

עבודת גמר בתכנון ותכנות מערכות

5 יח"ל - שאלון 883599



ChessMaster

שם התלמיד : ניצן ויינגרט

ת.ז. : 209857382

שם הבית ספר והעיר: תיכון מו"ר (מודיעין, מכבים

ורעות)

מורה: אורן גרוס

תאריך: 17.5.2022

## תוכן עניינים

2.....	<b>מסמך ייזום</b>
2.....	מבוא
3.....	תיאור מוצר
3.....	אילוצים ודרישות
3.....	תיחום הפרויקט
4.....	מצב השוק
5.....	<b>מסמך אפיון</b>
5.....	פונקציונליות המערכת
5.....	אילוצים עיקריים
5.....	פירוט והסבר על פונקציונליות המערכת
6.....	תרשים ארכיטקטורת המערכת
7.....	<b>מסמך עיצוב</b>
7.....	סביבת פיתוח
7.....	הגדרות ומושגים
8.....	הנושאים שנדרשתי ללימוד עצמי וחקר
8.....	תהליך החקר
13.....	תרשים מהלך משחק בצד הלקוח
16.....	תרשים פונקציונליות של הפרויקט והקשרים ביניהם
17.....	תרשים מחלקות של הפרויקט והקשרים ביניהם
18.....	פירוט המודלים של הפרויקט
20.....	עיצוב נתונים ופרוטוקולים
23.....	ממשק משתמש והוראות הפעלה
31.....	תרשים זרימה של מעבר בין המסכים
32.....	קודים עיקריים של מודלים והפונקציות העיקריות
37.....	רפלקציה
38.....	ביבליוגרפיה

# מסמך ייזום

## מבוא

### תיאור התחום שבו עוסקת העבודה

בתחום תורת המשחקים נשאלת השאלה: האם יש אסטרטגיה שתמיד תביא לניצחון, במשחק ללא מזל שבו כל הכלים חשופים לכלל המשתתפים? נכון להיום, לא נמצא משחק כזה שאין לו "פתרון" - או שהשחקן הראשון יכול תמיד לנצח, או שהשני, או שתמיד יהיה תיקו (לדוגמה באיקס עיגול כאשר 2 השחקנים הכי טובים תמיד התוצאה תהיה תיקו). התעלומה הגדולה היא במשחק שחמט, שבו אין רכיב של מזל וכל הכלים חשופים, אך עדיין לא נמצאה אסטרטגיה מנצחת עבורו. על מנת לגלות את ה"פתרון" לשחמט, החלו ליצור אלגוריתמים ובינה מלאכותית שבעזרתה יוכלו להבין מהו ה"פתרון" של שחמט.

### קהל יעד

- שחקנים מתחילים שינסו ללמוד אסטרטגיות מהמהלכים של האלגוריתם.
- שחקנים מנוסים שינסו לנצח את ה-AI.

### מוטיבציה לפיתוח

- עידוד של מגוון אנשים לשחק שחמט (ילדים ומבוגרים)
- משחק לתרגול בצורה ידידותית ללא צורך בדפדפן

### מה המערכת אמורה לבצע

השרת נפתח והוא ממתין למשתמשים חדשים. לקוח או מספר לקוחות, מנסים להתחבר לשרת והוא מנהל נגד כל אחד מהם משחק שחמט. המשחק הוא בין הלקוח לבין אלגוריתם הנמצא בשרת, לאחר כל מהלך של הלקוח, האפליקציה שולחת לשרת את המהלך שנבחר, האלגוריתם בוחר איזה מהלך אופטימלי לביצוע, שולח ומבצע את המהלך הנבחר.

## תיאור המוצר

### הגדרת המטרות המרכזיות של העבודה:

- תרגול משחק בדרך ידידותית ונוחה
- שמירה אוטומטית של המשחק באמצע והמשך בזמן אחר
- הגברת הביטחון למשתמשים

## אילוצים ודרישות

בעיות שהמערכת צריכה להתמודד איתם:

- השרת צריך להתמודד עם כמה לקוחות במקביל - לשם כך השתמשתי ב-Threading, על מנת לנהל כמה משחקים במקביל ולקבל הודעות מלקוחות שונים במקביל.
- חוסר תיאום בין החוקים בשרת ללקוח - הקודים לכללי המשחק שונים זה מזה עקב השימוש במנוע משחק בלקוח ולשם האחידות ולעבודה יעילה יותר, יצרתי ספרייה (Core) עם החוקים שגם הלקוח וגם השרת משתמשים בה.
- אלגוריתם פחות מדויק על מנת שתהיה תגובה מהירה - השתמשתי באלגוריתם שדורש הרבה זמן חישוב, ולכן הייתי צריך להוריד את הדיוק, על מנת שתהיה תגובה מהירה למהלכים של הלקוח.

## תיחום הפרויקט

הפרויקט מורכב משלושה חלקים ראשיים:

1. **core:** מכיוון שחוקי השחמט זהים גם בלקוח וגם בשרת, יצרתי ספרייה הנקראת core שבה יש את החוקים הבסיסיים, הספרייה נמצאת גם בשרת וגם בלקוח.
2. **שרת:** מנהל את המידע מהלקוחות ואת מאגר הנתונים ומכיל את האלגוריתם.
3. **לקוח:** מכיל את ממשק המשתמש ומאפשר לשחק לבצע מהלכים במשחק השחמט.

## מצב השוק

### מוצרים דומים בשוק:

ברחבי האינטרנט ישנם אתרים ותוכנות רבות של משחקי שחמט ושל אלגוריתמים עבורם, ועניין אותי מאוד לחקור כיצד הם עובדים ולכן בחרתי בנושא זהה. משחק השחמט נחשב למשחק מורכב מאוד ורציתי להנגיש את המשחק לקהלים רחבים יותר ולהנגיש את זה בצורה ידידותית. מוצר יהיה דומה ל-[chess.com](https://chess.com).



## מסמך אפיון

### פונקציונאליות המערכת

- ממשק לקוח
- אפשרות ללחוץ על כלי משחק, לראות את המהלכים האפשריים שלו ולבצע מהלך
- התחברות למערכת - יצירת קשר בין השרת לכל אחד מהלקוחות
- הרשמה והתחברות למאגר משתמשים (database)
- אשתמש בשיטת MinMax ל-AI על מנת למצוא את המהלך האופטימלי

### אילוצים עיקריים

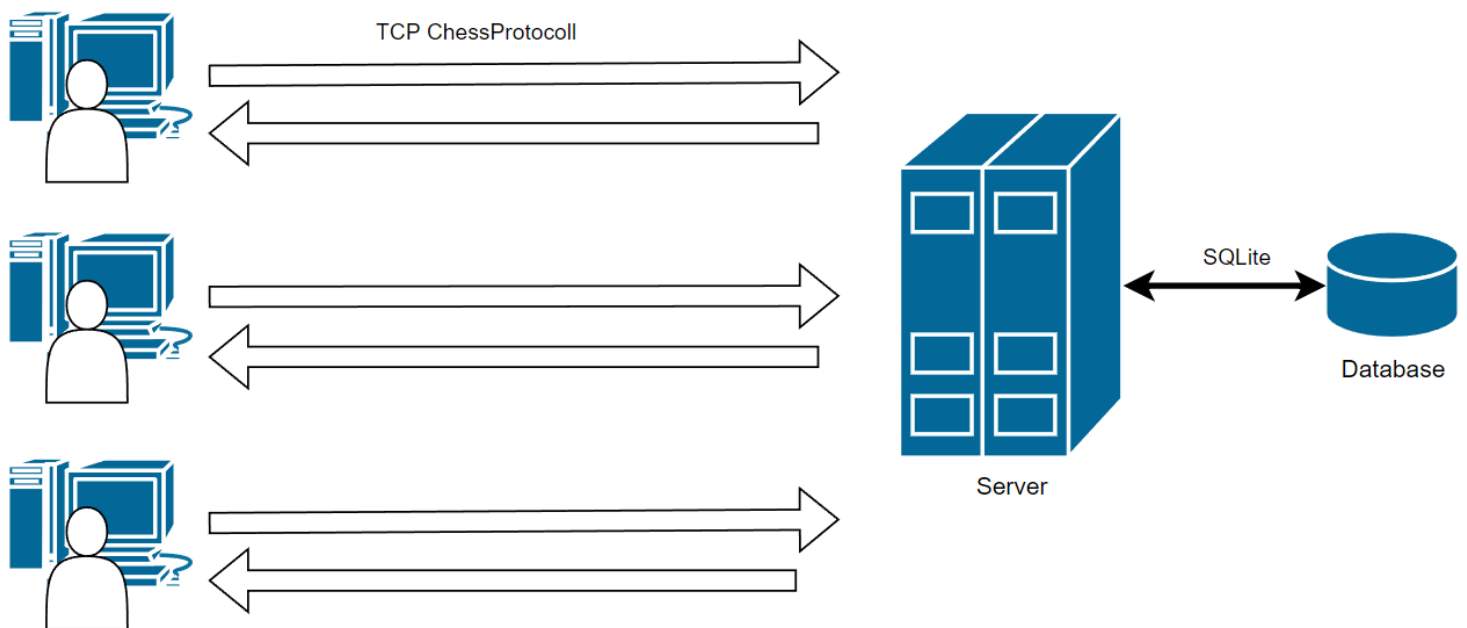
- התוכנה רצה על מערכת הפעלה windows 10.
- המערכת דורשת חיבור לרשת מקומית עם מספר מחשבים.
- האלגוריתם יהיה פשוט יותר ולכן פחות אפקטיבי - (ככל שהאלגוריתם רואה יותר קדימה כך מספר האופציות עולה בצורה אקספוננציאלית, לכן יש הגבלת עומק של 3 צעדים קדימה שהוא ינתח).

### פירוט והסבר על פונקציונליות המערכת

- ממשק לקוח - הממשק נבנה ב-Unity על מנת לאפשר מסכים הנוחים לשימוש הלקוח, כגון: מסך כניסה, הרשמה ומסך המשחק הראשי.
- אפשרות ללחוץ על כלי משחק, לראות את המהלכים האפשריים שלו ולבצע מהלך - בדיקה בספרייה לגבי המהלכים האפשריים ויצירה של ריבועים וויזואליים, על מנת להקל על המשתמש. בנוסף, הוספת אנימציה של תזוזת השחקן.
- התחברות למערכת - יצירת קשר בין השרת לכל אחד מהלקוחות - במסך הפתיחה יש מקום לכתיבת ה-IP של השרת על מנת להתחבר אליו.
- הרשמה והתחברות למאגר משתמשים (database) - במאגר הנתונים נשמרים שם המשתמש, הסיסמה והלוח האחרון שהמשתמש שיחק, בשרת החיבור ל-database בוא

- אשתמש בשיטת MinMax ל-AI על מנת למצוא את המהלך האופטימלי - בשרת יש אלגוריתם שמעריך את המהלך האופטימלי לביצוע בכך, שהוא מייצר את האפשרויות הבאות ומנתח כל אפשרות ובוחר את המהלך המיטבי. (זהו העומק של האלגוריתם, כמה צעדים קדימה הוא מנתח).

