| | | | | |
|---|---|---|---|---|
| **מספר קבוצה:** | 8 | | | |
| **חברי הקבוצה:** | **שם מלא:** יובל הים | **ת.ז:** 314989484 | **מייל:** Yuval.Hayam@e.braude.ac.il | |
| | **שם מלא:** נדב דרעי | **ת.ז:** 203327754 | **מייל:** Nadav.dert@e.braude.ac.il | |
| | **שם מלא:** הדר אילוז | **ת.ז:** 312588601 | **מייל:** Hadar.lluz@e.braude.ac.il | |
| | **שם מלא:** מטר אסף | **ת.ז:** 209401439 | **מייל:** Matar.Asaf@e.braude.ac.il | |
| | **שם מלא:** ידין אמסלם | **ת.ז:** 316325570 | **מייל:** Yadin.Amsalem@e.braude.ac.il | |
| | **שם מלא:** ליאור קריש | **ת.ז:** 318267242 | **מייל:** Lior.Karish@e.braude.ac.il | |
| תאריך הגשה סופי: | 5.5.2021 | | | |

**1 .תארו את תהליך הניתוח/ תכן ראשוני שביצעתם למרכיב: ביצוע בחינות (ממוחשב) .**
**קווים מנחים לתשובה : פרטו מה הם השאלות/הפרטים שהתייחסתם אליהם בתהליך הניתוח והחשיבה , וכן התייחסו לקשרים ולמעברים.**
**1.ממודל** case-Use **למודל תהליכי המיוצג בעזרת** Diagram Activity **,**
**2 .ממודל התהליכים הדינמיים לקראת מימושו בתוכנה.**

**<u>תשובה:</u>**
כחלק מתהליך הניתוח הראשוני שלנו שכלל בניית UC Diagram והתבסס על גזירת דרישות המערכת מסיפור הרקע שהוצג לנו. ניסינו להבין מי הם השחקנים במערכת, מה הגדרת התפקיד והפעולות שהם יכולים לבצע.
בנוסף, בפירוט ה- flow of events, תיארנו את התהליך שמבצע השחקן:

1. סטודנט - בעת פתירת המבחן, הוא נדרש לקבל ולהזין קוד למבחן ולהזין תעודת זהות. דבר זה דורש מסך ייחודי שבו יצטרך לבחור לטובת ביצוע הבחינה הממוחשבת.
נעזרנו בפירוט התרחיש ב- flow of events כדי שנוכל לענות על שאלות שעלו לפני בניית המידול כגון: האם יש תנאים מקדימים לביצוע בחינה מלבד קשרי extend & include המחוברים לפעולה/ האם יש "בועה"- פעולה נוספת, שצריכה להתקיים לפניה. ובכן, על מנת שנוכל לבצע בחינה, אנחנו צריכים שתהייה קיימת במערכת בחינה. רק בהינתן קוד בחינה קיימת נוכל לבצע את המרכיב "ביצוע בחינות" (ממוחשב).
2. מורה - פירטנו את תהליך יצירת המבחן, עבורו נדרשנו לבנות דיאגרמת activity.עקבנו אחרי ה flow of events של הפעולה "creat exam" בדיאגרמת UC.

המידול בדיאגרמת ה-UC היה כללי מדי ולא איפשר תיאור מפורט של פעולת המערכת. לכן, כל אחד מהתרחישים שהגדרנו בUC הוצג בדיאגרמת הActivity בפירוט. בעזרת טכניקת ה-Swim Lanes הצגנו בדיאגרמה את פעילויות האובייקטים במערכת.
במהלך המידול לדיאגרמת ה Activity גילינו שישנם דברים שלא חשבנו עליהם במהלך מידול UC ולכן התחדדו לנו שאלות/תהיות שבעזרתם יכולנו לתכנן בביטחון את הליך המימוש, דוגמה לשאלות שהתחדדו:
בהיבט בניית בחינה ע"י מורה:

- היות ובחינה משתייכת למקצוע וקורס מידלנו כך שהמורה מבצעת בחירה של מקצוע ולאחר מכן בחירת קורס.
- פעולת "create question"- איפשרנו הוספת שאלות באמצעות לולאה (ע"י צומת החלטה) שבודקת האם יש מספיק שאלות והאם אפשר להתקדם בתהליך יצירת המבחן.
- מהן הפעולות שחייבות להתבצע ומהן הפעולות האופציונליות. הגדרנו את זמן המבחן בדקות ומתן ניקוד לכל שאלה בעת הוספתה, אלו דרישות RF שהגדרנו ולכן הכנסנו אותן בדיאגמה בעזרת fork ו- join.
- פעולת הוספת הערה, היא אופציונליות. לכן, יצרנו מסלול חלופי לפעולה, שלא מחייב מעבר.

