תיק פרויקט

שם בית ספר: תיכון על שם דוד קלעי.

שם העבודה: משחק סטרטגו.

שם התלמיד: טל דיבון.

תעודת זהות של התלמיד: 214159873

שם המנחה: איתי בן סעדון.

שם החלופה: הגנת סייבר (תכנון תוכנות מערכות).

01/06/24תאריך הגשה:



תוכן עניינים

מבוא.........................................................................3

יזום הפרויקט.............................................................3

אפיון הפרויקט............................................................7

תיאור תחום הידע - פרק מילולי...........................................10

יכולות צד לקוח..........................................................10

יכולות צד שרת..........................................................14

מבנה ארכיטקטורה........................................................17

מסמך עיצוב............................................................17

מימוש הפרויקט...........................................................34

סקירת כל המודולים / מחלקות המרכיבים את המערכת...................34

קטעי קוד מיוחדים / אלגוריתמים.........................................40

שלב הבדיקות..........................................................46

מדריך למשתמש..........................................................48

רפלקציה...................................................................55

בביליאוגרפיה..............................................................56

נספחים....................................................................58

מבוא

שם פרויקט: משחק סטרטגו.

הקדמה: מאז שהייתי קטן אהבתי לשחק משחקי קופסא. אם זה מונופול, דמקה או סולמות ונחשים.

הייתי משחק במשך שעות עם חברים ומשפחה ואחד המשחקים שאהבתי במיוחד היה משחק הסטרטגו. המשחק מיועד לשני שחקנים ואורכו בין עשר לעשרים דקות. אני חושב שמה שמשך אותי למשחק היה האסטרטגיה הרבה שהיה צריך להפגין בשביל לנצח והעובדה שהמשחק לא דורש מזל בכלל.

ניסיתי כמה וכמה פעמים למצוא אפליקציה או אתר אינטרנטי שבו אוכל לשחק את המשחק מול שחקנים מרחבי העולם. אבל אני אישית לא התחברתי למה שמצאתי באינטרנט ולפי דעתי הגרפיקה של המשחק לא הייתה מספקת.

התחלתי לתכנת בגיל צעיר יחסית אבל מעולם לא היה לי מורה או תוכנית למידה מסודרת, לכן עד התיכון, התכנות היה בסך הכל אחד התחביבים שלי. לאחר העלייה לתיכון נרשמתי למגמת הסייבר. שם לראשונה גיליתי שאני באמת אוהב לתכנת ושאני רוצה להמשיך לעסוק בזה בצבא ואז בחיים.

במסגרת המגמה התלמידים צריכים להגיש עבודת גמר בהיקף חמש יחידות לימוד. בתחילת השנה התחלתי להעלות רעיונות למה יהיה פרויקט הסיום שלי. בהתחלה, לא חשבתי על המשחק וחשבתי לבנות משהו אחר. לאחר שחבר סיפר שהוא רוצה לבנות משחק שש בש, חשבתי על הרעיון שגם אני אבנה משחק. ישר קפץ לי לראש משחק הסטרטגו והחלטתי שזה יהיה הפרויקט שלי.

הגדרת הלקוח: קהל היעד של המשחק הוא אנשים שאוהבים אסטרטגיה ומשחקי קופסה ושיש להם כרבע שעה פנויה. המשחק הינו מתאים לכל הגילאים. כל מה שצריך בשביל לשחק במשחק הוא מכשיר אלקטרוני כמו מחשב ואייפד וחיבור לאינטרנט.

אתגר מרכזי: האתגר המרכזי בשבילי היה לבצע חיבור בין השרת ללקוח. עשיתי כמה משימות במהלך השנתיים האחרונות בפייתון שכוללת שרת ולקוח. החלטתי שאת הצד לקוח לכתוב בגאווה סקריפט ולא בפייתון מכיוון שתכנתתי בשפה זו בעבר ואהבתי את הפשטות שבה אפשר ליצור אתרים. אבל אף פעם לא למדתי לבצע תקשורת בגאווה סקריפט אז האתגר המרכזי הראשון שלי היה ללמוד איך לבצע חיבור בין שרת בפייתון ללקוח בגאווה סקריפט.

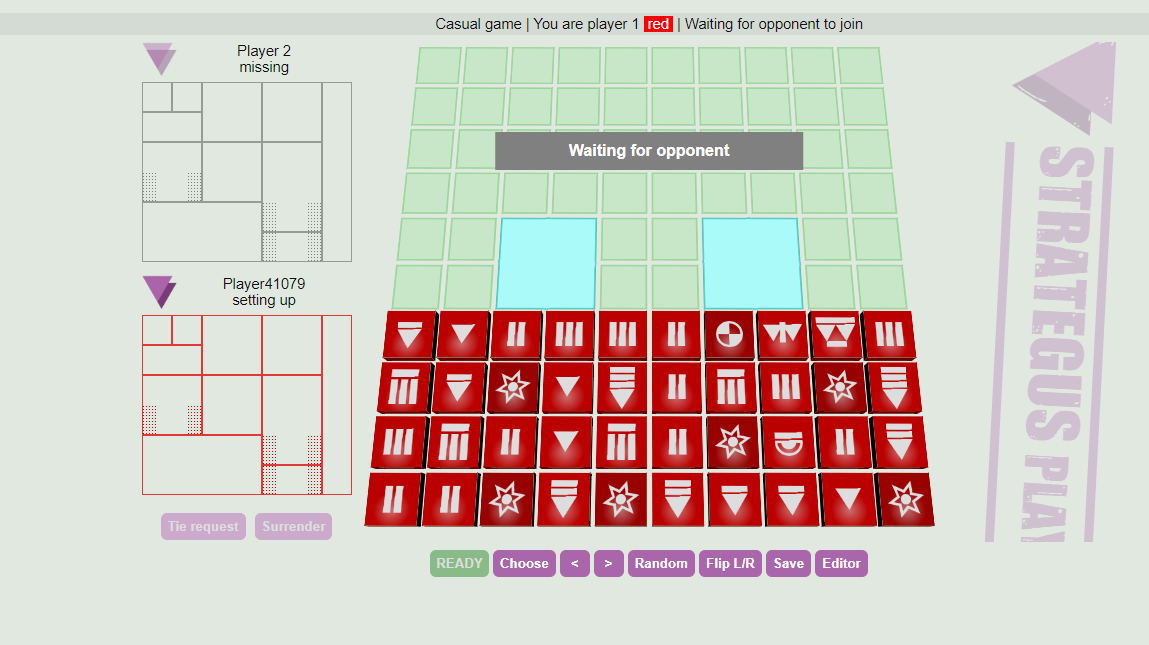
אתגר נוסף מרכזי בשבילי היה לייצר שרת מרובה משתתפים שידע להבדיל בין השחקנים והמשחקים השונים. בשביל להתגבר על הקושי יצרתי מחלקה שמטרה לנהל משחק (שעליה אני ארחיב בהמשך). במחלקה של השרת יצרתי מילון שהמפתח שלו הוא מזהה ייחודי והערך שלו הוא מופע של המחלקה "משחק". ככה שבכל פעם שיש בקשה לשרת לא צריך לעבור על כל המשחקים ולבדוק מה המשחק המתאים אלא ישירות לגשת אליו. ככה, השרת מוצא את המשחק במהירות רבה יותר וכשיש אלפי או עשרות אלפי משחקים בו זמנית החסכון הזה הוא הכרחי.

הגדרת יעדים:

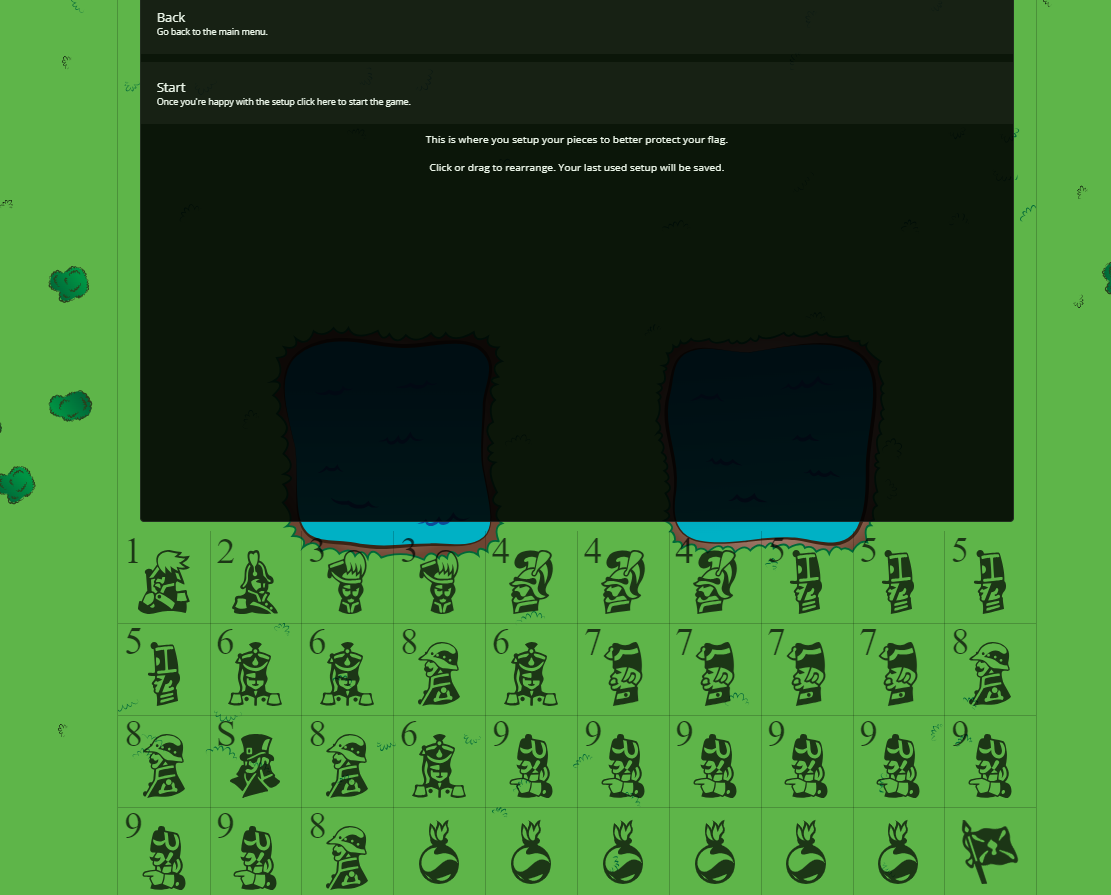
* הכנת משחק סטרטגו בין שני משתתפים.
* הכנת צאט שבו שני השחקנים יוכלו לדבר.
* הכנת שרת רב משתתפים שידע להריץ מספר משחקים בו זמנים.
* דף ליצירת משתמש.
* דף כניסת משתמש.
* שמירה מוצפנת של משתמשים.
* קשר מוצפן ומאובטח עם השרת אשר מאחסן את המידע.

סיקור מצב השוק כיום:

ישנם מספר אתרים שבהם אפשר לשחק סטרטגו באינטרנט אך בעיני הם לא נוחים ולא אטרקטיביים. לדוגמא:

<https://strategus.appspot.com/play>

<https://www.stratego.io/>



אלה דוגמאות למשחקים הקיימים בשוק. כשמחפשים סטרטגו אונליין שני האתרים הם בין האתרים הראשונים שעולים. בשביל למצוא שחקן אחר שישחק מולי רוב הפעמים צריך לחכות כמה דקות, בנוסף אני לא אוהב את הגרפיקה שהמשחקים מציעים.

בעיות תועלת וחסכנות:

אחת הבעיות העולות היא יצירת קשר מאובטח עם השרת. זה מצריך החלפת מפתח ציבורי אסימטרי ואז החלפת מפתח סימטרי. יצירת קשר מאובטח עשויה להיות מסובכת לעשייה בעיקר כשמדובר בשתי שפות תכנות שונות שמצריכות ספריות הצפנה שונות. יצירת תקשורת מאובטחת זו אולי משימה קשה למימוש, אך חשובה מאוד.

תיחום הפרויקט:

הפרויקט כולל בתוכו כמה תחומים בהם: OOP (תכנות מונחה עצמים) בו בניתי מספר מחלקות בצד השרת כמו מחלקה שמייצגת לוח או מחלקה שמייצרת משחק, ,database encryption, רשתות ועוד.

בהצפנה אני משתמש גם בפונקציית hash על מנת להצפין את הסיסמה וגם מצפין את התעבורה על מנת למנוע מקרה בו מישהו מקשיב לתעבורה וידע לפענח את התקשורת ולשנותה. תחילה מתבצעת החלפת מפתחות א-סימטרית, ולאחר מכן, שאר התקשורת היא דרך הצפנה סימטרית.

ברשתות השרת שלי הוא הוא שרת שמקבל בקשות דרך פרוטוקול http מסוג get הלקוח שלי שולח את הבקשה והשרת מפריד ממנו את הפרמטרים החשובים ומבצע את התוכנית בהתאם למה שהלקוח ביקש או שלח.

בdatabase אני שומר שם משתמש וסיסמה מוצפנת על מנת לשמור שחקנים. דבר נוסף שנשמר הוא הניקוד של השחקנים. מסד הנתונים שלי הוא מסוג sqLite. מבנה הנתונים נשמר בצד השרת.

במערכות ההפעלה, הפרויקט שלי עושה שימוש בריבוי ההליכונים (multi-threading) בכל פעם שנפתח משחק חדש אני משתמש בריבוי ההליכונים על מנת להציג את הזמן שנותר עד תחילת המשחק (כאשר השחקנים נמצאים בשלב ארגון הכלים) וכאשר אני רוצה להראות קלף מסוים לשני השחקנים לזמן מסוים.

המשחק שלי יכול לרוץ על כל מערכת הפעלה מכיוון שהמשחק רץ על דפדפן ופייתון היא שפת תכנות חוצה פלטפורמות.

אפיון הפרויקט

תיאור מפורט של המערכת- המשתמש מתחיל בדף הבית בו הוא יכול להירשם לאתר או ליצור משתמש חדש. בשני המקרים נשלחת בקשה לשרת שבודק את תקינות הבקשה. במקרה של יצירת משתמש. קודם כל נבדק שהשם משתמש והסיסמא עומד בדרישות שיפורטו בהמשך. במידה והמשתמש עומד בכל הדרישות נוצר משתמש חדש. השרת מחזיר ללקוח שהמשתמש נוצר שמציג הודאה למשתמש.

בנוסף, הלקוח עושה על הסיסמה פונקציית האש (hash) ככה שבמסד הנתונים נשמר רצף אותיות ומספרים ולא הסיסמא עצמה. זאת על מנת שבמקרה שפורצים למסד הנתונים הפורץ לא ימצא את הסיסמאות.

במקרה שהמשתמש כבר רשום וברצונו להיכנס למשתמש שלו. הוא שולח בקשת הרשמה. השרת פונה למסד הנתונים ובודק האם קיים משתמש והאם הסיסמא (המוצפנת ולא הסיסמה עצמה) היא אותו הדבר. במידה שכן הוא שולח שהמשתמש התחבר.

בנוסף יכול המשתמש לצפות בחוקי המשחק הכתובים בשפה האנגלית בשביל להיזכר או ללמוד.

לאחר ההרשמה המשתמש יכול ללחוץ על כפתור החל משחק. המחשב מחכה עד ששחקן שני מחכה, ולאחר מכן יוצר משחק חדש.

לאחר המשחק לשני השחקנים חמש דקות לארגן את הכלים שלהם בצורה שלפי דעתם תהיה הטובה ביותר. לאחר חמש הדקות כל שחקן משחק בתורו. המנצח הוא השחקן שמצא את דגל היריב.

לרשות השחקנים צאט שבו יוכלו לדבר אחד עם השני במהלך המשחק. בצאט מוצג מי מהשחקנים כתב מה.

לאחר סיום המשחק לשחקן המנצח יוספו נקודות ולשחקן המפסיד ירדו נקודות מהדירוג שלו.

פירוט היכולות:

* הרשמה לאתר.
* יצירת משתמש חדש.
* צפייה בדף ההוראות.
* משחק חדש דרך האינטרנט.
* דיבור עם השחקן היריב דרך צאט.

פירוט הבדיקות (קופסא שחורה):

* התחברות עם שם משתמש וסיסמה נכונים-הצגת כפתור להתחלת משחק.
* התחברות עם שם משתמש וסיסמה לא נכונים- הצגת הודעת שגיאה ששם המשתמש והסיסמה לא קיימים.
* יצירת שם משתמש וסיסמה בעלי פחות משלוש תווים- הצג הודעה מתאימה על כך ששם המשתמש או הסיסמה צריכים להכיל יותר משלושה תווים.
* יצירת שם משתמש וסיסמה בעלי יותר משש עשרה תווים- הצג הודעה מתאימה על כך ששם המשתמש או הסיסמה צריכים להכיל פחות משש עשרה תווים.
* יצירת משתמש חדש אך עם סיסמה וסיסמה חוזרת לא נכונה- הצג הודעה מתאימה על כך שהסיסמאות לא שוות.
* יצירת משתמש תקין- הצג הודעה מתאימה ועבור לדף ההתחברות.
* לחיצה על כפתור החל משחק- הצג מסך המתנה וחכה עד להתחלת המשחק.
* בזמן ההתארגנות גרור כלי אחד במקום כלי אחר שלך- החלף בין שני הכלים.
* בזמן ההתארגנות גרור כלי על מנת לשחק תור- אל תזיז את הכלי.
* לאחר שמתחיל המשחק הזז קלף אחד למקום ריק (מסומן במשבצת ירוקה)- העביר את הקלף למקום הזה.
* לאחר שמתחיל המשחק הזז קלף אחד לים (מסומן במשבצת כחולה)- אל תעביר את הקלף למקום הזה.
* לאחר שמתחיל המשחק הזז קלף אחד על הקלף של היריב (מסומן בתמונה של קלף יריב)- ראה שהקלף הגבוה מבין השניים נשאר.
* לאחר שמתחיל המשחק הזז קלף אחד על הקלף של היריב (מסומן בתמונה של קלף יריב) בעל אותו הרמה - ראה ששני הקלפים נעלמו.
* לאחר שמתחיל המשחק הזז קלף אחד כך שנעשה מהלך לא חוקי (מעבר של כמה משבצות) – ראה שהלוח נשאר ללא שינוי .
* לאחר שמתחיל המשחק הזז קלף אחד על פצצה של היריב - ראה שהקלף והפצצה נעלמו.
* לאחר שמתחיל המשחק הזז קלף אחד על הדגל של היריב -ראה שנגמר המשחק ומוצג הודעת ניצחון.
* כתוב הודעה בצאט ולחץ על שלח- ראה ששני הצדדים רואים את ההודעה.

תכנון וניהול הלוז:

ראשית בתחילת השנה התחלתי לשבת ולחשוב על רעיון לפרויקט. לאחר שהחלטתי שמשחק הסטרטגו יהיה הפרויקט שלי התחלתי לעבוד על איך המשחק יראה.

בהתחלה יצרתי בכלליות את לוח המשחק ואת התמונות שאני ארצה להשתמש בהם בעתיד. הכנתי את הסידור ההתחלתי של התמונות. קבעתי את מיקום הים והיבשה בהתאם ללוח המשחק. לאחר מכן עבדתי על כך שאצליח להחליף בין התמונות בעזרת העכבר. מה שבעתיד יסמל את הליכת הכלים.

לאחר עשיית הגרפיקה הבסיסית האתגר הבא שלי היה לבצע תקשורת בינתיים לא מוצפנת בין הלקוח שכתוב בגאווה סקריפט לשרת שכתוב בפייתון. אחר כך התחלתי לייצר את המחלקות של הלוח ושל הקלף ואת השרת שיעבד את בקשות הלקוח ויחזיר תשובות.

לאחר מכן התחלתי לבצע משחק בסיסי בין שני לקוחות. השרת עדיין לא תמך במספר משתתפים. ראיתי שאני מצליח לעשות מהלך ושהוא יופיע במשחק של השחקן השני.

החלקים הבאים כללו בדיקה של המהלכים האם הם חוקיים או לא ומה קורה כשכלי אחד תוקף כלי שני. מתן זמן התחלתי לשחקנים לייצר את הלוח לפי רצונם ובדיקת ניצחון והפסד.

לאחר כל השלבים האלה היה לי משחק בסיסי עובד. החלק הבא היה לוודא שהשרת יתמוך במספר משתמשים בו זמנית. הוספת צאט בין כל שני שחקנים שמשחקים בדיוק.

אחרי זה יצרתי את פונקציית ההתחברות וההרשמה, יצרתי את מסד הנתונים. הצפנת הסיסמא, בדיקת תקינות שם המשתמש והסיסמא ואת הדפים שיוצגו ללקוחות שייכנסו לאתר.

במצב הזה היה לי כבר אב טיפוס ראשוני והתחלתי לעבוד על התיק פרויקט.

לאחר מכן הוספתי הצפנות, שיפרתי את הגרפיקה ועשיתי עוד תיקונים קטנים.

ניהול סיכונים בפרויקט:

החלטתי לכתוב את הצד שרת בפייתון מכיוון שזוהי שפת התכנות המרכזית שלמדנו בכיתה בשנתיים האחרונות. אך החלטתי את הצד לקוח לכתוב javascript, css ו- html מכיוון שהם מספקים יכולות גרפיות בצורה פשוטה ואלגנטית.

אחד הסיכונים היה לכתוב את הקוד בשני שפות שונות. לכל שפה חוקים משלה ויתרונות משלה. בנוסף לא למדתי לפני הפרויקט איך מתבצעת התקשורת בגאווה סקירפט. זה היה הימור גם מכיוון שמעולם לא עשיתי את זה קודם.

