שאלות פתוחות מטלה

יש לכלול את השאלות במסמך התשובות.

1. תארו את תהליך הניתוח שביצעתם למרכיב: "הפקת דירוגים" באמצעות המערכת האנליטית. פרטו מה הם השאלות/פרטים שהתייחסתם אליהם. בתשובה התייחסו לקשרים ולמעברים 1. ממודל Use-case למודל תהליכי המיוצג בעזרת Activity Diagram, 2. ממודל תהליכים למימושו בתוכנה.

1. דיאגרמת Use-case מתארת קשר בין שחקנים הפועלים במערכת בתרחישים שונים, כך הבנו כי "הפקת דירוגים" הוא תרחיש שעובד מחלקת השיווק לוקח חלק בו, בעזרת תרחיש זה ניתן לבצע את תרחיש "הפקת מבצעי מכירות" שחלק מרכזי בו נעזר בדירוגים המופקים.
בעקבות זאת הסקנו כי בשלב האימפלמנטציה נצטרך לתת התייחסות לצורך של הפקת הדירוגים,שיהיו זמינים לאיש מכירות, וכתוצאה מכך הוספנו ב- Use-case, בחלק ה- Logic של המערכת תרחישי עזר.
לאחר מסקנות אלו ניתן לתת ביטוי לסדר הפעילויות ולתאר זאת בעזרת מודל activity.
מה-activity יעלו השאלות באיזה שלב הדירוגים מופקים ואיזה חלק במערכת מפיק אותם וכיצד נתונים אלו באים לידי ביטוי.
ובכך נסיק כי נצטרך למשוך מהשרת את הדירוגים לפני שניצור מבצע מכירות ואיך הנתונים יבוטאו במסך.
דיאגרמה זו תציג לנו בבירור בזמן מימוש המסך "הפקת מבצע מכירות" את הפרטים שהוא יכלול והמידע שנצטרך להציג על המסך.

2. בהרצאה הוגדרה ***Reusability*** כתכונה של תוצר של תהליך הפיתוח אשר משקפת את היכולת לבצע reuse בהקשר לתוצר זה. בהתאם להגדרה זו, תארו יישום של 3 התכונות המאפשרות לכם לשלב במערכת MyFuel שאתם מפתחים קטעי קוד ומרכיבים אחרים שלא אתם כתבתם או תכננתם. תארו בדיוק (ובהתייחסות ספציפית) ובפירוט את התכונות המאפשרות Reuse של אותם מרכיבים אשר בחרתם לשלב במערכת שלכם, תוך התייחסות בדוגמאות ספציפיות (לא 'עקרוניות' או 'כלליות') לדרישות הפונקציונליות של המערכת שתכננתם (התייחסות ספציפית בהקשר זה = התייחסות למרכיבים ספציפיים מתוך התיאור המילולי הראשוני של פעולת המערכת ששאתם מפתחים מהתחלת הסמסטר. לא כולל תהליך זיהוי משתמש).
אם יש מי מ-3 התכונות הנ"ל אשר לא באה לידי ביטוי ב-reuse שביצעתם - הסבירו את הסיבה לכך.

2. בתהליך הפיתוח באה לידי ביטוי עיקרון ה-"reusability"המקיים את התכונות גמישות,נגישות,וניתנות של המערכת להבנה.

לדוגמה בפרויקט שלנו:

- השתמשנו ב-OCSF שזה הוא framework של שרת-לקוח. נראה כיצד באה לידי ביטוי תכונת ניתנות להבנה. OCSF מקיים את התכונה "ניתנות להבנה" בזה שה-framework מחולק לשני חלקים של שרת ולקוח ברגע שמממשים את המחלקה האבסטרקטית מיד ניתן לדעת מה הקוד החסר ואיך להתאים אותו עבור המערכת. בקוד שלנו תכונה זו מתבטאת בכך שמימשנו בעזרת ה-framework פעולות של משיכת מידע מהשרת לדוגמה למשוך את המוצרים הנמכרים במערכת.

- השתמשנו ב-JDBC המתאר את תכונת הגמישות, תכונה זה באה לידי ביטוי על ממשק JDBC בזה שניתן להשתמש בשימוש חוזר בקטעי קוד לחלקים שונים במערכת, לדוגמה שמירת לקוח חדש ב-DB ושמירת דו"ח רבעוני חדש ב-DB מתבצעים ע"י פקודת insert של database controller .

- השתמשנו באובייקטים של javaFX ספרייה זו היא חלק מספריות ה-utility של java ולכן התקיימה תכונת הנגישות, מפני שעשינו שימוש חוזר בממשק באובייקטים שהספרייה מציעה אובייקטים אלו הם קטעי קוד מוכנים מראש הנגישים לכולם ואין צורך לכתיבת קוד מחדש למשל בממשק שלנו עשינו שימוש חוזר ב-label,gridpane,vbox וכדומה.

