



משרד החינוך

פרויקט גמר - מדעי המחשב סייבר

'Chessy'

משחק שחמט רב משתתפים +

יריב ממוחשב



שם התלמיד: עילי בן נון

ת.ז התלמיד: 214796443

שם בית הספר ועיר: קריית חינוך דרור, בני-דרור

המורים המלווים: אורית קלינגר

מועד הגשת המסמך: 23.06.2022

תוכן עניינים

3	פרק א' - מסמך ייזום
5	פרק ב' - אפיון הפרויקט
7	פרק ג' - מסמך ניתוח
9	פרק ד' - מבנה ועיצוב
17	פרק ה' - קטעי קוד נבחרים
25	פרק ו' - מדריך למשתמש
30	פרק ז' - מבט אישי
31	פרק ח' - ביבליוגרפיה

פרק א' - מסמך ייזום

תיאור ראשוני :

מטרת הפרויקט הינו יצירה של פלטפורמה לניהול משחקי שחמט בזמן אמת בין שחקנים במחשבים מרוחקים, בין שחקן ליריב ממוחשב, ובין שחקן לעצמו (או שחקן אחר, באופן לוקאלי). בנוסף, הפלטפורמה תחזיק במסד נתונים הכולל פרופילים של כל המשתמשים.

המערכת תאפשר לשחקן להירשם אליה, להתחבר, להתחיל משחק בזמן אמת מול שחקן מרוחק, או להתחיל משחק מול יריב ממוחשב המשתמש באלגוריתם אשר יאפשר לו לשחק בצורה הטובה ביותר.

המוטיבציה לפיתוח הפרויקט נובעת מתוך עניין אישי בתחום הבינה המלאכותית, ובמשחק השחמט.

הגדרת הלקוח :

הלקוחות אליהם המערכת מיועדת הם שחקני וחובבי שחמט, אשר מעוניינים לשחק מול חברים (באופן לוקאלי), מול זרים דרך הרשת, או מול אלגוריתמים מאומנים.

הגדרת יעדים ומטרות :

המטרה הראשונה והבסיסית ביותר של הפרויקט היא יצירת הלוגיקה עבור משחק השחמט. לשם כך נעזר בספריית pygame לייצוג הוויזואלי של המשחק. בכדי לאחסן את מצב המשחק בתוך הקוד נשתמש במערך דו-מימדי (שורות וטורים), ובפעולות מתמטיות עבור ביצוע מהלכים.

המטרה השנייה של הפרויקט היא יצירת היריב הממוחשב, אשר צורך שימוש באלגוריתם שיסתכל מספר מהלכים קדימה, ויחשב את המהלך הטוב ביותר לביצוע כעת. למשתמש תהיה אופציה להתחיל משחק נגד היריב הממוחשב.

המטרה השלישית היא הפיכת המשחק למרובה שחקנים, על ידי הקמת סרבר אשר יכול לחבר בין שחקנים מרוחקים, ולאחסן את המשחקים המתנהלים בין השחקנים בתוך thread. כאופציה נוספת, יוכל המשתמש להתחיל משחק נגד שחקן ממוחשב אחר.

המטרה הרביעית היא הקמת ושמירת הפרופילים של המשתמשים בפלטפורמה, אשר יכללו את השם והסיסמה של המשתמש. כמו כן, יהיה צורך בפיתוח חלון התחברות והרשמה לפלטפורמה.

בעיות, תועלות וחסכונות :

הבעיה אותה אנו מנסים לפתור היא הצורך בקרבה פיזית בכדי לשחק במשחק השחמט מול אדם אחר.

המערכת שאנו מפתחים תספק פתרון לבעיה זו, בכך שתאפשר לנהל משחקי שחמט מרוחקים דרך הרשת.

לבעיה זו קיימים כבר מספר פתרונות ברשת. הפופולרי מכולם הוא האתר chess.com, אשר מאפשר ניהול משחקי שחמט דרך הרשת מול זרים באותה רמת ידע ונסיון כמוך (בעזרת מערכת דירוג ELO), ומול חברים. כמו גם ניהול משחקים מול יריבים ממוחשבים בעזרת אלגוריתמים מתקדמים לבחירת מהלכים במשחק שחמט (stockfish וכדומה).

קשיים ומגבלות :

הבעיה העיקרית בפרויקט נובעת מפיתוח האלגוריתם של היריב הממוחשב. אלגוריתם זה צורך הסתכלות על מספר רב של מהלכים קדימה בתוך משחק השחמט.

מצב זה מתקיל אותנו במכשול - מספר האפשרויות לביצוע מהלך על לוח שחמט אקראי הוא עצום. מתמטיקאי בשם קלוד שאנון קבע כי מספר המהלכים האפשריים הממוצע עבור זוג מהלכים (מהלך של השחקן הלבן, מלווה במהלך של השחקן השחור) במצב לוח אקראי הוא 10^3 , וכי משחק ממוצע מכיל כ-40 מהלכים כאלה.

הדבר מגביל באופן משמעותי את היכולת שלנו לחפש עמוק אל תוך המשחק, ודורש מאיתנו לייעל את הקוד כמה שאפשר, תוך שימוש באלגוריתמים מתקדמים (minimax, alpha-beta pruning).

קושי זה אף מתגבר כשאנו לוקחים בחשבון שהמערכת מבוססת על שפת התכנות פייתון, שפה אשר אינה יעילה ועלולה להאט את תהליך חיפוש המהלכים אף יותר.

תיחום הפרויקט :

הפרויקט עוסק בתקשורת בין מחשבים מרוחקים, מסדי נתונים, ונוגע בתחום האלגוריתם חיפוש (עצים בינאריים).

פרק ב' - אפיון הפרויקט

פירוט המערכת :

כאשר נכנסים לראשונה למערכת, יפגוש המשתמש בחלון התחברות. במידה ולמשתמש קיים חשבון בפלטפורמה, הוא יכול להתחבר בעזרת שם המשתמש והסיסמה של אותו חשבון. במידה ואין למשתמש חשבון, הוא יאלץ ליצור אחד בחלון ההרשמה.

לאחר שהמשתמש התחבר למערכת בהצלחה, הוא יפגוש בחלון בחירת משחק. בחלון זה ינתנו למשתמש מספר אופציות למשחק - משחק לוקאלי, משחק דרך הרשת, או משחק מול אחד משלושת ה"בוטים" (אלגוריתמים) של המערכת.

כאשר ילחץ המשתמש על אחת מהאופציות, ייפתח חלון בו יתנהל משחק השחמט, ולידו ניתוח המשחק (מי מנצח כעת), שמות השחקנים, ואופציה לפרוש. המשחק יימשך עד שאחד מהצדדים מנצח על ידי מט או הכנעת היריב (פרישה), או שהמשחק מגיע למצב של פט (תיקו).

לאחר שהמשחק נגמר יוכל לשוב המשתמש לחלון בחירת המשחק.