## תרשים ארכיטקטורת המערכת



# מסמך עיצוב

## סביבת פיתוח

הפרוייקט נעשה בשפת C# ו-SQL בסביבת Visual Studio 2022 ובמנוע Unity. הצגת בסיס הנתונים באמצעות DBeaver.

הכלים והספריות בהם השתמשתי בהם:

- ◆ Socket - ספרייה לשימוש בתקשורת בין מחשבים שונים, השתמשתי בו על מנת לבסס תקשורת TCP.
- ◆ Time - השתמשתי בספרייה Time על מנת לבדוק את המהירות של האלגוריתם ומתי לנתק לקוח שנמצא זמן רב מדי מחובר.
- ◆ Threading - ספרייה המאפשרת שימוש ב-Threads.
- ◆ UnityEngine - ספרייה של המנוע הגרפי שמכיל עבודה עם אובייקטים, סצנות ו-scripts.
- ◆ Sqlite - ספרייה שמאפשרת תקשורת וגישה לממסד הנתונים
- ◆ Cryptography - ספרייה המשמשת להצפנה חד כיוונית (Hash) לסיסמאות, על מנת שלא יגלו אותם.

## הגדרות ומושגים

- Unity - מנוע גרפי המאפשר ליצור משחקים ואפליקציות בשפת C# עבור מגוון פלטפורמות.
- Scripts - במנוע הגרפי נוצרים אובייקטים שאליהם משייכים קובצי קוד הנקראים scripts.
- Scenes - המנוע מאפשר מעבר בין סצנות לדוגמא בין תפריט ראשי לסצנה של המשחק עצמו.
- MinMax Algorithm - האלגוריתם המשמש לבחירת המהלך האופטימלי הבא.
- שרת - מחשב המריץ את תוכנת השרת ושאר המחשבים מתחברים אליו (הלקוחות)
- לקוח - הלקוח מהווה את ממשק המשתמש, הוא מנהל את התקשורת בין המשתמש לשרת.
- Database - אמצעי המשמש לאחסון מסודר של נתונים במחשב, לשם איחזורם ועיבודם. הגישה לבסיס נתונים נעשית באמצעות תוכנה ייעודית.
- Thread - מערכת לניהול פקודות שרצות במקביל, בשרת הקוד שמנהל לקוח בודד, רץ במקביל בכמה Threads כדי לקבל הודעות מכולם.



- עומק (Depth) - באלגוריתם MinMax זהו מספר הצעדים שהוא מסתכל קדימה, עומק של 5 מייצר את כל הלוחות 5 צעדים קדימה ומנתח אותם ובוחר איזה מהלך כדאי לביצוע.

## הנושאים שנדרשתי ללימוד עצמי וחקר

- שימוש במנוע המשחק Unity - ניהול האובייקטים שעל המסך ושימוש בסצנות בשביל מסכים שונים (התחברות, משחק, ניצחון)
- שימוש ב-sockets ב-C# בשביל תקשורת בין הלקוח והשרת.
- שימוש ב-Threading כדי שהשרת ינהל מספר משתמשים במקביל.
- MinMax Algorithm - הדרך שבה המחשב יודע איזה מהלך לבצע.
- עבודה עם Database

## תהליך החקר

### שימוש במנוע המשחק Unity:

בתחילה ראיתי איך עשו פרויקטים אחרים ב-Unity ולמדתי את הדברים הבסיסיים, איך ליצור אובייקטים ואיך לסגור אותם, התחלתי לכתוב את צד הלקוח ובמהלך הכתיבה בדקתי באינטרנט כיצד לעבוד עם קלט מהעכבר ועם החלפת סצנות. השוני בין עבודה רק עם קוד לבין עבודה עם מנוע גרפי שלא הכרתי היה הקושי העיקרי בפרויקט אך בשביל ליצור משחק הייתי מוכרח להשתמש בו.

### שימוש ב-sockets ב-C# ליצור חיבור בין השרת:

לאחר חיפוש ממושך באינטרנט יצרתי שרת ולקוח בקובץ אחר על מנת לבחון את אמינות של הספרייה ולאחר מכן שילבתי בקובץ גם multiThreading

## AI לשחמט:

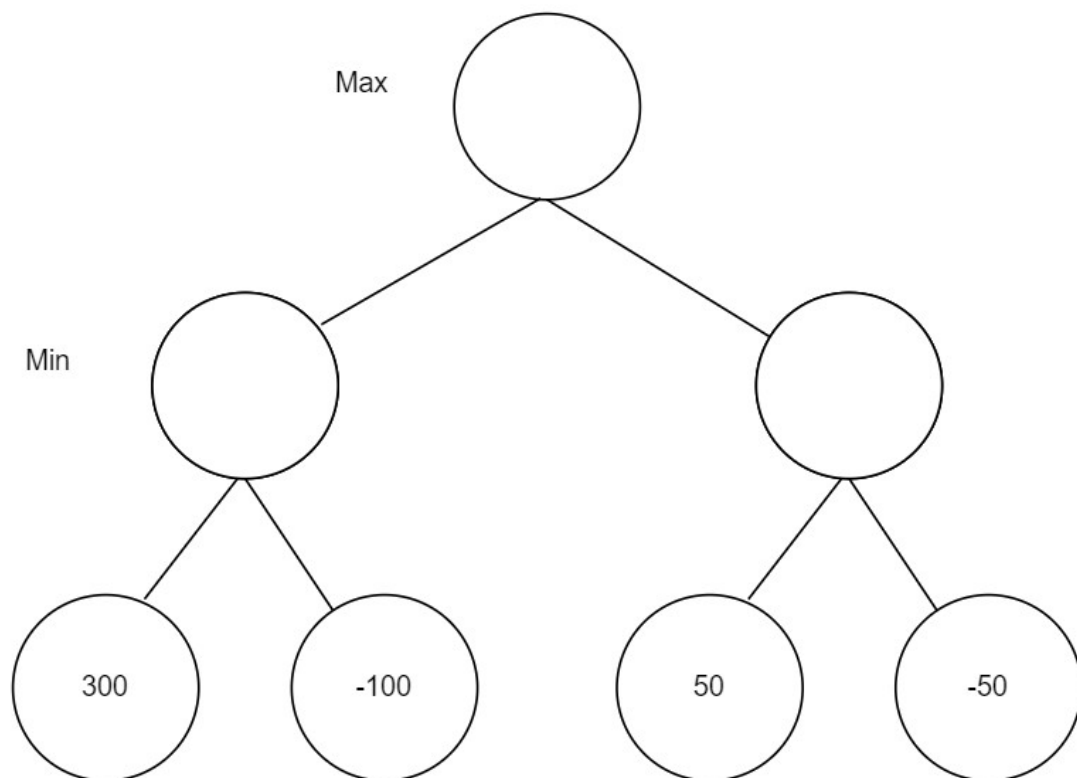
במסגרת הפרוייקט הייתי צריך ללמוד מספר נושאים:

### :MinMax Algorithm

ישנם אלגוריתמים רבים על מנת לבחון את המצבים האפשריים, האלגוריתם שבחרתי הוא אלגוריתם MinMax.

האלגוריתם מסתכל מספר קבוע של צעדים קדימה במשחק, הוא בוחן את כל האפשרויות ובכך יכול לבחור את המהלך האופטימלי לעשות בצעד הקרוב, למספר הצעדים שהוא בוחן קדימה נקרא עומק האלגוריתם (Depth).

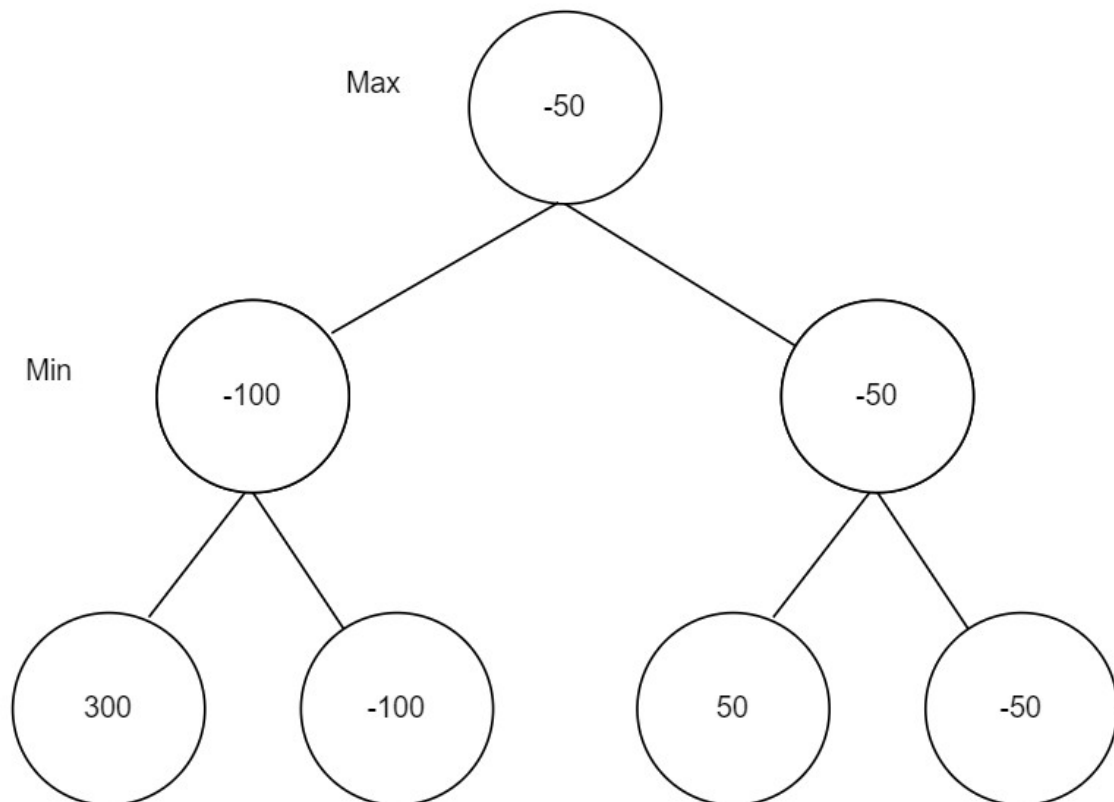
נניח שיש לכל מצב רק 2 מהלכים אפשריים כמו בדוגמא שלהלן:



אם נסתכל לעומק של 2, לנו יש בחירה בין העץ השמאלי לימני ומהאפשרויות אנו נבחר את האפשרות ששווה לנו יותר נקודות, אך אנו מסתמכים על כך שהשחקן היריב יבחר באפשרות

הפחות טובה בשבילנו ולכן בשורה השנייה נבחר את הניקוד המינימלי (המהלך יבחר ע"י השחקן היריב מכיוון שזהו תורו), לכן אלגוריתם זה נקרא MinMax.

