

שם העבודה: סופר איקס עיגול אונליין

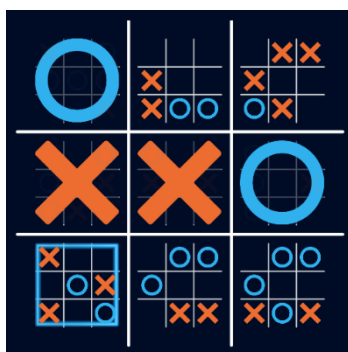
שם התלמיד/ה: אמיל זלסקי

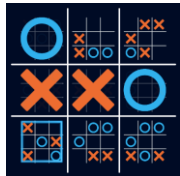
ת.ז.: 326464104

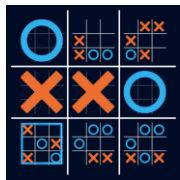
שם המנחה: אורי רוטנברג

שם החלופה: טלפונים חכמים

תאריך הגשה: 18/05/2024

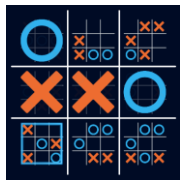






תוכן עניינים

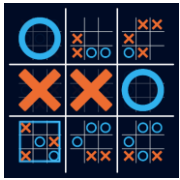
7.....	מענה לדרישות משרדו החינוך לפרויטק 5 יח"ל.....
8.....	מבוא.....
8.....	הרקע לפרויקט.....
8.....	שם הפרויקט.....
8.....	תיאור קצר.....
9.....	קהל היעד.....
9.....	הסיבות לבחירת הנושא.....
9.....	תהליך המחקר.....
9.....	מחקר על תחום הידע.....
10.....	בדיקת אפליקציות קימות.....
10.....	סקירת המצה הקיים בשוק.....
10.....	טכנולוגיות שאינן חלק מתוכנית הלימודים.....
10.....	אתגרים מרכזיים.....
10.....	בעיות שהתמודדתי במהלך פיתוח בפרויקט.....
11.....	על איזה צורך הפרויקט עונה.....
11.....	הצגת הפתרונות לבעיה.....
11.....	הפתרון הנבחר ומדוע.....
12.....	חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים.....
12.....	תיאור תחום הידע.....
12.....	אובייקטים נחוצים.....
14.....	מבנה / ארכיטקטורה.....
14.....	מסכי האפליקציה.....
14.....	מסך הפתיחה - AuthActivity.....
15.....	מסך הכניסה - RegisterActivity.....
16.....	מסך ההרשמה - LoginActivity.....
17.....	המסך הראשי - MainActivity.....
18.....	מסך פרופיל המשתמש - ProfileActivity.....
19.....	מסך טבלת דירוג - LeaderboardActivity.....



20.....	SettingsActivity – מסך ההגדרות
21.....	GameActivity – מסך המשחק
22.....	Screen flow diagram – תרשים מסכים
24.....	UML – תרשים מחלקות
24.....	מימוש הפרויקט
24.....	פירוט המחלקות
24.....	BoardLocation מחלקת
25.....	InnerBoard מחלקת
25.....	Player מחלקת
26.....	OuterBoard מחלקת
27.....	Game מחלקת
28.....	CpuGame מחלקת
28.....	LocalGame מחלקת
28.....	OnlineGame מחלקת
29.....	Cpu FINISH IT מחלקת
29.....	User מחלקת
30.....	EmptyQueryException מחלקת
30.....	GameFullException מחלקת
31.....	BaseGamesRepository מחלקת
32.....	CpuGamesRepository מחלקת
32.....	LocalGamesRepository מחלקת
32.....	OnlineGamesRepository מחלקת
33.....	ReplayGamesRepository מחלקת
34.....	UsersRepository מחלקת
34.....	GamesViewModel מחלקת



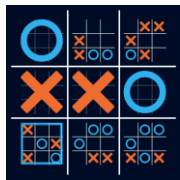
36.....	GamesViewModelFactory	מחלקת
37.....	UsersViewModel	מחלקת
37.....	AppMonitorService	מחלקת
39.....	TextInputLayoutUtil	מחלקת
39.....	UserSessionPreference	מחלקת
41.....	בסיס נתונים	
41.....	סקירת בסיס הנתונים	
43.....	פעולות על בסיס הנתונים	
64.....	מדריך למשתמש	
64.....	הוראות התקנה	
64.....	גרסאות עליהן נבדקה האפליקציה	
64.....	סוגי מכשירים	
64.....	אמולטורים	
66.....	אופן פעולת האפליקציה	
66.....	הסבר כללי	
67.....	AuthActivity	מסך
68.....	RegisterActivity	מסך
69.....	LoginActivity	מסך
70.....	MainActivity	מסך
71.....	ProfileActivity	מסך
73.....	LeaderboardActivity	מסך
74.....	SettingsActivity	מסך
75.....	GameActivity	מסך
77.....	רפלקציה / וראות סיכום אישי	
77.....	תיאור תהליך העבודה על הפרויקט	
77.....	תהליך הלמידה	
77.....	אילו כלים נלקחים להמשך	
78.....	תובנות מהתהליך	
78.....	בראייה לאחור	
79.....	האם וכיצד ניתן לשפר את האפליקציה	



79.....שאלת חקר עצמי

79.....ביבליוגרפיה

80.....נספחים



מענה לדרישות משרד החינוך לפרויטק 5 יח"ל

פרק במחזור	נושאים שמומשו	מחלקה/ות בהן מומש הנושא
6	<ul style="list-style-type: none"> • כתיבת מחלקת Service • שימוש ב AcvtivityForResult – בפורמט החדש <p>הערה: תלמיד שבחר ביותר מנושא אחד מהרשימה, יוכל לבחור הרחבה אחת בלבד מסעיף 9</p>	<ul style="list-style-type: none"> • AppMonitorService • GameActivity • MainActivity • ProfileActivity
7	<ul style="list-style-type: none"> • אחסון וטיפול בנתונים (חובה שמירה ושליפה) 	<ul style="list-style-type: none"> • מודול REPOSITORY
9	<p>נושאים מתקדמים:</p> <ul style="list-style-type: none"> • אפליקציית רב משתתפים • Threads • פיתוח ושימוש ב API בצד שרת (אפשר בכל שפת תכנות) • בינה מלאכותית • אחסון נתונים ב-Firebase/FireStore 	<ul style="list-style-type: none"> • GameActivity • OnlineGameRepository • GamesViewModel • Cpu • מודול REPOSITORY
10	<p>שילוב של 2 נושאים מרשימה זו יחשבו ביחד לנושא מתקדם:</p> <ul style="list-style-type: none"> • Shared Preference • Camera & Gallery 	<ul style="list-style-type: none"> • AuthActivity • LoginActivity • SettingsActivity • BaseGamesRepository • ProfileActivity



מבוא

הרקע לפרויקט

בזמן בחירת הנושא לפרויקט התעניינתי במשחק "סופר איקס עיגול" שזוהי אחת מהגרסאות המסובכות של איקס עיגול רגיל שכולנו מכירים. כבר בהתחלה הבנתי שאני רוצה לעשות אפליקציה ייחודית בנושא ושאין לה אף מתחרה. מהר מאוד הגעתי למסקנה שאין אף אפליקציה שמאפשרת לך לשחק עם אנשים אקראיים במשחק. במהלך הפיתוח הבנתי שרק משחק נגד אנשים אקראיים יהווה פרויקט קל מידי ולכן החלטתי שאוסיף אפשרות לשחק גם מול חבר באותו הטלפון וגם נגד המחשב.

שם הפרויקט

Ultimate Tic Tac Toe Online – סופר איקס עיגול, בזכות זה שאין אף אפליקציה/אתר שמאפשר לשחק בסופר איקס עיגול ברשת, וזה החלק הייחודי בפרויקט שלי, יש לי את הזכות לקרוא כך לאפליקציה

תיאור קצר

המשחק סופר איקס עיגול הוא משחק לא מאוד מפורסם אבל חד משמעית קשה כמו שחמט, המשחק עצמו זה לוח 3×3 של משחקי איקס עיגול שהמטרה זה לנצח בלוח החיצוני כמו באיקס עיגול רגיל. אבל כפי שכולם יודעים, איקס עיגול הוא משחק פתור, משמע אם שני שחקנים משחקים משחק "מושלם" אז המשחק תמיד יגמר בתיקו. אז מה שהמציאו בסופר איקס עיגול, זה שאתה לא יכול לשחק איפה שבא לך, אתה חייב לשחק בלוח פנימי התואם למיקומו של התור הקודם בתוך הלוח הפנימי. לדוגמה אם האיקס שיחק בלוח המרכזי בפניה השמאלית התחתונה, אז העיגול יהיה חייב לשחק בלוח השמאלי התחתון. במידה והמשחק בלוח הדרוש הסתיים (מישהו ניצח בו או נגמר בתיקו), השחקן יכול לשחק איפה שבא לו. כל הסיבוכים האלה גורמים למשחק ילדים להיות משחק אסטרטגיה מסובך איפה נדרש לחשוב מעל ל-5 צעדים קדימה!

חוץ מהמימוש של המשחק בשביל משחק לוקלי (באותו הטלפון) מימשתי גם משחק נגד האפליקציה על ידי אלגוריתם מינימקס שמחשב את הצעד האופטימלי למחשב, ובנוסף באפליקציה ניתן גם לשחק נגד שחקנים אקראיים ברחבי העולם ולראות את המיקום שלך בטבלת הדירוג העולמית. כמובן שאם זה משחק כל כך רציני עם אסטרטגיה וחשיבה, שחקן שרוצה להשתפר חייב גם לצפות במשחקים הקודמים שלו כדי ללמוד מטועיות, וגם האפשרות הזאת ממומשת באפליקציה!

העיצוב באפליקציה מצד אחד מאוד מינימליסטי, אבל מצד שני מאוד נוח לשימוש עקב כפתורים גדולים ובולטים, כך שלא יקרה מצב שהמשתמש יפספס איזושהו פיצ'ר!



קהל היעד

האפליקציה מיועדת לאנשים שמעוניינים לשחק בסופר איקס עיגול לכיף כמה פעמים עם חבר או לאנשים שרוצים לשחק ברמה גבוה ולראות את ההישגים שלהם בלוח הדירוג שמציג את השחקנים הטובים בעולם.

הסיבות לבחירת הנושא

1. אני משחק הרבה ב-Chess.com ואני רואה איך כל העולם מתלהב מזה כי הם הצליחו להפוך משחק בשחמט למשהו מאוד נגיש שאתה לא צריך להפגש עם מישהו או להרשם לתחרויות, אלה אתה פשוט פותח את הטלפון ומשחק, בכל מקום בכל זמן. והחלטתי שאני רוצה לנסות להפוך את הספור איקס עיגול למשהו דומה, כי הרבה מאוד פעמים קרה לי שרציתי לשחק אבל לא היה אף אחד לידי שרצה גם.
2. רצון ללמוד את האלגוריתם מינימקס בפרוייקט דיי מסובך (ולא באיקס עיגול רגיל)
3. רצון ליצור אפליקציה שיחידה או בין היחידות בנושא שלה

תהליך המחקר

במהלך שלב המחקר של פרויקט, הבנתי שאני רוצה לבנות חוויית משחק קלה, נגישה ואינטואיטיבית במיוחד עבור בני הנוער, שמחפשים פעילות מהנה ומהירה בזמנם הפנוי. בהתחלה הגדרתי את קהל היעד – נוער שמעוניין במשחק מאתגר חשיבתית אבל לא מסובך מדי. כדי להבין טוב יותר את הצרכים שלהם, בחנתי אפליקציות קיימות, כמו chess.com, וראיתי שהצלחתן נובעת בעיקר מממשק מינימליסטי שמאפשר ליצר משחק בלחיצת כפתור, ליצור אתגר עם חבר או למצוא יריב אקראי בקליק אח

בהמשך חשבתי, אילו אלמנטים ויזואליים משפרים עבורי את חוויית המשחק. והבנתי שמה שאני צריך זה מסך ראשי עם מספר כפתורים קטן, משוב ויזואלי עדין, ואפקטים מיידיים של ניצחון או הפסד. מתובנות אלו סיכמתי את הפריטים העיקריים לשילוב בשלב העיצוב הבא: מסך פתיחה מינימליסטי, אפשרות יצירת חדר משחק בלחיצה, ולוח אינטראקטיבי בעיצוב מינימליסטי עם דירוג אישי.

מחקר על תחום הידע

במהלך המחקר למדתי איך אתרים כמו Chess.com מצליחים להפוך משהו די מסובך לנערץ על ידי בני נוער ברחבי העולם – במשחק חייב להיות דירוג כדי שהמשתמשים יקבלו מוטיבציה להשתפר ולשחק עוד, כי כולם רוצים להיות גבוהים יותר בדירוג ממה שהם עכשיו. בנוסף מאוד חשוב לשמור על עיצוב מינימליסטי עם ניווט פשוט וכפתורים גדולים כדי לא להכביד על המתשמש יותר ממה שהמשחק עמו מכביד.



בדיקת אפליקציות קימות

קיימות מספר אפליקציות שמאפשרות לשחק בסופר איקס עיגול אך הן תמיד מכילות רק חלק מהדברים – או משחק נגד המחשב או משחק נגד שחקן באותו הטלפון או משחק נגד חבר ברשת (עם קוד התחברות), לכן החלטתי ליצור אפליקציה שמאחדת את הכל במקום אחד

סקירת המצה הקיים בשוק

כל האפליקציות הקיימות הן מאוד קטנות עם פונקציונליות מאוד קטנה שנראת כמו פרויקט שנכתב ביומיים, יש בהן לפעמים באגים ויזואליים, או פשוט עיצוב מחריד, לכן Ultimate Tic Tac Toe Online מבריק מאוד לעומתם, במיוחד עם זה שהקוד שלו נמצא בגיטהאב והמשחק עוד לא בגרסה האחרונה שלו ויקבל עדכונים רבים.

טכנולוגיות שאינן חלק מתוכנית הלימודים

1. שימוש ב-Firebase Cloud Functions בשביל פונקציות בצד שרת בלי לכתוב שרת
2. שימוש ב-EncryptedSharedPreferences ובפונקציית Hash
3. שימוש ב-Pagination בטעינת נתונים ממסד הנתונים
4. שימוש בספרייה Ucrop בשביל עיבוד תמונות
5. יצירת מנגנון דו כיווני בזמן אמת בשביל לנהל משחק
6. ירושה ממחלקת Exception
7. יצירת מחלקת Service
8. תכנות אסינכרוני בעזרת Executor
9. שימוש בקובץ strings.xml בשביל תרגום
10. מימוש אלגוריתם מינימקס
11. שימוש ב-TextInputLayout

אתגרים מרכזיים

בעיות שהתמודדתי במהלך פיתוח בפרויקט במהלך הפיתוח נתקלתי מספר בעיות רציניות:

1. מימוש המשחק ברשת שכולל העברת מידע דו כיווני דרך Firestore



2. מימוש אלגוריתם מינימקס בשביל סופר איקס עיגול
3. עיצוב UI יפה ונוח שנראה טוב בכל המכשירים כמו באפליקציות עם מפתחים מקצועיים

על איזה צורך הפרויקט עונה

הפרויקט עונה על הצורך ביצירת פלטפורמה חברתית ומשחקית שמאפשרת לבני נוער ולמשתמשים באופן כללי לקבל הנאה אתגרית. האופציה לשחק מול המחשב מספקת אימון קוגניטיבי ופיתוח מיומנויות אסטרטגיות ללא תלות בזמינות של שותף, בעוד שמשחק לוקלי ושחקן רנדומלי ברשת מאפשרים חוויית תחרות מגוונת ומיידית, שמגשרת על הפער בין מפגש פנים אל פנים לבין משחק מרוחק. תכונת צפייה בהיסטוריית המשחקים, כולל אפשרות לצפות במהלך עצמו, מעניקה למשתמשים כלי למידה ומעקב התקדמות אישי, ותורמת למוטיבציה להשתפר ולהמשיך לשחק.

בנוסף, עדכון תמונת פרופיל ומד כושר ממקדים את החוויה בפרטניות של כל שחקן, ומאפשרים לו להשתתף בטבלאות דירוג ולראות את עצמו עולה או יורד במקומות. שילוב טבלת דירוג גלובלית מספקת תמריץ להתחרות ולהשיג הישגים, בעוד שהנגשה פשוטה של ניהול הפרופיל והסטטיסטיקות הופכת את האפליקציה לפלטפורמה של ממש.

הצגת הפתרונות לבעיה

1. בעיה: טיפול בהתחברות משתמש בצורה מאובטחת.
פתרון: אפשרתי הצגת/הסתרת הסיסמה לשיפור חוויית המשתמש. הצפנתי את הסיסמה והשתמשתי ב- EncryptedSharedPreferences בשביל לשפר את רמת אבטחת המידע באפליקציה.
2. בעיה: שימוש בתמונה רגילה מהמצלמה עלולה להיות גדולה ולגרום לזמני טעינה ארוכים או לבעיות זיכרון.
פתרון: אופטימיזציה של תמונת התמונה על-ידי שימוש ב-cropping וסקיילינג של התמונה.
3. בעיה: קריאה למתודה סינכרונית מבלי לחסום את ממשק המשתמש.
פתרון: השתמשתי במחלקת Executor על מנת להריץ מתודות מה-repository בתהליכון נפרד

הפתרון הנבחר ומדוע

הפתרון שנבחר לפרויקט הוא אפליקציה לטלפון, המבוססת על שירותי Firebase וארכיטקטורת MVVM – שילוב שמאפשר גם גמישות טכנולוגית וגם חוויית משתמש חלקה ועקבית. הבחירה



באפליקציה ולא באתר נבעה ממחקר שוק ותצפיות בשימוש בפועל – כמעט כל בני האדם בעולם המודרני נמצאים תמיד בקרבת הטלפון שלהם, אפילו כשהם מול המחשב, לפעמים את פותחים את הטלפון פשוט מתוך הרגל. בזכות זה, הם יפתחו הרבה יותר את האפליקציה שתשב אצלם במסך בית מאשר את האתר שצריך לזכור את הכתובת שלו ולכתוב אותה.

הבחירה ב-Firebase הייתה פשוטה, מכיוון שהיא מספקת פלטפורמה שלמה המותאמת לאפליקציות: מסד נתונים בענן (Firestore Cloud), פונקציות צד שרת בלי שרת (Cloud Functions) ועוד שירותים רבים. כל אלה מנוהלים על ידי Firebase עצמו ללא שרת פרטי כלל – מה שחוסך המון זמן, משאבים, ובמיוחד מסייע לתלמיד שמפתח את המערכת במסגרת פרויקט יחידי. בנוסף, Firebase מאפשר סנכרון מהיר, בכל רחבי העולם המבטיח חוויית משתמש אחידה בין מכשירים שונים.

השימוש בארכיטקטורת View-ViewModel-Model (MVVM) נבחר על מנת לשמור על הפרדה ברורה בין הלוגיקה והבסיס נתונים לממשק המשתמש, מה שמקל מאוד את תחזוקת הקוד, הופך את הבאגים לקלים יותר לאיתור, ומאפשר פיתוח מאורגן ורציף. לכל שכבה באפליקציה יש את תחום האחריות שלה – מה שהופך את הקוד לקריא, ומוכן להרחבות עתידיות כמו הוספת של חברים.

חידושים, התאמות ועדכונים של אלמנטים טכנולוגיים, עיצוביים ואחרים

הפרוייקט שלי מתבסס על האפליקציה הידועה Chess.com, ממנה לקחתי את רב רעיונות העיצוב אך עם שינויים רבים שהופכים את האפליקציה שלי ליחודית, מבחינת המשחק עצמו, קיימות אפליקציות רבות שמממשות סופר איקס עיגול, אך אף אחת מהן לא הופכת את המשחק למשהו יותר גדול שמכוון לאסוף קהילה גדולה סביבו

תיאור תחום הידע

אובייקטים נחוצים

משתמש:

- יכול להירשם
- יכול לערוך את הפרופיל שלו
- יכול לצפות בהיסטוריית המשחקים שלו
- יכול להסתכל במהלכים במשחקים קודמים שלו
- יכול לשחק נגד המחשב
- יכול לשחק לוקלית



- יכול לשחק באונליין

"המחשב"

- יכול לחשב את המהלך האופטימלי בשבילו

- יכול להעריך את המשחק

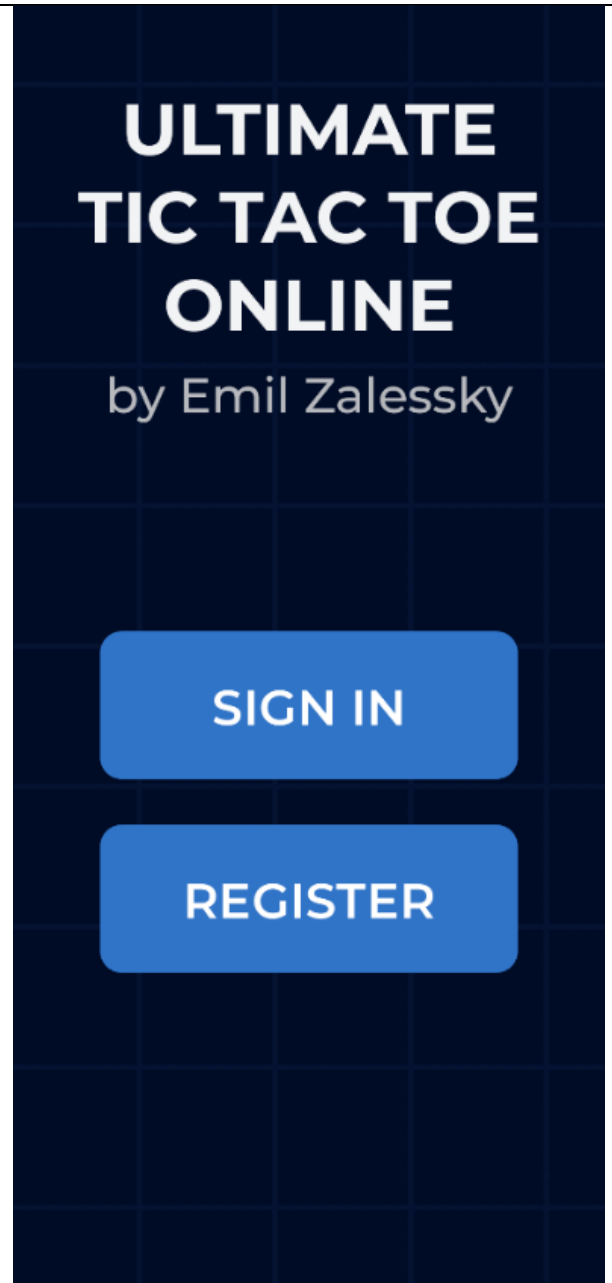


מבנה / ארכיטקטורה

מסכי האפליקציה

מסך הפתיחה - AuthActivity:

- תיאור המסך - מסך זה מוצג כאשר האפליקציה נפתחת, ומאפשר למשתמש לבחור האם להיכנס עם חשבון קיים או להירשם כמשתמש חדש. הוא מציג רקע, כותרת גרפית, ואת פרטי המפתח של האפליקציה.
- btnLogin – כפתור למעבר למסך ההתחברות.
- btnRegister – כפתור למעבר למסך ההרשמה.





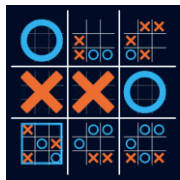
מסך הכניסה - RegisterActivity

- תיאור המסך - המסך מציג טופס הרשמה הכולל שדות להזנת שם משתמש, סיסמה ואישור סיסמה עם אפשרות להסתכל על הסיסמה, כפתור להרשמה וכפתור חזרה למסך קודם.
- etUsername – שדה להזנת שם המשתמש.
- etPassword – שדה להזנת הסיסמה.
- etConfirmPassword – שדה להזנת אימות הסיסמה.
- btnRegister – כפתור הרשמה ומעבר למסך ההתחברות.
- btnBack – כפתור חזרה למסך הקודם.



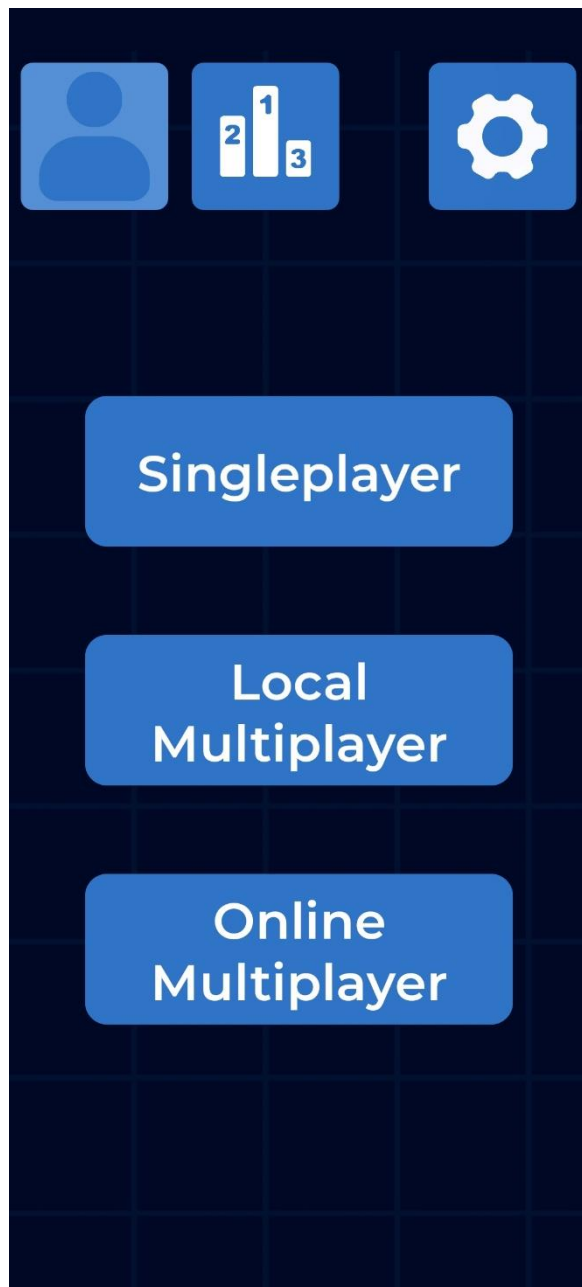
מסך ההרשמה - LoginActivity

- תיאור המסך – המסך מציג טופס התחברות הכולל שדות להזנת שם משתמש, סיסמה, אפשרות "זכור אותי" וכפתור להתחברות וכפתור חזרה למסך הקודם.
- etUsername – שדה להזנת שם המשתמש.
- etPassword – שדה להזנת הסיסמה.
- cbRememberMe – תיבת סימון לשמירת פרטי ההתחברות.
- btnSignIn - כפתור התחברות ומעבר למסך הראשי.
- btnBack – כפתור חזרה למסך הקודם.



המסך הראשי – MainActivity

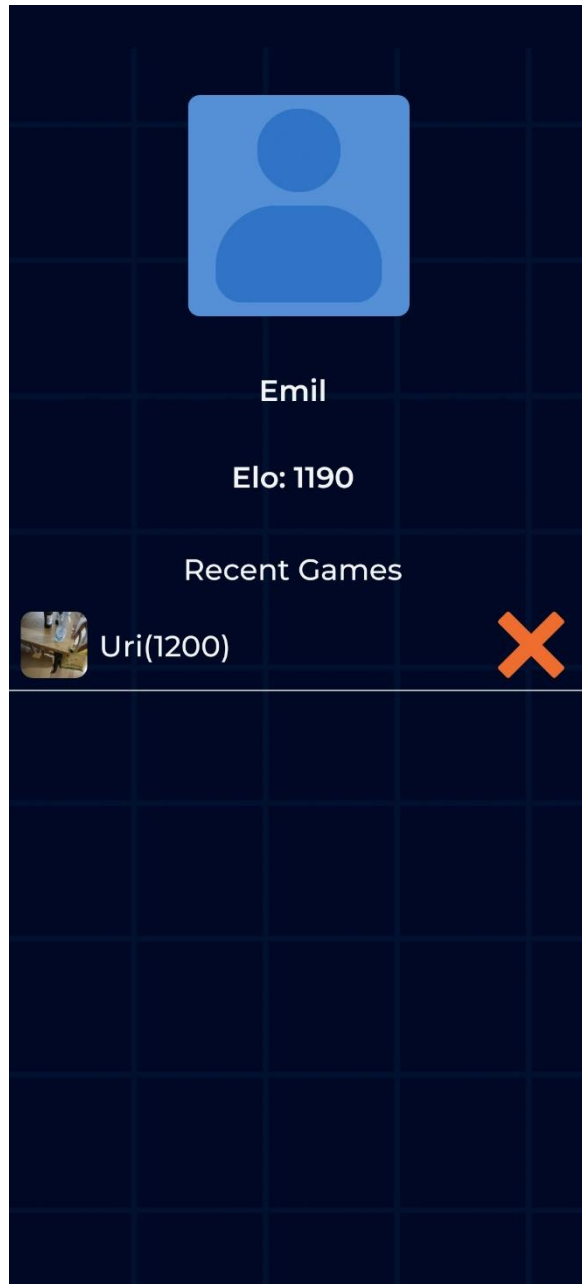
- תיאור המסך - במסך זה מוצגים כפתורים לשלושה סוגי משחקים – נגד המחשב, מול שחקן לוקלי, ומול שחקנים אונליין. בנוסף, מוצגים אייקונים המובילים למסך הפרופיל האישי, טבלת הדירוג והגדרות.
- ivProfile – כפתור מעבר לעמוד הפרופיל, מציג את תמונת הפרופיל.
- ivLeaderBoard – כפתור מעבר לטבלת הדירוג.
- ivSettings – כפתור מעבר להגדרות.
- btnCpu – כפתור להתחלת משחק נגד המחשב.
- btnLocal – כפתור להתחלת משחק לוקלי.
- btnOnline – כפתור להתחלת משחק אונליין.