יכולות המערכת :

- הרשמה והתחברות
- ניהול משחק שחמט מול יריב מרוחק דרך הרשת
- ניהול משחק שחמט מול יריב ממוחשב בעזרת אלגוריתם חיפוש מתקדם
- ניהול משחק שחמט מול עצמך \ יריב לוקאלי (על אותו מחשב)
- ניתוח משחק (מי מנצח)

לוח זמנים :

פעילות	זמן התחלה	זמן סיום	הערות
פיתוח לוגיקת משחק	28.05	31.05	יצירת לוח, ביצוע מהלכים (ללא ממשק גרפי)
ממשק גרפי למשחק	31.05	1.06	לוח משחק
ניהול משחק לוקאלי	1.06	1.06	
הרחבת לוגיקת משחק	1.06	2.06	מצב ניצחון (מט), תיקו, הצרחה ו en-passant
ממשק מסך בית	5.06	5.06	
יריבים ממוחשבים אלגוריתמים	5.06	9.06	יריבים 1 ו 2
הרחבת ממשק גרפי למשחק	8.06	8.06	שמות משתמשים, כפתור פרישה, ניתוח משחק
הרחבת לוגיקת משחק	8.06	8.06	הוספת אופציה לפרישה
סאונד	11.06	11.06	
פיתוח שרת	11.06	14.06	שרת המנהל את המשחקים המרוחקים
הרשמה והתחברות	16.06	17.06	

פרק ג' - מסמך ניתוח

פירוט מעמיק של יכולות המערכת :

- הרשמה

- ממשק
- קליטת נתונים
- בדיקת תקינות
- שליחה לשרת
- בדיקת ייחודיות
- הצפנת הנתונים
- שליחת אישור למשתמש

- התחברות

- ממשק
- קליטת נתונים
- שליחה לשרת
- בדיקת התאמה
- שליחת אישור למשתמש
- התחברות המשתמש

- מסך בית

- ממשק
- אופציה להתחלת משחק לוקאלי
- אופציה להתחלת משחק דרך הרשת
- אופציה להתחלת משחק מול יריב ממוחשב
- בחירת צבע (עבור משחק מול יריב ממוחשב)

- ניהול משחק שחמט מול יריב מרוחק דרך הרשת

- ממשק משחק (לוח)
- ניהול משחק (ביצוע מהלכים, הצגת מהלכים אפשריים)
- התחברות לשרת
- מציאת שחקן נוסף דרך השרת
- העברת מהלכים בין משתמשים
- בדיקת תקינות מהלכים
- סגירת משחק במצב של ניצחון/תיקו/פרישה

- ניהול משחק שחמט מול יריב ממוחשב בעזרת אלגוריתם חיפוש מתקדם

- ממשק משחק (לוח)
- ניהול משחק (ביצוע מהלכים, הצגת מהלכים אפשריים)
- מציאת המהלך הטוב עבור היריב הממוחשב

תיק פרויקט - 'Chessy'

- ניהול משחק שחמט מול עצמך \ יריב לוקאלי (על אותו מחשב)
 - ממשק משחק (לוח)
 - ניהול משחק (ביצוע מהלכים, הצגת מהלכים אפשריים)

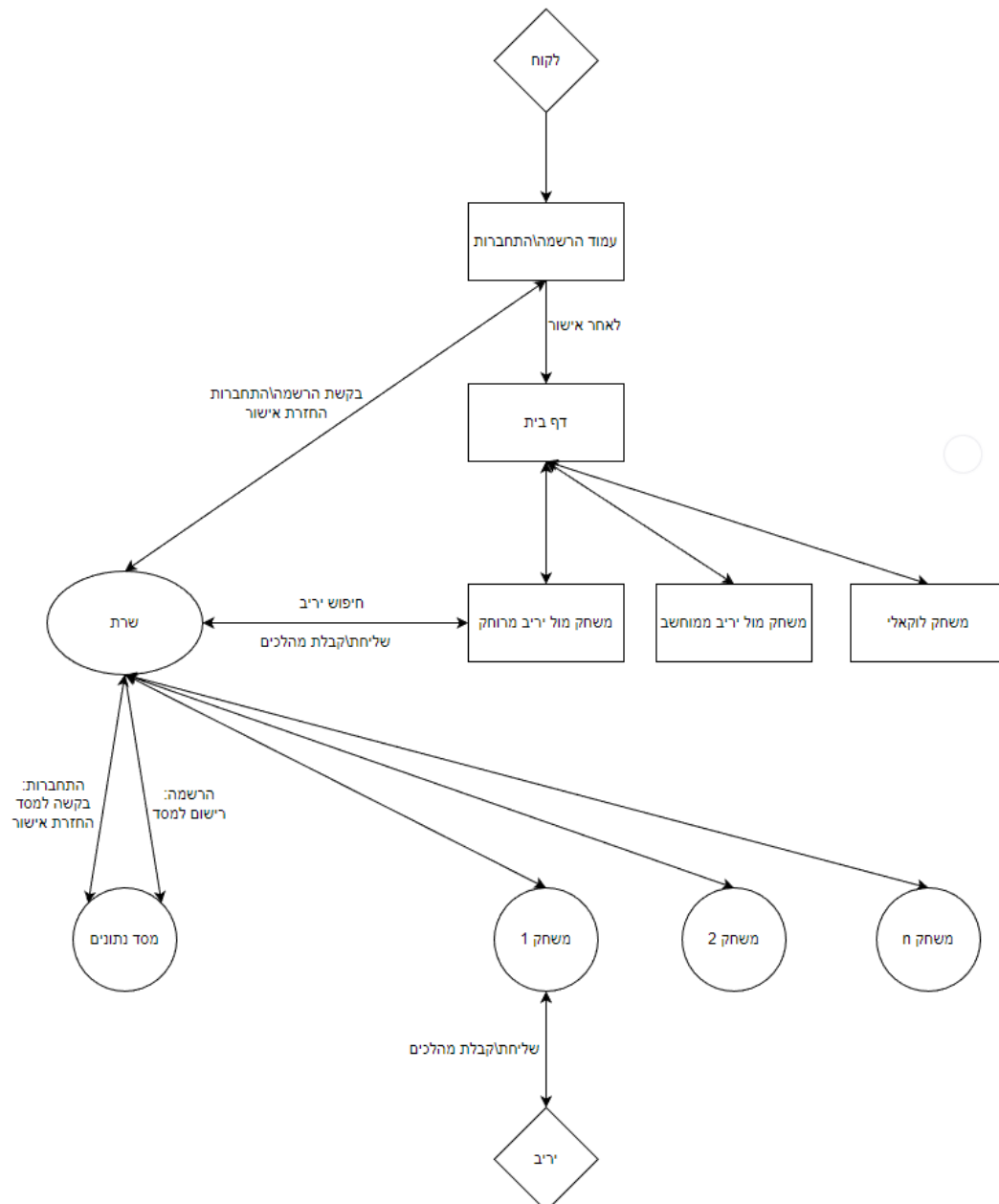
פרק ד' - מסמך ניתוח

תיאור מבנה המערכת :

המערכת כוללת עמוד הרשמה והתחברות אשר מובילים לעמוד בית, ממנו יכול לגשת הלקוח למספר אופציות משחק (מול יריב מרוחק, מול יריב ממוחשב ומול יריב לוקאלי על אותו מחשב).