3. א. הערכה כללית:

1. מהם היתרונות של מודל UML כעזר לתהליך התכנון?

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

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

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

ב. ניתוח ודיון:

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.

3. א. מידול בעזרת UML כלי מרכזי בתהליך התכנון בפרויקט.

היתרונות במודל ה-UML הם:

- זהו תהליך שאינו צורך זמן תכנות, כדי להבין מערכת שמעוניינים לבנות לא צריך להתחיל ולבנות חלקים ממנה, מספיק להבין את הקשרים ולהיעזר בדיאגרמות שונות .
  לדוגמה בפרויקט שלנו עשינו במלואו Class ו-Usecase לכל התרחישים הזמן שלקח לנו לעשות את הדיאגרמות הוא קצר יותר מהזמן שהיה לוקח לנו לבנות אבטיפוס מלא.
- מגדיר את גבולות הגזרה של המערכת, כלומר הבנו מה המערכת שלנו צריכה לכלול ומה מתבצע בגורמים חיצוניים.
  לדוגמה אנחנו יודעים שמערכת שלנו אינה צריכה לממש את התשלום, וזה מתקבל ישירות מ-Usecase היות ולא יצרנו תרחיש כזה.
- יוצר הבנה רחבה של המערכת. יוצר חיבור בין הדרישות השונות בסיפור למבנה המערכתי ומעלה דרישות נוסופות שאינם נכללות בסיפור בגלל הצורך העולה. לדוגמה בסיפור קיימת דרישה להוספת לקוח למערכת, מתהליך המידול הבנו כי בעת הוספת הלקוח המערכת צריכה לדרוש הוספה של רכב על מנת לסיים את שלב הרישום.
- מוכרת, כלומר קיים הרבה דוגמאות ברשת למידולים וגם frameworks רבים מתוארים בעזרתן. לדוגמה OCSF הוא חלק קוד שנרצה להשתמש וקיים לו class diagram ידוע.
- UML הוא כלי תכנון, עוזר לתכנן תוכנית עוד לפני שלב המימוש. כתוצאה ממידול המערכת מתקבל שלד ויזואלי שלה. זה יכול לעזור להפחית תקורה בשלב המימוש של המערכת. בנוסף, קל לשנות את תרשים מודל UML, בעוד שתכנות מחדש של קטע קוד יכול להיות מייגע וארוך זמן. לדוגמה כאשר בנינו את הממשק להוספת משתמש, דיאגרמת Class הקלה על ההחלטות של האובייקטים וקיצרה את זמן הבנייה מפני שכבר החלטנו על הרכיבים עוד במודל ה-Class . ניתן לראות זאת למשל ברכיב ה-"AddCustomerWindow".

3. 2. במהלך תהליך המידול עלו גם החסרונות בשיטת UML :

- רב צורתית כלומר אפשר למדל דרישה בדיאגרמה כלשהי במספר צורות. לדוגמה עלו בפגישות הצוותיות דיונים כיצד למדל תרחיש כאשר כל אחד מחזיק בדעתו בצורה שונה למידול לכן נתקלנו בבעיה: מהו המידול המתאים ביותר לסיפור.
- המידול הוא תהליך ארוך, אפילו שיחסית לכתיבת קוד המידול בעזרת דיאגרמות קצר יחסית אבל עדיין מידול איכותי של מערכת דורש זמן, לדוגמה 10השבועות הראשונים היו ברובם לשלב זה.
- במידה והתהליך מידול מסובך נפגעת מידת הקריאות והבנה שמתקבלת מהדיאגרמה. לדוגמה במידול הראשוני שלנו ב-Usecase היינו צריכים לעבור שוב ולפשט אותו מספר פעמים כדי שנוכל להשתמש בו בהמשך.

ב.3

בסעיף הקודם דיברנו על חסרונות בשימוש ב-UML אחד מהם הוא רב צורתיות.
מאד קשה לפתור בעיה זו כי אנשים חושבים בצורה שונה לדוגמה בזמן מידול ה-Use-case בזבזנו זמן רב בסיעור מוחות לגבי צורת המידול מפני שכל אחד בא מוכן עם מודל משלו.
לכן היינו מציעים להוסיף תתי דיאגרמות מובנים לחלקים מוכרים במערכת, אשר חוזרים על עצמם בפרויקטים רבים ושונים למשל, boundary של Sale או Order ,על ידי הוספת תת דיאגרמות אלו אנו מונעים רב צורתיות ומוסכמה כללית בין כל המשתתפים.