מסך פרופיל המשתמש – ProfileActivity

- תיאור המסך - מסך זה מאפשר למשתמש לראות את פרטי החשבון שלו. הוא כולל תצוגה תמונת פרופיל, שם משתמש, ניקוד ELO, המשחקים האחרונים מוצגים ברשימת גלילה.
- ivPfp – תמונת הפרופיל של המשתמש בלחיצה נותנת אפשרות להחלפה.
- tvUsername – שם המשתמש.
- tvElo – מד הכושר של המשתמש (Elo).
- rvGames – רשימת המשחקים האחרונים.





מסך טבלת דירוג – LeaderboardActivity

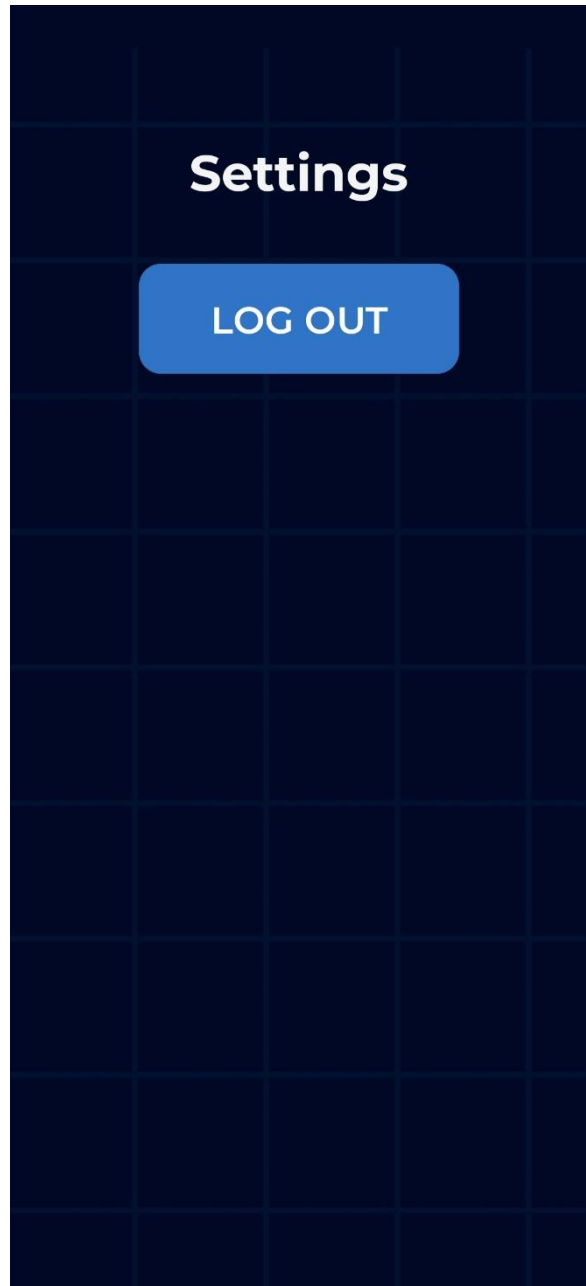
- תיאור המסך - מסך זה מציג את טבלת הדירוג של השחקנים, כשהמידע מופיע ברשימת גלילה מסודרת לפי מיקום. העמודות מוצגות מעל ה-RecyclerView. המשתמש המחובר מודגש מעט.
- tvRank – עמודת "דירוג".
- tvPlayer – עמודת "שחקן".
- tvRating – עמודת "Elo".
- rvLeaderboard – רשימת השחקנים הכי טובים.

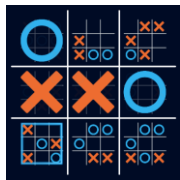
Leaderboard		
Rank	Player	Elo Rating
#1	 כצכ	9999
#2	 6	4444
#3	 Uri	1340
#4	 Emil.za07	1292
#5	 Mom	1201
#6	 123412	1200
#7	 Emil	1200
#8	 #@%^!^#&	1200
#9	 rqwga	1200
#10	 Toanls1	1200
#11	 g	1200



מסך ההגדרות – SettingsActivity

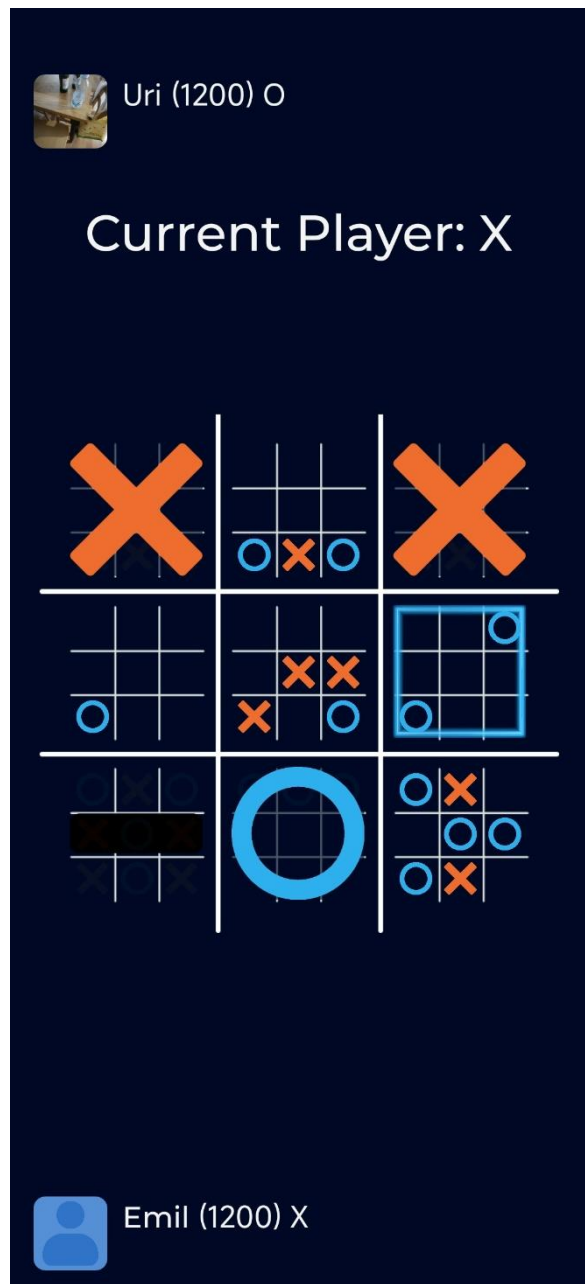
- תיאור המסך - מסך פשוט שבו מוצג כפתור מרכזי להתנתקות מהחשבון.
- btnLogOut – כפתור התנתקות מהחשבון.





מסך המשחק – GameActivity

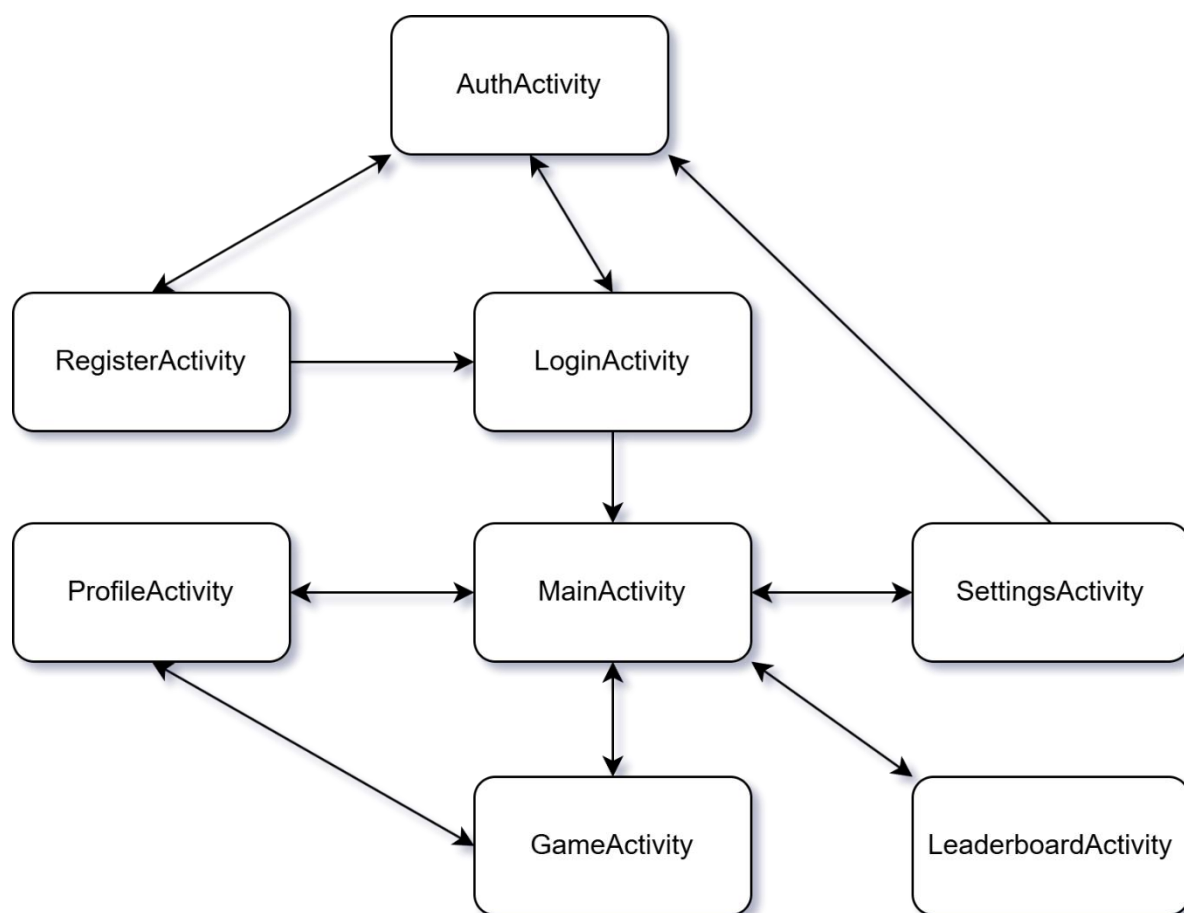
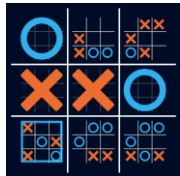
- תיאור המסך – המסך הראשי של האפליקציה, המסך מציג את לוח המשחק עם פרטים על שני השחקנים, אינדיקציה של מי התור. במצב "צפיה חוזרת" קיימים שני כפתורים בשביל התקדמות קדימה ואחורה במהלך המשחק. במשחק אונליין, יש מסך טעינה עם כפתור ביטול אם היריב עדיין לא נמצא.
- gridBoard – לוח המשחק עצמו.
- llP2 – תצוגת שחקן 2, מכילה את התמונה, שם, דירוג Elo ואת הסימן שלו (מוסתר בהתחלה).
- ivP2Pfp – תמונת הפרופיל של שחקן 2.
- tvP2Name – שם המשתמש של שחקן 2.
- tvP2Elo – ה-Elo של שחקן 2.
- tvP2Sign – הסימן של שחקן 2.
- llP1 – תצוגת שחקן 1, מכילה את התמונה, שם, דירוג Elo ואת הסימן שלו (מוסתר בהתחלה).
- ivP1Pfp – תמונת הפרופיל של שחקן 1.
- tvP1Name – שם המשתמש של שחקן 1.



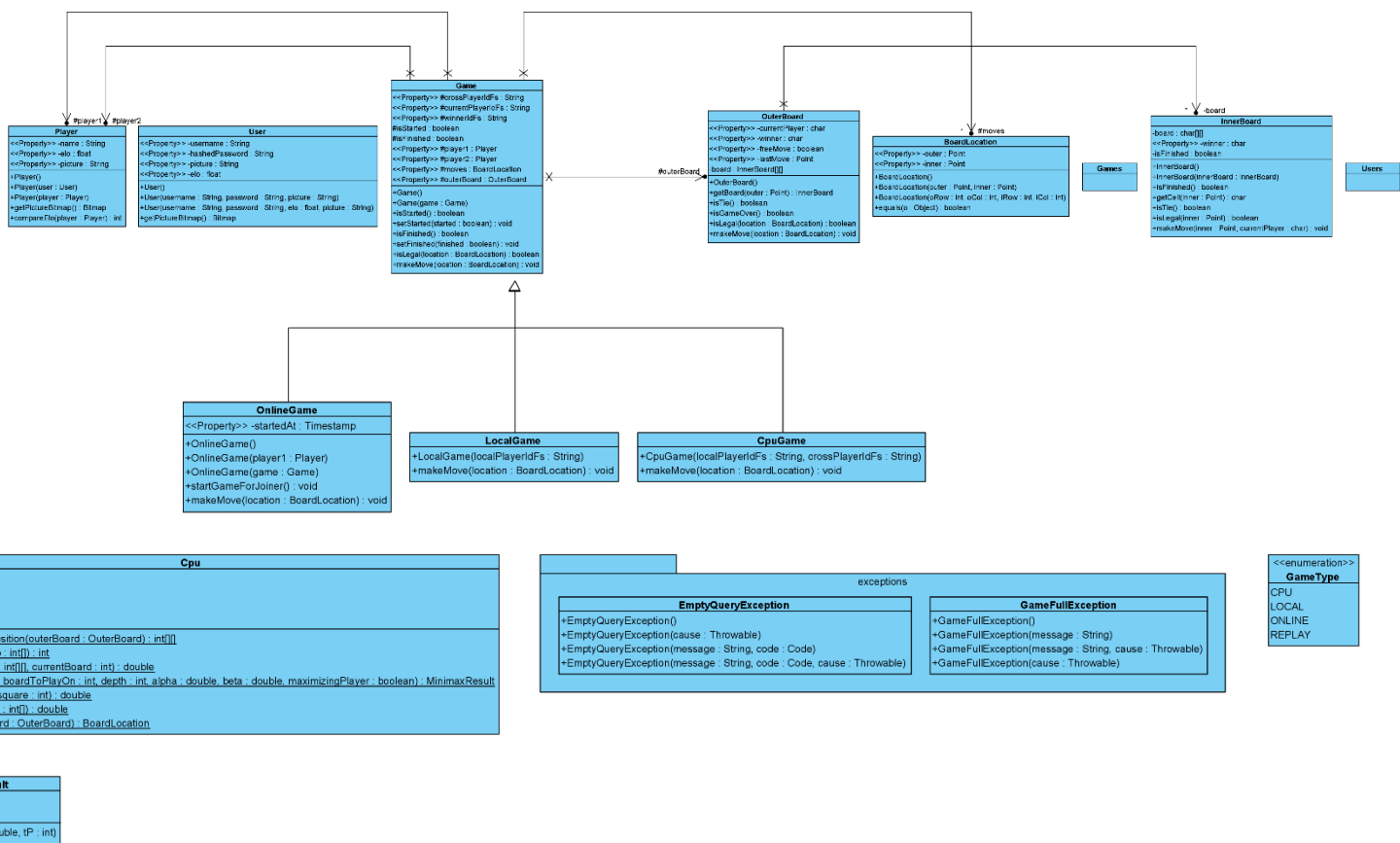


<ul style="list-style-type: none">• tvP1Elo – ה-Elo של שחקן 1.• tvP1Sign – הסימן של שחקן 1.• tvCurrentPlayer – מציג הודעה של מי התור.• btnAbort – כפתור לביטול חיפוש יריב.• llReview – מכיל כפתורים קדימה אחורה בשביל צפיה חוזרת, מוסתר תמיד חוץ מבמצב צפיה חוזרת.• btnForward – כפתור למעבר למהלך הבא. כאשר מגיע מהלך האחרון ניהיה אפור.• btnBackward – כפתור למעבר למהלך הקודם. כאשר מגיע למהלך הראשון ניהיה אפור.	
--	--

תרשים מסכים – Screen flow diagram



תרשים מחלקות – UML



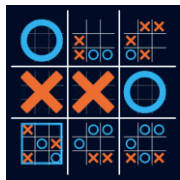
מימוש הפרויקט

פירוט המחלקות

BoardLocation מחלקת

- שם המחלקה: BoardLocation
- תפקיד המחלקה: שומר את מיקומו של תא במשחק – Ultimate Tic Tac Toe גם את מיקום הלוח הקטן בלוח הגדול, וגם את מיקום התא בתוך הלוח הקטן.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
-----------	------------	------------	-------	-------



boardRow	int	private	השורה של הלוח הקטן בתוך הלוח הגדול	לזיהוי מיקום שורת הלוח הקטן במבנה הכולל
boardCol	int	private	העמודה של הלוח הקטן בתוך הלוח הגדול	לזיהוי מיקום עמודת הלוח הקטן במבנה הכולל
cellRow	int	private	השורה של התא בתוך הלוח הקטן	לזיהוי שורת התא בתוך הלוח הקטן
cellCol	int	private	העמודה של התא בתוך הלוח הקטן	לזיהוי עמודת התא בתוך הלוח הקטן

מחלקת InnerBoard

- שם המחלקה: InnerBoard
- תפקיד המחלקה: מטפלת בלוח פנימי של המשחק, כולל ניהול מצב המשבצות, חוקיות מהלכים, בדיקת סיום וזיהוי מנצח.
- תכונות המחלקה:

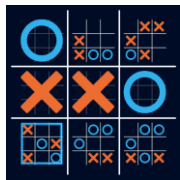
שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
board	char[][]	private final	לוח המשחק	שמירת מצב הלוח הפנימי (איזה שחקן תפס איזו משבצת)
winner	char	private	המנצח של המשחק בלוח הנוכחי	לזיהוי המנצח בלוח הפנימי
isFinished	boolean	private	מציין אם המשחק בלוח נגמר	בדיקה האם המשחק נגמר

- הסבר על פעולות המחלקה
- פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים) כדוגמת: רשימת פריטים, משתמשים... (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת Player

- שם המחלקה: Player
- תפקיד המחלקה: מאחסנת מידע על שחקן, כולל שם, דירוג ELO ותמונת פרופיל.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
name	String	private	שם השחקן	לצורך תצוגה, זיהוי ואחסון נתונים



לצורך תצוגה, זיהוי ואחסון נתונים	דירוג ה־ELO של השחקן	private	float	elo
לא נשלחת ל־Firestore, משמשת להצגת התמונה	תמונת פרופיל כטקסט מקודד Base64	@Exclude private	Sting(Base64)	picture

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת OuterBoard

- שם המחלקה: OuterBoard
- תפקיד המחלקה: מנהלת את לוח המשחק הראשי.
שומרת את מצב המשחק, השחקן הנוכחי, המנצח, חוקיות מהלכים, והמהלך האחרון.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
board	InnerBoard[][]	private final	מטריצה בגודל 3x3 של לוחות פנימיים	אחסון מצב כל הלוחות הפנימיים
currentPlayer	char	private	השחקן הנוכחי (X או O)	קובע של מי התור
winner	char	private	המנצח של המשחק 'X', 'O', 'T' לתיקו או 0 אם המשחק לא נגמר	מציע על מצב סיום המשחק
freeMove	boolean	private	האם השחקן יכול לשחק בכל לוח	משפיע על החוקים בתור הבא
lastMove	Point	private	מיקום המהלך האחרון	קובע איזה לוח ישחק בתור הבא

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.



מחלקת Game

- שם המחלקה: Game
- תפקיד המחלקה: מייצגת משחק פעיל של סופר איקס-עיגול, כולל ניהול שחקנים, מהלכים, מצב הלוח, וזיהוי מנצח.
- המחלקה שומרת את מצב המשחק לצורך הצגה או שמירה ב-Firebase.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
player1	Player	protected	השחקן הראשון	ניהול פרטי שחקן
player2	Player	protected	השחקן השני	ניהול פרטי שחקן
crossPlayerIdFs	String	protected	ID של השחקן שמשחק ב-X-	לדעת מי האיקס
currentPlayerIdFs	String	protected	ID של השחקן שבתורו כעת	ניהול תור
winnerIdFs	String	protected	ID של המנצח	זיהוי מנצח בסיום המשחק
isStarted	boolean	protected	האם המשחק התחיל	מעקב אחר מצב התחלה
isFinished	boolean	protected	האם המשחק הסתיים	מעקב אחר סיום
moves	List<BoardLocation>	protected	רשימת המהלכים שבוצעו במשחק	שמירה והעברת מהלכים בין שחקנים
outerBoard	OuterBoard	@Exclude protected	מייצג את מצב לוח המשחק בפועל, לא נשמר ב-Firebase	ניהול לוגיקת המשחק

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים) כדוגמת: רשימת פריטים, משתמשים... (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.



מחלקת CpuGame

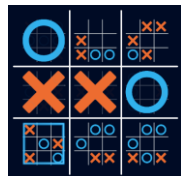
- שם המחלקה: CpuGame
- תפקיד המחלקה: מחלקת CpuGame יורשת מ Game ומשמשת למשחק בין שחקן אנושי לבין מחשב.
- היא אחראית לאתחול המשחק ולניהול תורות ומצבי סיום.
- תכונות המחלקה: אין
- הסבר על פעולות המחלקה
- פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה).

מחלקת LocalGame

- שם המחלקה: LocalGame
- תפקיד המחלקה: המחלקה LocalGame יורשת מהמחלקה Game, ומייצגת משחק לוקאלי (שני שחקנים פיזיים שמשחקים באותו מכשיר).
- היא אחראית לאתחול המשחק ולניהול תורות ומצבי סיום.
- תכונות המחלקה: אין
- הסבר על פעולות המחלקה
- פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה).

מחלקת OnlineGame

- שם המחלקה: OnlineGame
- תפקיד המחלקה: OnlineGame יורשת מהמחלקה Game ומייצגת משחק אונליין בין שני שחקנים דרך האינטרנט.
- היא אחראית על ניהול שחקנים, זמן התחלה, ביצוע מהלכים, והחלפת תור בין שחקנים שונים (עם מזהים שונים).
- תכונות המחלקה:



שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
startedAt	Timestamp	private	תיעוד זמן התחלת המשחק	סידור המשחקים בProfileActivity

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת Cpu FINISH IT

- שם המחלקה:
- תפקיד המחלקה:
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
תכונה 1	סוג 1			

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת User

- שם המחלקה: User
- תפקיד המחלקה: User מייצגת משתמש באפליקציה.

היא שומרת מידע על שם משתמש, סיסמה מוצפנת, תמונת פרופיל, ודירוג ELO של המשתמש.

- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
username	String	private	שם המשתמש	זיהוי בשתמש בהתחברות ובמשחקים
hashedPassword	String	private	סיסמה מוצפנת	משמשת לאימות פרטי התחברות בצורה מאובטחת
picture	String	private	תמונת פרופיל בקידוד Base64	משמשת להצגת תמונת המשתמש



משמש לדירוג המשתמשים על בסיס תוצאות משחקים	דירוג ELO של המשתמש	private	float	elo
---	------------------------	---------	-------	-----

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע
כפי שנשמר בקובץ / טבלה.

מחלקת EmptyQueryException

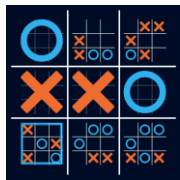
- שם המחלקה: EmptyQueryException
- תפקיד המחלקה: מחלקת Exception מותאמת אישית שמרחיבה את FirebaseFirestoreException, משמשת לציון מצב בו אין משחקים זמינים (תוצאת שאילתה ריקה).
- תכונות המחלקה: אין

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת GameFullException

- שם המחלקה: GameFullException
- תפקיד המחלקה: מחלקת Exception מותאמת אישית המשמשת לציון מצב שבו משחק כבר מלא ואינו יכול להכיל שחקנים נוספים.
- תכונות המחלקה: אין

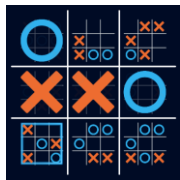
- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.



מחלקת BaseGamesRepository

- שם המחלקה: BaseGamesRepository
- תפקיד המחלקה: מחלקה אבסטרקטית המנהלת את לוגיקת המשחק ונתוני המשחק, מספקת LiveData למצב המשחק, למנצחים בלוח החיצוני, ולסטטוס המשחק, ומכילה פונקציות בסיסיות להתחלת משחק וביצוע מהלכים.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
lvErrorCode	MutableLiveData<Integer>	protected final	LiveData a לשמירת קוד שגיאה	דיווח שגיאות לממשק המשתמש
lvGame	MutableLiveData<Game>	protected final	LiveData a למצב הנוכחי של המשחק	צפייה ועדכון מצב המשחק
lvOuterBoardWinners	MutableLiveData<Char[][]>	protected final	LiveData לשמירת מצב מנצחי הלוח החיצוני	עדכון ומעקב אחר מנצחי לוחות פנימיים
lvIsStarted	MutableLiveData<Boolean>	protected final	LiveData לציון האם המשחק התחיל	מעקב אחר התחלת המשחק
lvIsFinished	MutableLiveData<Boolean>	protected final	LiveData a לציון האם המשחק הסתיים	מעקב אחר סיום המשחק
localPlayerIdFs	String	protected final	מזהה המשתמש המקומי	זיהוי השחקן לביצוע מהלכים ולוגיקה



- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים) כדוגמת: רשימת פריטים, משתמשים... (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת CpuGamesRepository

- שם המחלקה: CpuGamesRepository
- תפקיד המחלקה: מחלקת repo עבור משחקים מול המחשב. יורשת מהמחלקה BaseGamesRepository ומממשת את לוגיקת המשחק הספציפית למשחקים נגד מחשב, כולל התחלת משחק וביצוע מהלך עבור המחשב.
- תכונות המחלקה: אין

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים) כדוגמת: רשימת פריטים, משתמשים... (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

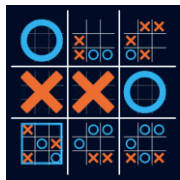
מחלקת LocalGamesRepository

- שם המחלקה: LocalGamesRepository
- תפקיד המחלקה: מחלקת repo לניהול לוגיקת משחקים לוקליים (2 שחקנים באותו מכשיר). יורשת מהמחלקה BaseGamesRepository ומטפלת באתחול המשחק ועדכון מצבים בהתאם להתקדמותו.
- תכונות המחלקה: אין

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים) כדוגמת: רשימת פריטים, משתמשים... (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת OnlineGamesRepository

- שם המחלקה: OnlineGamesRepository



- תפקיד המחלקה: מחלקה זו אחראית על ניהול משחקי אונליין באמצעות Firebase Firestore ו-Cloud Functions. היא יורשת את BaseGamesRepository ומיישמת לוגיקה להתאמת שחקנים לפי דירוג ELO, ניהול מצב המשחק בזמן אמת, וטיפול בדפדוף (pagination) של היסטוריית המשחקים של המשתמש.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
lvGameIds	MutableLiveData<String>	private final	שומר את ה-ID של המשחק הנוכחי ב-Firebase	נחשף ל ViewModel- לצורך קבלת ה-ID של המשחק ל UI-או לפונקציות
lvGames	MutableLiveData<Games>	private final	שומר את רשימת המשחקים של המשתמש (paginated)	מתעדכן ב- getUserGamesPaginated להצגת היסטוריית המשחקים
retryCounter	int	private	מונה מספר הנסיונות החוזרים ב-startGame במקרה של שגיאות	משמש להגבלת מספר הנסיונות להצטרפות למשחק

- הסבר על פעולות המחלקה
- פעולות לוגיות עיקריות בצורך קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה).

מחלקת ReplayGamesRepository

- שם המחלקה: ReplayGamesRepository
- תפקיד המחלקה: מחלקה לניהול צפיה חוזרת במשחקים קודמים. יורשת את BaseGamesRepository, מאפשרת טעינה של משחק קיים לצפייה מחדש מבלי לשנות את לוגיקת ה-CRUD הבסיסית.



- תכונות המחלקה: אין

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה

- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת UserRepository

- שם המחלקה: UserRepository
- תפקיד המחלקה: ניהול נתוני משתמשים ואימות ייחודיות שמות משתמש ב-Firebase יורשת מ-BaseRepository ומספקת פעולות CRUD עבור אובייקטי User ודפדוף (pagination) ברשימת השחקנים המובילים לפי דירוג ELO.

- תכונות המחלקה:

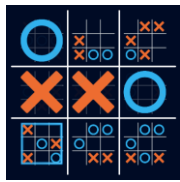
שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
lvTopUsers	MutableLiveData<Users>	private final	LiveData שמכיל את רשימת שחקני ה-ELO המובילים בדף הנוכחי	נתונים מתקבלים ב-getTopPlayersPaginated ומוצגים בממשק המשתמש

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה

- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

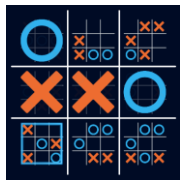
מחלקת GamesViewModel

- שם המחלקה: GamesViewModel



- תפקיד המחלקה: ניהול הלוגיקה ומצב המשחק עבור סוגי משחקים שונים (Replay, Online, Local, Cpu). מקשר בין ה-Repositories לבין ה-UI באמצעות LiveData ומטפל באירועי התחלה, סיום, ומהלכים.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
lvErrorCode	MediatorLiveData<Integer>	private final	LiveData לשמירת קוד שגיאה	דיווח שגיאות לממשק המשתמש
lvGame	MediatorLiveData<Game>	private final	LiveData למצב הנוכחי של המשחק	צפייה ועדכון מצב המשחק
lvOuterBoardWinners	MediatorLiveData<char[]>	private final	LiveData לשמירת מצב מנצחי הלוח החיצוני	עדכון ומעקב אחר מנצחי לוחות פנימיים
lvIsStarted	MediatorLiveData<Boolean>	private final	LiveData לציון האם המשחק התחיל	מעקב אחר התחלת המשחק
lvIsFinished	MediatorLiveData<Boolean>	private final	LiveData לציון האם המשחק הסתיים	מעקב אחר סיום המשחק
lvGameIdFs	MediatorLiveData<String>	private final	LiveData-ID של המשחק ב-Firebase-Online למשחקים (ne)	יצירת Monitor Service על המשחק
gameType	GameType	private final	סוג המשחק הנבחר (Cpu, Local, Online, Replay)	יצירת Repository מהסוג המתאים



נקודת הגישה לכל פעולות ה-CRUD והלוגיקה של המשחק בהתאם לסוג	Repository ה- המנוהל על-ידי ה-ViewModel	private	BaseGamesRepository	repository
--	---	---------	---------------------	------------

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

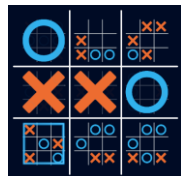
מחלקת GamesViewModelFactory

- שם המחלקה: GamesViewModelFactory
- תפקיד המחלקה: יצירת מופעים של GamesViewModel עם סוג המשחק המתאים. יורשת מ-GenericViewModelFactory ומוסיפה לוגיקה לבניית GamesViewModel עם הפרמטר הנדרש.

- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
gameType	GameType	private final	סוג המשחק שיוזן ל-GamesViewModel	עובר לבנאי של GamesViewModel כדי לקבוע את סוג המשחק

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.



מחלקת UsersViewModel

- שם המחלקה: UsersViewModel
- תפקיד המחלקה: ניהול פעולות משתמשים כגון אימות, בדיקת קיום משתמש והפקת רשימת השחקנים המובילים, תוך העברת LiveData ל-UI.
- תכונות המחלקה:

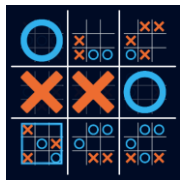
שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
Repository	UsersRepository	private	מופע של Repository לביצוע פעולות CRUD ושאלות משתמשים	משמש לקריאה למתודות exist, getTopPlayersPaginated, ו-Login

- הסבר על פעולות המחלקה
- פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.

מחלקת AppMonitorService

- שם המחלקה: AppMonitorService
- תפקיד המחלקה: שירות רקע המנטר את מצב המשחק באפליקציה ומציג התראה במצב foreground כשהמשחק פעיל. מטפל בניקוי נכון של המשחק כאשר המשתמש סוגר את האפליקציה או מוציא אותה מרשימת המשימות האחרונות.
- תכונות המחלקה:

שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
EXTRA_IN_GAME	String	private static final	מפתח לאקסטרה שמציין אם המשתמש נמצא במסך המשחק	מועבר ב-Intent ל-Service כדי לעדכן האם המשחק פעיל
EXTRA_GAME_ID_FS	String	private	מפתח לאקסטרה של מזהה המשחק ב-Firestore	מועבר ב-Intent לשירות כדי לשמור ולחזור עליו בהפעלות שונות



		static final		
מועבר ב- Intent מידע על השחקן הראשון	מפתח לאקסטרה של מזהה שחקן 1 ב- Firestore	private static final	String	EXTRA_PLAYER1_ID_ FS
מועבר ב- Intent מידע על השחקן השני	מפתח לאקסטרה של מזהה שחקן 2 ב- Firestore	private static final	String	EXTRA_PLAYER2_ID_ FS
מועבר ב- Intent לשירות כדי לקבוע האם השחקן הוא ההוסט	מפתח לאקסטרה המציין אם המשתמש הוא שחקן 1	private static final	String	EXTRA_IS_PLAYER1
נבדק ב- onTaskRemov עד כדי להחליט אם לקרוא ל- exitGameDirec תלפני עצירת השירות	מציין האם המשתמש כרגע במסך המשחק	private static	Boolean	inGameActivity
מועבר כפרמטר ל- exitGameDirec t	מזהה המשחק הנוכחי ב- Firestore	private static	String	gameId
מועבר כפרמטר ל- exitGameDirec t	מזהה שחקן 1 במסד הנתונים	private static	String	player1Id
מועבר כפרמטר ל- exitGameDirec t	מזהה שחקן 2 במסד הנתונים	private static	String	player2Id



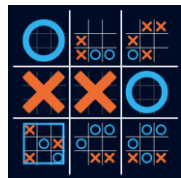
משמש בלוגיקת הסיום כדי לדעת מי יצטרך להודיע לשרת על סיום המשחק	מציין אם המשתמש הוא שחקן 1	private static	Boolean	isPlayer1
משמש לקריאה ל- exitGameDirec ולפעולות נוספות על המשחקים האונליין	מופע של ה- Repository לניהול משחקים אונליין	private	OnlineGamesRepository	repository

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה).
- מחלקת `TextInputLayoutUtil`
 - שם המחלקה: `TextInputLayoutUtil`
 - תפקיד המחלקה: מחלקת עזר המאפשרת להעביר הודעת שגיאה מ-`EditText` אל ה-`Layout` העוטף אותו ולמצוא את ה-`Layout` האב באופן גנרי.
 - תכונות המחלקה: אין
- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצורף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) (יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה).
- מחלקת `UserSessionPreference`
 - שם המחלקה: `UserSessionPreference`
 - תפקיד המחלקה: מחלקה לניהול נתוני סשן של המשתמש באמצעות `SharedPreferences`, כולל שמירת פרטי התחברות, מזהה המשתמש מ-`Firestore` וטוקן אימות. המחלקה יורשת מ-`LoginPreference`.
 - תכונות המחלקה:



שם התכונה	סוג התכונה	תחום ההכרה	תפקיד	שימוש
KEY_USER_ID_FS	String	private static final	פתח לשמירת מזהה המשתמש מ־Firestore בתוך SharedPreferences	שמירת מזהה המשתמש SharedPreferences
KEY_AUTH_TOKEN	String	private static final	מפתח לשמירת טוקן האימות בתוך SharedPreferences	שמירת הטוקן SharedPreferences
sharedPreferences	SharedPreferences	private final	אחראי על גישה לקובץ SharedPreferences של ההעדפות שנשמרות באפליקציה	מאותחל מה־superמנוהל לאורך חיי המחלקה

- הסבר על פעולות המחלקה
פעולות לוגיות עיקריות בצרוף קוד הפעולה
- מחלקות אשר עוסקות במסדי נתונים (כדוגמת: רשימת פריטים, משתמשים...) יש להציג תאור של המידע כפי שנשמר בקובץ / טבלה.



בסיס נתונים

Games > 1CwUBNN1omeffj20osRP			More in Google Cloud
(default)	Games	1CwUBNN1omeffj20osRP	
+ Start collection	+ Add document	+ Start collection	
Games	1CwUBNN1omeffj20osRP	+ Add field	
Users	2mVGBFrefx1YH4HJ3ZRTp	crossPlayerIdFs: "KiZ17v2w5GA7nXM9Dfbn"	
	4nFDhd8SGNtjA21Avoy4	currentPlayerIdFs: "dY4VJZbT5ZLM9yacrkck"	
	9iYfU0YrjwSR4LpUquro	finished: true	
	AbZpS6tsnfo7PlymEmE5	idFs: "1CwUBNN1omeffj20osRP"	
	CQ0IbnXhSVjnfYmoVuda	moves: [{"inner: {x: 2, y: 0, stab...}]	
	DM4emysMyfxbKfKpDuNR	player1	
	DZfCGMCe8oqjN6vOGWKz	elo: 1236.6959228515625	
	FnlcRQUChTFN448e1bDr	idFs: "dY4VJZbT5ZLM9yacrkck"	
	G9iPZYtDyn1idOP4eoIr	name: "Uni"	
	Hf4hQGvsbzzT0wYhwrus	player2	
	I2PLAxca8SB3LjIFW7AK	elo: 1171	
	IQfc18zEjVyJt7uHr5hN	idFs: "KiZ17v2w5GA7nXM9Dfbn"	
	Ifa3PxHFYxIfEqQ8C1ea	name: "Emil.za07"	
	O4Ga41IE4Wr6Gd8Yhzvt	started: true	
	PYWdaNAQALtnmbu1paQV	startedAt: 15 May 2025 at 23:44:14 UTC+3	
	QI6wrVoRvR11hxdIA5BP	winnerIdFs: "dY4VJZbT5ZLM9yacrkck"	
	R20oD5juXTQXeqvCxn8g		
	RhbGtsPQzRY3GsZsnabM		

סקירת בסיס הנתונים

בסיסי הנתונים בהם השתמשתי היו Firestore ו-SharedPreferences.

בסיס הנתונים העיקרי באפליקציה הוא Firestore מכיוון שזה בסיס נתונים בענן המאפשר שמירה של נתונים בצורה מאוד נוחה. ב-Firebase אני שומר את כל המידע על המשחקים כולל מי שיחק, מה היו המהלכים, מי ניצח, ומתי התחיל המשחק, בכל הנתונים האלה אני משתמש פעמיים, בפעם הראשונה בזמן המשחק עצמו - כל שחקן, בתורו מעדכן את המשחק הרלוונטי, והשחקן השני קולט את זה ומעדכן את המשחק אצלו. בפעם השנייה אני משתמש במסמכים הללו בשביל הפונקציה "צפיה חוזרת" איפה שהמשתמש יכול לצפות במשחקים שהוא שיחק בעבר.

בנוסף אני משתמש ב-Firebase בשביל שמירת המשתמשים של האפליקציה, באוסף הזה אני שומר את כל המידע על המשתמש כולל סיסמה מוצפנת בעזרת פונקציית hash.

השימוש ב-SharedPreferences נמצא בתחילת האפליקציה במסך ההתחברות איפה שהמשתמש יכול לבקש מהאפליקציה לשמור את פרטי ההתחברות שלו בשביל התחברות מהירה בפעמים הבאות. מכיוון שפרטיות ובטיחות משתמש חשובה לי, כל הפרטים שלו נשמרים ב-EncryptedSharedPreferences על מנת להקשות על אפליקציות זדוניות לגנוב מידע על המשתמש או לערוך אותו (לצורך העלאת הדירוג לדוגמה).



בנוסף אני משתמש בSharedPrefs במסך הפתיחה איפה שהאפליקציה כבר בטעינה בודקת האם המשתמש השתמש באפשרות "זכור אותי", שם, לאחר מכן, אם המשתמש אכן ביקש מהאפליקציה לזכור אותו, האפליקציה מוודאת שפרטי המשתמש לא שונו מהכניסה האחרונה של המשתמש (רלוונטי מתי שאוסיף אפשרות לשנות שם משתמש/סיסמה).



FirestoreInstance

```
private FirebaseFirestore (Context context) {  
    FirebaseOptions options = new FirebaseOptions.Builder()  
        .setProjectId("ultimate-tic-tac-toe-project")  
        .setApplicationId("1:273203212742:android:5e05863b0c6b3a1722c2c7")  
        .setApiKey("AlzaSyCqKjSsYdTE9JuwOTWBxpm55qxTiUFNQxs")  
        .setStorageBucket("ultimate-tic-tac-toe-project.firebaseio.com")  
        .build();  
  
    app = FirebaseApp.initializeApp(context, options, "ultimate-tic-tac-toe-project");  
}
```

יצירת אובייקט של Firebase בשביל קישור בין האפליקציה לבסיס נתונים

BaseGamesRepository

```
@Override protected Query getQueryForExist(Game entity) {  
    return getCollection().whereEqualTo("idFs", entity.getIdFs());  
}
```

OnlineGamesRepository

```
public LiveData<Games> getUserGamesPaginated(String userIdFs, int limit, String startAfterIdFs) {  
    if (startAfterIdFs != null) {  
        getCollection().document(startAfterIdFs).get()  
            .addOnSuccessListener(documentSnapshot -> {  
                if (documentSnapshot.exists()) {  
                    Timestamp timestamp = documentSnapshot.getTimestamp("startedAt");  
                    if (timestamp != null)  
                        executeQueriesWithTimestamp(userIdFs, limit, timestamp, lvGames);  
                }  
            })  
    }  
}
```



```
        lvGames.setValue(new Games());
    } else
        lvGames.setValue(new Games());
    })
    .addOnFailureListener(e -> lvGames.setValue(new Games()));
} else {
    executeInitialQueries(userIdFs, limit, lvGames);
}

return lvGames;
}

private void executeInitialQueries(String userId, int limit, MutableLiveData<Games> result) {
    executeQueriesWithTimestamp(userId, limit, new Timestamp(253402300799L, 999999999), result); //TODO: use
    a better timestamp?
}

private void executeQueriesWithTimestamp(String userId, int limit, Timestamp startAfter,
MutableLiveData<Games> result) {
    Games allGames = new Games();

    getCollection()
        .whereEqualTo("player1.idFs", userId)
        .orderBy("startedAt", Query.Direction.DESENDING)
        .startAfter(startAfter)
        .limit(limit)
        .get()
        .addOnSuccessListener(player1Snapshot -> {
            for (DocumentSnapshot doc : player1Snapshot.getDocuments()) {
                Game game = doc.toObject(OnlineGame.class);
                if (game != null) {
                    game.setIdFs(doc.getId());
                    allGames.add(game);
                }
            }
        })

    getCollection()
        .whereEqualTo("player2.idFs", userId)
```



```
.orderBy("startedAt", Query.Direction.DESCENDING)
.startAfter(startAfter)
.limit(limit)
.get()
.addOnSuccessListener(player2Snapshot -> {
    for (DocumentSnapshot doc : player2Snapshot.getDocuments()) {
        Game game = doc.toObject(OnlineGame.class);
        if (game != null) {
            game.setIds(doc.getId());
            allGames.add(game);
        }
    }
})

allGames.sort((g1, g2) -> ((OnlineGame) g2).getStartedAt().compareTo(((OnlineGame)
g1).getStartedAt()));
Games limitedGames = new Games();
for (int i = 0; i < Math.min(limit, allGames.size()); i++) {
    limitedGames.add(allGames.get(i));
}
result.setValue(limitedGames);
})
.addOnFailureListener(e -> {
    allGames.sort((g1, g2) -> ((OnlineGame) g2).getStartedAt().compareTo(((OnlineGame)
g1).getStartedAt()));
    result.setValue(allGames);
});
})
.addOnFailureListener(e -> result.setValue(new Games()));
}

public void startGame(Player player, String crossPlayerIds) {
    try {
        Games games = Tasks.await(getNotStartedGames());
        String gameId = getGameIdWithSimilarElo(player.getElo(), games);
        Tasks.await(joinGame(gameId, player));
        addSnapshotListener(gameId);
    } catch (ExecutionException | NoSuchElementException e) {
```



```
Throwable cause = e.getCause();
if (cause instanceof EmptyQueryException || e instanceof NoSuchElementException) {
    hostOnlineGame(player).continueWith(gameld -> {
        addSnapshotListener(gameld.getResult());
        return null;
    });
} else if (cause instanceof GameFullException && retryCounter < 10) {
    retryCounter++;
    startGame(player, null);
}
} catch (InterruptedException e) {
    lvErrorCode.setValue(4);
}
}

private Task<Games> getNotStartedGames() {
    TaskCompletionSource<Games> taskGetNotStartedGames = new TaskCompletionSource<>();
    Games games = new Games();

    getQueryForNotStarted().get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            for (QueryDocumentSnapshot document : queryDocumentSnapshots) {
                OnlineGame game = document.toObject(OnlineGame.class);
                if (Objects.equals(game.getPlayer1().getIds(), localPlayerIds))
                    delete(game);
                else
                    games.add(game);
            }
            games.sort((g1, g2) -> g1.getPlayer1().compareElo(g2.getPlayer1()));
            if (!games.isEmpty())
                taskGetNotStartedGames.setResult(games);
            else
                taskGetNotStartedGames.setException(new EmptyQueryException());
        })
        .addOnFailureListener(taskGetNotStartedGames::setException);
    return taskGetNotStartedGames.getTask();
}
```



```
private Query getQueryForNotStarted() {
    return getCollection().whereEqualTo("started", false).whereEqualTo("player2.idFs", "");
}

public Task<String> hostOnlineGame(Player player) {
    TaskCompletionSource<String> taskCreateGame = new TaskCompletionSource<>();
    OnlineGame game = new OnlineGame(player);
    add(game)
        .addOnSuccessListener(aBoolean -> {
            lvGame.setValue(game);
            taskCreateGame.setResult(game.getIdFs());
            lvGameIdFs.setValue(game.getIdFs());
        })
        .addOnFailureListener(taskCreateGame::setException);
    return taskCreateGame.getTask();
}

public void beginOnlineGame() {
    OnlineGame game = (OnlineGame) lvGame.getValue();
    game.setStarted(true);
    getCollection().document(lvGame.getValue().getIdFs())
        .update("started", true, "startedAt", FieldValue.serverTimestamp())
        .addOnSuccessListener(voidTask -> getCollection().document(lvGame.getValue().getIdFs()).get())
        .addOnSuccessListener(documentSnapshot -> {
            if (documentSnapshot.exists()) {
                Timestamp serverTimestamp = documentSnapshot.getTimestamp("startedAt");
                ((OnlineGame) lvGame.getValue()).setStartedAt(serverTimestamp);
            }
        });
    lvIsStarted.setValue(true);
}

public Task<Void> joinGame(String gameId, Player player) {
    Map<String, Object> data = new HashMap<>();
    Gson gson = new Gson();
    String json = gson.toJson(player);
    data.put("game_id", gameId);
}
```



```
data.put("player", json);
```

```
return function
    .getHttpsCallable("join_game")
    .call(data).continueWith(task -> {
        if (task.getException() instanceof FirebaseFunctionsException) {
            if (((FirebaseFunctionsException) task.getException()).getStatusCode() ==
                FirebaseFunctionsException.Code.ABORTED)
                throw new GameFullException();
        } else if (!task.isSuccessful()) {
            throw task.getException();
        }
        return null;
    });
}
```

```
public void addSnapshotListener(String gameIdFs) {
    final DocumentReference gameRef = getCollection().document(gameIdFs);
    gameRef.addSnapshotListener((snapshot, e) -> {
        if (e != null || snapshot == null) return;

        boolean isLocal = snapshot.getMetadata().hasPendingWrites();
        if (isLocal || !snapshot.exists()) {
            if (!isLocal) {
                lvGame.setValue(null); // The document does not exist (deleted)
            }
            return;
        }
    })
}
```

```
OnlineGame game = snapshot.toObject(OnlineGame.class);
```

```
boolean isStarted = game.isStarted();
```

```
boolean isFinished = game.isFinished();
```

```
List<BoardLocation> moves = game.getMoves();
```

```
try {
    if (!isStarted && !game.getPlayer2().getIdFs().isEmpty() &&
        Objects.equals(game.getPlayer1().getIdFs(), localPlayerIdFs)) {
```




```
//the joiner joined the game
lvGame.setValue(game);
beginOnlineGame();
} else if (isStarted && lvGame.getValue() == null && (moves.isEmpty())) {
    //the Host started the game
    lvGame.setValue(game);
    ((OnlineGame) (lvGame.getValue())).startGameForJoiner();
    lvIsStarted.setValue(true);
} else if (isFinished && moves.size() == lvGame.getValue().getMoves().size()) {
    //the opponent resigns
    lvGame.setValue(game);
    lvIsFinished.setValue(true);
} else if (moves != null && !moves.isEmpty()) {
    //it's a move and send it to handle a move
    BoardLocation lastMove = moves.get(moves.size() - 1);
    lvGame.getValue().makeMove(lastMove);
    lvGame.setValue(lvGame.getValue());
    this.checkInnerBoardFinish(lastMove.getOuter());
} else if (((OnlineGame) lvGame.getValue()).getStartedAt() == null) {
    Timestamp serverTimestamp = snapshot.getTimestamp("startedAt");
    ((OnlineGame) lvGame.getValue()).setStartedAt(serverTimestamp);
}
} catch (Exception ex) {
    //intentionally empty
}
});
}

private Task<Void> finishGame(String player1IdFs, String player2IdFs) {
    Map<String, Object> data = new HashMap<>();
    data.put("player_1_id", player1IdFs);
    data.put("player_2_id", player2IdFs);
    data.put("score", Objects.equals(lvGame.getValue().getWinnerIdFs(), "T") ? 0.5 :
        (lvGame.getValue().getWinnerIdFs().equals(player1IdFs) ? 1.0 : 0.0));
    return function
        .getHttpsCallable("finish_game")
        .call(data).continueWith(task -> {
            Exception ex = task.getException();
```



```
if (ex instanceof FirebaseFunctionsException &&
    ((FirebaseFunctionsException) ex).getStatusCode() == FirebaseFunctionsException.Code.ABORTED) {
    throw new FirebaseTooManyRequestsException("");
}
if (ex != null)
    throw ex;
return null;
});
}
```

```
public Task<Boolean> exitGame() {
    TaskCompletionSource<Boolean> taskAbortGame = new TaskCompletionSource<>();
    Game game = lvGame.getValue();
    if (game == null || game.getWinnerIds() != null) {
        taskAbortGame.setResult(true);
    } else if (game.getMoves().size() < 2) {
        delete(game)
            .addOnSuccessListener(aBoolean -> {
                lvGame.setValue(null);
                taskAbortGame.setResult(true);
            })
            .addOnFailureListener(e -> taskAbortGame.setResult(false));
    } else {
        game.setFinished(true);
        game.setWinnerIds(
            Objects.equals(game.getPlayer1().getIds(), localPlayerIds) ?
                game.getPlayer2().getIds() :
                game.getPlayer1().getIds());
        update(game);
        finishGame(game.getPlayer1().getIds(), game.getPlayer2().getIds())
            .addOnSuccessListener(aBoolean -> lvIsFinished.setValue(true))
            .addOnFailureListener(e -> {
            });
        taskAbortGame.setResult(true);
    }
    return taskAbortGame.getTask();
}
```



```
public Task<Boolean> exitGameDirect(String gameId, String player1Ids, String player2Ids, boolean isPlayer1) {
    TaskCompletionSource<Boolean> taskAbortGame = new TaskCompletionSource<>();

    if (gameId == null) {
        taskAbortGame.setResult(true);
        return taskAbortGame.getTask();
    }

    getCollection().document(gameId).get()
        .addOnSuccessListener(documentSnapshot -> {
            if (documentSnapshot.exists()) {
                OnlineGame game = documentSnapshot.toObject(OnlineGame.class);

                if (game != null) {
                    if (game.getMoves().isEmpty()) {
                        getCollection().document(gameId).delete()
                            .addOnSuccessListener(aVoid -> taskAbortGame.setResult(true))
                            .addOnFailureListener(e -> taskAbortGame.setResult(false));
                    } else {
                        String winnerId = isPlayer1 ? player2Ids : player1Ids;
                        Map<String, Object> updates = new HashMap<>();
                        updates.put("finished", true);
                        updates.put("winnerIds", winnerId);

                        getCollection().document(gameId).update(updates)
                            .addOnSuccessListener(aVoid -> {
                                Map<String, Object> data = new HashMap<>();
                                data.put("player_1_id", player1Ids);
                                data.put("player_2_id", player2Ids);
                                data.put("score", isPlayer1 ? 0.0 : 1.0); // Player who quit loses

                                function.getHttpsCallable("finish_game")
                                    .call(data)
                                    .addOnSuccessListener(result -> taskAbortGame.setResult(true))
                                    .addOnFailureListener(e -> taskAbortGame.setResult(false));
                            })
                            .addOnFailureListener(e -> taskAbortGame.setResult(false));
                    }
                }
            }
        });
}
```



```
    }  
  } else {  
    taskAbortGame.setResult(true);  
  }  
  } else {  
    taskAbortGame.setResult(true);  
  }  
})  
.addOnFailureListener(e -> taskAbortGame.setResult(false));  
  
return taskAbortGame.getTask();  
}
```

UserRepository

```
@Override  
protected Query getQueryForExist(User entity) {  
  return getCollection().whereEqualTo("idFs", entity.getIdFs());  
}  
  
public Task<Boolean> exist(String username) {  
  TaskCompletionSource<Boolean> tcs = new TaskCompletionSource<>();  
  Query query = getCollection().whereEqualTo("username", username);  
  
  get(query)  
    .addOnSuccessListener(savedEntity -> {  
      if (savedEntity != null && username.equals(savedEntity.getUsername())) {  
        tcs.setResult(true);  
        return;  
      }  
      tcs.setResult(false);  
    })  
    .addOnFailureListener(tcs::setException);  
  
  return tcs.getTask();  
}
```



```
public LiveData<Users> getTopPlayersPaginated(int limit, float lastElo, String lastIdFs) {  
    Query query = getCollection()  
        .orderBy("elo", Query.Direction.DESENDING)  
        .orderBy("idFs")  
        .limit(limit);  
  
    if (lastElo != -1) {  
        query = query.startAfter(lastElo, lastIdFs);  
    }  
  
    query.get().addOnSuccessListener(queryDocumentSnapshots -> {  
        Users users = new Users();  
        for (DocumentSnapshot doc : queryDocumentSnapshots.getDocuments()) {  
            User user = doc.toObject(User.class);  
            if (user != null) {  
                users.add(user);  
            }  
        }  
        lvTopUsers.setValue(users);  
    });  
    return lvTopUsers;  
}
```



GamesViewModel

```
public void getUserGamesPaginated(String userIdFs, int limit, String startAfterIdFs) {
    if (repository instanceof OnlineGamesRepository)
        lvCollection = ((OnlineGamesRepository) repository).getUserGamesPaginated(userIdFs, limit, startAfterIdFs);
}

public void startOnlineGame(Player player) {
    setGameIdFsObserver();
    Executors.newSingleThreadExecutor().execute() -> repository.startGame(player, null);
}

public void exitGame() {
    ((OnlineGamesRepository) repository).exitGame().addOnSuccessListener(aBoolean ->
lvSuccess.setValue(aBoolean))
        .addOnFailureListener(e -> lvSuccess.setValue(false));
}

public void makeMove(BoardLocation boardLocation) {
    repository.makeMove(boardLocation)
        .addOnSuccessListener(voidTask -> {
            if (repository instanceof OnlineGamesRepository) {
                repository.update(lvGame.getValue());
            } else if (repository instanceof CpuGamesRepository && !lvGame.getValue().isFinished()) {
                Executors.newSingleThreadExecutor().execute() -> ((CpuGamesRepository) repository).makeCpuMove();
            }
        })
        .addOnFailureListener(e -> lvErrorCode.setValue(Integer.valueOf(e.getMessage())));
}
```

UsersViewModel

```
public void getTopPlayersPaginated(int limit, float lastElo, String lastIdFs) {
    lvCollection = repository.getTopPlayersPaginated(limit, lastElo, lastIdFs);
}

public void login(String Username, String password) {
```



```
repository.getCollection().whereEqualTo("username", Username).get()
    .addOnSuccessListener(queryDocumentSnapshots -> {
        if (!queryDocumentSnapshots.isEmpty()) {
            if (PasswordUtil.verifyPassword(password,
                queryDocumentSnapshots.getDocuments().get(0).getString("hashedPassword"))) {
                User user = queryDocumentSnapshots.getDocuments().get(0).toObject(User.class);
                lvEntity.setValue(user);
            }
            else {
                lvSuccess.setValue(false);
            }
        } else {
            lvSuccess.setValue(false);
        }
    })
    .addOnFailureListener(e -> lvEntity.setValue(null));
}

public void exist(String username) {
    repository.exist(username)
        .addOnSuccessListener(lvExist::setValue)
        .addOnFailureListener(e -> lvExist .setValue(false));
}
```

AuthActivity

```
private void checkForLogin() {
    if (sessionPreference.hasValidSession()) {
        String userIdFs = sessionPreference.getUserIdFs();
        viewModel.get(userIdFs);
    } else {
        hideProgressDialog();
    }
}
```

RegisterActivity



```
@Override
protected void setListeners() {
    btnRegister.setOnClickListener(v -> {
        if (validate()) {
            viewModel.exist(etUsername.getText().toString());
        }
    });
    btnBack.setOnClickListener(v -> finish());
}

protected void registerUser() {
    Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.default_pfp);
    String hashedPassword = PasswordUtil.hashPassword(etPassword.getText().toString());
    User user = new User(etUsername.getText().toString(),
        hashedPassword,
        BitMapHelper.encodeTobase64(bitmap));
    viewModel.save(user);
    Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
    startActivity(intent);
    finish();
}
```




LoginActivity

```
@Override
protected void setListeners() {
    btnSignIn.setOnClickListener(v -> {
        if (validate()) {
            String username = etUsername.getText().toString();
            String password = etPassword.getText().toString();
            viewModel.logIn(username, password);
        }
    });
    btnBack.setOnClickListener(v -> finish());
}
```

MainActivity

```
@SuppressWarnings("ConstantConditions")
private void registerLaunchers() {
    gameLauncher = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(),
        o -> {
            GameType gameType = (GameType) o.getData().getSerializableExtra(EXTRA_GAME_TYPE);
            if (gameType == GameType.ONLINE && o.getResultCode() == RESULT_OK) {
                viewModel.get(currentUser.getIdFs());
            }
        }
    );
    profileLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == RESULT_OK) {
                ivProfile.setImageBitmap(currentUser.getPictureBitmap());
            }
        }
    );
}
```