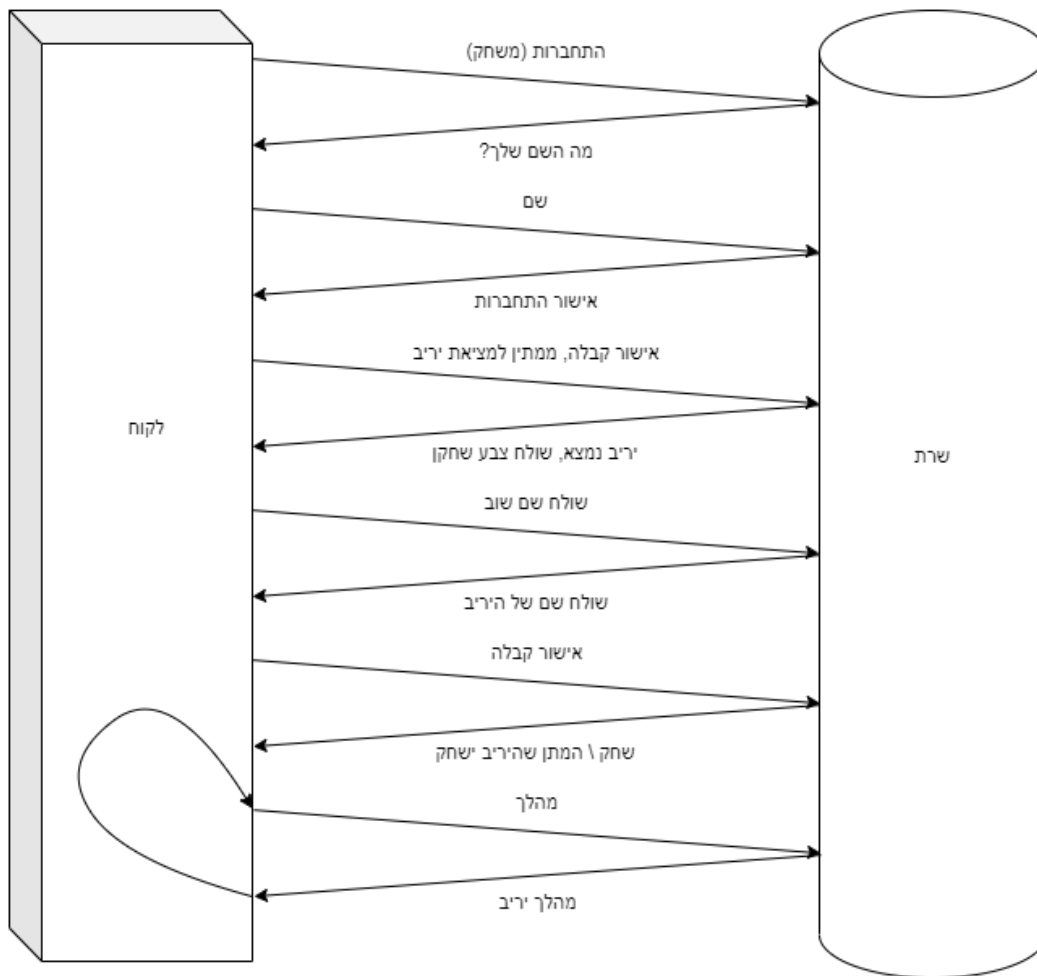
המערכת נעזרת בשרת אחד אשר מבדיל בין בקשות התחברות/הרשמה, לבין בקשות משחק (חיפוש משחק, העברת מהלכים וכדומה).

מבנה מערכת כללי:



תיק פרויקט - 'Chessy'

מבנה פרוטוקול תקשורת עם השרת בזמן משחק:



מבנה פרוטוקול תקשורת עם השרת לשם גישה למסד נתונים (הרשמה/התחברות):



תיאור הטכנולוגיה הרלוונטית :

בחרתי בשפת התכנות פייתון משום שעסקתי בה במהלך שנת הלימודים, ומתוך הנוחות והאינטואיטיביות שלה. במבט לאחור החלטה זו הייתה שגויה, משום שנוחות השפה באה על חשבון היעילות של השפה, דבר אשר הגביל את מהירות האלגוריתם של היריב הממוחשב.

נעשה שימוש בפרוטוקול התקשורת TCP משום שאין צורך להעברת מידע מהירה (שליחת מהלך כל מספר שניות עד דקות), ועדיף לוודא את אמינות המידע שהועבר.

נעשה שימוש במסד הנתונים sqlite מתוך נוחות והיכרות עם האופן בו עובד המסד.

תיאור מודולים מיובאים :

- chess
ספרייה אשר יוצרת סוג משתנה מיוחד מסוג Board המקבל פקודות להזזת כלים של משחק השחמט
- pygame
ספרייה המאפשרת הצגת ממשק גרפי, נוחה בעיקר ליצירת משחקים.
- sys
ספרייה המאפשרת ביצוע פקודות מערכת דרך פייתון
הספרייה יובאה אך ורק עבור השימוש ב `sys.maxsize`, הערך המקסימלי עבור משתנה integer אשר היה נחוץ במקרים מסוימים (אלגוריתם minimax)
- time
ספרייה המאפשרת הדפסת זמן נוכחי, כמו גם אופציה לחכות לזמן מסוים לפני המשכת הקוד (`time.sleep`).
- math
ספרייה המאפשרת ביצוע פעולות מתמטיות מתקדמות.
- socket
ספרייה המאפשרת תקשורת בין מחשבים דרך סוקטים.
- sqlite3
ספרייה המאפשרת בנייה וגישה למסד נתונים דרך פייתון.
- hashlib
ספרייה המאפשרת הצפנה של מידע בעזרת פונקציית `hash`.
נעשה שימוש במסד הנתונים לשם אבטחת מידע.

תיאור סביבת הפיתוח :

- שפת תכנות: python 3.8
- ממשק עיצוב: pygame
- סביבת עבודה: pycharm community
- מסד נתונים: sqlite3

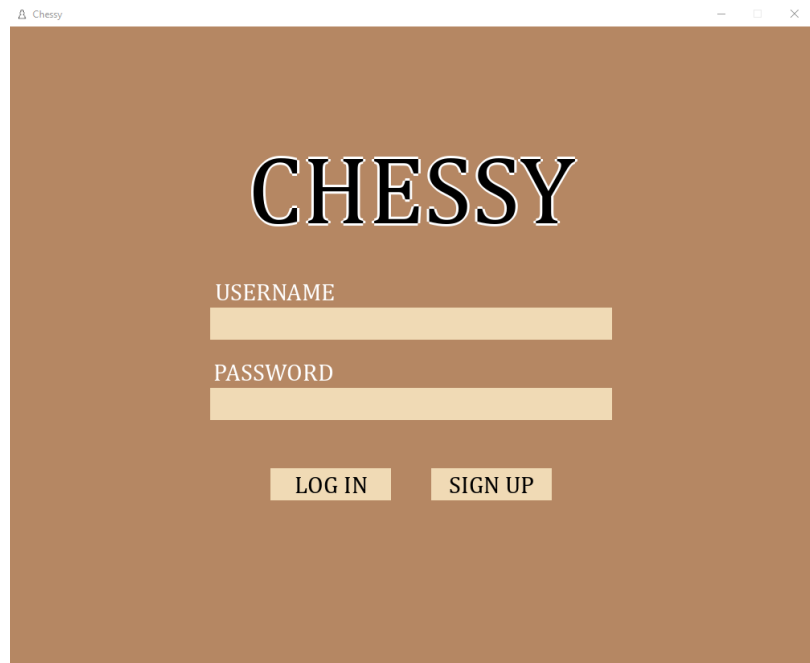
בעיות אלגוריתמיות :

- הבעיה הראשונה והעיקרית בפרויקט הייתה הצורך להסתכל מספר מהלכים קדימה לתוך משחק השחמט, ולנתח את הסיטואציה על מנת שהיריב הממוחשב יוכל לשחק בדרך הטובה ביותר.
הדבר מעלה בעיה קריטית מאוד. מכיוון שככל שנסתכל קדימה לתוך המשחק יוצרו יותר ויותר אופציות למהלכים באופן אקספוננציאלי (כ70 טריליון אופציות לאחר עשרה מהלכים בלבד!), משך הזמן שלוקח למצוא את המהלך הטוב ביותר עבור היריב הממוחשב עולה גם הוא באופן משמעותי.
הדבר מכריח אותנו להשתמש באלגוריתם חיפוש מתקדם, אשר נעזר בשיטות כמו minimax, דרך מהירה למצוא את האופציה הטובה ביותר מתוך מאגר של אופציות. ו alpha-beta pruning, דרך מהירה לפסילת אופציות (ענפים בעץ הבינארי) אשר אין להם פוטנציאל להתעלות על האופציה הנוכחית.
- בעיה נוספת הייתה האופציה לפרוש מהמשחק. כידוע במשחק השחמט לכל שחקן קיימת האופציה להיכנע ולהכריז שהוא פורש מהמשחק הנוכחי, בין אם בתור שלו או בתור של היריב שלו. הדבר מציב בעיה כאשר אנו משתמשים בפרוטוקול תקשורת המתבסס על פורמט של:
קבלת מהלך שחקן < שליחת מהלך ליריב < קבלת מהלך מהיריב < שליחת מהלך לשחקן היות והשרת לא מסוגל לקבל בקשות בזמן אמת מהשחקן במהלך התור של היריב שלו. הפתרון היה לאפשר לשחקן לשלוח בקשה לשרת בכל עת, ושהשרת יקרא אותה רק לאחר שהיריב סיים את תורו. לאחר סיום תורו, יקבל היריב את הודעת הפרישה של השחקן, והמשחק יסתיים עבורו.
הפתרון אינו אידיאלי, היות והוא לא מאפשר לשחקן לפרוש בזמן אמת (עבור השחקן המשחק יסתיים באותו עת, אך עבור היריב המשחק יסתיים רק לאחר שביצע את מהלכו), אך פתרון אחר ידרוש שינוי טוטאלי של כל פרוטוקול התקשורת לשם מטרה יחסית זניחה.

