המרכז האקדמי רופין
Ruppin Academic Center
המחלקה להנדסת חשמל ומחשבים
Department of Electrical and Computer Engineering

המרכז האקדמי רופין

הפקולטה להנדסה

המחלקה להנדסת חשמל ומחשבים

קורס הנדסת תוכנה אינטגרטיבית

ספר פרויקט – 'פוקר אונליין'

'Poker Online' Project Book

מוגש לידי :

מנחה הקורס מר תמיר דרשר וצוות השופטים.

מוגש על ידי :

שם: עמית זוהר מס' זהות: 313307720

שם: גולן פרשה מס' זהות: 305001885

שם: אופק בן עטר מס' זהות: 322208430

שם: אופיר נחשוני מס' זהות: 204616718

תוכן עניינים

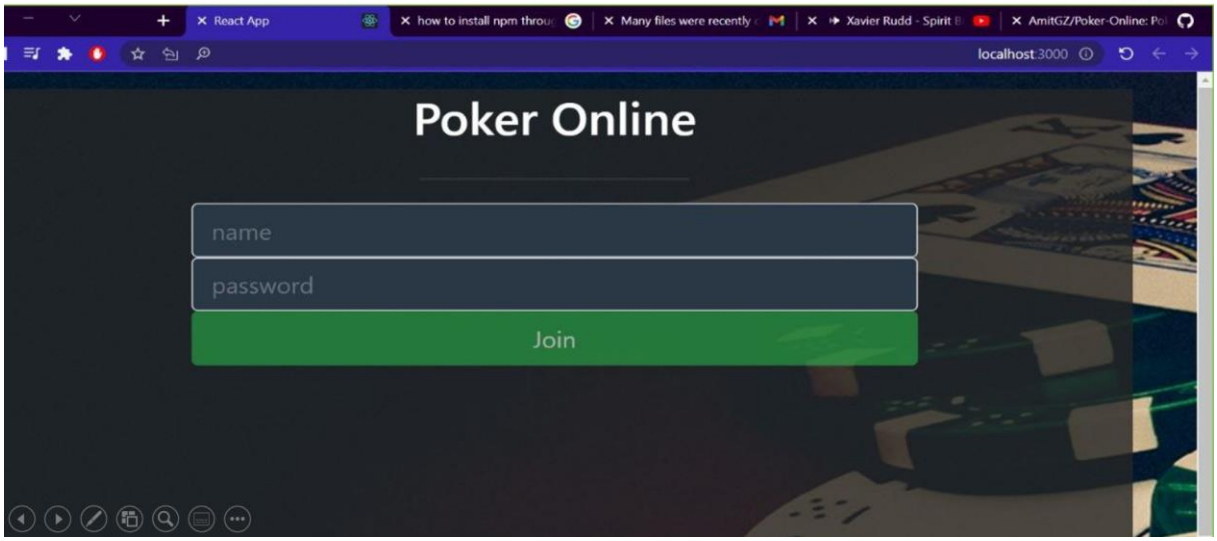
1. תקציר מנהלים

'פוקר אונליין' הינו משחק רשת רב-משתתפים, מבוסס web. המשחק שומר ב-DataBase את נתוני המשתמשים (כמו שם, כסף וכו'). השרת, אשר עובד ע"י Signal-R, מאפשר לכמה משחקים ('חדרים') להתקיים במקביל, וכן מאפשר למשתמש לגשת ממגוון מכשירים במקביל. הקוד כתוב בשפת c-sharp. תהליך העבודה בצורה בשיטת DevOps בצורה סדורה ושיטתית.