ProfileActivity

@Override

protected void setViewModel() {

```
    gamesViewModel = new ViewModelProvider(this, new GamesViewModelFactory(getApplication(),
    GameType.ONLINE)).get(GamesViewModel.class);
    usersViewModel = new ViewModelProvider(this).get(UsersViewModel.class);
    loadGames(false);
```

```
    usersViewModel.getLiveDataSuccess().observe(this, success -> {
        if (success) {
            ivPfp.setImageBitmap(currentUser.getPictureBitmap());
            setResult(RESULT_OK);
        } else {
            usersViewModel.get(currentUser.getIdFs());
        }
    });
```

```
    usersViewModel.getLiveDataEntity().observe(this, user -> {
        if (user != null) {
            for (Game game : games) {
                if (game == null) continue;
                if (Objects.equals(currentUser.getIdFs(), game.getPlayer1().getIdFs())) {
                    game.getPlayer2().setPicture(user.getPicture());
                } else {
                    game.getPlayer1().setPicture(user.getPicture());
                }
            }
        }
        adapter.setItems(this.games);
    });
```

```
    gamesViewModel.getLiveDataCollection().observe(this, newGames -> {
        if (adapter.getItems() != null) {
            int lastIndex = adapter.getItems().indexOf(null);
            if (lastIndex != -1) {
                adapter.getItems().remove(lastIndex);
                adapter.notifyItemRemoved(lastIndex);
            }
        }
    });
```



```
    }
}

if (!newGames.isEmpty()) {
    if (this.games == null)
        this.games = newGames;
    else
        this.games.addAll(newGames);
    isLoading = false;
    lastLoadedGameId = newGames.get(newGames.size() - 1).getIdFs();
    for (Game game : newGames) {
        usersViewModel.get(Objects.equals(currentUser.getIdFs(),
            game.getPlayer1().getIdFs()) ? game.getPlayer2().getIdFs() :
            game.getPlayer1().getIdFs());
    }
}
});
}

private void loadGames(boolean loadMore) {
    isLoading = true;
    if (loadMore) {
        showLoadingMore();
    }
    gamesViewModel.getUserGamesPaginated(currentUser.getIdFs(), PAGE_SIZE, lastLoadedGameId);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == UCrop.REQUEST_CROP && resultCode == RESULT_OK && data != null) {
        Uri resultUri = UCrop.getOutput(data);
        try {
            Bitmap croppedBitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), resultUri);
            // Now update the image view and save to user profile
            ivPfp.setImageBitmap(croppedBitmap);
            String base64Image = BitmapHelper.encodeTobase64(croppedBitmap);
            currentUser.setPicture(base64Image);
        }
    }
}
```



```
usersViewModel.update(currentUser);
} catch (IOException e) {
    Toast.makeText(this, "Failed to load cropped image", Toast.LENGTH_SHORT).show();
}
}
}
```

LeaderBoard

```
private void loadUsers(boolean loadMore) {
    isLoading = true;
    if (loadMore) {
        showLoadingMore();
    }
    viewModel.getTopPlayersPaginated(PAGE_SIZE, lastLoadedElo, lastLoadedIds);
}
```

GameActivity

```
getOnBackPressedDispatcher().addCallback(this, new OnBackPressedCallback(true) {
    @Override
    public void handleOnBackPressed() {
        if (gameType == GameType.REPLAY) {
            finish();
            return;
        }
        Game game = gamesViewModel.getLiveDataGame().getValue();
        AlertUtil.alert(
            GameActivity.this,
            (game != null && game.getMoves().size() > 1) ? "Resign" : "Abort",
            "Are you sure you want to exit the game?",
            true,
            0,
            "Yes",
            "No",
            null,

```



```
() -> {
    if (gameType == GameType.ONLINE)
        gamesViewModel.exitGame();
    setResult((game != null && game.getMoves().isEmpty()) ? RESULT_OK : RESULT_CANCELED, intent);
    if (gameType != GameType.ONLINE || game == null)
        finish();
}},
null,
null
);
}
});

private void handleBoardButtonClick(Imageview btn) {
    String tag = (String) btn.getTag();
    // Handle button click using the tag (e.g., "btn0101" for 1st outerRow, 2st outerColumn and 1th innerRow, 2st
    innerColumn)
    gamesViewModel.makeMove(new BoardLocation(tag.charAt(3) - '0', tag.charAt(4) - '0', tag.charAt(5) - '0',
    tag.charAt(6) - '0'));
}

gamesViewModel.getLiveDataIsStarted().observe(this, aBoolean -> {
    if (aBoolean) {
        Game game = gamesViewModel.getLiveDataGame().getValue();
        boolean isHost = Objects.equals(game.getPlayer1().getIdFs(), currentUser.getIdFs());

        if (gameType == GameType.ONLINE || gameType == GameType.REPLAY)
            usersViewModel.get(isHost ? game.getPlayer2().getIdFs() : game.getPlayer1().getIdFs());
        else {
            setPlayers(game.getPlayer1(), game.getPlayer2());
            clLoading.setVisibility(View.GONE);
            tvCurrentPlayer.setVisibility(View.VISIBLE);
            gridBoard.setBackground(ResourcesCompat.getDrawable(getResources(), R.drawable.board, null));
        }

        if (gameType == GameType.ONLINE) {
            String gameIdFs = game.getIdFs();
            String player1IdFs = game.getPlayer1().getIdFs();
        }
    }
});
```



```
String player2IdFs = game.getPlayer2().getIdFs();

if (!monitorServiceStarted) {
    monitorServiceStarted = true;
    AppMonitorService.startService(this, true, gameIdFs, player1IdFs, player2IdFs, isHost);
} else {
    AppMonitorService.updateGameState(true, gameIdFs, player1IdFs, player2IdFs, isHost);
}
}
}
});

@Override
protected void onDestroy() {
    super.onDestroy();
    AppMonitorService.userClosedActivity(this);
}

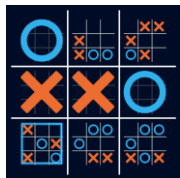
private void gameInit(GameType gameType) {
    switch (gameType) {
        case CPU:
            char sign = intent.getCharExtra(MainActivity.EXTRA_SIGN, 'X');
            gamesViewModel.startCpuGame(sign == 'X' ? currentUser.getIdFs() : "CPU");
            break;
        case LOCAL:
            gamesViewModel.startLocalGame();
            break;
        case ONLINE:
            try {
                clLoading.setVisibility(View.VISIBLE);
                gamesViewModel.startOnlineGame(new Player(currentUser));
                setPlayers(new Player(currentUser), null);
            } catch (Exception e) {
                Toast.makeText(this, R.string.game_connection_error, Toast.LENGTH_SHORT).show();
                setResult(RESULT_CANCELED, intent);
                finish();
            }
            break;
    }
}
```



```
case REPLAY:
    gridBoard.setVisibility(View.INVISIBLE);
    gamesViewModel.get(intent.getStringExtra(MainActivity.EXTRA_GAME_ID_FS));
    break;
}
}
```

AppMonitorService

```
@Override
public void onTaskRemoved(Intent rootIntent) {
    super.onTaskRemoved(rootIntent);
    if (inGameActivity && gameIdFs != null) {
        repository.exitGameDirect(gameIdFs, player1IdFs, player2IdFs, isPlayer1)
            .addOnSuccessListener(a -> {
            })
            .addOnFailureListener(e -> {
            });
    }
    stopSelf();
}
```



מדריך למשתמש

פרק זה יכיל את ההסבר לדרישות ההתקנה בסביבת העבודה, הוראות התקנה.

(בסיום מחק/י) את 2 השורות למעלה

הוראות התקנה

יש להתקין את האפליקציה מחנות האפליקציות GooglePlay או לקבל קישור אליה. לאחר מיכן יש לאשר גישה לדברים הנחוצים.

גרסאות עליהן נבדקה האפליקציה

סוגי מכשירים

Redmi Note 11 Pro+ 5G API 33

Samsung S10e API 30

OnePlus 7 Pro API 30

Redmi A1+ API 31

אמולטורים

Android Studio באפליקציה Pixel – API 26

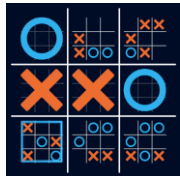
Android Studio באפליקציה Pixel 7 Pro – API 29

Android Studio באפליקציה Pixel 7 pro – API 34

Android Studio באפליקציה Pixel 9 pro – API 34

Android Studio באפליקציה Pixel 9 pro – API 35

Android Studio באפליקציה Pixel 9 pro XL– API 36





אופן פעולת האפליקציה

הסבר כללי

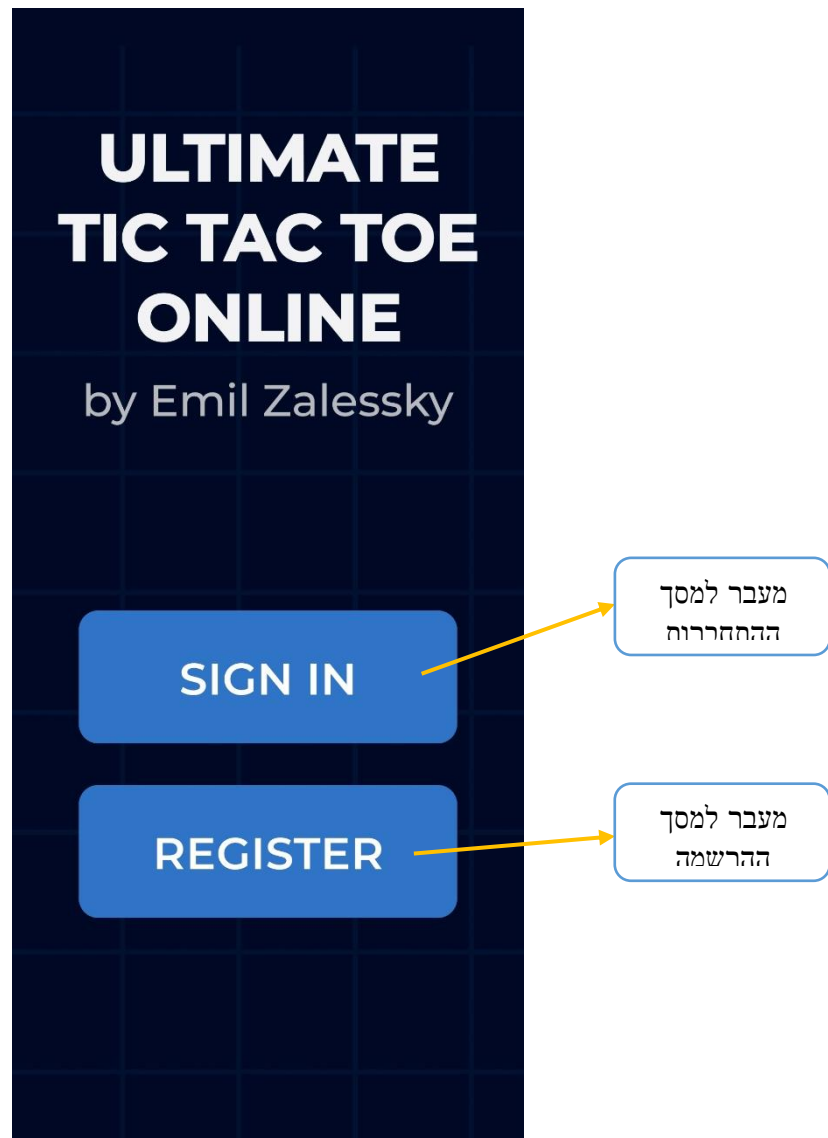
האפליקציה "סופר איקס עיגול" מציעה חוויית משחק אינטראקטיבית ומשולבת עם תכונות חברתיות ותחרותיות. המשתמש יכול לבחור באחד ממספר מצבי משחק – נגד המחשב, נגד שחקן נוסף באותו מכשיר (לוקאלי), או מול יריב אקראי דרך האינטרנט. לאחר בחירת מצב המשחק, מוצג לוח המשחק, והמשתמשים משחקים בתורות עד להכרעה או לתיקו.

במקביל, כל משתמש באפליקציה מנהל פרופיל אישי, שכולל שם, תמונת פרופיל, סטטיסטיקות ומד כושר. המשחקים נשמרים במאגר, וניתן לצפות בהם בדיעבד דרך מסך ההיסטוריה. תכונה זו מאפשרת ללמוד ממהלכים קודמים ולנתח את דרך המשחק. בנוסף, טבלת דירוג עולמית ודירוג אישי מספקים למשתמשים מוטיבציה להתקדם ולהתחרות מול אחרים. האפליקציה שומרת את פרטי המשתמש באופן מאובטח, מציעה חוויית שימוש נוחה ונגישה, ופועלת תוך שמירה על ביצועים טובים גם במכשירים חלשים יחסית.

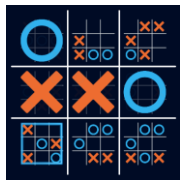


מסך AuthActivity

כאשר פותחים את האפליקציה, מגיעים למסך הפתיחה שבו מתבצעת טעינה לחשבון אם המשתמש בחר בעבר בפונקציה "זכור אותי". אם לא, אז המשתמש יכול לבחור לעבור לעמוד ההרשמה או לעמוד ההתחברות במידה



ויש לו כבר חשבון.



מסך RegisterActivity

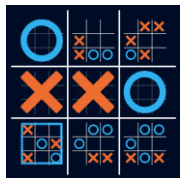
במידה והמשתמש בחר להירשם לאפליקציה, הוא מגיע למסך הזה. במסך יש 3 שדות אותם המשתמש חייב למלא – שם משתמש (בין 1-9 תווים), סיסמה (בין 8-64 תווים, עם אות גדולה, קטנה, מספר ותו מיוחד) ואימות סיסמה. במידה והמשתמש מזין משהו לא מתאים באחד השדות, מופיעה שגיאה ליד השדה עם הסבר על הבעיה. בנוסף המשתמש יכול ללחוץ על אייקון העין ליד השדה של הסיסמה ולראות את הסיסמה שהזין. בסיום התהליך,

The screenshot shows a dark blue background with a grid pattern. At the top, the word "REGISTER" is written in large white letters. Below it are three input fields: "Username", "Password", and "Confirm Password". The "Password" and "Confirm Password" fields have an eye icon to the right. Below the input fields are two large blue buttons: "REGISTER" and "Back".

Annotations in Hebrew boxes with arrows pointing to the UI elements:

- הכנסת שם משחמש (Username field)
- הצגת הסיסמה (Password field)
- הצגת אימות הסיסמה (Confirm Password field)
- ניסיון הרשמה לאפליקציה ובמידה וכל השדות תקינים, מעבר למסך (REGISTER button)
- חזרה למסך הפתיחה (Back button)
- הכנסת סיסמה (Password field)
- הכנסת אימות הסיסמה (Confirm Password field)

במידה וההרשמה הסתיימה בהצלחה, המשתמש יוכל לשמור את פרטי ההתחברות בחשבון גוגל שלו.



מסך LoginActivity

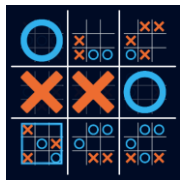
במידה והמשתמש בחר להתחבר לחשבון שלו או עבר ממסך ההרשמה, הוא מגיע למסך הזה. הכל מאוד דומה פה למסך ההתחברות, רק שבמידה והמשתמש מזין שם משתמש או סיסמה שגויים, הוא יקבל הודעה קופצת המציינת זאת. שדה הכנסת אימות הסיסמה הוחלף בתיבת הסימון בשביל פונקציית "זכור אותי" וכפתור ההרשמה הוחלף בכפתור ההתחברות.

The image shows a mobile app screen titled "SIGN IN" on a dark blue background with a grid pattern. The screen contains the following elements:

- Username** input field: Annotated with "הכנסת שם משתמש" (Entering username).
- Password** input field: Annotated with "הצגת הסיסמה" (Showing password).
- Remember me** checkbox: Annotated with "פונקציית 'זכור'" (Remember function).
- SIGN IN** button: Annotated with "ניסיון התחברות ובמידה וכל הפרטים נכונים, מעבר לאחד הרשאות" (Attempt login and if all details are correct, proceed to one of the permissions).
- Back** button: Annotated with "חזרה למסך הפתיחה" (Return to the start screen).

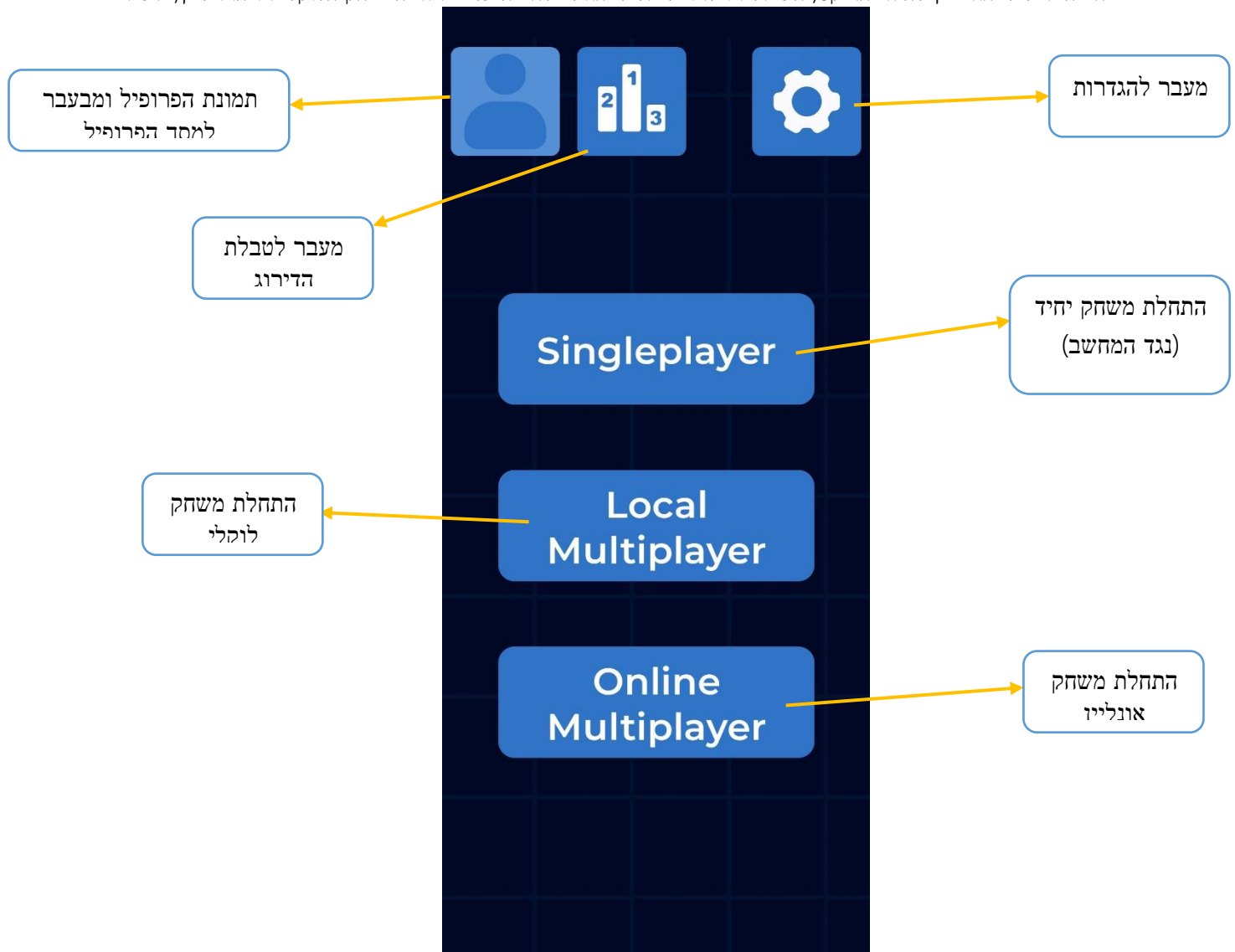
Additional annotations on the left side:

- Pointing to the Password field: "הכנסת סיסמה" (Entering password).
- Pointing to the Remember me checkbox: "פונקציית 'זכור'" (Remember function).
- Pointing to the Back button: "חזרה למסך הפתיחה" (Return to the start screen).



מסך MainActivity

לאחר התחברות מוצלחת, המשתמש מגיע למסך הזה. כאן הוא יכול לראות את תמונת הפרופיל שלו שבלחיצה עליה יעבור למסך הפרופיל שלו ו-5 כפתורים: טבלת הדירוג, הגדרות, משחק יחיד, משחק לוקלי ומשחק אונליין. בלחיצה על שני הכפתורים הראשונים, יועבר המשתמש למסכים הרלוונטיים, בלחיצה על כפתור המשחק היחיד, יקבל המשתמש תיבת דו שיח בה יוכל לבחור בין להיות האיקס, העיגול או לתת למזל לבחור. בלחיצה על כפתור המשחק הלוקלי או האונליין, יועבר

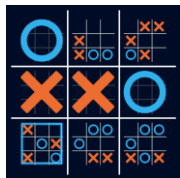


ישר המשתמש למסך המשחק.



מסך ProfileActivity


בפתיחת המסך מיד תתחיל טעינה של מספר משחקים אחרונים של המשתמש. במרכז המסך נמצאת תמונת הפרופיל של המשתמש, בלחיצה עליה יוכל לבחור מאיזה מקום לקחת תמונה להחלפה – מהמצלמה או מהגלריה, בשני המקרים אם זה השימוש הראשון בפונקציה הזאת, יצטרך לתת לאפליקציה גישה למצלמה/לגלריה בהתאמה למה שהוא בחר, לאחר מכן כאשר יצלם/ יבחר תמונה מתאימה, ידרש לחתוך את התמונה לריבוע כך שתתאים לפורמט של התמונת פרופיל, בחזרה למסך, אם הטעינה עברה בהצלחה, יראה את תמונת הפרופיל החדשה שלו. מתחת לתמונת פרופיל ניתן לראות את שם המשתמש והמד כושר הנוכחי שלו (Elo), מד הכושר מייצג את חוזקו של השחקן. מתחת לנתונים על המשתמש נמצאים כל המשחקים של המשתמש עם תמונה של היריב, שמו, מד הכושר שלו במשחק ותוצאת המשחק X – הפסד, V – ניצחון, - – תיקו.



Emil

Elo: 1190

Recent Games

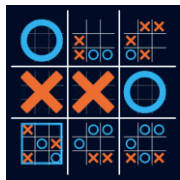
	Uri(1200)	X
---	-----------	---

תמונת הפרופיל
של המשתמש
האנונימי

שם המשתמש

מד הכושר
הנוכחי

היסטוריית
המשחקים של
המשתמש



מסך LeaderboardActivity

בפתיחת המסך, מיד תתחיל טעינה של מספר שחקנים הנמצאים בצמרת הטבלה, משמאל נמצא הדירוג של כל משתמש יחסית לאחרים, באמצע תמונת הפרופיל יחד עם שם המשתמש, ומימין מד הכושר שלו. הרשומה של המשתמש המחובר מודגשת מעט בשביל זיהוי קל.

Leaderboard		
Rank	Player	Elo Rating
#1	 כצכ	9999
#2	 6	4444
#3	 Uri	1340
#4	 Emil.za07	1292
#5	 Mom	1201
#6	 123412	1200
#7	 Emil	1200
#8	 #@^%#!^#&	1200
#9	 rqwga	1200
#10	 Toanls1	1200
#11	 g	1200

שם המשתמש

תמונת הפרופיל ושם המשתמש

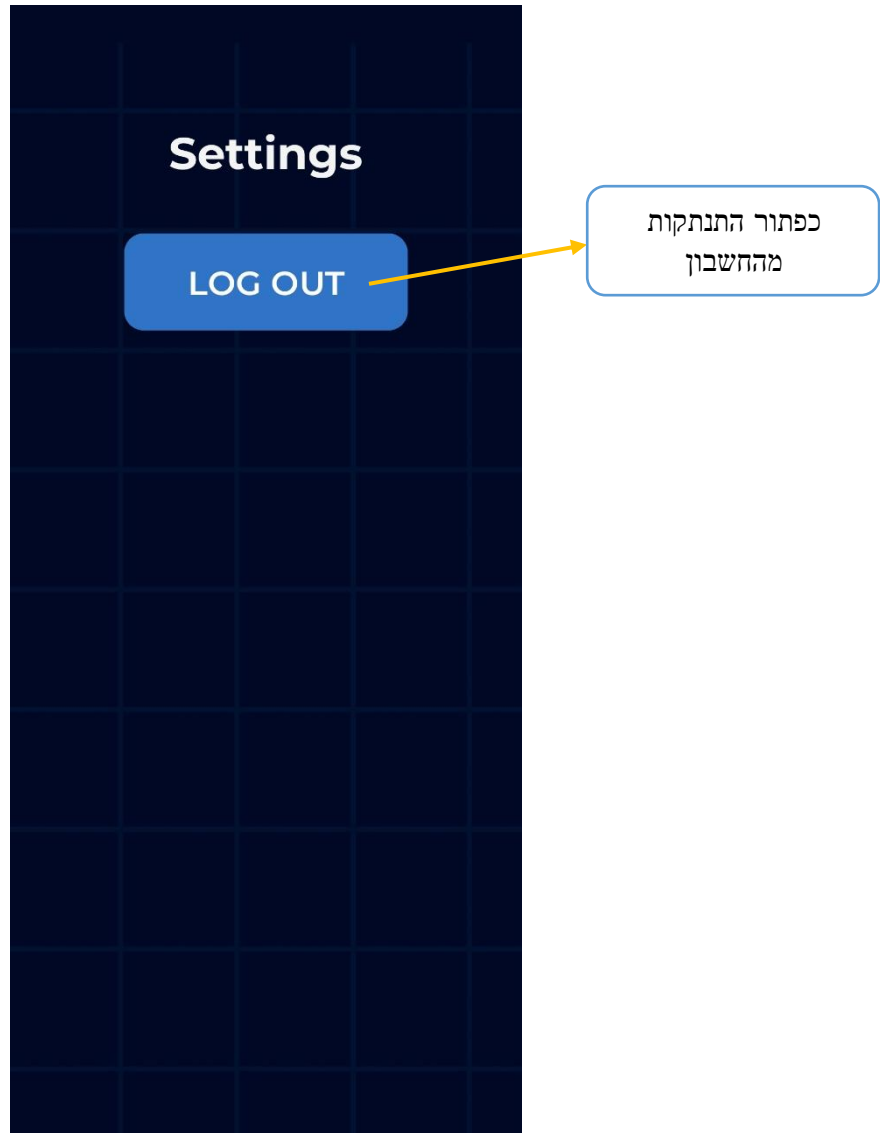
הדירוג של המשתמש

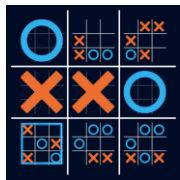
המשתמש המחובר



מסך SettingsActivity

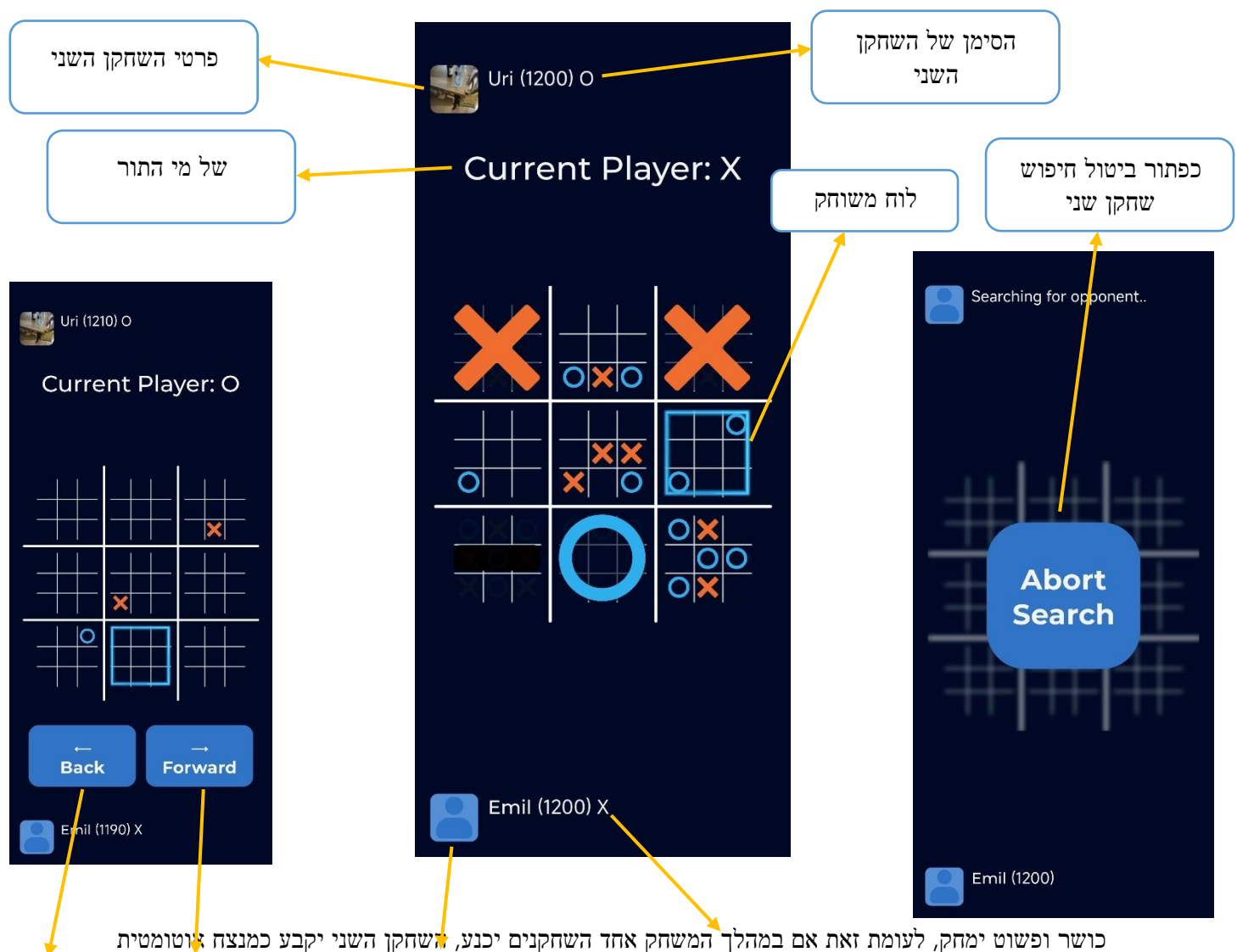
במסך הזה קיים כפתור אחד המאפשר למשתמש להתנתק מחשבונו ולהרשם מחדש או להכנס עם חשבון אחר.