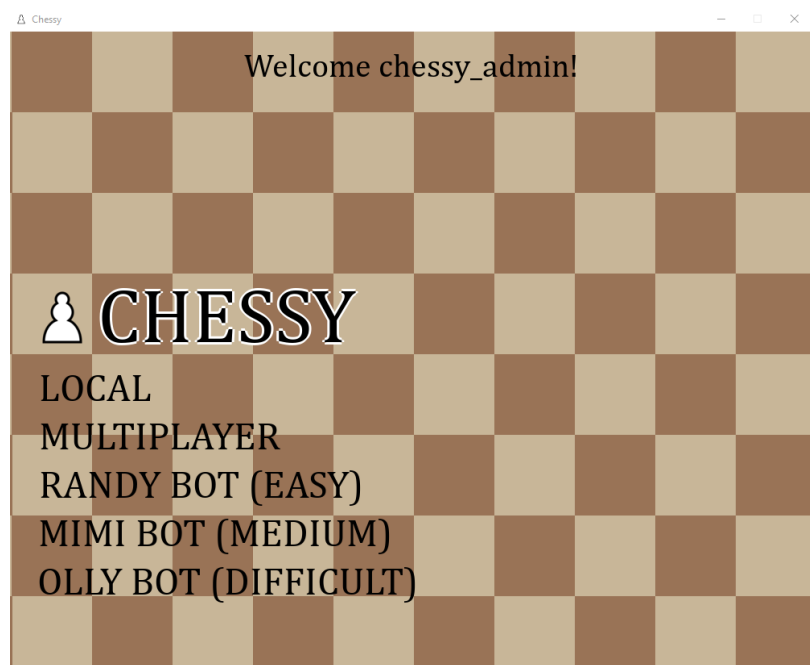
- בעיה נוספת בפיתוח הפרויקט הייתה הצורך להציג את לוח השחמט הפוך במקרים מסוימים. במקרה שהשחקן משחק בתור הכלים השחורים בשחמט, הוא יצטרך לראות את הלוח בהיפוך מראה על הציר האופקי של הלוח (כלים שחורים למטה, כלים לבנים למעלה). פתרון אחד אשר שקלתי ליישם היה בניית לוח שחמט נפרד לשחקן המשחק בתור הכלים השחורים, והתאמת כל הלוגיקה ללוח זה, אך פתרון זה דרש כתיבה מחדש של חלקים נכבדים בקוד. במקום זה, מצאתי פתרון יותר אלגנטי. הלוח בתוך הקוד ישאר כפי שהוא, אך בהצגה היוזואלית שלו, נקרא ונציג אותו מהסוף להתחלה כך שתיווצר אשליה שהוא הפוך על צירו האופקי. אך פתרון זה העלה עוד בעיה - כאשר השחקן ינסה לבצע מהלך עם הכלים הנמצאים למטה, אשר על מסכו של השחקן נראים כשחורים, הקוד יחשוב שהוא מנסה לבצע מהלך עם הכלים אשר נמצאים למטה בקוד המשחק - הלבנים. לכן, בקריאה לפעולה `find_square`, אשר נעזרת במיקום העכבר כדי למצוא את ריבוע המשחק עליו מצביע השחקן, נקרא את מיקום העכבר כאילו גם הוא הפוך על צירו האופקי (נחסיר את הערך האנכי של מיקום העכבר מגובה החלון המקסימלי).
- הבעיה האחרונה בה נתקלתי הייתה הצורך לבדוק מול השרת אם התקבל מהלך מהיריב בכל רגע נתון. היות ואנו משתמשים בפרוטוקול `tcp` נצטרך להוסיף פקודת `socket.recv` בכל הרצה של קוד המשחק (כל פריים). הדבר מציב בעיה משמעותית - שורת הקוד `socket.recv` ממתינה עד שהיא מקבלת תשובה מהשרת, ובכך עוצרת את הקוד עד שהיא מקבלת את אותה תשובה. דבר זה מוביל לתקיעת הממשק היוזואלי של המשחק וקריסתו. לשם פתרון בעיה זו, נעזרתי בפקודה `socket.settimeout`. שורת קוד זו אפשרה לי להחליט שעבור קבלת המהלך מהיריב הפקודה `socket.recv` תחכה 0.001 שניות עד שהיא תמשיך הלאה. אם בזמן זה התקבלה בקשה מהשרת - בצע אותה, אם לא - המשך הלאה בקוד. היות ובפרוטוקול `tcp` הודעות לא נמחקות במקרה שלא התקבלו, אלא נאגרות עד שהן נקראות, לא נוצר מצב בו הלקוח מפספס את בקשת השרת. בעזרת פתרון זה יוכל הלקוח להמשיך להשתמש בממשק היוזואלי של המשחק גם כאשר הוא מחכה למהלך מיריבו.

תיאור מסכי הפרויקט :

מסך הרשמה\התחברות:
מסך המופיע ברגע שפותחים את התוכנה. מביא ללקוח את האופציה להירשם לפלטפורמה או להתחבר למשתמש קיים.



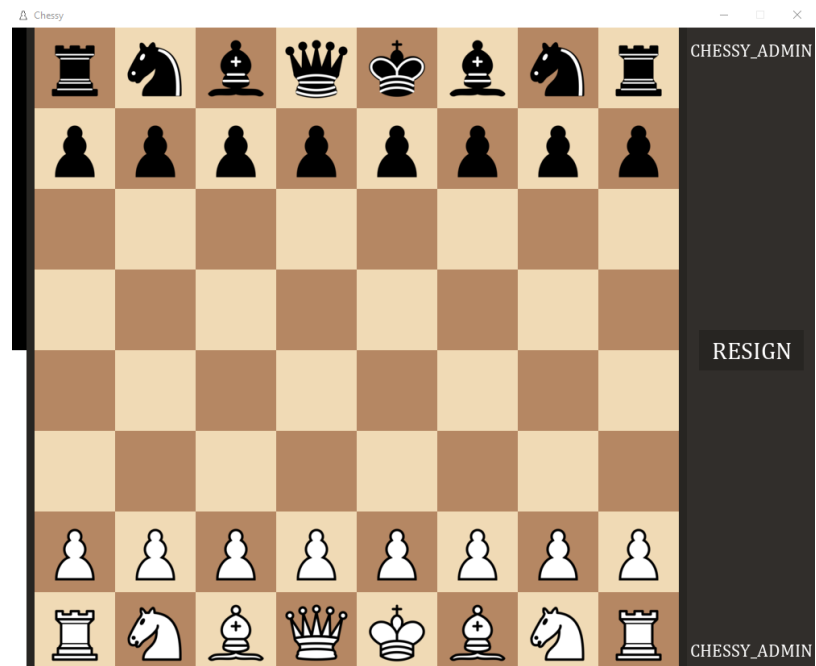
מסך בית:
מופיע מיד לאחר התחברות לפלטפורמה. מאפשר ללקוח להתחיל אחד משלושה סוגי משחק (לוקאלי, אונליין, נגד יריב ממוחשב).



תיק פרויקט - 'Chessy'

מסך משחק:

מופיע לאחר התחלת משחק מכל סוג שהוא. מציג את שמות היריבים, כפתור פרישה, הערכת משחק (מי מנצח) ואת לוח המשחק האינטראקטיבי.



מסך סיום משחק:

מופיע לאחר שהמשחק נגמר דרך מט, תיקו או פרישה של אחד מהשחקנים. לחיצה נוספת על המסך במצב זה תחזיר את המשתמש למסך הבית.



