|  |  |  |
| --- | --- | --- |
| **Name** | **ID** | **Email** |
| Taimor fares | 308137355 | [Fares.taimor@gmail.com](mailto:Fares.taimor@gmail.com) |
| Miras Safadi | 315891549 | [safadimiras@gmail.com](mailto:safadimiras@gmail.com) |
| Rani Hassan | 315060103 | [ranihassan8@gmail.com](mailto:ranihassan8@gmail.com) |
| Fares Jaraisy | 208451682 | [jaraisy.fares@gmail.com](mailto:jaraisy.fares@gmail.com) |
| Date Of Submission | 10/12/2018 | |

**שאלה 1:**

**תארו איך השתמשתם במודל Use Case בעבודתכם. הסבירו את מקומו ותרומתו של המודל לתהליך פיתוח המערכת בעזרת דוגמאות פרטניות (ספציפיות) מהמערכת "OBL".**

אופן העבודה שבו החלטנו לעבוד במטלה הראשונה והשנייה הוא עבודה קבוצתית, מהסביה ששלב התכנון הוא השלב החשוב היותר בפרויקט כי עם תכנון מושלם בא קוד איכותי ומדויק.

ולכן במפגש הראשון של הצוות הדפסנו את הסיפור עבור כל סטודנט והחלטנו שכל אחד קורא את הסיפור לבד ומסמן את השחקנים(actors) ובנוסף, מסמן את הפעולות הנדרשות מהמערכת כלומר ניתוח דרישות.

לאחר מכן ישבנו כלל חברי הצוות והתחלנו לדון יחד במודל ה-UC. וזה התחלק לשלבים:

1. הגדרת כלל השחקנים: כל אחד הציג את השחקנים שהוא סמן ומתוך כלל השחקנים שנבחרו בחרנו את השחקנים הנכונים שמשתתפים במערכת (אלה שמבצעים פעולות במערכת או משפיעים על המערכת שלנו), למשל חלק מחברי הקבוצה חשבו שיש לצרף את "מערכת ניהול העובדים" כשחקן, לאחר דיון הוצגו הסיבות למה לא לכלול את השחקן הנ"ל והגענו להסכמה שהוא לא שחקן במודל UC שלנו (כי היא מערכת חיצונית שלא קשורה למערכת שלנו).
2. ניתוח דרישות: עברנו על כלל הדרישות והוספנו אותם למודל ה-UC שלנו בהתחלה הוספנו את כלל הדרישות של כלל חברי הקבוצה, כולל חיבור הדרישה לשחקן שמשפיע עליה באופן ישיר או משני

(primary actor, secondary actor),

1. בניית טבלת שחקנים דרישות: בנינו טבלה שבה מצויינים כלל השחקנים ועבור כל דרישה דנו יחד עבור כל שחקן בנפרד האם הוא מבצע את הדרישה הנ"ל או הוא שחקן משני או שהוא לא מבצע את הדרישה, זה מאוד עזר לנו להבין את הסיפור ואת כלל הפעולות שכל שחקן מבצע.

מודל ה-UC עזר לנו מאוד בהבנת הסיפור מנקודת מבט של שחקנים-דרישות, כלום הצלחנו להבין מה דרוש מהמערכת, וכן כאשר בנינו את ה-UC הצלחנו לראות דרישות מיוחדות שלא היו ניתנות להבנה במהלך קריאת הסיפור, כאן בעצם הבהרנו לעצמנו כמה דברים:

1. מה המערכת והמשתמש עושים או יכולים לעשות.
2. זיהוי מקרים מיוחדים שלא ניתן לזהות מקריאת הסיפור בלבד.
3. הבהרת דרישות המערכת.
4. הבהרת סדר הפעולות שמבצע שחקן במערכת כדי לבשיג את מטרתו (וראינו את זה ב flow of events).
5. רצף האירועים מרגע שהשחקן דורש ועד להשגת המטרה.

מודל ה-UC יעזור לנו גם בהמשך העבודת, ניתן בעזרתו לחלק את העבודה בתוך הצוות לחלקים שכל אחד יוכל בהמשך במיוחד בכתיבת הקוד להתעסק בפעולות שקשורות לעצמן (מחלקות) וגם להבין את הרצון והנכונות לחלק בצורה מסוימת (למשל השאלת ספר והארכת ההשאלה אם זה ידני או אוטומטי וההזמנה, אלה דוגמאות לפעולות קשורות מאוד וזהות)

אחד הכלים החשובים ביותר במודל ה-UC לדעתנו הוא כמות השחקנים והדרישות מעידות על גודל המערכת וסיבוכיות המערכת שאנו רוצים לבנות, למשל זה שאנו כבר יודעים כמה שחקנים ומה כל שחקן מבצע ואת השתלשלות האירועים (flow of events) כבר בנקודת הזמן הזו התחלנו לחשוב על המימוש שלנו.