2. תצלומי מסך

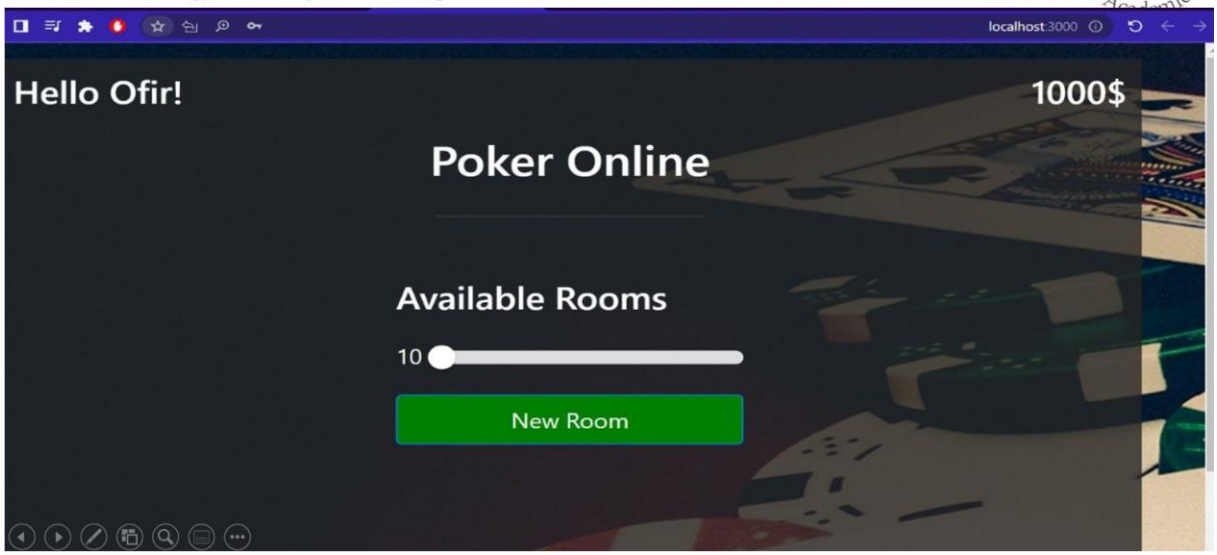
Game Play Screenshots:

Sign-In screen:



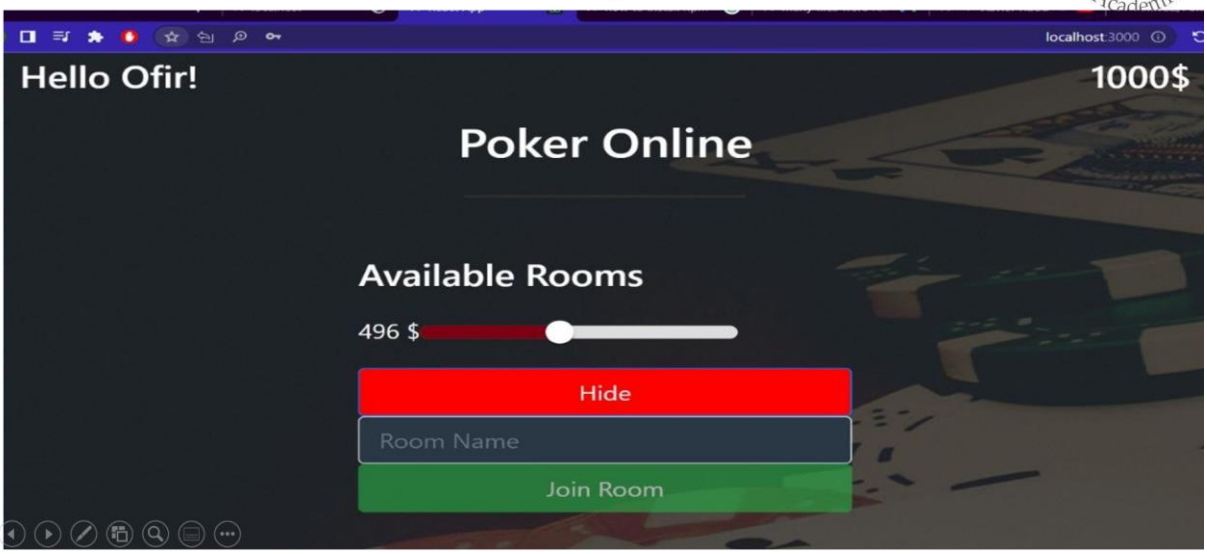
Game Play Screenshots:

Lobby, after Ophir has signed in:



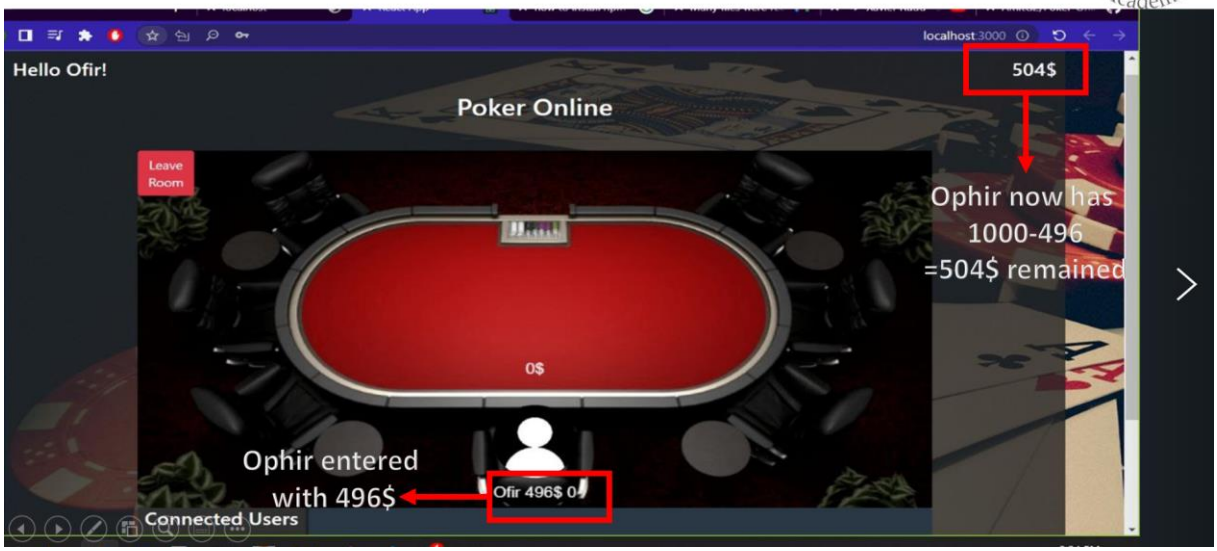
Game Play Screenshots:

Providing the starting amount of money:



Game Play Screenshots:

The created room, with just Ophir inside:



Game Play Screenshots:

Amit's screen after he joined the room. Notice that he's centered, and can only see his cards:

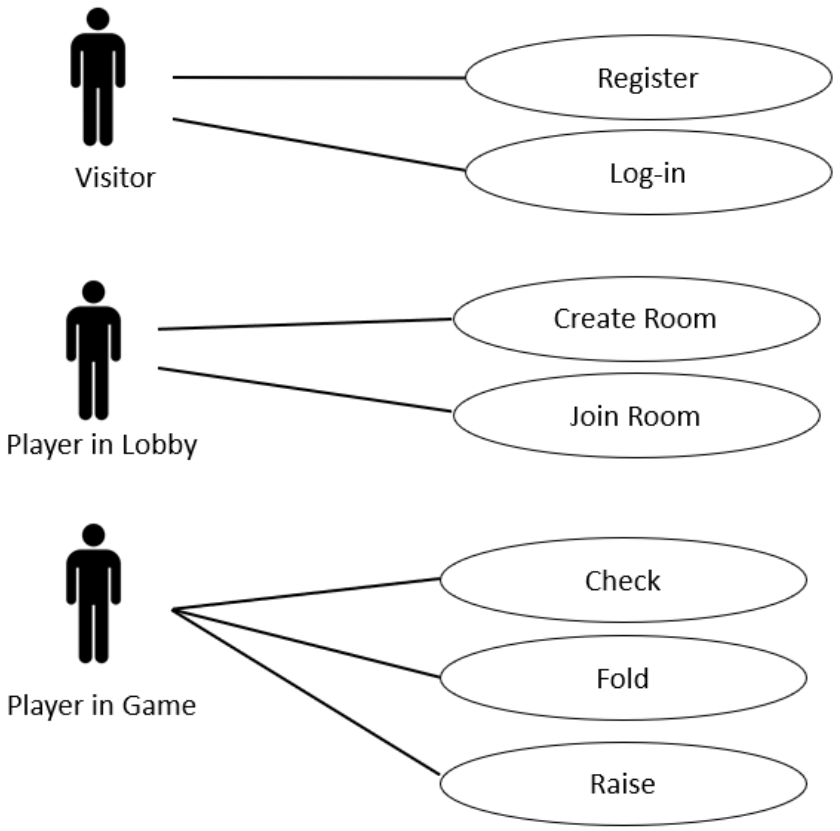
Game Play Screenshots:

Golan's screen when he joins. Notice again that he's centered, and can see only his cards :

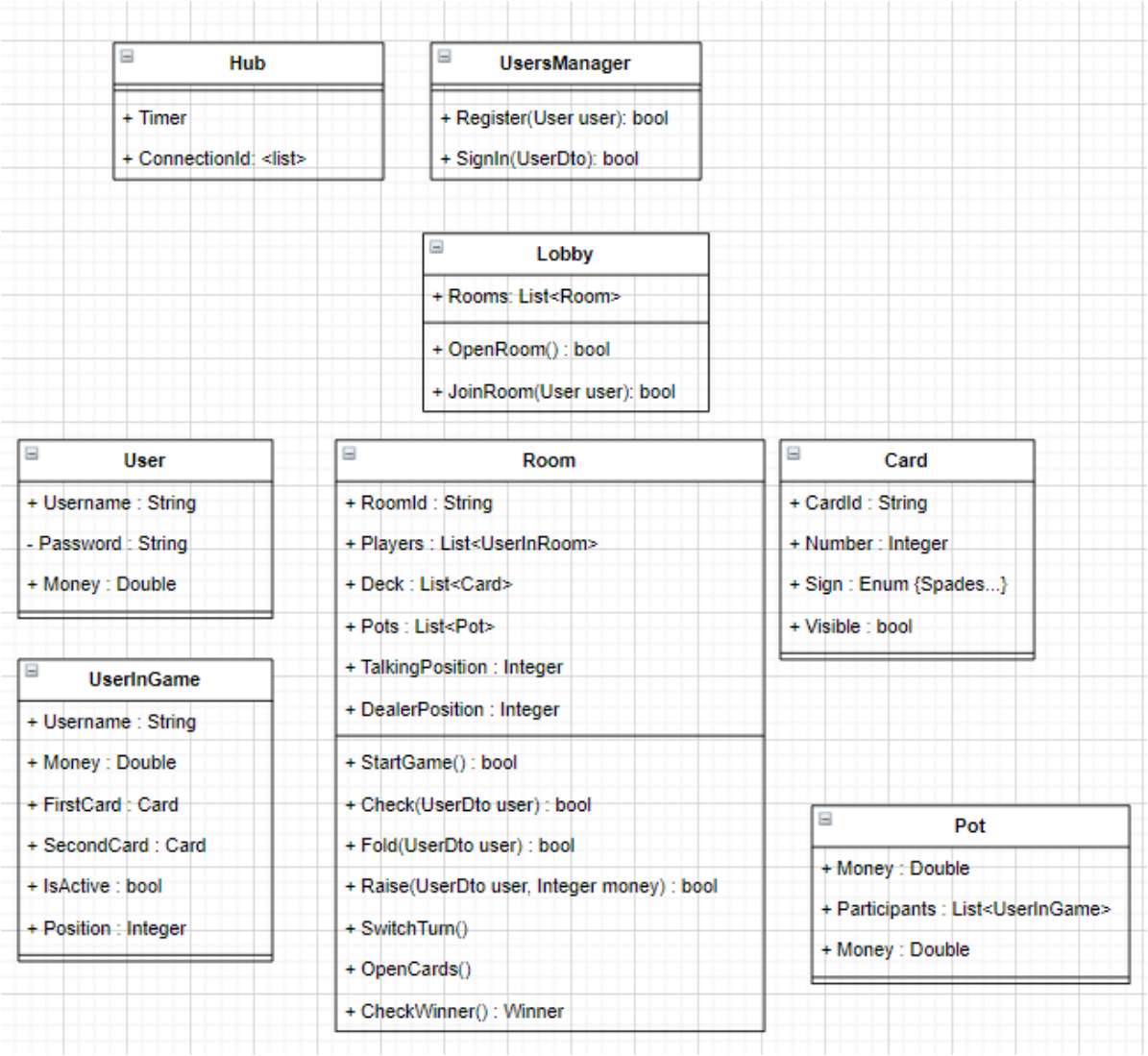
3. טבלת שחקנים

שם שחקן	פירוט	הערות
Visitor	כל משתמש אשר נכנס לדף הפתיחה, וטרם הזדהה מול המערכת. באפשרותו להירשם (Register) או להזדהות ולהתחבר למערכת (Log-In)	
User	לאחר ביצוע הזדהות מול המערכת, המשתמש מועבר ללובי, שם הוא יכול לפתוח חדר (משחק), או להצטרף לחדר קיים.	
Player	בעת כניסה לחדר, הופך לשחקן פעיל, ויכול להשתתף במשחק.	