בהיבט ביצוע בחינה ממוחשבת:

- שקלנו איזו פעולה של הסטודנט תגיב להתחלת תרחיש ביצוע בחינה ממוחשבת. ראשית עליו לזהות כפתור ייעודי לכך שיפתח מסך. כעת, רצף האירועים יתבצע ע"י מילוי השדות הרלוונטים בשדות החובה ואישורם ע"י הcontroller, במידה ונכשל מסיבה כולשהי, נאפשר לו להזין שוב בלולאה(בעזרת צומת) עד שפרטיו הנכונים יאומתו.
- מרגע זה הפעלת הטיימר לבחינה מתחיל וממלא את טופס הבחינה. הדילמות שעלו, עסקו בין היתר בתרחישים שיקרו במידה והסטודנט לא הגיש את הבחינה בזמן- האם הטופס ישמר אוטומטית או שהסטודנט ייחשב כלא נבחן.
- מה יקרה במידה ונשארו שאלות שהסטודנט לא ענה עליהן והגיש את הבחינה, האם נרצה לאפשר זאת או לא.

בכדי ליצור את דיאגרמת ה sequence של הפעולה "create exam" עקבנו אחר דיאגרמת הactivity בסדר הפעולות וחשבנו על המימוש שנרצה לכתוב בקוד. חשבנו לאן נחבר כל פעולה, איזה פעולות קשורות לboundry/controller/entitiy מסוים.
ראשית, היות והמורה הוא השחקן שברשותו ההרשאות והאחריות לפעולה אזי רצף הפעולות יתחיל ממנו.
עבור כל פעולה חשבנו מה צריך לקרות- כלומר, האם נרצה שיופיע טופס מסוים במסך, אילו מהפונקציות שמידלנו בclass יקדמו אותנו לפעולה הבאה של השחקן/ המערכת, כמו שמירת נתונים במסד או לוגיקה.
חשבנו האם כחלק מהפעולות שביצענו והפעלת פונקציות נוצר ערך מסוים שנרצה לשמור אותו כתכונה של יישות.
השאלות הללו, עזרו לנו לבחור את ה boundry, controller, entitiy המתאימים מה-class Diagram שבנינו.
מספור הפעולות מתבססות על פי רצף הפעולות של דיאגרמת ה activity.
שלב הדיאגרמות במטלה זו "עשה לנו סדר" מבחינת העברת הסיפור לקוד. כלל ההתחבטויות וההתלבטויות שקרו בשלב זה סייעו לכך שנוכל להגיע לשלב המימוש עם תוכנית אלגוריתמית מסודרת והבנה טובה יותר של דרישות המערכת. דבר שלמעשה מקדם אותנו משלבי הבעיה אל שלבי הפתרון.

**2 . בהרצאה הוגדרה** Reusability **כתכונה של תוצר של תהליך הפיתוח אשר משקפת את היכולת לבצע** reuse **בהקשר לתוצר זה. בהתאם להגדרה זו, תארו בדיוק (ובהתייחסות ספציפית) ובפירוט איך באות לידי ביטוי 3 דרישות ליישום מוצלח של** Reusability **בהקשר של אותם מרכיבים שלא אתם כתבתם או תכננתם ובחרתם לשלב במערכת שלכם באמצעות** Reuse **,תוך התייחסות בדוגמאות ספציפיות לא 'עקרוניות' או 'כלליות' לדרישות הפונקציונליות של המערכת שתכננת.**
**התייחסות ספציפית בהקשר זה = התייחסות ל מרכיבים ספציפיים קונקרטיים (לא גנריים) מתוך התיאור המילולי הראשוני של פעולת המערכת שאתם מפתחים מתחילת הסמסטר.**
**לא כולל תהליך** Login **או זיהוי משתמש ).**
**אם יש מי מ -3 הדרישות הנ"ל אשר לא באה לידי ביטוי ב -**reuse **שביצעתם - הסבירו את הסיבה לכך.**

