|  |  |
| --- | --- |
| **Group Number 10** | **02/05/2016** |
| Maisam Marjieh | [mar.m94@hotmail.com](mailto:mar.m94@hotmail.com) |
| Muhamad Igbaria | [muhamadig.ac@gmail.com](mailto:muhamadig.ac@gmail.com) |
| Bolous Abu-Jaber | [ajbolous@gmail.com](mailto:ajbolous@gmail.com) |
| Ahdab Serhan | [ms.ahdab@gmail.com](mailto:ms.ahdab@gmail.com) |
| Ahmad Mnasra | [ahmad.mnasra@gmail.com](mailto:ahmad.mnasra@gmail.com) |

שאלה 1:

תארו את תהליך התכנון שביצעתם ליצירת תיק רפואי ראשוני למטופל במרפאות הרשת -

בהתאם לעקרונות שנלמדו בהרצאה. פרטו מה הם ה-issues Design שהתייחסתם אליהם? מה היו החלופות ומה השיקולים לבחירת הפתרון שיושם?

כאשר לקוח מגיע בפעם ראשונה למרפאה, בעת פתיחת הרישום גם נפתח תיק רפואי ראשוני, תיק זה יהיה ריק וייפתח באופן אוטומטי כאשר המערכת מזהה שהלקוח חדש.

בעת יצירת תיק רפואי מתעוררות כמה בעיות תכנון, אחרי הוספת הלקוח ויצירת התיק אם קיים צורך להודיע לכל העמדות על העדכון, לבעיה זו אפשר להשתמש ב Design pattern של Observer ובעת הוספת מטופל לעשות notify לכל המשתמשים.

בעיה אחרת היא במבנה של המחלקה עצמה, וכול להיות שתיק רפואי של מטופל יכלול אלפי ביקורים אחרי מספר שנים, וזה יכול לפוצץ את גודל הזכרון הדרוש לכן ניתן להשתמש ב Design pattern של Proxy ולקבל את הטיפולים לפי הצורך.

שאלה 2:

א. ציינו עקרון של גישת התכן: Design Oriented-Object שבו עשיתם שימוש במהלך התכן

)design )של הפרויקט. תארו בדיוק ובפירוט את תרומתו של עקרון בתהליך התכן שביצעתם

ובתוצאתו הסופית, תוך התייחסות ספציפית )לא 'עקרונית'( לדרישות הפונקציונליות של המערכת שתכננתם.

השתמשנו בעיקרון ה Inheritance hierarchies בין המחלקות אשר יורש את המתודות והתכונות ממחלקת האב למחלקת הבן , בנוסף ניתן להוסיף עוד תכונות ומתודות למחלקת הבן כך שתהיה יותר ספיציפית , וכך גם ניתן לבצע Override למתודות שירשנו ממחלקת האב.

עקרון זה משתמש בעקרון ה- reuse שהוא מקל על השימוש במחלקות הבן ומקצר זמן המימוש בכתיבת קוד כך שנמנעים משכפול קוד מיותר.

בנוסף ניתן להגדיר מחלקות abstract שיש בהן מתודות אבסטרקטיות אשר יממומשו במחלקות הבן באופן שונה זה מזה.

בתכנון שלנו השתמשנו בעיקרון זה המחלקות הבאות:

|  |  |
| --- | --- |
| **Super Class** | **Subclass** |
| Person | User |
| Person | Patient |
| User | GlobalEmployee |
| User | ClinicEmployee |
| GlobalEmplyee | CEO |
| GlobalEmplyee | Dispatcher |
| ClinicEmployee | Doctor |
| ClinicEmployee | Manager |
| ClinicEmployee | Labortian |
| ClinicEmployee | Secretary |
| Static | WeeklyReport |
| Static | Daily |

ב. ציינו לפחות עקרון אחד של OOD שלא בא לידי ביטוי במודלים שלכם והסבירו למה.

לא השתמשנו בעיקרון ה – Interface וזאת כיוון שהעדפנו להשתמש בעקרון המחלקות האבסטרקטיות וההורשה שניתן להכניס בהם גם מימושים משותפים למחלקות היורשות.

