQueryName.EditValue != null)
            {
                queryWorkItem.Name =
this.barEditItemQueryName.EditValue.ToString();
            }
            if (_agileMngEntities2.QueryWorkItem.ToList().Count > 0)
            {
                int maxid = _agileMngEntities2.QueryWorkItem.ToList().Max(u =>
u.QueryWorkItemID);
                queryWorkItem.QueryWorkItemID = maxid + 1;

            }


            foreach (QueryWorkItem dl in
_agileMngEntities2.QueryWorkItem.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in
_agileMngEntities2.QueryWorkItem.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            _mainWindows.AddQuery(queryWorkItem.Name, queryWorkItem);
            _agileMngEntities2.AddToQueryWorkItem(queryWorkItem);
            _agileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            _agileMngEntities2 = new AgileMngEntities2();
            _backlogList = BacklogModel.Get();
```

;

```
            //gridControl1.ItemsSource = AgileMngEntities2.Backlog;

        }

        private void itemclickAssignItem(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
         /*   WorkItemSelectView workItemSelectView = new WorkItemSelectView();

            WorkItem workItem = workItemSelectView.ShowDialogw(5);

          //  int[] listSelect = this.tableView1.GetSelectedRowHandles();
          //  Backlog backlog =
(Backlog)this.gridControl1.GetRow(listSelect[0]);
          //  workItem.BackLogID = backlog.Name;
            workItem.SprintID = null;

            //Update the WorkItem

            Backlog backlogtTemp = _backlogList.First(u => u.BacklogID ==
backlog.BacklogID);
            WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
            workItemTemp.SprintID = null;
            workItemTemp.BackLogID = backlogtTemp.Name;*/
        }

        private void textBox1_TextChanged(object sender, TextChangedEventArgs
e)
        {

        }

        private void checkBox3_Checked(object sender, RoutedEventArgs e)
        {

        }

        private void checkBox3_Checked_1(object sender, RoutedEventArgs e)
        {

        }

        public void CheckWorkItemStatus()
        {
            int status = 0;
            if (this.checkBoxMStatus.IsChecked == true)
            {

                if (this.checkBoxStatusNew.IsChecked == true)
                {
                    status = 1;
                }
                if (this.checkBoxStatusInProgress.IsChecked == true)
                {
                    status= status+10;
                }
                if(this.checkBoxStatusDone.IsChecked == true)
                {
                    status= status+100;
                }
```

;

```
                if (this.checkBoxStatusReject.IsChecked == true)
                {
                    status = status + 1000;
                }
             }
            else
            {
            }
            _status = status;
        }

        private void checkBox99_Checked(object sender, RoutedEventArgs e)
        {

        }
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
ReportView.xaml

This class contain the definition of UI for Report.
//--------------------------------------------------------------------------------
```

```
<UserControl x:Class="AgileManagement.ReportView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="556" d:DesignWidth="885"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded"
xmlns:dxlc="http://schemas.devexpress.com/winfx/2008/xaml/layoutcontrol"
xmlns:dxe="http://schemas.devexpress.com/winfx/2008/xaml/editors"
xmlns:dxdo="http://schemas.devexpress.com/winfx/2008/xaml/docking">
    <Grid>

        <!--<Grid.ColumnDefinitions>
            <ColumnDefinition Width="182*" />
            <ColumnDefinition Width="250*" />
        </Grid.ColumnDefinitions>-->
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True"  Name="barManager1"
ItemClick="barManager1_ItemClick" Margin="0,12,0,0">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png"
ItemClick="itemUndo_ItemClick" />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"
ItemClick="itemAdd_ItemClick" />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png"
ItemClick="itemDelete_ItemClick" />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"
ItemClick="barButtonItem2_ItemClick" />
                <dxb:BarButtonItem Content="Assign workItem"
Name="barItemAssignWorkItem"
Glyph="/AgileManagement;component/Images/Edit_16x161.png"
ItemClick="itemclickAssignItem" />
                <dxb:BarEditItem Content="Query Name"
Name="barEditItemQueryName" IsReadOnly="False">
```

;

```
                    <dxb:BarEditItem.EditSettings>
                        <dxe:TextEditSettings Name="textEditSettings1" />
                    </dxb:BarEditItem.EditSettings>
                </dxb:BarEditItem>
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                        <dxb:BarButtonItemLink
BarItemName="barItemAssignWorkItem" />
                        <dxb:BarEditItemLink BarItemName="barEditItemQueryName"
EditWidth="170" />
                    </dxb:Bar.ItemLinks>
                </dxb:Bar>

            </dxb:BarManager.Bars>
            <dxlc:LayoutGroup Name="layoutGroup1" Height="auto" Width="auto">
                <dxdo:DockLayoutManager Name="dockLayoutManager1" Height="auto"
Width="auto" HorizontalContentAlignment="Stretch"
VerticalContentAlignment="Stretch" UseLayoutRounding="False">
                    <dxdo:DockLayoutManager.LayoutRoot>
                        <dxdo:LayoutGroup Caption="LayoutRoot">
                            <dxdo:LayoutGroup Orientation="Vertical">
                                <dxdo:LayoutPanel Caption="Work Item Status"
ShowCloseButton="False" ShowControlBox="False" ShowPinButton="False">
                                    <dxdo:LayoutGroup Orientation="Vertical"
ShowCloseButton="False" ShowControlBox="False" ShowScrollNextButton="False"
ShowScrollPrevButton="False">
                                        <dxdo:LayoutControlItem >
                                            <CheckBox Content="Enable"
Name="checkBoxMStatus" />
                                        </dxdo:LayoutControlItem>
                                        <dxdo:LayoutControlItem Caption=""
CaptionHorizontalAlignment="Stretch" ShowControlBox="False" ShowControl="True"
ShowCloseButton="False" ShowCaptionImage="False" ShowCaption="False">
                                            <dxlc:GroupBox Header="WorkItem
Status" Name="groupBox1" HorizontalContentAlignment="Stretch"
MinimizationDirection="Vertical" VerticalContentAlignment="Stretch"
MinHeight="200" ForceCursor="True">
                                                <dxlc:LayoutControl
Name="layoutControl1" Orientation="Vertical">
                                                    <CheckBox Height="17"
HorizontalAlignment="Left" Name="checkBoxStatusNew" VerticalAlignment="Top"
Width="125" Content="New" />
                                                    <CheckBox
Content="InProgress" Height="16" HorizontalAlignment="Left"
Name="checkBoxStatusInProgress" VerticalAlignment="Top"/>
```

;

```
                                                    <CheckBox Content="Done"
Height="16" HorizontalAlignment="Left" Name="checkBoxStatusDone"
VerticalAlignment="Top" />
                                                    <CheckBox
Content="Rejected" Height="16" HorizontalAlignment="Left"
Name="checkBoxStatusReject" VerticalAlignment="Top" />
                                                </dxlc:LayoutControl>
                                            </dxlc:GroupBox>
                                        </dxdo:LayoutControlItem>
                                    </dxdo:LayoutGroup>
                                </dxdo:LayoutPanel>
                                <dxdo:LayoutPanel Caption="Assign To"
ShowCloseButton="False" ShowControlBox="False" ShowPinButton="False">
                                    <dxdo:LayoutGroup Orientation="Vertical"
ShowCloseButton="False" ShowControlBox="False" ShowScrollNextButton="False"
ShowScrollPrevButton="False">
                                        <dxdo:LayoutControlItem >
                                            <CheckBox Content="Enable"
Name="checkBoxWorkItem" />
                                        </dxdo:LayoutControlItem>
                                        <dxdo:LayoutControlItem Caption=""
CaptionHorizontalAlignment="Stretch" ShowControlBox="False" ShowControl="True"
ShowCloseButton="False" ShowCaptionImage="False" ShowCaption="False">
                                            <dxlc:GroupBox Header="WorkItem
Assign To" Name="groupBox2" HorizontalContentAlignment="Stretch"
MinimizationDirection="Vertical" VerticalContentAlignment="Stretch"
MinHeight="200">
                                                <dxlc:LayoutControl
Name="layoutControl2">
                                                    <dxlc:LayoutItem
Label="Name" Name="layoutItem1"
dxlc:LayoutControl.AllowHorizontalSizing="True">
                                                        <TextBox
Name="textBoxAssignTo" />
                                                    </dxlc:LayoutItem>
                                                </dxlc:LayoutControl>
                                            </dxlc:GroupBox>
                                        </dxdo:LayoutControlItem>
                                    </dxdo:LayoutGroup>
                                </dxdo:LayoutPanel>
                            </dxdo:LayoutGroup>
                            <dxdo:LayoutGroup Orientation="Vertical">
                                <dxdo:LayoutPanel Caption="Owner"
ShowCloseButton="False" ShowControlBox="False" ShowPinButton="False">
                                    <dxdo:LayoutGroup Orientation="Vertical"
ShowCloseButton="False" ShowControlBox="False" ShowScrollNextButton="False"
ShowScrollPrevButton="False">
                                        <dxdo:LayoutControlItem >
                                            <CheckBox Content="Enable"
Name="checkBoxOwner" />
                                        </dxdo:LayoutControlItem>
                                        <dxdo:LayoutControlItem Caption=""
CaptionHorizontalAlignment="Stretch" ShowControlBox="False" ShowControl="True"
ShowCloseButton="False" ShowCaptionImage="False" ShowCaption="False">
                                            <dxlc:GroupBox Header="Owner` To"
Name="groupBoxOwner" HorizontalContentAlignment="Stretch"
MinimizationDirection="Vertical" VerticalContentAlignment="Stretch"
MinHeight="200">
                                                <dxlc:LayoutControl
Name="layoutControlOwner">
```

;

```
                                                    <dxlc:LayoutItem
Label="Name" Name="layoutItemOwner"
dxlc:LayoutControl.AllowHorizontalSizing="True">
                                                        <TextBox
Name="textBoxOwner" />
                                                    </dxlc:LayoutItem>
                                                </dxlc:LayoutControl>
                                            </dxlc:GroupBox>
                                        </dxdo:LayoutControlItem>
                                    </dxdo:LayoutGroup>
                                </dxdo:LayoutPanel>
                                <dxdo:LayoutPanel Caption="Optional" />
                            </dxdo:LayoutGroup>
                            <dxdo:LayoutGroup Orientation="Vertical">
                                <dxdo:LayoutPanel Caption="Sprint"
ShowCloseButton="False" ShowControlBox="False" ShowPinButton="False">
                                    <dxdo:LayoutGroup Orientation="Vertical"
ShowCloseButton="False" ShowControlBox="False" ShowScrollNextButton="False"
ShowScrollPrevButton="False">
                                        <dxdo:LayoutControlItem >
                                            <CheckBox Content="Enable"
Name="checkBoxSprint" IsChecked="False" />
                                        </dxdo:LayoutControlItem>
                                        <dxdo:LayoutControlItem Caption=""
CaptionHorizontalAlignment="Stretch" ShowControlBox="False" ShowControl="True"
ShowCloseButton="False" ShowCaptionImage="False" ShowCaption="False">
                                            <dxlc:GroupBox Header="Sprint"
Name="groupBox87998" HorizontalContentAlignment="Stretch"
MinimizationDirection="Vertical" VerticalContentAlignment="Stretch"
MinHeight="200">
                                                <dxlc:LayoutControl
Name="layoutControlyy98y">
                                                    <dxlc:LayoutItem
Label="Name" Name="layoutItemtyut"
dxlc:LayoutControl.AllowHorizontalSizing="True">
                                                        <TextBox
Name="textBoxSprintName" />
                                                    </dxlc:LayoutItem>
                                                </dxlc:LayoutControl>
                                            </dxlc:GroupBox>
                                        </dxdo:LayoutControlItem>
                                    </dxdo:LayoutGroup>
                                </dxdo:LayoutPanel>
                                <dxdo:LayoutPanel Caption="Panel" />
                            </dxdo:LayoutGroup>
                        </dxdo:LayoutGroup>
                    </dxdo:DockLayoutManager.LayoutRoot>
                </dxdo:DockLayoutManager>
            </dxlc:LayoutGroup>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</UserControl>
```