בבחירה אנו מתחילים מהענפים התחתונים, אנו נבחר לכל אחד משני העצים את האפשרות הנמוכה יותר מכיוון שמהלך זה הכי פוגע בנו, אם היריב יבחר מהלך אחר אנו נהיה במצב אף יותר טוב ולכן זו אינה בעיה.



לאחר שקיבלנו במהלך הימני -50 ובשמאלי -100 אנו נבחר את האפשרות ששווה לנו יותר נקודות ולכן נבחר במהלך הימני.

### עומק

העומק הוא מספר הצעדים שהאלגוריתם בודק קדימה, מספר הצעדים האפשריים עולה בקצב אקספוננציאלי (הוא מגיע למיליארדי אפשרויות בעומק 5 בשחמט) והוא דורש יותר זמן חישוב ולכן הגבלתי לעומק של 3.

## FEN

FEN הוא ראשי תיבות של Forsyth-Edwards Notation זהו פורמט טקסט שהומצא על מנת לתאר מצב של כל כלי המשחק בלוח בצורה קצרה וברורה, אני משתמש בפורמט על מנת להעביר בין פורמטים אחרים של לוחות שחמט בין החלקים השונים.

הפורמט עובד כך:

לכל כלי משחק יש סימן:

רץ-b, מלך-k, מלכה-q וכו'...

לכל אחד מהתווים כותבים באות קטנה כדי לסמן כלי משחק שחור ובאות גדולה בשביל לבן. מתחילים משמאל למעלה ומוסיפים את הסימן r לצריח, לאחר מכן סופרים את מספר הרווחים עד לכלי המשחק הבא עד שמגיעים לסוף השורה, בסוף השורה מוסיפים '/' וכך ממשיכים לכל שורה עד שמתקבל טקסט המתאר את כל הלוח:

r1b1k1nr/p2p1pNp/n2B4/1p1NP2P/6P1/3P1Q2/P1P1K3/q5b1

8									r1b1k1nr/
7									p2p1pNp/
6									n2B4/
5									1p1NP2P/
4									6P1/
3									3P1Q2/
2									P1P1K3/
1									q5b1
	a	b	c	d	e	f	g	h	

## Board Evaluation – הערכת הלוח

האלגוריתם ה-MinMax בוחר את המהלך הכדאי כשנתון את הניקוד של הלוחות, Board Evaluation הוא הדרך שבה מעריכים לוח נתון.

אני השתמשתי בהערכה שבקישור: [Evaluation\\_Function](#)

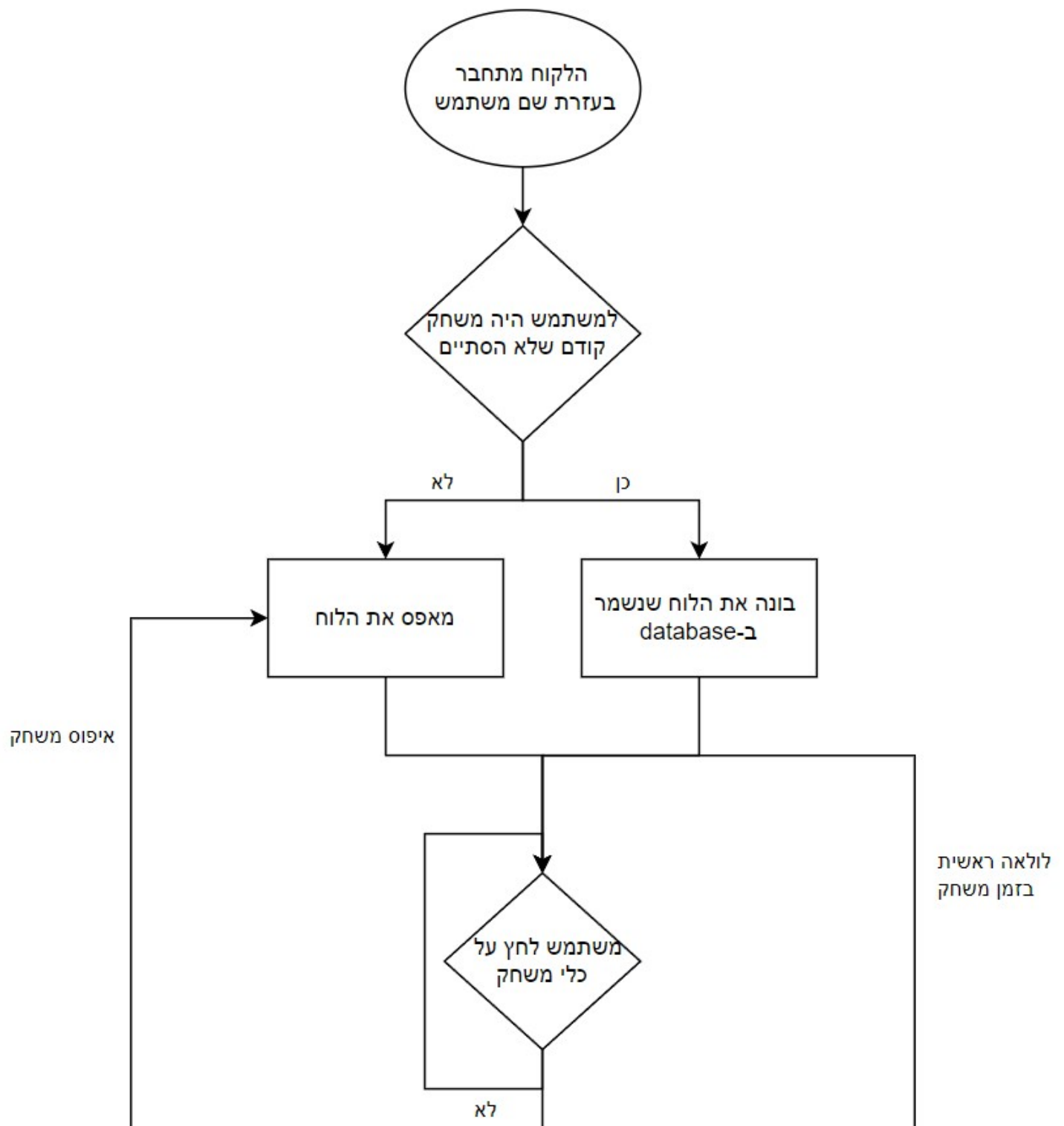
בהערכה זו נותנים לכל כלי משחק ערך משלו ולפי המיקום שלו נותנים לו נקודות בונים ככל שהוא במיקום יותר טוב, בדוגמא יש לוח ניקוד לחייל:

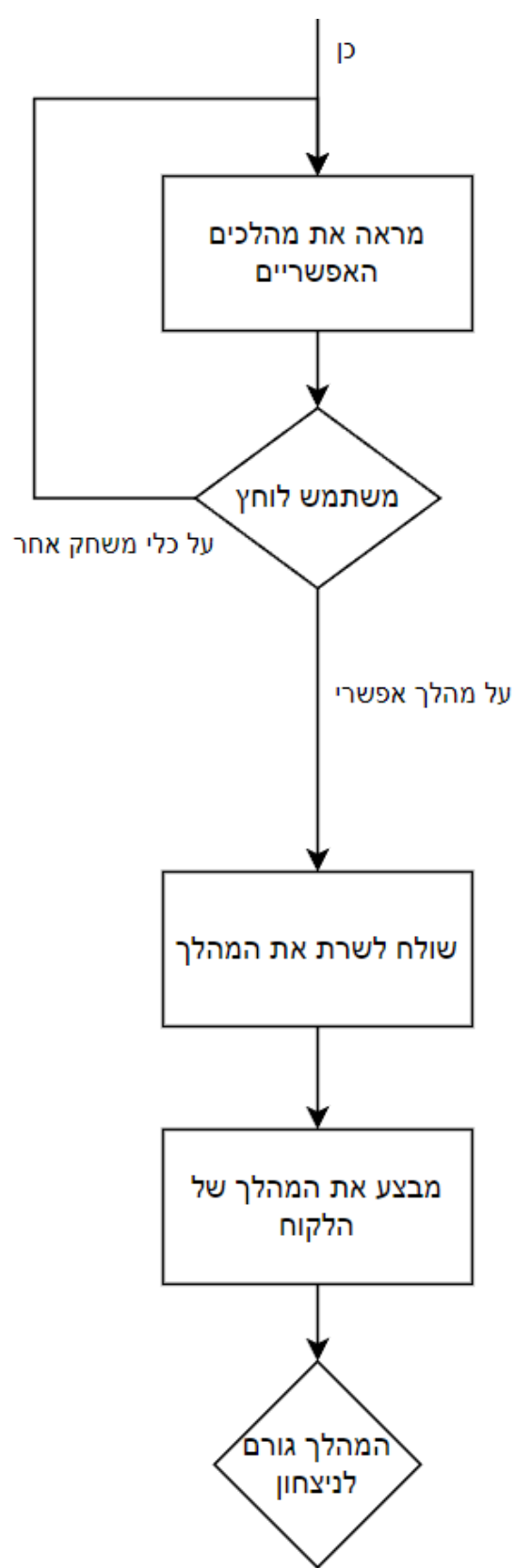
```
protected static sbyte[, ] pawn =
{
{ 0, 0, 0, 0, 0, 0, 0, 0 },
{ 50, 50, 50, 50, 50, 50, 50, 50 },
{ 10, 10, 10, 10, 10, 10, 10, 10 },
{ 10, 10, 20, 30, 30, 20, 10, 10 },
{ 5, 5, 10, 25, 25, 10, 5, 5 },
{ 0, 0, 0, 20, 20, 0, 0, 0 },
{ 5, -5, -10, 0, 0, -10, -5, 5 },
{ 5, 10, 10, -20, -20, 10, 10, 5 },
{ 0, 0, 0, 0, 0, 0, 0, 0 }
};
```

לדוגמה חייל שווה 100 נקודות ואם הוא נמצא בשורה השנייה מלמעלה הוא מקבל בונים של 50 נקודות.