תיאור תחום הידע

יכולות הלקוח:

שם היכולות: החלפת מפתחות

מהות: החלפת מפתחות התחלתית עם השרת – פעולה אוטומטית

אוסף יכולות נדרשות:

* קליטת מפתח ציבורי של השרת
* הכנת מפתח ציבורי
* הצפנה עם המפתח של הרשת
* שליחת מפתח ציבורי לשרת
* החזרת מפתח סימטרי אישי להמשך תקשורת

• אובייקטים נחוצים: הצפנה / פענוח, תקשורת

שם היכולות: חיבור לשרת

מהות: חיבור לשרת- מתחילה אחרי יצירת משתמש או התחברות.

אוסף יכולות נדרשות:

* התחברות לשרת
* החלפת מפתחות
* קליטת כינוי ממשתמש
* הצפנה
* שליחה לשרת
* הרצת ממשק התחברות

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת

שם היכולות: הרשמה למערכת.

מהות: רישום משתמש חדש במערכת – קליטת פרטים אישיים נדרשים

אוסף יכולות נדרשות:

* ממשק משתמש – מסך הרשמה
* קליטת נתונים
* הצפנה
* בדיקת הקלט
* שליחה לשרת
* רשימה במאגרי המידע
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת, בסיס נתונים

שם היכולות: התחברות למערכת

מהות: חיבור משתמש קיים למערכת – קליטת פרטים אישיים נדרשים

אוסף יכולות נדרשות:

* ממשק משתמש – מסך התחברות
* קליטת נתונים
* הצפנה
* בדיקת הקלט
* שליחה לשרת המרכזי
* בדיקה מול מאגרי המידע
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש
* טעינת כפתור חדש

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת, בסיס נתונים

שם היכולות: הצגת ההוראות

מהות: הצגת ההוראות למשתמש בצורה נוחה.

אוסף יכולות נדרשות:

* ממשק משתמש-דף הוראות
* טעינת תמונות וכפתורים.
* עיבוד הלחיצה וחזרה לדפים אחרים.

אובייקטים נחוצים: ממשק משתמש.

שם היכולות: התחברות למשחק חדש.

מהות: בקשה להתחיל משחק- לחקות עד ששחקן אחר יצטרף.

אוסף יכולות נדרשות:

* ממשק משתמש – לחיצה על כפתור
* הצפנה
* שליחה בקשה לשרת המרכזי
* המתנה לשחקן חדש
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש
* טעינת משחק

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת

שם היכולות: הזזת כלי.

מהות: הזזת הכלי- השחקן גורר את הכלי ממשבצת אחת לשנייה.

אוסף יכולות נדרשות:

* ממשק משתמש – הצגת הלוח וגרירת התמונה
* הצפנה
* שליחת הודעה לשרת המרכזי
* פענוח המעבר (האם חוקי או לא)
* קבלת תשובה מהשרת
* פענוח
* שינוי הלוח
* טעינת משחק

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת

שם היכולות: דיבור דרך הצאט.

מהות: בקשה להתחיל משחק- לחקות עד ששחקן אחר יצטרף.

אוסף יכולות נדרשות:

* ממשק משתמש – קליטת הטקסט והצגתו למשתמשים
* הצפנה
* שליחת הטקסט לשרת המרכזי
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש
* טעינת הטקסט החדש לשני המשתמשים.

אובייקטים נחוצים: ממשק משתמש, הצפנה /פענוח, תקשורת

יכולות צד שרת:

שם היכולות: החלפת מפתחות

מהות: החלפת מפתחות עם לקוח כאשר הוא מתחבר

אוסף יכולות נדרשות:

* הכנת מפתח ציבורי ופרטי
* שליחת מפתח ציבורי ללקוח
* פענוח קלט עם מפתח פרטי
* החזרת מפתח סימטרי של לקוח להמשך תקשורת

אובייקטים נחוצים: הצפנה / פענוח, תקשורת.

שם היכולות: חיבור לקוח חדש

מהות: חיבור לקוח חדש למערכת. הלקוח שולח את שם המשתמש שלו.

אוסף יכולות נדרשות:

* האזנה
* התחברות המשתמש
* החלפת מפתחות
* קליטת שם משתמש מהלקוח
* פענוח
* המתנה ללקוח נוסף על מנת להתחיל משחק ביניהם.

אובייקטים נחוצים: הצפנה / פענוח, תקשורת

שם היכולות: אימות משתמש

מהות: אימות פרטי התחברות של משתמש עם מאגר נתונים – קבלת פלט מלקוח נדרש

אוסף יכולות נדרשות:

* קבלת בקשה מהמשתמש
* פענוח
* בדיקת תקינות הבקשה
* בדיקה עם מאגר הנתונים
* ההצפנה
* החזרת תשובה למשתמש

אובייקטים נחוצים: הצפנה / פענוח, תקשורת, בסיס נתונים

שם היכולות: רישום משתמש

מהות: רישום פרטי התחברות של משתמש במאגר נתונים – קבלת פלט מלקוח נדרש

אוסף יכולות נדרשות:

* קבלת בקשה מהמשתמש
* פענוח
* בדיקת תקינות הבקשה
* רשימה במאגר הנתונים
* הצפנה
* החזרת תשובה למשתמש

• אובייקטים נחוצים: הצפנה / פענוח, תקשורת, בסיס נתונים

שם היכולות: יצירת משחק חדש

מהות: התחלת משחק עם שני לקוחות.

אוסף יכולות נדרשות:

* החזרת מידע ללקוח
* פענוח
* בדיקת תקינות הבקשה
* שמירת המשחק והפרדתו ממשחקים אחרים
* הצפנה
* שימוש בthreads

אובייקטים נחוצים: הצפנה / פענוח, תקשורתthreads,.

שם היכולות: צאט בין משתתפים

מהות: יצירת צאט כך שהשחקנים שמשחקים יחדיו יוכלו לדבר.

אוסף יכולות נדרשות:

* החזרת מידע ללקוח
* פענוח
* שמירת הצאט והפרדתו מצאטים אחרים
* הצפנה
* החזרת תשובה למשתמש

אובייקטים נחוצים: הצפנה / פענוח, תקשורת.

שם היכולות: החזרת הלוח

מהות: החזרת הלוח למשתמש כמערך דו ממדי

אוסף יכולות נדרשות:

* החזרת מידע ללקוח
* פענוח
* בדיקת הבקשה
* החזרת לוח ספציפי
* הצפנה

אובייקטים נחוצים: הצפנה / פענוח, תקשורת.

שם היכולות: שינוי זמן והחזרתו.

מהות: החזרת הזמן ללקוח עד להתחלת המשחק וסיום זמן ההתארגנות.

אוסף יכולות נדרשות:

* שינוי הזמן
* החזרת הזמן בצורה נוחה
* החזרת הזמן ללקוח
* פענוח
* בדיקת הבקשה
* הצפנה
* שימוש בThreads.

אובייקטים נחוצים: הצפנה / פענוח, תקשורת, threads.

חלק :3 מבנה – ארכיטקטורה

תיאור הטכנולוגיה הרלוונטית:

שפת תכנות – Python. כל הצד שרת שלי כתוב בפייתון מלבד הפקודות שקשורות למסד הנתונים שכתובים ב- SQLite. פייתון היא שפה פשוטה למדי אשר מאפשר כתיבה נוחה יותר של שפת הקוד. בחרתי בשפה זו מכיוון שזוהי שפת התכנות העקרית שלמדנו במגמה ובגלל הפשטות ומרחב הכלים שהיא מציעה.

שפת תכנות javascript- JavaScript היא אחת משפות הפיתוח הכי פופולריות בתעשייה. כיום JavaScript משמשת בין היתר לפיתוח אתרים ואפליקציות full stack . JavaScript היא שפת תכנות דינמית המאפשרת להוסיף פונקציונליות אינטראקטיבית לדפי אינטרנט. ניתן להשתמש בה כדי ליצור אנימציות, משחקים, טפסים ועוד.

Html- שפת תגיות לתצוגה ועיצוב דפי אינטרנט ותוכן לתצוגה בדפדפן. זו שפת התגיות המרכזית בעולם האינטרנט, המהווה שלד למרבית עמודי התוכן באינטרנט. השפה מאפשרת עיצוב תוכן בצורה מהירה, קלה ללימוד באופן יחסי וקלה לכתיבה.

- Css היא שפת תכנות המשמשת לעיצוב חזותי של דפי אינטרנט. היא מאפשרת לשלוט בצבעים, בגופנים, בפריסה ועוד, ובכך מעניקה לאתר מראה ייחודי ומותאם אישית.

בעזרת JavaScript html ו-CSS כתבתי את הצד לקוח ואת הדפים המוצגים למשתמשים. Htmlמהווה את השלד של האתר, בעזרת css אפשר לעצב את האתר. JavaScript מוסיפה את "המוח" שמאחורי הקוד.

מסד נתונים – SQLite. מסד הנתונים אשר משמש אותי בכדי לשמור את נתוני

המערכת הוא מסד נתונים מסוג SQLite, אשר מחובר בצד השרת. הוא יחסית פשוט לשימוש.

תיאור זרימת המידע במערכת:

יכולות הלקוח:

1. קליטת מידע מהשרת – על הלקוח לקלוט מהשרת את המפתח הציבורי שלו.

יצירת מפתח סימטרי והצפנה שלו עם המפתח שהתקבל מהשרת.

שליחת המפתח המוצפן לשרת.

החזרת מפתח אישי לקוד הלקוח להמשך התקשורת והצפנת הודעות.

2. הרצת ממשק המשתמש לרישום המשתמש.

קבלת קלט של שם משתמש, סיסמה ואימות סיסמה מהמשתמש.

בדיקה שהשם משתמש והסיסמה עומדים בדרישות.

בדיקה שהסיסמה והאימות סיסמה שווים.

הצפנה עם המפתח הסימטרי ושליחה לשרת.

קבלת תשובה מהשרת ופענוח ההודעה.

במידה והמשתמש נקלט העבר לדף הכניסת משתמש ובמידה שיש בעיה להציג הודעה מתאימה.

3. הרצת ממשק המשתמש לכניסת המשתמש.

קבלת קלט של שם משתמש וסיסמה מהמשתמש.

בדיקה שהשם משתמש והסיסמה עומדים בדרישות.

הצפנה עם המפתח הסימטרי ושליחה לשרת.

קבלת תשובה מהשרת ופענוח ההודעה.

הצגת כפתור החל משחק.

4. המתן לתחילת המשחק. מותנה בלחיצת כפתור החל משחק.

יצירת בקשה של האם התחיל המשחק. הבקשה תשלח כל שנייה.

הצפנה עם המפתח הסימטרי ושליחה לשרת.

קבלת תשובה מהשרת ופענוח ההודעה.

במידה שהתחיל משחק תיצור את הלוח ואת הצאט.

במידה ולא שלח בקשה בשנית.

5. עדכן את הלוח, את הזמן ואת הצאט כל שניה.

יצירת בקשה של עדכון משחק. הבקשה תשלח כל שנייה.

הצפנה עם המפתח הסימטרי ושליחה לשרת.

קבלת תשובה מהשרת ופענוח ההודעה.

עדכן את כל הרכיבים ובדוק האם התקבל ניצחון או הפסד.

במידה ולא המשך לשלוח את הבקשה.

6. לאחר קבלת ההודעה של עדכון המשחק. במקרה שנגמר המשחק.

הצג הודעה מתאימה למשתמש.

עדכן את ניקוד המשתמש.

הסר את הלוח ואת כל המרכיבים.

הצג את כפתור החל משחק. בלחיצה עליו חזור לארבע.

יכולות השרת:

1. האזנה לפורט ספציפי. במקרה של המשחק שלי בחרתי בפורט 8000

2. יצירת צמד מפתחות א-סימטריים

שליחה ללקוח של מפתח ציבורי

קליטת הודעה מהלקוח ופענוח עם מפתח פרטי

שמירת המפתח הסימטרי.

3. קליטת בקשה מהמשתמש ופענוח

הפרדת המשתנים השונים שקיבלנו מהמשתמש ובדיקת תקינותם.

4. במקרה שהמשתנה שמחזיק את הפעולה הוא הירשמות לאחר הפענוח.

הפרדת השם משתמש והסיסמה משאר הפרמטרים.

פתיחת מסד הנתונים.

בדיקה ששם המשתמש אינו קיים. במקרה שלא קיים הוסף את שם המשתמש והסיסמה ועדכן את הניקוד ההתחלתי.

סגירת מסד הנתונים.

הכן הודעה מתאימה בהתאם להאם המשתמש נרשם או לא.

הצפנת ההודעה בעזרת המפתח הסימטרי.

החזרת ההודעה ללקוח.

5. במקרה שהמשתנה שמחזיק את הפעולה הוא כניסת משתמש לאחר הפענוח.

הפרדת השם משתמש והסיסמה משאר הפרמטרים.

פתיחת מסד הנתונים.

בדיקה ששם המשתמש קיים והסיסמה המוצפנת שקיבלנו שווה לסיסמה השמורה.

סגירת מסד הנתונים.

הכן הודעה מתאימה בהתאם להאם המשתמש נמצא או לא.

הצפנת ההודעה בעזרת המפתח הסימטרי.

החזרת ההודעה ללקוח.

6. במקרה שהמשתנה שמחזיק את הפעולה הוא לחקות למשחק לאחר הפענוח.

במידה וקיים משתמש אחר שכבר מחכה החל משחק בין שני השחקנים.

במידה ולא קיים משתמש אחר שמור את המזהה של השחקן עד ששחקן אחר נכנס.

במידה והתחיל המשחק צור משתנה יחודי שיהווה את מזהה המשחק.

הצפן אותו והחזר אותו לשני השחקנים.

במידה ולא החזר שעדיין לא נמצא שחקן נוסף (הצפן לפני השליחה).

7. במקרה שהמשתנה שמחזיק את הפעולה הוא לשחק טור למשחק לאחר הפענוח.

מצא את המשחק הספציפי על פי המזהה היחודי שהשחקן שלח.

בדוק האם תורו של השחקן לשחק

בדוק האם הצעד עומד בחוקים של המשחק.

במקרה שכן שחק את הצעד ושמור אותו.

8. במקרה שהמשתנה שמחזיק את הפעולה הוא לשלוח טקסט למשחק לאחר הפענוח.

מצא את המשחק הספציפי על פי המזהה היחודי שהשחקן שלח.

הוסף לטקסט של המשחק את מהשחקן שלח וכתוב וזהה את הטקסט על פי המזהה של השחקן.

9. במקרה שהמשתנה שמחזיק את הפעולה הוא קבלת עדכון על המשחק לאחר הפענוח.

מצא את המשחק הספציפי על פי המזהה היחודי שהשחקן שלח.

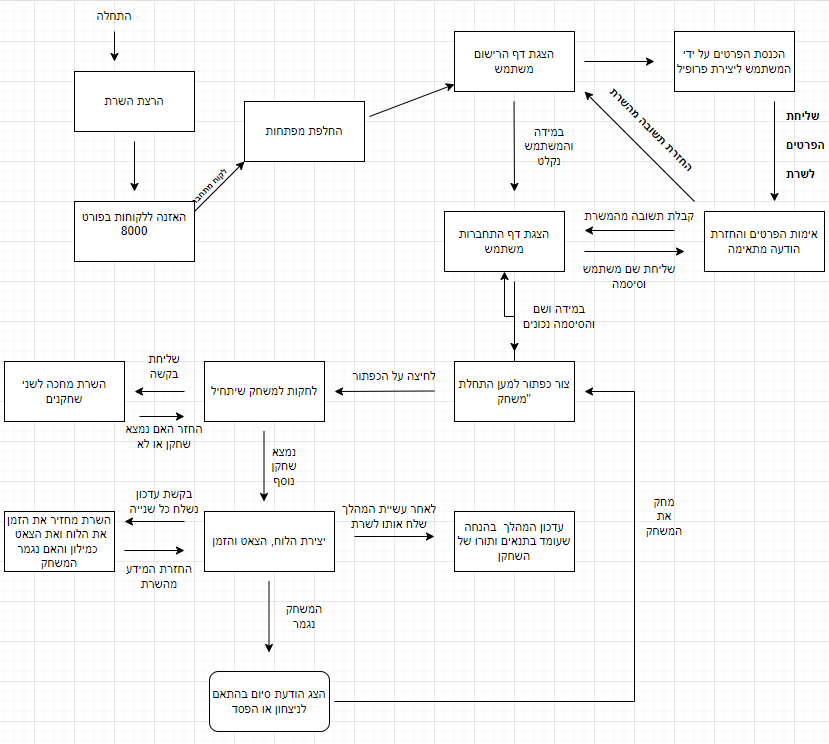
הכן את הלוח כמערך דו ממדי שבו כל מקום הוא סוג הכלי.

הכן את הטקסט כרצף של שורות.

הכן את הזמן של השחקנים.

שמור את כולם כמילון.

הצפן אותו והחזר למשתמש.

הצגת זרימת המידע ביו הרכיבים המרכזיים במערכת באמצעות שרטוט:

תיאור סביבת הפיתוח:

1. Python: הצד שרת ייכתב בשפת התכנות Python, יחד עם הספריות והמודולים

הדרושים לפרויקט שלי.

2. סביבת העבודה שלי לכתיבת השרת היא :Pycharm

Pycharm היא סביבת פיתוח משולבת לפיתוח תוכנות בעיקר בשפת פייתון, אשר פותחה על ידי תאגיד התוכנה הצ'כי JetBrains. הסביבה מספקת שירותים כגון ניתוח קוד וניפוי שגיאות בקוד גרפי. בחרתי בסביבה זו כי היא סביבת עבודה נוחה, מציעה כלים שעוזרים לעבודה וכי למדו לתכנת python בסביבת עבודה זו במסגרת הלימודים שלנו.

3. את הצד לקוח אני כתוב ב html, javascript, css.

4. סביבת העבודה לצד הלקוח הוא notepad++ .Notepad++‎ הוא עורך טקסט ועורך קוד מקור ל-Windows. התוכנה מציעה מגוון רחב של אפשרויות מעבר לפנקס רשימות המובנה בחלונות, כגון שימוש בלשוניות. התוכנה נוחה מאוד לכתיבת צד הלקוח שלי.

5. את הרצת השרת אני מריץ דרך דפדפן .chrome

6. את מסד הנתונים אני מריץ דרך SQLite. SQLite היא מערכת מסד נתונים קלת משקל מבוססת קבצים שבה אעשה שימוש לאחסון מידע משתמשים והניקוד שלהם.

תיאור התקשורת:

לאחר הרצת השרת, השרת מקשיב לפורט מסוים. במקרה שלנו השרת מקשיב לפורט 8000. התקשורת מתבצעת דרך פרוטוקול http. הלקוחות השונים שולחים בקשות אל השרת, השרת מעבד את הבקשות ובחלקן מחזיר תשובה.

לאחר הרצת לקוח, הלקוחות השונים מבקשים מהמשרת את המפתח הציבורי ומקבלים אותו. הם מצפינים בעזרתו את המפתח הסימטרי ומחזירים אותו לשרת מוצפן. השרת

לדוגמא כשהשחקן רוצה לבצע מהלך או לשלוח טקסט השרת לא יחזיר תשובה.

כשמדובר על קבלת מזהה יחודי או קבלת עדכון על המשחק השרת כן יחזיר תשובה.

הבקשות כולן הן בקשות מסוג .getכל לקוח שולח בבקשתו ארבעה פרמטרים שונים.

Action- הפעולה שאנחנו רוצים לבצע. אם מדובר בהרשמה, כניסת משתמש או פעולות הקשורות למשחק.

-Value שליחת הפרמטרים השונים הדרושים לכל בקשה. לדוגמה אם הפעולה היא הרשמות אז value יכיל את השם משתמש + הסיסמה. או אם הפעולה היא לשחק טור אז ה value יכיל את המשבצת שממנה הכלי זז למשבצת שלה הכלי יגיע. בחלק מהמקרים ה value ישאר רק כי אין מה לשלוח. כמו בבקשת עדכון המשחק.

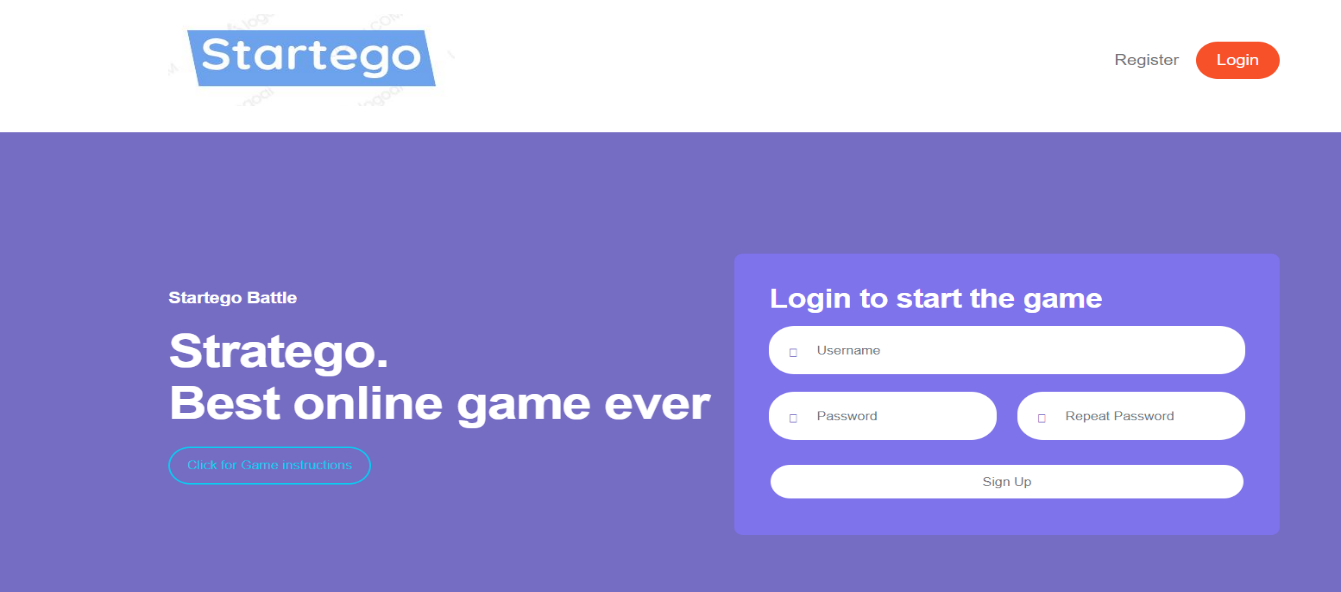
gameId- מזהה ייחודי שיבדיל בין המשחקים השונים. כך נדע לדעת כל שחקן לאיזה משחק הוא שייך. עד להתחלת המשחק gameId ישאר ריק.