```
;
//--------------------------------------------------------------------------------
-
RequirementModel.cs
This class data access  layer and give database service for report
//--------------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;
namespace AgileManagement
{
    public class RequirementModel
    {


        public static List<Requiments> Get()
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {
                    return agileMngEntities.Requiments.ToList();
                }
                catch (Exception ex)
                {
                    throw;
                }
            }

        }
        /*
        public static Categories GetByID(int id)
        {
            using (TestDBEntities testDBContext = new TestDBEntities())
            {
                try
                {
                    return testDBContext.Categories.First(cat => cat.Id == id);
                }
                catch (Exception ex)
                {
                    throw;
                }
            }

        }

        public static Categories GetByName(string name)
        {
            using (TestDBEntities testDBContext = new TestDBEntities())
            {
                try
                {
                    return testDBContext.Categories.First(cat =>
cat.Name.ToLower() == name.ToLower());
                }
                catch (Exception ex)
                {
```

;

```
                    throw;
                }
            }
        }
         * */

        public static int Add(Requiments req)
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {
                     agileMngEntities.AddToRequiments(req);
                    agileMngEntities.SaveChanges();

                    return 1;
                }
                catch (Exception ex)
                {
                    return 0;
                }
            }
        }

        public static void UpdateChanges(List<Requiments> requirementList)
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {

                    foreach (Requiments req in requirementList)
                   {

                         Requiments requimentTmp = new Requiments();
                         requimentTmp = (from c in agileMngEntities.Requiments
where c.RequimentID == req.RequimentID select c).First();
                         requimentTmp.Name = req.Name;
                         agileMngEntities.SaveChanges();
                   }
                }
                catch (Exception ex)
                {

                }
            }
        }

       /*
        public static int Update(int id, string newName)
        {
            using (TestDBEntities testDBContext = new TestDBEntities())
            {
                try
                {
                    Categories catEdit = new Categories();
                    catEdit = (from c in testDBContext.Categories where c.Id ==
id select c).First();
```

;

```
                    catEdit.Name = newName;

                    testDBContext.SaveChanges();

                    return 1;
                }
                catch (Exception ex)
                {
                    return 0;
                }
            }
        }
        public static int Delete(int id)
        {
            using (TestDBEntities testDBContext = new TestDBEntities())
            {
                try
                {
                    Categories catEdit = new Categories();
                    catEdit = (from c in testDBContext.Categories where c.Id ==
id select c).First();

                    testDBContext.DeleteObject(catEdit);
                    testDBContext.SaveChanges();

                    return 1;
                }
                catch (Exception ex)
                {
                    return 0;
                }
            }
        }
        */


    }
}
```

;

```
//------------------------------------------------------------------------------
-
BacklogView.xaml
This class contain the definition of UI and logic  for Requirment screen
//------------------------------------------------------------------------------
-


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class RequirementView : UserControl
    {

        AgileMngEntities2 _agileMngEntities2 = new AgileMngEntities2();
        private List<Requiments> _requirementList;
        RequirementViewModel _requirementViewModel;

        public RequirementView()
        {
            InitializeComponent();
         //   _requirementList= RequirementModel.Get();
            _requirementList = _agileMngEntities2.Requiments.ToList();
          //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
            gridControl1.ItemsSource = _requirementList;
            _requirementViewModel = new RequirementViewModel(ref
this.gridControl1,ref this.tableView1,ref _agileMngEntities2);
            this.itemAdd.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_requirementViewModel.itemAdd_ItemCli
ck);
            this.itemDelete.ItemClick+=new
DevExpress.Xpf.Bars.ItemClickEventHandler(_requirementViewModel.itemDelete_Item
Click);
            this.barButtonItem2.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_requirementViewModel.barButtonItem2_
ItemClick);
            this.itemUndo.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_requirementViewModel.itemUndo_ItemCl
ick);


        }
```

```
;
        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e  )
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        private void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
          //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Requiments req = new Requiments();

            // AgileMngEntities2.Requiments.AddObject(req);
           //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();

              req.RequimentID = _requirementList.Max(u => u.RequimentID);
              req.RequimentID += 1;
              _requirementList.Add(req);
              _agileMngEntities2.AddToRequiments(req);
              gridControl1.ItemsSource = _requirementList;
             gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
             //AgileMngEntities2.SaveChanges();


        }

        //Delete
        private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

_agileMngEntities2.Requiments.DeleteObject((Requiments)gridControl1.GetFocusedR
ow());
            _requirementList.Remove((Requiments)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
           // RequirementModel.UpdateChanges(_requirementList);
            foreach (Requiments dl in _agileMngEntities2.Requiments.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
```

;

```
                int count = (from spr in
_agileMngEntities2.Requiments.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            _agileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            _agileMngEntities2 = new AgileMngEntities2();
            _requirementList = RequirementModel.Get();

            gridControl1.ItemsSource = _agileMngEntities2.Requiments;

        }
    }
}
```

;

```
//---------------------------------------------------------------------------------
-
BacklogView.xaml
This class contain definition of UI for Requiment
//---------------------------------------------------------------------------------
-



<UserControl x:Class="AgileManagement.RequirementView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="322" d:DesignWidth="432"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded">
    <Grid>
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True" Margin="12"
Name="barManager1" ItemClick="barManager1_ItemClick">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png" />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"  />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png"  />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"  />
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                    </dxb:Bar.ItemLinks>
                </dxb:Bar>

            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
```

;

```
                        <dxg:GridColumn FieldName="RequimentID"  Name="colID"/>
                        <dxg:GridColumn FieldName="Name" Name="colReq" />
                        <dxg:GridColumn FieldName="Description"
Name="colDescription" Width="700"  />

                    </dxg:GridControl.Columns>
                    <dxg:GridControl.View>
                        <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
/>
                    </dxg:GridControl.View>
                </dxg:GridControl>
            </dxb:BarManager>
            <!--endregion #1-->
        </Grid>
</UserControl>
```

;

```
//--------------------------------------------------------------------------------
-
RequirementViewModel.cs
This class contain logic for display the Requirment   and function as mediator
to database
//--------------------------------------------------------------------------------




using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;
using DevExpress.Xpf.Grid;
using System.Windows;

namespace AgileManagement
{
    public class RequirementViewModel
    {


        AgileMngEntities2 _agileMngEntities2;
        private List<Requiments> _requirementList;
        GridControl gridControl1;
        TableView tableView1;


        public RequirementViewModel(ref GridControl gridControl, ref  TableView
tableView, ref AgileMngEntities2 agileMngEntities2)
        {
            _agileMngEntities2 = agileMngEntities2;
            _requirementList = _agileMngEntities2.Requiments.ToList();
            gridControl1 = gridControl;
            tableView1 = tableView;
            gridControl1.ItemsSource = _requirementList;
            gridControl1.RefreshData();

        }



        public void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Requiments req = new Requiments();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
```

;

```
                req.RequimentID = _requirementList.Max(u => u.RequimentID);
                req.RequimentID += 1;
                _requirementList.Add(req);
                _agileMngEntities2.AddToRequiments(req);
                gridControl1.ItemsSource = _requirementList;
                gridControl1.RefreshData();
                // AgileMngEntities2.Requiments.Attach(req);
                //AgileMngEntities2.SaveChanges();


            }

            //Delete
            public void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

_agileMngEntities2.Requiments.DeleteObject((Requiments)gridControl1.GetFocusedR
ow());
                _requirementList.Remove((Requiments)gridControl1.GetFocusedRow());
                gridControl1.RefreshData();
            }

            //save
            public void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                // RequirementModel.UpdateChanges(_requirementList);
                foreach (Requiments dl in _agileMngEntities2.Requiments.ToArray())
                {
                    string currentName = dl.Name;
                    // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                    int count = (from spr in
_agileMngEntities2.Requiments.ToArray()
                                 where spr.Name == currentName
                                 select spr).Count();

                    if (count > 1)
                    {
                        MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                        return;

                    }
                }
                _agileMngEntities2.SaveChanges();
            }

            //undo
            public void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
            {
                _agileMngEntities2 = new AgileMngEntities2();
                _requirementList = RequirementModel.Get();

                gridControl1.ItemsSource = _agileMngEntities2.Requiments;
```

```
                                                                            ;
        }
    }
}
```

;

```
//-----------------------------------------------------------------------------
-
RequirementModel.cs
This class data access layer and give database service for Requirment
//-----------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;

namespace AgileManagement
{
    class SprintModel
    {

        public static List<Sprint> Get()
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {
                    return agileMngEntities.Sprint.ToList();
                }
                catch (Exception ex)
                {
                    throw;
                }
            }

        }

        public static List<Sprint> Sprints
        {
            get
            {
                using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
                {
                    try
                    {
                        return agileMngEntities.Sprint.ToList();
                    }
                    catch (Exception ex)
                    {
                        throw;
                    }
                }
            }
        }


        public static List<WorkItemStatus> Status
        {

            get
```

;

```
            {

                try
                {
                    return  WorkItemStatus.GetWorItemStatus() ;
                }
                catch (Exception ex)
                {
                    throw;
                }

            }
        }
    }
}
```

;

```
//---------------------------------------------------------------------------
-
SprintView.xaml.cs
This class contain logic for display  the  Sprint
//---------------------------------------------------------------------------
-

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class SprintView : UserControl
    {

        SprintViewModel _sprintViewModel;


        public SprintView()
        {
            InitializeComponent();

            _sprintViewModel = new SprintViewModel(ref gridControl1,  ref
tableView1);
            this.itemAdd.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_sprintViewModel.itemAdd_ItemClick);
            this.itemDelete.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_sprintViewModel.itemDelete_ItemClick
);
            this.barButtonItem2.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_sprintViewModel.barButtonItem2_ItemC
lick);
            this.itemUndo.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_sprintViewModel.itemUndo_ItemClick);
            this.barButtonItem3.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_sprintViewModel.barButtonItem3_ItemC
lick);

            //   _requirementList= RequirementModel.Get();
            // _sprintList = _agileMngEntities2.Sprint.ToList();
            //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
          //  gridControl1.ItemsSource = _sprintList;

        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
```

;

```
        }
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e
            )
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        //private void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        //{
        //    //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
 xxxxx.");
        //    //  Requiments req = new Requiments();
        //    //RequirementModel.Add(req);

        //    //_requirementList = RequirementModel.Get();
        //    //gridControl1.ItemsSource = _requirementList;

        //    Sprint sprint = new Sprint();

        //    // AgileMngEntities2.Requiments.AddObject(req);
        //    //  AgileMngEntities2.AddToRequiments(req);
        //    // AgileMngEntities2.SaveChanges();
        //    sprint.SprintID = 0;
        //    if (_sprintList.Count != 0)
        //    {
        //        sprint.SprintID = _sprintList.Max(u => u.SprintID);
        //    }

        //    sprint.SprintID += 1;
        //    _sprintList.Add(sprint);
        //    _agileMngEntities2.AddToSprint(sprint);
        //    gridControl1.ItemsSource = _sprintList;
        //    gridControl1.RefreshData();
        //    // AgileMngEntities2.Requiments.Attach(req);
        //    //AgileMngEntities2.SaveChanges();