סקירת חולשות ואיומים:

- גניבת סיסמאות ממסד הנתונים:
המערכת מגינה נגד גניבת סיסמאות בעזרת הצפנת כל סיסמה במסד בעזרת פקודת ההצפה האי-סימטרית hash. כך נוכל לוודא כי הסיסמה שהמשתמש הזין בכדי להתחבר היא תקינה (השוואת תוצאות hash של הנתונים במסד והסיסמה שהזין המשתמש), וגם נגן מפני גניבת סיסמאות היות וגם אם פורץ יגיע למאגר הסיסמאות במסד הנתונים, לא יהיה לו שימוש בהן היות והן יהיו מוצפנות.
- sql injection:
המערכת מגינה מפני sql injection בכך שבודקת אם התווים שהזין המשתמש בכדי להירשם\להתחבר הם חוקיים, ולא מבצעת את פעולת הsql אל מול המסד במקרה שנמצאו תווים לא חוקיים.

פרק ה' - קטעי קוד נבחרים

תיאור מודולים שאני יצרתי :

- chessy

מודול מרכזי, מכיל את הרצת כל המודולים והחלונות הרלוונטיים בסדר הנחוץ.

שם פעולה	טענת כניסה	טענת יציאה
hovering_square	מיקום עכבר, צבע	מחזיר את ריבוע המשחק עליו מצביע עכבר המשתמש
find_sound	לוח, מהלך	מחזיר את הסאונד הרלוונטי אותו יש להשמיע על פי מצב הלוח הנוכחי
main	אין	מכיל את הרצת מסך ההרשמה\התחברות ומסך הבית, מוביל לפעולת game לאחר בחירת מצב המשחק
game	מצב משחק, צבע, שם משתמש	מכיל את הרצת המשחק על פי מצב המשחק שנבחר (לוקאלי, אונליין, יריב ממוחשב)

- register_menu

מודול זה מכיל את כל הממשק הכלול במסך ההרשמה\התחברות, כמו גם בדיקת תקינות שם משתמש וסיסמה.

שם פעולה	טענת כניסה	טענת יציאה
form	שם משתמש, סיסמה, מיקום עכבר	מכילה את ממשק ההרשמה\התחברות ומחזירה את הכפתור עליו מצביע עכבר המשתמש.
check_valid	מחרוזת	בודק אם שם המשתמש והסיסמה שהמשתמש הזין תקינים (תויים חוקיים, אורך מקסימלי 12).
message	מחרוזת	מציג הודעות במסך ההתחברות\הרשמה (הרשמה בהצלחה, סיסמה או שם משתמש שגויים\לא תקינים וכדומה).

• chess_board

מודול אשר אחראי להצגת לוח השחמט במהלך משחק, כמו גם הממשק הסובב אותו והמהלכים החוקיים שיכול לבצע השחקן.

שם פעולה	טענת כניסה	טענת יציאה
draw_board	לוח, צבע, מיקום עכבר, שם משתמש, שם יריב	מצייר את לוח השחמט ואת כלי המשחק הנמצאים עליו. כמו כן מציג את שם המשתמש ואת שם היריב.
draw_side_bar	מיקום עכבר, שם משתמש, שם יריב	מצייר את התפריט שליד לוח השחמט, כמו גם כפתור הפרישה.
draw_eval_bar	לוח	מציג ויזואל של הערכת המשחק (מי מנצח) ליד לוח השחמט.
show_legal_moves	ריבוע, צבע, לוח	מציג את המהלכים האפשריים לביצוע על ידי השחקן.

• evaluation_algorithms

מודול זה אחראי על היריב הממוחשב, והאלגוריתם למציאת המהלך בטוב ביותר עבורו. כמו
גם הערכת המשחק בכל רגע נתון (מי מנצח).

שם פעולה	טענת כניסה	טענת יציאה
render_all_moves	עומק, לוח	מבצע את כל המהלכים האפשריים עבור עומק משחק מסוים, ומחזיר את מספר המהלכים האפשריים.
moves_per_iteration	עומק	נעזר בפעולה render_all_moves בכדי למצוא את מספר המהלכים האפשריים עבור כל עומק עד עומק מסוים, ומחזיר את הזמן אשר לקח למצוא אותם.
evaluate_board	לוח, תור	מחזיר את הערכת המשחק (מי מנצח) עבור לוח משחק נתון.
generate_random_move	מהלכים	בוחר מהלך אחד באופן אקראי מתוך רשימה של מהלכים אפשריים.

• main_menu

מודול זה מכיל את כל הממשק הכלול במסך הבית.

שם פעולה	טענת כניסה	טענת יציאה
draw_main_menu	מיקום עכבר, צבע, שם משתמש, רקע	מצייר את מסך הבית והכפתורים המשוייכים אליו.

• messages

מודול זה אחראי להצגת הודעות למיניהן (שגיאה, סיום משחק וכדומה).

שם פעולה	טענת כניסה	טענת יציאה
display_message	מחרוזת, צבע	מציג הודעה כללית, אין שימוש ישיר בפעולה זו, אלא רק בתור חלק מפעולה אחרת.
game_over_message	מנצח	נעזר בפעולה display_message בכדי להציג את המנצח של המשחק על פי מצב הלוח הנוכחי.
looking_for_players_message	פריים	נעזר בפעולה display_message בכדי להציג את המנצח של המשחק על פי מצב הלוח הנוכחי. משתמש במספר הפריימים שעברו על מנת לדמות אנימצית המתנה.
server_error_message	אין	נעזר בפעולה display_message בכדי להציג הודעות שגיאה למיניהן.

• constants

מודול זה אחראי על אחסון משתנים קבועים של סאונד, תמונות, צבעים וכדומה.

מודול זה אינו מכיל פעולות.

• server

מודול זה אחראי על מציאת יריבים למשחק שחמט, פתיחת חדרי משחק רבים במקביל ושליחת מהלכים בין יריבים. כמו גם ניהול בקשות הרשמה והתחברות מול מסד הנתונים.

שם פעולה	טענת כניסה	טענת יציאה
main	אין	אחראי להקמת הסוקט של השרת ומקבל בקשות התחברות ממשתמשים. מבדיל בין בקשת משחק לבקשת התחברות\הרשמה ומוסיף את המשתמש לתהליך המתאים.
game	סוקט לקוח 1, סוקט לקוח 2	מנהל משחק בין שני יריבים. אחראי להתחלת המשחק וקביעת הצבעים, כמו גם קבלת והעברת מהלכים בין השחקנים.
register	סוקט לקוח	מנהל את הפנייה של המשתמש למסד הנתונים, מבדיל בין בקשות הרשמה והתחברות ומעביר אותן לפעולה המתאימה.
login	שם משתמש, סיסמה	מנהל פניית התחברות. בודק מול המסד אם הנתונים שהוזנו תואמים את הנתונים במסד ומחזיר אישור או ביטול במקרים המתאימים.
signup	שם משתמש, סיסמה	מנהל פניית הרשמה. מוודא מול המסד שלא מתקיימות כפילויות בשמות המשתמשים ומוסיף את המשתמש למסד במידה ואפשרי. שולח הודעת אישור.