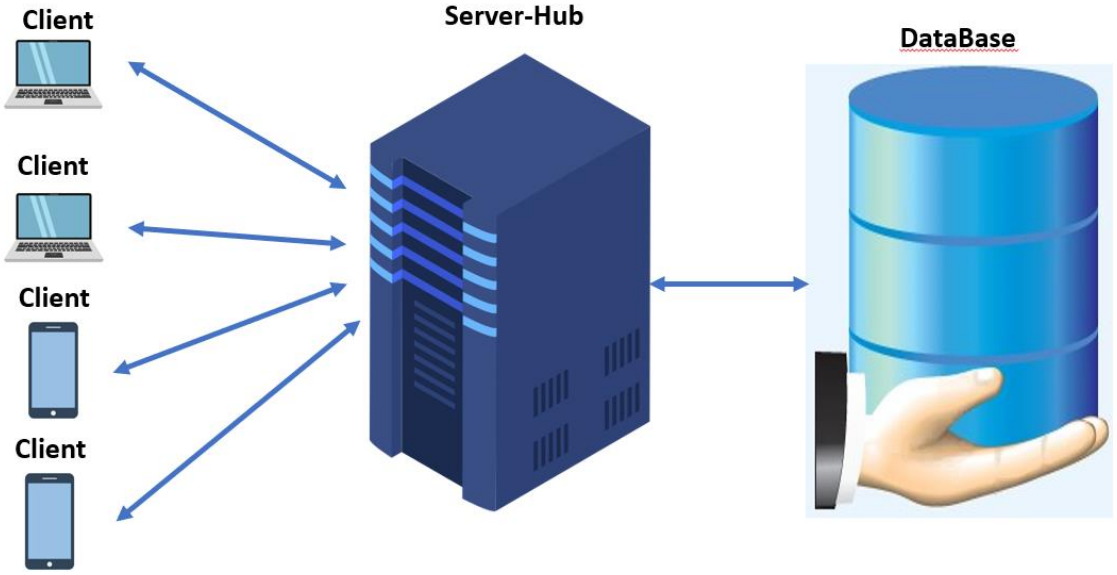
4. תרשים UseCases



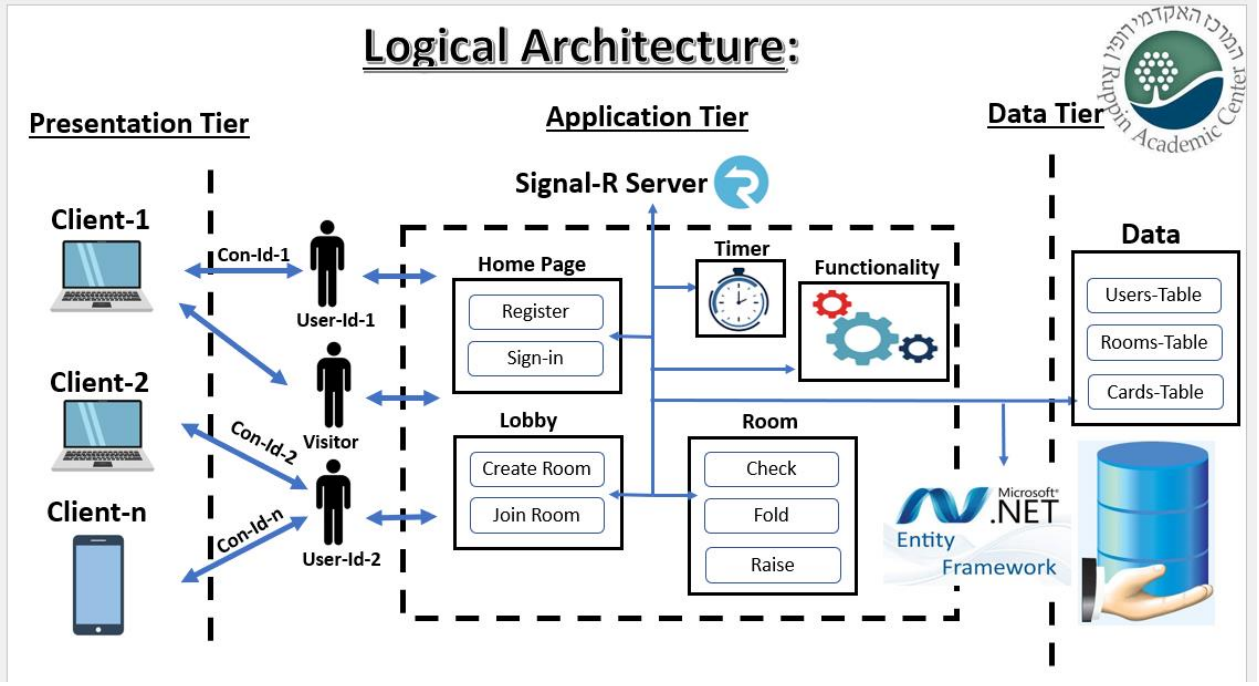
5. תרשים מחלקות



6. ארכיטקטורה פיזית



7. ארכיטקטורה לוגית



כל משתמש יכול להתחבר לאפליקציה בעזרת דפדפן, גם מספר פעמים (כלומר ע"י TABים שונים), כל חיבור כזה מקבל User-ID ייחודי. לאחר ההתחברות השרת מחבר בין הCon-Id ל-User ID וכך משתמש יכול לשחק ממספר מכשירים (וגם מספר משתמשים ממכשיר אחד). כל הפקודות הניתנות ע"י השחקנים הולכות לשרת Signal-R, והוא זה המפיץ גם את השינויים לשאר השחקנים. כלומר כל הפונקציונליות מתבצעת ב Signal-R. בנוסף, קיים timer אשר עושה fold למשתמש אם לא הגיב מעל 60 שניות. כל המידע הרלוונטי נשמר בDataBase, והקריאות נעטפות בעזרת Entity Framework. בצורה זו אנו משיגים שני דברים: האחד, ווידוא את תקינות הקריאות והמידע, והשני היא שמירת מידע רלוונטי ב-DB כך שאם השרת נופל יש Disaster Recovery ואפילו ארכיון.