        //}

        ////Delete
        //private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        //{
        //    MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");
        //
_agileMngEntities2.Sprint.DeleteObject((Sprint)gridControl1.GetFocusedRow());
        //    _sprintList.Remove((Sprint)gridControl1.GetFocusedRow());
        //    gridControl1.RefreshData();
        //}

        ////save
        //private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        //{
        //    try
        //    {
        //        // RequirementModel.UpdateChanges(_requirementList);
```

;

```
        //        foreach (Sprint sprint in
_agileMngEntities2.Sprint.ToArray())
        //        {
        //            string currentName = sprint.Name;
        //            // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
        //            int count = (from spr in
_agileMngEntities2.Sprint.ToArray()
        //                         where spr.Name == currentName
        //                         select spr).Count();

        //            if (count > 1)
        //            {
        //                MessageBox.Show("There is Name that is not unique
therefor is not save.\n The name :" + currentName, "Error Saving",
MessageBoxButton.OK, MessageBoxImage.Error);
        //                return;

        //            }
        //        }

        //        _agileMngEntities2.SaveChanges();
        //        /*_agileMngEntities2 = new AgileMngEntities2();
        //        _sprintList = _agileMngEntities2.Sprint.ToList();
        //         gridControl1.ItemsSource = _sprintList;*/
        //    }
        //    catch (Exception ex)
        //    {
        //        MessageBox.Show(ex.Message);
        //    }
        //}

        ////undo
        //private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        //{
        //    _agileMngEntities2 = new AgileMngEntities2();
        //    _sprintList = SprintModel.Get();

        //    gridControl1.ItemsSource = _agileMngEntities2.Backlog;

        //}

        ////Assign task to sprint
        //private void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        //{
        //    WorkItemSelectView workItemSelectView = new WorkItemSelectView();
        //    WorkItem workItem = workItemSelectView.ShowDialogw(3);

        //    int[] listSelect = this.tableView1.GetSelectedRowHandles();
        //    Sprint sprint = (Sprint)this.gridControl1.GetRow(listSelect[0]);
        //    workItem.SprintID = sprint.SprintID;


        //    //Update the WorkItem
        //    Sprint sprintTemp = _sprintList.First(u => u.SprintID ==
sprint.SprintID);
        //    WorkItem workItemTemp = _agileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
        //    workItemTemp.SprintID = sprint.SprintID;
```

;

```
        //    //AgileMngEntities2.SaveChanges();

        //}


        public void Refresh()
        {
            _sprintViewModel.Refresh();
        }
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
SprintView.xaml
This class contain the definition of UI for SprintView
//--------------------------------------------------------------------------------
-



<UserControl x:Class="AgileManagement.SprintView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="322" d:DesignWidth="432"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded">
    <Grid>
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True" Margin="12"
Name="barManager1" ItemClick="barManager1_ItemClick">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png"  />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"  />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png"  />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"  />
                <dxb:BarButtonItem Content="Assign WorkItem to Sprint"
Name="barButtonItem3" Glyph="/AgileManagement;component/Images/Edit_16x161.png"
/>
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem3" />
                    </dxb:Bar.ItemLinks>
                </dxb:Bar>
```

;

```
            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
                    <dxg:GridColumn FieldName="SprintID"  Name="colID"/>
                    <dxg:GridColumn FieldName="Name" Name="colName" />
                    <dxg:GridColumn FieldName="Description"
Name="colDescription" />

                </dxg:GridControl.Columns>
                <dxg:GridControl.View>
                    <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
/>
                </dxg:GridControl.View>
            </dxg:GridControl>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</UserControl>
```

;

```
//---------------------------------------------------------------------------
-
SprintViewModel.cs
This class contain logic for display the Sprint  and function as mediator to
database.
//---------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;
using DevExpress.Xpf.Grid;
using System.Windows;


namespace AgileManagement
{
   public  class SprintViewModel
    {


       AgileMngEntities2 _agileMngEntities2 = new AgileMngEntities2();
        private List<Sprint> _sprintList;
        GridControl gridControl1;
        TableView tableView1;
        public SprintViewModel(ref GridControl gridControl, ref  TableView
tableView)
        {
            _sprintList = _agileMngEntities2.Sprint.ToList();

            gridControl1 = gridControl;
            tableView1 = tableView;
            gridControl1.ItemsSource = _sprintList;
            gridControl1.RefreshData();
        }


        public void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Sprint sprint = new Sprint();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            sprint.SprintID = 0;
            if (_sprintList.Count != 0)
            {
                sprint.SprintID = _sprintList.Max(u => u.SprintID);
            }
```

;

```
            sprint.SprintID += 1;
            _sprintList.Add(sprint);
            _agileMngEntities2.AddToSprint(sprint);
            gridControl1.ItemsSource = _sprintList;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        public void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {


_agileMngEntities2.Sprint.DeleteObject((Sprint)gridControl1.GetFocusedRow());
            _sprintList.Remove((Sprint)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
            MessageBox.Show("Item " + e.Item.Content + " has been deleted.");
        }

        //save
        public void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            try
            {
                // RequirementModel.UpdateChanges(_requirementList);

                foreach (Sprint sprint in _agileMngEntities2.Sprint.ToArray())
                {
                    string currentName = sprint.Name;
                    // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                    int count = (from spr in
_agileMngEntities2.Sprint.ToArray()
                                 where spr.Name == currentName
                                 select spr).Count();

                    if (count > 1)
                    {
                        MessageBox.Show("There is Name that is not unique
therefor is not save.\n The name :" + currentName, "Error Saving",
MessageBoxButton.OK, MessageBoxImage.Error);
                        return;

                    }
                }

                _agileMngEntities2.SaveChanges();
                /*_agileMngEntities2 = new AgileMngEntities2();
                _sprintList = _agileMngEntities2.Sprint.ToList();
                 gridControl1.ItemsSource = _sprintList;*/
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

;

```
        //undo
        public void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            _agileMngEntities2 = new AgileMngEntities2();
            _sprintList = SprintModel.Get();

            gridControl1.ItemsSource = _agileMngEntities2.Sprint;

        }

        //Assign task to sprint
        public void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            WorkItemSelectView workItemSelectView = new WorkItemSelectView();
            WorkItem workItem = workItemSelectView.ShowDialogw(3);

            int[] listSelect = this.tableView1.GetSelectedRowHandles();
            Sprint sprint = (Sprint)this.gridControl1.GetRow(listSelect[0]);
            workItem.SprintID = sprint.SprintID;


            //Update the WorkItem
            Sprint sprintTemp = _sprintList.First(u => u.SprintID ==
sprint.SprintID);
            WorkItem workItemTemp = _agileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
            workItemTemp.SprintID = sprint.SprintID;

            //AgileMngEntities2.SaveChanges();

        }
        public void Refresh()
        {
            _sprintList = _agileMngEntities2.Sprint.ToList();
            this.gridControl1.RefreshData();
        }
    }
}
```

;

```
//------------------------------------------------------------------------------
-
TreeListControlDataModel.cs
This class contain logic to navigation between  a screen.
//------------------------------------------------------------------------------



using System;
using System.Windows;
using System.Windows.Data;
using System.ComponentModel;
using System.Collections.Specialized;
using DevExpress.Data.Filtering;
using DevExpress.Xpf.Core;
using DevExpress.Xpf.Grid;
using DevExpress.Xpf.Grid.TreeList;
using AgileManagement;

namespace TreeListControlViewModel
{
    public enum ModelFilterMode
    {
        CollectionViewFilterPredicate,
        FilterCriteria,
    }
    public class TreeListControlDataModel
    {
        #region static
        public static readonly DependencyProperty
TreeListControlDataModelProperty;
        public static readonly DependencyProperty
IsSynchronizedWithCurrentItemProperty;
        public static readonly DependencyProperty CollectionViewProperty;
        public static readonly DependencyProperty PanelViewProperty;


        public static TreeListControlDataModel
GetTreeListControlDataModel(TreeListControl treeList)
        {
            return
(TreeListControlDataModel)treeList.GetValue(TreeListControlDataModelProperty);
        }
        public static void SetTreeListControlDataModel(TreeListControl
treeListControl, TreeListControlDataModel value)
        {
            treeListControl.SetValue(TreeListControlDataModelProperty, value);
        }
        public static bool GetIsSynchronizedWithCurrentItem(TreeListControl
treeList)
        {
            return
(bool)treeList.GetValue(IsSynchronizedWithCurrentItemProperty);
        }
        public static void SetIsSynchronizedWithCurrentItem(TreeListControl
treeList, bool value)
        {
            treeList.SetValue(IsSynchronizedWithCurrentItemProperty, value);
```

```
;
        }
        public static ICollectionView GetCollectionView(TreeListControl 
treeList)
        {
            return (ICollectionView)treeList.GetValue(CollectionViewProperty);
        }
        public static void SetCollectionView(TreeListControl treeList, 
ICollectionView value)
        {
            treeList.SetValue(CollectionViewProperty, value);
        }

        static TreeListControlDataModel()
        {
            TreeListControlDataModelProperty = 
DependencyProperty.RegisterAttached("TreeListControlDataModel", 
typeof(TreeListControlDataModel), typeof(TreeListControlDataModel), new 
UIPropertyMetadata(null, OnTreeListControlDataModelChanged));

            IsSynchronizedWithCurrentItemProperty = 
DependencyProperty.RegisterAttached("IsSynchronizedWithCurrentItem", 
typeof(bool), typeof(TreeListControlDataModel), new UIPropertyMetadata(true));
            CollectionViewProperty = 
DependencyProperty.RegisterAttached("CollectionView", typeof(ICollectionView), 
typeof(TreeListControlDataModel), new UIPropertyMetadata(null, 
OnCollectionViewChanged));
           // PanelViewProperty = 
DependencyProperty.RegisterAttached("CollectionView", typeof(ICollectionView), 
typeof(TreeListControlDataModel), new UIPropertyMetadata(null, 
OnCollectionViewChanged));


        }
        static void OnTreeListControlDataModelChanged(DependencyObject d, 
DependencyPropertyChangedEventArgs e)
        {
            TreeListControl treeListControl = (TreeListControl)d;
            TreeListControlDataModel model = 
(TreeListControlDataModel)e.NewValue;
            if (model != null)
                model.AttachToTreeListControl(treeListControl);
        }
        static void OnCollectionViewChanged(DependencyObject d, 
DependencyPropertyChangedEventArgs e)
        {
            TreeListControl treeListControl = (TreeListControl)d;
            ICollectionView view = (ICollectionView)e.NewValue;

            SetTreeListControlDataModel(treeListControl, new 
TreeListControlDataModel() { CollectionView = view });
        }


        #endregion
        ICollectionView collectionView;
        TreeListControl treeListControl;
        Locker syncGroupSortLocker = new Locker();
        ModelFilterMode filterMode;
        MainWindows _mainWindows;
        CriteriaOperator filterCriteria;

        public TreeListControlDataModel()
```