קטעי קוד מיוחדים :

1. לולאת המשחק בצד השרת. אחראית לשליחת וקבלת המהלכים מהשחקנים.

```

96 while True:
97     # white's turn :
98     move = ""
99     # tell black to wait
100    client2.send("wait".encode())
101    while True:
102        move = client1.recv(1024)
103        if move:
104            # white has sent a move, send black and break out of the listening loop
105            client2.send(move)
106            print("CLIENT 1 PLAYED", move.decode())
107            break
108        # if the move resulted in game over, break out of the game loop entirely and close the thread
109        if move == "resign" or move[-4:] == "MATE":
110            break
111
112    # black's turn :
113    move = ""
114    # tell white to wait
115    client1.send("wait".encode())
116    while True:
117        move = client2.recv(1024)
118        if move:
119            # black has sent a move, send white and break out of the listening loop
120            client1.send(move)
121            print("CLIENT 2 PLAYED", move.decode())
122            break
123        # if the move resulted in game over, break out of the game loop entirely and close the thread
124        if move == "resign" or move[-4:] == "MATE":
125            break

```

2. פעולת render_all_moves. סוכמת את מספר המהלכים האפשריים עבור עומק מסוים בלוח מסוים.

```

8 def render_all_moves(depth, n_board):
9     # exit condition, depth has reached 0
10    if not depth:
11        return 1
12
13    # count the number of available moves per this iteration
14    move_sum = 0
15    for move in n_board.legal_moves:
16        # make move
17        n_board.push(move)
18        # recursively check all available moves branching from this move
19        move_sum += render_all_moves(depth - 1, n_board)
20        # unmake move
21        n_board.pop()
22
23    return move_sum

```

3. פעולת `evaluate_board`. מנתחת את הערכת המשחק הנוכחית (מי מנצח).

```
37 def evaluate_board(n_board, turn):
38     evaluation = 0
39     # values of each piece, king is worth infinitely many points
40     value_dict = {"P": 1, "N": 3, "B": 3, "R": 5, "Q": 9, "K": sys.maxsize,
41                  "p": -1, "n": -3, "b": -3, "r": -5, "q": -9, "k": -sys.maxsize}
42
43     # for each square on the board, if there is a piece on it, add its value to the total evaluation
44     for square in chess.SQUARES:
45         if n_board.piece_at(square):
46             evaluation += value_dict[n_board.piece_at(square).symbol()]
47
48     # black's evaluation is the exact opposite of white's evaluation
49     if turn:
50         return evaluation
51     return -evaluation
```

4. פעולת `evaluate_immediate_moves`. מנתחת לעומק של 1 (עד המהלך הבא) את כל המהלכים האפשריים ומוצאת את הטוב ביותר מביניהם.

```
54 def evaluate_immediate_moves(n_board, turn):
55     # lowest possible evaluation, base minimum value
56     best_moves_eva = [(-sys.maxsize, generate_random_move(n_board.legal_moves))]
57
58     for move in n_board.legal_moves:
59         # make move
60         n_board.push(move)
61
62         # if the move that has been played is equal in evaluation to the other best moves, append it to the best...
63         # ...moves list
64         if evaluate_board(n_board, turn) == best_moves_eva[0][0]:
65             best_moves_eva.append((evaluate_board(n_board, turn), move))
66
67         # if the move that has been played is greater in evaluation to the other best moves, remove all best moves...
68         # ...from the list and create a new one with just the recently played move
69         elif evaluate_board(n_board, turn) > best_moves_eva[0][0]:
70             best_moves_eva = [(evaluate_board(n_board, turn), move)]
71
72         # unmake move
73         n_board.pop()
74
75     # pick a move at random from the found equally best moves
76     best_moves = []
77     for move in best_moves_eva:
78         best_moves.append(move[1])
79     return generate_random_move(best_moves)
```

5. לולאת המשחק בצד לקוח (משחק לוקאלי)

```

81 for event in pygame.event.get():
82     # player has quit the game
83     if event.type == pygame.QUIT:
84         running = False
85     # player has clicked
86     if event.type == pygame.MOUSEBUTTONDOWN:
87         mouse_pos = pygame.mouse.get_pos()
88         # if the game is over, return false for game() and return to main menu
89         if stop_sound:
90             return False
91         # if the player is hovering over the resign button, resign
92         elif hovering_button != 0:
93             resigned = turn
94
95         # player has clicked within the bounds of the chess board
96         if mouse_pos[0] < 830:
97             # the player currently has a piece selected
98             if selected_piece != -1:
99                 move = chess.Move(selected_piece, hovering_square(mouse_pos, my_color))
100
101                 # pawn promotion if reached eighth rank
102                 if board.piece_type_at(selected_piece) == chess.PAWN \
103                     and (move.to_square in range(0, 8) or move.to_square in range(56, 64)):
104                     move.promotion = chess.QUEEN
105
106                 # if the move is legal, make it
107                 if move in board.legal_moves:
108                     board.push(move)
109
110                     # play appropriate sound and switch turns with opponent
111                     play_sound = find_sound(board, move)
112                     turn = not turn
113
114                 # regardless of whether the move has been made or not, deselect the piece
115                 selected_piece = -1
116
117             # the player currently has no piece selected
118             elif board.color_at(hovering_square(mouse_pos, my_color)) == turn and selected_piece == -1:
119                 # select the piece currently hovered over
120                 selected_piece = hovering_square(mouse_pos, my_color)

```

6. פעולת signup. אחראית לבדיקת תקינות הנתונים מול מסד הנתונים ורישום המשתמש לפלטפורמה.

```

49 def signup(username, password):
50     # establish connection with database
51     conn = sqlite3.connect("database.db")
52     c = conn.cursor()
53
54     # execute sql command to find if the selected username already exists
55     c.execute("SELECT * FROM users WHERE username='" + username + "'")
56
57     if len(c.fetchall()) != 0:
58         # a match has been found, signup unsuccessful
59         return False
60     else:
61         # no matches found
62         # hash password
63         password = hashlib.sha256(password.encode()).hexdigest()
64         # execute sql command to insert the new user's information to the database
65         c.execute("INSERT INTO users VALUES ('" + username + "', '" + password + "')")
66
67         # apply changes to db, signup successful
68         conn.commit()
69         return True

```

7. פעולת `draw_board`. אחראית לציור לוח המשחק וכלי המשחק שעליו. הופכת את הלוח על צירו האופקי במקרה הצורך וקוראת לפעולות המציירות את הממשק הסובב את הלוח.

