מגישים : גל בוזגלו – 203886528, מיטל יניב -307938969
שי ארץ קדושה – 203276258, יסמין אברהם- 208063453, רותם מיארה- 301886776

קורס הכנה לפרויקט גמר בהנדסת מערכות מידע

דו"ח פיתוח גרסאות 3+4

תפקידי חברי הצוות בפיתוח גרסאות 3+4:

מנהל הגרסה: שי ארץ קדושה

מפתחים: מיטל יניב, יסמין אברהם , רותם מיארה

אחראי טסטים: גל בוזגלו

פירוט הספק לגרסאות 3+4:

משימות שבוצעו במלואן בגרסה הנוכחית:

- תיקון ועדכון של כלל התוצרים מגרסה 1+2 שיהיו תואמים למערכת שהמשכנו לפתח בגרסה 3+4: תרשים ארכיטקטורה, תרשים מחלקות, עדכון מילון מונחים ועדכון USC.
- המשך פיתוח האפליקציה שהתחלנו לפתח בגרסה 2- הוספת פונקציונאליות למחלקות הDomain .
- פיתוח בסיס נתונים לשמירה על הנתונים לאורך זמן ולשימוש העת הצורך
- פיתוח ממשק משתמש

את גרסה זו פיתחנו עפ"י ארכיטקטורת שרת-לקוח. צד השרת ינהל את כל הלוגיקה ובנוסף אחראי על יצירת קשר ושמירה בבסיס הנתונים. צד הלקוח אחראי על הצגת המסכים ללקוח, ויצירת קשר עם השרת בכל פעולה שמשתמש רוצה לבצע.

פיתוח:

Data Base:

יצרנו בסיס נתונים על השרת באמצעות MongoDB. שימוש ב- mongodb מאפשר לנו לנהל את המידע השמור בצורה נוחה יותר, ולשלוף נתונים במהירות.

צד שרת:

שכבת ה- service:

- בשכבת הservice הוספנו 2 מחלקות: Server- תפקיד המחלקה זה לפתוח חיבור עם הclients ולשלוח להם את הנתונים בהתאם לנעשה בDomain. ClientHandler- תפקיד מחלקה זה לנהל את כל הלקוחות הרוצים לתקשר עם השרת.

שכבת ה- Domain:

- בשכבה זו נכון לגרסה זו נכתבו מחלקות המאפשרות להקים קבוצה והנכסים עבורה (שחקנים, מאמנים, וכו'). בנוסף נכתבו מחלקות המאפשרות להתאחדות לבצע בקרה על הקבוצות וכן לנהל ולקבוע משחקים בליגות ובעונות וכן מערכת תקציבית בה ארגון יכול לנהל ולרשום את הטרנזקציות שהוא מבצע.
המחלקות מחולקות ל-Packages שונים על מנת ליצור תלותיות נמוכה ביניהן.
בשכבה זו הוספנו את מחלקת Model- שתפקידה לנהל את הקשר בין מחלקות הDomain לשרת.
  - **יומן אירועים SystemEventsLog**- יצרנו מחלקה השומרת בקובץ חיצוני בשרת את כל האירועים שקרו במערכת. יומן אירועים זה שומר את כל האירועים עבור כל משתמשי המערכת.
  - **יומן שגיאות SystemerrorLogs**- מחלקה השומרת בקובץ חיצוני בתוך השרת את כל אירועי המערכת, לדוגמה: כשל בעת נסיון להתחברות לבסיס הנתונים, נפילה של המערכת וכדומה.
  - **Alerts:** שליחת התראות למשתמשים בזמן אמת ומתן אפשרות להירשם להתראות על משחקים המתקיימים במסגרת ההתאחדות.