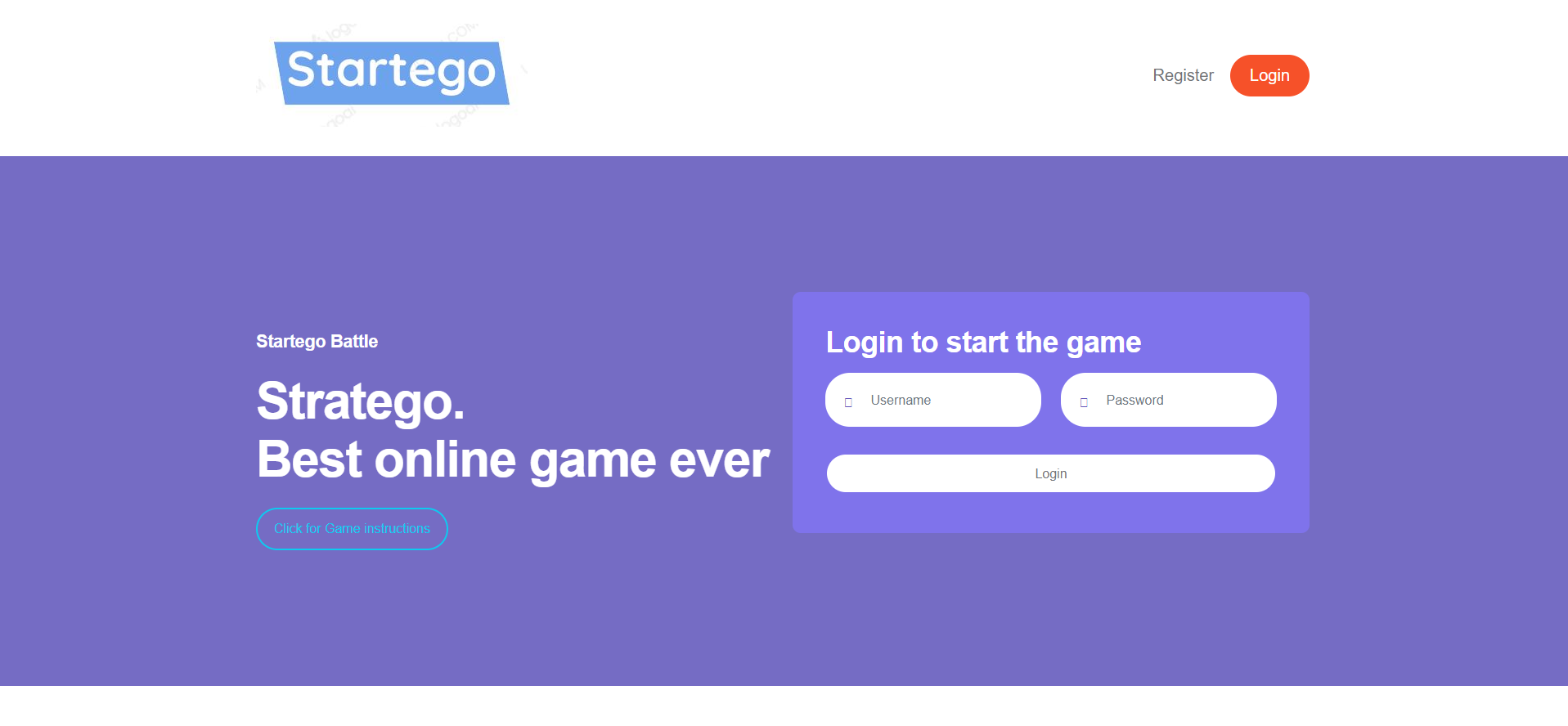
playerId- שם המשתמש של השחקן. כך נדע איזה שחקן התחבר ונדע להבדיל בין שני השחקנים של המשחק.

Response- לאחר עיבוד הבקשה השרת יחזיר תשובה בהתבסס על הפעולה ועל הפעולות שנעשו. בחלק מהמקרים response יכיל נכון או לא נכון, כמו בבקשת רישום המשתמש. הוא יכיל את מזהה המשחק היחודי במקרה שהתחיל משחק. או שבחלק מהמקרים הוא לא יכיל כלום כי אין מה להחזיר לשרת כמו בעשיית הצעד שאין מה להחזיר.

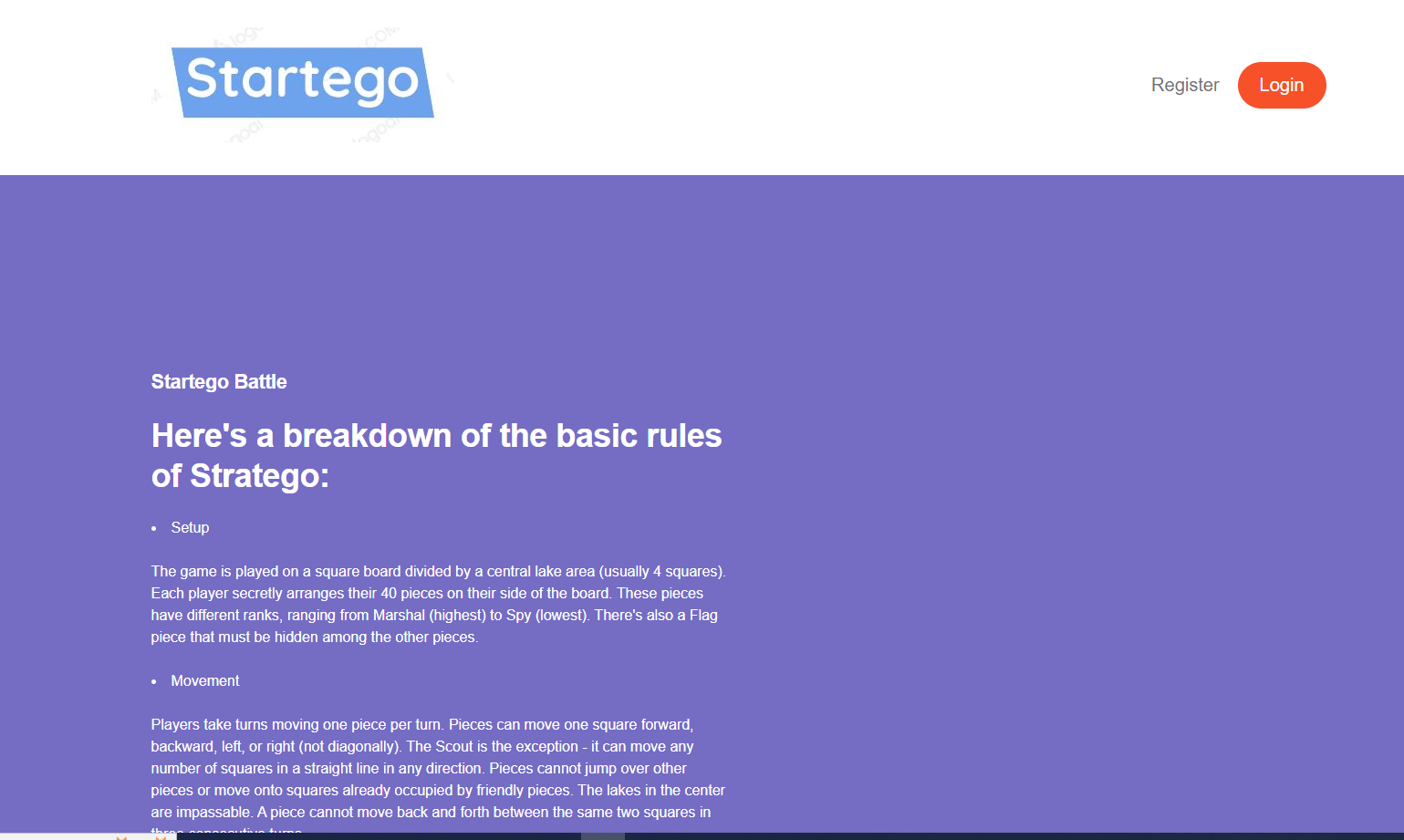
תיאור מסכי המערכת:

המסך הזה הוא מסך הכנסת המשתמש השחקן מכניס שם משתמש, סיסמה ואימות סיסמה. במידה וחוזרת הודעה חיובית הוא ממשיך למסך הבא ובמידה וחוזרת הודעה שלילית הוא מציג הודעת שגיאה ונשאר במסך.

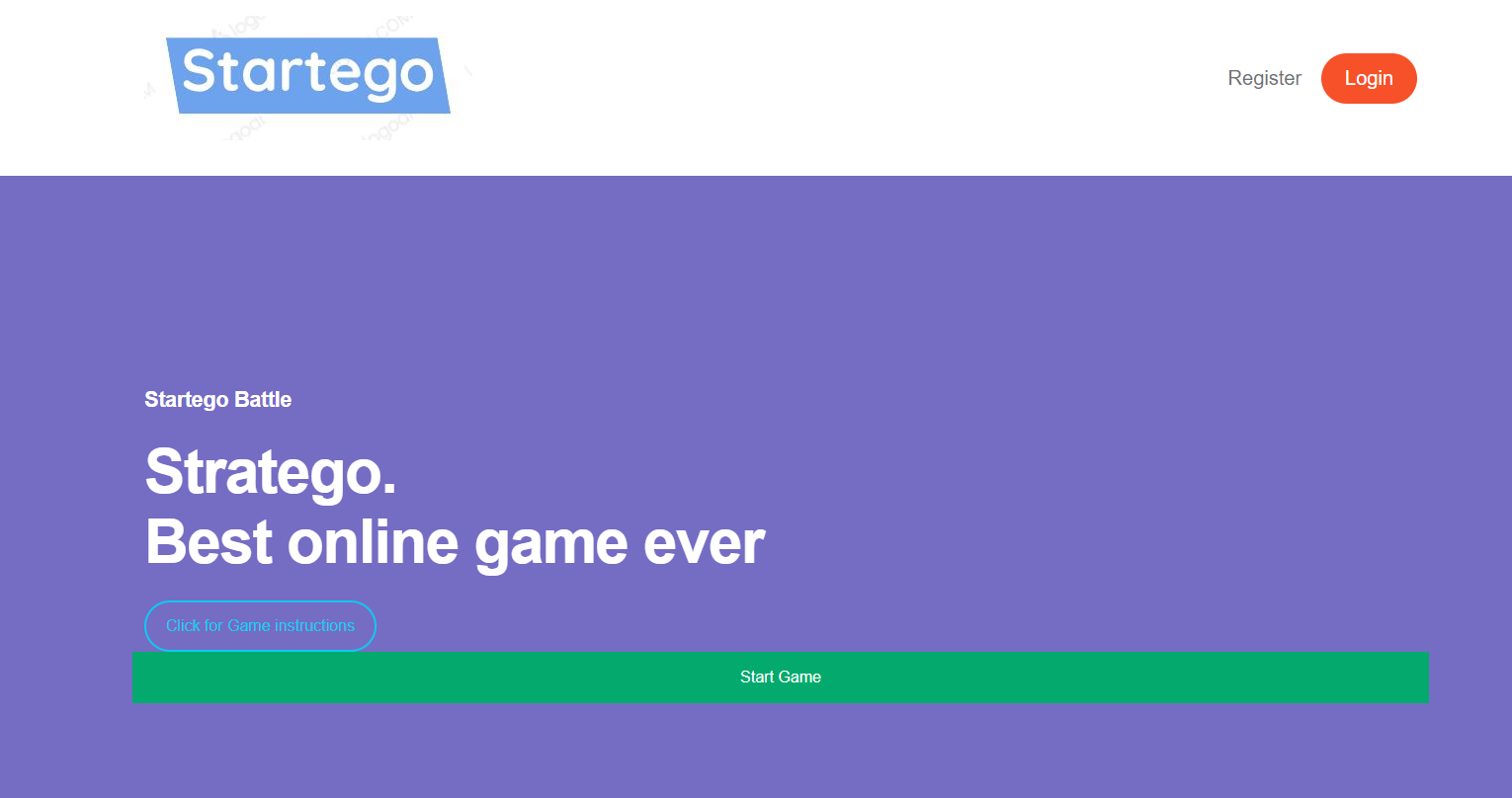


המסך הבא הוא מסך הוא מסך ההתחברות. השחקן מכניס שם משתמש וסיסמה שולח אותם לשרת. במידה וחוזר תשובה חיובית הוא הוא ממשיך לשחקן הבא ובמידה שחוזר תשובה שלילית הוא מציג הודעת שגיאה ונשאר במסך.

הוראות. השחקנים יכולים לעיין בהוראות המשחק.



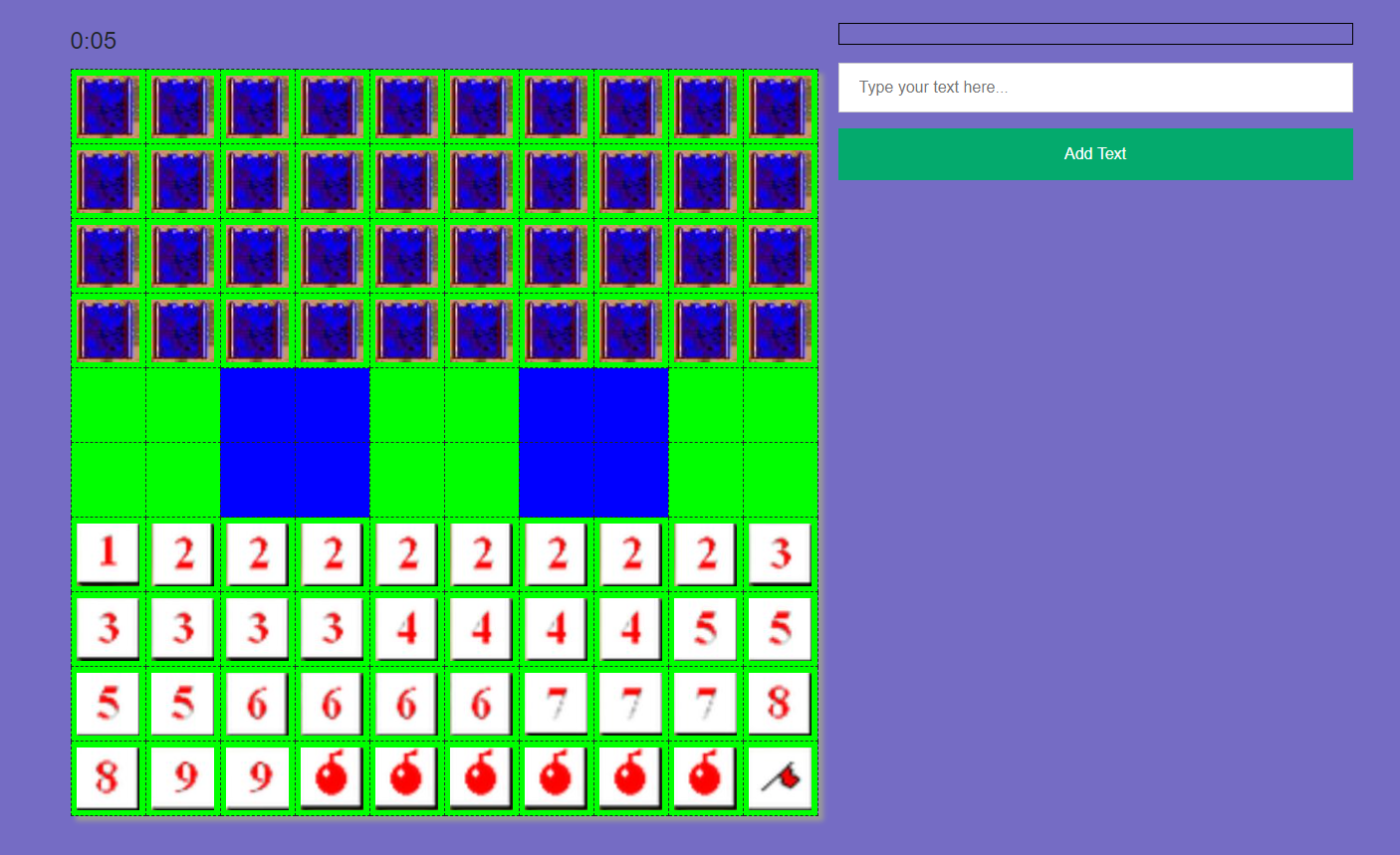
אחרי כניסת המשתמש השחקנים יצוץ כפתור שבו השחקנים יכלו ללחוץ עליו על מנת להתחיל משחק.



\* העיצוב נראה כך נכון להגשת תיק הפרויקט, עד הגשת המשחק הסופי אשפר את הנראות.

לאחר לחיצת הכפתור ועד מציאת יריב יופיע מסך טעינה.

המשחק נראה כך:



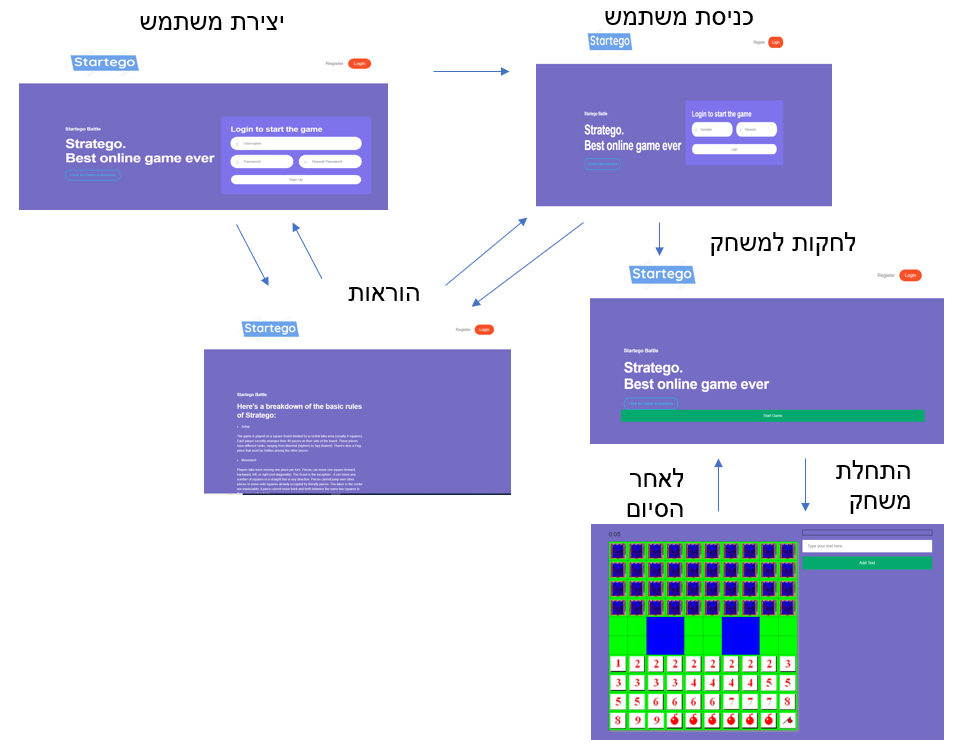
לשני היריבים יהיה חמש דקות התחלתיות לסדר את הכלים שלהם.

לאחר חמש הדקות כל אחד בתורו ישחק תור. כאשר כלי אחד תוקף כלי שני. הכלי בעל המספר הגדול יותר נשאר במשחק. הכלי שנשאר יהיה נראה לשני השחקנים למשך שלוש שניות.

לשני השחקנים צאט שבו הם יכולים לדבר אחד עם השני. הצאט נראה כך:

שם שחקן: הודעה.

\* עד הבגרות אני הוסיף את הניקוד שיראה לשני השחקנים. וזמן שונה לכל אחד מהשחקנים, כך שלאחר הסידור לשני השחקנים יהיה כמות זמן שבו הם יכולים לבצע את המהלכים שלהם.

תרשים מסכים לתיאור ההיררכיה:

מבנה הנתונים:

מסד הנתונים שלי הוא כולל שני טבלאות אחד של רישום משתמשים והשני הוא של סידור ראשוני של הלוח.