;

```
        {
            AutoPopulateColumns = true;
        }
        public TreeListControlDataModel(MainWindows mainWindows)
        {
            AutoPopulateColumns = true;
            this._mainWindows = mainWindows;
        }
        public bool AutoPopulateColumns { get; set; }
        public ModelFilterMode FilterMode
        {
            get { return filterMode; }
            set
            {
                if (filterMode == value)
                    return;
                filterMode = value;
                SyncFilter();
                if (treeListControl != null)
                    treeListControl.RefreshData();
            }
        }
        public CriteriaOperator FilterCriteria
        {
            get { return filterCriteria; }
            set
            {
                if (object.ReferenceEquals(filterCriteria, value))
                    return;
                filterCriteria = value;
                SyncFilter();
            }
        }
        DevExpress.Xpf.Docking.LayoutPanel m_paneToolbox2 = null;
        public ICollectionView CollectionView
        {
            get { return collectionView; }
            set
            {
                if (collectionView == value)
                    return;
                if (collectionView != null)
                {
                    INotifyPropertyChanged notifyPropertyChanged =
collectionView as INotifyPropertyChanged;
                    if (notifyPropertyChanged != null)
                        notifyPropertyChanged.PropertyChanged -= new
PropertyChangedEventHandler(OnCollectionViewPropertyChanged);
                    if (collectionView.SortDescriptions != null)

((INotifyCollectionChanged)collectionView.SortDescriptions).CollectionChanged -
= new NotifyCollectionChangedEventHandler(OnSortDescriptionsCollectionChanged);
                    collectionView.CurrentChanged -= new
EventHandler(OnCollectionViewCurrentChanged);
                }
                collectionView = value;
                if (collectionView != null)
                {
                    INotifyPropertyChanged notifyPropertyChanged =
collectionView as INotifyPropertyChanged;
                    if (notifyPropertyChanged != null)
```

```
;
                        notifyPropertyChanged.PropertyChanged += new
PropertyChangedEventHandler(OnCollectionViewPropertyChanged);
                    if (collectionView.SortDescriptions != null)

((INotifyCollectionChanged)collectionView.SortDescriptions).CollectionChanged
+= new
NotifyCollectionChangedEventHandler(OnSortDescriptionsCollectionChanged);
                    collectionView.CurrentChanged += new
EventHandler(OnCollectionViewCurrentChanged);
                }
            }
        }

        void OnCollectionViewCurrentChanged(object sender, EventArgs e)
        {
            SyncFocusedRowHandle();
        }
        void SyncFocusedRowHandle()
        {
            if (CanSyncCurrentRow())
                treeListControl.View.FocusedRow = collectionView.CurrentItem;
        }
        void OnTreeListControlFocusedRowChanged(object sender,
FocusedRowChangedEventArgs e)
        {
           // if (CanSyncCurrentRow() && treeListControl.View.FocusedRowHandle
!= GridControl.InvalidRowHandle)
           //
collectionView.MoveCurrentTo(treeListControl.View.FocusedRow);

            MessageBox.Show(" Click on node  "
+((ProjectObject)treeListControl.View.FocusedRow).ElementName.ToString());
        if (((ProjectObject)treeListControl.View.FocusedRow).ElementName ==
"Sprint")
        {

        }
        if (((ProjectObject)treeListControl.View.FocusedRow).ElementName ==
"Requirement")
        {


        }

this._mainWindows.PopulateControl(((ProjectObject)treeListControl.View.FocusedR
ow).ElementName);
        }
        bool CanSyncCurrentRow()
        {
            return treeListControl != null &&
GetIsSynchronizedWithCurrentItem(treeListControl);
        }
        void OnSortDescriptionsCollectionChanged(object sender,
NotifyCollectionChangedEventArgs e)
        {
            if (treeListControl == null)
                return;
            SyncSorting();
        }
        void OnCollectionViewPropertyChanged(object sender,
PropertyChangedEventArgs e)
        {
```

;

```
            if (e.PropertyName == "Count")
                SyncFilter();
        }
        void SyncFilter()
        {
            if (treeListControl == null)
                return;
            if (FilterMode == ModelFilterMode.FilterCriteria)
            {
                treeListControl.FilterCriteria = FilterCriteria;
                treeListControl.View.AllowColumnFiltering = true;
            }
            else
            {
                treeListControl.FilterCriteria = null;
                treeListControl.RefreshData();
                treeListControl.View.AllowColumnFiltering = false;
            }
        }
        public void AttachToTreeListControl(TreeListControl treeListControl)
        {
            if (this.treeListControl != null)
            {
                this.treeListControl.View.CustomNodeFilter -= new
TreeListNodeFilterEventHandler(OnTreeListCustomNodeFilter);
                this.treeListControl.SortInfo.CollectionChanged -= new
NotifyCollectionChangedEventHandler(OnSortInfoCollectionChanged);
                this.treeListControl.View.FocusedRowChanged -= new
FocusedRowChangedEventHandler(OnTreeListControlFocusedRowChanged);

TypeDescriptor.GetProperties(typeof(GridControl))[GridControl.FilterCriteriaPro
perty.Name].RemoveValueChanged(treeListControl,
OnTreeListControlFilterCriteriaChanged);
            }
            this.treeListControl = treeListControl;
            if (treeListControl == null)
                return;
            treeListControl.AutoPopulateColumns = AutoPopulateColumns;

            treeListControl.BeginInit();
            try
            {
                treeListControl.View.CustomNodeFilter += new
TreeListNodeFilterEventHandler(OnTreeListCustomNodeFilter);
                treeListControl.SortInfo.CollectionChanged += new
NotifyCollectionChangedEventHandler(OnSortInfoCollectionChanged);
                treeListControl.View.FocusedRowChanged += new
FocusedRowChangedEventHandler(OnTreeListControlFocusedRowChanged);

TypeDescriptor.GetProperties(typeof(GridControl))[GridControl.FilterCriteriaPro
perty.Name].AddValueChanged(treeListControl,
OnTreeListControlFilterCriteriaChanged);

            }
            finally
            {
                treeListControl.EndInit();
            }
        }
        void OnTreeListControlFilterCriteriaChanged(object sender, EventArgs e)
        {
            FilterCriteria = treeListControl.FilterCriteria;
```

;

```
        }
        void OnSortInfoCollectionChanged(object sender,
NotifyCollectionChangedEventArgs e)
        {
            if (syncGroupSortLocker.IsLocked)
                return;
            syncGroupSortLocker.DoLockedAction(SyncSortInfo);
        }
        void SyncSortInfo()
        {
            if (CollectionView == null)
                return;
            if (CollectionView.SortDescriptions != null)
            {
                CollectionView.SortDescriptions.Clear();
                for (int i = 0; i < treeListControl.SortInfo.Count; i++)
                {
                    GridSortInfo info = treeListControl.SortInfo[i];
                    CollectionView.SortDescriptions.Add(new
SortDescription(info.FieldName, info.SortOrder));
                }
            }
        }
        void SyncSorting()
        {
            if (syncGroupSortLocker.IsLocked)
                return;
            syncGroupSortLocker.DoLockedAction(delegate()
            {
                if (CollectionView.SortDescriptions != null)
                {
                    treeListControl.SortInfo.BeginUpdate();
                    try
                    {
                        treeListControl.ClearSorting();
                        foreach (SortDescription sortDescription in
CollectionView.SortDescriptions)
                        {
                            treeListControl.SortInfo.Add(new GridSortInfo() {
FieldName = sortDescription.PropertyName, SortOrder = sortDescription.Direction
});
                        }
                    }
                    finally
                    {
                        treeListControl.SortInfo.EndUpdate();
                    }
                }
            });
        }
        void OnTreeListCustomNodeFilter(object sender,
TreeListNodeFilterEventArgs e)
        {
        //    if (CollectionView.Filter == null || FilterMode ==
ModelFilterMode.FilterCriteria)
        //        return;
        //    e.Visible = CollectionView.Filter(e.Node.Content);
        //    e.Handled = true;
        }
        public void PopulateDetailPanel()
        {
```

;

```
        }

        internal void PopulateDetailPanel(DevExpress.Xpf.Docking.LayoutPanel
paneToolbox2)
        {
            this.m_paneToolbox2 = paneToolbox2;
        }
    }


}
```

;

```
//--------------------------------------------------------------------------------
-
UserModel.cs
The class give service for data acces layer for User table.
//-------------------------------------------------------------------------------

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;

namespace AgileManagement
{
    public class UserModel
    {

        private static Users _currentUser;

        public static Users CurrentUser
        {
            get { return _currentUser; }
            set { _currentUser = value; }
        }

        public static List<Users> Get()
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {
                    return agileMngEntities.Users.ToList();
                }
                catch (Exception ex)
                {
                    throw;
                }
            }
        }

        public static List<Users> Users
        {
            get
            {
                using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
                {

                     return agileMngEntities.Users.ToList();
                }
            }
        }
```

;

```
        public static List<UserGroups> GroupsName
        {
            get
            {
                using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
                {

                    return agileMngEntities.UserGroups.ToList();
                }
            }
        }

        public static string[] Data
        {
            get
            {
                AgileMngEntities2 agileMngEntities = new AgileMngEntities2();
                //List<string>  names = agileMngEntities.Users.Select(u=>;
                //     var numsInPlace = numbers.Select((num, index) => new {
Num = num, InPlace = (num == index) });
                //    var usersNames =
                // from p in agileMngEntities.Users
                // select p.ProductName;
                List<string> names = new List<string>();
                foreach (Users currntUser in agileMngEntities.Users)
                {
                    names.Add(currntUser.Name);
                }
                return names.ToArray();
            }
        }
    }
}
```

```
;

//--------------------------------------------------------------------------------
-
BacklogView.xaml
This class contain definition of UI  and logic for UserView.
//--------------------------------------------------------------------------------
-


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class UserView : UserControl
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<Users> _usersList;
        UserViewModel _userViewModel;
        public UserView()
        {
            InitializeComponent();
            //    _requirementList= RequirementModel.Get();
            _usersList = AgileMngEntities2.Users.ToList();
            //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
            gridControl1.ItemsSource = _usersList;
            _userViewModel = new UserViewModel(ref this.gridControl1, ref
this.tableView1,ref AgileMngEntities2);
            this.itemAdd.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_userViewModel.itemAdd_ItemClick);
             this.itemDelete.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_userViewModel.itemDelete_ItemClick);
             this.barButtonItem2.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_userViewModel.barButtonItem2_ItemCli
ck);
             this.itemUndo.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_userViewModel.itemUndo_ItemClick);
             this.barButtonItem3.ItemClick += new
DevExpress.Xpf.Bars.ItemClickEventHandler(_userViewModel.barButtonItem3_ItemCli
ck);
        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {
```

```
;
        }
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e
            )
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        private void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Users user = new Users();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            user.UsersID = 0;
            if (_usersList.Count != 0)
            {
                user.UsersID = _usersList.Max(u => u.UsersID);
            }

            user.UsersID += 1;
            _usersList.Add(user);
            AgileMngEntities2.AddToUsers(user);
            gridControl1.ItemsSource = _usersList;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.Users.DeleteObject((Users)gridControl1.GetFocusedRow());
            _usersList.Remove((Users)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);
            // RequirementModel.UpdateChanges(_requirementList);
            foreach (Users dl in AgileMngEntities2.Users.ToArray())
            {
                string currentName = dl.Name;
```