**** נכון להגשה זו, לא ממשנו פונקציונאליות זו במלואה והיא תושלם ותוצג בפגישה על איטרציה זו.**

מגישים : גל בוזגלו – 203886528, מיטל יניב -307938969
שי ארץ קדושה – 203276258, יסמין אברהם- 208063453, רותם מיארה- 301886776

שכבת הDB:
שכבה זו מאפשר לשרת ליצור חיבור עם בסיס הנתונים אותו פיתחנו על מנת לייבא מידע נחוץ ולשמור מידע לאורך זמן. שכבה זו מכילה ממשק לDB, ומחלקות המאפשרות את ההתממשקות עם בסיס הנתונים.
ישנן שתי מחלקות של בסיסי נתונים שממשות את הממשק: אחת מדמה בסיס נתונים בזיכרון והשנייה מתקשרת עם בסיס נתונים MongoDB..

External Systems:
על מנת ליצור חיבור למערכות חיצוניות תוך שימור על אבטחה והסתרת המימוש מהמשתמש השתמשנו בתבנית העיצוב Proxy. עבור כל מערכת חיצונית יצרנו מחלקה שתשתמש במערכות החיצוניות האמיתיות, והיא תהיה עבור המשתמש.

<u>צד לקוח:</u>
שכבת ה- Presentation:
שכבה זו אחראית על הפעלת הפונקציונאליות בתוך ממשק המשתמש שלא דורשות את החיבור לשרת- החלפת מסכים, הפעלת כפתורים וכדומה. בנוסף שכבה זו "מודיעה" לpresenter על השינויים שחלים בה.

שכבת ה- service:
בשכבה זו אחראית ליצירת התקשורת בין הלקוח לשרת באמצעות מחלקת Client , הפעלת ממשק המשתמש וחיבורו למחלקה Client באמצעות presenter.

טסטים ל-UI:
ביצענו טסטים למנשק המשתמש באמצעות תוכנת Squish שבעצם יוצרת תסריטי בדיקה אוטומטיים לקובץ jar וניתן כל פעם להריץ את התסריטי בדיקה שנכתבים אוטמטית ב-Pyhton בעת הקלטת תסריט הבדיקה.

<u>תיעוד ניהול גרסה 3+4 (על סמך העקרונות במסמך המתודולוגיה):</u>

<u>ניהול הפרויקט (Project Management)-</u> בכל פגישת צוות שביצענו כלל המשימות בתהליך הפיתוח של גרסה זו חולקו בין חברי הצוות ומונה אחראי על כל חלק במערכת בהתאם לחלוקת התפקידים שמצוינת לעיל, כלל המשימות תועדו במערכת Trello. עבור כל משימה במערכת נכתב תוכן המשימה, דד ליין עבור המשימה ומי אחראי על ביצועה. כלל חברי הצוות נעזרו בתיעוד המשימות שנכתבו ב-Trello וכך בוצע בקרה על ביצוע כל משימה ומשימה. חבר צוות שסיים את משימתו עדכן זאת ב-Trello ואז בפגישה הבאה היינו עוברים על סטטוס המשימות שהיו צריכים להתבצע ומחלקים משימות נוספות.

<u>עיצוב, ניהול ותחזוקה של המודלים (Design, Management and Maintenance of models)</u>- כתיבת המחלקות השונות בגרסה זו התבססו על כלל התרשימים והמודלים שנכתבו בגרסה 1-2. במהלך תהליך הפיתוח בגרסה זו בוצעו מספר שינויים שמצאנו לנכון שיש לעשותם על מנת שיהפכו את המערכת ליעילה יותר. בין היתר הוספו מחלקות שונות ובוצעה חלוקה ל-Packages על מנת ליצור סדר, ארגון והיררכיה בין המחלקות בקוד. את כלל השינויים הדינמיים שנעשו עדכנו בכלל התרשימים שעליהם מתבססת המערכת ביניהם Class Diagram ו- Architecture Diagram ובנוסף בוצעו עדכונים במילון מונחים שחשבנו לנכון שיש להוסיף.

<u>ניהול קוד (Source code Management)</u>- את גרסה זו כתבנו בשפת Java בעזרת שימוש בתוכנה IntelliJ. את בסיס הנתונים מימשנו באמצעות Mongo DB . את המבנה הבסיסי של המערכת (מחלקות, ממשקים ו-Packages) כתבנו יחד ולאחר מכן נעשתה חלוקה שכל חבר צוות קיבל אחריות לפתח ולממש מספר מחלקות.