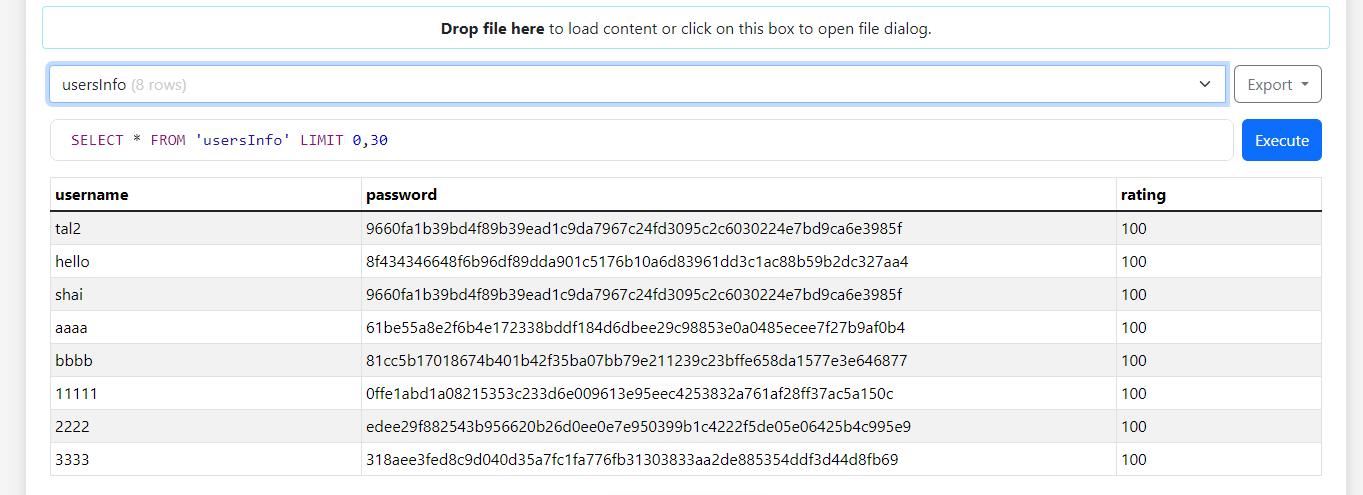
טבלת שמירת השחקנים. כוללות את השדות הבאים:

שם משתמש- שם המשתמש של כל שחקן. השם משתמש הוא יחודי וללא כפילויות.

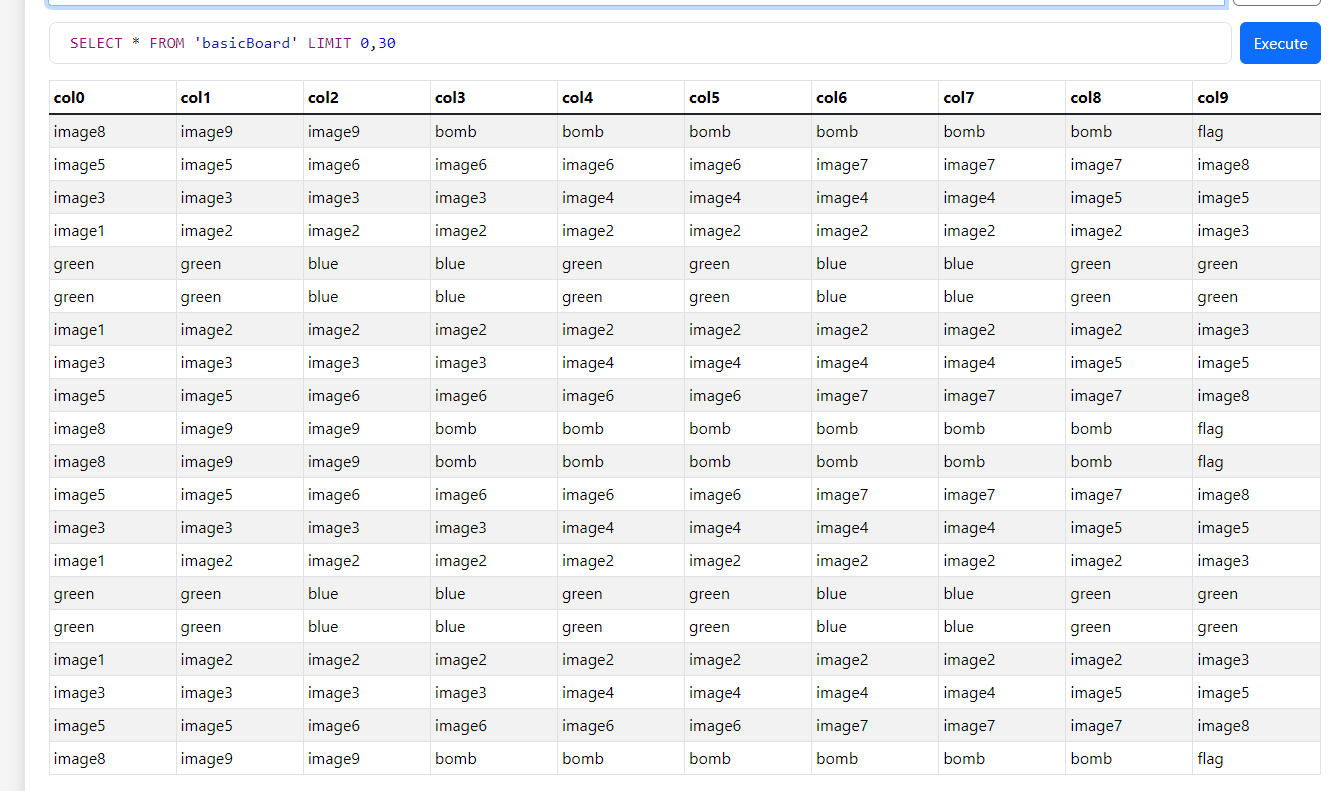
הסיסמה- הסיסמה של כל שחקן. יכולות להיות סיסמאות זהות בין שחקנים שונים.

על הסיסמה לפני השמירה במסד הנתונים נעשה הצפנת האש (hash) כך שאם יפרצו אל מסד הנתונים הפורץ לא ידע את הסיסמאות.

דירוג- דירוג אישי של כל שחקן. הדירוג ההתחלתי של כל שחקן הוא 100. על כל משחק השחקנים זוכים בנקודות לדירוג שלהם ובכל הפסד הם מאבדים נקודות.

הדירוג המינימלי של כל שחקן הוא 0. כלומר ברגע ששחקנים מגיעים ל0 נקודות הם לא מפסידים יותר.

הטבלה השנייה היא טבלת הסידור הראשוני של הכלים (לפני ההזזות של הכלים):



הטבלה מייצגת את מיקומים הכלים כך שנדע שבשורה 0 עמודה 0 יופיע image8 כלומר קלף מספר שמונה.

סקירת החולשות והאיומים:

1. הזרקת SQL (באנגלית: SQL Injection) היא שיטה לניצול פרצת אבטחה בתוכנית מחשב בעזרת פניה אל מסד הנתונים. על ידי הזרקת sql המשתמשים יכולים לקבל מידע שמור שאין ברצונו של המתכנת להשיג אותו. כך, למשל על ידי פנייה מסוימת הפורץ יוכל לנסות להשיג סיסמאות, מספרים של כרטיסי אשראי ועוד.

הקוד שלי לביצוע השאילתה הוא:

לכניסת המשתמש:

conn = sqlite3.connect("database")  
curr = conn.cursor()  
  
# create and use the query to check if user is in the database  
check\_query = '''SELECT COUNT(\*) FROM usersInfo WHERE username = ? AND password = ?;'''  
curr.execute(check\_query, (username, password))  
  
# Fetch the result of the query  
result = curr.fetchone()[0]  
  
# Close the connection  
curr.close()  
conn.close()

ליצירת משתמש:

# Connect to the database  
conn = sqlite3.connect("database")  
cursor = conn.cursor()  
  
# Check if username already exists  
check\_query = "SELECT COUNT(\*) FROM usersInfo WHERE username = ?;"  
cursor.execute(check\_query, (username,))  
result = cursor.fetchone()[0]  
  
if result == 1:  
 conn.close()  
 return "False" # Username already exists, return False  
  
# Insert the user into the database and set default rating for the user  
insert\_query = "INSERT INTO usersInfo (username, password, rating) VALUES (?, ?, 100);"  
cursor.execute(insert\_query, (username, password))  
  
# Commit the changes  
conn.commit()  
  
# Close the connection  
conn.close()  
  
return "True" # User successfully signed up, return True

אני קודם בודק ששם המשתמש איננו תפוס ואז מכניס אותו למסד הנתונים במידה ולא.

ההגנה בפני ההתקפה היא שאת הפרמטרים אנו מסמנים בסימן שאלה (?) ולא מעבירים את הפרמטרים ישר.

על ידי שליחת הפרמטרים לא בצורה ישירה אלה דרך (?), הספרייה דואגת להכניס את הערכים בצורה בטוחה מבלי לאפשר קוד זדוני להתבצע.

לכן: הכנסת ערכים באופן נפרד: ערכי הפרמטרים (username ו־password) מועברים לפונקציה execute בנפרד מהשאילתה עצמה, כך שהספרייה דואגת לטפל בערכים בצורה שמונעת הזרקת SQL.

2. man in the middle- מתקפת "איש באמצע" (Man in the Middle) היא מתקפת אבטחה בה תוקף יושב בין שני צדדים מתקשרים ומתחזה לכל אחד מהם. כך, התוקף יכול ליירט את התקשורת ביניהם, לגנוב מידע, לשנות אותו, או אפילו למנוע את התקשורת בכלל.

על מנת למנוע מתקפה זו אני מצפין את התעבורה לאורך כל הדרך. כשרק כל לקוח מתחבר הוא שולח בקשה לשרת לבקשת המפתח הציבורי שלו. השרת מחזיר לו אותו. הלקוח מכין מפתח סימטרי מצפין אותו בעזרת המפתח הציבורי ושולח לשרת. השרת מפענח בעזרת המפתח הפרטי ושומר את המפתח הסימטרי.

ההצפנה מונעת מהאיש באמצע לפענח את התעבורה אך הוא עדיין יכול לנתק את התקשורת בין השרת ללקוח או לשבש אותה על ידי שינוי ההודעות בצורה לא הגיונית.

3. מתקפת DDos- מתקפת DDoS היא ניסיון להשבית שירות מקוון על ידי הצפת השרת שלו בבקשות רבות עד כדי קריסת השירות. המתקפה מתבצעת באמצעות רשת של מחשבים נגועים, המכונים "בוטים", המנוהלים על ידי התוקף. הבוטים שולחים בקשות רבות בו זמנית לשרת, מה שגורם לו להיות עמוס יתר על המידה ולפסיק לפעול.

השרת שלי איננו חסין למתקפה הזו.

4. מתקפת Hash- מתקפת Hash היא ניסיון לפרוץ מערכת אבטחה על ידי מציאת שני קלטים שונים שיוצרים את אותו ערך Hash.

פונקציות Hash הן פונקציות מתמטיות שלוקחות קלט (כגון סיסמה, קובץ או הודעה) ומחזירות ערך Hash ייחודי (נקרא "Hash"). ערך ה-Hash הוא בעל אורך קבוע מראש. כל שינוי קטן בקלט יחזיר שינוי גדול בפלט.

מתקפת Hash מנצלת את העובדה שניתן למצוא שני קלטים שונים שיוצרים את אותו Hash. תוקף יכול ליצור קלט מזויף בעל אותו Hash כמו קלט לגיטימי, ואז להשתמש בקלט המזויף כדי להונות את מערכת האבטחה.

על מנת להתמודד עם התקיפה הזו אני משתמש בSHA-256 שהוא אלגוריתם הצפנה חזק מאוד. מה שמקטין מאוד את האפשרות ליצור התנגשות hash.

אך חשוב לציין שאין פונקציית Hash מושלמת: אף פונקציית Hash אינה חסינה לחלוטין בפני התקפות עתידיות. ייתכן שיתגלו חולשות חדשות בעתיד. ולכן הפרויקט לא חסין לגמרי מפני מתקפה זו.

התקפת "לחיצת יד משולשת" היא חולשה קריפטוגרפית המשפיעה על יישומים מסוימים של פרוטוקול (TLS (Security Layer Transport, המשמש לאבטחת תקשורת דרך האינטרנט. התקפה זו מאפשרת לתוקף ליצור חיבור זדוני עם שרת, מה שמוביל לגישה לא מורשית או לחשיפת מידע. בלחיצת יד טיפוסית של TLS, מתרחשות שתי לחיצות יד נפרדות - אחת בין הלקוח לשרת, ואחרת בין השרת ללקוח. מטרת לחיצת היד השנייה היא לאמת את זהות הלקוח וליצור חיבור מאובטח. בהתקפת לחיצת יד משולשת, התוקף מנצל את העובדה שחלקים מסוימים של TLS מאפשרים לשרת ליזום לחיצת יד שנייה לפני שהלקוח השלים את החלק שלו בלחיצת היד הראשונה. התוקף יכול לתמרן את לחיצות הידיים על ידי הזרקת הודעות והטעיית השרת והלקוח ליצור חיבור לא מורשה. התקפה זאת לא רלוונטית לפרויקט שלי כיוון שאני לא עושה שימוש בפרוטוקול ה-TLS. במקום זאת אני משתמש בהצפנה על ידי מפתחות.

מימוש הפרויקט

מודולים/ מחלקות מיובאות:

1. http.server- חבילת http.server בפייתון מספקת שרת HTTP בסיסי וקל לשימוש המיועד לפיתוח ובדיקות מקומיות. היא כוללת מחלקות ומודולים לניהול בקשות HTTP ויצירת תגובות בצורה מהירה וללא צורך בתצורה מורכבת.

2. חבילתsocketserver בפייתון מספקת מסגרת עבודה לבניית שרתים מבוססי סוקט, עם תמיכה בפרוטוקולים שונים כמו TCP ו-UDP. היא מאפשרת ליצור שרתים יעילים ומותאמים אישית על ידי שימוש במחלקות בסיסיות.

אני משתמש בשני החבילות האלה על מנת ליצור את השרת.

3. Random- היא מספקת את האפשרות ליצור מספרים רנדומליים בפייתון.

4. המודול String- המודול string מספק כלים לעבודה נוחה עם מחרוזות בפייתון.

אני משתמש בשני החבילות על מנת לייצר מזהה ייחודי לכל משחק בעל שש עשרה תווים.

5. threading - מודול ההליכונים מספק יכולת להריץ כמה תת תוכניות בפיתון.

הוא מאפשר ליצור ולנהל thread-ים, שהם יחידות ביצוע עצמאיות שיכולות לפעול

במקביל במסגרת תוכנית. דבר זה שימושי לטיפול במשימות במקביל ולשיפור

ביצועי התוכנית.

אני משתמש בזה כשאני צריך להריץ כמה דברים במקביל. לדוגמה אם אני רוצה לספור זמן אחורה ולהציג שעון. אני צריך לעשות זאת במקביל עם ריצת השרת. במקרה כזה לדוגמה אני משתמש בthreads.

6. 3sqlite - מודול 3sqlite מאפשר אינטראקציה עם מסדי נתונים של SQLite. הוא

מספק ממשק API נוח לחיבור למסדי נתונים, ביצוע שאילתות SQL ואחזור

תוצאות. SQLite הוא מנוע מסד נתונים מבוסס קבצים וחוצה פלטפורמות.

7. json- המודול json ב-Python משמש לעבודה עם פורמט JSON שהוא פורמט טקסטואלי קל משקל להחלפת נתונים. הוא נפוץ מאוד ביישומי אינטרנט וב-API.

אני משתמש בו על מנת להחזיר מידע מהשרת ללקוחות השונים.

8. המודול time- המודול time בפייתון מספק פונקציות לעבודה עם זמן ושעונים. הוא כולל פונקציות למדידת זמן, השהייה, וקבלת הזמן הנוכחי במגוון פורמטים.

אני משתמש בו כאשר אני צריך לחקות כמה שניות עד עשיית פעולה מסוימת. לדוגמה אם אני רוצה לחשוף קלף מסוים לזמן ספציפי אני משתמש במודול time על מנת לחקות את הזמן הרצוי.

9. CryptoJS.SHA256 – הוא חלק מתיקיית CryptoJS. הוא מאפשר להצפין בjavascript קלט מסוים בשיטת ההצפנה "SHA256"

SHA-256 הוא קיצור שפירושו אלגוריתם לגיבוב (hash) מאובטח ב-256 ביט'. האלגוריתם הזה משמש להצפנה מאובטחת.

אני משתמש בו על מנת להצפין את הסיסמאות בצד הלקוח כך שאם מישהו מקשיב לתקשורת הוא לא יכול לשים את ידו על הסיסמה של המשתמש.

10. מודול Pycryptodome הוא מודול פופולרי ב-Python לעיבוד מידע בצורה מאובטחת. זהו אחד מהפתרונות המובילים לצורך הצפנה, פענוח, חתימה דיגיטלית והעתקת מחלקות לקידוד סימטרי ואסימטרי, ייצוב וירידה נמוכה.

11. מודול 64base מספק פונקציות לקידוד ופענוח נתונים בינאריים

באמצעות קידוד 64Base 64.Base משמש בדרך כלל לייצוג נתונים בינאריים

כטקסט, במיוחד בעת העברת נתונים על פרוטוקולים מבוססי טקסט.

מחלקות שאני פיתחתי

שם המחלקה: card.

תפקיד: המחלקה שמייצגת קלף במשחק.

תכונות המחלקה:

1. Kind- איזה קלף הוא. יכול להכיל קלפים רגילים שמיוצגים על ידי המספרים מאחד עד תשע. או יכול לייצג קלפים מיוחדים כדוגמת פצצה או דגל. הקלפים יכולים להישאר ריקים (כלומר להכיל ירוק או כחול) שזה בעצם הלוח עצמו.
2. Player- הקלף שייך לאיזה שחקן. זאת על מנת להבדיל בין הקלפים שמיוצגים על ידי כל אחד מהשחקנים או קלפים שלא שייכים לאף אחד במקרה שהקלף מכיל ירוק או כחול (kind).
3. Show- מכיל נכון או לא נכון. רוב הזמן show יהיה לא נכון אך במקרה שבו קלף מסוים תוקף קלף של השחקן היריב הקלף צריך להיראות לשני היריבים. אז נהפוך את show לנכון.

פעולות המחלקה:

1. פעולות get ו set- לכל אחד מהמשתנים של המחלקה. זאת על מנת או לשנות כל אחד מהתכונות או לקבל אותם.
2. פעולת -Copy פעולה שמטרתה להעתיק קלף אחד לקלף שני.

טענת כניסה: קלף אחר. היא משנה את כל אחד מהפרמטרים לפרמטרים של הקלף השני.

מה מוחזר: הפונקציה לא מחזירה שום דבר.

1. פעולה shouldShow-מטרתה להחליט האם הקלף "יראה" על ידי השחקן או לא.

טענת כניסה: שם השחקן.

מה מוחזר: נכון או לא נכון בהתבסס על כמה גורמים. נכון הכוונה בלהראות את הkind של הקלף ולא נכון הכוונה בלהראות קלף של היריב.

שם המחלקה: Board.

תפקיד: לייצג את הלוח של כל משחק.

תכונת המחלקה.

1. Board- מערך דו ממדי שכל תא בו הוא מסוג Card.
2. playerUp- השחקן שכליו נמצאים בהתחלה בשורות 6-9. כשנרצה להציג לו את הלוח "נהפוך" את הלוח.
3. playerDown- השחקן שכליו שנמצאים בשורות 0-4 כאשר נרצה להציג לו את הלוח נציג אותו כמו שהוא.

פעולות המחלקה:

1. get לכל אחד מהתכונות.

טענת כניסה: ריקה.

מה מוחזר: המשתנה שברשותנו להחזיר כך לדוגמה בgetBoard יוחזר הלוח.

1. getPiece- קבלת הקלף במקום ספציפי בלוח.

טענת כניסה: שורה ועמודה.

מה מוחזר: משתנה מסוג card.

1. movePiece: מעביר קלף ממקום אחד בלוח למקום שני. הוא בודק האם בכלל המהלך הוא חוקי. לאחר מכן הוא בודק האם ישנו קלף במקום שאליו אנו מזיזים. אם כן הוא בודק על פי החוקים איזה קלף מנצח ומשנה בהתאם את הלוח והופך את הקלף לנראה לכולם.
2. טענת כניסה: שורה ועמודה של לפני, שורה ועמודה של אחרי. שמסמלים מאיפה הצד נעשה ולאן הצעד נעשה.

מה מוחזר: נכון או לא נכון. במידה והקלף שמסמל את הלאן, הוא דגל המשחק נגמר והחזר נכון, במידה ולא החזר לא נכון.

1. getBoardAsArray- מחזיר את הלוח כמערך דו ממדי.

טענת כניסה: שם שחקן. כדאי שנדע להחזיר את הלוח מאיזה פרספקטיבה. ככה שהלוח יראה כך שכליו של כל שחקן יהיה למטה. אנו מחזירים את השורות הפוכות.

מה מוחזר: מערך דו ממדי בהתאם לשחקן. בכל מקום יהיה הסוג של הקלף ולא האובייקט של הקלף. הכלים של היריב שלא נראים נכון לעכשיו יוחזרו כ"enemy" וכליו של השחקן יוחזרו לדוגמה כ"bomb".

1. determineCellContent- בסידור ההתחלתי צריך להחליט בכל מקום בלוח איזה קלף התחלתי יהיה (לפני השינויים) הפונקציה בודקת ומחזירה.

טענת כניסה: שורה ועמודה של המקום שברצוננו לדעת.

מה מוחזר: string שמשמעותו הוא סוג הקלף כדוגמת "green" או "bomb".

1. switchCards- החלפה בין שני קלפים. נעשה בשלב התכנון.

טענת כניסה: שורה ועמודה של קלף אחד, שורה ועמודה של קלף שני.

מה מוחזר: אין- הפונקציה לא מחזירה דבר.

1. getCardOwner- מחזיר את ה"בעלים" של הקלף.

טענת כניסה: שורה ועמודה של קלף.