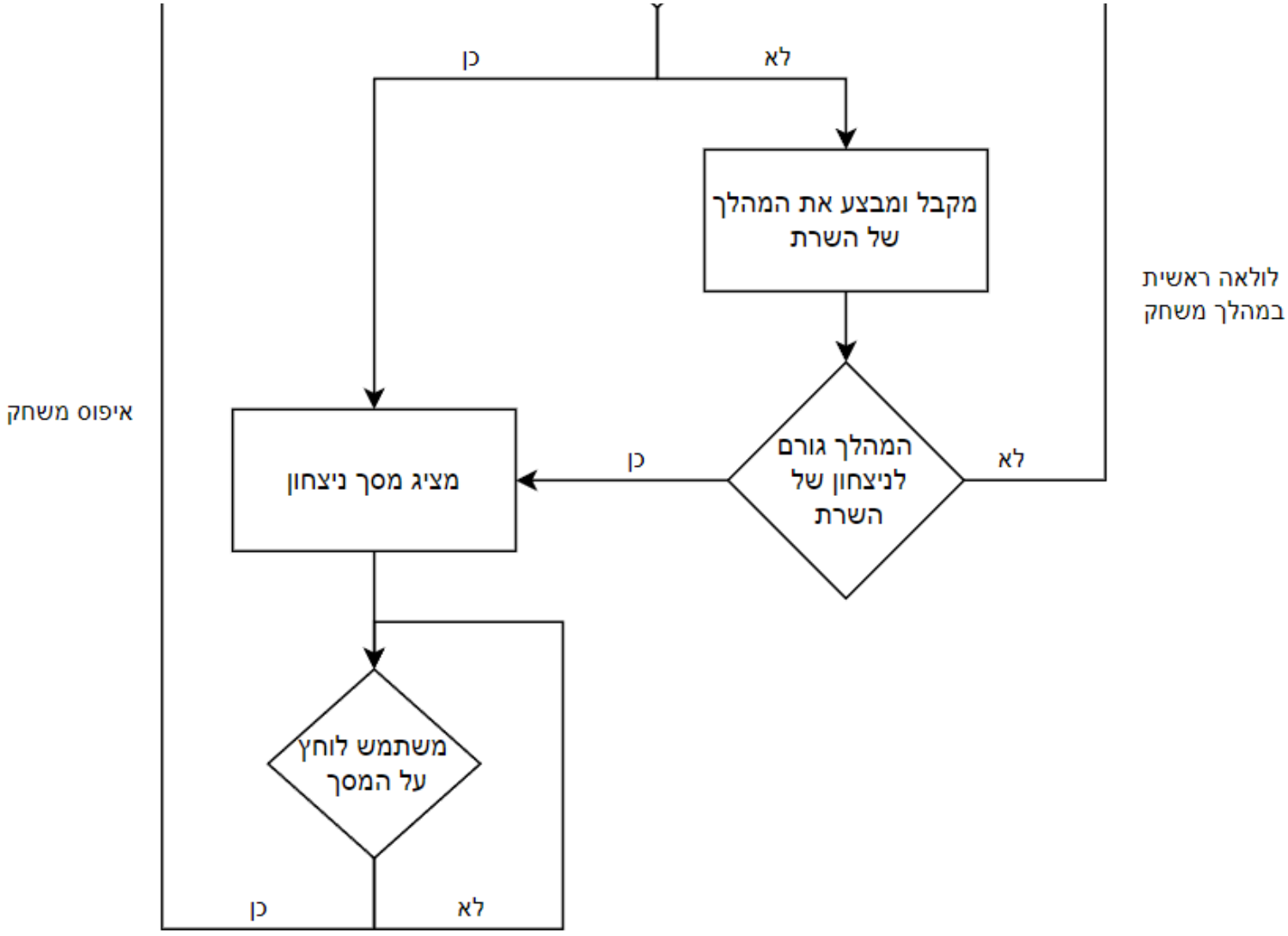
בחישוב הלוח סוכמים את הניקוד של כל החיילים בצבע מסוים ומחסירים את הניקוד של החיילים בצבע אחר.