;

```
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in AgileMngEntities2.Users.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _usersList = AgileMngEntities2.Users.ToList();

            gridControl1.ItemsSource = AgileMngEntities2.Backlog;

        }

        //Assign task to sprint
        private void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
        //    WorkItemSelectView workItemSelectView = new WorkItemSelectView();
        //    WorkItem workItem = workItemSelectView.ShowDialogw(3);

        //    int[] listSelect = this.tableView1.GetSelectedRowHandles();
        //    Users user = (Sprint)this.gridControl1.GetRow(listSelect[0]);
        //    workItem.SprintID = user.UsersID;


        //    //Update the WorkItem
        //    Sprint sprintTemp = _usersList.First(u => u.UsersID ==
user.SprintID);
        //    WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
        //    workItemTemp.SprintID = user.SprintID;

            //AgileMngEntities2.SaveChanges();

        }

        public void Refresh()
        {
            _usersList = AgileMngEntities2.Users.ToList();
            this.gridControl1.RefreshData();
        }
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
UserView.xaml
This class contain the definition of UI for BacklogView
//--------------------------------------------------------------------------------
-


<UserControl x:Class="AgileManagement.UserView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:local="clr-namespace:AgileManagement"
             mc:Ignorable="d"
             d:DesignHeight="322" d:DesignWidth="432"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxe="http://schemas.devexpress.com/winfx/2008/xaml/editors"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
Loaded="UserControl_Loaded">
    <Grid>
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True" Margin="12"
Name="barManager1" ItemClick="barManager1_ItemClick">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
                <dxb:BarButtonItem x:Name="itemUndo" Content="Undo"
Glyph="/AgileManagement;component/Images/Return_16x161.png"  />
                <dxb:BarButtonItem x:Name="itemAdd" Content="Add"
Glyph="pack://application:,,,/Images/Add_16x16.png"  />
                <dxb:BarButtonItem x:Name="itemDelete" Content="Delete"
Glyph="pack://application:,,,/Images/Delete_16x16.png"  />
                <dxb:BarButtonItem Content="barButtonItem1"
Name="barButtonItem1" />
                <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"  />
                <dxb:BarButtonItem Content="Assign WorkItem to Sprint"
Name="barButtonItem3" Glyph="/AgileManagement;component/Images/Edit_16x161.png"
 />
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                        <dxb:BarButtonItemLink BarItemName="itemAdd"  />
                        <dxb:BarButtonItemLink BarItemName="itemDelete"  />
                        <dxb:BarButtonItemLink BarItemName="itemUndo" />
                        <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem3" />
                    </dxb:Bar.ItemLinks>
```

;

```
                </dxb:Bar>

            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
                    <dxg:GridColumn FieldName="UsersID"  Name="colID"/>
                    <dxg:GridColumn FieldName="Name" Name="colName" />
                    <dxg:GridColumn FieldName="UserGroupName"
Name="colUserGroupName"
                                    EditSettings="{dxe:ComboBoxSettings
DisplayMember= Name, ValueMember=Name, ItemsSource={x:Static
local:UserModel.GroupsName}}" />
                    <dxg:GridColumn FieldName="Description"
Name="colDescription" />
                    <dxg:GridColumn FieldName="Password" Name="colPassword" />
                </dxg:GridControl.Columns>
                <dxg:GridControl.View>
                    <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
 AutoWidth="True"/>
                </dxg:GridControl.View>
            </dxg:GridControl>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</UserControl>
```

;

```
//------------------------------------------------------------------------------
-
UserViewModel.cs
This class contain logic for display the user  and function as mediator to
database.
//------------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using DevExpress.Xpf.Grid;

namespace AgileManagement
{
    public  class UserViewModel
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<Users> _usersList;
        UserViewModel _userViewModel;
        GridControl gridControl1;
        TableView tableView1;

        public UserViewModel(ref GridControl gridControl, ref  TableView
tableView, ref AgileMngEntities2 agileMngEntities)
        {
            _usersList = AgileMngEntities2.Users.ToList();
            AgileMngEntities2 = agileMngEntities;
            gridControl1 = gridControl;
            tableView1 = tableView;
            gridControl1.ItemsSource = _usersList;
            gridControl1.RefreshData();
        }

        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
           // MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        public  void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);
```

```
;
            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            Users user = new Users();

            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            user.UsersID = 0;
            if (_usersList.Count != 0)
            {
                user.UsersID = _usersList.Max(u => u.UsersID);
            }

            user.UsersID += 1;
            _usersList.Add(user);
            AgileMngEntities2.AddToUsers(user);
            gridControl1.ItemsSource = _usersList;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        public  void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.Users.DeleteObject((Users)gridControl1.GetFocusedRow());
            _usersList.Remove((Users)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        public  void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);
            // RequirementModel.UpdateChanges(_requirementList);
            foreach (Users dl in AgileMngEntities2.Users.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in AgileMngEntities2.Users.ToArray()
                             where spr.Name == currentName
                             select spr).Count();

                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }
```

;

```
        //undo
        public void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _usersList = AgileMngEntities2.Users.ToList();

            gridControl1.ItemsSource = AgileMngEntities2.Backlog;

        }

        //Assign task to sprint
        public void barButtonItem3_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
        //    WorkItemSelectView workItemSelectView = new WorkItemSelectView();
        //    WorkItem workItem = workItemSelectView.ShowDialogw(3);

        //    int[] listSelect = this.tableView1.GetSelectedRowHandles();
        //    Users user = (Sprint)this.gridControl1.GetRow(listSelect[0]);
        //    workItem.SprintID = user.UsersID;


        //    //Update the WorkItem
        //    Sprint sprintTemp = _usersList.First(u => u.UsersID ==
user.SprintID);
        //    WorkItem workItemTemp = AgileMngEntities2.WorkItem.First(u =>
u.WorkItemID == workItem.WorkItemID);
        //    workItemTemp.SprintID = user.SprintID;

            //AgileMngEntities2.SaveChanges();

        }

        public void Refresh()
        {
            _usersList = AgileMngEntities2.Users.ToList();
            this.gridControl1.RefreshData();
        }

    }
}
```

;

```
//------------------------------------------------------------------------
-
WorkItemModel.cs
This class data access  layer and give database service for workItem
//------------------------------------------------------------------------
-



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using AgileManagement.DAL;

namespace AgileManagement
{
    class WorkItemModel
    {

        public static List<WorkItem> Get()
        {
            using (AgileMngEntities2 agileMngEntities = new
AgileMngEntities2())
            {
                try
                {
                    return agileMngEntities.WorkItem.ToList();
                }
                catch (Exception ex)
                {
                    throw;
                }
            }

        }

         public static List<WorkItemStatus>  Status
         {
           get
           {
               return WorkItemStatus.GetWorItemStatus();
            }
         }

    }
}
```

;

```
//------------------------------------------------------------------------
-
WorkItemSelectView .cs
This class contain UI logic for WorkItemSelectView
It  give to user option to select  workitem.
//------------------------------------------------------------------------
-


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using DevExpress.Data.Selection;

namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class WorkItemSelectView : Window
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<WorkItem> _workItemList;


        public static List<Users> Users
        {
            get
            {
                using (AgileMngEntities2 AgileMngEntities2 = new 
AgileMngEntities2())
                {


                    return AgileMngEntities2.Users.ToList();


                }
            }
        }

        public WorkItemSelectView()
        {
            InitializeComponent();
            this.WindowStyle =System.Windows.WindowStyle.None;
         //    _requirementList= RequirementModel.Get();
            _workItemList = AgileMngEntities2.WorkItem.ToList();
          //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
```

```
;
            gridControl1.ItemsSource = _workItemList;
            //colAssignTo.EditSettings.

        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e
            )
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked.");
        }

        private void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
          //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            WorkItem workItem = new WorkItem();
            workItem.WorkItemID = 0;
            workItem.Owner = UserModel.CurrentUser.Name;
            // AgileMngEntities2.Requiments.AddObject(req);
           //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            if (_workItemList.Count != 0)
            {
                workItem.WorkItemID = _workItemList.Max(u => u.WorkItemID);
            }
             workItem.WorkItemID += 1;
           _workItemList.Add(workItem);
           AgileMngEntities2.AddToWorkItem(workItem);
              gridControl1.ItemsSource = _workItemList;
             gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
             //AgileMngEntities2.SaveChanges();

        }

        //Delete
        private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.WorkItem.DeleteObject((WorkItem)gridControl1.GetFocusedRow())
;
             _workItemList.Remove((WorkItem)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
```

;

```
        private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
           // RequirementModel.UpdateChanges(_requirementList);
            this.Close();
          //  AgileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _workItemList = WorkItemModel.Get();

            gridControl1.ItemsSource = AgileMngEntities2.WorkItem;

        }

        public WorkItem ShowDialogw(int i)
        {
            this.ShowDialog();

            int[] listSelect = this.tableView1.GetSelectedRowHandles();
            WorkItem workItem =
(WorkItem)this.gridControl1.GetRow(listSelect[0]);

            return workItem;

        }


    }
}
```

;

```
//------------------------------------------------------------------------------
-
WorkItemSelectView.xaml
This class contain the definition of UI for WorkItemSelectView
//------------------------------------------------------------------------------

<Window x:Class="AgileManagement.WorkItemSelectView"
             xmlns:local="clr-namespace:AgileManagement"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:dxe="http://schemas.devexpress.com/winfx/2008/xaml/editors"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="322" d:DesignWidth="432"
             xmlns:dxg="http://schemas.devexpress.com/winfx/2008/xaml/grid"
             xmlns:dxb="http://schemas.devexpress.com/winfx/2008/xaml/bars"
            Title="Window1" Height="300" Width="300"

            Loaded="UserControl_Loaded">


    <Grid>
        <!--region #1-->
        <dxb:BarManager CreateStandardLayout="True" Margin="12"
Name="barManager1" ItemClick="barManager1_ItemClick">

            <!--Create Button Items-->
            <dxb:BarManager.Items>
               <dxb:BarButtonItem Content="Save" Name="barButtonItem2"
Glyph="/AgileManagement;component/Images/Save_16x161.png"
ItemClick="barButtonItem2_ItemClick" />
            </dxb:BarManager.Items>

            <dxb:BarManager.Bars>
                <!--Use the BarItemHorzIndent property to increase the distance
between items-->
                <dxb:Bar x:Name="barMainMenu" Caption="Main Menu"
IsMainMenu="True" BarItemHorzIndent="10">
                    <dxb:Bar.DockInfo>
                        <dxb:BarDockInfo ContainerType="Top" Row="0" />
                    </dxb:Bar.DockInfo>
                    <dxb:Bar.ItemLinks>
                         <!--Create a separator between items-->
                        <dxb:BarItemLinkSeparator />
                        <dxb:BarButtonItemLink BarItemName="barButtonItem2" />
                    </dxb:Bar.ItemLinks>
                </dxb:Bar>

            </dxb:BarManager.Bars>
            <dxg:GridControl Name="gridControl1">
                <dxg:GridControl.Columns>
                    <dxg:GridColumn FieldName="WorkItemID"  Name="colID"/>
                    <dxg:GridColumn FieldName="Name" Name="colReq" />
                    <dxg:GridColumn FieldName="Description"
Name="colDescription" />
                    <dxg:GridColumn FieldName="AssignTo" Name="colAssignTo"
```

;

```
                                    EditSettings="{dxe:ComboBoxSettings
DisplayMember=Name, ValueMember=Name, ItemsSource={x:Static
local:UserModel.Users}}" />

                    <dxg:GridColumn FieldName="Owner" Name="colOwner" />
                    <dxg:GridColumn FieldName="EstimateTImeLow"
Name="colEstimateTImeLow" />
                    <dxg:GridColumn FieldName="EstimateTImeHigh"
Name="colEstimateTImeHigh" />
                    <dxg:GridColumn FieldName="SprintID" Name="colSprintID" />
                    <dxg:GridColumn FieldName="BackLogID" Name="colBackLogID"
/>
                </dxg:GridControl.Columns>
                <dxg:GridControl.View>
                    <dxg:TableView Name="tableView1"
AllowMoveColumnToDropArea="False" AllowMoving="False"
ClipboardCopyWithHeaders="True" ShowColumnHeaders="True" ShowGroupPanel="False"
/>
                </dxg:GridControl.View>
            </dxg:GridControl>
        </dxb:BarManager>
        <!--endregion #1-->
    </Grid>
</Window>
```