נעזרנו בפלטפורמה לשיתוף קוד- Git כאשר מהענף של ה-master יצרנו 3 branches וכל חבר צוות עבד ב-branch שלו על המשימות שהוגדרו לו (שלוש חברות צוות עבדו על אותו ענף ושניים נוספים על ענפים אחרים). כשחבר צוות החליט שסיים משימה מסוימת בענף שלו והוא מעוניין לשלב אותה במערכת הוא ביצע Push לענף שלו עם commit רלוונטי ועדכן את מנהל הגרסה על כך. מנהל הגרסה עשה Pull ל-branch של אותו חבר צוות, בחן את הקוד, ווידא שאין בו באגים ע"י קימפול והרצת דו"ח Pit ורק לאחר מכן מיזג את הקוד לענף ה-master ועשה לו push.

מגישים : גל בוזגלו – 203886528, מיטל יניב -307938969
שי ארץ קדושה – 203276258, יסמין אברהם- 208063453, רותם מיארה- 301886776

ניהול גרסאות (Version Control Management)- השתמשו במערכת CI שנקראת Jenkins (זו שנלמדה בסדנה) בכדי לבצע פיקוח ובקרה על כל עדכון שבוצע. כלומר ברגע שמנהל הגרסה ביצע Push עבור העדכון לענף ה- master ב-Git , אוטומטית בוצע build במערכת ה-Jenkins. במהלך ביצוע ה-build לאותו עדכון ב-Jenkins אוטומטית הורצו כלל הטסטים שנכתבו עבור המערכת על מנת לבדוק ולוודא שהשינויים שנעשו לא פגעו במערכת במקומות אחרים. אם ה-build הסתיים בהצלחה מוצג דו"ח של כלל הטסטים שהורצו והסטטוס שלהם (ניתן לייצא את הדוח לאקסל). בנוסף התוצר של הגרסה המעודנת אחרי השינוי נשמר אוטומטית במערכת נוספת -Nexus (נלמדה גם בסדנה) כקובץ jar כך שאפשר תמיד להשתמש בתוצר העדכני של הגרסה לאחר הבדיקות (נציין כי מימשנו את הממשק הנ"ל ואכן נשמר קובץ jar ב-repository ב-Nexus אבל השימוש בו יבוא יותר לידי ביטוי בגרסאות הבאות כשיבוצעו בדיקות מערכת ע"י משתמשים). לבסוף נשלח מייל למנהל הגרסה בו נאמר האם העדכון בוצע בהצלחה או נכשל. במידה והעדכון נכשל נבדקת מה הסיבה לכך באמצעות המעקב בין הגרסאות השונות שהועלו ל-Git ובוצע עליהן Build (קורה פעמים בודדות בעקבות התהליך המבוקר הנ"ל) ובמידה במייל נכתב כי העדכון בוצע בהצלחה מנהל הגרסה מודיע לכלל חברי הצוות שיש גרסה מעודכנת ועליהם לבצע pull מהענף של ה-master לענף שלהם.

עקיבות בין כלל תוצרי הגרסה (Traceability among the software artifacts)- כל הגרסה הנוכחית הייתה מבוססת על מסמך הדרישות שהוצג לנו קצת לפני תחילתה של גרסה 1. המעקב אחרי מימוש של הדרישות בגרסה הנוכחית מוצג בטבלת עקיבות (traceability) בה מפורטות כלל הדרישות עבור כל משתמש במערכת ומבחני הקבלה שבודקים את המימוש והביצוע של אותה דרישה. מבחני קבלה הם כמובן בהתאם ל-USC שנכתבו כמו שפורט קודם בחלק של הטסטים. בנוסף כלל התרשימים והמודלים עודכנו בהתאם לשינויים שבוצעו במהלך הפיתוח של גרסה זו.