## תרשים מהלך משחק בצד הלקוח



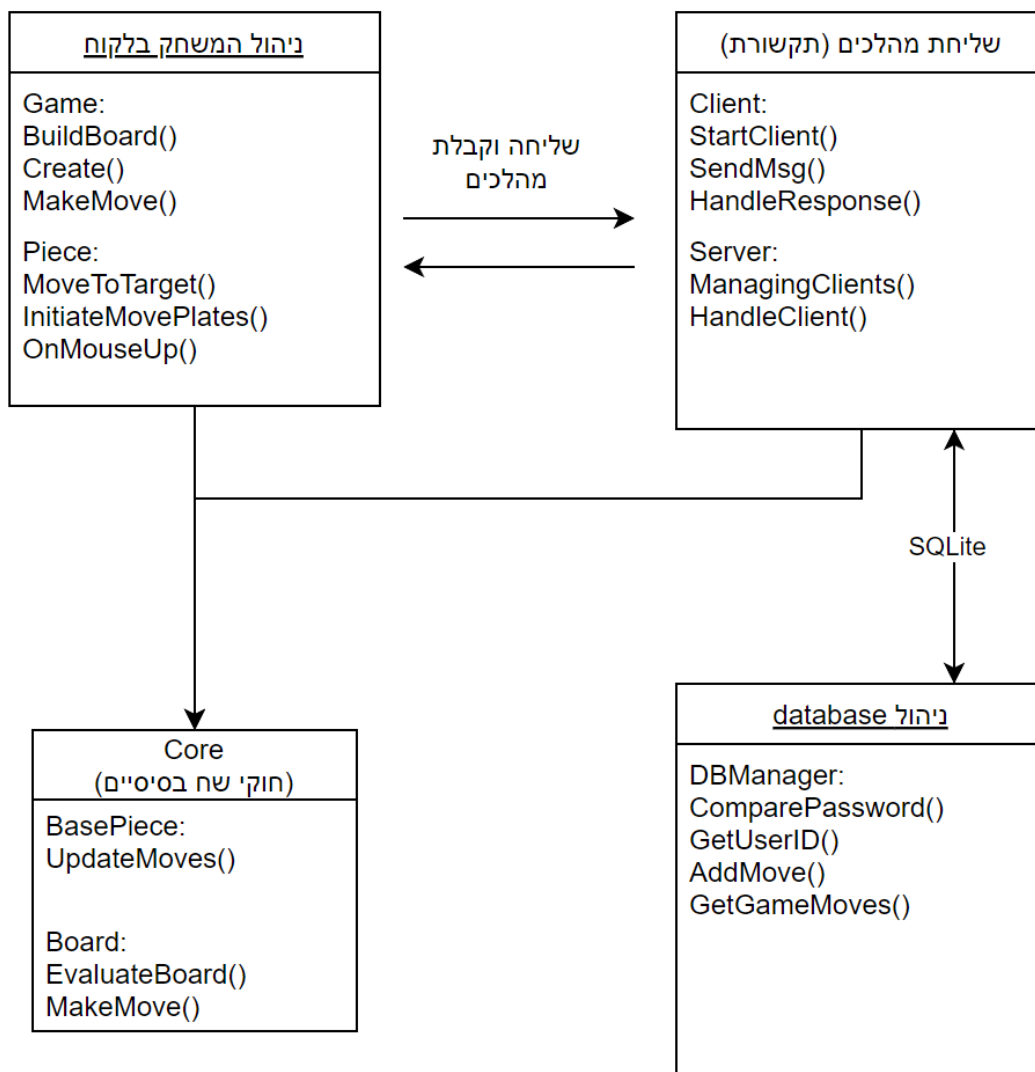


לולאה ראשית  
במהלך משחק

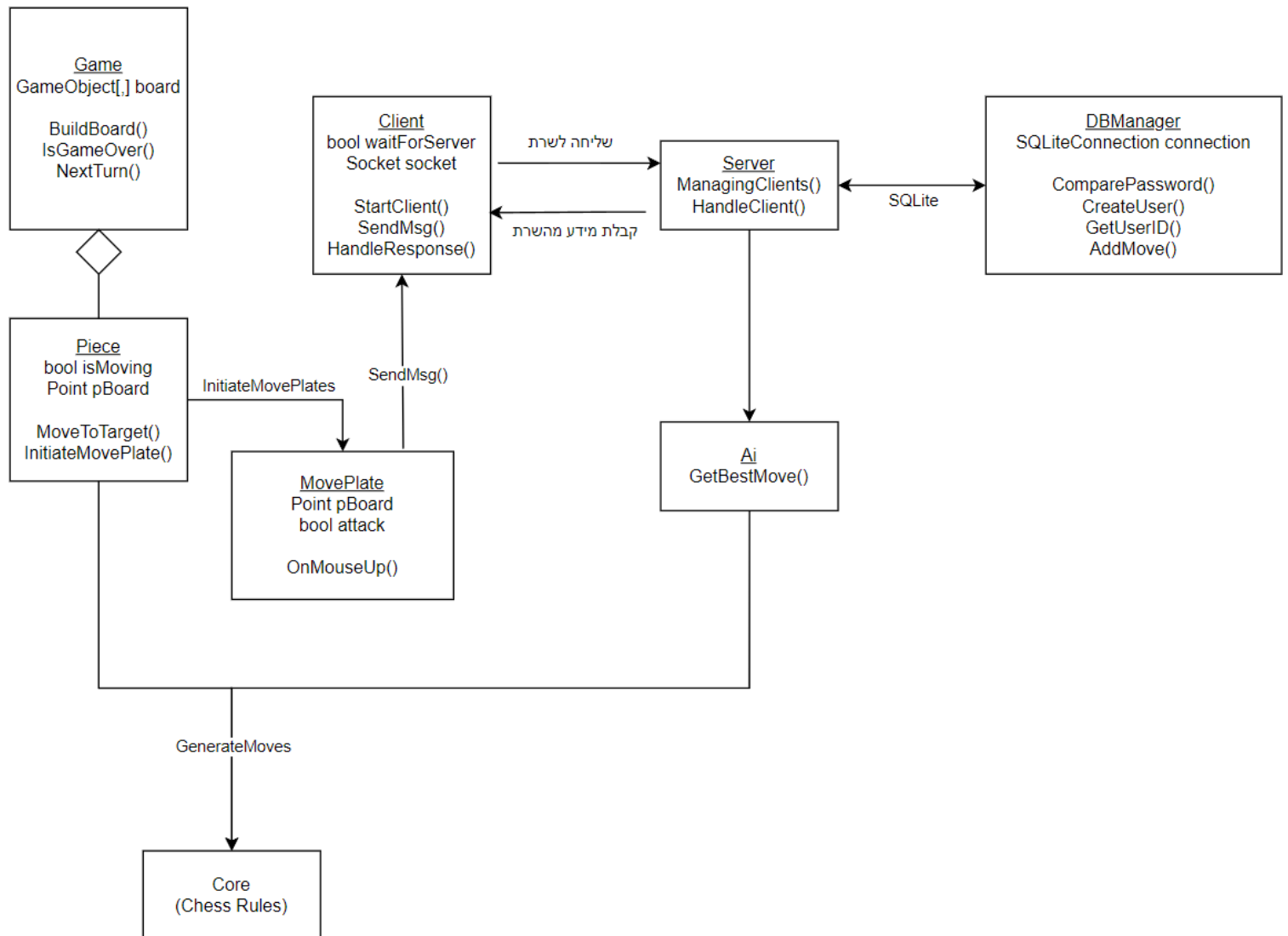




## תרשים פונקציונליות של הפרויקט והקשרים ביניהם



## תרשים מחלקות של הפרויקט והקשרים ביניהם



## פירוט המודלים של הפרויקט

### Core

**BasePiece** – מחלקה לכל כלי משחק, מחלקה זו פשוטה יותר מ-Piece מכיוון שהיא צריכה לעבוד מהר כשהיא מייצרת מיליוני אפשרויות על מנת לבחור את האופציה המיטבית. המחלקה מכיל את הלוגיקה לאפשרויות למהלך הבא של אותו שחקן.

Board – לוח מצומצם יותר על מנת שיעבוד מהר יותר.

Point – מחלקה פשוטה של  $x$  ו- $y$ .

Move – מחלקה שמכילה נקודת התחלה ונקודת סיום.

### Server

#### ManagingClients

הפעולה הראשית שפותחת שרת ומחכה למשתמשים להתחבר, לאחר החיבור היא משייכת לכל אחד Thread socket ומפעילה את הפעולה HandleClient לכל Thread

#### HandleClient

פעולה שמנהלת את ההודעות ללקוח מסויים, היא מקבלת ממנו מידע ומנתחת אותה ושולחת הודעה מתאימה בחזרה.

**Client** – מנהל את התקשורת מול השרת, שולח ומקבל מידע ומנתח את המידע שהגיע.

**-Game** מחלקה ראשית שמיועדת לניהול המשחק את הלקוח, המחלקה מכילה את יצירת הלוח הוויזואלי, ביצוע של מהלכים חדשים ויצירה של כלי משחק חדשים על הלוח.

**MovePlate** – מחלקה שמשוייכת למהלכים האופציונאליים כאשר הלקוח בוחר כלי משחק מסויים, אם העכבר נלחץ על MovePlate המהלך יתבצע וישלח לשרת בעזרת Client.cs.

**Piece** – מחלקה השייכת לכל כלי משחק בצד הלקוח, המחלקה מכילה פעולה שנוגעות לוויזואליות של כלי המשחק, כמו יצירת מהלכים אפשריים, התקדמות לכיוון היעד (כאשר נבחר מהלך).

## **DBManager**

**ComparePassword** – מקבל את המיקום של המשתמש וסיסמה שהלקוח הכניס, הפעולה מחזירה עם הסיסמה שווה לסיסמה שב-Database.

**GetUserID** – הפעולה מקבלת שם משתמש וסיסמה ואם הסיסמה נכונה יוחזר המיקום ב database, במידה והסיסמה לא נכונה יוחזר 1- ואם שם המשתמש לא קיים ב-database יוחזר 0.

**AddMove** – מקבל מהלך ומיקום של המשתמש, הפעולה מוסיפה את המהלך לרשימת המהלכים ב-databases.

**GetGameMoves** – הפעולה מחזירה את רשימת המהלכים של משתמש מסוים.

## עיצוב נתונים ופרוטוקולים

### פירוט הפרוטוקולים - ChessProtocol

1. לאחר שהלקוח מבצע מהלך, הלקוח שולח את המהלך (Move) בפורמט הבא:

**1\_x1, y1\_x2, y2,PType**

1 מסמן את סוג ההודעה - זוהי הודעה של מהלך שהתבצע על ידי הלקוח

x1, y1 זו נקודת ההתחלה של האובייקט ו-x2,y2 היא נקודת הסיום של האובייקט.

PType - תו בודד שמסמן את כלי המשחק שנאכל באותו המהלך, הסימונים לחיילים הם:

n - knight

k - king

q - queen

r - rook

b - bishop

p - pawn

' - אף כלי משחק לא נאכל בתור זה

מהלך זה גם מוחזר מהשרת לאחר שהלקוח שלח את המהלך שלו, ההודעה באותו הפורמט

2. התחברות - הלקוח מנסה להתחבר למשתמש קיים בעזרת ההודעה הבאה:

**2\_username\_password**

אם הנתונים נכונים השרת יחזיר:

**9\_FEN**

FEN - פורמט להעברת לוח של שחמט בעזרת שורת טקסט, השרת מחזיר את הלוח של המשחק הקודם שהמשתמש לא סיים, ובכך יוכל להמשיך מאותו המשחק.

אם הנתונים לא נכונים תוחזר ההודעה הבאה:

**10\_wrong user or password**

3. יצירת משתמש - הלקוח שולח את ההודעה כאשר הוא מנסה ליצור משתמש חדש:

**3\_username\_password**

אם הנתונים נכונים השרת יישלח את ההודעה הבאה:

**7\_ok**

אם הנתונים לא נכונים תשלח מהשרת ההודעה הבאה:

**8\_user taken**

במידה והמידע לא בפורמט מתאים השרת שולח את ההודעה הבאה:

**11\_msg not in format**

## מבני נתונים:

בפרייקט השתמשתי ב-databases מהספרייה SQLite:

בשרת נמצאת הטבלה chessUsers שבא נשמרים הנתונים הבאים לכל משתמש:

- שם משתמש
- Hash של הסיסמה
- טקסט עם כל המהלכים של המשחק הנוכחי
- טקסט עם כל המהלכים של המשחק הקודם

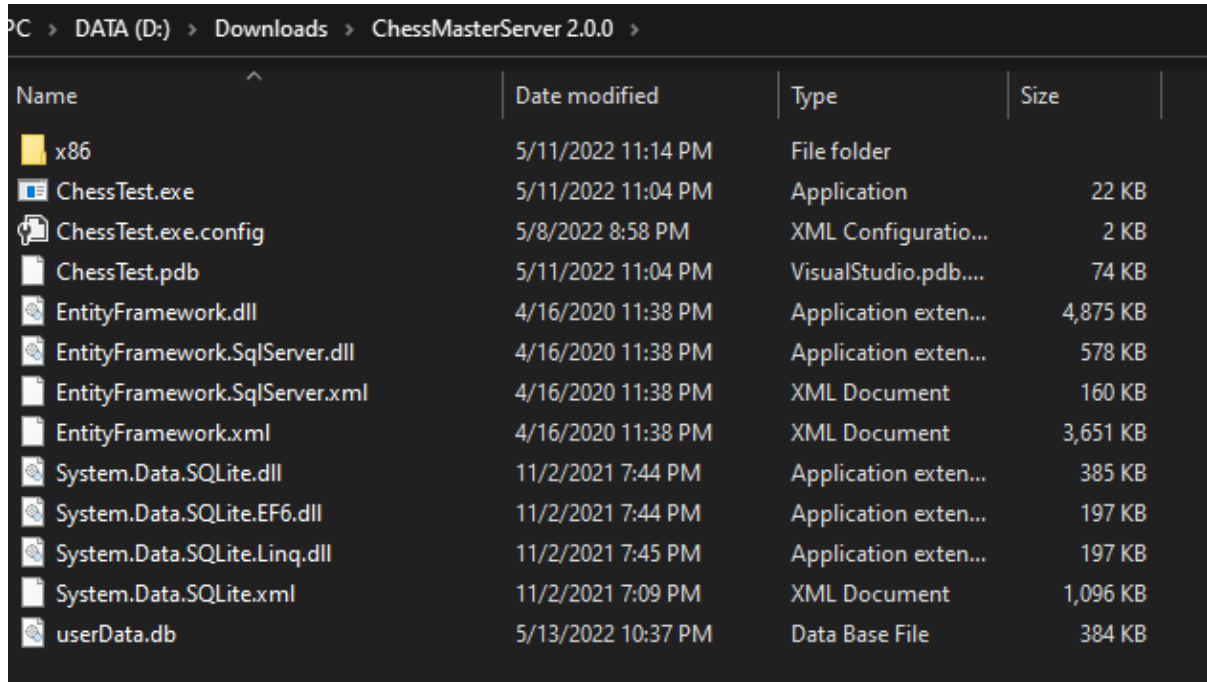
השימוש ב-databases נבחר מכיוון שרציתי לאסוף את הנתונים על כל משתמש בצורה אחידה, נוחה לשימוש והכי חשוב מסודרת בטבלה שאיתה נוח לבדוק תקלות ולראות את נתוני הלקוח.

id	username	password	currentGameMoves	lastGameMoves
1	1937	897a5faa1a4c4773d4033e2c8e2		[NULL]
2	2	d4735e3a265e16eee03f59718b9	6, 0_7, 2;3, 6_3, 5;	[NULL]
3	a	ca978112ca1bbdcafac231b39a2	6, 1_6, 2;3, 6_3, 5;	[NULL]
4	b	3e23e8160039594a33894f6564e	1, 0_2, 2;3, 6_3, 5;	[NULL]
5	3	4e07408562bedb8b60ce05c1de	6, 0_7, 2;3, 6_3, 5;	[NULL]
6	1	6b86b273ff34fce19d6b804eff5a		[NULL]
7	4	4b227777d4dd1fc61c6f884f486	6, 1_6, 3;3, 6_3, 5;	[NULL]
8	q	8e35c2cd3bf6641bdb0e2050b7f	3, 1_3, 3;3, 6_3, 5;4, 1_4, 3;2, 7_3,	[NULL]
9	12	6b51d431df5d7f141cbececcf79	2, 1_2, 2;3, 6_3, 5;2, 2_2, 3;2, 7_3,	[NULL]
10	23	535fa30d7e25dd8a49f15367797	6, 0_7, 2;3, 6_3, 5;7, 2_5, 3;2, 7_3,	[NULL]
11	2323	61503690505f84b144e6ac89124		[NULL]
12	123	a665a45920422f9d417e4867efd		[NULL]
13	2w	7c1c5ee5c6a4dec209832011c36	6, 0_7, 2;3, 6_3, 5;7, 2_6, 4;2, 7_3,	[NULL]
14	w2	06f8faea3b5f697691b6d063a07f		[NULL]