מה מוחזר: string של שם השחקן שהקלף שייך לו.

שם המחלקה: showCardThread.

תפקיד: להראות קלף לשלוש שניות.

תכונות המחלקה: משתנה מסוג .card

פעולות המחלקה:

1. run- מציגה את הקלף לשלוש שניות.

טענת כניסה: אין.

מה מוחזר: אין.

שם המחלקה: game.

תפקיד: לנהל משחק בין שני שחקנים.

תכונות המחלקה:

1. playerUp- שחקן מספר אחד שכליו נמצאים בהתחלה בשורות 6-9.
2. playerDown- שחקן מספר שתיים שכליו שנמצאים בשורות 0-4.
3. Board- משתנה מסוג board שמדמה את לוח המשחק.
4. Turn- משתנה שמצביע על אחד משמות השחקנים. הוא נוצר על מנת שנדע תור של לשחק.
5. Chat- רצף המשפטים שהשחקנים שלחו (הצאט שלהם).
6. Status- הסטטוס של המשחק. האם הוא עדיין קורה, האם הוא נגמר ואם כן הוא מחזיק את שם המנצח.
7. timeThread משתנה של thread שמטרתו להחזיק את הזמן שנותר לשחקנים לשנות את הכלים שלהם.

פעולות המחלקה:

1. get לשחקנים שמטרתם להחזיר את התכונות.

טענת כניסה: אין.

מה מוחזר: המשתנה שברשותנו להחזיר כך לדוגמה getPlayerDown יחזיר את השחקן.

1. Set של השחקנים שמטרתם לשנות משתנה מסוים.

טענת כניסה: ערך חדש.

מה מוחזר: אין.

1. getBoardAsArray- מחזיר את הלוח כמערך דו ממדי.

טענת כניסה: שם שחקן. כדאי שנדע להחזיר את הלוח מאיזה פרספקטיבה. ככה שהלוח יראה כך שכליו של כל שחקן יהיה למטה. אנו מחזירים את השורות הפוכות.

מה מוחזר: מערך דו ממדי בהתאם לשחקן. בכל מקום יהיה הסוג של הקלף ולא האובייקט של הקלף. הכלים של היריב שלא נראים נכון לעכשיו יוחזרו כ"enemy" וכליו של השחקן יוחזרו לדוגמה כ"bomb".

הפונקציה קוראת ל getBoardAsArrayשל המחלקה board.

1. playTurn – בודק האם תורו של השחקן לשחק. אם כן קורא ל movePiece של המחלקה board ואז מחליף את התור לשחק השני.

טענת כניסה: שורה ועמודה של מאיפה לזוז. ושורה ועמודה של לאן לזוז.

מה מוחזר: אין.

1. getTime- מחזיר את הזמן שיש לשחקנים לסדר את הכלים שלהם.

טענת כניסה: אין.

מה מוחזר: הזמן שנותר. הזמן מוחזר בצורה הבאה: mm:ss כלומר כמה דקות נקודתיים וכמה שניות לדוגמה עבור דקה וחצי יוחזר 01:30.

1. switchCards- מחליף בין שני קלפים. הפונקציה קוראת ל switchCards של המחלקה board.

טענת כניסה: שם השחקן.

מה מוחזר: אין.

1. sendText- מטרתו להוסיף טקסט לצאט של שני השחקנים.

טענת כניסה: איזה שחקן אמר את הדברים כך שנדע להציג את הטקסט בצורה הבאה: שם שחקן: אמר מה. עוד דבר ששולחים הוא את המשפט שהשחקן רוצה לשלוח.

מה מוחזר: אין.

1. getText- קבל את הטקסט.

טענת כניסה: אין.

טענת יציאה: חמשת המשפטים האחרונים שנשלחו.

שם מחלקה: showTimeThread-

מטרה לשנות כל שנייה את הזמן של השחקנים ולהחזיר אותו בצורה הנאותה.

תכונות המחלקה:

1. Time- כמות בשניות שיש למשתמשים לארגן את הכלים שלהם. כל שנייה המשתנה יורד באחד.

פעולות המחלקה:

1. getTime- הפעולה מחזירה את זמן המשתמשים.

טענת כניסה: אין.

מה מוחזר: הזמן שנותר. הזמן מוחזר בצורה הבאה: mm:ss כלומר כמה דקות נקודתיים וכמה שניות לדוגמה עבור דקה וחצי יוחזר 01:30.

שם מחלקה: Player

תכונות מחלקה:

1. שם משתמש- שם משתמש של השחקן הוא המזהה בין בשחקנים השונים.
2. מפתח סימטרי- לכל שחקן ישנו מפתח סימטרי לתקשורת עם השרת.

פעולות: Set ו get.

תיאור אלגוריתמיים מרכזיים בפרויקט:

שיטת ההצפנה- נעזרתי במאמר הבא:

<https://www.lbscience.org/2018/08/01/%D7%A9%D7%99%D7%98%D7%AA-%D7%94%D7%94%D7%A6%D7%A4%D7%A0%D7%94-rsa/>

בשיטות הצפנה "קלאסיות", המידע מוצפן ומפוענח בעזרת מידע סודי הידוע למצפין ולמפענח, ומכונה "מפתח": למשל, סיסמה משותפת או מספר גדול. הקושי בגישה זו הוא שעל המצפין והמפענח להסכים מראש על הסיסמה בצורה מאובטחת. שיטת הצפנה זו נקראת שיטת הצפנה סימטרית.

בשיטת המפתח הפומבי, השרת מכין מראש שני מפתחות: פומבי ופרטי. את הפומבי הוא מפרסם ואת הפרטי שומרת בסוד. כדי להצפין, יש לדעת רק את המפתח הפומבי, אבל כדי לפענח אפשר להשתמש רק במפתח הפרטי - אפילו המצפין לא יכול לפענח את ההודעה שהצפין! שיטה זו נקראת שיטת הצפנה אסימטרית.

הסיבה שאנחנו לא משתמשים בהצפנה האסימטרית תמיד היא שהיא איטית משמעותית מההצפנה הסימטרית. הסיבה לכך היא שהיא משתמשת בפעולות מתמטיות מורכבות כמו חישוב מכפלות מודולריות. כתוצאה מכך, שימוש בהצפנה אסימטרית לכל תקשורת עלול להוביל לירידה משמעותית בביצועים. לכן אנחנו משתמשים בהצפנה א סימטרית רק בשביל להצפין את ההצפנה הסימטרית.

קטעי קוד מיוחדים:

שליחת הבקשה לשרת: שמוצפנת. אנו שולחים מילון של הפעולה, המזהה היחודי של המשחק, המזהה היחודי של השחקן ואת מה שצריך לשלוח לכל פעולה.

function callServer(actionToDo, valueToPass, callback) {  
 // Building the dictionary  
 let data = {  
 action: actionToDo,  
 game\_Id: gameId,  
 value: valueToPass,  
 clientId: userName  
 };  
  
 // Convert the dictionary to a JSON string  
 let jsonString = JSON.stringify(data);  
  
 // Encrypt the JSON string  
 let encryptedData = encryptMsg(jsonString);  
  
 // Send the encrypted message to the server  
 $.get("http://127.0.0.1:8000", {  
 data: encryptedData  
 }, function (msg\_from\_server) {  
 callback(msg\_from\_server);  
 });  
}

קוד במחלקה game לשחק את התור.

def playTurn(self, value, row1, col1, row2, col2):  
 # Update the destination board with the piece  
 if str(self.turn) == str(self.board.playerDown):  
 if self.turn != self.board.getCardOwner(row1, col1):  
 return  
 if self.board.movePiece(row1, col1, row2, col2, self.turn):  
 # if game is over winner is the one who played right now  
 self.status = self.turn  
 else:  
 if self.turn != self.board.getCardOwner(9 - row1, col1):  
 return  
 if self.board.movePiece(9 - row1, col1, 9 - row2, col2, self.turn):  
 # if game is over winner is the one who played right now  
 self.status = self.turn  
  
 # Switch player turn  
 if self.turn == self.playerDown:  
 self.turn = self.playerUp  
 else:  
 self.turn = self.playerDown

ראשית הוא בודק איזה שחקן משחק עכשיו, האם זה השחקן למטה או למעלה. אם זה השחקן למעלה הוא הופך את הלוח. לאחר מכן הוא קורא לפעולה movePiece של המחלקה board. אם הוחזר true נגמר המשחק ויש לעדכן את הסטטוס. לאחר עשיית המהלך יש לשנות את תור השחקן.

קוד לעשיית התור במחלקה board.

def movePiece(self, fromRow, fromCol, toRow, toCol):  
 placeFrom = self.board[fromRow][fromCol]  
 placeTo = self.board[toRow][toCol]  
  
 if placeTo.getKind() == "flag":  
 # game is over  
 return True  
 elif placeTo.getKind() == "green":  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 self.board[fromRow][fromCol] = card("green")  
 return  
 # there is a fight between two cards  
 elif placeTo.getKind() == "bomb":  
 # incase of a bomb only card 3 survives  
 if placeFrom.getKind() == "image3":  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 else:  
 placeTo.copy(card("green"))  
 placeFrom.copy(card("green"))  
 # if there is an enemy card or a one attacking a nine  
 elif (placeFrom.getKind()[-1] > placeTo.getKind()[-1]) or (  
 placeFrom.getKind()[-1] == "1" and placeTo.getKind()[-1] == "9"):  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 elif placeFrom.getKind()[-1] == placeTo.getKind()[-1]:  
 placeTo.copy(card("green"))  
 self.board[fromRow][fromCol] = card("green")  
 # game is not over  
 showCard = showCardThread(self.board[toRow][toCol])  
 showCard.start()  
 return False

הקוד מזיז כלי ממקום למקום. הוא פועל על פי מספר חוקים. לדוגמא:

אם כלי אחד תוקף את הכלי החזק ביותר (תשע) האחד מנצח. אם תוקפים פצצה כל הקלפים "מתפוצצים" מלבד כלי 3.

בנוסף עם מדובר ב"מלחמה" בין שני קלפים יש להראות את הכלי המנצח לשני השחקנים. זאת נעשה על ידי יצירת הthread ששמו showCardThread.

הפעולה מחזירה נכון אם המשחק נגמר ולא נכון אם הוא לא נגמר.

הקוד הבא בודק את תקינות המהלך.

// check if we want to move the image to a new location which is legal  
function checkIfLocationOk(lastTdId, tdId) {  
 var lastTd = document.getElementById(lastTdId);  
 var td = document.getElementById(tdId);  
   
 // Calculate the absolute difference between rows and columns  
 var rowDiff = Math.abs(tdId[0] - lastTdId[0]);  
 var colDiff = Math.abs(tdId[1] - lastTdId[1]);  
   
 // If the difference in rows or columns is more than 1  
 if (rowDiff > 1 || colDiff > 1) {  
 // If the card is not a 2, return false  
 if (!lastTd.className.includes("2")) {  
 return false;  
 } else {  
 // If the card is a 2, it can jump more than one tile  
 // Check if it's a valid jump (only vertical or horizontal)  
 if (rowDiff > 0 && colDiff > 0) {  
 return false; // diagonal jump is not allowed  
 }  
   
 // Check if there are cards between initial and final positions  
 if (rowDiff === 0) { // Horizontal jump  
 var startCol = Math.min(lastTdId[1], tdId[1]);  
 var endCol = Math.max(lastTdId[1], tdId[1]);  
 for (var col = startCol + 1; col < endCol; col++) {  
 var card = document.getElementById(lastTdId[0] + col);  
 if (!card.className.includes("green")) {  
 return false; // Invalid jump over non-green or non-blue cards  
 }  
 }  
 } else if (colDiff === 0) { // Vertical jump  
 var startRow = Math.min(lastTdId[0], tdId[0]);  
 var endRow = Math.max(lastTdId[0], tdId[0]);  
 for (var row = startRow + 1; row < endRow; row++) {  
 var card = document.getElementById(row + lastTdId[1]);  
 if (!card.className.includes("green")) {  
 return false; // Invalid jump over non-green or non-blue cards  
 }  
 }  
 }  
 }  
 }  
 return true;  
}

הקטע בודק את תקינות הצעד. כל קלף שאני 2 יכול ללכת אך ורק משבצת אחת לכל כיוון (כולל אלכסון).

הקלף 2 הוא קלף מיוחד. הוא יכול לזוז או באלכסון צעד אחד או כמה צעדים שהקלף רוצה במאוזן או במאונך. מלבד שהוא עובר מעל קלף אחר או מעל ים.

הקטע קוד הבא שולח לשרת את הצעד כל עוד הכל תקין.

function moveImage(lastTdId, tdId, gameStarted) {  
 var td = document.getElementById(tdId);  
 var lastTd = document.getElementById(lastTdId);  
  
 // if game is not started   
 if (!gameStarted&&tdId[0] > 5) {  
 // Send the move to the server  
 callServer("switchCards", lastTdId.toString() + tdId.toString(), function(response) {  
 updateBoard();  
 });  
 return;  
 }  
  
 // if game has started and the cell contains an image  
 if (gameStarted && lastTd.classList[1].includes("image")&&!td.classList[1].includes("image")) {  
 // Check if the location of the new image is legal  
 if (!checkIfLocationOk(lastTdId, tdId))  
 return;  
  
 // Send the move to the server  
 callServer("playTurn", lastTdId.toString() + tdId.toString(), function(response) {  
 updateBoard();  
 });  
 }  
}

שלב הבדיקות:

1. מטרה: לבדוק את מסד הנתונים.

בוצע בפועל: הדפסתי עבור כל משתמש שנכנס את שם המשתמש, הסיסמה והניקוד שלו. והשוויתי עם תוכן מסד הנתונים דרך תוכנה שמציגה מבני נתונים מסוג SQLite.

תוצאות: ראיתי כי הפרטים אכן נשמרים בצורה נכונה ומוצפנת.

אם התגלו בעיות, איך נפתרו: לא התגלו בעיות.

1. מטרה: לבדוק מה קורה כשאני מכניס שם משתמש וסיסמה בניגוד לחוקים. כלומר שם משתמש ארוך מדי, קצר מדי, או שהוא מכיל רווח.

בוצע בפועל: בדף יצירת המשתמש כתבתי נתונים שאינם חוקיים, הסתכלתי שהם לא הוספו למסד הנתונים, ושמוצגים ההודעות המתאימות.

תוצאות: ההודעות המתאימות הוצגו והמשתמשים לא הוספו.

אם התגלו בעיות, איך נפתרו: לא התגלו בעיות.

1. מטרה: לבדוק כשאני יוצר משתמש שאם הסיסמה והסיסמה החוזרת לא נכונה מוצגת ההודעה המתאימה

בוצע בפועל: בדף יצירת המשתמש כתבתי סיסמה וסיסמה חוזרת שונים אחד מהשנייה.

תוצאות: הוצג שהסיסמאות אינן זהות

אם התגלו בעיות, איך נפתרו: לא התגלו בעיות.

1. מטרה: לבדוק שהשרת מטפל במספר משתמשים בו זמנית.

בוצע בפועל: הרצתי ארבע לקוחות שרוצים לשחק בו זמנית וניסיתי להריץ שני משחקים בו זמנית.

תוצאות: המשחקים אינם עבדו בו זמנית. גיליתי שמשחק אחד כל פעם עובד אך כמנסים יותר מאחד השרת קורס.

אם התגלו בעיות, איך נפתרו: הוספתי מילון חדש שבו המפתח הוא מזהה ייחודי של המשחק והערך היה אובייקט של משחק. ככה שכל שחקן שמשחק נוכל להבדיל בין המשחקים השונים.

1. מטרה: לבדוק את שהתעבורה אכן מוצפנת.

בוצע בפועל: הרצתי מספר פעמים את השרת וחיברתי אליו את הלקוח. תוך כדי

שעשיתי זאת הרצתי את אפליקציית ניתוח התעבורה ברשת Wireshark, על מנת להאזין לפקטות המידע.

תוצאות: ראיתי כי כל המידע מועבר בצורה מוצפנת.

אם התגלו בעיות, איך נפתרו: לא התגלו בעיות.

1. מטרה: לבדוק שכל קלף הולך בצורה הנכונה.

בוצע בפועל: התחלתי משחק בין שני לקוחות והתחלתי להזיז כלים שונים.

תוצאות: רוב הכלים הלכו בצורה טובה. דגלים ופצצות שלא אמורים לזוז לא זזו. היה רק בעיה עם קלף שתיים שהולך באופן מיוחד.

אם התגלו בעיות, איך נפתרו: גיליתי שקלף שתיים אמנם הולך בצורה הנכונה, אך הוא יכול לדלג מעל הים מה שלא אמור להיות חוקי. שיניתי את הפונקציה הבודקת האם המהלך חוקי בשביל לפתור את הבעיה.

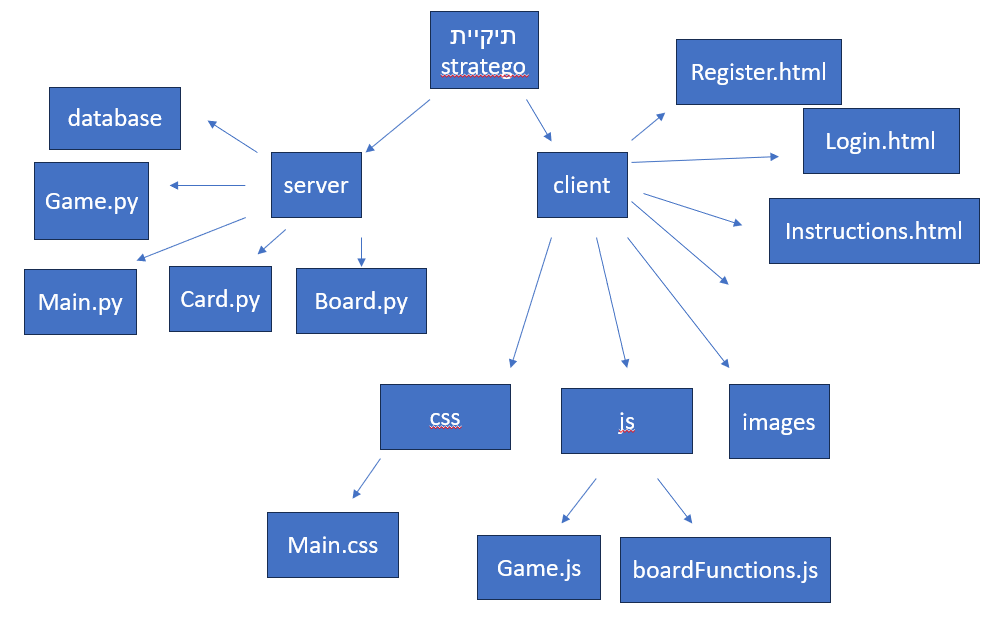
1. מטרה: לבדוק את הצאט שמציג את ההודעות בזמן אמת לשני השחקנים.

בוצע בפועל: התחלתי משחק וכתבתי הודעה משני השחקנים השונים.

תוצאות: ההודעות התעדכנו בשני השחקנים

אם התגלו בעיות, איך נפתרו: לא התגלו.

מדריך למשתמש:

פירוט כללי קבצי מערכת:

הרחבה על כל קובץ:

תיקיית stratego- התיקייה שמאחדת את כל הפרויקט שלי.

תיקיית server- תיקייה שמאחדת את כל קבצי השרת שלי.

תיקיית client- תיקייה שמאחדת את כל קבצי הלקוחות שלי.

תיקיית css בתוך client- מאחדת את הקבצים מסוג css.

תיקיית js בתוך client- מאחדת את הקבצים מסוג js.

תיקיית images בתוך client- מאחדת את כל התמונות שלי. נכלל בתוכו תמונות הכלים שלי והלוגו של המשחק.

בתוך server- ישנם 5 קבצים.

* Game,card,board אלא הקבצים שנמצאים המחלקות שלי.
* Main.py פה אני מריץ את השרת.
* Database- זה קובץ מסד הנתונים בו נשמרים הסיסמאות, שמות המשתמשים והניקוד של כל שחקן.

בתוך client נמצאים גם קבצי הhtml שלי.

* Login.html- מסך ההרשמות.
* Register.html- מסך יצירת המשתמש.
* Instructions.html- הוראות המשחק.

בתוך js- נמצאים הקבצים מסוג js.

* boardFunctions.js- פה נשמרים כל הפונקציות הקשורות ללוח כמו updateBoard שאחראי על לעדכן את הלוח createGame שאחראי על ליצור את הלוח ההתחלתי.
* Game.js- אחראי על הפונקציות שקשורות למשחק עצמו ולא ללוח. כמו checkStartGame שמטרתו לחקות עד תחילת המשחק או main שנקרא לאחר שנרשמו למערכת.

התקנת המערכת:

* Python: השרת כתוב ב - Python, כך שהמשתמש צריך להתקין את שפת התכנות Python, בגרסה עדכנית. נדרש שיהיה מותקן python בגרסת לפחות 3.8 מאחר ויש modules שלא פועלים בגרסאות אשר נמוכות מגרסה זו.
* על מנת להריץ את המערכת נדרש ראשית התוכנה pycharm – תוכנת עריכת קוד python – גרסת התוכנה לא משפיעה.
* דפדפן- על מנת להריץ את הלקוח. המשתמשים צריכים שיהיה להם דפדפן כמו chrome או Firefox.
* חיבור לרשת- על מנת לשחק עם מחשבים אחרים יש צורך לדבר עם השרת. על מנת לעשות זאת המחשב צריך להיות מחובר לרשת.
* יש להוריד חלק מהמודולים בפייתון שאני משתמש בהם בשרת כדוגמת Crypto חבילה זו מספקת פונקציונליות קריפטוגרפית בפיתון, כולל

הצפנה ופענוח. אפשר להתקין אותה באמצעות pip.