**<u>תשובה:</u>**
העיקרון של שימוש חוזר, מאפשר למתכנת לשלב קטעי קוד ומרכיבים אחרים שלא הוא כתב או תכנן, ובכך מקל על המתכנת כיוון שהמתכנת משתמש בקוד מוכן. בתרגול נחשפנו לפרויקט OCSF (Object Client Server Framework) שבאמצעותו מימשנו צ'אט בין שרת ללקוח (Simple Chat). בפרויקט אנו משתמשים ארכיטקטורת Client-Server.
נפרט על 3 דרישות ליישום מוצלח של reusability ששילבנו במערכת שלנו:

1. <u>זמינות</u>OCSF - קיים בשוק והוא מוכר וידוע. אצלנו בפרויקט השתמשנו בו בהנחה שהוא כבר נבדק וקיים כקוד שניתן להשתמש בו. קטע הקוד נכתב בעבר וזמין עבורנו ולכן עונה על תכונת ה- availability, והתאמנו אותו בקלות רבה למערכת שלנו כך שיתאים למשימה שרצינו שיבצע עבורנו ובצורה זו באה לידי ביטוי תכונת הגמישות (flexibility) של הקוד של פרויקט ה .OCSF
2. <u>גמישות-</u> ברוב מערכות OCSF קיימת גישת שרת- לקוח ויצירת מחלקות אבסטרקטיות שממשות תהליך שקורה בכל מערכת כזאת. כלומר, הכוונה לחיבור בין השרת ללקוח ע"י העברת מסרים בינהם. בעצם ניתן ליצור מחלקות שיורשות את המחלקות האבסטרקטיות ולממש אותן לצרכנו במערכת שלנו CEMS.
3. <u>הבנה-</u> קיבלנו כחלק מהכלים של הקורס את הידע איך לבצע את התקשורת בין שרת-לקוח בעזרת המחלקות ואיך לשלב אותו בתוך ה OCSF כך שנוכל ליצור את הפרויקט שלנו ולממש אותו.

**3 .א. <u>הערכה כללית :</u>**

**1 .מהם היתרונות של מודל** UML **כעזר לתהליך התכנון?**

**(i) הסבירו איך מתקבלים (מתממשים) היתרונות שציינתם.**

**(ii) ציינו דוגמה אחת קונקרטית ממוקדת (לא כללית ולא Login ) מתוך תהליך הניתוח והתכן שאתם בצעתם לשימוש מועיל ב-**UML **תוך תיאור והתייחסות ספציפית למרכיבים של מערכת "CEMS " שתכננתם ומידלתם.**

**<u>תשובה:</u>**

**1-(i) -**למודל UML יתרונות רבים כעזר לתהליך התכנון. כאשר רוצים לממש מערכת גדולה שמורכבת מהמון חלקים קשה לבנות אותה ללא תכנון מוקדם. התכנון אמור להיות מפורט, בייחוד שמדובר במערכת כה גדולה עם המון מחלקות ופונקציונליות רבה. בעזרת UML שכוללת דיאגרמות רבות, ניתן לפרק את המערכת לתתי נושאים, מחלקות, פעולות, פונקציות ותכונות ובכך יכולה לתרום לצוות המתכנתים להבין יחד את בניית המערכת והקשרים בתוכה.

יתרון נוסף לUML הוא אחידות ותאימות. כאשר ניתן לראות את התכנון כתוב, נוצרת הבנה משותפת ובהירות לגבי מה צריך לבצע. כך כל מתכנת יודע בבירור מה כוללת המערכת ומהם הקשרים. כלומר, יש תאימות מוחלטת בין המתכנתים בצוות, שכן הכל כתוב ורשום.

בנוסף, בדיאגרמות UML, לכל דיאגרמה יש את משמעות ייחודית למידול שלה, שכן כל מתכנת בצוות שלנו מכיר.