## ממשק משתמש והוראות הפעלה

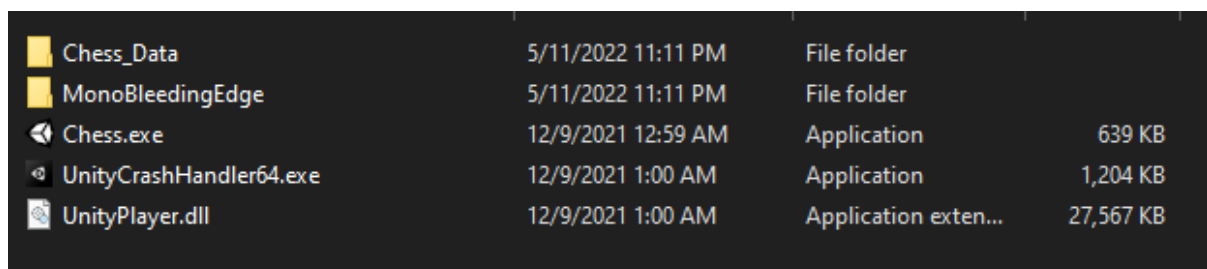
למערכת יש 2 קטעים שצריך להריץ:

- את השרת שפותח cmd באחד מן המחשבים באותה רשת מקומית
- במחשב אחד או מספר מחשבים לפתוח את תוכנת הלקוח



Name	Date modified	Type	Size
x86	5/11/2022 11:14 PM	File folder	
ChessTest.exe	5/11/2022 11:04 PM	Application	22 KB
ChessTest.exe.config	5/8/2022 8:58 PM	XML Configuratio...	2 KB
ChessTest.pdb	5/11/2022 11:04 PM	VisualStudio.pdb....	74 KB
EntityFramework.dll	4/16/2020 11:38 PM	Application exten...	4,875 KB
EntityFramework.SqlServer.dll	4/16/2020 11:38 PM	Application exten...	578 KB
EntityFramework.SqlServer.xml	4/16/2020 11:38 PM	XML Document	160 KB
EntityFramework.xml	4/16/2020 11:38 PM	XML Document	3,651 KB
System.Data.SQLite.dll	11/2/2021 7:44 PM	Application exten...	385 KB
System.Data.SQLite.EF6.dll	11/2/2021 7:44 PM	Application exten...	197 KB
System.Data.SQLite.Linq.dll	11/2/2021 7:45 PM	Application exten...	197 KB
System.Data.SQLite.xml	11/2/2021 7:09 PM	XML Document	1,096 KB
userData.db	5/13/2022 10:37 PM	Data Base File	384 KB

כך נראים קובצי השרת, המשתמש צריך לפתוח את ChessTest.exe

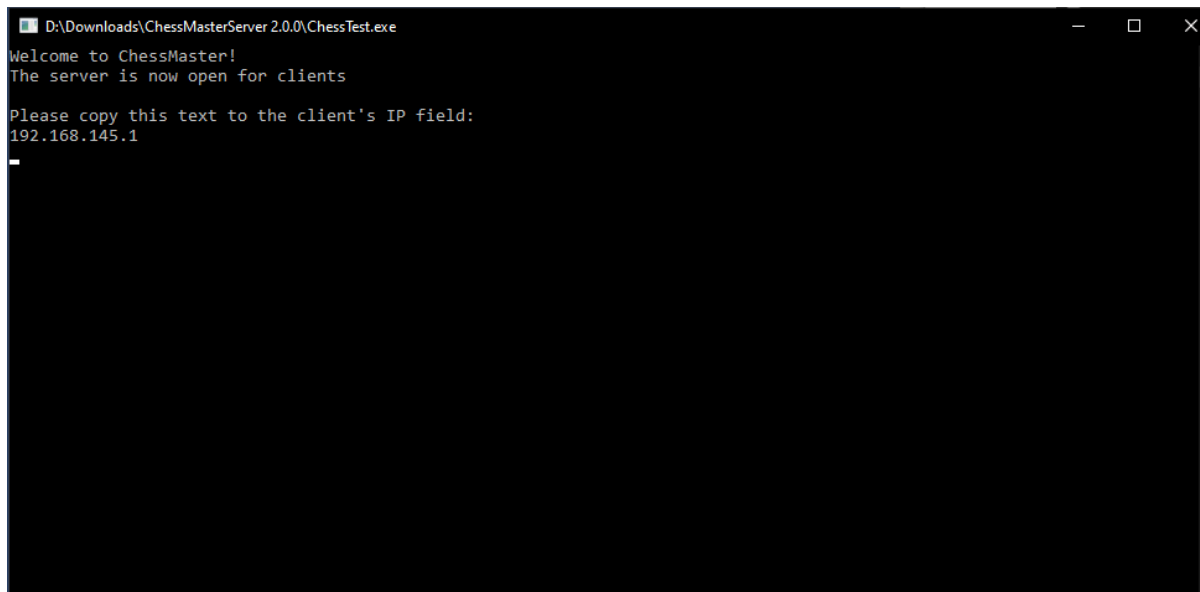


Name	Date modified	Type	Size
Chess_Data	5/11/2022 11:11 PM	File folder	
MonoBleedingEdge	5/11/2022 11:11 PM	File folder	
Chess.exe	12/9/2021 12:59 AM	Application	639 KB
UnityCrashHandler64.exe	12/9/2021 1:00 AM	Application	1,204 KB
UnityPlayer.dll	12/9/2021 1:00 AM	Application exten...	27,567 KB

כך נראית תיקיית המשחק הראשי, המשתמש צריך לפתוח את Chess.exe



## מסך השרת:



```
D:\Downloads\ChessMasterServer 2.0.0\ChessTest.exe
Welcome to ChessMaster!
The server is now open for clients

Please copy this text to the client's IP field:
192.168.145.1
```

במסך זה השרת מראה תחילה את ה-IP שלו שדרוש להעביר ללקוחות. בהמשך כאשר לקוחות מתחברים לשרת במסך זה ניתן לראות את ההודעות שעוברות לשרת.

הוראות הפעלה - למסך השרת אין שום קלט נדרש, זהו מסך המיועד לניטור.

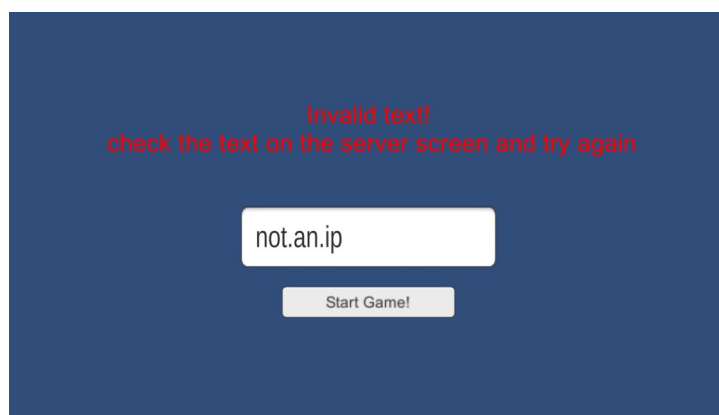
### מסך פתיחה:



### הוראות הפעלה למסך הפתיחה:

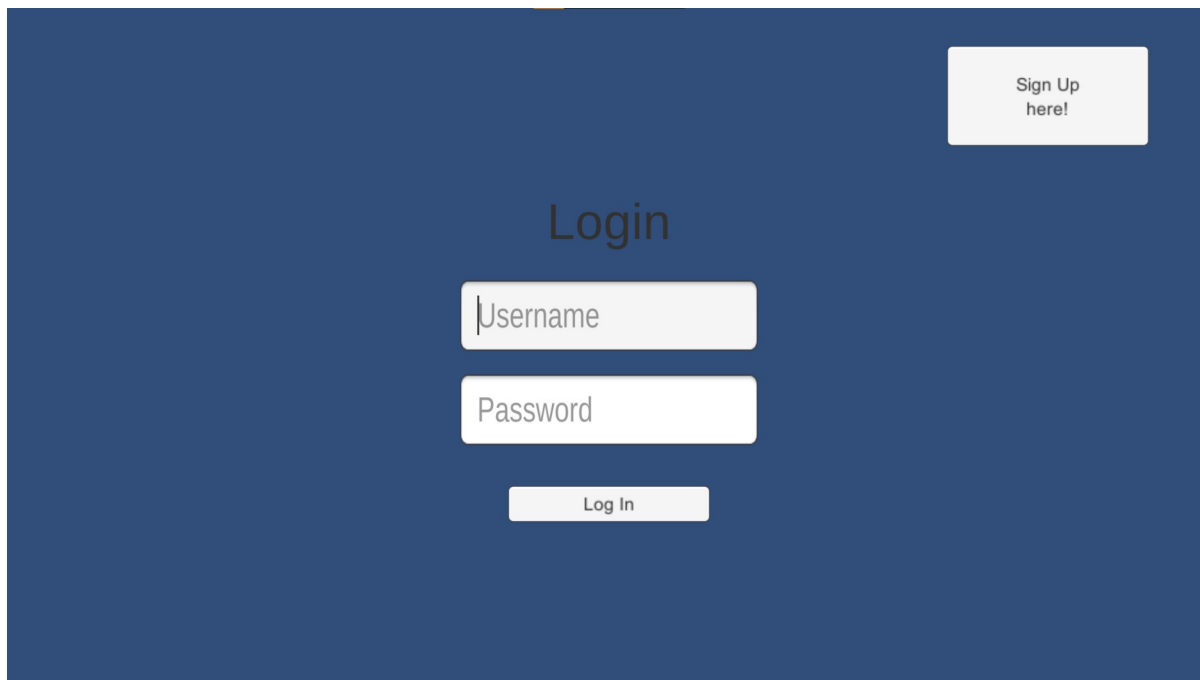
במסך השרת יש IP שהשרת אומר לך להכניס ללקוח, המשתמש צריך להכניס את ה-IP וללחוץ על Enter או על הכפתור "start game" על מנת לעבור למסך ה-Log in.

במקרה שממלאים טקסט לא תקין או IP שבו לא פתוח השרת יופיע המסך הבא:



אם ה-IP תקין הלקוח יוצר חיבור אל השרת ועובר אל מסך ההתחברות.