* קבצי התוכנית- על המשתמשים להוריד את תיקיית הclient שמכילה כל מה שהמשתמש צריך. בנוסף יש צורך בתיקיית הserver על מנת להריץ את השרת. אך יש צורך רק בשרת אחד ולכן התיקייה לא צריכה להיות מותקנת על כל המחשבים.

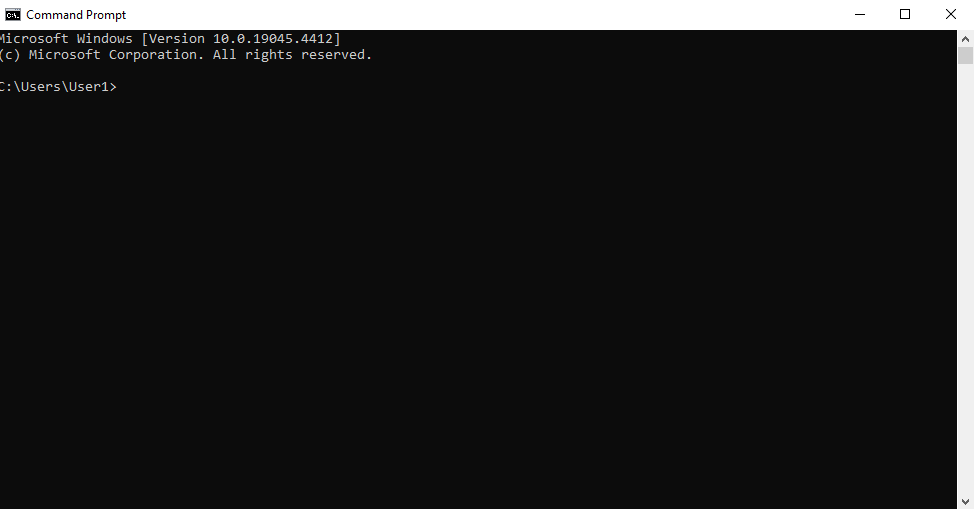
אופן הפעלת המערכת:

למפעיל השרת- יש לוודא כי שפת התכנות וה-interpreter של python מותקנים על מערכת ההפעלה. לאחר מכן יש להוריד את כל המודולים הנדרשים לביצוע התוכנית. ולבסוף יש להוריד את תיקיית הserver ולהריץ את main.py

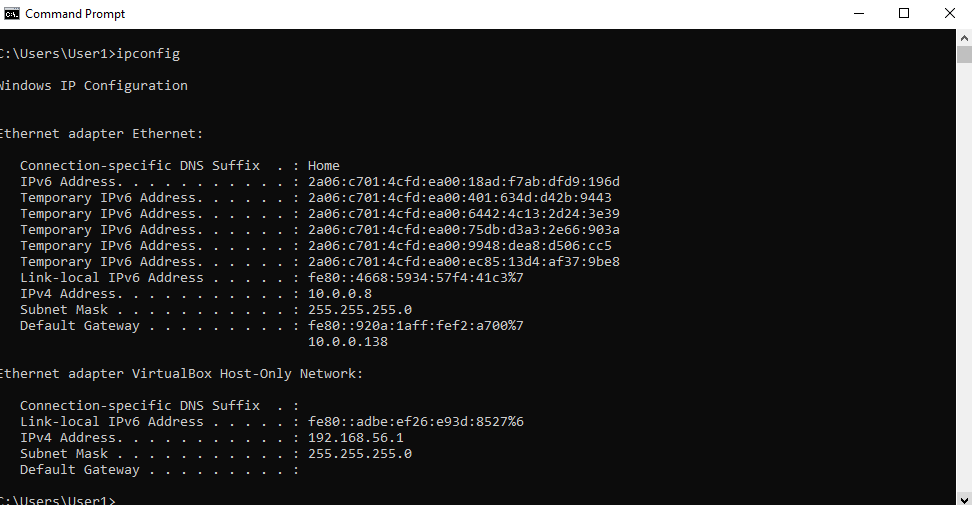
לאחר שנבדק חיבור לרשת המקומית יש לעקוב אחר השלבים הבאים:

1. למצוא את הip הפנימי שבו מורץ השרת. אפשר לעשות זאת על ידי פתיחת הcmd וכתיבת הפעולה ipconfig.
2. לאחר מציאת הip הפנימי של השרת יש לשנות את המשתנה serverIp למה שמצאנו.
3. כל מה שצריך לעשות בשביל להתחיל להריץ את התוכנית הוא לפתוח את איך מקבצי הhtml. מכל קובץ html אפשר להגיע לכל קובץ אחר אז לא משנה הבחירה.
4. אם ברצוננו להתחיל משחק יש להכניס שם משתמש וסיסמה (במידה וקיים) במידה ולא קיים יש ליצור משתמש ואז להיכנס. לאחר מכן יש ללחוץ על כפתור החל משחק.

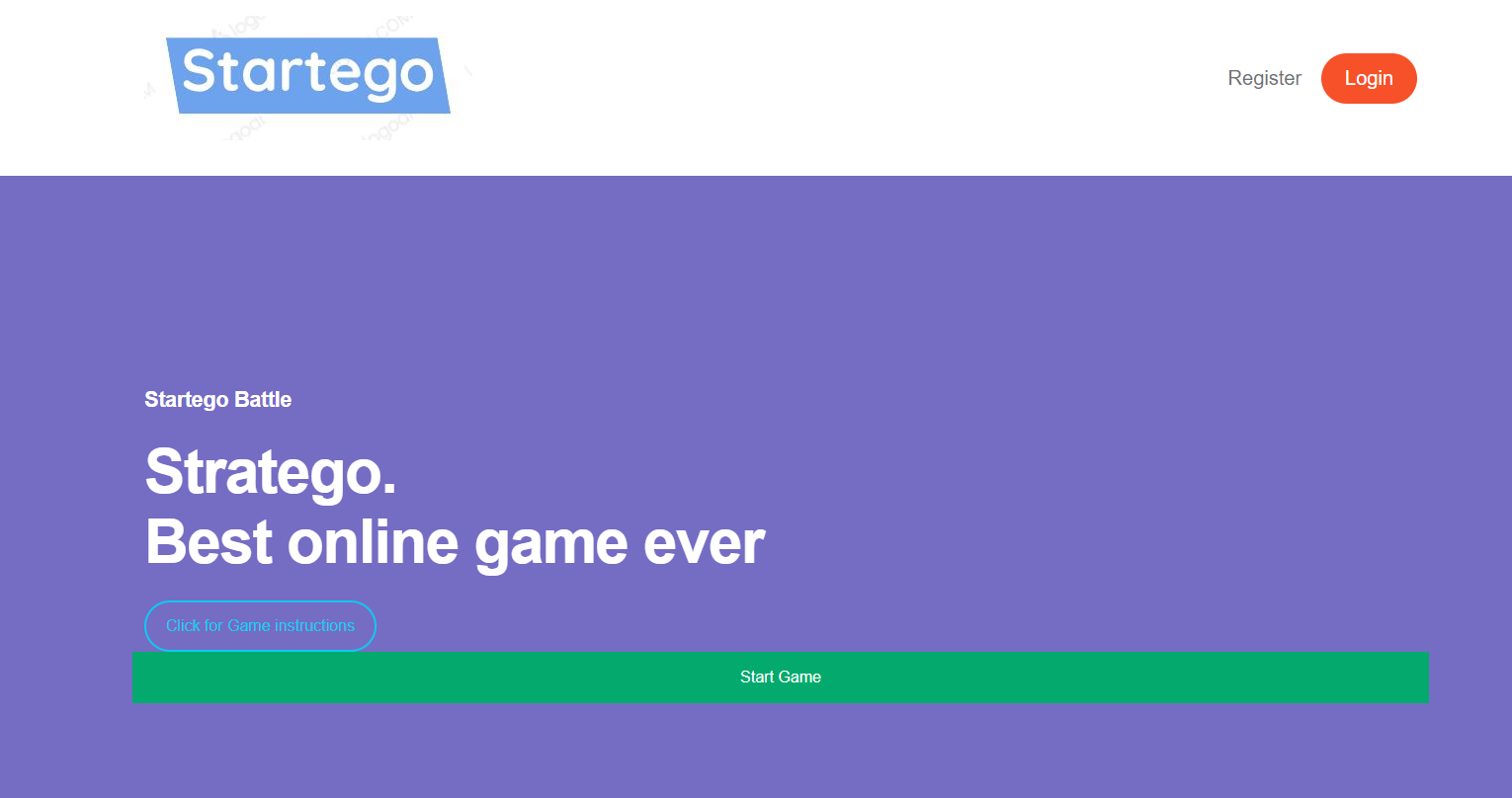
ראשית יש להיכנס לcmd במחשב השרת.

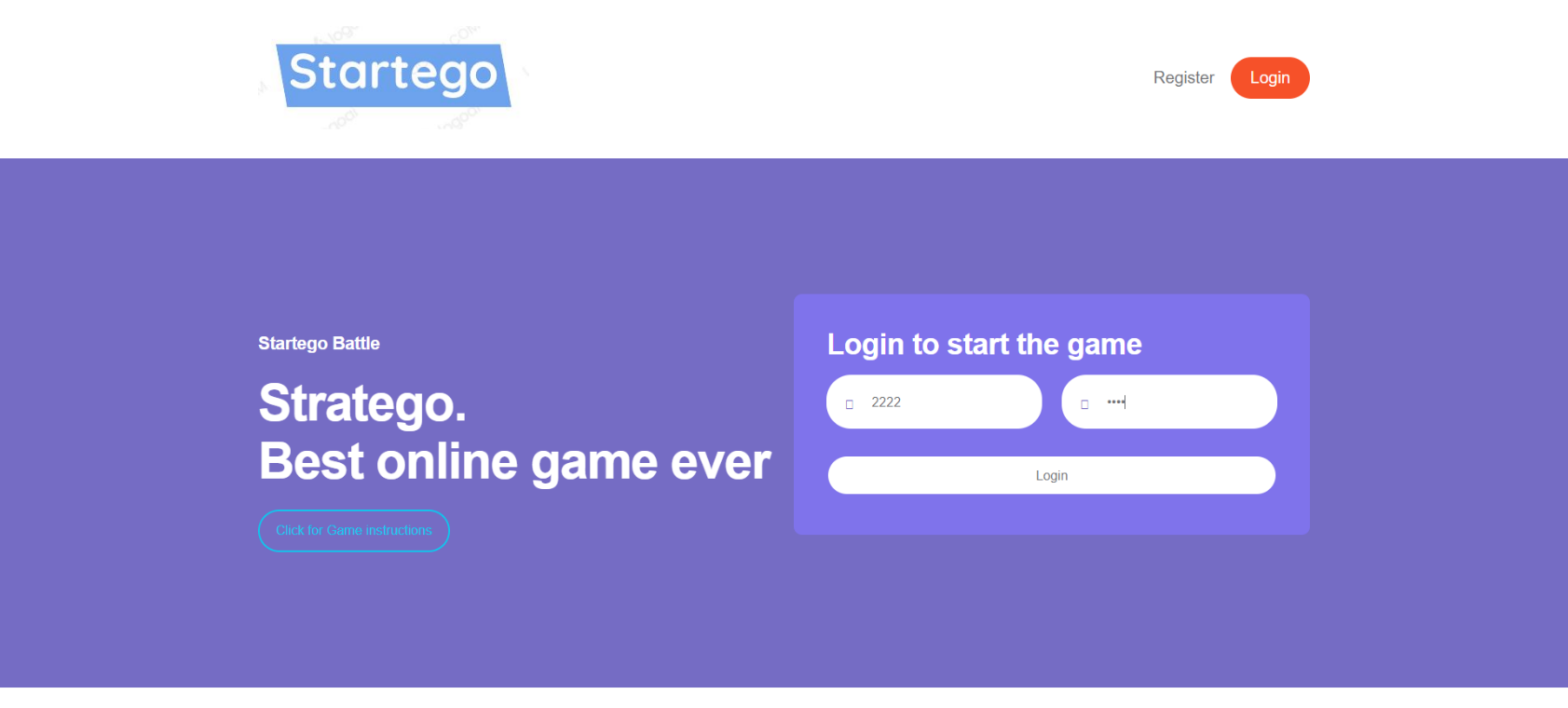


לאחר מכן יש לרשום את הפקודה ipconfig ולקבל תשובה.

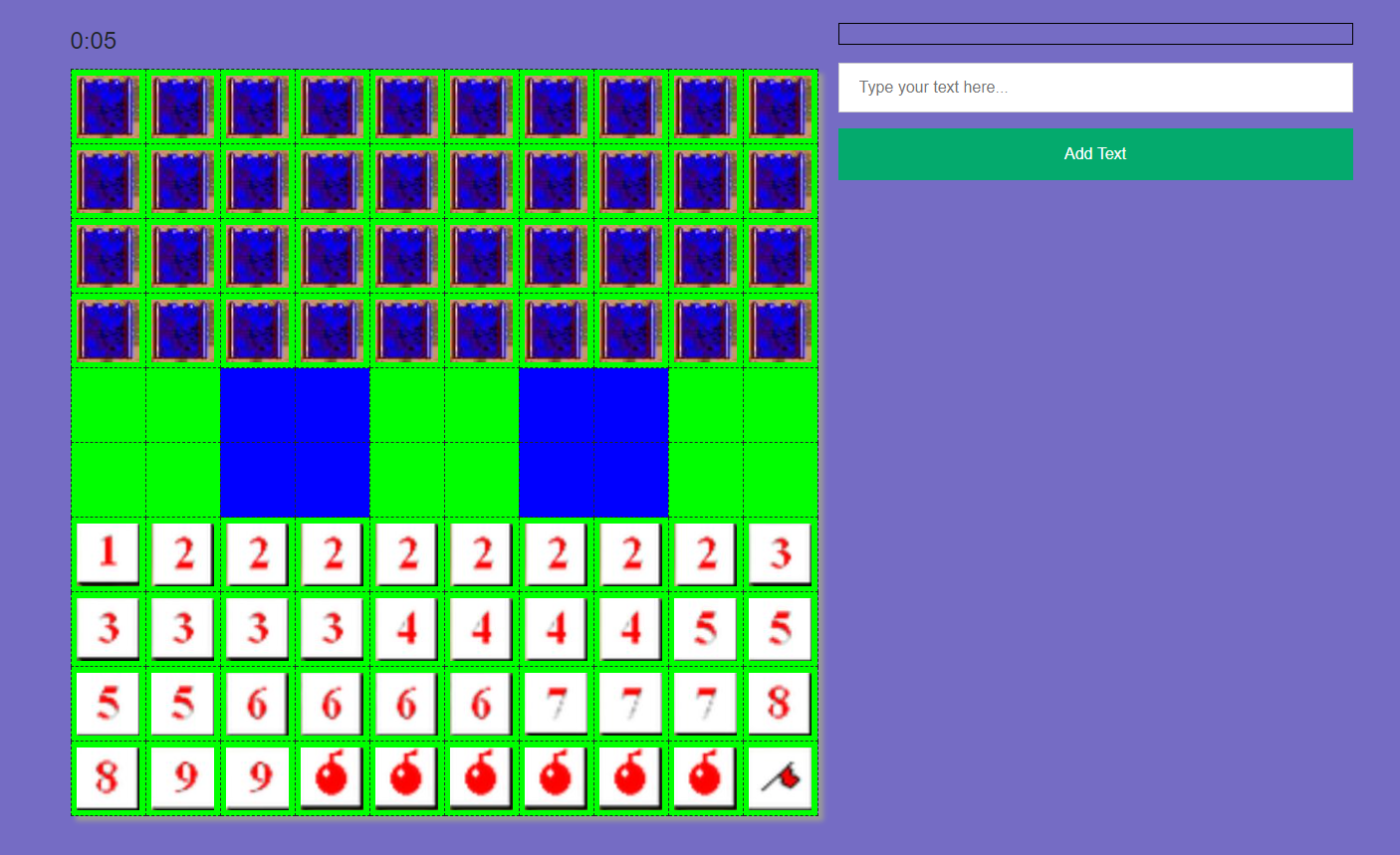
יש לבחור את הipv4 Address שהיא הip הפנימי בתוך הרשת המקומית. במקרה שלנו מדובר ב 10.0.0.8

לאחר מכן יש להכניס את הip לתוך המשתנה serverIp שנמצא בקובץ game.js לשמור ולצאת.

לאחר מכן יש לפתוח את אחד קבצי הhtml ואם ברצוננו להתחיל משחק יש להיכנס עם שם משתמש וסיסמה.

ללחוץ על התחל משחק.

ולחכות ששחקן נוסף יצטרף. אפשר גם להריץ שני לקוחות שונים משני משתמשים שונים בשביל לשחק נגד עצמנו (לא שיש בזה הגיון אלא רק בשביל לראות את הפרוייקט).



כך נראה המשחק לאחר ההתחלה.

רפלקציה

כבר מכיתה יא התחלתי להעלות רעיונות על מה אני ארצה לעשות את הפרויקט שלי בשנה הבאה. העלתי הרבה רעיונות ובתחילת כיתה יב גם התחלתי לחקור עליהם ולהתחיל לכתוב משהו קטן שיהיה אב טיפוס ממש ראשוני לרעיון.

הרעיון לכתוב את הפרויקט שלי על המשחק לא עלה בדעתי עד שחבר סיפר לי שהוא מתכנן לבנות שש בש. הרעיון להפוך משחק קופסה עניין אותי. ישר נזכרתי בסטרטגו ובשעות שהייתי מעביר בלשחק עם חברים והחלטתי שזה יהיה הפרויקט שלי.

העבודה כללה בתוכה אתגרים רבים. קודם כל מדובר על הפרויקט התכנותי הגדול ביותר שעשיתי בהפרש גדול. זו גם הפעם הראשונה שעשיתי פרויקט אמיתי שכלל שרת ולקוח (מה שכן עשיתי כמה עבודה לאורך השנה אבל לאף פעם משהו בסדר גודל הזה). הייתי צריך ללמוד הרבה נושאים שלא ידעתי כמו שימוש במסד נתונים, שימוש threads ורשתות.

למדתי המון מהפרויקט, אם מדובר על חומר כמו הצפנות או רשתות אבל לא רק. למדתי שימוש בgit, איך להיעזר באינטרנט ובמגוון אתרים על מנת ללמוד נושא חדש או חיפוש פתרון לבעיה. בנוסף, אני בן אדם מאוד מבולגן, והתוכניות שכתבתי לפני לרוב היו בקובץ אחד ארוך. מכיוון גודל הפרויקט הייתי צריך להשקיע לא רק בקוד ובתיק פרויקט אלה גם בלסדר את הקוד, ולחלק אותו לקבצים באופן הגיוני.

במידה והיה לי יותר זמן לעבוד על הפרויקט הייתי משקיע יותר זמן בלהפוך את הפרויקט ליפה יותר ויזואלית ויותר אטרקטיבי למשתמש. חשבתי גם אולי לנסות לכתוב בוט עם אסטרטגיה פשוטה ולא מסובכת מדי כך, שהמשתמשים יוכלו לשחק מול עצמם. אבל מדובר בנושא מורכב שלא התעסקתי איתו בעבר ואת הזמן החלטתי להשקיע בנושאים אחרים במשחק.

במבט לאחר הייתי משנה את אופן העבודה שלי. אני קודם כל עבדתי על הגרפיקה ההתחלית ועל איך המשחק אמור להיראות, ורק אחר כך התחלתי עם התקשורת שרת לקוח ועל הפונקציונליות. במבט לאחור אני חושב שהייתי חוסך זמן ועבודה אם קודם הייתי מתכנן את איך המשחק אמור לעבוד ורק אחר כך דואג איך המשחק יראה.

עבור הכנת הפרויקט אני רוצה להודות למורה שליווה אותי במסע הזה- איתי. כשביקשתי הוא ישב איתי באופן פרטני ועזר לי עם המשחק. אני רוצה לבקש תודה גם מהחברים שלי. אני מאמין שתרמנו אחד לשני בצורה משמעותית. כל פעם שחבר ביקש עזרה שמחתי לעזור והיו כאלו שהתנדבו לעזור לי כשלא הבנתי נושאים מסוימים.

בבילוגרפיה

Building a basic HTTP Server from scratch in Python

<https://www.geeksforgeeks.org/building-a-basic-http-server-from-scratch-in-python/>

chatGpt

<https://chatgpt.com/?oai-dm=1>

CryptoJS

<https://cryptojs.gitbook.io/docs>

how do I make a time delay

<https://stackoverflow.com/questions/510348/how-do-i-make-a-time-delay>

How TO - Create a Draggable HTML Element

<https://www.w3schools.com/howto/howto_js_draggable.asp>

HTML DOM Button Object

<https://www.w3schools.com/jsref/dom_obj_pushbutton.asp>

HTML DOM Table Object

<https://www.w3schools.com/jsref/dom_obj_table.asp>

HTTP GET request in JavaScript?

<https://stackoverflow.com/questions/247483/http-get-request-in-javascript>

Multithreading in Python

<https://www.geeksforgeeks.org/multithreading-python-set-1/>

Python JSON

<https://www.w3schools.com/python/python_json.asp>

SQLite in Python || Python Tutorial || Learn Python Programming

<https://www.youtube.com/watch?v=c8yHTlrs9EA>

stop setinterval call in javascript

<https://stackoverflow.com/questions/109086/stop-setinterval-call-in-javascript>

נספחים:

החוקים המלאים:

לקוח מויקיפדיה ובפרויקט שלי החלטתי לשנות חלק קטן מהחוקים: <https://he.wikipedia.org/wiki/%D7%A1%D7%98%D7%A8%D7%98%D7%92%D7%95>

המשחק משוחק על גבי לוח של עשר על עשר. לכל שחקן 40 קלפים. 33 מהם הם כלים רגילים ברמות שונות, שישה מהם פצצות ודגל אחד. כל שחקן ראשי לראות את הדרגות רק של כליו ולא של היריב.