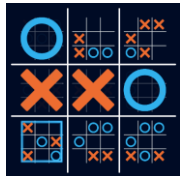


מסך GameActivity

זהו המסך העיקרי באפליקציה כי בו מתנהל כל המשחק. בטעינה למשחק לוקלי או נגד המחשב, מיד מוצג המשחק עם כל הנתונים ואפשר להתחיל לשחק. לעומת זאת בהתחלה של משחק אונליין, המשתמש בהתחלה יראה מסך המתנה לשחקן שני, בזמן ההמתנה השחקן יכול ללחוץ על הכפתור הגדול ולהפסיק את חיפוש המשחק, כאשר נמצא שחקן שני, כל הנתונים יטענו ויהיה אפשר להתחיל לשחק. בשביל שכל שחקן ידע איזה סימן הוא (O/X) ליד השמות שלהם רשום סימנם, בנוסף על מנת שהשחקנים ידעו בקלות של מי התור, יש טקסט גדול במרכז המיידיע את שני השחקנים לגבי התור, יתרה מזאת, על מנת שהשחקנים לא ישכחו באיזה לוח עליהם לשחק, הלוח המתאים מסומן במסגרת. אם בתחילת המשחק (לפני ששני השחקנים עשו את תורם) אחד מהשחקנים רוצה לפרוש, המשחק לא יחשב למד



כושר ופשוט ימחק, לעומת זאת אם במהלך המשחק אחד השחקנים יכנע, השחקן השני יקבע כמנצח אוטומטית ויקבל את מלוא הנקודות. במצב צפיה חוזרת, מופיעים על המסך שני כפתורים חדשים בשביל מעבר בין המהלכים.



מהלך
עבורך

מהלך
היגויך

פרטי השחקן הראשון
(המשתמש המהיר)

הסימן של השחקן
הראשון



רפלקציה / וראות סיכום אישי

תיאור תהליך העבודה על הפרויקט

תהליך העבודה על אפליקציית "סופר איקס עיגול" עבר דרך מספר שלבים עיקריים, החל מהרעיון הראשוני ועד למוצר מוגמר ומתפקד. בתחילה, זיהיתי את הצורך באפליקציה שתתאים לבני נוער, עם ממשק פשוט ונגיש. הגדרתי את המטרות המרכזיות של האפליקציה, כגון משחק מול המחשב, משחק לוקאלי, משחק מקוון, פרופיל אישי, היסטוריית משחקים, מד כושר וטבלת דירוג.

בהמשך, תכננתי את מבנה האפליקציה וזרימת המשתמשים, תוך התחשבות בחוויית המשתמש (UX). יצרתי wireframes ותרשימי זרימה כדי לאפיין את המסכים והפונקציות השונות. בשלב הפיתוח, השתמשתי ב-Android Studio ובשפה Java. הטמעתי את הפיצ'רים השונים, כולל שימוש ב-SharedPreferences לאחסון נתוני משתמשים ובספריות כמו Gson לעיבוד נתונים.

לאחר הפיתוח, ביצעתי בדיקות רבות כדי לוודא את תקינות הפונקציות השונות.

העבודה העצמאית על הפרויקט אפשרה לי להעמיק את הידע שלי בפיתוח אפליקציות אנדרואיד, להתמודד עם אתגרים טכניים ולפתח פתרונות יצירתיים, תוך כדי שמירה על חוויית משתמש איכותית ומעורבות גבוהה.

תהליך הלמידה

תהליך הלמידה שלי בפיתוח אפליקציית "סופר איקס עיגול" היה מסע מעשיר ומאתגר של למידה עצמית, שהתבסס על סקרנות, התמדה וגישה פרקטית. נעזרתי במדריכים רשמיים של Android Developers, סיפקו לי תשתית מוצקה להבנת עקרונות הפיתוח.

במהלך הפיתוח, התמודדתי עם אתגרים שונים, כמו יצירת ממשק משתמש אינטואיטיבי, ניהול נתונים באמצעות Firebase Firestore Shared Preferences והטמעת פיצ'רים מתקדמים כגון משחק מקוון, היסטוריית משחקים ומד כושר. התהליך כלל ניסוי וטעייה, חיפוש פתרונות באינטרנט, והתעמקות בתיעוד ובקהילות מפתחים. למדתי כיצד לייעל ביצועים, לשפר את חוויית המשתמש, ולהבטיח את יציבות האפליקציה. העצמאית אפשרה לי לפתח מיומנויות טכניות, חשיבה יצירתית ויכולת פתרון בעיות, תוך כדי בניית מוצר שלם ומקצועי.

אילו כלים נלקחים להמשך

במהלך העבודה על "סופר איקס עיגול" הסתמכתי על Android Studio, סביבת הפיתוח הרשמית של Google שמציעה כלים ואמולטורים מובנים להרצה ולדיבוג מהירים של אפליקציות אנדרואיד. לכלל הקוד



השתמשתי ב Java-בלבד, שפה ותיקה ויציבה בפיתוח אנדרואיד עם תמיכה רחבה בספריות ובקהילה. להגדרת ושימור העדפות משתמשים קטנים בתצורה מקומית, נהגתי ב SharedPreferences-מנגנון פשוט ופרטי לאחסון זוגות מפתח-ערך שמנוהל על ידי המערכת עצמה. לצורך אחסון וטעינת נתונים מורכבים יותר בחרתי ב-Cloud Firestore מסד נתונים בענן של Firebase שמציע יכולות יעילות בשאלות.

את הצד השרתי של לוגיקת האפליקציה המתקדמת – התחלה של משחק והודעה על כך לשני השחקנים וסיום המשחק וחיוש המד כושר – ארגנתי באמצעות Cloud Functions for Firebase שירות "ללא שרת" שמריץ קוד python בתגובה לאירועים בענן בצורה מאובטחת ומנוהלת. לעיבוד נתונים במבנה JSON השתמשתי בספריית Gson של Google שממפה אובייקטים ל JSON-ולהיפך בקלות, מטפלת בשדות חסרים ושומרת על תאימות לאחר. ניהול הגרסאות והקוד שביצעתי דרך Git ו GitHub-כלי בקרה מבוצרת המאפשר מעקב אחרי שינויים, שיתוף פעולה וסנכרון בין סניפים בקליק.

תובנות מהתהליך

התהליך לימד אותי כמה נקודות מפתח שכל פיתוח אפליקציה חייב לשים עליהן דגש: ראשית, חשיבות תכנון מוקדם ודיוק באפיון – ברגע שהקדשתי זמן לשרטוט זרימות משתמש ו-wireframes ברורים, הפיתוח התבצע ביתר מהירות ובהירות. שנית, התמודדות עם עבודה אסינכרונית – השילוב של Firestore יחד עם Cloud Functions אילץ אותי לחדד את הבנת המנגנונים של קריאות רשת וחזרה על ניסיונות, ולוודא שתמיד יש טיפול במצבים של חוסר חיבור או שגיאות.

פיתוח כמפתח יחיד חיזק את כישורי פתרון הבעיות וההתמדה הטכנית שלי – כל אתגר גרם לי לחקור לעומק, למצוא דוגמאות קוד, ולבנות פתרונות גמישים ותחזוקתיים. לבסוף.

בראייה לאחר

בסתכלות לאחור, אני שמ לב כמה כל שלב ועיקוף בתכנון ובפיתוח הביאו אותי לתובנות חשובות. כשאפיינתי את הזרימה והמסכים מראש, חסכתי בזמנים של תיקונים חוזרים ושינויים גורפים בשלב הפיתוח—זה לימד אותי שכמה ששרטוטים יביאו הרבה סדר ואיפשרו לי להתקדם בביטחון. העבודה עם Firestore ו Cloud-Functions חשפה אותי לקשיים של סנכרון נתונים בזמן אמת ולטיפול באיציבות של רשת.

עוד הבנתי שהעלאות תדירות של גרסאות—אפילו קטנות—סיפקו לי משובים ששיפרו את המוצר בבטחה. ההתמודדות עם אתגרים טכניים כעצמאי חיזקה בי את היכולת לחקור, להתגבר ולמצוא פתרונות יצירתיים בלי פחד משינויים באמצע הדרך. בסוף, הפשטות בממשק והעיצוב המינימליסטי הסתברו כקריטיים: כשלא מסבכים.



האם וכיצד ניתן לשפר את האפליקציה

באפליקציה יש מקום רב לשיפור, קיימים פיצ'רים רבים שרציתי להוסיף אך לא הספקתי עקב מגבלת הזמן והעבודות הנוספות שהיו לי באותו זמן, צריך להוסיף הרבה מאוד פעילות למשתמש, שינוי שם משתמש, סיסמה. להוסיף מערכת של חברים כדי לשחק בקלות נגד אדם ספציפי ברשת. בנוסף האלגוריתם של המחשב מאוד רחוק מלהיות מושלם (מה שהיה בתכנון). בנוסף חסרה גם אפשרות לשחק ללא יצירת משתמש (או באופליין)

שאלת חקר עצמי

שאלת החקר העצמית שלי בפרויקט הייתה איך אפשר לבנות אפליקציית "סופר איקס עיגול" שמרגישה ברמה של אתרים מתקדמים כמו Chess.com, אבל עדיין נשארת פשוטה, מהירה ונגישה – גם כשיש בה פיצ'רים מתקדמים כמו משחק ברשת, צפייה בהיסטוריה, וטבלת דירוג. רציתי להבין איפה עובר הגבול בין אפליקציה בסיסית למשחק פשוט, לבין חוויית משתמש עשירה ומעניינת – בלי להכביד או לסבך את השימוש.

במהלך העבודה שאלתי את עצמי איך לשלב את כל זה בצורה שתעבוד טוב – גם מבחינה טכנית וגם מבחינת ממשק. למשל, איך לעדכן את הדירוג בלי עיכובים, איך לאפשר לשחקן לצפות במשחקים ישנים בלי להעמיס על הממשק, ואיך לגרום למשחק להרגיש חי גם כשמשחקים נגד שחקן אקראי ברשת. השאלות האלה הובילו אותי להרבה פתרונות וגם לא מעט ניסוי וטעייה, אבל עזרו לי להישאר ממוקד ולבנות משהו שבאמת עובד טוב.

ביבליוגרפיה

ZeSardine. (2021, June 5). *I made an (unstoppable) ULTIMATE Tic-Tac-Toe AI*. YouTube.

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

Firebase. (2019). *Documentation* / Firebase. Firebase; Google.

<https://firebase.google.com/docs>

OpenAI. (2025). *ChatGPT*. ChatGPT; OpenAI.

<https://chatgpt.com/>



Stack Overflow. (2024). *Stack Overflow - Where Developers Learn, Share, & Build Careers*. Stack Overflow.

<https://stackoverflow.com/>

Reddit. (2005). *Reddit*. Reddit.

<https://www.reddit.com/>

Emilza07. (2025). *GitHub - Emilza07/Ultimate_tic_tac_toe: Ultimate Tic-Tac-Toe with local online multiplayer*

and a strategic bot. My final high school Android project. GitHub.

https://github.com/Emilza07/Ultimate_tic_tac_toe

Android Studio. (2019). *Android Studio and SDK tools*. Android Developers.

<https://developer.android.com/studio>

נספחים



AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

    <uses-feature
        android:name="android.hardware.camera"
        android:required="false" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE_DATA_SYNC" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
        android:maxSdkVersion="32" />

    <queries>
        <intent>
            <action android:name="android.intent.action.GET_CONTENT" />
            <data android:mimeType="image/*" />
        </intent>
        <intent>
            <action android:name="android.intent.action.PICK" />
            <data android:mimeType="image/*" />
        </intent>
    </queries>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.UTTT">
        <activity
            android:name=".ACTIVITIES.SettingsActivity"
            android:exported="false" />
        <activity
            android:name=".ACTIVITIES.LeaderboardActivity"
            android:exported="false" />
        <activity
            android:name=".ACTIVITIES.ProfileActivity"
            android:exported="false" />
        <activity
            android:name="com.yalantis.ucrop.UCropActivity"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar" />
        <activity
            android:name=".ACTIVITIES.GameActivity"
```



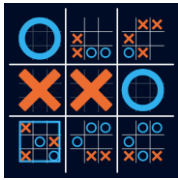
```
        android:exported="false" />
    <activity
        android:name=". ACTIVITIES. MainActivity"
        android:exported="false" />
    <activity
        android:name=". ACTIVITIES. LoginActivity"
        android:exported="false" />
    <activity
        android:name=". ACTIVITIES. RegisterActivity"
        android:exported="false" />
    <activity
        android:name=". ACTIVITIES. BASE. BaseActivity"
        android:exported="false" />
    <activity
        android:name=". ACTIVITIES. AuthActivity"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    <service
        android:name=". SERVICES. AppMonitorService"
        android:enabled="true"
        android:exported="false"
        android:foregroundServiceType="dataSync" />

</application>

</manifest>
```



AuthActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.emil_z.helper.UserSessionPreference;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.UsersViewModel;

/**
 * Activity that handles user authentication on application startup.
 * <p>
 * Provides login and registration options, and checks for existing sessions.
 */
public class AuthActivity extends BaseActivity {

    private Button btnLogin;
    private Button btnRegister;

    private UsersViewModel viewModel;

    private UserSessionPreference sessionPreference;

    /**
     * Initializes the authentication activity, sets up the UI,
     * and checks for existing login sessions.
     *
     * @param savedInstanceState The previously saved instance state, if any.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        AppCompatActivity.setDefaultNightMode(AppCompatActivity.MODE_NIGHT_NO);
        EdgeToEdge.enable(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auth);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```



```
});  
setBottomNavigationVisibility(false);  
  
sessionPreference = new UserSessionPreference(this);  
  
initializeViews();  
setListeners();  
setViewModel();  
checkForLogin();  
}  
  
/**  
 * Initializes view components and shows the login progress dialog.  
 */  
@Override  
protected void initializeViews() {  
    btnLogin = findViewById(R.id.btnLogin);  
    btnRegister = findViewById(R.id.btnRegister);  
  
    showProgressDialog(getString(R.string.logging_in), "");  
}  
  
/**  
 * Sets up click listeners for login and registration buttons.  
 */  
@Override  
protected void setListeners() {  
    btnLogin.setOnClickListener(v -> startActivity(new Intent(AuthActivity.this,  
LoginActivity.class)));  
    btnRegister.setOnClickListener(v -> startActivity(new Intent(AuthActivity.this,  
RegisterActivity.class)));  
}  
  
/**  
 * Initializes the ViewModel and sets up observers for authentication data.  
 */  
@Override  
protected void setViewModel() {  
    viewModel = new ViewModelProvider(this).get(UsersViewModel.class);  
  
    viewModel.getLiveDataEntity().observe(this, user -> {  
        if (user != null) {  
            String storedEmail = sessionPreference.getEmail();  
            String storedPassword = sessionPreference.getPassword();  
            if (user.getUsername().equals(storedEmail) &&  
                user.getHashedPassword().equals(storedPassword)) {  
                currentUser = user;  
                Intent intent = new Intent(AuthActivity.this, MainActivity.class);  
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);  
                startActivity(intent);  
            }  
        }  
    });  
}
```



```
        }
    }
    hideProgressDialog();
});
}

/**
 * Checks if user has a valid session and attempts to log in automatically.
 */
private void checkForLogin() {
    if (sessionPreference.isValidSession()) {
        String userIdFs = sessionPreference.getUserIdFs();
        viewModel.get(userIdFs);
    } else {
        hideProgressDialog();
    }
}
}
```



GameActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.Point;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.activity.OnBackPressedCallback;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;
import androidx.core.content.res.ResourcesCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.gridlayout.widget.GridLayout;
import androidx.lifecycle.ViewModelProvider;

import com.emil_z.helper.AlertUtil;
import com.emil_z.model.BoardLocation;
import com.emil_z.model.Game;
import com.emil_z.model.GameType;
import com.emil_z.model.OuterBoard;
import com.emil_z.model.Player;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.ultimate_tic_tac_toe.SERVICES.AppMonitorService;
import com.emil_z.viewmodel.GamesViewModel;
import com.emil_z.viewmodel.GamesViewModelFactory;
import com.emil_z.viewmodel.UsersViewModel;

import java.util.Objects;

/**
 * Activity for managing and displaying a single Ultimate Tic Tac Toe game.
 * Handles game initialization, board creation, move handling, player display, and game state
 * updates.
 * Supports different game types: CPU, LOCAL, ONLINE, and REPLAY.
 */
```



```
*/
public class GameActivity extends BaseActivity {

    private GridLayout gridBoard;
    private LinearLayout lIP2;
    private ImageView ivP2Pfp;
    private TextView tvP2Name;
    private TextView tvP2Elo;
    private TextView tvP2Sign;
    private LinearLayout lIP1;
    private ImageView ivP1Pfp;
    private TextView tvP1Name;
    private TextView tvP1Elo;
    private TextView tvP1Sign;
    private TextView tvCurrentPlayer;
    private Button btnAbort;
    private LinearLayout lIReview;
    private Button btnForward;
    private Button btnBackward;

    private GamesViewModel gamesViewModel;
    private UsersViewModel usersViewModel;
    private boolean monitorServiceStarted = false;

    private String[] errorCodes;

    private int moveIndex = 0;
    private char[][] outerBoardWinners;

    private GameType gameType;
    private int boardSize;
    private float conversionFactor;

    private Intent intent;

    /**
     * Initializes the activity, sets up UI, listeners, ViewModels, and game state.
     */
    * @param savedInstanceState The previously saved instance state, if any.
    */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        EdgeToEdge.enable(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_game);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```



```
        initializeViews();
        setListeners();
        setViewModel();
        gameInit(gameType);
    }

    /**
     * Initializes all view components for the game screen.
     */
    @Override
    protected void initializeViews() {

        IIP2 = findViewById(R.id.IIP2);
        ivP2Pfp = findViewById(R.id.ivP2Pfp);
        tvP2Name = findViewById(R.id.tvP2Name);
        tvP2Elo = findViewById(R.id.tvP2Elo);
        tvP2Sign = findViewById(R.id.tvP2Sign);
        IIP1 = findViewById(R.id.IIP1);
        ivP1Pfp = findViewById(R.id.ivP1Pfp);
        tvP1Name = findViewById(R.id.tvP1Name);
        tvP1Elo = findViewById(R.id.tvP1Elo);
        tvP1Sign = findViewById(R.id.tvP1Sign);
        tvCurrentPlayer = findViewById(R.id.tvCurrentPlayer);
        btnAbort = findViewById(R.id.btnAbort);
        IIRReview = findViewById(R.id.IIRReview);
        btnForward = findViewById(R.id.btnForward);
        btnBackward = findViewById(R.id.btnBackward);

        intent = getIntent();
        gameType = (GameType) intent.getSerializableExtra(MainActivity.EXTRA_GAME_TYPE);
        intent.putExtra(MainActivity.EXTRA_GAME_TYPE, gameType);
        errorCodes = getResources().getStringArray(R.array.error_codes);
        createBoard();
        outerBoardWinners = new char[3][3];
    }

    /**
     * Sets up click listeners for game controls, including abort, move review, and back navigation.
     */
    @SuppressWarnings("ConstantConditions")
    @Override
    protected void setListeners() {
        btnAbort.setOnClickListener(v -> {
            if (gameType.equals(GameType.ONLINE)) {
                gamesViewModel.exitGame();
            }
            setResult(RESULT_CANCELED, intent);
            finish();
        });
    }
}
```




```
});

btnForward.setOnClickListener(v -> {
    if (btnBackward.getBackgroundTintList().getDefaultColor() == ContextCompat.getColor(this,
R.color.hintTextColor))
        btnBackward.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this,
R.color.buttonColor)));
    Game game = gamesViewModel.getLiveDataGame().getValue();

    if (moveIndex >= game.getMoves().size()) {
        Toast.makeText(GameActivity.this, R.string.last_move, Toast.LENGTH_SHORT).show();
        return;
    }

    for (int i = 0; i < gridBoard.getChildCount(); i++) {
        gridBoard.getChildAt(i).setBackgroundColor(Color.TRANSPARENT);
    }

    BoardLocation lastMove = game.getMoves().get(moveIndex);
    int innerGridIndex = lastMove.getOuter().x * 3 + lastMove.getOuter().y;
    int btnIndex = lastMove.getInner().x * 3 + lastMove.getInner().y;
    GridLayout innerGrid = (GridLayout) gridBoard.getChildAt(innerGridIndex);
    ImageView btn = (ImageView) innerGrid.getChildAt(btnIndex);
    btn.setImageResource(moveIndex % 2 == 1 ? R.drawable.o : R.drawable.x);

    rebuildGameStateToMove(moveIndex + 1);

    if (moveIndex < game.getMoves().size() - 1) {
        boolean isNextGridPlayable = outerBoardWinners[btnIndex / 3][btnIndex % 3] == 0;
        if (isNextGridPlayable) {
            GridLayout nextInnerGrid = (GridLayout) gridBoard.getChildAt(btnIndex);
            nextInnerGrid.setBackgroundResource(R.drawable.border);
        }
    }

    tvCurrentPlayer.setText(moveIndex % 2 == 1 ? R.string.player_x_turn :
R.string.player_o_turn);
    moveIndex++;
    if (moveIndex >= game.getMoves().size()) {
        btnForward.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this,
R.color.hintTextColor)));
    }
});

btnBackward.setOnClickListener(v -> {
    if (btnForward.getBackgroundTintList() ==
ColorStateList.valueOf(ContextCompat.getColor(this, R.color.hintTextColor))
        btnForward.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this,
R.color.buttonColor)));
    moveIndex--;
```



```
if (moveIndex < 0) {
    Toast.makeText(GameActivity.this, R.string.last_move, Toast.LENGTH_SHORT).show();
    moveIndex = 0;
    return;
}

Game game = gamesViewModel.getLiveDataGame().getValue();
resetBoardDisplay();
rebuildGameStateToMove(moveIndex);

if (moveIndex > 0) {
    BoardLocation previousMove = game.getMoves().get(moveIndex - 1);
    int prevInnerPos = previousMove.getInner().x * 3 + previousMove.getInner().y;

    boolean isNextGridPlayable = outerBoardWinners[prevInnerPos / 3][prevInnerPos % 3] ==
0;

    if (isNextGridPlayable) {
        GridLayout nextInnerGrid = (GridLayout) gridBoard.getChildAt(prevInnerPos);
        nextInnerGrid.setBackgroundResource(R.drawable.border);
    }
}

tvCurrentPlayer.setText(moveIndex % 2 == 0 ? R.string.player_x_turn :
R.string.player_o_turn);
if (moveIndex == 0) {
    btnBackward.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this,
R.color.hintTextColor)));
}
});

getOnBackPressedDispatcher().addCallback(this, new OnBackPressedCallback(true) {
    @Override
    public void handleOnBackPressed() {
        if (gameType == GameType.REPLAY) {
            finish();
            return;
        }
        Game game = gamesViewModel.getLiveDataGame().getValue();
        AlertUtil.alert(
            GameActivity.this,
            (game != null && game.getMoves().size() > 1) ? "Resign" : "Abort",
            "Are you sure you want to exit the game?",
            true,
            0,
            "Yes",
            "No",
            null,
            () -> {
                if (gameType == GameType.ONLINE)
                    gamesViewModel.exitGame();
            }
        );
    }
});
```



```

        setResult((game != null && game.getMoves().isEmpty()) ? RESULT_OK :
RESULT_CANCELED, intent);
        if (gameType != GameType.ONLINE || game == null)
            finish();
    }},
    null,
    null
);
}
});

}

/**
 * Handles a board button click, parses the tag, and makes a move via the ViewModel.
 *
 * @param btn The ImageView button that was clicked.
 */
private void handleBoardButtonClick(ImageView btn) {
    String tag = (String) btn.getTag();
    // Handle button click using the tag (e.g., "btn0101" for 1st outerRow, 2st outerColumn and
1th innerRow, 2st innerColumn)
    gamesViewModel.makeMove(new BoardLocation(tag.charAt(3) - '0', tag.charAt(4) - '0',
tag.charAt(5) - '0', tag.charAt(6) - '0'));
}

/**
 * Sets up ViewModels, observes LiveData for game and user state, and updates the UI
accordingly.
 */
@SuppressWarnings("ConstantConditions")
@Override
protected void setViewModel() {
    gamesViewModel = new ViewModelProvider(this, new GamesViewModelFactory(getApplication(),
gameType)).get(GamesViewModel.class);
    usersViewModel = new ViewModelProvider(this).get(UsersViewModel.class);

    gamesViewModel.getLiveDataErrorCode().observe(this, code -> {
        if (code != 0) {
            Toast.makeText(GameActivity.this, errorCodes[code], Toast.LENGTH_SHORT).show();
            gamesViewModel.resetLiveDataErrorCode();
            if (code == 4) {
                setResult(RESULT_CANCELED, intent);
                finish();
            }
        }
    });

    gamesViewModel.getLiveDataGame().observe(this, game -> {
        if (game == null) {

```



```
        setResult(RESULT_OK, intent);
        finish();
    } else if (!game.getMoves().isEmpty() && !game.isFinished()) {
        BoardLocation lastMove = game.getMoves().get(game.getMoves().size() - 1);
        if (game.getMoves().size() > 1) {
            int previousMoveIndex = lastMove.getOuter().x * 3 + lastMove.getOuter().y;
            GridLayout prevInnerGrid = (GridLayout) gridBoard.getChildAt(previousMoveIndex);
            prevInnerGrid.setBackgroundColor(Color.TRANSPARENT);
        }
        int innerGridIndex = lastMove.getOuter().x * 3 + lastMove.getOuter().y;
        int btnIndex = lastMove.getInner().x * 3 + lastMove.getInner().y;
        GridLayout innerGrid = (GridLayout) gridBoard.getChildAt(innerGridIndex);
        ImageView btn = (ImageView) innerGrid.getChildAt(btnIndex);

        char currentPlayer = game.getOuterBoard().getCurrentPlayer();
        btn.setImageResource(currentPlayer == 'O' ? R.drawable.o : R.drawable.x);
        if (!game.getOuterBoard().isFreeMove()) {
            GridLayout nextMoveGrid = (GridLayout) gridBoard.getChildAt(btnIndex);
            nextMoveGrid.setBackgroundResource(R.drawable.border);
        }
        tvCurrentPlayer.setText(currentPlayer == 'X' ? R.string.player_x_turn :
R.string.player_o_turn);
    }
});

gamesViewModel.getLiveDataOuterBoardWinners().observe(this, boardWinners -> {
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            char winner = boardWinners[row][col];
            if (winner != 0) {
                GridLayout innerGrid = (GridLayout) gridBoard.getChildAt(row * 3 + col);
                if (winner == 'X') {
                    innerGrid.setBackground(ResourcesCompat.getDrawable(getResources(),
R.drawable.x_blurred, null));
                } else if (winner == 'O') {
                    innerGrid.setBackground(ResourcesCompat.getDrawable(getResources(),
R.drawable.o_blurred, null));
                } else {
                    innerGrid.setBackground(ResourcesCompat.getDrawable(getResources(),
R.drawable.tie, null));
                }

                for (int i = 0; i < innerGrid.getChildCount(); i++) {
                    innerGrid.getChildAt(i).setAlpha(0.05f);
                }
            }
        }
    }
});
```



```
gamesViewModel.getLiveDataIsFinished().observe(this, aBoolean -> {
    Game game = gamesViewModel.getLiveDataGame().getValue();
    String winner;

    switch (gameType) {
        case CPU:
        case LOCAL:
            winner = game.getWinnerIdFs();
            break;
        case ONLINE:
            if (Objects.equals(game.getWinnerIdFs(), "T"))
                winner = "T";
            else
                winner = Objects.equals(game.getWinnerIdFs(), game.getCrossPlayerIdFs()) ? "X" :
"O";

            break;
        default:
            throw new IllegalStateException("Unexpected value: " + gameType);
    }
    AlertUtil.alert(GameActivity.this,
        getString(R.string.match_complete),
        Objects.equals(winner, "T") ?
            getString(R.string.game_outcome_tie) :
            getString(R.string.game_outcome_win, winner),
        false,
        0,
        getString(R.string.return_to_menu),
        null,
        null,
        () -> {
            intent.putExtra(MainActivity.EXTRA_GAME_TYPE, gameType);
            setResult(RESULT_OK, intent);
            finish();
        }),
        null,
        null);
});

gamesViewModel.getLiveDataIsStarted().observe(this, aBoolean -> {
    if (aBoolean) {
        Game game = gamesViewModel.getLiveDataGame().getValue();
        boolean isHost = Objects.equals(game.getPlayer1().getIdFs(), currentUser.getIdFs());

        if (gameType == GameType.ONLINE || gameType == GameType.REPLAY)
            usersViewModel.get(isHost ? game.getPlayer2().getIdFs() :
game.getPlayer1().getIdFs());
        else {
            setPlayers(game.getPlayer1(), game.getPlayer2());
            btnAbort.setVisibility(View.GONE);
            tvCurrentPlayer.setVisibility(View.VISIBLE);
        }
    }
});
```



```
        gridBoard.setBackground(ResourcesCompat.getDrawable(getResources(),
R.drawable.board, null));
    }

    if (gameType == GameType.ONLINE) {
        String gameIdFs = game.getIdFs();
        String player1IdFs = game.getPlayer1().getIdFs();
        String player2IdFs = game.getPlayer2().getIdFs();

        if (!monitorServiceStarted) {
            monitorServiceStarted = true;
            AppMonitorService.startService(this, true, gameIdFs, player1IdFs, player2IdFs,
isHost);
        } else {
            AppMonitorService.updateGameState(true, gameIdFs, player1IdFs, player2IdFs,
isHost);
        }
    }
}