8. תהליכים מרכזיים

1. ניהול משתמשים – יצירת משתמש, הזדהות וכניסה.
2. ניהול חדרי משחק – פתיחה, הצגה והצטרפות לחדר קיים.
3. ניהול המשחק – שימוש בתורות, טיימר, חלוקת קלפים, אלגוריתם קביעת ניצחון.
4. Disaster Recover – שמירת כלל הנתונים והמצבים הרלוונטיים ב-DB, כך שניתן להמשיך ככל הניתן לאחר תקלות.

9. בדיקות

עבור כל מחלקה ב-BackEnd (DataModel class) ישנו קובץ נפרד (.cs) שמכיל בדיקות יחידה (Unit Tests). סוג הבדיקות שהשתמשנו בהן מסוג XUnit. כל המחלקות שנמצאות תחת תיקייה DataModel ומייצגות איזושהי ישות בתוכנה, מכילות פונקציות שמבצעות פעולות לוגיות, ללא ממשק עם External Dependencies (כמו ממשק עם DataBase). עבור כל מחלקה (למשל: Room, UserInGame) אשר קיימות פונקציות שמכילות קוד לוגי בלבד, ללא תלות באובייקטים חיצוניים כמו DataBase למשל, כתבנו בדיקות שבודקות את תפקוד הפונקציות הלוגיות שנמצאות בה. הרציונל בכתיבת הבדיקות היא שבכל אחת כזו, נבדקת (באופן בלתי תלוי בבדיקות האחרות) פעולה אחת שאחראית פונקציה מסויימת. עבור אותה פונקציה יהיו לרוב יותר מבדיקה אחת. למשל מחלקת Room מכילה 11 פונקציות, שלכל אחת כתבנו מספר בדיקות שבודקות פונקציונליות שונה של הפונקציה.

להלן טבלה המסכמת את הבדיקות שנכתבו ולאיזו מחלקה/פונקציה הן שייכות:

S.Number	Number of tests	Test class	DataModel class	Main Function
1	1	UnitTestsCardHandler.cs	CardHandler.cs	GenerateShuffleDeck()
2	5	UnitTestsUserInGame.cs	UserInGame.cs	ResetUser()
3	2	UnitTestsRoom.cs	Room.cs	AddUser(User,Money)
4	3			RemoveUser(User)
5	1			EndGame()
6	1			StartGame()
7	1			Raise(UserInGame,amount)
8	1			Check()
9	3			ResetGame()
Total Unit Tests: 18				

- המחלקות שלא נכתבו אליהן בדיקות הן: Card, Dbinitializer, PokerContext, Pot, User. הסיבות לכך יכולות להיות אחת מאפשרויות רבות: מחלקה שאינה מכילה פונקציות בכלל (מאפיינים/שדות ובנאי/ם), מחלקה שמכילה רק פונקציות בעלות ממשק חיצוני ל-DataBase ועוד.

10. תהליך עבודה

תהליך העבודה בוצע בשיטת DevOps, כאשר הפלטפורמות העיקריות בהן השתמשנו הן AzureDevOps, ו-GitHub. כאשר כל ספרינט נמשך שבועיים, בו חולקו משימות לחברי הצוות. כל חבר צוות ביצע את משימותיו, והעלה את התוצרים הנדרשים ל-Azure. בכל יום שלישי בו התקיים מפגש הקורס, ביצענו תיאום, סנכרון, וקיבלנו החלטות ומשימות להמשך.

11. סיכום

בקורס זה ביצענו לראשונה פרויקט מורכב, מלא, בסדר הגודל הזה. למדנו רבות הן מבחינה מקצועית, לדוג' שימוש ב-Signal-R, עבודה מול DataBase וכו', והן מבחינת ניהול ופיתוח הפרויקט בצורה נכונה מבחינת ארכיטקטורה, והן בשיטת הפיתוח, ע"י שימוש בפלטפורמות כמו GitHub ו-AzureDevOps.