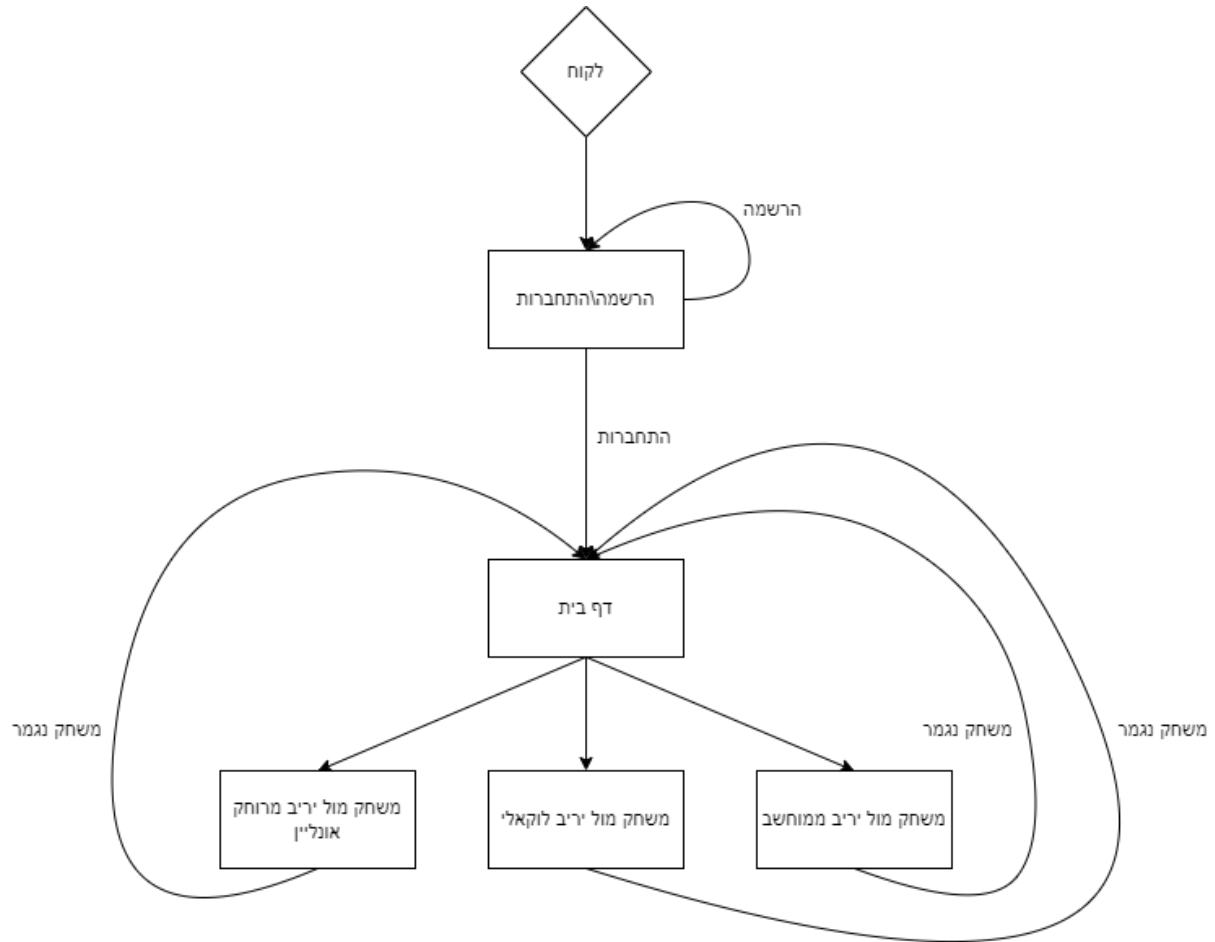
```

9 def draw_board(n_board, color, mouse_pos, my_name, opponent_name):
10     # for each square on the board, draw it with the appropriate color
11     # !! in practice only paints the dark squares, light squares are simply made as the background
12     for square in chess.SQUARES:
13         file = chess.square_file(square)
14         rank = chess.square_rank(square)
15         x = 30 + file * constants.square_size
16         y = (7 - rank) * constants.square_size
17
18         # if the player color is black, flip the y values (flip board across its horizontal axis)
19         if not color:
20             y = rank * constants.square_size
21
22         # file number + rank number is odd, meaning the square should be colored dark
23         if (file + rank) % 2 == 1:
24             pygame.draw.rect(chessy.win, constants.board_black, (x, y, constants.square_size, constants.square_size))
25
26         # if there is a piece at the currently selected square, draw it
27         if n_board.piece_at(square):
28             chessy.win.blit(constants.piece_dict[n_board.piece_at(square).symbol()], (x + 5, y + 5))
29
30     # draw side bar outline
31     pygame.draw.rect(chessy.win, constants.darker_gray,
32                      (8 * constants.square_size + 30, 0, 10, 800 * constants.square_size))
33     pygame.draw.rect(chessy.win, constants.dark_gray,
34                      (8 * constants.square_size + 40, 0, 2 * constants.square_size, 8 * constants.square_size))
35
36     draw_eval_bar(n_board)
37
38     # draws side bar and returns the currently hovered over button
39     return draw_side_bar(mouse_pos, color, my_name, opponent_name)

```

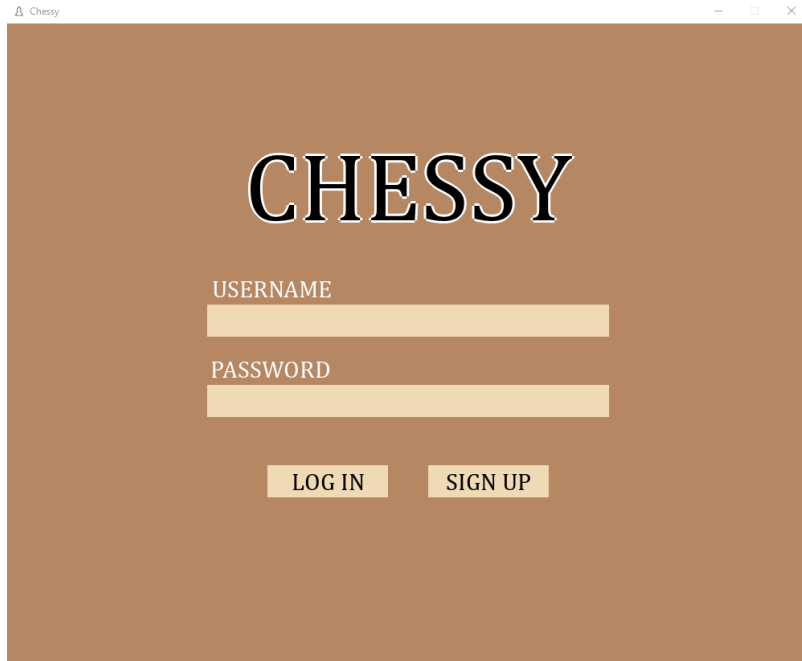

פרק ו' - מדריך למשתמש

דיאגרמת מערכת למשתמש :



מדריך למשתמש :

- שלב א' - הרשמה והתחברות
ברגע שתיפתח תוכנת Chessy לראשונה, יפגוש הלקוח במסך הרשמה והתחברות הכולל שתי תיבות טקסט, כפתור הרשמה וכפתור התחברות.



דרך מסך זה יידרש לקוח חדש לפלטפורמה ליצור משתמש חדש על ידי הזנת שם משתמש ייחודי וסיסמה, על פי הנחיות הפלטפורמה (לא יותר מ12, אך ורק תווים חוקיים), ולחיצה על כפתור ההרשמה (SIGN UP)

במידה ואירעה שגיאה בשלב ההרשמה, יש לוודא ששם המשתמש והסיסמה עוקבים אחר הנחיות הפלטפורמה. אם השגיאה טוענת כי משתמש בעל אותו שם משתמש כבר קיים, יש לבחור שם משתמש אחר.

אם ההרשמה צלחה, תופיעה הודעה ירוקה ויידרש המשתמש להתחבר לפלטפורמה.

במידה וללקוח קיים כבר משתמש בפלטפורמה, יידרש להזין את פרטיו (שם המשתמש והסיסמה) וללחוץ על כפתור ההתחברות (LOG IN)

במידה ואירעה שגיאה בשלב ההתחברות, יש לוודא כי הנתונים שהוזנו תואמים את הנתונים איתם נרשם הלקוח.

- שלב ב' - מסך הבית
לאחר שהלקוח התחבר בהצלחה למערכת, הוא יועבר למסך הבית. המסך כולל חמישה כפתורים המובילים לסוגי משחק שונים, וכפתור בחירת צבע בצורת חייל שחמט.



לחיצה על כפתור ה-LOCAL תכניס את הלקוח למשחק לוקאלי, בו כל המהלכים, הן של הלבן והן של השחור, נעשים על אותו המחשב.

לחיצה על כפתור ה-MULTIPLAYER תכניס את הלקוח למשחק אונליין מול יריב מרוחק, בו יבחרו לשני היריבים צבעים אקראיים.

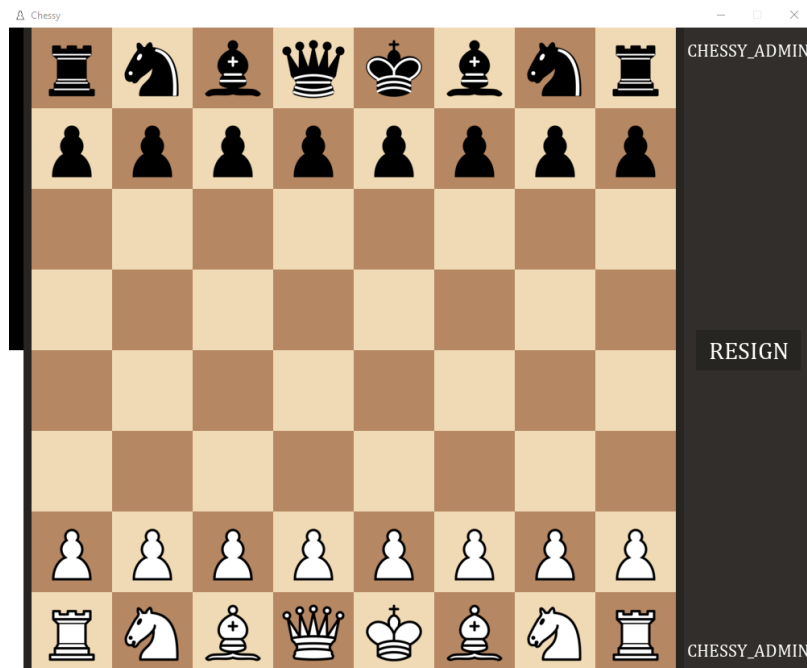
לחיצה על הכפתורים RANDY BOT, MIMI BOT, OLLY BOT תכניס את הלקוח למשחק מול יריב ממוחשב, בו ישחק הלקוח בתור הצבע שהוא בחר מול יריב הבנוי על אלגוריתם שחמט.