;

```
//--------------------------------------------------------------------------------
-
WorkItemStatus.cs
This class contqain the status of workitem.
//--------------------------------------------------------------------------------


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.Objects.DataClasses;

namespace AgileManagement.DAL
{
    public class WorkItemStatus
    {

        private string name;

        public WorkItemStatus(string mname)
        {
            name = mname;
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }

        public static List<WorkItemStatus> GetWorItemStatus()
        {

            List<WorkItemStatus> list = new List<WorkItemStatus>();

            list.Add(new WorkItemStatus("New"));
            list.Add(new WorkItemStatus("InProgress"));
            list.Add(new WorkItemStatus("Done"));
            list.Add(new WorkItemStatus("Reject"));

            return list;

        }

    }


    public partial  class WorkItem
    {

        public string StatusName
        {
            get
            {
                if (Status == 1)
                {
                    return "New";
                }
                else if(Status==2)
                {
```

;

```
                    return "InProgress";
                }
                else if(Status ==3)
                {
                    return "Done";
                }
                else if (Status == 4)
                {
                    return "Reject";
                }
                else
                {
                    return "New";
                }

            }
            set
            {
                if (value == "New")
                {
                    Status = 1;
                }
                else if (value == "InProgress")
                {
                    Status = 2;

                }
                else if (value =="Done")
                {
                    Status = 3;
                }
                else if (value == "Reject")
                {
                    Status = 4;
                }
                else
                {
                    Status = 1;
                }

            }

        }
    }
}
```

;

```
//--------------------------------------------------------------------------------
-
WorkItemView.xaml.cs
This class contain UI and  logic for display  the  WorkItem
//--------------------------------------------------------------------------------
-


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using System.Data.Objects;
using Microsoft.Win32;

using System.ComponentModel;
using System.IO;



namespace AgileManagement
{
    /// <summary>
    /// Interaction logic for SprintView.xaml
    /// </summary>
    public partial class WorkItemView : UserControl
    {

        AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
        private List<WorkItem> _workItemList;

        WorkItemViewModel _workItemViewModel;
        public static List<Users> Users
        {
            get
            {
                using (AgileMngEntities2 AgileMngEntities2 = new
AgileMngEntities2())
                {


                    return AgileMngEntities2.Users.ToList();


                }
            }
        }

        public WorkItemView()
        {
            InitializeComponent();
         //   _requirementList= RequirementModel.Get();
```

;

```
            _workItemViewModel = new WorkItemViewModel(ref 
this.gridControl1,ref this.tableView1,ref this.AgileMngEntities2);
            _workItemList = AgileMngEntities2.WorkItem.ToList();
          //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
            gridControl1.ItemsSource = _workItemList;
            //colAssignTo.EditSettings.
            this.itemAdd.ItemClick += new 
DevExpress.Xpf.Bars.ItemClickEventHandler(_workItemViewModel.itemAdd_ItemClick)
;
            this.itemDelete.ItemClick += new 
DevExpress.Xpf.Bars.ItemClickEventHandler(_workItemViewModel.itemDelete_ItemCli
ck);
            this.itemUndo.ItemClick += new 
DevExpress.Xpf.Bars.ItemClickEventHandler(_workItemViewModel.itemUndo_ItemClick
);
            this.barButtonItem2.ItemClick += new 
DevExpress.Xpf.Bars.ItemClickEventHandler(_workItemViewModel.barButtonItem2_Ite
mClick);
            this.barButtonItemExcel.ItemDoubleClick += new 
DevExpress.Xpf.Bars.ItemClickEventHandler(_workItemViewModel.barButtonItemExcel
_ItemDoubleClick);


        }

        private void UserControl_Loaded(object sender, RoutedEventArgs e)
        {

        }

      //  Export to excel
        private void barManager1_ItemClick(object sender, 
DevExpress.Xpf.Bars.ItemClickEventArgs e )
        {



        }

        private void itemAdd_ItemClick(object sender, 
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
          //  MessageBox.Show("Item " + e.Item.Content + " has been clicked 
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            WorkItem workItem = new WorkItem();
            workItem.WorkItemID = 0;
            workItem.Status = 1;
            workItem.Owner = UserModel.CurrentUser.Name;
            // AgileMngEntities2.Requiments.AddObject(req);
           //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            if (_workItemList.Count != 0)
            {
                workItem.WorkItemID = _workItemList.Max(u => u.WorkItemID);
            }
```

```
;
             workItem.WorkItemID += 1;
            _workItemList.Add(workItem);
            AgileMngEntities2.AddToWorkItem(workItem);
               gridControl1.ItemsSource = _workItemList;
             gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
             //AgileMngEntities2.SaveChanges();

        }

        //Delete
        private void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.WorkItem.DeleteObject((WorkItem)gridControl1.GetFocusedRow())
;
             _workItemList.Remove((WorkItem)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
        private void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
           // RequirementModel.UpdateChanges(_requirementList);
            foreach (WorkItem dl in AgileMngEntities2.WorkItem.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in AgileMngEntities2.WorkItem.ToArray()
                             where spr.Name == currentName
                             select spr).Count();
                    if (dl.Status == null)
                    {
                        dl.Status = 1;
                    }
                 if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }

        //undo
        private void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _workItemList = WorkItemModel.Get();

            gridControl1.ItemsSource = AgileMngEntities2.WorkItem;

        }
```

;

```
        public void Refresh()
        {
            AgileMngEntities2 = new AgileMngEntities2();
            this._workItemList = AgileMngEntities2.WorkItem.ToList();
            gridControl1.ItemsSource = AgileMngEntities2.WorkItem;
            this.gridControl1.RefreshData();
      }

        public void  Queuery(QueryWorkItem queryWorkItem)
        {
            AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
            List<WorkItem> workItemLst=  AgileMngEntities2.WorkItem.ToList();
           //string queuery ="score.Status > 80"

            List<WorkItem> workItemCurrentLst=new List<WorkItem>();
            bool bReject = false;
            bool bAlreadyExist=false;
            if (queryWorkItem.Status != 0)
            {
                bAlreadyExist = true;
                //reject status
                if (queryWorkItem.Status / 1000 == 1)
                {
                    bReject = true;
                    List<WorkItem> workItemRejectLst =
                   (from score in workItemLst
                    where score.Status >= 4
                    select score).ToList<WorkItem>();
                    if (workItemRejectLst != null)
                    {
                        workItemCurrentLst.AddRange(workItemRejectLst);
                    }

                    gridControl1.ItemsSource = workItemCurrentLst;
                    this.gridControl1.RefreshData();
                }

                //Done
                if ((queryWorkItem.Status / 100 >= 1 && bReject == false) ||
(queryWorkItem.Status / 100 > 10 && bReject == true))
                {
                    List<WorkItem> workItemDoneLst =
                   (from score in workItemLst
                    where score.Status == 3
                    select score).ToList<WorkItem>();

                    if (workItemDoneLst != null)
                    {
                        workItemCurrentLst.AddRange(workItemDoneLst);
                    }
                    gridControl1.ItemsSource = workItemCurrentLst;
                    this.gridControl1.RefreshData();

                }

                //InProgress
                float myProgess = (float)queryWorkItem.Status % 100;
                if (myProgess >= 10)
                {
                    List<WorkItem> workItemInProgressLst =
                   (from score in workItemLst
```

```
;
                    where score.Status == 2
                    select score).ToList<WorkItem>();

                    if (workItemInProgressLst != null)
                    {
                        workItemCurrentLst.AddRange(workItemInProgressLst);
                    }
                    gridControl1.ItemsSource = workItemCurrentLst;
                    this.gridControl1.RefreshData();

                }


                //New
                float myNew = (float)queryWorkItem.Status % 10;
                if (myNew >= 1)
                {
                    List<WorkItem> workItemInNewLst =
                   (from score in workItemLst
                    where score.Status == 1
                    select score).ToList<WorkItem>();

                    if (workItemInNewLst != null)
                    {
                        workItemCurrentLst.AddRange(workItemInNewLst);
                    }
                    gridControl1.ItemsSource = workItemCurrentLst;
                    this.gridControl1.RefreshData();

                }
            }
            else
            {
                workItemCurrentLst = workItemLst;
            }

            /*
             if (queryWorkItem.AssignTo != null && queryWorkItem.AssignTo!="")
             {

                 List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
                 List<WorkItem> workItemResultAssinToLst =
                 (from score in workItemAssignTo
                  where score.AssignTo == queryWorkItem.AssignTo
                  select score).ToList<WorkItem>();

                 gridControl1.ItemsSource = workItemResultAssinToLst;
                 this.gridControl1.RefreshData();
             }
             */
            //Assign To
             if (queryWorkItem.AssignTo != null && queryWorkItem.AssignTo !=
"")
             {
                 bAlreadyExist = true;

                 //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
                 List<WorkItem> workItemResultAssinToLst =
                 (from score in workItemCurrentLst
                  where score.AssignTo == queryWorkItem.AssignTo
```

;

```
                 select score).ToList<WorkItem>();
                workItemCurrentLst = workItemResultAssinToLst;
                gridControl1.ItemsSource = workItemResultAssinToLst;
                this.gridControl1.RefreshData();
            }

            //Owner
            if (bAlreadyExist == false)
            {
                workItemCurrentLst = AgileMngEntities2.WorkItem.ToList();
            }
            if (queryWorkItem.Owner != null && queryWorkItem.Owner != "")
            {
                bAlreadyExist = true;
                //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
                List<WorkItem> workItemResultwnerLst =
                (from score in workItemCurrentLst
                 where score.Owner == queryWorkItem.Owner
                 select score).ToList<WorkItem>();

                gridControl1.ItemsSource = workItemResultwnerLst;
                this.gridControl1.RefreshData();
            }
            if (bAlreadyExist == false)
            {
                workItemCurrentLst = AgileMngEntities2.WorkItem.ToList();
            }
            if (queryWorkItem.SprintID != null && queryWorkItem.SprintID != 0)
            {
                bAlreadyExist = true;
                //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
                List<WorkItem> workItemResultwnerLst =
                (from score in workItemCurrentLst
                 where score.SprintID == queryWorkItem.SprintID
                 select score).ToList<WorkItem>();

                gridControl1.ItemsSource = workItemResultwnerLst;
                this.gridControl1.RefreshData();
            }

        }

        private void ExcelBarItem(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {

        }

        private void barButtonItemExcel_ItemDoubleClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();

            saveFileDialog1.InitialDirectory = @"C:\";

            saveFileDialog1.Title = "Save text Files";

            //saveFileDialog1.CheckFileExists = true;

           // saveFileDialog1.CheckPathExists = true;
```

;

```
            saveFileDialog1.DefaultExt = "csv";
            saveFileDialog1.Filter = "Excel files (*.xlsx)|*.txt|All files
(*.*)|*.*";
            saveFileDialog1.FilterIndex = 2;
            saveFileDialog1.RestoreDirectory = true;
            saveFileDialog1.ShowDialog();


            tableView1.ExportToXlsx(saveFileDialog1.FileName);
        }

    }
}
```