});

gamesViewModel.getLiveDataGameIdFs().observe(this, gameIdFs -> {
    if (gameIdFs != null && !monitorServiceStarted) {
        monitorServiceStarted = true;
        AppMonitorService.startService(this, true, gameIdFs, null, null, false);
    }
});

gamesViewModel.getLiveDataEntity().observe(this, game ->
gamesViewModel.startReplayGame(game));

usersViewModel.getLiveDataEntity().observe(this, user -> {
    btnAbort.setVisibility(View.GONE);
    tvCurrentPlayer.setVisibility(View.VISIBLE);
    gridBoard.setBackground(ResourcesCompat.getDrawable(getResources(), R.drawable.board,
null));
    gridBoard.setVisibility(View.VISIBLE);

    Game game = gamesViewModel.getLiveDataGame().getValue();
    boolean isHost = Objects.equals(game.getPlayer1().getIdFs(), currentUser.getIdFs());
    if (user != null) {
        Player p1 = isHost ? new Player(currentUser) : new Player(user);
        Player p2 = isHost ? new Player(user) : new Player(currentUser);
        setPlayers(p1, p2);
    } else
        setPlayers(game.getPlayer1(), game.getPlayer2());

    if (gameType == GameType.REPLAY)
        llReview.setVisibility(View.VISIBLE);
});
```



```
}

/**
 * Cleans up resources and notifies the monitor service when the activity is destroyed.
 */
@Override
protected void onDestroy() {
    super.onDestroy();
    AppMonitorService.userClosedActivity(this);
}

/**
 * Initializes the game based on the selected game type.
 *
 * @param gameType The type of game (CPU, LOCAL, ONLINE, REPLAY).
 */
private void gameInit(GameType gameType) {
    switch (gameType) {
        case CPU:
            char sign = intent.getCharExtra(MainActivity.EXTRA_SIGN, 'X');
            gamesViewModel.startCpuGame(sign == 'X' ? currentUser.getIdFs() : "CPU");
            break;
        case LOCAL:
            gamesViewModel.startLocalGame();
            break;
        case ONLINE:
            try {
                btnAbort.setVisibility(View.VISIBLE);
                gamesViewModel.startOnlineGame(new Player(currentUser));
                setPlayers(new Player(currentUser), null);
            } catch (Exception e) {
                Toast.makeText(this, R.string.game_connection_error, Toast.LENGTH_SHORT).show();
                setResult(RESULT_CANCELED, intent);
                finish();
            }
            break;
        case REPLAY:
            gridBoard.setVisibility(View.INVISIBLE);
            btnBackward.setBackgroundTintList(ColorStateList.valueOf(ContextCompat.getColor(this,
R.color.hintTextColor)));
            gamesViewModel.get(intent.getStringExtra(MainActivity.EXTRA_GAME_ID_FS));
            break;
    }
}

/**
 * Sets up the player views and information for the game screen based on the game type and
 * player roles.
 *
 * Handles display of player names, profile pictures, ELO ratings, and X/O signs for both
 * players.
 */
```



```
* Adjusts the UI for CPU, LOCAL, ONLINE, and REPLAY game types, including handling
host/opponent logic.
*
* @param p1 The first player (may be the current user or opponent depending on context).
* @param p2 The second player (may be the current user or opponent depending on context).
*/
@SuppressWarnings("ConstantConditions")
private void setPlayers(Player p1, Player p2) {
    IIP1.setVisibility(View.VISIBLE);
    IIP2.setVisibility(View.VISIBLE);
    Game game = gamesViewModel.getLiveDataGame().getValue();

    if (gameType == GameType.CPU || gameType == GameType.LOCAL) {
        ivP1Pfp.setImageResource(R.drawable.default_pfp);
        tvP1Name.setText(p1.getName());
        tvP1Elo.setText("");
        tvP1Sign.setText("X");

        ivP2Pfp.setImageResource(gameType == GameType.LOCAL ? R.drawable.default_pfp :
R.drawable.cpu_pfp);
        tvP2Name.setText(p2.getName());
        tvP2Elo.setText("");
        tvP2Sign.setText("O");
        if (!game.getCrossPlayerIds().equals(currentUser.getIdFs())) {
            tvCurrentPlayer.setText(R.string.player_o_turn);
            tvP1Sign.setText("O");
            tvP2Sign.setText("X");
        } else {
            tvCurrentPlayer.setText(R.string.player_x_turn);
        }
    } else if (gameType == GameType.ONLINE) {
        if (!gamesViewModel.getLiveDataIsStarted().getValue()) {
            ivP1Pfp.setImageBitmap(p1.getPictureBitmap());
            tvP1Name.setText(currentUser.getUsername());
            tvP1Elo.setText(getString(R.string.player_elo_format,
Math.round(currentUser.getElo())));
            tvP1Sign.setText("");

            ivP2Pfp.setImageBitmap(BitmapFactory.decodeResource(getResources(),
R.drawable.default_pfp));
            tvP2Name.setText(R.string.searching_for_opponent);
            tvP2Elo.setText("");
            tvP2Sign.setText("");
        } else {
            boolean isHost =
Objects.equals(gamesViewModel.getLiveDataGame().getValue().getPlayer1().getIdFs(),
currentUser.getIdFs());
            if (isHost) {
                ivP2Pfp.setImageBitmap(p2.getPictureBitmap());
                tvP2Name.setText(p2.getName());
```




```
tvP2Elo.setText(getString(R.string.player_elo_format, Math.round(p2.getElo())));

tvP1Sign.setText(Objects.equals(gameViewModel.getLiveDataGame().getValue().getCrossPlayerIds(),
p1.getIds()) ? "X" : "O");

tvP2Sign.setText(Objects.equals(gameViewModel.getLiveDataGame().getValue().getCrossPlayerIds(),
p2.getIds()) ? "X" : "O");
    } else {
        ivP2Pfp.setImageBitmap(p1.getPictureBitmap());
        tvP2Name.setText(p1.getName());
        tvP2Elo.setText(getString(R.string.player_elo_format, Math.round(p1.getElo())));

tvP1Sign.setText(Objects.equals(gameViewModel.getLiveDataGame().getValue().getCrossPlayerIds(),
p1.getIds()) ? "O" : "X");

tvP2Sign.setText(Objects.equals(gameViewModel.getLiveDataGame().getValue().getCrossPlayerIds(),
p2.getIds()) ? "O" : "X");
    }
} else {
    boolean isHost =
Objects.equals(gameViewModel.getLiveDataGame().getValue().getPlayer1().getId(),
currentUser.getId());
    Player firstPlayer = isHost ? p1 : p2;
    Player secondPlayer = isHost ? p2 : p1;

    ivP1Pfp.setImageBitmap(firstPlayer.getPictureBitmap());
    tvP1Name.setText(firstPlayer.getName());
    tvP1Elo.setText(getString(R.string.player_elo_format, Math.round(firstPlayer.getElo())));

    ivP2Pfp.setImageBitmap(secondPlayer.getPictureBitmap());
    tvP2Name.setText(secondPlayer.getName());
    tvP2Elo.setText(getString(R.string.player_elo_format, Math.round(secondPlayer.getElo())));

    boolean isFirstPlayerX = Objects.equals(game.getCrossPlayerIds(), firstPlayer.getId());
    tvP1Sign.setText(isFirstPlayerX ? "X" : "O");
    tvP2Sign.setText(isFirstPlayerX ? "O" : "X");
}
}

//region BoardInitialization

/**
 * Creates the main 3x3 outer board for Ultimate Tic Tac Toe.
 * Initializes the board size, conversion factor, and adds inner grids to the main board.
 */
private void createBoard() {
    final float BOARD_DRAWABLE_SIZE = 653;
    boardSize = getBoardSize();
    conversionFactor = boardSize / BOARD_DRAWABLE_SIZE;
```



```
gridBoard = findViewById(R.id.gridBoard);
setOuterGrid(gridBoard);

for (int row = 0; row < 3; row++) {
    for (int col = 0; col < 3; col++) {
        gridBoard.addView(createInnerGrid(row, col));
    }
}

/**
 * Creates a 3x3 inner grid (sub-board) at the specified outer board position.
 *
 * @param row The row index of the outer board.
 * @param col The column index of the outer board.
 * @return The initialized inner GridLayout.
 */
private GridLayout createInnerGrid(int row, int col) {
    GridLayout grid = new GridLayout(this);
    setInnerGrid(row, col, grid);

    for (int iRow = 0; iRow < 3; iRow++) {
        for (int iCol = 0; iCol < 3; iCol++) {
            grid.addView(createBoardButton(row, col, iRow, iCol));
        }
    }
    return grid;
}

/**
 * Creates a single cell button for the inner grid.
 * Sets up size, padding, margins, click listener, and a unique tag for identification.
 *
 * @param row Outer board row index.
 * @param col Outer board column index.
 * @param iRow Inner grid row index.
 * @param iCol Inner grid column index.
 * @return The initialized ImageView button.
 */
private ImageView createBoardButton(int row, int col, int iRow, int iCol) {
    final float INNER_CELL_DRAWABLE_SIZE = 55;
    final int INNER_CELL_DRAWABLE_REF_SIZE = (int) (INNER_CELL_DRAWABLE_SIZE * conversionFactor);
    final int MARGIN = (int) (1 * conversionFactor);
    final int PADDING = (int) (7 * conversionFactor);

    ImageView btn = new ImageView(this);
    btn.setId(View.generateViewId());
    btn.setScaleType(ImageView.ScaleType.FIT_XY);
    btn.setForegroundGravity(ImageView.TEXT_ALIGNMENT_CENTER);
```



```
btn.setClickable(true);

GridLayout.LayoutParams btnParams = new GridLayout.LayoutParams();
btnParams.width = INNER_CELL_DRAWABLE_REF_SIZE;
btnParams.height = INNER_CELL_DRAWABLE_REF_SIZE;
btn.setPadding(PADDING, PADDING, PADDING, PADDING);
btnParams.setMargins(MARGIN, MARGIN, MARGIN, MARGIN);
btnParams.rowSpec = GridLayout.spec(iRow);
btnParams.columnSpec = GridLayout.spec(iCol);
btn.setLayoutParams(btnParams);

btn.setOnClickListener(v -> handleBoardButtonClick((ImageView) v));
btn.setTag("btn" + row + col + iRow + iCol);
return btn;
}

/**
 * Configures the outer GridLayout for the main board, including padding, alignment, and size.
 *
 * @param gridLayout The GridLayout to configure.
 */
private void setOuterGrid(GridLayout gridLayout) {
    final int PADDING = (int) (21 * conversionFactor);

    gridLayout.setPadding(PADDING, PADDING, PADDING, PADDING);
    gridLayout.setAlignmentMode(GridLayout.ALIGN_BOUNDS);

    ViewGroup.LayoutParams params = gridLayout.getLayoutParams();
    params.width = boardSize;
    params.height = boardSize;
    gridLayout.setLayoutParams(params);
}

/**
 * Configures an inner GridLayout (sub-board) with size, padding, margins, and alignment.
 *
 * @param row Outer board row index.
 * @param col Outer board column index.
 * @param grid The inner GridLayout to configure.
 */
private void setInnerGrid(int row, int col, GridLayout grid) {
    final int INNER_BOARD_DRAWABLE_SIZE = 167;
    final int INNER_BOARD_DRAWABLE_REF_SIZE = (int) (INNER_BOARD_DRAWABLE_SIZE *
conversionFactor);
    final int MARGIN = (int) (18.5f * conversionFactor);
    final int PADDING = (int) (1 * conversionFactor);

    grid.setId(View.generateViewId());
    grid.setColumnCount(3);
    grid.setRowCount(3);
```



```
grid.setAlignmentMode(GridLayout.ALIGN_BOUNDS);
grid.setPadding(PADDING, PADDING, PADDING, PADDING);

GridLayout.LayoutParams params = new GridLayout.LayoutParams();
params.width = INNER_BOARD_DRAWABLE_REF_SIZE;
params.height = INNER_BOARD_DRAWABLE_REF_SIZE;
params.setMargins(MARGIN, MARGIN, MARGIN, MARGIN);
params.rowSpec = GridLayout.spec(row);
params.columnSpec = GridLayout.spec(col);
grid.setLayoutParams(params);
}

private int getBoardSize() {
    DisplayMetrics displayMetrics = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);
    int screenWidth = Math.min(displayMetrics.widthPixels, displayMetrics.heightPixels);
    return (int) (screenWidth * 0.9);
}

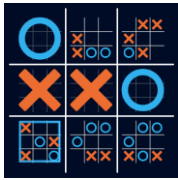
//endregion

/**
 * Resets the entire board display (clears moves and winners)
 */
private void resetBoardDisplay() {
    // Clear all inner board winners
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            outerBoardWinners[row][col] = 0;
            GridLayout innerGrid = (GridLayout) gridBoard.getChildAt(row * 3 + col);

            innerGrid.setBackground(null);
            innerGrid.setBackgroundColor(Color.TRANSPARENT);

            for (int i = 0; i < innerGrid.getChildCount(); i++) {
                ImageView btn = (ImageView) innerGrid.getChildAt(i);
                btn.setImageDrawable(null);
                btn.setAlpha(1.0f);
            }
        }
    }
}

/**
 * Rebuilds the game state up to the specified move index
 */
@SuppressWarnings("ConstantConditions")
private void rebuildGameStateToMove(int targetMoveIndex) {
    Game game = gamesViewModel.getLiveDataGame().getValue();
    OuterBoard tempBoard = new OuterBoard();
```



```
for (int moveIndex = 0; moveIndex < targetMoveIndex; moveIndex++) {
    BoardLocation move = game.getMoves().get(moveIndex);
    tempBoard.makeMove(move);

    int outerRow = move.getOuter().x, outerCol = move.getOuter().y;
    int innerRow = move.getInner().x, innerCol = move.getInner().y;
    int innerGridIndex = outerRow * 3 + outerCol;
    int btnIndex = innerRow * 3 + innerCol;

    GridLayout innerGrid = (GridLayout) gridBoard.getChildAt(innerGridIndex);
    ImageView btn = (ImageView) innerGrid.getChildAt(btnIndex);
    btn.setImageResource(moveIndex % 2 == 0 ? R.drawable.x : R.drawable.o);

    Point outerPoint = new Point(outerRow, outerCol);
    if (tempBoard.getBoard(outerPoint).isFinished()) {
        char winner = tempBoard.getBoard(outerPoint).getWinner();
        if (winner != 0) {
            outerBoardWinners[outerRow][outerCol] = winner;

            int drawableId = winner == 'X' ? R.drawable.x_blurred :
                winner == 'O' ? R.drawable.o_blurred :
                R.drawable.tie;
            innerGrid.setBackground(ResourcesCompat.getDrawable(getResources(), drawableId,
null));

            for (int i = 0; i < innerGrid.getChildCount(); i++) {
                innerGrid.getChildAt(i).setAlpha(0.05f);
            }
        }
    }
}
```



Leaderboard.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.graphics.Color;
import android.graphics.Typeface;
import android.os.Bundle;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.emil_z.model.Users;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.ADPTERS.UsersAdapter;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.UsersViewModel;

import java.util.ArrayList;

/**
 * Activity that displays the leaderboard of top players.
 * <p>
 * Shows a paginated list of users sorted by ELO, highlights the current user,
 * and loads more users as the user scrolls.
 */
public class LeaderboardActivity extends BaseActivity {
    private static final int PAGE_SIZE = 15;

    private RecyclerView rvLeaderboard;

    private UsersViewModel viewModel;
    private UsersAdapter adapter;

    private Users users;
    private boolean isLoading = false;
    private float lastLoadedElo = -1;
    private String lastLoadedIds = null;

    /**
     * Initializes the leaderboard activity, sets up UI, listeners, ViewModel, and adapter.
     */
}
```



```
* @param savedInstanceState The previously saved instance state, if any.
*/
@Override
protected void onCreate(Bundle savedInstanceState) {
    EdgeToEdge.enable(this);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_leaderboard);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    initializeViews();
    setListeners();
    setupRecyclerViewScrollListener();
    setViewModel();
    setAdapter();
    showLoadingMore();
}

/**
 * Initializes view components for the leaderboard screen.
 */
@SuppressWarnings("ConstantConditions")
@Override
public void initializeViews() {
    rvLeaderboard = findViewById(R.id.rvLeaderboard);

    DividerItemDecoration divider = new DividerItemDecoration(rvLeaderboard.getContext(),
        LinearLayoutManager.VERTICAL);
    divider.setDrawable(ContextCompat.getDrawable(this, R.drawable.dividor));
    rvLeaderboard.addItemDecoration(divider);
}

/**
 * Sets up click listeners for the leaderboard (none in this implementation).
 */
@Override
public void setListeners() {
}

/**
 * Sets up a scroll listener for the RecyclerView to load more users when reaching the end.
 */
@SuppressWarnings("ConstantConditions")
private void setupRecyclerViewScrollListener() {
    rvLeaderboard.addOnScrollListener(new RecyclerView.OnScrollListener() {
        @Override
        public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
```



```
super.onScrolled(recyclerView, dx, dy);

LinearLayoutManager layoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
int visibleItemCount = layoutManager.getChildCount();
int totalItemCount = layoutManager.getItemCount();
int firstVisibleItemPosition = layoutManager.findFirstVisibleItemPosition();

if (!isLoading && (visibleItemCount + firstVisibleItemPosition) >= totalItemCount
    && firstVisibleItemPosition >= 0
    && totalItemCount >= PAGE_SIZE) {
    loadUsers(true);
}
}
});
}

/**
 * Initializes the ViewModel, loads users, and observes user data changes.
 */
@Override
public void setViewModel() {
    viewModel = new ViewModelProvider(this).get(UsersViewModel.class);
    loadUsers(false);

    viewModel.getLiveDataCollection().observe(this, newUsers -> {
        if (adapter.getItems() != null) {
            int lastIndex = adapter.getItems().indexOf(null);
            if (lastIndex != -1) {
                adapter.getItems().remove(lastIndex);
                adapter.notifyItemRemoved(lastIndex);
            }
        }

        if (!newUsers.isEmpty()) {
            if (this.users == null)
                this.users = newUsers;
            else
                this.users.addAll(newUsers);
            isLoading = false;
            lastLoadedElo = newUsers.get(newUsers.size() - 1).getElo();
            lastLoadedIdFs = newUsers.get(newUsers.size() - 1).getIdFs();
            adapter.setItems(this.users);
        }
    });
}

/**
 * Sets up the adapter for the leaderboard RecyclerView and handles item display.
 */
```




```

private void setAdapter() {
    adapter = new UsersAdapter(new ArrayList<>(),
        R.layout.user_single_layout,
        holder -> {
            holder.putView("tvRank", holder.itemView.findViewById(R.id.tvRank));
            holder.putView("ivPfp", holder.itemView.findViewById(R.id.ivPfp));
            holder.putView("tvUsername", holder.itemView.findViewById(R.id.tvUsername));
            holder.putView("tvElo", holder.itemView.findViewById(R.id.tvElo));
        },
        ((holder, item, position) -> {
            ((TextView) holder.getView("tvRank")).setText(getString(R.string.rank_format, position
+ 1));

            ((ImageView) holder.getView("ivPfp")).setImageBitmap(item.getPictureBitmap());
            ((TextView) holder.getView("tvUsername")).setText(item.getUsername());
            ((TextView)
holder.getView("tvElo")).setText(String.valueOf(Math.round(item.getElo())));

            if (currentUser != null && item.getUsername().equals(currentUser.getUsername())) {
                holder.itemView.setBackgroundColor(ContextCompat.getColor(this,
R.color.colorAccent));
                ((TextView) holder.getView("tvUsername")).setTypeface(null, Typeface.BOLD);
                ((TextView) holder.getView("tvElo")).setTypeface(null, Typeface.BOLD);
            } else {
                holder.itemView.setBackgroundColor(Color.TRANSPARENT);
                ((TextView) holder.getView("tvUsername")).setTypeface(null, Typeface.NORMAL);
                ((TextView) holder.getView("tvElo")).setTypeface(null, Typeface.NORMAL);
            }
        })
    );

    rvLeaderboard.setAdapter(adapter);
    rvLeaderboard.setLayoutManager(new LinearLayoutManager(this));
}

/**
 * Loads users for the leaderboard, paginated. Shows loading indicator if loading more.
 *
 * @param loadMore Whether to load more users (pagination).
 */
private void loadUsers(boolean loadMore) {
    isLoading = true;
    if (loadMore) {
        showLoadingMore();
    }
    viewModel.getTopPlayersPaginated(PAGE_SIZE, lastLoadedElo, lastLoadedIds);
}

/**
 * Shows a loading indicator in the leaderboard list.
 */

```



```
private void showLoadingMore() {  
    adapter.getItems().add(null);  
    adapter.notifyItemInserted(adapter.getItemCount() - 1);  
}  
}
```



LoginActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.emil_z.helper.TextInputLayoutUtil;
import com.emil_z.helper.UserSessionPreference;
import com.emil_z.helper.inputValidators.Rule;
import com.emil_z.helper.inputValidators.RuleOperation;
import com.emil_z.helper.inputValidators.Validator;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.UsersViewModel;

/**
 * Activity that handles user login.
 * <p>
 * Validates user credentials, manages session preferences, and navigates to the main activity upon
 * successful login.
 */
public class LoginActivity extends BaseActivity {
    private EditText etUsername;
    private EditText etPassword;
    private CheckBox cbRememberMe;
    private Button btnSignIn;
    private Button btnBack;

    private UsersViewModel viewModel;

    /**
     * Sets up click listeners for registration and back buttons.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        EdgeToEdge.enable(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
```



```
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    initializeViews();
    setListeners();
    setViewModel();
}

/**
 * Initializes view components for login.
 */
@Override
protected void initializeViews() {
    etUsername = findViewById(R.id.etUsername);
    etPassword = findViewById(R.id.etPassword);
    cbRememberMe = findViewById(R.id.cbRememberMe);
    btnSignIn = findViewById(R.id.btnSignIn);
    btnBack = findViewById(R.id.btnBack);
}

/**
 * Sets up click listeners for sign-in and back buttons.
 */
@Override
protected void setListeners() {
    btnSignIn.setOnClickListener(v -> {
        if (validate()) {
            String username = etUsername.getText().toString();
            String password = etPassword.getText().toString();
            viewModel.logIn(username, password);
        }
    });
    btnBack.setOnClickListener(v -> finish());
}

/**
 * Sets up click listeners for sign-in and back buttons.
 */
@Override
protected void setViewModel() {
    viewModel = new ViewModelProvider(this).get(UsersViewModel.class);

    viewModel.getLiveDataSuccess().observe(this, success -> {
        if (!success) {
            Toast.makeText(LoginActivity.this, R.string.invalid_credentials,
                Toast.LENGTH_SHORT).show();
        }
    });
}
```



```
viewModel.getLiveDataEntity().observe(this, user -> {
    if (user != null) {
        BaseActivity.currentUser = user;
        UserSessionPreference sessionPreference = new UserSessionPreference(this);
        if (cbRememberMe.isChecked())
            sessionPreference.saveFullSession(user.getUsername(), user.getHashedPassword(),
            user.getIdFs(), sessionPreference.generateToken(user.getIdFs()));
        else
            sessionPreference.saveLoginCredentials(user.getUsername(), user.getHashedPassword(),
            user.getIdFs());
        Intent intent = new Intent(LoginActivity.this, MainActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }
});
}

/**
 * Sets up validation rules for login fields.
 */
public void setValidation() {
    Validator.clear();
    Validator.add(new Rule(etUsername, RuleOperation.REQUIRED, getString(R.string.no_username)));
    Validator.add(new Rule(etPassword, RuleOperation.REQUIRED, getString(R.string.no_password)));
}

/**
 * Validates login fields and updates error messages.
 *
 * @return true if all fields are valid, false otherwise.
 */
public boolean validate() {
    setValidation();
    boolean isValid = Validator.validate();

    TextInputLayoutUtil.transferErrorsToTextInputLayout(etUsername);
    TextInputLayoutUtil.transferErrorsToTextInputLayout(etPassword);
    return isValid;
}
}
```



MainActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;
import android.widget.ImageView;

import androidx.activity.EdgeToEdge;
import androidx.activity.OnBackPressedCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.emil_z.helper.AlertUtil;
import com.emil_z.model.GameType;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.UsersViewModel;

import java.util.Random;

/**
 * Main activity for the Ultimate Tic Tac Toe app.
 * <p>
 * Handles navigation to profile, leaderboard, settings, and game activities.
 * Manages user interactions for starting different game modes and handles back press events.
 */
public class MainActivity extends BaseActivity {
    public static final String EXTRA_GAME_TYPE = "com.emil_z.EXTRA_GAME_TYPE";
    public static final String EXTRA_SIGN = "com.emil_z.EXTRA_SIGN";
    public static final String EXTRA_GAME_ID_FS = "com.emil_z.EXTRA_GAME_ID_FS";

    private ImageView ivProfile;
    private ImageView ivLeaderboard;
    private ImageView ivSettings;
    private Button btnCPU;
    private Button btnLocal;
    private Button btnOnline;

    private UsersViewModel viewModel;
    private ActivityResultLauncher<Intent> gameLauncher;
    private ActivityResultLauncher<Intent> profileLauncher;

    /**
     * Initializes the main activity, sets up UI, listeners, ViewModel, and activity result
     * launchers.
     */
}
```



```
*
* @param savedInstanceState The previously saved instance state, if any.
*/
@Override
protected void onCreate(Bundle savedInstanceState) {
    EdgeToEdge.enable(this);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ViewCompat.setOnApplyWindowInsetsListener(
        findViewById(R.id.main),
        (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

    initializeViews();
    setListeners();
    setViewModel();
    registerLaunchers();
}

/**
 * Initializes view components for the main activity.
 */
@Override
protected void initializeViews() {
    ivProfile = findViewById(R.id.ivProfile);
    ivLeaderboard = findViewById(R.id.ivLeaderboard);
    ivSettings = findViewById(R.id.ivSettings);
    btnCPU = findViewById(R.id.btnCpu);
    btnLocal = findViewById(R.id.btnLocal);
    btnOnline = findViewById(R.id.btnOnline);

    ivProfile.setImageBitmap(currentUser.getPictureBitmap());
}

/**
 * Sets up click listeners for profile, leaderboard, settings, and game mode buttons.
 * Handles back press with a confirmation dialog.
 */
@Override
protected void setListeners() {
    ivProfile.setOnClickListener(v -> {
        Intent intent = new Intent(MainActivity.this, ProfileActivity.class);
        profileLauncher.launch(intent);
    });
}
```



```
ivLeaderboard.setOnClickListener(v -> {
    Intent intent = new Intent(MainActivity.this, LeaderboardActivity.class);
    startActivity(intent);
});

ivSettings.setOnClickListener(v -> {
    Intent intent = new Intent(MainActivity.this, SettingsActivity.class);
    startActivity(intent);
});

btnCPU.setOnClickListener(v -> {
    //TODO: add an option to choose the level of the CPU (will do when the CPU will be better)
    AlertUtil.alert(this,
        getString(R.string.play_against_ai),
        getString(R.string.choose_your_sign),
        true,
        R.drawable.cpu_pfp,
        getString(R.string.cross),
        getString(R.string.nought),
        getString(R.string.random),
        () -> startGameActivity(GameType.CPU, 'X'),
        () -> startGameActivity(GameType.CPU, 'O'),
        () -> startGameActivity(GameType.CPU, new Random().nextBoolean() ? 'X' : 'O'));
});

btnLocal.setOnClickListener(v -> startGameActivity(GameType.LOCAL));

btnOnline.setOnClickListener(v -> startGameActivity(GameType.ONLINE));

getOnBackPressedDispatcher().addCallback(this, new OnBackPressedCallback(true) {
    @Override
    public void handleOnBackPressed() {
        AlertUtil.alert(MainActivity.this,
            getString(R.string.exit),
            getString(R.string.exit_confirmation),
            true,
            0,
            getString(R.string.yes),
            getString(R.string.no),
            null,
            () -> finish(),
            null,
            null);
    }
});
}

/**
 * Initializes the ViewModel and observes user data updates.
```




```
*/
@Override
protected void setViewModel() {
    viewModel = new ViewModelProvider(this).get(UsersViewModel.class);

    viewModel.getLiveDataEntity().observe(this, user -> {
        if (user != null) {
            currentUser = user;
        }
    });
}

/**
 * Registers activity result launchers for game and profile activities.
 */
@SuppressWarnings("ConstantConditions")
private void registerLaunchers() {
    gameLauncher = registerForActivityResult(new
ActivityResultContracts.StartActivityForResult(),
        o -> {
            GameType gameType = (GameType) o.getData().getSerializableExtra(EXTRA_GAME_TYPE);
            if (gameType == GameType.ONLINE && o.getResultCode() == RESULT_OK) {
                viewModel.get(currentUser.getIdFs());
            }
        }
    );
    profileLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == RESULT_OK) {
                ivProfile.setImageBitmap(currentUser.getPictureBitmap());
            }
        }
    );
}

/**
 * Starts a game activity with the specified game type and default sign.
 *
 * @param gameType The type of game to start.
 */
private void startGameActivity(GameType gameType) {
    startGameActivity(gameType, '-');
}

/**
 * Starts a game activity with the specified game type and player sign.
 *
 * @param gameType The type of game to start.
 * @param sign      The player's sign ('X', 'O', or '-').
 */
}
```



```
*/  
private void startGameActivity(GameType gameType, char sign) {  
    Intent intent = new Intent(MainActivity.this, GameActivity.class);  
    intent.putExtra(EXTRA_GAME_TYPE, gameType);  
    intent.putExtra(EXTRA_SIGN, sign);  
    gameLauncher.launch(intent);  
}  
}
```



ProfileActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.EdgeToEdge;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.core.content.ContextCompat;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.DividerItemDecoration;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.emil_z.helper.AlertUtil;
import com.emil_z.helper.BitMapHelper;
import com.emil_z.helper.Global;
import com.emil_z.model.Game;
import com.emil_z.model.GameType;
import com.emil_z.model.Games;
import com.emil_z.model.Player;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.ADPTERS.GamesAdapter;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.GamesViewModel;
import com.emil_z.viewmodel.GamesViewModelFactory;
import com.emil_z.viewmodel.UsersViewModel;
import com.yalantis.ucrop.UCrop;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Objects;

/**
 * Activity that displays the user's profile, including profile picture, username, ELO rating,
 * and a paginated list of recent games. Allows updating the profile picture via camera or gallery,
 * and supports cropping the selected image.
 */
```



```
*/
public class ProfileActivity extends BaseActivity {
    private final int PAGE_SIZE = 10;

    private ImageView ivPfp;
    private TextView tvUsername;
    private TextView tvElo;
    private RecyclerView rvGames;

    private GamesViewModel gamesViewModel;
    private UsersViewModel usersViewModel;
    private GamesAdapter adapter;

    private ActivityResultLauncher<Void> cameraLauncher;
    private ActivityResultLauncher<Intent> galleryLauncher;
    private ActivityResultLauncher<String> requestPermissionLauncher;

    private Games games;
    private boolean isLoading = false;
    private String lastLoadedGameId = null;

    /**
     * Initializes the profile activity, sets up UI, listeners, ViewModels, adapter, and launchers.
     *
     * @param savedInstanceState The previously saved instance state, if any.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        EdgeToEdge.enable(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        initializeViews();
        setListeners();
        setupRecyclerViewScrollListener();
        setViewModel();
        setAdapter();
        registerLaunchers();
        showLoadingMore();
    }

    /**
     * Initializes view components for the profile screen.
     */
    @SuppressWarnings("ConstantConditions")
```



```
@Override
public void initializeViews() {
    ivPfp = findViewById(R.id.ivPfp);
    tvUsername = findViewById(R.id.tvUsername);
    tvElo = findViewById(R.id.tvElo);
    rvGames = findViewById(R.id.rvGames);

    ivPfp.setImageBitmap(currentUser.getPictureBitmap());
    tvUsername.setText(currentUser.getUsername());
    tvElo.setText(getString(R.string.elo_format, Math.round(currentUser.getElo())));

    DividerItemDecoration divider = new DividerItemDecoration(this,
LinearLayoutManager.VERTICAL);
    divider.setDrawable(ContextCompat.getDrawable(this, R.drawable.dividor));
    rvGames.addItemDecoration(divider);
}

/**
 * Sets up click listeners for profile picture.
 */
@Override
protected void setListeners() {
    ivPfp.setOnClickListener(v -> showImageOptions());
}

/**
 * Sets up a scroll listener for the RecyclerView to load more games when reaching the end.
 */
@SuppressWarnings("ConstantConditions")
private void setupRecyclerViewScrollListener() {
    rvGames.addOnScrollListener(new RecyclerView.OnScrollListener() {
        @Override
        public void onScrolled(@NonNull RecyclerView recyclerView, int dx, int dy) {
            super.onScrolled(recyclerView, dx, dy);

            LinearLayoutManager layoutManager = (LinearLayoutManager)
recyclerView.getLayoutManager();
            int visibleItemCount = layoutManager.getChildCount();
            int totalItemCount = layoutManager.getItemCount();
            int firstVisibleItemPosition = layoutManager.findFirstVisibleItemPosition();

            if (!isLoading && (visibleItemCount + firstVisibleItemPosition) >= totalItemCount
                && firstVisibleItemPosition >= 0
                && totalItemCount >= PAGE_SIZE) {
                // Load more games
                loadGames(true);
            }
        }
    });
}
```



```
/**
 * Initializes ViewModels, loads games, and observes user and games data changes.
 */
@Override
protected void setViewModel() {
    gamesViewModel = new ViewModelProvider(this, new GamesViewModelFactory(getApplication(),
    GameType.ONLINE)).get(GamesViewModel.class);
    usersViewModel = new ViewModelProvider(this).get(UsersViewModel.class);
    loadGames(false);

    usersViewModel.getLiveDataSuccess().observe(this, success -> {
        if (success) {
            ivPfp.setImageBitmap(currentUser.getPictureBitmap());
            setResult(RESULT_OK);
        } else {
            usersViewModel.get(currentUser.getIdFs());
        }
    });

    usersViewModel.getLiveDataEntity().observe(this, user -> {
        if (user != null) {
            for (Game game : games) {
                if (game == null) continue;
                if (Objects.equals(currentUser.getIdFs(), game.getPlayer1().getIdFs())) {
                    game.getPlayer2().setPicture(user.getPicture());
                } else {
                    game.getPlayer1().setPicture(user.getPicture());
                }
            }
        }
        adapter.setItems(this.games);
    });

    gamesViewModel.getLiveDataCollection().observe(this, newGames -> {
        if (adapter.getItems() != null) {
            int lastIndex = adapter.getItems().indexOf(null);
            if (lastIndex != -1) {
                adapter.getItems().remove(lastIndex);
                adapter.notifyItemRemoved(lastIndex);
            }
        }

        if (!newGames.isEmpty()) {
            if (this.games == null)
                this.games = newGames;
            else
                this.games.addAll(newGames);
            isLoading = false;
            lastLoadedGameId = newGames.get(newGames.size() - 1).getIdFs();
        }
    });
}
```



```
        for (Game game : newGames) {
            usersViewModel.get(Objects.equals(currentUser.getIdFs(),
                game.getPlayer1().getIdFs()) ? game.getPlayer2().getIdFs() :
                game.getPlayer1().getIdFs());
        }
    }
});
}

/**
 * Sets up the adapter for the games RecyclerView and handles item clicks.
 */
public void setAdapter() {
    adapter = new GamesAdapter(new ArrayList<>(),
        R.layout.game_single_layout,
        holder -> {
            holder.putView("ivPfp", holder.itemView.findViewById(R.id.ivPfp));
            holder.putView("tvUsername", holder.itemView.findViewById(R.id.tvUsername));
            holder.putView("tvElo", holder.itemView.findViewById(R.id.tvElo));
            holder.putView("ivGameResult", holder.itemView.findViewById(R.id.ivGameResult));
        },
        ((holder, item, position) -> {
            Player opponent = (Objects.equals(item.getPlayer1().getIdFs(), currentUser.getIdFs()) ?
            item.getPlayer2() : item.getPlayer1());
            ((ImageView) holder.getView("ivPfp")).setImageBitmap(opponent.getPictureBitmap());
            ((TextView) holder.getView("tvUsername")).setText(opponent.getName());
            ((TextView) holder.getView("tvElo")).setText(getString(R.string.player_elo_format,
            Math.round(opponent.getElo())));
            if (Objects.equals(item.getWinnerIdFs(), currentUser.getIdFs()))
                ((ImageView) holder.getView("ivGameResult")).setImageResource(R.drawable.checkmark);
            else if (Objects.equals(item.getWinnerIdFs(), "T"))
                ((ImageView) holder.getView("ivGameResult")).setImageResource(R.drawable.tie);
            else
                ((ImageView) holder.getView("ivGameResult")).setImageResource(R.drawable.x);
        })
    );

    rvGames.setAdapter(adapter);
    rvGames.setLayoutManager(new LinearLayoutManager(this));

    adapter.setOnItemClickListener((item, position) -> {
        Intent intent = new Intent(this, GameActivity.class);
        intent.putExtra(MainActivity.EXTRA_GAME_ID_FS, item.getIdFs());
        intent.putExtra(MainActivity.EXTRA_GAME_TYPE, GameType.REPLAY);
        startActivity(intent);
    });
}

/**
 * Registers activity result launchers for camera and gallery image selection.
```



```
*/
private void registerLaunchers() {
    cameraLauncher = registerForActivityResult(
        new ActivityResultContracts.TakePicturePreview(),
        bitMap -> {
            if (bitMap != null)
                processNewProfileImage(bitMap);
        });

    galleryLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        result -> {
            if (result.getResultCode() == RESULT_OK && result.getData() != null) {
                try {
                    Bitmap bitmap = MediaStore.Images.Media.getBitmap(
                        getContentResolver(),
                        result.getData().getData()
                    );
                    processNewProfileImage(bitmap);
                } catch (Exception e) {
                    Toast.makeText(this, R.string.failed_to_load_image, Toast.LENGTH_SHORT).show();
                }
            }
        });

    requestPermissionLauncher = registerForActivityResult(
        new ActivityResultContracts.RequestPermission(),
        isGranted -> {
            if (isGranted) {
                if (Global.getCurrentRequestType() == 0)
                    cameraLauncher.launch(null);
                else
                    galleryLauncher.launch(new Intent(Intent.ACTION_GET_CONTENT).setType("image/*"));
            } else {
                AlertUtil.alertOk(
                    this,
                    getString(R.string.permission_required),
                    getString(R.string.permission_required_message),
                    true,
                    0
                );
            }
        });
}

/**
 * Displays a dialog or options for the user to choose between taking a new profile picture
 * using the camera or selecting one from the gallery. Handles permission requests as needed.
 */
```




```
*/
private void showImageOptions() {
    Global.takePicture(this, cameraLauncher, galleryLauncher, requestPermissionLauncher);
}

/**
 * Processes a new profile image, resizes it, and starts the crop activity.
 *
 * @param bitmap The new profile image bitmap.
 */
private void processNewProfileImage(Bitmap bitmap) {
    try {
        File outputDir = getCacheDir();
        File outputFile = File.createTempFile("temp_image", ".jpg", outputDir);

        Bitmap resizedBitmap = getResizedBitmap(bitmap);
        FileOutputStream fos = new FileOutputStream(outputFile);
        resizedBitmap.compress(Bitmap.CompressFormat.JPEG, 85, fos);
        fos.close();

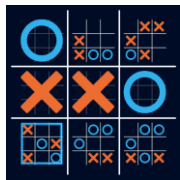
        startCropActivity(Uri.fromFile(outputFile));
    } catch (IOException e) {
        Toast.makeText(this, R.string.failed_to_load_image, Toast.LENGTH_SHORT).show();
    }
}

/**
 * Resizes a bitmap to a maximum dimension of 512px, maintaining aspect ratio.
 *
 * @param image The original bitmap.
 * @return The resized bitmap.
 */
private Bitmap getResizedBitmap(Bitmap image) {
    int width = image.getWidth();
    int height = image.getHeight();

    float bitmapRatio = (float) width / (float) height;
    if (bitmapRatio > 1) {
        width = 512;
        height = (int) (width / bitmapRatio);
    } else {
        height = 512;
        width = (int) (height * bitmapRatio);
    }

    return Bitmap.createScaledBitmap(image, width, height, true);
}

/**
 * Starts the crop activity for the selected image.
 */
```



```
*
* @param sourceUri The URI of the image to crop.
*/
private void startCropActivity(Uri sourceUri) {
    UCrop.Options options = new UCrop.Options();
    options.setToolbarColor(getResources().getColor(R.color.colorBackground, getTheme()));
    options.setStatusBarColor(getResources().getColor(R.color.colorBackground, getTheme()));
    options.setToolbarTitle("Crop Image");

    options.setToolbarWidgetColor(getResources().getColor(R.color.textColor, getTheme()));

    options.withAspectRatio(1, 1);

    File destinationFile = new File(getCacheDir(), "cropped_image_" + System.currentTimeMillis()
+ ".jpg");
    Uri destinationUri = Uri.fromFile(destinationFile);

    UCrop.of(sourceUri, destinationUri)
        .withOptions(options)
        .start(this);
}

/**
 * Loads games for the user, paginated. Shows loading indicator if loading more.
 *
 * @param loadMore Whether to load more games (pagination).
 */
private void loadGames(boolean loadMore) {
    isLoading = true;
    if (loadMore) {
        showLoadingMore();
    }
    gamesViewModel.getUserGamesPaginated(currentUser.getIdFs(), PAGE_SIZE, lastLoadedGameId);
}

/**
 * Shows a loading indicator in the games list.
 */
private void showLoadingMore() {
    adapter.getItems().add(null);
    adapter.notifyItemInserted(adapter.getItemCount() - 1);
}

/**
 * Handles the result from the crop activity and updates the profile picture.
 *
 * @param requestCode The request code.
 * @param resultCode The result code.
 * @param data The intent data.
 */
```



```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == UCrop.REQUEST_CROP && resultCode == RESULT_OK && data != null) {
        Uri resultUri = UCrop.getOutput(data);
        try {
            Bitmap croppedBitmap = MediaStore.Images.Media.getBitmap(getContentResolver(),
resultUri);
            // Now update the image view and save to user profile
            ivPfp.setImageBitmap(croppedBitmap);
            String base64Image = BitMapHelper.encodeTobase64(croppedBitmap);
            currentUser.setPicture(base64Image);
            usersViewModel.update(currentUser);
        } catch (IOException e) {
            Toast.makeText(this, "Failed to load cropped image", Toast.LENGTH_SHORT).show();
        }
    }
}
```



RegisterActivity.java

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import androidx.activity.EdgeToEdge;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
import androidx.lifecycle.ViewModelProvider;

import com.emil_z.helper.BitMapHelper;
import com.emil_z.helper.PasswordUtil;
import com.emil_z.helper.TextInputLayoutUtil;
import com.emil_z.helper.inputValidators.CompareRule;
import com.emil_z.helper.inputValidators.NameRule;
import com.emil_z.helper.inputValidators.PasswordRule;
import com.emil_z.helper.inputValidators.Rule;
import com.emil_z.helper.inputValidators.RuleOperation;
import com.emil_z.helper.inputValidators.Validator;
import com.emil_z.model.User;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;
import com.emil_z.viewmodel.UsersViewModel;
import com.google.android.material.textfield.TextInputLayout;

/**
 * Activity that handles user registration.
 * <p>
 * Validates user input, checks for existing usernames, and saves new user data.
 */
public class RegisterActivity extends BaseActivity {
    private EditText etUsername;
    private EditText etPassword;
    private EditText etConfirmPassword;
    private TextView tvError;
    private Button btnRegister;
    private Button btnBack;

    private UsersViewModel viewModel;

    /**
     * Initializes the registration activity, sets up the UI,
     * and prepares listeners and ViewModel.
     */
}
```



```
*
* @param savedInstanceState The previously saved instance state, if any.
*/
@Override
protected void onCreate(Bundle savedInstanceState) {
    EdgeToEdge.enable(this);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
        Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
        return insets;
    });

    initializeViews();
    setListeners();
    setViewModel();
}

/**
 * Initializes view components for registration.
 */
@Override
protected void initializeViews() {
    etUsername = findViewById(R.id.etUsername);
    etPassword = findViewById(R.id.etPassword);
    etConfirmPassword = findViewById(R.id.etConfirmPassword);
    tvError = findViewById(R.id.tvError);
    btnRegister = findViewById(R.id.btnRegister);
    btnBack = findViewById(R.id.btnBack);
}

/**
 * Sets up click listeners for registration and back buttons.
 */
@Override
protected void setListeners() {
    btnRegister.setOnClickListener(v -> {
        if (validate()) {
            viewModel.exist(etUsername.getText().toString());
        }
    });
    btnBack.setOnClickListener(v -> finish());
}

/**
 * Initializes the ViewModel and observes username existence.
 */
@Override
```



```
protected void setViewModel() {
    viewModel = new ViewModelProvider(this).get(UsersViewModel.class);

    viewModel.getLiveDataExist().observe(this, exist -> {
        if (exist) {
            etUsername.setError(getString(R.string.username_taken));
            TextInputLayoutUtil.transferErrorsToTextInputLayout(etUsername);
        } else {
            etUsername.setError(null);
            registerUser();
        }
    });
}

/**
 * Registers a new user and navigates to the login screen.
 */
protected void registerUser() {
    Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.default_pfp);
    String hashedPassword = PasswordUtil.hashPassword(etPassword.getText().toString());
    User user = new User(etUsername.getText().toString(),
        hashedPassword,
        BitMapHelper.encodeTobase64(bitmap));
    viewModel.save(user);
    Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
    startActivity(intent);
    finish();
}

/**
 * Sets up validation rules for registration fields.
 */
public void setValidation() {
    Validator.clear();
    Validator.add(new Rule(etUsername, RuleOperation.REQUIRED, getString(R.string.no_username)));
    Validator.add(new NameRule(etUsername, RuleOperation.NAME,
        getString(R.string.username_invalid)));
    Validator.add(new Rule(etPassword, RuleOperation.REQUIRED, getString(R.string.no_password)));
    Validator.add(new PasswordRule(etPassword, RuleOperation.PASSWORD,
        getString(R.string.password_invalid), 8, 64));
    Validator.add(new Rule(etConfirmPassword, RuleOperation.REQUIRED,
        getString(R.string.no_confirm_password)));
    Validator.add(new CompareRule(etConfirmPassword, etPassword, RuleOperation.COMPARE,
        getString(R.string.passwords_dont_match)));
}

/**
 * Validates registration fields and updates error messages.
 *
 * @return true if all fields are valid, false otherwise.
 */
```



```
*/  
public boolean validate() {  
    setValidation();  
    boolean isValid = Validator.validate();  
  
    TextInputLayoutUtil.transferErrorsToTextInputLayout(etUsername);  
    TextInputLayout til = TextInputLayoutUtil.getTextInputLayout(etPassword);  
    boolean hasPasswordError = etPassword.getError() != null;  
    til.setError(hasPasswordError ? "" : null);  
    etPassword.setError(null);  
    tvError.setVisibility(hasPasswordError ? TextView.VISIBLE : TextView.INVISIBLE);  
  
    TextInputLayoutUtil.transferErrorsToTextInputLayout(etConfirmPassword);  
    return isValid;  
}  
}
```



Settings.activity

```
package com.emil_z.ultimate_tic_tac_toe.ACTIVITIES;

import android.content.Intent;
import android.os.Bundle;
import android.widget.Button;

import androidx.activity.EdgeToEdge;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.emil_z.helper.UserSessionPreference;
import com.emil_z.ultimate_tic_tac_toe.ACTIVITIES.BASE.BaseActivity;
import com.emil_z.ultimate_tic_tac_toe.R;

/**
 * Activity for managing user settings, including logging out.
 * <p>
 * Handles UI initialization, button listeners, and user session management.
 */
public class SettingsActivity extends BaseActivity {
    private Button btnLogOut;

    /**
     * Initializes the settings activity, sets up UI, listeners, and window insets.
     *
     * @param savedInstanceState The previously saved instance state, if any.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        EdgeToEdge.enable(this);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_settings);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });

        initializeViews();
        setListeners();
    }

    /**
     * Initializes view components for the settings screen.
     */
    @Override
    protected void initializeViews() {
```




```
        btnLogOut = findViewById(R.id.btnLogOut);
    }

    /**
     * Sets up click listeners for the settings screen.
     */
    @Override
    protected void setListeners() {
        btnLogOut.setOnClickListener(v -> logOut());
    }

    /**
     * No ViewModel setup required for this activity.
     */
    @Override
    protected void setViewModel() {
    }

    /**
     * Logs out the current user by clearing session data and navigating to the authentication
screen.
     * Clears the activity stack to prevent navigation back to previous activities.
     */
    private void logOut() {
        new UserSessionPreference(this).clearLoginCredentials();
        BaseActivity.currentUser = null;
        Intent intent = new Intent(this, AuthActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
        startActivity(intent);
    }
}
```



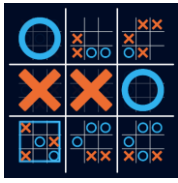
activity_auth.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES. AuthActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="50dp"
        android:fontFamily="@font/montserrat_bold"
        android:gravity="center"
        android:text="@string/ultimate_tic_tac_toe_online"
        android:textColor="#F2F3F4"
        android:textSize="50sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvCreator"
        android:layout_width="wrap_content"
```



```
android:layout_height="wrap_content"
android:layout_marginTop="8dp"
android:fontFamily="@font/montserrat_medium"
android:text="@string/by_emil_zallessky"
android:textColor="#D0E1E1"
android:textSize="34sp"
app:layout_constraintEnd_toEndOf="@+id/tvTitle"
app:layout_constraintStart_toStartOf="@+id/tvTitle"
app:layout_constraintTop_toBottomOf="@+id/tvTitle" />
```

<Button

```
android:id="@+id/btnLogin"
android:layout_width="290dp"
android:layout_height="110dp"
android:layout_marginTop="430dp"
android:backgroundTint="@color/buttonColor"
android:backgroundTintMode="src_atop"
android:fontFamily="@font/montserrat_semibold"
android:text="@string/sign_in"
android:textSize="34sp"
app:cornerRadius="15dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

<Button

```
android:id="@+id/btnRegister"
android:layout_width="0dp"
android:layout_height="110dp"
android:layout_marginTop="24dp"
android:backgroundTint="@color/buttonColor"
android:fontFamily="@font/montserrat_semibold"
android:text="@string/register"
android:textSize="34sp"
app:cornerRadius="15dp"
app:layout_constraintEnd_toEndOf="@+id/btnLogin"
app:layout_constraintStart_toStartOf="@+id/btnLogin"
app:layout_constraintTop_toBottomOf="@+id/btnLogin" />
```

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_game.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorBackground">

    <androidx.gridlayout.widget.GridLayout
        android:id="@+id/gridBoard"
        android:layout_width="403dp"
        android:layout_height="403dp"
        android:layout_gravity="center"
        android:background="@drawable/board_blurred"
        app:columnCount="3"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:rowCount="3">

    </androidx.gridlayout.widget.GridLayout>

    <LinearLayout
        android:id="@+id/l1P2"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="top"
        android:layout_marginTop="16dp"
        android:orientation="horizontal"
        android:visibility="invisible"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <Space
            android:layout_width="16dp"
            android:layout_height="match_parent"
            android:layout_weight="0" />

        <com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/ivP2Pfp"
            android:layout_width="50dp"
            android:layout_height="match_parent"
            app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip">
```



```
tools:srcCompat="@drawable/default_pfp" />
```

```
<Space
```

```
    android:layout_width="10dp"
    android:layout_height="match_parent"
    android:layout_weight="0" />
```

```
<TextView
```

```
    android:id="@+id/tvP2Name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0"
    android:textColor="@color/textColor"
    android:textSize="20sp"
    tools:text="{PLAYER_2_NAME}:" />
```

```
<Space
```

```
    android:layout_width="6dp"
    android:layout_height="match_parent"
    android:layout_weight="0" />
```

```
<TextView
```

```
    android:id="@+id/tvP2ELO"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0"
    android:textColor="@color/textColor"
    android:textSize="20sp"
    tools:text="{PLAYER_2_ELO}" />
```

```
<Space
```

```
    android:layout_width="5dp"
    android:layout_height="match_parent"
    android:layout_weight="0" />
```

```
<TextView
```

```
    android:id="@+id/tvP2Sign"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0"
    android:textColor="@color/textColor"
    android:textSize="20sp"
    tools:text="{SIGN}" />
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:id="@+id/llP1"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_gravity="bottom"
```



```
android:layout_marginBottom="16dp"  
android:orientation="horizontal"  
android:visibility="invisible"  
app:layout_constraintBottom_toBottomOf="parent"  
tools:layout_editor_absoluteX="0dp">
```

<Space

```
android:id="@+id/space1"  
android:layout_width="16dp"  
android:layout_height="match_parent"  
android:layout_weight="0" />
```

<com.google.android.material.imageview.ShapeableImageView

```
android:id="@+id/ivP1Pfp"  
android:layout_width="50dp"  
android:layout_height="match_parent"  
android:layout_weight="0"  
app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"  
tools:srcCompat="@drawable/default_pfp" />
```

<Space

```
android:layout_width="10dp"  
android:layout_height="match_parent"  
android:layout_weight="0" />
```

<TextView

```
android:id="@+id/tvP1Name"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_weight="0"  
android:fontFamily="@font/montserrat_medium"  
android:textColor="@color/textColor"  
android:textSize="20sp"  
tools:text="{PLAYER_1_NAME}:" />
```

<Space

```
android:layout_width="6dp"  
android:layout_height="match_parent"  
android:layout_weight="0" />
```

<TextView

```
android:id="@+id/tvP1ELO"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:layout_weight="0"  
android:textColor="@color/textColor"  
android:textSize="20sp"  
tools:text="{PLAYER_1_ELO}" />
```

<Space



```
android:layout_width="5dp"
android:layout_height="match_parent"
android:layout_weight="0" />
```

```
<TextView
    android:id="@+id/tvP1Sign"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="0"
    android:textColor="@color/textColor"
    android:textSize="20sp"
    tools:text="{SIGN}" />
</LinearLayout>

<TextView
    android:id="@+id/tvCurrentPlayer"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/montserrat_medium"
    android:text="@string/player_x_turn"
    android:textColor="@color/textColor"
    android:textSize="34sp"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/gridBoard"
    app:layout_constraintEnd_toEndOf="@+id/gridBoard"
    app:layout_constraintStart_toStartOf="@+id/gridBoard"
    app:layout_constraintTop_toBottomOf="@+id/llP2"
    app:layout_constraintVertical_bias="0.25" />
```

```
<Button
    android:id="@+id/btnAbort"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:backgroundTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_bold"
    android:text="@string/abort_search"
    android:textColor="@color/textColor"
    android:textSize="37sp"
    android:visibility="gone"
    app:cornerRadius="50dp"
    app:layout_constraintBottom_toBottomOf="@+id/gridBoard"
    app:layout_constraintEnd_toEndOf="@+id/gridBoard"
    app:layout_constraintStart_toStartOf="@+id/gridBoard"
    app:layout_constraintTop_toTopOf="@+id/gridBoard" />
```

```
<LinearLayout
    android:id="@+id/llReview"
    android:layout_width="335dp"
    android:layout_height="100dp"
    android:orientation="horizontal"
```



```
android:visibility="gone"
app:layout_constraintBottom_toTopOf="@+id/llp1"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/gridBoard"
app:layout_constraintVertical_bias="0.41000003">
```

<Button

```
android:id="@+id/btnBackward"
android:layout_width="160dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:backgroundTint="@color/hintTextColor"
android:fontFamily="@font/montserrat_bold"
android:text="@string/review_back"
android:textColor="@color/textColor"
android:textSize="25sp"
app:cornerRadius="15dp" />
```

<Space

```
android:layout_width="15dp"
android:layout_height="match_parent"
android:layout_weight="1" />
```

<Button

```
android:id="@+id/btnForward"
android:layout_width="160dp"
android:layout_height="match_parent"
android:layout_weight="1"
android:backgroundTint="@color/buttonColor"
android:fontFamily="@font/montserrat_bold"
android:text="@string/review_forward"
android:textColor="@color/textColor"
android:textSize="25sp"
app:cornerRadius="15dp" />
```

</LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_leaderboard.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES. LeaderboardActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:fontFamily="@font/montserrat_bold"
        android:text="@string/leaderboard"
        android:textColor="@color/textColor"
        android:textSize="34sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvRank"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="6dp"
```



```
android:layout_marginBottom="4dp"
android:fontFamily="@font/montserrat_medium"
android:text="@string/rank"
android:textColor="@color/textColor"
android:textSize="16sp"
app:layout_constraintBottom_toTopOf="@+id/rvLeaderboard"
app:layout_constraintStart_toStartOf="parent" />
```

<TextView

```
android:id="@+id/tvPlayer"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="64dp"
android:fontFamily="@font/montserrat_medium"
android:text="@string/player"
android:textColor="@color/textColor"
android:textSize="16sp"
app:layout_constraintBottom_toBottomOf="@+id/tvRank"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="@+id/tvRank"
app:layout_constraintVertical_bias="0.0" />
```

<TextView

```
android:id="@+id/tvRating"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginEnd="8dp"
android:fontFamily="@font/montserrat_medium"
android:text="@string/elo_rating"
android:textColor="@color/textColor"
android:textSize="16sp"
app:layout_constraintBottom_toBottomOf="@+id/tvRank"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="@+id/tvRank" />
```

<androidx.recyclerview.widget.RecyclerView

```
android:id="@+id/rvLeaderboard"
android:layout_width="0dp"
android:layout_height="0dp"
android:layout_marginTop="40dp"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvTitle" />
```

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES.LoginActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:fontFamily="@font/montserrat_semibold"
        android:text="@string/sign_in"
        android:textColor="@color/textColor"
        android:textSize="50sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/tilUsername"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
```



```
android:layout_marginTop="60dp"
android:hint="@string/username"
android:textColorHint="@color/hintTextColor"
app:boxStrokeColor="@color/colorPrimary"
app:cursorColor="@color/textColor"
app:hintTextColor="@color/colorPrimary"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvTitle">
```

```
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/etUsername"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:autofillHints="username"
    android:fontFamily="@font/montserrat_medium"
    android:inputType="text"
    android:textColor="@color/textColor"
    android:textColorHint="#1E88E5" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tilPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:hint="@string/password"
    android:textColorHint="@color/hintTextColor"
    app:boxStrokeColor="@color/colorPrimary"
    app:cursorColor="@color/textColor"
    app:hintTextColor="@color/colorPrimary"
    app:layout_constraintEnd_toEndOf="@+id/tilUsername"
    app:layout_constraintStart_toStartOf="@+id/tilUsername"
    app:layout_constraintTop_toBottomOf="@+id/tilUsername"
    app:passwordToggleEnabled="true"
    app:passwordToggleTint="@color/colorPrimary">
```