- בעזרת מודלים שונים של UML, יצרנו דיאגרמות של תהליכים שיתרחשו במערכת, ואת האובייקטים הקיימים בה. בעזרתם ניתן להציג ללקוח בצורה ויזואלית וברורה את פעולות המערכת בתרשים, שנבנה על פי מסמך הדרישות שקיבלנו ממנו.
  אנו יוצאים מנקודת הנחה כי בעולם האמיתי ללקוח אין בהכרח מושג בכל הקשור למושגי התכנות ולכן בעזרת מידולי UML הוא יכול לקרוא את הדיאגרמה בקלות יחסית, ולבדוק האם המערכת עתידה למלא את דרישותיו בצורה מדויקת כפי שהתכוון.
- המודל עזר לנו להביט קדימה לשלבי איטרציות הפיתוח- המערכת שלנו ממודלת בעזרת עקרונות OOP, כך שיכולנו לצפות בניית מחלקות ופונקציות שיעזרו לנו למנוע שיכפול קוד. דוגמה מובהקת לכך היא ע"י שימוש בהורשה.
- מידול class diagram עזר לנו להבין את הפונקציונליות שצריכה ומעניקה לנו כל מחלקה בserver ו בclient ואת הקשר ביניהן. ע"י מידול זה היה לנו קל יותר לגשת לכתיבת הקוד, ליצור משתנים של המחלקות, את המחלקות עצמן, מתודות, ולהבין את הקריאות בכל מחלקה למתודות אחרות (ובעצם ליצור חיבור בין המחלקות). בנוסף, בזכות דיאגרמת ה class הצלחנו להתפצל לקבוצות קטנות יותר ולכתוב קוד לא בהרכב מלא ובכך לחסוך זמן (כיוון שכבר ידועים הקשרים, המחלקות והמשתנים).

**(ii)- דוגמה ספציפית לשימוש מועיל ב UML:**

- מידול class diagram המתאר את כל האובייקטים שעליהם נשמור מידע במערכת, היישויות הללו הם מחלקות שישמרו בהם מידע, ומחלקות כמו סטודנט ומורה יהיו טפסים שישמשו אותם במערכת.
  התרשים עזר לנו להבין את המערכת כפי שהלקוח רואה אותה: מהם הטפסים שהוא יכול לראות, איזה כפתורים במסך יהיו שייכים לכל סוגי המשתמשים ומהם השדות והכפתורים בכל טופס.
  דבר זה סייע מאוד בחשיבה שלנו על ממשק אדם מחשב מצב הלקוח, מה יהייה לו נוח ויזואלית ידידותי לתפעול.
  לדוגמה, בתרשים sequence ליצירת מבחן, התרשים היה מועיל בהבנת הflow של הקוד, באילו מחלקות ומסכים נצטרך לעשות שימוש ואת הקשרים שביניהם. מהו המידע שצריך לעבור בין השרת ללקוח, וכיצד נעביר מידע זה.

- דוגמה נוספת, בתרשים ה class ניתן לראות כי בנינו מחלקת יחס בין Active Exam לבין Student. שמה של המחלקה הוא Exam Of Student ומטרתה- לחבר את היישות מבחן ליישות סטודנט. איפיון זה מסייע לנו בהבנה אילו טבלאות נצטרך לממש במסד הנתונים.

**3.א.**
**2. ציינו קשיים הנובעים מחסרונות של** UML **שנתקלתם בהם.**
**גם כאן התייחסו ספציפית לתהליך שבצעתם לפיתוח מערכת זו.**

**<u>תשובה:</u>**
עבור מערכת CEMS מספר הדרישות הפונקציונליות של המערכת גדולה יחסית, כל אחד מחברי הצוות עבד על ניתוחים שונים במקביל וזאת לאחר שיישרנו קו בפגישה צוותית על קו מחשבה וראייה אחידה לטובת תאימות.
נוצר מצב של חוסר עקביות לגבי פרטים קטנים מכיוון שכל שינוי דורש עדכון של שאר האנשים /צוותים, לכן מצאנו דרך בה חילקנו את בניית הUML-ים לאיטרציות בהן נבצע פגישה צוותית וניתן פידבק על העבודה שנעשתה עד כה וכך יכלנו להתקדם בצורה אחידה.

השימוש במתודולוגית UML לא נותן מענה מלא לצורכי תהליך ה DESIGN אך הוא נותן מענה עבור קשרי המחלקות השונות המטפלות באירועים משותפים עבורם .
דוגמא שממחישה את הקשרים היא יצירת מבחן ובה הקשר בין מחלקת מבחן למחלקת שאלה בא לידי ביטוי בכך שמחלקת מבחן משתמשת בשאלות שיצרה מחלקת שאלה .
דוגמא לחסרונות של הUML היא בהיבט תזמון, כלומר לא ניתן לסנכרן באופן מלא בין מורה שיכולה לבקש הארכת זמן עבור מבחן כלשהו ובינתיים המבחן הסתיים עד שיגיע אישור המנהלת ,ובכך הבקשה הופכת להיות לא רלוונטית.