;

```
//---------------------------------------------------------------------------
-
WorkItemViewModel.cs
This class contain logic for display the WorkItem  and function as mediator to
database.
//---------------------------------------------------------------------------



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using AgileManagement.DAL;
using DevExpress.Data.Selection;
using DevExpress.Xpf.Grid;

namespace AgileManagement
{
   public  class WorkItemViewModel
   {


       AgileMngEntities2 AgileMngEntities2;
       private List<WorkItem> _workItemList;

       GridControl gridControl1;
       TableView tableView1;

       public WorkItemViewModel(ref GridControl gridControl, ref  TableView
tableView, ref AgileMngEntities2 agileMngEntities2)
       {
           try
           {

               AgileMngEntities2 = agileMngEntities2;
               gridControl1 = gridControl;
               tableView1 = tableView;

               _workItemList = AgileMngEntities2.WorkItem.ToList();
               //  gridControl1.ItemsSource = AgileMngEntities2.Requiments;
               gridControl1.ItemsSource = _workItemList;
               //colAssignTo.EditSettings.

               gridControl1.ItemsSource = _workItemList;
               gridControl1.RefreshData();
           }
           catch (Exception ex)
           {
```

```
;
                MessageBox.Show(ex.Message);

            }


        }

        //  Export to excel
        private void barManager1_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {



        }

        public void itemAdd_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            //  MessageBox.Show("Item " + e.Item.Content + " has been clicked
xxxxx.");
            //  Requiments req = new Requiments();
            //RequirementModel.Add(req);

            //_requirementList = RequirementModel.Get();
            //gridControl1.ItemsSource = _requirementList;

            WorkItem workItem = new WorkItem();
            workItem.WorkItemID = 0;
            workItem.Status = 1;
            workItem.Owner = UserModel.CurrentUser.Name;
            // AgileMngEntities2.Requiments.AddObject(req);
            //  AgileMngEntities2.AddToRequiments(req);
            // AgileMngEntities2.SaveChanges();
            if (_workItemList.Count != 0)
            {
                workItem.WorkItemID = _workItemList.Max(u => u.WorkItemID);
            }
            workItem.WorkItemID += 1;
            _workItemList.Add(workItem);
            AgileMngEntities2.AddToWorkItem(workItem);
            gridControl1.ItemsSource = _workItemList;
            gridControl1.RefreshData();
            // AgileMngEntities2.Requiments.Attach(req);
            //AgileMngEntities2.SaveChanges();

        }

        //Delete
        public  void itemDelete_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            MessageBox.Show("Item " + e.Item.Content + " has been clicked
yyyyy.");

AgileMngEntities2.WorkItem.DeleteObject((WorkItem)gridControl1.GetFocusedRow())
;
            _workItemList.Remove((WorkItem)gridControl1.GetFocusedRow());
            gridControl1.RefreshData();
        }

        //save
```

;

```
        public  void barButtonItem2_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            // RequirementModel.UpdateChanges(_requirementList);
            foreach (WorkItem dl in AgileMngEntities2.WorkItem.ToArray())
            {
                string currentName = dl.Name;
                // List<Sprint> sprintLst=
AgileMngEntities2.Sprint.ToArray().Select(u=>u.Name==currentName);
                int count = (from spr in AgileMngEntities2.WorkItem.ToArray()
                             where spr.Name == currentName
                             select spr).Count();
                if (dl.Status == null)
                {
                    dl.Status = 1;
                }
                if (count > 1)
                {
                    MessageBox.Show("There is Name that is not unique therefor
is not save.\n The name :" + currentName, "Error Saving", MessageBoxButton.OK,
MessageBoxImage.Error);
                    return;

                }
            }
            AgileMngEntities2.SaveChanges();
        }

        //undo
        public  void itemUndo_ItemClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            AgileMngEntities2 = new AgileMngEntities2();
            _workItemList = WorkItemModel.Get();

            gridControl1.ItemsSource = AgileMngEntities2.WorkItem;

        }

        public void Refresh()
        {
            AgileMngEntities2 = new AgileMngEntities2();
            this._workItemList = AgileMngEntities2.WorkItem.ToList();
            gridControl1.ItemsSource = AgileMngEntities2.WorkItem;
            this.gridControl1.RefreshData();
        }

        public void Queuery(QueryWorkItem queryWorkItem)
        {
            AgileMngEntities2 AgileMngEntities2 = new AgileMngEntities2();
            List<WorkItem> workItemLst = AgileMngEntities2.WorkItem.ToList();
            //string queuery ="score.Status > 80"

            List<WorkItem> workItemCurrentLst = new List<WorkItem>();
            bool bReject = false;
            bool bAlreadyExist = false;
            if (queryWorkItem.Status != 0)
            {
                bAlreadyExist = true;
                //reject status
                if (queryWorkItem.Status / 1000 == 1)
                {
```

;

```
                        bReject = true;
                        List<WorkItem> workItemRejectLst =
                       (from score in workItemLst
                        where score.Status >= 4
                        select score).ToList<WorkItem>();
                        if (workItemRejectLst != null)
                        {
                            workItemCurrentLst.AddRange(workItemRejectLst);
                        }

                        gridControl1.ItemsSource = workItemCurrentLst;
                        this.gridControl1.RefreshData();
                    }

                    //Done
                    if ((queryWorkItem.Status / 100 >= 1 && bReject == false) ||
(queryWorkItem.Status / 100 > 10 && bReject == true))
                    {
                        List<WorkItem> workItemDoneLst =
                       (from score in workItemLst
                        where score.Status == 3
                        select score).ToList<WorkItem>();

                        if (workItemDoneLst != null)
                        {
                            workItemCurrentLst.AddRange(workItemDoneLst);
                        }
                        gridControl1.ItemsSource = workItemCurrentLst;
                        this.gridControl1.RefreshData();

                    }

                    //InProgress
                    float myProgess = (float)queryWorkItem.Status % 100;
                    if (myProgess >= 10)
                    {
                        List<WorkItem> workItemInProgressLst =
                       (from score in workItemLst
                        where score.Status == 2
                        select score).ToList<WorkItem>();

                        if (workItemInProgressLst != null)
                        {
                            workItemCurrentLst.AddRange(workItemInProgressLst);
                        }
                        gridControl1.ItemsSource = workItemCurrentLst;
                        this.gridControl1.RefreshData();

                    }


                    //New
                    float myNew = (float)queryWorkItem.Status % 10;
                    if (myNew >= 1)
                    {
                        List<WorkItem> workItemInNewLst =
                       (from score in workItemLst
                        where score.Status == 1
                        select score).ToList<WorkItem>();

                        if (workItemInNewLst != null)
                        {
```

;

```
                workItemCurrentLst.AddRange(workItemInNewLst);
            }
            gridControl1.ItemsSource = workItemCurrentLst;
            this.gridControl1.RefreshData();

        }
    }
    else
    {
        workItemCurrentLst = workItemLst;
    }

    /*
     if (queryWorkItem.AssignTo != null && queryWorkItem.AssignTo!="")
     {

         List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
         List<WorkItem> workItemResultAssinToLst =
         (from score in workItemAssignTo
          where score.AssignTo == queryWorkItem.AssignTo
          select score).ToList<WorkItem>();

         gridControl1.ItemsSource = workItemResultAssinToLst;
         this.gridControl1.RefreshData();
     }
     */
    //Assign To
    if (queryWorkItem.AssignTo != null && queryWorkItem.AssignTo != "")
    {
        bAlreadyExist = true;

        //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
        List<WorkItem> workItemResultAssinToLst =
        (from score in workItemCurrentLst
         where score.AssignTo == queryWorkItem.AssignTo
         select score).ToList<WorkItem>();
        workItemCurrentLst = workItemResultAssinToLst;
        gridControl1.ItemsSource = workItemResultAssinToLst;
        this.gridControl1.RefreshData();
    }

    //Owner
    if (bAlreadyExist == false)
    {
        workItemCurrentLst = AgileMngEntities2.WorkItem.ToList();
    }
    if (queryWorkItem.Owner != null && queryWorkItem.Owner != "")
    {
        bAlreadyExist = true;
        //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
        List<WorkItem> workItemResultwnerLst =
        (from score in workItemCurrentLst
         where score.Owner == queryWorkItem.Owner
         select score).ToList<WorkItem>();

        gridControl1.ItemsSource = workItemResultwnerLst;
        this.gridControl1.RefreshData();
    }
    if (bAlreadyExist == false)
```

;

```
            {
                workItemCurrentLst = AgileMngEntities2.WorkItem.ToList();
            }
            if (queryWorkItem.SprintID != null && queryWorkItem.SprintID != 0)
            {
                bAlreadyExist = true;
                //List<WorkItem> workItemAssignTo =
AgileMngEntities2.WorkItem.ToList();
                List<WorkItem> workItemResultwnerLst =
                (from score in workItemCurrentLst
                 where score.SprintID == queryWorkItem.SprintID
                 select score).ToList<WorkItem>();

                gridControl1.ItemsSource = workItemResultwnerLst;
                this.gridControl1.RefreshData();
            }

        }


        public void barButtonItemExcel_ItemDoubleClick(object sender,
DevExpress.Xpf.Bars.ItemClickEventArgs e)
        {
            System.Windows.Forms.SaveFileDialog saveFileDialog1 = new
System.Windows.Forms.SaveFileDialog();

            saveFileDialog1.InitialDirectory = @"C:\";

            saveFileDialog1.Title = "Save text Files";

            //saveFileDialog1.CheckFileExists = true;

            // saveFileDialog1.CheckPathExists = true;

            saveFileDialog1.DefaultExt = "csv";

            saveFileDialog1.Filter = "Excel files (*.xlsx)|*.txt|All files
(*.*)|*.*";

            saveFileDialog1.FilterIndex = 2;

            saveFileDialog1.RestoreDirectory = true;

            saveFileDialog1.ShowDialog();



            tableView1.ExportToXlsx(saveFileDialog1.FileName);

        }



    }
}
```

;

8.2 הצעת הפרוייקט

<u>פרטים אישיים:</u>

שם הסטודנט: אורן טביב

ת.ז.: 027154780

טלפון: 052-6180093

דואר אלקטרוני: orentabib@yahoo.com

<u>שם הפרוייקט:</u> **Agile Management**

<u>שם המנחה:</u> **מאיה הרמן**

<u>מטרת הפרוייקט:</u>

הפרוייקט אמור לספק לראש צוות /מנהל פיתוח / מפתח ,אפלקצייה לניהול הפרוייקט במתודלגיה Agile
האפלקצייה אמורה לעזור לתעד את כל שלביי הפיתוח ע"פ מתודולוגיית Agile כולל מעקב על הפרוייקט.
האפלקצייה אמורה להיות מותאמת לכל שלביי הפיתוח של המתודלגיה בשלב ראשון מותאמת לפרוייקט אחד ובהמשך לכמה פרוייקטים.
האפלקצייה אמורה לתת אפשרות לתעד את דרישות המערכת את הישבות שנערכות ואת התקדמות הפרוייקט .
לדרישות המערכת צרכים להיות עקיבות לבצוע המשימה כלומר כל דרישה צריכה להצביע למשימה שכיסתה את הדרישה .
כל משימה יכולה להצביע לכמה דרישות ,וכמה דרישות יכולת להצביע למשימה אחת .
כל משימה תכיל את סטטוס הפיתוח ותוכל להשתנות מעת לעת.
הפיתוח יחולק לאטרציות (Sprint) וכל אטרציה תכיל את הדרישות שהיא מכילה וכל דרישה תכיל את המשימה שאמורה לכסות את הדרישה
פרוט שלבי העבודה באפלקצייה:

- האפלקציה אמורה לתת אופצייה לתעד את תחזוקה של רשימת פריטי העבודה לביצוע, מסודרים לפי קדימויות, המכונה עתודת המוצר (Product Backlog).
  כלומר האפלקציה אמורה לתת אפשרות תיעוד של כל משימות של חבריי הצוות.

