קבוצה 6

יהונתן גבאי – jgabbayj@gmail.com

אלון דויטש – alon9510@gmail.com

ענבל קטראריו – inbalos22@gmail.com

בת אל קוק - batel571992@gmail.com

תאריך הגשה: 21.5.2018

**שאלה 1:**

כשסטודנט רוצה להתחיל בחינה, הוא מכניס קוד בחינה אשר קיבל מהמורה.

המערכת שולחת את הקוד לserver , אשר בודק מול הDB- האם קיים קוד כזה שנמצע בסטטוס ביצוע.

במידה וקיים, מחזיר לclient- שהקוד תקין, ולאחר מכן המערכת מציגה לסטודנט 2 אפשרויות: בחינה ממוחשבת או בחינה ידנית. הסטודנט בוחר בבחינה ממוחשבת - המערכת שולחת לו טופס בחינה עם שאלות ותשובות מוסתרות וכן, שדה למילוי ת"ז. לאחר הכנסת ת"ז, הסטודנט לוחץ על כפתור התחל בחינה, ה-server בודק שת"ז נכונה - במידה וכן, הסטודנט מתווסף לטבלת הנבחנים הפעילים עם ת"ז, כתובת הip-, קוד הבחינה, זמן תחילה וזמן סיום. לאחר מכן, תוכן הבחינה המוסתר מתגלה.

אכיפת זמן הבחינה תתבצע ע"י חוט שרץ ב-server , שתפקידו לבדוק האם זמן הבחינה של אחד הנבחנים הפעילים הגיע לקיצו. במידה וכן, תישלח הודעה ל-client של אותו נבחן(ניתן להגיע אליו - כיוון ששמרנו נתונים כגון ip וכיוצא באלה בDB) אשר מודיעה על סיום מידי של הבחינה. בעת קבלת הודעה, על סיום זמן הבחינה,

מה-server, הבחינה שמתבצעת בclient תישלח לserver ומסך הבחינה ייסגר ויתחלף בהודעה על סיום זמן הבחינה. הserver ישמור את הבחינה במסד הנתונים בנוסף לאופן סיום הבחינה ושאר הנתונים הרלוונטיים. בעת סיום אחרון הסטודנטים את הבחינה, הבחינה תצא מרשימת הבחינות בביצוע ולא יתאפשר לקחת אותה שוב.

אחת הדילמות שנתקלנו בהן היא איך לסיים את הבחינה לכל הסטודנטים בעת ביצוע נעילה ע"י המורה. הפתרון שיישמנו לבעיה זו הינו כזה: בעת קבלת הוראת נעילת בחינה אשר אושרה ע"י המנהל, ניצור חוט בserver- אשר יעבור על רשימת הנבחנים הפעילים ולפי קוד הבחינה של הנעילה, ישנה את זמן הסיום לזמן העכשווי. כעת החוט האחראי על סיום זמן בחינה יעבור על הרשימה וישלח הודעת סיום בחינה לכל הנבחנים המתאימים לקוד הבחינה המבוקש כפי שתיארנו מקודם.

**שאלה 2:**

א) אחת התכונות המאפשרות לנו שימוש בקוד חוזר הינה תכונת האבסטרקטיות. לדוגמה, בפרוייקט זה השתמשנו ב Object client server framework- (OCSF) אשר מאפשר לנו עבודה במסגרת client-server תוך שינוי מינימלי באופן מימוש הקוד, ללא התמצאות עמוקה באופן מימוש התקשורת.

ב) אחד העקרונות אשר איננו בא לידי ביטוי הואdesign patterns - לא נתקלנו בבעיה שכיחה אשר ניתן להתאים לה פתרון חוזר.

**שאלה 3:**

הערכה כללית :

כל המידע הניתן מהמתשתמש והדרישות שלו ממערכת AES מפורטים כסיפור.

בניית תרשים UML ימקד אותנו בתחום הידע של המשתמש.

לכל מחלקה בדיאגרמת קלאס, יש חלוקה ל-3 שכבות (שם מחלקה, מאפיינים,מתודות) - ידידותי למשתמש. כל טבלה תתאר מבנה נתונים ואת מערכת היחסים בינהם.

כל התהליכים המתוארים בתרשים UML אינם מכילים נתונים טכניים או איך אנחנו נממש אותם, אלא דברים קריאים, אשר רמתם מותאמת לדרך החשיבה של המשתמש.

לדוגמה, אופן ביצוע תהליך הLogin מתואר בפשטות רבה(משתמש מזין שם משתמש וסיסמא, המערכת בודקת את הנתונים ומחזירה הודעה מתאימה), שזהו תיאור שללקוח קל מאוד להבין.

בנוסף, תהליכים נוספים ומורכבים יותר, כמו הוספת שאלה (המשתמש בוחר להוסיף שאלה, מזין נתונים בהתאם, מאשר נשמש במערכת והשאלה נוספת למאגר) - תהליך המתואר באופן פשוט אך מורכב ברמת המימוש (במימוש יהיה צורך לבדוק מקרים של תקינות שאלות, מילוי שדות, ביצוע השמירה וכו') - בעזרת UML, ניתן לפשט אותם בשביל המשתמש.

מערכת AES מכילה תהליכים מורכבים, המקיימים תרחישים שונים. למשל, לקיחת בחינה, הוצאת בחינה לפועל, תהליכים שכוללים הרבה פעולות ותנאים שונים, אשר מעמיסים על הדיאגרמות (בתהליכים אלו עומס בACTIVITY DIAGRAM). מימוש של דיאגרמת הCLASS מכיל מחלקות וקשרים רבים הנמצאים במערכת AES.

כתוצאה מכך, הן עלולות להיות לא קריאות וקשות מאוד להבנה למי שאינו חלק מתהליך המידול עצמו.

חסרון נוסף של UML במערכת היא בעיית ריבוי המודלים.

נאצלנו לממש מודלים שונים במערכת AES (activity ,class ,use case).

כל מודל נבנה בזמן אחר, יש צורך לבדוק ולשמר קשר בין הדיאגרמות השונות כל הזמן, על מנת למנוע חוסר תאימות בין הדיאגרמות השונות.

ניתוח ודיון :

דיאגרמת Use Case מהווה מימוש של זרימת האירועים בכל case במערכת.

דיאגרמת Class מכילה את תרשימי המחלקות שלנו במערכת, המורכבת מבעלי התפקידים (שחקנים) והפעולות שיבצועו.

כל התרשימים קשורים זה בזה. מכל זרימה של אירוע ניתן לראות מי השחקנים ומה הם עושים. דיאגרמה שתכיל את כל המידע המשותף, תכיל אפשרות לעריכה ושינוי בהתאם לכל אחת מהדיאגרמות ותביא לרצף עבודה ושמירה על אחידות ותיאום ותאפשר למשתמש לראות את "התמונה הגדולה".