למשל, ביצירת מנוי חדש החלטנו שהמערכת תתן לו אוטומטים שם משתמש שהוא ת"ז של אותו משתמש בכדי שהשדה השדה יהיה מפתח במערכת.

ועוד כמה דוגמאות, שגם עבור ספר החלטנו שהמק"ט שלו יהיה המפתח ולא שם הספר, כי לדעת לקשר בין ספר ספיציפי לבין מנוי שמזמין או משאיל אותו.

לסיכום, היינו אומרים ששלב התכנון הוא השלב הקריטי במערכת, ובנוסף הוא השלב שפחות מתחברים לעסוק איתו (כי כולנו רוצים כבר לתכנת ולכתוב קוד), אך במחשבה מעמיקה הבנו את ההשפעה הגדולה והעיקרית של השלב הזה, בשלבי התכנון אנו קובעים את אופן התנהלות המערכת (מערכת ידידודית למשתמש, מסובכת או יעילה ועוד הרבה דברים), ולכן החלטנו להשקיע מאמץ רב וחשיבה רבה בשלב הנ"ל.

**שאלה 2:**

**תארו בפירוט איזה מרכיבים פונקציונליים ספציפיים של האופיין של מערכת "OBL" (כפי שמתואר במסמך semester project”" ) לא הצלחתם לבטא בעזרת מודל UC ?**

במהלך בניית ה-UC נתקלנו בכמה דרישות שלא הצלחנו לבטא אותם בדיאגרמה שלנו, מהסיבה שאין שחקן מסויים מבצע את הדרישות/פעולות האלו אלה הן דרישות פנימיות שעל המערכת לבצע אותם:

1. הפקת דוחות שהמנהלת דורשת: לא מובן מי השחקן שהוא מכין את הדוחות והאם הדוחות מוכנים או לא! זאת דרישה שהמערכת הפנימית צריכה לבצע ברגע שהמנהלת דורשת לקבל דו"ח מסויים.
2. יכולים להיות מספר משתמשים המחוברים בו זמנית למערכת: אין אפשרות לבטא את זה או לבדוק האם זה אכן מתקיים או לא.
3. אותו משתמש לא יכול להיות מחובר למערכת בו זמנית יותר מפעם אחת: דרישה שהיא מובנת והגיונית אך קשה לבטא אותה, מי השחקן שיוזם אותה? מי בודק? וכמובן זאת דרישה שהמערכת הפנימית צריכה לבצע בשלב בניית המערכת עצמה.
4. שמירת היסטוריית ההשאלות: איזה שחקן שומר את ההיסטוריה? האם הספרנית שומרת אותה בעת ביצוע השאלה? אם כן לכמה זמן? והאם היא יכולה למחוק את ההיסטוריה הנ"ל? כמובן שהמערכת הפנימית צריכה לתעד ולשמור הכול ללא אפשרות מחיקה או שינוי.

**שאלה 3:**

**בתשובות 1 ו-2 תיארתם יתרונות ומגבלות שונות של מודל UCהציעו דרכים להתגבר על המגבלות שציינתם, ונמקו למה הצעותיכם נותנות מענה למגבלות אלה. הסבירו את תרומתה של הגישה שאתם מציעים כאן לפתרון אותן המגבלות שתיארתם תוך התייחסות ישירה למערכת "OBL" ובסיוע דוגמאות פרטניות (ספציפיות) מהמערכת.**

אחת המגבלות שנתקלנו בה היא היכולת לבטא דרישה במודל, למשל צפוי שמנהלת הספרייה תבקש לקבל דו"חות נוספים שדומים באופיים אבל שונים בתכולתם. ולכן נדרש לבנות את המערכת בצורה גמישה כך שלשם הפקת דו"חות חדשים תידרש עבודת פיתוח מינמלית, לא מובן איך ניתן למדל את זה ואיך ניתן לבדוק האם הדו"ח שמתקבל הוא התקבל בצורה יעילה, ולכן כדי להתגבר על בעיה זו הוספנו הערה בדרישה קבלת דו"חות כדי להסביר היטב את הבקשה שלנו.

בנוסף ישנן פעולות פונקצונאליות ופעולות שקוראות ברצף מסוים לפי הדרישות ובמודל ה-UC קשה מאוד לתאר אותו ולכן הפתרון הוא בעת ביצוע ה-Activity Diagram, שבה אנחנו יכולים לתאר בדיוק ונכונות רבה יותר את סדר השתלשלות האירועים ואת הסדר שבהן הן קורות, למשל בעת יצירת מנוי חדש שהדרישה היא שהספרנית תיצור את המנוי עבור קורא מזדמן, היכולת לתאר את האירוע קל יותר בעזרת Activity Diagram, כאשר ניתן להציג את סדר ההתרחשות במדויק עבור כל שחקן.