- האפלקציה אמורה לתת אופצייה לתעד את איטרציות הפיתוח (Sprint) הכולל משימות לכל מפתח תוכן האטרצייה יעד לסיום המשימות.

- האפלקציה אמורה לתת אופצייה לתעד את פגישת צוות יומית.

- האפלקציה אמורה לתת אופצייה לתעד פגישת תכנון ספרינט (Sprint Planning) מוגדרים פריטי העתודה לאותו Sprint.

;

- האפלקציה אמורה לתת את אופצייה לתעד פגישת ניתוח ספרינט (Sprint Retrospective) קצרה להפקת לקחים מה- Sprint הקודם.
- האפלקציה אמורה לתת ניהול משתמשים שיכללו הרשאות שונות.
- האפלקציה אמורה לתת דיווח ומעקב על כל משימה .
- האפלקציה אמורה לתת אופצייה להעברת משימה ל Sprint אחר.
- האפלקציה אמורה לתת אופצייה להעברת דוח מצב על הפרוייקט.

רקע עם התייחסות למקורות (כעמוד אחד)

מבוא תוכנה אג'ילי

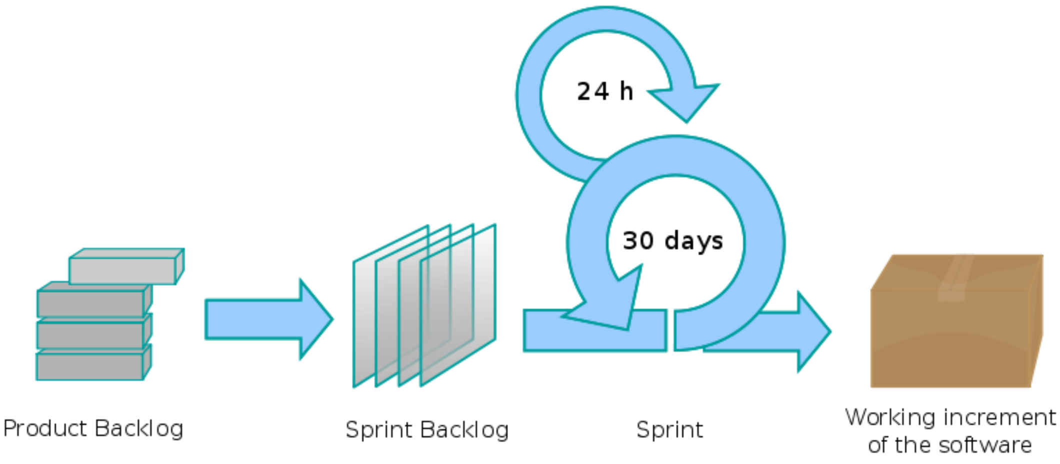
משמעות התואר Agile היא זריז, גמיש, והוא מתייחס ליכולתו של גוף להתאים את עצמו כראוי לסביבה משתנה. הניסיון המצטבר ביחס לניהול פרויקטי תוכנה מלמד שרצוי שגם תהליכי פיתוח תוכנה יהיו בעלי תכונות אלה. בפרט, התפיסה האג'ילית לפיתוח תוכנה מבוססת על הנחת העבודה כי תהליכי פיתוח תוכנה מאופיינים בשינויים רבים, ולכן, יש לבנות עבורם מנגנון ניהול המתמודד בהצלחה עם מאפיין זה. בנוסף, התפיסה האג'ילית שמה את הדגש על פיתוח תוכנה איכותית, הן מבחינת קיום דרישות הלקוחות והן מבחינת העדר באגים. רעיונותיה של התפיסה האג'ילית מיושמים באמצעות מספר מתודולוגיות פיתוח תוכנה אג'יליות, כמו למשל, Extreme Programming, Lean Software Development, Crystal, Scrum.

המאפיינים העיקריים של מתודלוגיה :
א) שביעות רצון הלקוחות
ב) קבוצה טובה עם אחריות גבוה יותר
ג) דרישות קל משקל, גמישות לשינויים מהירים
ד) מחזורים קטנים איטרטיבי
ה) רמה גבוהה של תקשורת
תהליך זריז מאפשר ללקוח לקבל תצוגה מהירה של המוצר וכמו כן מספקת מוקדם ולעיתים קרובות דבר זה נותן יתרון גדול שכן המשוב נעשה בזמן קצר ולא מחכה לסוף הפרוייקט כמו מודל מפל המים .
כמו כן דבר זה נותן ללקוח אופצייה לשינוי שכן ללקוחות יש סיכוי לבצע שינויים בדרישה אם הם חשים כי מה שהם רואים בכלל? לא תואם עם מה שהם מדמיינים. שינויים דרישות דינמי יתרון המקנה גמישות בשלביי הפיתוח.

מתודולוגיית Agile מספקת משלוח מהיר בסביבה דינאמית. זה הטבע מצטבר מספקת התנהגות חסכונית עבור פיתוח המוצר. מתודולוגיה זו לא רק מאפשר לספק אב טיפוס חסכוני, אלא גם מספק הפקטורינג גמישות מחדש. בסביבה תחרותית, העסק תמיד מתחרה עם עסקים אחרים על ידי שחרור מוצרים חדשניים בעולם מהיר דינמי, שיכול להיות נתמך לחלוטין מתודולוגיה זריזה.

;

תהליך פיתוח אג'ילי



1. תחזוקה של רשימת פריטי העבודה לביצוע, מסודרים לפי קדימויות, המכונה עתודת המוצר (Product Backlog).

2. השלמת מנה קבועה של פריטי עתודה בסדרה של איטרציות קצרות המכונות Sprint. משך כל Sprint הוא 4 שבועות, ובסיומו מסופקת תוכנה עובדת למשתמשים.

3. פגישת צוות יומית קצרצרה (עד 15 דקות) המכונה 'Stand up Meeting'. בפגישה מציג כל אחד מחברי הצוות את ההתקדמות, העבודה המתוכננת וקשיים אפשריים. הפגישה מתקיימת לרוב בעמידה.

4. פגישת תכנון ספרינט (Sprint Planning) קצרה שבה מוגדרים פריטי העתודה לאותו Sprint.

5. פגישת ניתוח ספרינט (Sprint Retrospective) קצרה להפקת לקחים מה- Sprint הקודם.

6. השיטה מיושמת בהנחיית Scrum Master שתפקידו העיקרי לסלק מכשולים המפריעים לצוות לעמוד ביעדי הSprint . ה-Scrum Master אינו ראש הצוות (מאחר שמדובר בצוות בהכוונה עצמית), אלא חוצץ בין הצוות לבין השפעות העלולות להפריע לו.

7. כדי לאפשר את יצירתם של צוותים בהכוונה עצמית, השיטה מעודדת ריכוז של כל חברי הצוות במיקום אחד, וכן תקשורת מילולית בין חברי הצוות ועם צוותים תומכים.

;

8. אחד מעקרונות המפתח של Scrum הוא הכרה בכך שאתגרים אמפיריים ביסודם אינם ניתנים לפתרון בשיטות מסורתיות המתבססות על חיזוי או תכנון. מכיוון שכך, שיטת Scrum מאמצת גישה אמפירית, המניחה שלא ניתן להבין או להגדיר את הבעיה במלואה מראש. במקום זאת, השיטה מתמקדת בשיפור יכולתו של הצוות לספק תוצרים במהירות ולהגיב לדרישות העולות תוך כדי התהליך.

תיאור הפרויקט

הפרוייקט אמור להיות מחולק לחלקים הבאים:

- שלב אפיון
- שלב הניתוח
- שלב עיצוב(Design)
- שלב המימוש

- שלב האפיון :האפיון יבוצע ע"פ דרישות מתודלוגיית Agile ניהול :
אפיון התוכנה הוא שלב מרכזי בפרוייקט תוכנה ובו מוגדרת מהות המערכת הנדרשת .
בשלב האפיון נגדיר מה אנו רוצים על פי ספרות ומאמרים שמדברים על Agile.
בשלב זה מוגדר ה-'מה'. לא מוגדר ה-'איך.'
תוצרים בפרוייקט שלנו : איפיון קצר של הפרוייקט.
- שלב הניתוח :
שלב הניתוח הוא השלב בו מפרטים את הדרישות שהוגדרו בשלב האפיון לאובייקטים ולתהליכים. עבור כל דרישה נגדיר אוסף תהליכים שהמערכת צריכה לבצע.
תוצרים בפרוייקט שלנו : מידול של הפרוייקט בעזרת UML .
(use case ,sequence diagram ,etc)

- שלב עיצוב(Design)
שלב העיצוב הוא השלב שבו כבר נכנסות לתמונה הארכיטקטורה של המערכת, הטכנולוגיה, השפה (פרוצדורלית/מוכוונת עצמים). בשלב זה מתבצעת החלוקה לאובייקטים במערכת, כל אובייקט/מחלקה מקבל את הפעולות - פונקציות או פרוצדורות שעליו לממש, נקבעת הטכנולוגיה שבה תמומש המערכת ובהתאם לכך גם משתנה העיצוב (חלוקה לפרוצדורות ב-C, למחלקות ב-JAVA וב-++C או הגדרת Interface-ים (ממשקים) ב-COM).
תוצרים בפרוייקט שלנו : מידול של הפרוייקט בעזרת UML.
(Sequence diagram,class diagram)
- שלב המימוש
שלב הקידוד והתיכנות הוא השלב בו לוקחים את תיק העיצוב ומממשים את הפעולות כפי שהוגדרו בשלב העיצוב. זהו השלב המרכזי והטכני ביותר שעם סיומו יש תוצר כלשהו שניתן לראות, לעבוד ולבדוק.
תוצרים בפרוייקט שלנו : קוד הפרוייקט .

חשיבות הפרויקט

לימוד/קריאה

- קריאת ספרים על Agile לצורך איפיון
- לימוד WPF לצורך מימוש UI
- לימוד Enterprise Entity לצורך מימוש שמירה ל בסיס נתונים

;

<u>כלים</u>

- Visual Studio 2008
- Sql Server 2008
- Enterprise Architect

- **לוח זמנים לביצוע הפרויקט** (פירוט ברמה של חודשים)
  - **איפיון** :חודש לאחר אישור ההצעה .
  - **ניתוח** :חודשיים אחריי אישור ההצעה.
  - **עיצוב:** 4 חודשים אחריי אישור ההצעה
  - **לימוד וקידוד:** 8 חודשים לאחר אישור ההצעה

- **רשימת מקורות ראשונית** (בגוף ההצעה צריכה להיות התייחסות למקורות. המקורות צריכים להיכתב לפי הכללים המקובלים בספרות המקצועית, כמפורט בנספח ב.)

- Addison Wesley - Lean Software Development - An Agile Toolkit
- Agile Software Development With Scrum
- Agile Web development with Ruby on Rails

;

9 מקורות

רשימת מקורות ראשונית (בגוף ההצעה צריכה להיות התייחסות למקורות. המקורות צריכים להיכתב לפי הכללים המקובלים בספרות המקצועית, כמפורט בנספח ב.)

- Addison Wesley , Lean Software Development, Mary Popendieck, May 08, 2003
- Ken Schwabe ,Agile Project Management with Scrum Microsoft Press 2004
- Dave Thomas, David Heinemeier Hansson,Agile Web development with Ruby on Rails ,The Pragmatic Programmers,December 2005