קושי נוסף הוא תכנון מסד הנתונים של המערכת: על מנת לנתח ולאפיין את מסד הנתונים של המערכת יש להחליט אילו טבלאות יש לבנות ומה המפתחות והקשרים ביניהם, לעיתים נדרש להוסיף ולשנות דברים תוך כדי המימוש עצמו והדבר מקשה על בניית מסד הנתונים מראש.
המידול שביצענו באמצעות Class Diagram בנושא זה הינו חלקי ולא מנורמל(כלומר ייתכנו כפילויות ).
חשבנו על פתרון של שימוש דיאגרמת ERD שמסייעת בניתוח ואפיון יעיל של טבלאות ה-DB. ואם היא בנויה היטב היא פותרת את נרמול הטבלאות.

קושי נוסף היה בריבוי המודלים שיצרנו- כאשר התחלנו לנתח את הפרויקט, התרשים הראשון שנדרש לבצע היה ה-Use Case . בהמשך יצרנו את ה-Class Diagram & activity ושמנו לב לחוסר תאימות עם הדיאגרמה של ה-Use Case. כאשר הגענו ל-Sequence Diagram וניתוח כל תרחיש בנפרד שמנו לב לחוסרים ב-class למתודות ומשתנים שונים שדרשו השלמה.

**3.ב. <u>ניתוח ודיון:</u>**
**בהתאם לניסיון שרכשתם במהלך העבודה על מטלה זו, תארו אפשרויות לשינויים ושיפורים במתודולוגית** UML **אשר נותנים מענה לחסרונות שנתקלתם בהם במהלך ה-**design **שביצעתם בפרויקט שלכם. הסבירו את תשובתכם תוך תיאור דוגמה ספציפית )כולל שמות של רכיבים, לא כולל** Login **)מתוך עבודתכם.**

**<u>תשובה:</u>**
כדי לשפר את מתודולוגית UML, ההצעה שלנו היא היכולת לבצע את Use Case Diagram במקביל ל Activity Diagram. ביצוע רישום של תהליך רצף האירועים שקיים בכל בועה דרש השקעה רבה ונכתב אופן כזה שהוא מפורט ברמה שניתן לבנות באמצעותו את דיאגרמת הActivity.
ההצעה שלנו היא להשתמש בפיצ'ר שיהפוך את ה Flow of events ישירות לדיאגרמת activity, כך ש:
ששחקן מסוג כלשהו יהפוך לswimlane וכל שורה ב Flow of events תהווה action, ושורות עוקבות יקושרו ע"י חץ. ואז גם If ו-else יתפצלו על ידי decision node, וכן הלאה.
ניתן יהיה כמובן לערוך ולדייק את התרשים שייווצר, אך התשתית תהיה מוכנה ותזמן עבודה מיידית על תרשים הactivity, דבר שיועיל מאוד בתהליך העבודה.

באותו אופן היינו מציעים את השימוש בפיצ'ר זה לבניית Sequence Diagram שתתבסס על תרחיש מוצלח בActivity וכמובן שזוהי לא הדיאגרמה השלמה אך היא אכן תוכל להוות עבורנו בסיס שעליו ניתן יהייה לעבוד ולהוסיף פונקציונליות מתרשים הclass.

בנוסף, במתודולוגית העבודה שלנו עבדנו על פי מודל "מפל המים". כלומר, ביצוע של דיאגרמה אחת מהווה תלות בביצוע דיאגרמה אחרת. לדוגמא, כדי לממש תרשים sequence יש צורך בתרשימי class & activity תחילה.
בעת כתיבת כל תרשים עולות הרבה תהיות אשר באות לידי ביטוי גם בתרשימים שהתבצעו לפני.
כל שינוי קטן בתרשים ה activity וה sequence בא לידי ביטוי גם בצורך לתקן ולתאם את ה flow of events של דיאגרמת ה use case.
היינו משפרים בכך שהיינו מבצעים את העבודה בשיטה שונה במקצת, שיש אופציה לעבוד על דברים בו זמנית בכך שנשתמש בפיצ'רים שהצענו לעיל.