לחיצה על כפתור בחירת הצבע בצורת חייל השחמט תחליף את צבעו של החייל. אם החייל לבן ישחק הלקוח בתור הצד הלבן בשחמט. אם החייל שחור ישחק הלקוח בתור הצד השחור.

לאופציית בחירת הצבע ישנה השפעה אך ורק על המשחק הלוקאלי והמשחק מול היריב הממוחשב, היות ובמשחק אונליין הצבעים נבחרים באקראיות.

- שלב ג' - משחק

לאחר שהלקוח בחר את אחד מסוגי המשחקים לשחק בהם, הוא ישלח לחלון המכיל לוח שחמט אשר עליו מונחים כלי השחמט במיקום ההתחלתי. חלון זה יכול גם מד הערכת משחק, המעריך מי מנצח מבין שני היריבים, כפתור פרישה ואת שמות שני היריבים למשחק.



אם הלקוח משחק בתור הכלים הלבנים (הכלים הלבנים בחלק התחתון של המסך) הוא יבצע את המהלך הראשון. אם הלקוח משחק בתור הכלים השחורים (הכלים השחורים בחלק התחתון של המסך) הוא יצטרך להמתין עד שהיריב יבצע את מהלכו.

בכדי לבצע מהלך, יש ללחוץ על אחד מכלי המשחק שלך בזמן תורך. לאחר לחיצה על אחד מהכלים הכלי "ייבחר" ויודגשו כל ריבועי המשחק אליהם יכול הכלי הנבחר לעבור. לחיצה נוספת על כל מקום בלוח תבטל את בחירת הכלי, אך לחיצה על אחד מהריבועים אליו הוא יכול לעבור תבצע את המהלך ותעביר את התור ליריב.

המשחק יימשך עד שאחד מהתנאים הבאים יתקיימו:

1. אחד מהצדדים פרש על ידי שימוש בכפתור הפרישה (RESIGN).
2. אחד מהצדדים ניצח בעזרת מט.
3. המשחק הגיע למצב של פט (תיקו) בדרך כזו או אחרת.

תיק פרויקט - 'Chessy'

לאחר שהמשחק נגמר יוצג מסך בו רשום המנצח במשחק (או תיקו).



לחיצה נוספת על כל מקום במסך תוביל את הלקוח חזרה למסך הבית, בו יוכל שוב לבחור משחק חדש.

פרק ז' - מבט אישי

לקח לי לא מעט זמן להינעל על הרעיון לפרויקט. עבדתי על פרוייקטון שחמט אונליין עוד הרבה לפני תחילת העבודה על הפרויקט גמר. הגעתי איתו למצב שהוא עובד באופן טוב ויעיל, אך הקוד היה בלתי קריא בעליל ולקראת סוף העבודה שלי על הפרוייקטון לא היה לי מושג מה חצי מהקוד עושה, והקונספט בכללי כבר התחיל להימאס עליי.

לקח ממני לא מעט תעוזה לחזור שוב לרעיון המאוס הזה, אבל סרטון יוטיוב אחד היווה השראה גדולה עבורי, וחשף אותי לעולם המורכב והמרתק של אלגוריתמים של שחמט.

החלטתי שאני רוצה לחזור אל רעיון השחמט, להפוך אותו לפלטפורמה שתתאים להגשת פרויקט גמר, ולכלול בתוכו בונוס קטן עבורי - אלגוריתם שמשחק שחמט.

בעוד שבפרוייקטון הישן שלי יצרתי את המשחק כמעט מאפס, בפרויקט גמר הרשיתי לעצמי להיעזר בכלים פשוטים, ביניהם ספריית chess בפייתון, אשר הפכה את העבודה המייגעת עם מערך דו-מימדי לעבודה קלילה ופשוטה עם מחלקת Board שכלולה בספרייה.

הקלה זו, כמו גם היכרותי עם הלוגיקה הנחוצה הובילו לכך שהיה לי משחק שחמט לוקאלי בסיסי תוך פחות מיומיים, ומשחק גמור לחלוטין כולל ממשק ידידותי תוך ארבעה-חמישה ימים.

הרגשתי אופטימי לגבי המשך הפרויקט, אבל בשלב הזה כבר התחלתי לסבול מהטעויות הקטנות שעשיתי לאורך הדרך. בתחילת דרכי עם הפרויקט הייתי יהיר ולא תיארתי לעצמי שאצטרך לארגן את המבנה של המערכת מראש, אבל בשלב זה דברים כבר התחילו להשתבש כתוצאה מכך שהבסיס של הפרויקט לא התחבר באופן אלגנטי לחלקים אחרים בו.

מעידה נוספת באה לידי ביטוי בבניית האלגוריתם של היריבים הממוחשבים, מטלה אשר הייתה הרבה יותר קשה משציפיתי.

אך בסופו של דבר הצלחתי להגיע לתוצר סופי שאני שלם איתו. הוא בהחלט יכל להיות הרבה יותר מסודר, אבל לצערי הייתי קצר בזמן מבחינת הגשת הפרויקט.

אילו הייתי מתחיל את הפרויקט עכשיו, הייתי בוחר שיטת עבודה יעילה יותר, מסדר לעצמי לוחות זמנים ויוצר את המחלקות הנדרשות מלכתחילה.

אילו היה לי יותר זמן הייתי מוסיף לפרויקט אלגוריתמים יותר מתקדמים עבור היריבים הממוחשבים, שיטת דירוג ELO אשר נשמרת בין משחקים, matchmaking אשר מציב אותך מול יריבים ברמת ELO דומה לשלך וסוגי משחק שונים (chess 960, fairy chess, וכדומה)

פרק ח' - ביבליוגרפיה

- סרטון יוטיוב של Sebastian Lague על פיתוח אלגוריתם שחמט:
<https://www.youtube.com/watch?v=U4ogK0MIzqk>
- דוקומנטציה של ספריית chess בפייתון:
<https://python-chess.readthedocs.io/en/latest/>
- דוקומנטציה של ספריית pygame בפייתון:
<https://www.pygame.org/docs>
- מדריך יוטיוב של Avery Makes Games לספריית sqlite בפייתון:
<https://www.youtube.com/watch?v=HQQwgk6XklA>
- מאמר של PasiduPerera על פרויקט שחמט בפייתון:
<https://levelup.gitconnected.com/chess-python-ca4532c7f5a4>
- דף ויקיפדיה על "המספר של שאנון", וידוא תקינות חישובים:
https://en.wikipedia.org/wiki/Shannon_number
- מדריך של w3schools של sql:
<https://www.w3schools.com/sql>
- שאלה בstackoverflow על יצירת תיבת טקסט בספריית pygame (טעות חמורה):
<https://stackoverflow.com/questions/46390231/how-can-i-create-a-text-input-box-with-pygame>
- שאלה בstackoverflow על שימוש ב nonblocking socket בפייתון:
<https://stackoverflow.com/questions/16745409/what-does-pythons-socket-recv-return-for-non-blocking-sockets-if-no-data-is-r>
- מדריך של Lauri Hartikka לבניית אלגוריתם שחמט:
<https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977>