וזה מכיוון שבממשקים –Interfaces ניתן רק להכריז על מתודות שצריך לממש אותן במחלקות הבן – העיקרון טוב רק במקרה שאין מתודות בעלות מימוש דומה במחלקת הבן.

שאלה 3 :

א. הערכה כללית:

1. מהם היתרונות של מודל UML כעזר לתהליך התכנון? ציינו דוגמה אחת קונקרטית

)לא כללית( מתוך התהליך שבצעתם לשימוש כזה ב-UML תוך תיאור והתייחסות

ספציפית למרכיבים של מערכת GHealth שתכננתם ומידלתם.

יתרונות מודל UML :

* התקשרות בין מתכנתים יותר קלה.
* זה מספק שרטוט עבור מתכנתי תחזוקה כדי לקבל סקירה של האופן בו בנוי היישום לפני בחינת הקוד בפועל. זה עשוי להפחית את זמן תחזוקה.
* מראה את התמונה הכללית של הפרוייקט.
* הגדרת דרישות התוכנית באופן מדויק וברור יותר.
* עבודה עם OOD.

לדוגמה השימוש ב – Sequence Diagram עוזר לנו לקבוע את תהליך התקשורת בין המחלקות השונות שהגדרנו ע"י class diagram ומודל 3 השכבות ,וניתן להבין את האינטראקציה בין 3 סוגי המחלקות (Boundary, Controller ,Entity) ובכך ניתן לזהות את התכונות והמתודות של כל מחלקה ודבר זה מקל את עבודתנו בשלב התכנות (כתיבת קוד).

למשל תרחיש הזדהות המטופל לקביעת תור במערכת , ע"י Class diagramתארנו את המחלקות PatientAppointmentUI , PatientsController Patient שעוזרים לנו לממש את תהליך זה , וב Sequence Diagram תארנו את סדר הפעולות והתגובות ביניהם ובכך היה ניתן לזהות את המתודות בכל מחלקה שיבצעו תהליך זה .באותו אופן ניתן לעשות עבור כל התרחישים במערכת ובכך ניתן לזהות את המבנה וארכיטקטורת ואת הלוגיקה של המערכת .באופן כזה ניתן לממש את הקוד באופן מהיר וברור יותר בהסתמך על דיאגרמות ה- UML.

2. ציינו קשיים הנובעים מחסרונות UML שנתקלתם בהם. הציעו פתרונות. גם כאן

התייחסו ספציפית לתהליך שבצעתם בפרויקט זה.

* זמן : שלב התכנון של המערכת לוקח הרבה זמן , ניתן להשתמש בעיקרון Devide and Conquer לפיצול המערכת לתת משימות ולבצע את התכנון במקביל.
* קושי בהבנת מערכת מורכבת וגדולה , כמו קודם ניתן לפעמים להשתמש בעיקרון Devide and Conquer .
* עבור שימוש ב client server ו OCSF היינו צריכים לדעת את הארכיטקטורה של ה- Database דבר זה לא ניתן להגדיר ב- UML לכן היה צריך להשתמש ב ERD.

ב. ניתוח ודיון: בהתאם לניסיון שרכשתם במהלך העבודה על מטלה זו, האם שימוש

במתודולוגית UML נותן מענה מלא לצורכי תהליך ה-design שביצעתם בפרויקט שלכם?

הסבירו את תשובתכם תוך תיאור דוגמה ספציפית )כולל שמות של רכיבים( מתוך עבודתכם.

לא, הוא לא ניתן מענה מלא לצורכי תהליך ה design , למשל בשלב מסוים היינו צריכים גם להגדיר איך חלונות ה- GUI ייראו ומה הקשרים בין החלונות השונים כדי לתאר את מחלקות ה- Boundary בתכנון שלנו , אבל בדיאגרמות ה- UML לא ניתן לייצור עיצוב וקביעת הקשרים בין החלונות השונים.

למשל בתהליך קביעת תור היינו צריכים להחליט האם לפצל את חלון קביעת התור לכמה חלונות כך שבכל חלון בוחרים שאלתה מסוימת כמו בחירת התמחות בהתחלה ולאחר מכן בחירת רופא ורק לאחר מכן בחירת תאריך ושעה לתור או לעשות זאת בחלון אחד , אשר בכל פעם מציג לנו מידע בהתאם למידע שבחרנו מקודם.