לפני תחילת המשחק, כל שחקן מסדר את 40 כליו כרצונו על פני הלוח, בארבע השורות הקרובות אליו. סידור מושכל של הכלים הוא חלק חיוני מאסטרטגיית המשחק ובעל השפעה רבה על מהלך המשחק.

מטרת המשחק היא לגלות את דגל היריב וללכוד אותו. שחקן התופס את דגל היריב מוכרז מיידית כמנצח. דרך נוספת לנצח היא במקרה בו לא נשארים חיילים היכולים לזוז לאחד מהשחקנים (ואז השחקן השני מנצח).

הזזת שחקנים:

כל חייל יכול לנוע משבצת אחת - לפנים, לאחור או לצדדים או באלכסון. יוצאים מן הכלל הם הפצצה והדגל, שאינם רשאים לנוע, וקלף מספר שתיים, היכול לנוע מספר משבצות כרצונו באחד מארבעת הכיוונים, אך לא לדלג מעל כלים, מעל הים או באלכסון. במרכז הלוח ישנם שני ריבועים בשטח כל אחד, הנחשבים כים ולא ניתן לנוע דרכם. בכל תור יכול השחקן להניע אחד מחייליו על פי כללים אלו. אם המשבצת פנויה, זוהי תנועה רגילה. אם המשבצת תפוסה על ידי חייל יריב מהלך זה הוא התקפה.

התקפה:

כאשר כלי אחד תוקף כלי אחר.

* כאשר חייל תוקף חייל מדרגה נמוכה ממנו החייל המותקף מורד מן הלוח, והחייל בעל הדרגה הגבוהה תופס את מקומו.
* כאשר חייל תוקף חייל מדרגה גבוהה ממנו, הוא מורד מן הלוח, והחייל המותקף נשאר על מקומו.
* כאשר שתי הדרגות שוות, שני החיילים מורדים מן הלוח.
* אם קלף אחד תוקף את אחד מקלפי התשע, קלף אחד מנצח ותופס את מקום החייל היריב.
* שחקן תשע שתוקף תמיד מנצח, אלא אם תקף פצצה או קלף תשע אחר של היריב.
* כל שחקן שתוקף פצצה מלבד קלף שלוש מתפוצץ גם הפצצה וגם הקלף יוצאים מהמשחק.
* דגל ופצצה לא יכולים לזוז.

נספח שתיים: קוד השרת

Main.py

import http.server  
import socketserver  
import random  
from Board import Board  
from game import game  
import string  
import sqlite3  
import json  
  
PORT = 8000  
  
playersWaiting = []  
startGame = False  
games = {}  
randomId = ""  
  
  
class MyHandler(http.server.BaseHTTPRequestHandler):  
 def do\_GET(self):  
 global playersWaiting, startGame, games, randomId  
 response = ""  
 params = self.parse\_QS() # Parse query parameters  
 action = params.get("action")  
 value = params.get("value")  
 player\_id = params.get("clientId")  
 gameId = params.get("game\_Id")  
  
 if action == "login":  
 # check if the user exist  
 # Replace "%20" with space  
 value = value.replace("%20", " ")  
 response = str(self.check\_credentials(value.split()[0], value.split()[1]))  
 elif action == "signUp":  
 # Replace "%20" with space  
 value = value.replace("%20", " ")  
 response = self.sign\_up(value.split()[0], value.split()[1])  
 elif action == "sendText":  
 games[gameId].sendText(player\_id, value)  
  
 elif action == "playTurn":  
 games[gameId].playTurn(value)  
  
 elif action == "lookingForGame":  
 if player\_id in playersWaiting and startGame:  
 response = randomId  
 startGame = False  
  
 playersWaiting.remove(games[randomId].getPlayerUp())  
 playersWaiting.remove(games[randomId].getPlayerDown())  
  
 elif player\_id not in playersWaiting:  
 playersWaiting.append(player\_id)  
 if len(playersWaiting) >= 2 and not startGame:  
 randomId = self.RandomId()  
 games[randomId] = game(player\_id, playersWaiting[0])  
 response = randomId  
 startGame = True  
  
 elif action == "getGame":  
 response = {} # board,chat,time  
 response["board"] = games[gameId].getBoardAsArray(player\_id)  
 response["chat"] = games[gameId].getText()  
 response["time"] = games[gameId].getTime()  
 if isinstance(response, list) and games[gameId].deletePlayer(player\_id):  
 # if reached here it means both player got the message and we can erase the game  
 games.pop(gameId, None)  
 else:  
 response = json.dumps(response)  
  
 elif action == "isMyTurn":  
 response = str(games[gameId].turn == player\_id)  
  
 elif action == "switchCards":  
 games[gameId].switchCards(value, player\_id)  
  
 self.send\_response(200)  
 self.send\_header("Content-type", "text/plain")  
 self.send\_header("Access-Control-Allow-Origin", "\*")  
 self.end\_headers()  
 self.wfile.write(bytes(f"{response}", "utf-8"))  
  
 def parse\_QS(self):  
 *"""Parses the query string from the request URL."""* qs = {}  
 if "?" in self.path:  
 qs = dict(qc.split("=") for qc in self.path.split("?")[1].split("&"))  
 return qs  
  
 def RandomId(self):  
 return ''.join([random.choice(string.ascii\_letters + string.digits) for n in range(16)])  
  
 def check\_credentials(self, username, password):  
 # Connect to the database  
 conn = sqlite3.connect("database")  
 curr = conn.cursor()  
  
 # create and use the query to check if user is in the database  
 check\_query = '''SELECT COUNT(\*) FROM usersInfo WHERE username = ? AND password = ?;'''  
 curr.execute(check\_query, (username, password))  
  
 # Fetch the result of the query  
 result = curr.fetchone()[0]  
  
 # Close the connection  
 curr.close()  
 conn.close()  
  
 # If the result is 1, the username and password exist; otherwise, they don't  
 return result == 1  
  
 def sign\_up(self,username, password):  
 # Connect to the database  
 conn = sqlite3.connect("database")  
 cursor = conn.cursor()  
  
 # Check if username already exists  
 check\_query = "SELECT COUNT(\*) FROM usersInfo WHERE username = ?;"  
 cursor.execute(check\_query, (username,))  
 result = cursor.fetchone()[0]  
  
 if result == 1:  
 conn.close()  
 return "False" # Username already exists, return False  
  
 # Insert the user into the database and set default rating for the user  
 insert\_query = "INSERT INTO usersInfo (username, password, rating) VALUES (?, ?, 100);"  
 cursor.execute(insert\_query, (username, password))  
  
 # Commit the changes  
 conn.commit()  
  
 # Close the connection  
 conn.close()  
  
 return "True" # User successfully signed up, return True  
  
  
with socketserver.TCPServer(("", PORT), MyHandler) as httpd:  
 print("Serving at port", PORT)  
 httpd.serve\_forever()

game.py

from Board import Board  
import threading  
import time  
  
  
class game():  
 def \_\_init\_\_(self, playerDown, playerUp):  
 self.playerDown = playerDown  
 self.playerUp = playerUp  
 self.board = Board(playerUp, playerDown)  
 self.turn = playerDown  
 self.chat = ""  
 self.status = "not started"  
  
 self.timeThread = showTimeThread(10)  
 self.timeThread.start()  
  
 def getBoardAsArray(self, player):  
 # if status is not going someone won  
 if self.status not in ["going", "not started"]:  
 return self.status  
 return self.board.getBoardAsString(player)  
  
 def getPlayerDown(self):  
 return self.playerDown  
  
 def getPlayerUp(self):  
 return self.playerUp  
  
 def deletePlayer(self, player):  
 if player == self.playerDown:  
 self.playerDown = ""  
 elif player == self.playerUp:  
 self.playerUp = ""  
 return self.playerUp == "" and self.playerDown == ""  
  
 def setPlayerDown(self, playerDown):  
 self.playerDown = playerDown  
  
 def SetPlayerUp(self, playerUp):  
 self.playerUp = playerUp  
  
 def playTurn(self, value):  
 row1, col1, row2, col2 = map(int, value)  
 # Update the destination board with the piece  
 if str(self.turn) == str(self.board.playerDown):  
 if self.turn != self.board.getCardOwner(row1, col1):  
 return  
 if self.board.movePiece(row1, col1, row2, col2, self.turn):  
 # if game is over winner is the one who played right now  
 self.status = self.turn  
 else:  
 if self.turn != self.board.getCardOwner(9 - row1, col1):  
 return  
 if self.board.movePiece(9 - row1, col1, 9 - row2, col2, self.turn):  
 # if game is over winner is the one who played right now  
 self.status = self.turn  
  
 # Switch player turn  
 if self.turn == self.playerDown:  
 self.turn = self.playerUp  
 else:  
 self.turn = self.playerDown  
  
 def getTime(self):  
 return self.timeThread.getTime()  
  
 def switchCards(self, value, player):  
 row1, col1, row2, col2 = map(int, value)  
  
 # Update the destination board with the piece  
 if str(player) == str(self.board.playerDown):  
 self.board.switchCards(row1, col1, row2, col2)  
 else:  
 self.board.switchCards(9 - row1, col1, 9 - row2, col2)  
  
 def sendText(self, player, value):  
 self.chat += player + ": " + value + "<br>"  
  
 def getText(self):  
 return self.chat  
  
  
class showTimeThread(threading.Thread):  
 def \_\_init\_\_(self, seconds):  
 threading.Thread.\_\_init\_\_(self)  
 self.time = seconds  
  
 def run(self):  
 while self.time > 0:  
 time.sleep(1)  
 self.time -= 1  
  
 def getTime(self):  
 minutes = self.time // 60  
 remaining\_seconds = self.time % 60  
 return f"{minutes}:{remaining\_seconds:02d}"

board.py

import threading  
import time  
import sqlite3  
  
class card:  
 def \_\_init\_\_(self, kind, player=0, show=False):  
 # show is true or false so when a card beats another it should be seen by both players  
 self.kind = kind  
 self.player = player  
 self.show = show  
  
 def getPlayer(self):  
 return self.player  
  
 def getShow(self):  
 return self.show  
  
 def setShow(self, show):  
 self.show = show  
  
 def getKind(self):  
 return self.kind  
  
 def setKind(self, kind):  
 self.kind = kind  
  
 def setPlayer(self, player):  
 self.player = player  
  
 def copy(self, other):  
 self.kind = other.kind  
 self.player = other.player  
 self.show = other.show  
  
 def shouldShow(self, player):  
 return str(self.player) == str(player) or str(self.player) == "0" or self.show  
  
  
class Board:  
 def \_\_init\_\_(self, playerUp, playerDown, rows=10, columns=10):  
 self.board = []  
 self.playerUp = playerUp # Assign playerUp  
 self.playerDown = playerDown # Assign playerDown  
 for row in range(rows):  
 current\_row = []  
 for col in range(columns):  
 cell\_content = self.determineCellContent(row, col)  
 if row < 4:  
 current\_row.append(card(cell\_content, playerUp))  
 elif row > 5:  
 current\_row.append(card(cell\_content, playerDown))  
 else:  
 current\_row.append(card(cell\_content))  
 self.board.append(current\_row)  
  
 def getTopPlayer(self):  
 return self.playerUp  
  
 def getbuttomPlayer(self):  
 return self.playerDown  
  
 def getBoard(self):  
 return self.board  
  
 def getPiece(self, row, col):  
 return self.board[int(row)][int(col)].getKind()  
  
 def movePiece(self, fromRow, fromCol, toRow, toCol):  
 placeFrom = self.board[fromRow][fromCol]  
 placeTo = self.board[toRow][toCol]  
  
 if placeTo.getKind() == "flag":  
 # game is over  
 return True  
 elif placeTo.getKind() == "green":  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 self.board[fromRow][fromCol] = card("green")  
 return  
 # there is a fight between two cards  
 elif placeTo.getKind() == "bomb":  
 # incase of a bomb only card 3 survives  
 if placeFrom.getKind() == "image3":  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 else:  
 placeTo.copy(card("green"))  
 placeFrom.copy(card("green"))  
 # if there is an enemy card or a one attacking a nine  
 elif (placeFrom.getKind()[-1] > placeTo.getKind()[-1]) or (  
 placeFrom.getKind()[-1] == "1" and placeTo.getKind()[-1] == "9"):  
 self.board[toRow][toCol] = self.board[fromRow][fromCol]  
 elif placeFrom.getKind()[-1] == placeTo.getKind()[-1]:  
 placeTo.copy(card("green"))  
 self.board[fromRow][fromCol] = card("green")  
 # game is not over  
 showCard = showCardThread(self.board[toRow][toCol])  
 showCard.start()  
 return False  
  
 def getBoardAsArray(self, player):  
 # if the player is the player and the top the board should be reversed  
 reversedOrNormalBoard = self.board  
 if str(player) == str(self.playerUp):  
 reversedOrNormalBoard = self.board[::-1]  
 boardToSend = []  
 for row in reversedOrNormalBoard:  
 rowArr = []  
 for card in row:  
 if card.shouldShow(player):  
 rowArr.append(card.getKind())  
 else:  
 rowArr.append("enemy")  
 boardToSend.append(rowArr)  
 return boardToSend  
  
 @staticmethod  
 def determineCellContent(row, col):  
 # Connect to SQLite database  
 conn = sqlite3.connect("database")  
 cursor = conn.cursor()  
  
 query = f"SELECT col{col} FROM basicBoard LIMIT 1 OFFSET ?"  
 # Execute the query with the given row (adjust for zero-based index)  
 cursor.execute(query, (row,))  
  
 # Fetch the result  
 result = cursor.fetchone()  
  
 # Close the connection  
 conn.close()  
  
 # Return the result  
 return result[0]  
  
 def switchCards(self, fromRow, fromCol, toRow, toCol):  
 tempCard = card(self.board[toRow][toCol].getKind(), self.board[toRow][toCol].getPlayer())  
 self.board[toRow][toCol].copy(self.board[fromRow][fromCol])  
 self.board[fromRow][fromCol].copy(tempCard)  
  
 def getCardOwner(self,row,col):  
 return self.board[row][col].getPlayer()  
  
  
class showCardThread(threading.Thread):  
 def \_\_init\_\_(self, card):  
 threading.Thread.\_\_init\_\_(self)  
 self.card = card  
  
 def run(self):  
 # make the card visible to both players so both player will now what card was that.  
 player = self.card.setShow(True)  
 time.sleep(3)  
 player = self.card.setShow(False)

קוד הלקוח:

Boardfunctions.js

function createGame(rows, cols) {  
 // Get the container element  
 var container = document.createElement('div');  
 container.id = 'contain\_game';  
 container.setAttribute("style", "display: flex; justify-content: space-between;");  
 document.getElementById("left\_div").appendChild(container);  
   
   
 // Create the table-clock section div  
 var tableClockSection = document.createElement("div");  
 tableClockSection.setAttribute("class", "table-clock-section");  
 tableClockSection.setAttribute("style", "display: flex; flex-direction: column; align-items: flex-start;");  
  
 // Create the table using the createTable function  
 let table = createTable(rows, cols);  
  
 // Create the clock div  
 var clockDiv = document.createElement("div");  
 clockDiv.id = "clock";  
 clockDiv.setAttribute("style", "font-size: 24px; margin-bottom: 10px;");  
  
 // Append the table and clock div to the table-clock section  
 tableClockSection.appendChild(clockDiv);  
 tableClockSection.appendChild(table);  
  
 // Create the text section div  
 var textSection = document.createElement("div");  
 textSection.setAttribute("class", "text-section");  
 textSection.setAttribute("style", "flex: 1; margin-left: 20px;");  
  
 // Create the textContainer div  
 var textContainer = document.createElement("div");  
 textContainer.setAttribute("id", "textContainer");  
 textContainer.setAttribute("style", "border: 1px solid black; padding: 10px; margin-bottom: 10px;");  
  
 // Create an input element for text input  
 var inputText = document.createElement("input");  
 inputText.setAttribute("type", "text");  
 inputText.setAttribute("id", "input-text");  
 inputText.setAttribute("placeholder", "Type your text here...");  
  
 // Create a button for adding text  
 var addButton = document.createElement("button");  
 addButton.textContent = "Add Text";  
 addButton.addEventListener("click", sendText);  
  
 // Append the textContainer, inputText, and addButton to the text section  
 textSection.appendChild(textContainer);  
 textSection.appendChild(inputText);  
 textSection.appendChild(addButton);  
  
 // Append the text section and table-clock section to the container  
 container.appendChild(tableClockSection);  
 container.appendChild(textSection);  
}  
   
function createTable(rows, cols) {  
 // Create a table element  
 var table = document.createElement('table');  
  
 // Create rows and cells  
 for (var i = 0; i < rows; i++) {  
 var row = document.createElement('tr');  
 for (var j = 0; j < cols; j++) {  
 var cell = document.createElement('td');  
 cell.setAttribute('id', i.toString() + j.toString());  
  
 // Add dragstart event to each cell  
 cell.addEventListener('dragstart', function(event) {  
 event.dataTransfer.setData('text/plain', event.target.id);  
 });  
  
 // Make the cell draggable  
 cell.setAttribute('draggable', 'true');  
 row.appendChild(cell);  
 }  
 table.appendChild(row);  
 }  
  
 // Add drop and dragover event listeners to the table  
 table.addEventListener('drop', function(event) {  
 event.preventDefault();  
 var data = event.dataTransfer.getData('text/plain');  
 var target = event.target;  
  
 // Ensure the drop target is a table cell  
 if (target.tagName.toLowerCase() === 'td') {  
 playTurn(data, target.id);  
 }  
 });  
  
 table.addEventListener('dragover', function(event) {  
 event.preventDefault();  
 });  
  
 return table;  
}  
  
// check if we want to move the image to a new location which is legal  
function checkIfLocationOk(lastTdId, tdId) {  
 var lastTd = document.getElementById(lastTdId);  
 var td = document.getElementById(tdId);  
   
 // Calculate the absolute difference between rows and columns  
 var rowDiff = Math.abs(tdId[0] - lastTdId[0]);  
 var colDiff = Math.abs(tdId[1] - lastTdId[1]);  
   
 // If the difference in rows or columns is more than 1  
 if (rowDiff > 1 || colDiff > 1) {  
 // If the card is not a 2, return false  
 if (!lastTd.className.includes("2")) {  
 return false;  
 } else {  
 // If the card is a 2, it can jump more than one tile  
 // Check if it's a valid jump (only vertical or horizontal)  
 if (rowDiff > 0 && colDiff > 0) {  
 return false; // diagonal jump is not allowed  
 }  
   
 // Check if there are cards between initial and final positions  
 if (rowDiff === 0) { // Horizontal jump  
 var startCol = Math.min(lastTdId[1], tdId[1]);  
 var endCol = Math.max(lastTdId[1], tdId[1]);  
 for (var col = startCol + 1; col < endCol; col++) {  
 var card = document.getElementById(lastTdId[0] + col);  
 if (!card.className.includes("green")) {  
 return false; // Invalid jump over non-green or non-blue cards  
 }  
 }  
 } else if (colDiff === 0) { // Vertical jump  
 var startRow = Math.min(lastTdId[0], tdId[0]);  
 var endRow = Math.max(lastTdId[0], tdId[0]);  
 for (var row = startRow + 1; row < endRow; row++) {  
 var card = document.getElementById(row + lastTdId[1]);  
 if (!card.className.includes("green")) {  
 return false; // Invalid jump over non-green or non-blue cards  
 }  
 }  
 }  
 }  
 }  
 return true;  
}  
  
function moveImage(lastTdId, tdId, gameStarted) {  
 var td = document.getElementById(tdId);  
 var lastTd = document.getElementById(lastTdId);  
  
 // if game is not started   
 if (!gameStarted&&tdId[0] > 5) {  
 // Send the move to the server  
 callServer("switchCards", lastTdId.toString() + tdId.toString(), function(response) {  
 updateBoard();  
 });  
 return;  
 }  
  
 // if game has started and the cell contains an image  
 if (gameStarted && lastTd.classList[1].includes("image")&&!td.classList[1].includes("image")) {  
 // Check if the location of the new image is legal  
 if (!checkIfLocationOk(lastTdId, tdId))  
 return;  
  
 // Send the move to the server  
 callServer("playTurn", lastTdId.toString() + tdId.toString(), function(response) {  
 updateBoard();  
 });  
 }  
}  
  
function updateBoard(){  
 //update the board  
 callServer("getGame","", function(response) {  
 // board will be an array inside of array until the game is over  
 //once over it will be the name of the winner  
 response = JSON.parse(response);  
 let board = response["board"];  
 let chat = response["chat"];  
 let time = response["time"];  
   
 if (!Array.isArray(board)){  
 // if the first one is not an array it means game is over  
 gameOver(board);  
 return;  
 }  
   
 for (var i = 0; i < 10; i++) {  
 for (var j = 0; j < 10; j++) {  
 let td = document.getElementById(i.toString()+j.toString());  
 let td\_class = board[i][j];  
 td.className = '';  
 if (td\_class!="blue") {  
 td.classList.add("board\_piece");  
 }  
 td.classList.add(td\_class);  
 }  
 }  
   
 //update the chat  
 if (chat != "")  
 document.getElementById("textContainer").innerHTML = chat + "<br />";  
   
