**עבודה 4 – דוח**

בעבודה זה התבקשנו לממש 3 Use case, בחרנו לממש את –

1. Login למערכת
2. הוספת שאלה לקורס
3. מחיקת שאלה מקורס
4. הוספת הערה לשאלה
5. הוספת סילבוס לקורס

הDesign Pattern שבהם בחרנו להשתמש הם -  
א. **Singelton** – תבנית אשר מגבילה את יצירת המופעים של מחלקה מסוימת למופע יחיד.  
בעבודתנו החלטנו לממש Singleton ב4 מחלקות שונות CourseCharger,QuestionCharger,TestCharger,DButils-  
מחלקות הCharger הינם מחלקות בעל מופע יחיד אשר מקושר להרבה מופעים של המחלקה שעליה היא שולטת,  
ניקח לדוגמא את UserCharger – מופע יחיד של המחלקה הזו מקושר להרבה מופעים של users (ניתן לראות זאת בתרשים המחלקות).  
לכן, אנו רוצים להגביל את מספר המופעים של מחלקות אלה למופע יחיד, ומהסיבה הזו נשתמש בSingleton.  
לגבי הDButils, אנו עובדים עם DB אחד, לכן נרצה שברגע ההתחברות לDB, נשמור על החיבור יציב ולא ננתק אותו.  
מהסיבה הזאת, נרצה ליצור רק מופע אחד של המחלקה, ואין צורך במופע נוסף.  
ב. **Facade** – תבנית אשר משתמשת באובייקט מסוים אשר מספק ממשק פשוט לאובייקט אחר בעל קוד רחב, לדוגמא ספריית קוד במקרים מסוימים.  
בעבודתנו החלטנו להשתמש בFacade במחלקה Model(יורחב עליה בהמשך) מכיוון שהמודל מהווה מין דלת הכניסה לעולם הקוד האמיתי שבתוכנה.  
הModel בעל 3 מופעים של UserCharger,CourseCharger וQuestionCharger והוא מפנה את הבקשות שהגיעו אליו לCharger המתאים.  
כפי שהסברנו קודם, מחלקת הModel מהווה ממשק פשוט לאובייקטים מסוך Charger ומסתירה את הקוד הטמון בהם, זאת כדי לממש ממשק פשוט להבנה.  
  
הארכיטקטורה שבעזרתה כתבנו את התוכנית היא **MVC**. הקוד מחולק ל3 חלקים, View, Controller וModel. למשתמש יש אינטראקציה רק עם מחלקת הView, מחלקת הView מעבירה את הInput מהמשתמש אל מחלקת הController שמעביר את המידע למחלקת הModel.  
מחלקת הModel(המממשת את הDesign Pattern מסוג Facade), מעבירה את המידע אל מחלקת את הCharger הרלוונטית ומחכה שתחזור תשובה, ברגע שחוזרת התשובה היא מחזירה את המידע חזרה לController שמעביר זאת לView המציג זאת למשתמש.  
  
הסיבה שבחרנו לא לממש את **Observer** Design Pattern היא מהסיבה שמימוש Observer בארכיטקטורת MVC פירושו Data Binding ישירות מהModel לView, למעשה, לדלג על הController במידע החוזר מהCharger המתאים.  
בעבודתנו, הController משמש תפקיד עיקרי במערכת, הוא המשלב בין הView לModel.   
לכן, לא נרצה לדלג עליו בData Binding מהModel לView.  
מהסיבה הזאת החלטנו לא לממש את Observer Design Pattern.