## מסך התחברות



### הוראות הפעלה למסך ההתחברות:

במסך זה אפשר להיכנס למשתמש קיים או לעבור למסך של יצירת המשתמש,

הכפתור "Sign up here" מעביר למסך יצירת המשתמש

Username - הכנסת שם המשתמש

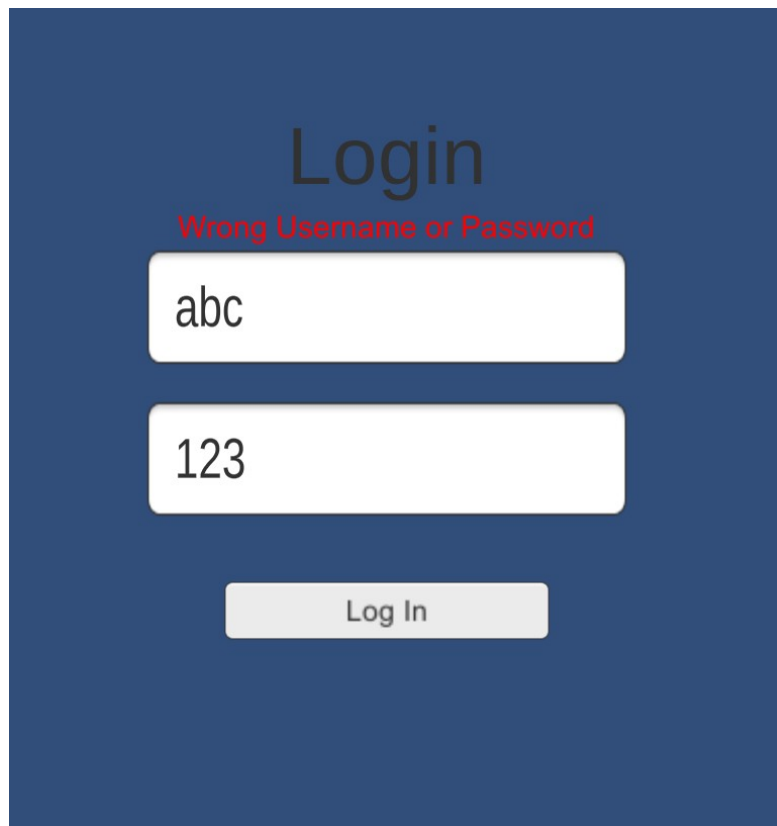
Password - הכנסת הסיסמא

Login - מתחבר למשתמש

הלקוח שולח לשרת את הנתונים והוא משווה עם מאגר הנתונים.

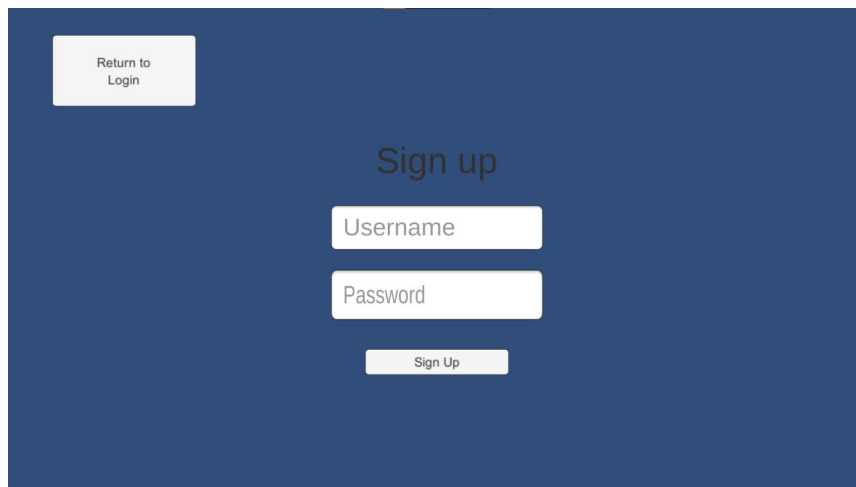
(הסיסמה עוברת הצפנה חד כיוונית על מנת שלא יהיה דרך לבעל השרת או לתוכנה בצד שלישי לגלות את הסיסמה).

במידה והמשתמש לא קיים או שהסיסמה אינה מתאימה יופיע המסך הבא:



A login form on a dark blue background. At the top, the word "Login" is written in a large, dark font. Below it, the text "Wrong Username or Password" is displayed in red. There are two white input fields: the first contains the text "abc" and the second contains "123". Below the input fields is a light gray button with the text "Log In".

## מסך יצירת משתמש



Return to Login

### Sign up

Username

Password

Sign Up

### הוראות הפעלה למסך יצירת משתמש:

במסך זה נועד ליצירת משתמש חדש

הכפתור "Return to Login" מחזיר למסך ההתחברות.

Username - הכנסת שם המשתמש חדש

Password - הכנסת סיסמה

Sign Up - מעביר לשרת את הנתונים

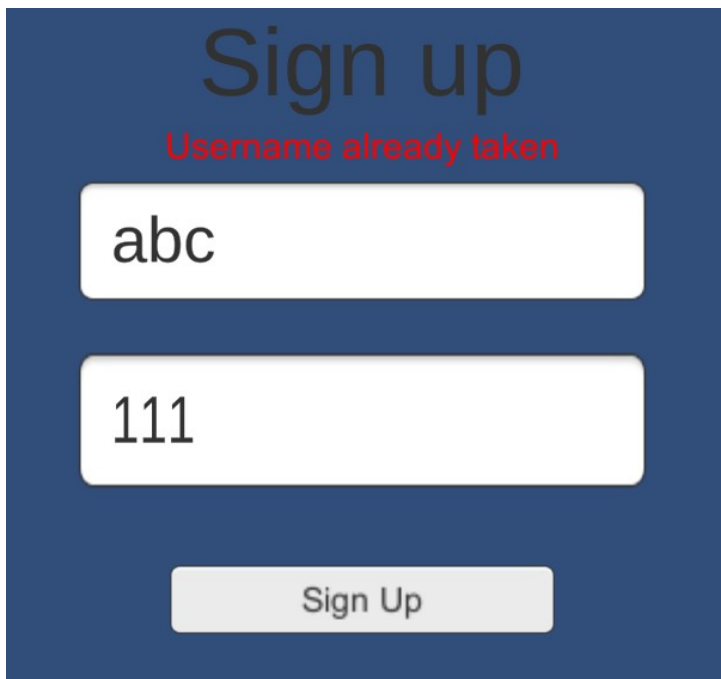
הלקוח שולח לשרת את הנתונים והוא בודק שאין משתמש באותו השם במאגר הנתונים.

במידה ואין משתמש בשם זה הנתונים יוכנסו למאגר הנתונים ותופיעה ההודעה הבאה:



A sign-up form on a dark blue background. At the top, the text "Sign up" is displayed in a large, dark font. Below it, a green message "User successfully created" is shown. The form contains two white input fields: the first contains the text "abcd" and the second contains "111". At the bottom of the form is a light gray button with the text "Sign Up".

במידה ויש משתמש באותו השם:



A sign-up form on a dark blue background. At the top, the text "Sign up" is displayed in a large, dark font. Below it, a red message "Username already taken" is shown. The form contains two white input fields: the first contains the text "abc" and the second contains "111". At the bottom of the form is a light gray button with the text "Sign Up".

## מסך המשחק הראשי



ניתן לשחק בכך שלוחצים על כלי משחק, לאחר מכן יופיעו המהלכים האפשריים של אותו כלי משחק. לאחר מכן לחץ על המשבצת שאליה תרצה לעבור או לחץ על כלי משחק אחר, על מנת לראות את המהלכים האפשריים לביצוע. בשלבים הבאים השרת יבצע מהלך וחוזר חלילה עד שיתקבל מנצח. במסך המשחק ישנם שני כפתורים בצידי המשחק:

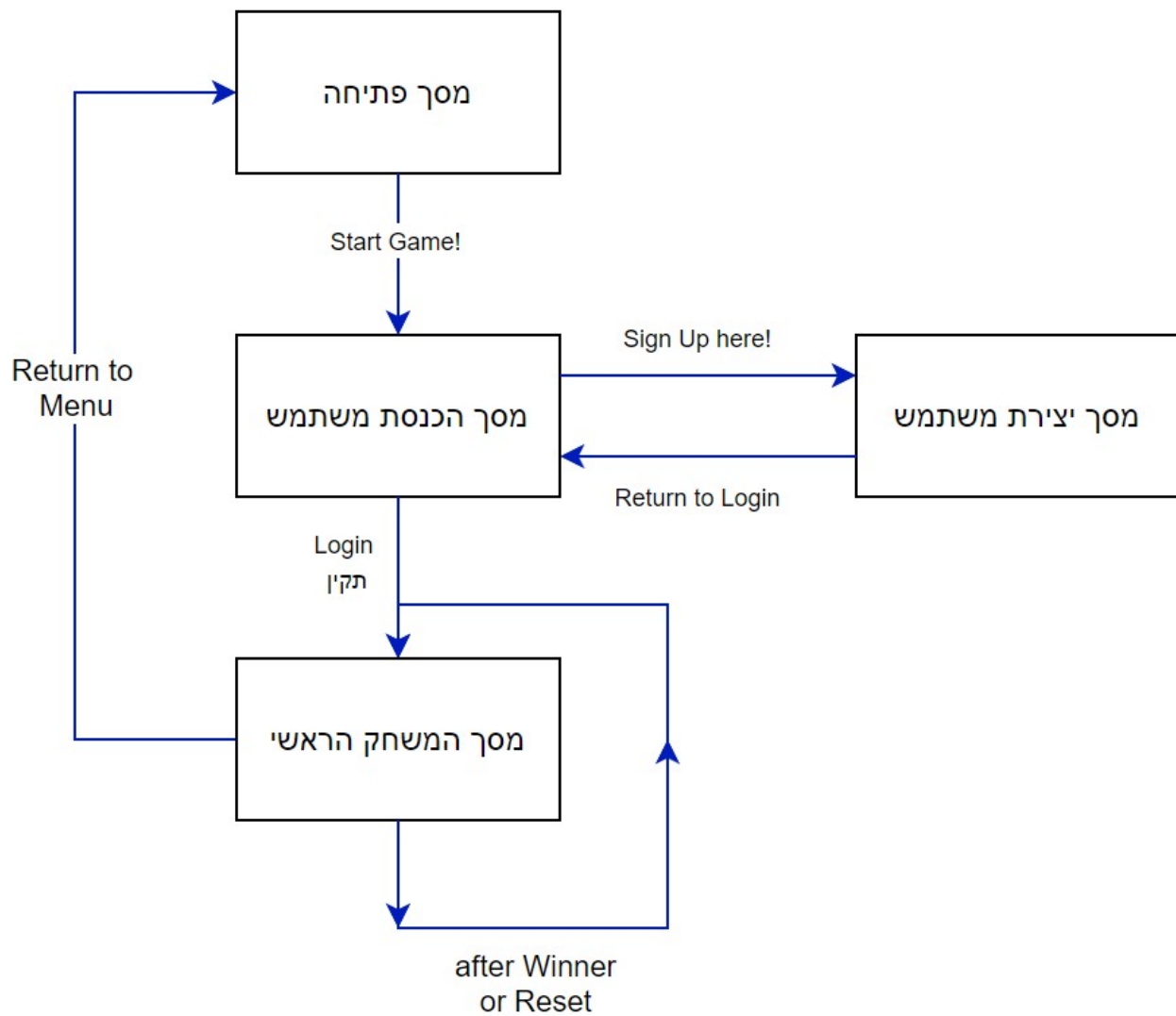
- Reset: הכפתור מאפס את הלוח במקרה שרוצים להתחיל משחק חדש.
- Back to Menu: הכפתור מחזיר למסך הראשי במקרה שרוצים להחליף משתמש או להתחבר לשרת אחר.