 //update the time  
 document.getElementById("clock").innerHTML = time;  
 if(time == "0:00"){  
 gameStarted = true;  
 }  
 });  
}  
  
//end the game  
function gameOver(winner){  
 if (userName == winner)  
 alert("you won");  
 else  
 alert("you lost");  
 document.getElementById("contain\_game").remove();  
 clearInterval(updateBoardInterval);  
 gameStarted = false;  
 main();   
}

game.js

var refreshIntervalId;  
 var gameStarted = false;  
 var gameId = "";  
 var serverIp = "10.0.0.8";  
   
 function sendText() {  
 var textToAdd = document.getElementById("input-text").value;  
 if (textToAdd.trim() !== "") {  
 callServer("sendText", textToAdd, function(response) {  
 document.getElementById("input-text").value = ""; // Clear the input field after adding text  
 });  
 }  
 }  
 function main() {  
 // Create a button to start the game  
 const button = document.createElement('button');  
 button.innerText = 'Start Game';  
 button.id = "idToRemove";  
 document.getElementById('left\_div').appendChild(button);  
 button.addEventListener('click', () => {   
 document.getElementById('idToRemove').remove();  
 refreshIntervalId = setInterval(function() {  
 checkStartGame();  
 }, 1000);  
 });  
 }  
function encryptMsg(message) {  
 // still not written  
 return message;  
}  
  
function callServer(actionToDo, valueToPass, callback) {  
 // encrypt message  
 let encryptedValue = encryptMsg(valueToPass);  
  
 // send encrypt message  
 $.get("http://127.0.0.1:8000", {  
 action: actionToDo,  
 game\_Id: gameId,  
 value: encryptedValue,  
 clientId: userName  
 }, function (msg\_from\_server) {  
 callback(msg\_from\_server);  
 });  
}  
  
function playTurn(lastTdId,tdId) {  
 //see if game started or its still time to create the board  
 if (!gameStarted){  
 moveImage(lastTdId,tdId,gameStarted);  
 }else {  
 //send the move to the other player  
 callServer("isMyTurn","", function(response) {  
 if (response == "True") {  
 moveImage(lastTdId,tdId,gameStarted);  
 }  
 else{  
 alert("waiting for opponent to play");  
 }  
 });  
 }  
}  
  
function sendText() {  
 var textToAdd = document.getElementById("input-text").value;  
 if (textToAdd.trim() !== "") {  
 callServer("sendText", textToAdd, function(response) {  
 document.getElementById("input-text").value = ""; // Clear the input field after adding text  
 });  
 }  
}  
  
let updateBoardInterval;  
function checkStartGame() {  
 callServer("lookingForGame", "", function(response) {  
 if (response != "") {  
 // if response is not false it means its the gameId  
 gameId = response;  
   
 console.log("Two players joined. Starting the game...");  
 //stop calling checkStartGame  
 clearInterval(refreshIntervalId);  
   
 //remove the top  
 document.getElementById('instructions').remove();  
   
 // create Board  
 createGame(10, 10);  
 updateBoard();  
 updateBoardInterval = setInterval(updateBoard, 1000);  
   
 } else {  
 console.log("Waiting for another player...");  
 }  
 });  
 }

signUp.html

<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
  
 <meta name="description" content="">  
 <meta name="author" content="">  
  
 <title>Stratego online game by Tal Divon</title>  
  
 <!-- CSS FILES -->  
  
   
  
  
  
  
 <link href="css/bootstrap.min.css" rel="stylesheet">  
  
 <link href="css/bootstrap-icons.css" rel="stylesheet">  
  
 <link href="css/owl.carousel.min.css" rel="stylesheet">  
  
 <link href="css/owl.theme.default.min.css" rel="stylesheet">  
  
 <link href="css/tooplate-gotto-job.css" rel="stylesheet">  
   
   
 <script src="js/jquery.min.js"></script>  
 <script src="js/owl.carousel.min.js"></script>  
 <script src="js/counter.js"></script>  
 <script src="js/custom.js"></script>  
  
   
 <link rel="stylesheet" href="css/main.css" />   
 <script src="js/jquery-3.7.1.min.js"></script>  
 <script src="js/game.js"></script>  
 <script src="js/boardFunctions.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>  
  
  
 <script>  
 let userName;  
 function signUp(){  
 var userName = document.getElementById('uname').value;  
 var password = document.getElementById('psw').value;  
 var password2 = document.getElementById('repeatPsw').value;  
 if (password!= password2){  
 alert("Passwords do not match.")  
 return;  
 }else if (validateName(userName)!= "ok"){  
 alert("username " + validateName(userName));  
 return;  
 }else if (validateName(password)!= "ok"){  
 alert("password "+validateName(password));  
 return;  
 }  
 callServer("signUp",userName+" "+ hashPassword(password), function(response) {  
 if (response == "True"){  
 alert("joined successfully")  
 }else{  
 alert("username already exist");  
 }  
 });   
 }  
   
 function hashPassword(password) {  
 return CryptoJS.SHA256(password).toString();  
 }  
   
 function validateName(name) {  
 let isPasswordOk = "ok";  
 if (name.includes(' ')) {  
 isPasswordOk = "should not consist spaces";  
 }  
 if (name.length <= 3) {  
 isPasswordOk = "should be higher than 3";  
 }  
 if (name.length >= 16) {  
 isPasswordOk = "should be lower than 16";  
 }  
 return isPasswordOk;  
 }  
 </script>  
  
 </head>  
   
 <body id="top">  
  
 <nav class="navbar navbar-expand-lg">  
 <div class="container">  
 <a class="navbar-brand d-flex align-items-center" href="index.html">  
 <img src="images/logo.jpg" class="logo">  
 </a>  
  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
  
 <div class="collapse navbar-collapse" id="navbarNav">  
 <ul class="navbar-nav align-items-center ms-lg-5">  
  
 <li class="nav-item ms-lg-auto">  
 <a class="nav-link" href="register.html">Register</a>  
 </li>  
  
 <li class="nav-item">  
 <a class="nav-link custom-btn btn" href="index.html">Login</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
  
 <main>  
  
 <section class="hero-section d-flex justify-content-center align-items-center">  
 <div class="section-overlay"></div>  
  
 <div class="container" id="container" name="container">  
 <div class="row" id="left\_div" name="left\_div">  
  
 <div class="col-lg-6 col-12 mb-5 mb-lg-0">  
 <div class="hero-section-text mt-5">  
 <h6 class="text-white">Startego Battle</h6>  
 <h1 class="hero-title text-white mt-4 mb-4">Stratego. <br> Best online game ever</h1>  
  
 <a href="instructions.html" class="custom-btn custom-border-btn btn">Click for Game instructions</a>  
 </div>  
 </div>  
  
 <div class="col-lg-6 col-12" id="form" name="form">  
 <div class="custom-form hero-form" action="#" method="get" role="form">  
 <h3 class="text-white mb-3">Login to start the game</h3>  
  
 <div class="row">  
 <div class="col-12">  
 <div class="input-group">  
 <span class="input-group-text" id="basic-addon1"><i class="bi-person custom-icon"></i></span>  
 <input type="text" name="uname" id="uname" class="form-control" placeholder="Username" required>  
 </div>  
 </div>  
 <div class="col-md-6 col-12">  
 <div class="input-group">  
 <span class="input-group-text" id="basic-addon2"><i class="bi-geo-alt custom-icon"></i></span>  
 <input type="password" name="psw" id="psw" class="form-control" placeholder="Password" required>  
 </div>  
 </div>  
 <div class="col-md-6 col-12">  
 <div class="input-group">  
 <span class="input-group-text" id="basic-addon3"><i class="bi-geo-alt custom-icon"></i></span>  
 <input type="password" name="repeatPsw" id="repeatPsw" class="form-control" placeholder="Repeat Password" required>  
 </div>  
 </div>  
 <div class="col-12">  
 <button class="form-control" onclick="signUp()">  
 Sign Up  
 </button>  
 </div>  
 </div>  
 </div>  
 </div>  
   
 <div id = "contain\_board">  
 </div>  
  
 </div>  
 </div>  
 </section>  
   
 </main>  
  
  
  
 </body>  
</html>

Register.html

<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
  
 <meta name="description" content="">  
 <meta name="author" content="">  
  
 <title>Stratego online game by Tal Divon</title>  
  
 <!-- CSS FILES -->  
  
   
  
  
  
  
 <link href="css/bootstrap.min.css" rel="stylesheet">  
  
 <link href="css/bootstrap-icons.css" rel="stylesheet">  
  
 <link href="css/owl.carousel.min.css" rel="stylesheet">  
  
 <link href="css/owl.theme.default.min.css" rel="stylesheet">  
  
 <link href="css/tooplate-gotto-job.css" rel="stylesheet">  
   
   
 <script src="js/jquery.min.js"></script>  
 <script src="js/owl.carousel.min.js"></script>  
 <script src="js/counter.js"></script>  
 <script src="js/custom.js"></script>  
  
   
 <link rel="stylesheet" href="css/main.css" />   
 <script src="js/jquery-3.7.1.min.js"></script>  
 <script src="js/game.js"></script>  
 <script src="js/boardFunctions.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>  
  
  
 <script>  
 let userName;  
 function login(){  
 userName = document.querySelector('input[name="uname"]').value;  
 var password = document.querySelector('input[name="psw"]').value;  
   
 if (validateName(userName)!= "ok"){  
 alert("username " + validateName(userName));  
 return;  
 }else if (validateName(password)!= "ok"){  
 alert("password "+validateName(password));  
 return;  
 }  
   
 callServer("login",userName+" "+hashPassword(password), function(response) {  
 if (response == "True"){  
 var elementToRemove = document.getElementById("form");  
 elementToRemove.remove();  
 main();  
 }else{  
 alert("user name or password is incorrect");  
 }  
 });   
 }  
   
 function hashPassword(password) {  
 return CryptoJS.SHA256(password).toString();  
 }  
   
 function validateName(name) {  
 let isPasswordOk = "ok";  
 if (name.includes(' ')) {  
 isPasswordOk = "should not consist spaces";  
 }  
 if (name.length <= 3) {  
 isPasswordOk = "should be higher than 3";  
 }  
 if (name.length >= 16) {  
 isPasswordOk = "should be lower than 16";  
 }  
 return isPasswordOk;  
 }  
 </script>  
  
 </head>  
   
 <body>  
  
 <nav class="navbar navbar-expand-lg">  
 <div class="container" id = "top">  
 <a class="navbar-brand d-flex align-items-center" href="index.html">  
 <img src="images/logo.jpg" class="logo">  
 </a>  
  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
  
 <div class="collapse navbar-collapse" id="navbarNav">  
 <ul class="navbar-nav align-items-center ms-lg-5">  
  
 <li class="nav-item ms-lg-auto">  
 <a class="nav-link" href="register.html">Register</a>  
 </li>  
  
 <li class="nav-item">  
 <a class="nav-link custom-btn btn" href="#">Login</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
  
 <main>  
  
 <section class="hero-section d-flex justify-content-center align-items-center">  
 <div class="section-overlay"></div>  
  
 <div class="container" id="container" name="container">  
 <div class="row" id="left\_div" name="left\_div">  
  
 <div class="col-lg-6 col-12 mb-5 mb-lg-0" id = "instructions">  
 <div class="hero-section-text mt-5">  
 <h6 class="text-white">Startego Battle</h6>  
 <h1 class="hero-title text-white mt-4 mb-4">Stratego. <br> Best online game ever</h1>  
  
 <a href="instructions.html" class="custom-btn custom-border-btn btn">Click for Game instructions</a>  
 </div>  
 </div>  
  
 <div class="col-lg-6 col-12" id="form" name="form">  
 <div class="custom-form hero-form" action="#" method="get" role="form">  
 <h3 class="text-white mb-3">Login to start the game</h3>  
  
 <div class="row">  
 <div class="col-lg-6 col-md-6 col-12">  
 <div class="input-group">  
 <span class="input-group-text" id="basic-addon1"><i class="bi-person custom-icon"></i></span>  
  
 <input type="text" name="uname" id="uname" class="form-control" placeholder="Username" required>  
 </div>  
 </div>  
  
 <div class="col-lg-6 col-md-6 col-12">  
 <div class="input-group">  
 <span class="input-group-text" id="basic-addon2"><i class="bi-geo-alt custom-icon"></i></span>  
  
 <input type="password" name="psw" id="psw" class="form-control" placeholder="Password" required>  
 </div>  
 </div>  
  
 <div class="col-lg-12 col-12">  
 <button class="form-control" onclick = "login()">  
 Login  
 </button>  
 </div>  
   
 </div>  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </section>  
   
 </main>  
  
  
  
 </body>  
</html>

table {  
 border-collapse: collapse;  
 box-shadow: 5px 5px 5px #999  
 }  
 td {  
 border: 1px dashed;  
 width: 75px;  
 height: 75px;  
 box-sizing: border-box;  
 }  
 .green { background: #00FF00; } /\* Green background for green squares \*/  
 .blue { background: #0000FF; } /\* Blue background for blue squares \*/  
 img{  
 width: 50px;  
 height: 50px;  
 background-size: cover; /\* or use 'contain' based on your preference \*/  
 display:table-cell;   
 background: #00FF00;   
 background-repeat: no-repeat;  
 background-size: 75% 75%;  
 }  
 .logo {  
 width: 332px;  
 height: 126px;  
 }  
 .image1 {  
 background-image: url('../images/image1.jpg');  
 }  
   
 .image2 {  
 background-image: url('../images/image2.jpg');  
 }  
   
 .image3 {  
 background-image: url('../images/image3.jpg');  
 }  
   
 .image4 {  
 background-image: url('../images/image4.jpg');  
 }  
 .image5 {  
 background-image: url('../images/image5.jpg');  
 }  
   
 .image6 {  
 background-image: url('../images/image6.jpg');  
 }  
 .image7 {  
 background-image: url('../images/image7.jpg');  
 }  
   
 .image8 {  
 background-image: url('../images/image8.jpg');  
 }  
 .image9 {  
 background-image: url('../images/image9.jpg');  
 }  
   
 .flag {  
 background-image: url('../images/flag.jpg');  
 }  
   
 .bomb {  
 background-image: url('../images/bomb.jpg');  
 }  
 .enemy {  
   
 background-image: url('../images/enemy.jpg');   
 }  
  
 .board\_piece {  
 background-color: #00FF00;

instractions.html

<!doctype html>  
<html lang="en">  
 <head>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
  
 <meta name="description" content="">  
 <meta name="author" content="">  
  
 <title>Stratego online game by Tal Divon</title>  
  
 <!-- CSS FILES -->  
  
   
  
  
  
  
 <link href="css/bootstrap.min.css" rel="stylesheet">  
  
 <link href="css/bootstrap-icons.css" rel="stylesheet">  
  
 <link href="css/owl.carousel.min.css" rel="stylesheet">  
  
 <link href="css/owl.theme.default.min.css" rel="stylesheet">  
  
 <link href="css/tooplate-gotto-job.css" rel="stylesheet">  
   
   
 <script src="js/jquery.min.js"></script>  
 <script src="js/bootstrap.min.js"></script>  
 <script src="js/owl.carousel.min.js"></script>  
 <script src="js/counter.js"></script>  
 <script src="js/custom.js"></script>  
  
   
 <link rel="stylesheet" href="css/main.css" />   
 <script src="js/jquery-3.7.1.min.js"></script>  
 <script src="js/game.js"></script>  
 <script src="js/boardFunctions.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/crypto-js/4.1.1/crypto-js.min.js"></script>  
  
 </head>  
   
 <body id="top">  
  
 <nav class="navbar navbar-expand-lg">  
 <div class="container">  
 <a class="navbar-brand d-flex align-items-center" href="index.html">  
 <img src="images/logo.jpg" class="logo">  
 </a>  
  
 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">  
 <span class="navbar-toggler-icon"></span>  
 </button>  
  
 <div class="collapse navbar-collapse" id="navbarNav">  
 <ul class="navbar-nav align-items-center ms-lg-5">  
  
 <li class="nav-item ms-lg-auto">  
 <a class="nav-link" href="#">Register</a>  
 </li>  
  
 <li class="nav-item">  
 <a class="nav-link custom-btn btn" href="#">Login</a>  
 </li>  
 </ul>  
 </div>  
 </div>  
 </nav>  
  
 <main>  
  
 <section class="hero-section d-flex justify-content-center align-items-center">  
 <div class="section-overlay"></div>  
  
 <div class="container" id="container" name="container">  
 <div class="row" id="left\_div" name="left\_div">  
  
 <div class="col-lg-6 col-12 mb-5 mb-lg-0">  
 <div class="hero-section-text mt-5">  
 <h6 class="text-white">Startego Battle</h6>  
  
 <h3 class="hero-title text-white mt-4 mb-4">Here's a breakdown of the basic rules of Stratego:</h3>  
  
 <li class="bold text-white mt-4 mb-4">Setup</li>  
 <div class="text-white">The game is played on a square board divided by a central lake area (usually 4 squares).  
 Each player secretly arranges their 40 pieces on their side of the board. These pieces have different ranks, ranging from Marshal (highest) to Spy (lowest). There's also a Flag piece that must be hidden among the other pieces.  
 </div>  
 <li class="bold text-white mt-4 mb-4">Movement</li>  
 <div class="text-white">  
 Players take turns moving one piece per turn. Pieces can move one square forward, backward, left, or right (not diagonally).  
 The Scout is the exception - it can move any number of squares in a straight line in any direction.  
 Pieces cannot jump over other pieces or move onto squares already occupied by friendly pieces.  
 The lakes in the center are impassable.  
 A piece cannot move back and forth between the same two squares in three consecutive turns.  
 </div>  
 <li class="bold text-white mt-4 mb-4">Combat</li>  
 <div class="text-white">When a red and blue piece occupy adjacent squares, they can attack.  
 To attack, the attacker taps their piece on the defender's.  
 Ranks are then revealed - the higher-ranked piece captures the lower-ranked one and stays on the square. If the ranks are equal, both pieces are removed from the board.  
 The Flag cannot move and is automatically captured if attacked by any enemy piece.  
 Winning  
 </div>  
 <div class="text-white">  
 The first player to capture the opponent's Flag wins the game.  
 Alternatively, if one player loses all their movable pieces, the other player wins.  
 There are also some optional rule variations that can add more challenge, such as "Aggressor Advantage" (attacker wins in tie rank battles) and "Silent Defense" (defender keeps their rank hidden during combat).  
 </div><br>  
  
 <a href="index.html" class="custom-btn custom-border-btn btn">Click for Homepage</a>  
  
  
 </div>  
 </div>  
  
 </div>  
 </div>  
 </section>  
   
 </main>  
 </body>  
</html>

Css.html

table {  
 border-collapse: collapse;  
 box-shadow: 5px 5px 5px #999  
 }  
 td {  
 border: 1px dashed;  
 width: 75px;  
 height: 75px;  
 box-sizing: border-box;  
 }  
 .green { background: #00FF00; } /\* Green background for green squares \*/  
 .blue { background: #0000FF; } /\* Blue background for blue squares \*/  
 img{  
 width: 50px;  
 height: 50px;  
 background-size: cover; /\* or use 'contain' based on your preference \*/  
 display:table-cell;   
 background: #00FF00;   
 background-repeat: no-repeat;  
 background-size: 75% 75%;  
 }  
 .logo {  
 width: 332px;  
 height: 126px;  
 }  
 .image1 {  
 background-image: url('../images/image1.jpg');  
 }  
   
 .image2 {  
 background-image: url('../images/image2.jpg');  
 }  
   
 .image3 {  
 background-image: url('../images/image3.jpg');  
 }  
   
 .image4 {  
 background-image: url('../images/image4.jpg');  
 }  
 .image5 {  
 background-image: url('../images/image5.jpg');  
 }  
   
 .image6 {  
 background-image: url('../images/image6.jpg');  
 }  
 .image7 {  
 background-image: url('../images/image7.jpg');  
 }  
   
 .image8 {  
 background-image: url('../images/image8.jpg');  
 }  
 .image9 {  
 background-image: url('../images/image9.jpg');  
 }  
   
 .flag {  
 background-image: url('../images/flag.jpg');  
 }  
   
 .bomb {  
 background-image: url('../images/bomb.jpg');  
 }  
 .enemy {  
   
 background-image: url('../images/enemy.jpg');   
 }  
  
 .board\_piece {  
 background-color: #00FF00;   
 background-repeat: no-repeat;  
 background-position: center;  
 background-size: 85% 85%;  
 }  
   
   
 body {font-family: Arial, Helvetica, sans-serif;}  
form {border: 3px solid #f1f1f1;}  
  
input[type=text], input[type=password] {  
 width: 100%;  
 padding: 12px 20px;  
 margin: 8px 0;  
 display: inline-block;  
 border: 1px solid #ccc;  
 box-sizing: border-box;  
}  
  
button {  
 background-color: #04AA6D;  
 color: white;  
 padding: 14px 20px;  
 margin: 8px 0;  
 border: none;  
 cursor: pointer;  
 width: 100%;  
}  
  
button:hover {  
 opacity: 0.8;  
}  
  
.cancelbtn {  
 width: auto;  
 padding: 10px 18px;  
 background-color: #f44336;  
}  
  
.imgcontainer {  
 text-align: center;  
 margin: 24px 0 12px 0;  
}  
  
img.avatar {  
 width: 40%;  
 border-radius: 50%;  
}  
  
.container {  
 padding: 16px;  
}  
  
span.psw {  
 float: right;  
 padding-top: 16px;  
}  
  
/\* Change styles for span and cancel button on extra small screens \*/  
@media screen and (max-width: 300px) {  
 span.psw {  
 display: block;  
 float: none;  
 }  
 .cancelbtn {  
 width: 100%;  
 }  
}