כאשר המחשב או השחקן מנצח יופיע המסך הבא (חלק ממסך המשחק הראשי):



כאשר השחקן לוחץ בכל מקום עם העכבר המשחק יתאפס ויתחיל משחק חדש.

## תרשים זרימה של מעבר בין המסכים





## קודים עיקריים של מודלים והפונקציות העיקריות

הרצת השרת:

1. הפעולה ManagingClients נקראת בתחילת הקוד והיא פותחת את השרת ומחברת משתמשים לThread שמריץ את הפעולה HandleClient.

```
1 reference
public static void ManagingClients()
{
    //main command which handles new clients and creates thread for each
    List<Thread> threads = new List<Thread>();
    IPAddress ipAddress = IPAddress.Parse("10.100.102.61");
    IPEndPoint localEndPoint = new IPEndPoint(ipAddress, SERVER_PORT);
    // Create a Socket that will use Tcp protocol
    Socket serverSocket = new Socket(ipAddress.AddressFamily, SocketType.Stream, ProtocolType.Tcp);
    serverSocket.Bind(localEndPoint);
    int id = 0;
    while (true)
    {
        serverSocket.Listen(10);
        Socket handler = serverSocket.Accept();
        RemoveFinishedThreads(threads);
        threads.Add(new Thread(() => HandleClient(handler)) { Name = "t" + id });
        threads.Last().Start();
        id++;
    }
}
```

2. הפעולה HandleClient מקבלת את הsocket ומנהלת את המשחק עם הלקוח.

```
1 reference
public static void HandleClient(Socket clientSocket)
{
    Board board = new Board();
    try
    {
        //handle sockets for each client
        string data = null;
        byte[] bytes = null;
        byte[] msg;
        while (true)
        {
            bytes = new byte[1024];
            if (data != "")
            {
                int bytesRec = clientSocket.Receive(bytes);
                data = Encoding.ASCII.GetString(bytes, 0, bytesRec);
                Console.WriteLine("{0}: {1}", Thread.CurrentThread.Name, data);

                msg = Encoding.ASCII.GetBytes(AnalizingMsg(data, board));
                clientSocket.Send(msg);
            }
            Thread.Sleep(100);
        }
    }
    catch
    {
        clientSocket.Shutdown(SocketShutdown.Both);
        clientSocket.Close();
        Console.WriteLine("close");
    }
}
```

```
void OnMouseUp()
{
    if (!game.IsGameOver() && game.GetWhiteTurn() && isWhite)
    {
        game.DestroyAllMovePlates();
        InitiateMovePlates();
    }
}
```

כאשר לוחצים על כלי משחק הוא יקרא לפונקציה שמייצרת את הריבועים של המהלכים שהכלי יכול לבצע.

```
void InitiateMovePlates()
{
    //Choose which MovePlate To create
    basePiece.UpdateMoves(new Board(game.GetBoard()));
    foreach (Move m in basePiece.GetMoves())
        CreateMovePlate(m);
}
```

הפעולה בודקת עם Core מהם המהלכים האפשריים ויוצרת לכל אחד ריבוע של מהלך פוטנציאלי במסך.

כאשר לוחצים על אחד מהריבועים תקרא פונקציה הבאה:

```

public void OnMouseUp()
{
    Piece piece = pieceObject.GetComponent<Piece>();
    if (attack)
    {
        GameObject enemyChessPiece = game.GetGameObjectOnPosition(pBoard);

        if (enemyChessPiece.name == "whiteKing")
            game.Winner("black");
        if (enemyChessPiece.name == "blackKing")
            game.Winner("white");
        Destroy(enemyChessPiece);
    }

    //set empty in the old piece's board

    Point tempP = piece.GetPBoard();
    game.SetEmptyPosition(piece.GetPBoard());
    piece.SetPBoard(new Point(pBoard));
    piece.MoveToTarget();
    game.SetPosition(pieceObject);
    game.DestroyAllMovePlates();

    client.SetWaitForServer(true);
    client.SendMsg(string.Format("1_{0}_{1}", tempP, pBoard));
}

```

הפעולה לוקחת את האובייקט של כלי המשחק במיקום המטרה המיועדת ואם נמצא שם חייל היא משמידה את כלי המשחק, אם השחקן שנאכל הוא מלך, יוכרז ניצחון.

לאחר מכן החייל שבמיקום ההתחלתי, מתחיל לנוע לכיוון היעד והודעה נשלחת לשרת המכילה את המהלך שנבחר.

```
public bool Login(string username, string password)
{
    //returns if found user and correct password (only both)
    string toSend = string.Format("2_{0}_{1}", username, password);
    string recivedStr = SendAndWaitForResponce(toSend);
    string[] arr = recivedStr.Split('_');
    string answerNumber = arr[0];
    FEN = arr[2];
    FEN = CleanFEN(FEN);
    return answerNumber == "9";
}
```

הפעולה נקראת במסך ההתחברות: אם נלחץ כפתור ההתחברות, או enter והוא יקח את המידע משתי תיבות טקסט.

לאחר מכן הוא ישלח לשרת את פקודה 2 בפרוטוקול, שמנסה להתחבר. היא מקבלת תגובה מהשרת ומגדירה את הלוח שהתקבל מהשרת (כדי להמשיך ממשחק קודם) ומחזירה אם המספר שהוחזר הוא 9 כלומר אם ההתחברות הצליחה.

```
public void AttemptLogin()
{
    GameObject network = GameObject.FindGameObjectWithTag("NetworkManager");
    string usernameStr = username.text;
    string passwordStr = password.text;
    usernameStr = usernameStr.Substring(0, usernameStr.Length - 1);
    passwordStr = Hash(passwordStr.Substring(0, passwordStr.Length - 1));

    bool response = network.GetComponent<Client>().Login(usernameStr, passwordStr);
    if (response)
    {
        SceneManager.LoadScene("Game");
    }
    else
    {
        wrongUserText.gameObject.SetActive(true);
    }
}
```

אם ההתחברות הצליחה הסצנה של המסך יטען ואם לא תופיע הודעת "שם משתמש שגוי".

```

else if (arr[0] == "2")
{
    int tempId = db.GetUserID(arr[1], arr[2]);
    if (tempId == -1 || tempId == 0)
    {
        msgToSend = "10_wrong user or password";
    }
    else
    {
        userID = tempId;
        ReplaceThreads(userID, Thread.CurrentThread.Name, idThreades, threads, threadIDSocket);
        List<Move> moves = db.GetGameMoves(userID);
        board = new AiBoard();
        foreach (Move move in moves)
            board.MakeMove(move);
        msgToSend = "9_ok_" + board.GetFen();
    }
}
}

```

זהו הניתוח של הפקטה מהלקוח בשרת, אם שם המשתמש והסיסמה תקינים יוגדר ה-ID לחיבור של המשתמש במאגר הנתונים וייסגר Thread אחר שמחובר לאותו משתמש, הלוח יתאפס בשרת על פי המשחק הקודם ששוחק באותו המשתמש והלוח יישלח.

## רפלקציה

הפרויקט שלי עסק ביצירת AI לשחמט.

בפרויקט למדתי כיצד לנהל פרויקטים בסדר גודל רחב הרבה יותר מכאלה שעשיתי בעבר, למדתי כיצד משתמשים במנוע גרפי UNITY ואיך הקוד מתנהל עם התוכנה: לכל חלק ויזואלי יש תכונות בסיסיות (מיקום וכו') וקוד שמצורף אליו, מאוד נהנתי להשתמש בדרך חדשה לכתוב קוד ובנוסף לשלב אותו עם שרת.

אני מרוצה מהתוצר הסופי, אך לדעתי שאפתי גבוה מדי ולא היה לי מטרות ברורות, כל הזמן המשכתי להוסיף עוד ועוד דברים, כמו האפשרות לחזור אחורה, כתיבת חוקי השחמט בעצמי במקום להשתמש בספרייה. הדברים הללו חשובים משמעותית פחות ממשחק שעובד או פרוטוקול תקשורת. למדתי מכך שחשוב לקבוע מטרות ראיסטיות לפרויקטים ורק לאחר שהשלמת את המטרות המינימליות להוסיף דברים חדשים.

מאוד נהנתי מיצירת הפרויקט, למדתי בשנה זו הרבה יותר מכל בגרות, גם הכישורים שאתה לומד בהנדסת תוכנה הם מאוד חשובים בהמשך החיים, למידה מהאינטרנט, ניהול זמן וכתיבת קוד קריא ומודולרי.

## ביבליוגרפיה

דרך להעריך את הניקוד ללוח מסויים	<a href="https://www.chessprogramming.org/Simplified_Evaluation_Function">https://www.chessprogramming.org/Simplified_Evaluation_Function</a>
Unity	<a href="https://unity.com/">https://unity.com/</a>
Chess.com	<a href="https://www.chess.com">https://www.chess.com</a>
MinMax Algorithm	<a href="https://en.wikipedia.org/wiki/Minimax">https://en.wikipedia.org/wiki/Minimax</a>
Board Evaluation	<a href="https://www.chessprogramming.org/Simplified_Evaluation_Function">https://www.chessprogramming.org/Simplified_Evaluation_Function</a>
SQLite Library in Visual Studio	<a href="https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/?tabs=netcore-cli">https://docs.microsoft.com/en-us/dotnet/standard/data/sqlite/?tabs=netcore-cli</a>
תמונות של כלי המשחק בהם השתמשתי	<a href="https://opengameart.org/content/pixel-chess-pieces">https://opengameart.org/content/pixel-chess-pieces</a>