```
<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/etPassword"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:autofillHints="password"
    android:fontFamily="@font/montserrat_medium"
    android:inputType="textPassword"
    android:textColor="@color/textColor"
    android:textColorHint="@color/hintTextColor" />
```

```
</com.google.android.material.textfield.TextInputLayout>
```



<CheckBox

```
    android:id="@+id/cbRememberMe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:buttonTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_medium"
    android:text="@string/remember_me"
    android:textColor="@color/textColor"
    app:layout_constraintStart_toStartOf="@+id/tiIPassword"
    app:layout_constraintTop_toBottomOf="@+id/tiIPassword" />
```

<Button

```
    android:id="@+id/btnSignIn"
    android:layout_width="290dp"
    android:layout_height="110dp"
    android:layout_marginTop="430dp"
    android:backgroundTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_semibold"
    android:text="@string/sign_in"
    android:textSize="34sp"
    app:cornerRadius="15dp"
    app:layout_constraintEnd_toEndOf="@+id/tiIPassword"
    app:layout_constraintStart_toStartOf="@+id/tiIPassword"
    app:layout_constraintTop_toTopOf="parent" />
```

<Button

```
    android:id="@+id/btnBack"
    android:layout_width="0dp"
    android:layout_height="110dp"
    android:layout_marginTop="24dp"
    android:backgroundTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_semibold"
    android:text="@string/back"
    android:textSize="34sp"
    app:cornerRadius="15dp"
    app:layout_constraintEnd_toEndOf="@+id/btnSignIn"
    app:layout_constraintStart_toStartOf="@+id/btnSignIn"
    app:layout_constraintTop_toBottomOf="@+id/btnSignIn" />
```

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES. MainActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/ivProfile"
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:focusable="true"
        android:src="@drawable/cpu_pfp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip" />

    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/ivLeaderboard"
        android:layout_width="100dp"
        android:layout_height="0dp"
```



```
android:layout_marginStart="16dp"
android:scaleType="fitXY"
app:layout_constraintBottom_toBottomOf="@+id/ivProfile"
app:layout_constraintStart_toEndOf="@+id/ivProfile"
app:layout_constraintTop_toTopOf="@+id/ivProfile"
app:layout_constraintVertical_bias="0.0"
app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"
app:srcCompat="@drawable/leaderboard" />
```

<com.google.android.material.imageview.ShapeableImageView

```
android:id="@+id/ivSettings"
android:layout_width="100dp"
android:layout_height="100dp"
android:layout_marginTop="8dp"
android:layout_marginEnd="8dp"
android:scaleType="fitXY"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"
app:srcCompat="@drawable/settings" />
```

<Button

```
android:id="@+id/btnCpu"
android:layout_width="290dp"
android:layout_height="110dp"
android:layout_marginTop="230dp"
android:backgroundTint="@color/buttonColor"
android:fontFamily="@font/montserrat_semibold"
android:text="@string/singleplayer"
android:textSize="34sp"
app:cornerRadius="15dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.503"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent" />
```

<Button

```
android:id="@+id/btnLocal"
android:layout_width="0dp"
android:layout_height="110dp"
android:layout_marginTop="52dp"
android:backgroundTint="@color/buttonColor"
android:fontFamily="@font/montserrat_semibold"
android:text="@string/local_multiplayer"
android:textSize="34sp"
app:cornerRadius="15dp"
app:layout_constraintEnd_toEndOf="@+id/btnCpu"
app:layout_constraintStart_toStartOf="@+id/btnCpu"
app:layout_constraintTop_toBottomOf="@+id/btnCpu" />
```



```
<Button
    android:id="@+id/btnOnline"
    android:layout_width="0dp"
    android:layout_height="110dp"
    android:layout_marginTop="52dp"
    android:backgroundTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_semibold"
    android:text="@string/online_multiplayer"
    android:textSize="34sp"
    app:cornerRadius="15dp"
    app:layout_constraintEnd_toEndOf="@+id/btnLocal"
    app:layout_constraintStart_toStartOf="@+id/btnLocal"
    app:layout_constraintTop_toBottomOf="@+id/btnLocal" />

</androidx.constraintlayout.widget.ConstraintLayout>
```




Activity_profile

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES.ProfileActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/ivPfp"
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:layout_marginTop="32dp"
        android:clickable="true"
        android:scaleType="fitXY"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"
        tools:src="@tools:sample/avatars" />

    <TextView
        android:id="@+id/tvUsername"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="36dp"
```



```
android:fontFamily="@font/montserrat_semibold"
android:textColor="@color/textColor"
android:textSize="20sp"
app:layout_constraintEnd_toEndOf="@+id/ivPfp"
app:layout_constraintHorizontal_bias="0.461"
app:layout_constraintStart_toStartOf="@+id/ivPfp"
app:layout_constraintTop_toBottomOf="@+id/ivPfp"
tools:text="{USERNAME}" />
```

<TextView

```
android:id="@+id/tvElo"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="32dp"
android:fontFamily="@font/montserrat_semibold"
android:textColor="@color/textColor"
android:textSize="20sp"
app:layout_constraintEnd_toEndOf="@+id/tvUsername"
app:layout_constraintStart_toStartOf="@+id/tvUsername"
app:layout_constraintTop_toBottomOf="@+id/tvUsername"
tools:text="Elo: {ELO}" />
```

<TextView

```
android:id="@+id/tvGames"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="36dp"
android:fontFamily="@font/montserrat_medium"
android:text="@string/recent_games"
android:textColor="@color/textColor"
android:textSize="20sp"
app:layout_constraintEnd_toEndOf="@+id/tvElo"
app:layout_constraintStart_toStartOf="@+id/tvElo"
app:layout_constraintTop_toBottomOf="@+id/tvElo" />
```

<androidx.recyclerview.widget.RecyclerView

```
android:id="@+id/rvGames"
android:layout_width="0dp"
android:layout_height="270dp"
android:layout_marginTop="8dp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvGames" />
```

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_register.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES. RegisterActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="30dp"
        android:fontFamily="@font/montserrat_semibold"
        android:text="@string/register"
        android:textColor="@color/textColor"
        android:textSize="50sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/tiUsername"
        android:layout_width="250dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
```



```
android:hint="@string/username"
android:textColorHint="@color/hintTextColor"
app:boxStrokeColor="@color/colorPrimary"
app:cursorColor="@color/textColor"
app:hintTextColor="@color/colorPrimary"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@+id/tvTitle">

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/etUsername"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:autofillHints="new_username"
    android:fontFamily="@font/montserrat_medium"
    android:inputType="text"
    android:textColor="@color/textColor"
    android:textColorHint="@color/hintTextColor" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tilPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="8dp"
    android:hint="@string/password"
    android:textColorHint="@color/hintTextColor"
    app:boxStrokeColor="@color/colorPrimary"
    app:cursorColor="@color/textColor"
    app:hintTextColor="@color/colorPrimary"
    app:layout_constraintEnd_toEndOf="@+id/tilUsername"
    app:layout_constraintStart_toStartOf="@+id/tilUsername"
    app:layout_constraintTop_toBottomOf="@+id/tilUsername"
    app:passwordToggleEnabled="true"
    app:passwordToggleTint="@color/colorPrimary">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/etPassword"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:autofillHints="new_password"
        android:fontFamily="@font/montserrat_medium"
        android:inputType="textPassword"
        android:textColor="@color/textColor"
        android:textColorHint="@color/hintTextColor" />

</com.google.android.material.textfield.TextInputLayout>

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/tilConfirmPassword"
```

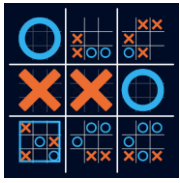


```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginTop="77dp"
android:hint="@string/confirm_password"
android:textColorHint="@color/hintTextColor"
app:boxStrokeColor="@color/colorPrimary"
app:cursorColor="@color/textColor"
app:hintTextColor="@color/colorPrimary"
app:layout_constraintEnd_toEndOf="@+id/tillPassword"
app:layout_constraintStart_toStartOf="@+id/tillPassword"
app:layout_constraintTop_toBottomOf="@+id/tillUsername"
app:passwordToggleEnabled="true"
app:passwordToggleTint="@color/colorPrimary">

<com.google.android.material.textfield.TextInputEditText
    android:id="@+id/etConfirmPassword"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:autofillHints="new_password"
    android:fontFamily="@font/montserrat_medium"
    android:inputType="textPassword"
    android:textColor="@color/textColor"
    android:textColorHint="@color/hintTextColor" />
</com.google.android.material.textfield.TextInputLayout>

<TextView
    android:id="@+id/tvError"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="6dp"
    android:background="@drawable/error_box_background"
    android:padding="4dp"
    android:text="@string/password_invalid"
    android:textColor="#ea6139"
    android:textSize="12sp"
    android:visibility="invisible"
    app:layout_constraintEnd_toEndOf="@+id/tillConfirmPassword"
    app:layout_constraintStart_toStartOf="@+id/tillConfirmPassword"
    app:layout_constraintTop_toBottomOf="@+id/tillConfirmPassword" />

<Button
    android:id="@+id/btnRegister"
    android:layout_width="290dp"
    android:layout_height="110dp"
    android:layout_marginTop="430dp"
    android:backgroundTint="@color/buttonColor"
    android:fontFamily="@font/montserrat_semibold"
    android:text="@string/register"
    android:textSize="34sp"
    app:cornerRadius="15dp"
```



```
app:layout_constraintEnd_toEndOf="@+id/tilConfirmPassword"  
app:layout_constraintStart_toStartOf="@+id/tilConfirmPassword"  
app:layout_constraintTop_toTopOf="parent" />
```

<Button

```
android:id="@+id/btnBack"  
android:layout_width="0dp"  
android:layout_height="110dp"  
android:layout_marginTop="24dp"  
android:backgroundTint="@color/buttonColor"  
android:fontFamily="@font/montserrat_semibold"  
android:text="@string/back"  
android:textSize="34sp"  
app:cornerRadius="15dp"  
app:layout_constraintEnd_toEndOf="@+id/btnRegister"  
app:layout_constraintStart_toStartOf="@+id/btnRegister"  
app:layout_constraintTop_toBottomOf="@+id/btnRegister" />
```

</androidx.constraintlayout.widget.ConstraintLayout>



Activity_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=". ACTIVITIES. SettingsActivity">

    <ImageView
        android:id="@+id/ivBackground"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:alpha="1"
        android:importantForAccessibility="no"
        android:scaleType="centerCrop"
        android:scaleX="1.3"
        android:scaleY="1.3"
        android:src="@drawable/background"
        android:tintMode="multiply"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
        app:tint="#1F2E4D"
        tools:ignore="RtlSymmetry" />

    <TextView
        android:id="@+id/tvTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="60dp"
        android:fontFamily="@font/montserrat_bold"
        android:text="@string/settings"
        android:textColor="@color/textColor"
        android:textSize="34sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnLogOut"
        android:layout_width="217.5dp"
        android:layout_height="82.5dp"
        android:layout_marginTop="36dp"
```



```
android:backgroundTint="@color/buttonColor"
android:fontFamily="@font/montserrat_semibold"
android:text="@string/log_out"
android:textSize="25.5sp"
app:cornerRadius="15dp"
app:layout_constraintEnd_toEndOf="@+id/tvTitle"
app:layout_constraintStart_toStartOf="@+id/tvTitle"
app:layout_constraintTop_toBottomOf="@+id/tvTitle" />
</androidx.constraintlayout.widget.ConstraintLayout>
```




Game_single_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardBackgroundColor="@android:color/transparent"
    app:cardCornerRadius="0dp"
    app:cardElevation="0dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">

        <com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/ivPfp"
            android:layout_width="45dp"
            android:layout_height="0dp"
            android:layout_marginStart="8dp"
            android:scaleType="fitXY"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"
            tools:srcCompat="@tools:sample/avatars" />

        <TextView
            android:id="@+id/tvUsername"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:fontFamily="@font/montserrat_medium"
            android:textColor="@color/textColor"
            android:textSize="20sp"
            app:layout_constraintBottom_toBottomOf="@+id/ivPfp"
            app:layout_constraintStart_toEndOf="@+id/ivPfp"
            app:layout_constraintTop_toTopOf="@+id/ivPfp"
            tools:text="{USERNAME}" />

        <TextView
            android:id="@+id/tvELO"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
```



```
android:fontFamily="@font/montserrat_medium"
android:textColor="@color/textColor"
android:textSize="20sp"
app:layout_constraintBottom_toBottomOf="@+id/tvUsername"
app:layout_constraintStart_toEndOf="@+id/tvUsername"
app:layout_constraintTop_toTopOf="@+id/tvUsername"
tools:text="({ELO})" />
```

<ImageView

```
android:id="@+id/ivGameResult"
android:layout_width="45dp"
android:layout_height="45dp"
android:layout_marginEnd="16dp"
android:importantForAccessibility="no"
android:scaleType="fitXY"
app:layout_constraintBottom_toBottomOf="@+id/tvElo"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="@+id/tvElo"
app:srcCompat="@drawable/ok" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>



Item_loading.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="6.3dp">

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</FrameLayout>
```



User_single_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:cardBackgroundColor="@android:color/transparent"
    app:cardCornerRadius="0dp"
    app:cardElevation="0dp">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp">

        <TextView
            android:id="@+id/tvRank"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="10dp"
            android:fontFamily="@font/montserrat_bold"
            android:textColor="@color/textColor"
            android:textSize="20sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:text="#10" />

        <com.google.android.material.imageview.ShapeableImageView
            android:id="@+id/ivPfp"
            android:layout_width="45dp"
            android:layout_height="45dp"
            android:layout_marginStart="64dp"
            android:scaleType="fitXY"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:shapeAppearanceOverlay="@style/ShapeAppearanceOverlay.Material3.Chip"
            tools:srcCompat="@tools:sample/avatars" />

        <TextView
            android:id="@+id/tvUsername"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:fontFamily="@font/montserrat_medium"
            android:textColor="@color/textColor"
```



```
android:textSize="20sp"
app:layout_constraintBottom_toBottomOf="@+id/ivPfp"
app:layout_constraintStart_toEndOf="@+id/ivPfp"
app:layout_constraintTop_toTopOf="@+id/ivPfp"
tools:text="{USERNAME}" />
```

<TextView

```
android:id="@+id/tvELO"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginEnd="8dp"
android:ems="3"
android:fontFamily="@font/montserrat_medium"
android:gravity="center_horizontal"
android:textColor="@color/textColor"
android:textSize="20sp"
app:layout_constraintBottom_toBottomOf="@+id/tvUsername"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintTop_toTopOf="@+id/tvUsername"
tools:text="{ELO}" />
```

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>



EmptyQueryException.java

```
package com.emil_z.model.exceptions;

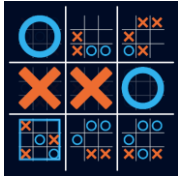
import com.google.firebase.firestore.FirebaseFirestoreException;

/**
 * Custom exception class that extends FirebaseFirestoreException.
 * This exception is used to indicate that no games are available in the queue.
 */
public class EmptyQueryException extends FirebaseFirestoreException {
    /**
     * Default constructor that initializes the exception with a default message
     * and a NOT_FOUND error code.
     */
    public EmptyQueryException() {
        this("No games in queue", Code.NOT_FOUND);
    }

    /**
     * Constructor that initializes the exception with a default message,
     * a NOT_FOUND error code, and a specified cause.
     *
     * @param cause The throwable cause of the exception.
     */
    public EmptyQueryException(Throwable cause) {
        this("No games in queue", Code.NOT_FOUND, cause);
    }

    /**
     * Constructor that initializes the exception with a custom message and error code.
     *
     * @param message The custom error message.
     * @param code The error code associated with the exception.
     */
    public EmptyQueryException(String message, Code code) {
        super(message, code);
    }

    /**
     * Constructor that initializes the exception with a custom message, error code,
     * and a specified cause.
     *
     * @param message The custom error message.
     * @param code The error code associated with the exception.
     * @param cause The throwable cause of the exception.
     */
    public EmptyQueryException(String message, Code code, Throwable cause) {
        super(message, code, cause);
    }
}
```



}



GameFullException.java

```
package com.emil_z.model.exceptions;

/**
 * Exception thrown to indicate that a game is already full and cannot accept more players.
 */
public class GameFullException extends Exception {
    /**
     * Constructs a new GameFullException with a default message.
     */
    public GameFullException() {
        super("Game is full");
    }

    /**
     * Constructs a new GameFullException with the specified detail message.
     *
     * @param message the detail message.
     */
    public GameFullException(String message) {
        super(message);
    }

    /**
     * Constructs a new GameFullException with the specified detail message and cause.
     *
     * @param message the detail message.
     * @param cause the cause of the exception.
     */
    public GameFullException(String message, Throwable cause) {
        super(message, cause);
    }

    /**
     * Constructs a new GameFullException with the specified cause.
     *
     * @param cause the cause of the exception.
     */
    public GameFullException(Throwable cause) {
        super(cause);
    }
}
```




BoardLocation.java

```
package com.emil_z.model;

import android.graphics.Point;

/**
 * Represents a location on a board, defined by two points: outer and inner.
 * Typically used to specify a position with two levels of granularity (e.g., an outer and inner
 * grid).
 */
public class BoardLocation {
    private Point outer;
    private Point inner;

    /**
     * Default constructor for BoardLocation.
     */
    public BoardLocation() {
    }

    /**
     * Constructs a BoardLocation with specified outer and inner points.
     *
     * @param outer the outer point
     * @param inner the inner point
     */
    public BoardLocation(Point outer, Point inner) {
        this.outer = outer;
        this.inner = inner;
    }

    /**
     * Constructs a BoardLocation with specified row and column values for both outer and inner
     * points.
     *
     * @param oRow the row of the outer point
     * @param oCol the column of the outer point
     * @param iRow the row of the inner point
     * @param iCol the column of the inner point
     */
    public BoardLocation(int oRow, int oCol, int iRow, int iCol) {
        {
            this.outer = new Point(oRow, oCol);
            this.inner = new Point(iRow, iCol);
        }
    }
}
```



```
/**
 * Gets the outer point.
 *
 * @return the outer point
 */
public Point getOuter() {
    return outer;
}

/**
 * Gets the inner point.
 *
 * @return the inner point
 */
public Point getInner() {
    return inner;
}

/**
 * Checks if this BoardLocation is equal to another object.
 * Two BoardLocations are equal if both their outer and inner points are equal.
 *
 * @param o the object to compare with
 * @return true if equal, false otherwise
 */
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (!(o instanceof BoardLocation)) return false;

    BoardLocation that = (BoardLocation) o;

    if (!outer.equals(that.outer)) return false;
    return inner.equals(that.inner);
}
```



Cpu.java

```
package com.emil_z.model;

import android.graphics.Point;

class MinimaxResult {
    public double mE; // minimax evaluation
    public int tP;    // tmpPlay (board/square index)

    public MinimaxResult(double mE, int tP) {
        this.mE = mE;
        this.tP = tP;
    }
}

public class Cpu {
    public static int RUNS = 0;
    public static int ai = -1;
    public static int player = 1;
    private static int moves = 0;

    /**
     * Convert OuterBoard to the 2D integer array format expected by minimax
     * @param outerBoard the current game state
     * @return 2D int array representing game state (9x9)
     */
    private static int[][] convertOuterBoardToPosition(OuterBoard outerBoard) {
        int[][] position = new int[9][9];

        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                InnerBoard innerBoard = outerBoard.getBoard(new Point(i, j));
                for (int k = 0; k < 3; k++) {
                    for (int l = 0; l < 3; l++) {
                        char cell = innerBoard.getCell(new Point(k, l));
                        int boardIndex = i * 3 + j;
                        int squareIndex = k * 3 + l;

                        if (cell == 'X') {
                            position[boardIndex][squareIndex] = player;
                        } else if (cell == 'O') {
                            position[boardIndex][squareIndex] = ai;
                        } else {
                            position[boardIndex][squareIndex] = 0;
                        }
                    }
                }
            }
        }
    }
}
```



```
        return position;
    }

    //SIMPLY CHECKS A NORMAL TIC TAC TOE BOARD, RETURNS 1 or -1 if a specific player has won,
    returns 0 if no one has won.
    private static int checkWinCondition(int[] map) {
        int a = 1;
        if (map[0] + map[1] + map[2] == a * 3 || map[3] + map[4] + map[5] == a * 3 || map[6] + map[7]
+ map[8] == a * 3 || map[0] + map[3] + map[6] == a * 3 || map[1] + map[4] + map[7] == a * 3 ||
        map[2] + map[5] + map[8] == a * 3 || map[0] + map[4] + map[8] == a * 3 || map[2] +
map[4] + map[6] == a * 3) {
            return a;
        }
        a = -1;
        if (map[0] + map[1] + map[2] == a * 3 || map[3] + map[4] + map[5] == a * 3 || map[6] + map[7]
+ map[8] == a * 3 || map[0] + map[3] + map[6] == a * 3 || map[1] + map[4] + map[7] == a * 3 ||
        map[2] + map[5] + map[8] == a * 3 || map[0] + map[4] + map[8] == a * 3 || map[2] +
map[4] + map[6] == a * 3) {
            return a;
        }
        return 0;
    }

    //The most important function, returns a numerical evaluation of the whole game in it's current
    state
    private static double evaluateGame(int[][] position, int currentBoard) {
        double evale = 0;
        int[] mainBd = new int[9];
        double[] evaluatorMul = {1.4, 1, 1.4, 1, 1.75, 1, 1.4, 1, 1.4};
        for (int eh = 0; eh < 9; eh++) {
            evale += realEvaluateSquare(position[eh])*1.5*evaluatorMul[eh];
            if(eh == currentBoard) {
                evale += realEvaluateSquare(position[eh])*evaluatorMul[eh];
            }
            int tmpEv = checkWinCondition(position[eh]);
            evale -= tmpEv*evaluatorMul[eh];
            mainBd[eh] = tmpEv;
        }
        evale -= checkWinCondition(mainBd)*5000;
        evale += realEvaluateSquare(mainBd)*150;
        return evale;
    }

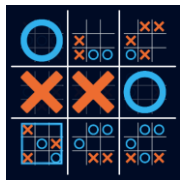
    //minimax algorithm
    private static MinimaxResult miniMax(int[][] position, int boardToPlayOn, int depth, double
alpha, double beta, boolean maximizingPlayer) {
        RUNS++;

        int tmpPlay = -1;
```



```
double calcEval = evaluateGame(position, boardToPlayOn);
if(depth <= 0 || Math.abs(calcEval) > 5000) {
    return new MinimaxResult(calcEval, tmpPlay);
}
//If the board to play on is -1, it means you can play on any board
if(boardToPlayOn != -1 && checkWinCondition(position[boardToPlayOn]) != 0) {
    boardToPlayOn = -1;
}
//If a board is full (doesn't include 0), it also sets the board to play on to -1
if(boardToPlayOn != -1) {
    boolean includesZero = false;
    for (int val : position[boardToPlayOn]) {
        if (val == 0) {
            includesZero = true;
            break;
        }
    }
    if (!includesZero) {
        boardToPlayOn = -1;
    }
}

if(maximizingPlayer) {
    double maxEval = Double.NEGATIVE_INFINITY;
    for(int mm = 0; mm < 9; mm++) {
        double evalut = Double.NEGATIVE_INFINITY;
        //If you can play on any board, you have to go through all of them
        if(boardToPlayOn == -1) {
            for(int trr = 0; trr < 9; trr++) {
                //Except the ones which are won
                if(checkWinCondition(position[mm]) == 0) {
                    if(position[mm][trr] == 0) {
                        position[mm][trr] = ai;
                        //tmpPlay = pickBoard(position, true);
                        evalut = miniMax(position, trr, depth-1, alpha, beta, false).mE;
                        //evalut+=150;
                        position[mm][trr] = 0;
                    }
                    if(evalut > maxEval) {
                        maxEval = evalut;
                        tmpPlay = mm;
                    }
                }
                alpha = Math.max(alpha, evalut);
            }
        }
        if(beta <= alpha) {
            break;
        }
    }
}
```



```

        //If there's a specific board to play on, you just go through it's squares
    }else{
        MinimaxResult evalutResult = null;
        if(position[boardToPlayOn][mm] == 0){
            position[boardToPlayOn][mm] = ai;
            evalutResult = miniMax(position, mm, depth-1, alpha, beta, false);
            position[boardToPlayOn][mm] = 0;
        }
        // Check if evalutResult is not null before accessing mE
        double blop = (evalutResult != null) ? evalutResult.mE : Double.NEGATIVE_INFINITY;
        // Handle case where no move was possible in this square
        if(blop > maxEval){
            maxEval = blop;
            //Saves which board you should play on, so that this can be passed on when the AI
            //is allowed to play in any board
            tmpPlay = evalutResult.tP;
        }
        alpha = Math.max(alpha, blop);
        if(beta <= alpha){
            break;
        }
    }
}
return new MinimaxResult(maxEval, tmpPlay);
}else{
    double minEval = Double.POSITIVE_INFINITY;
    for(int mm = 0; mm < 9; mm++){
        double evalua = Double.POSITIVE_INFINITY;
        if(boardToPlayOn == -1){
            for(int trr = 0; trr < 9; trr++){
                if(checkWinCondition(position[mm]) == 0){
                    if(position[mm][trr] == 0){
                        position[mm][trr] = player;
                        //tmpPlay = pickBoard(position, true);
                        evalua = miniMax(position, trr, depth-1, alpha, beta, true).mE;
                        //evalua -= 150;
                        position[mm][trr] = 0;
                    }
                    if(evalua < minEval){
                        minEval = evalua;
                        tmpPlay = mm;
                    }
                }
                beta = Math.min(beta, evalua);
            }
        }
        if(beta <= alpha){
            break;
        }
    }
}
}

```



```
MinimaxResult evaluaResult = null;
if(position[boardToPlayOn][mm] == 0) {
    position[boardToPlayOn][mm] = player;
    evaluaResult = miniMax(position, mm, depth-1, alpha, beta, true);
    position[boardToPlayOn][mm] = 0;
}
double blep = (evaluaResult != null) ? evaluaResult.mE : Double.POSITIVE_INFINITY;
// Handle case where no move was possible in this square
if(blep < minEval) {
    minEval = blep;
    tmpPlay = evaluaResult.tP;
}
beta = Math.min(beta, blep);
if(beta <= alpha) {
    break;
}
}
}
return new MinimaxResult(minEval, tmpPlay);
}
}

//Low number means losing the board, big number means winning
//Tbf this is less an evaluation algorithm and more something that figures out where the AI
should move to win normal Tic Tac Toe
private static double evaluatePos(int[] pos, int square) {
    int[] posCopy = new int[pos.length];
    System.arraycopy(pos, 0, posCopy, 0, pos.length);

    posCopy[square] = ai;
    double evaluation = 0;
    //Prefer center over corners over edges
    //evaluation -=
    (pos[0]*0.2+pos[1]*0.1+pos[2]*0.2+pos[3]*0.1+pos[4]*0.25+pos[5]*0.1+pos[6]*0.2+pos[7]*0.1+pos[8]*0.2
    );
    double[] points = {0.2, 0.17, 0.2, 0.17, 0.22, 0.17, 0.2, 0.17, 0.2};

    int a = 2;
    evaluation+=points[square];
    //Prefer creating pairs
    a = -2;
    if(posCopy[0] + posCopy[1] + posCopy[2] == a || posCopy[3] + posCopy[4] + posCopy[5] == a ||
    posCopy[6] + posCopy[7] + posCopy[8] == a || posCopy[0] + posCopy[3] + posCopy[6] == a || posCopy[1]
    + posCopy[4] + posCopy[7] == a ||
    posCopy[2] + posCopy[5] + posCopy[8] == a || posCopy[0] + posCopy[4] + posCopy[8] == a
    || posCopy[2] + posCopy[4] + posCopy[6] == a) {
        evaluation += 1;
    }
    //Take victories
    a = -3;
}
```



```
if(posCopy[0] + posCopy[1] + posCopy[2] == a || posCopy[3] + posCopy[4] + posCopy[5] == a ||
posCopy[6] + posCopy[7] + posCopy[8] == a || posCopy[0] + posCopy[3] + posCopy[6] == a || posCopy[1]
+ posCopy[4] + posCopy[7] == a ||
posCopy[2] + posCopy[5] + posCopy[8] == a || posCopy[0] + posCopy[4] + posCopy[8] == a
|| posCopy[2] + posCopy[4] + posCopy[6] == a) {
    evaluation += 5;
}

//Block a players turn if necessary
posCopy[square] = player;

a = 3;
if(posCopy[0] + posCopy[1] + posCopy[2] == a || posCopy[3] + posCopy[4] + posCopy[5] == a ||
posCopy[6] + posCopy[7] + posCopy[8] == a || posCopy[0] + posCopy[3] + posCopy[6] == a || posCopy[1]
+ posCopy[4] + posCopy[7] == a ||
posCopy[2] + posCopy[5] + posCopy[8] == a || posCopy[0] + posCopy[4] + posCopy[8] == a
|| posCopy[2] + posCopy[4] + posCopy[6] == a) {
    evaluation += 2;
}

posCopy[square] = ai;

evaluation -= checkWinCondition(posCopy)*15;

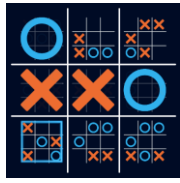
//evaluation -= checkWinCondition(pos)*4;

return evaluation;
}

//This function actually evaluates a board fairly
private static double realEvaluateSquare(int[] pos) {
    double evaluation = 0;
    double[] points = {0.2, 0.17, 0.2, 0.17, 0.22, 0.17, 0.2, 0.17, 0.2};

    for(int bw = 0; bw < pos.length; bw++){
        evaluation -= pos[bw]*points[bw];
    }

    int a = 2;
    if(pos[0] + pos[1] + pos[2] == a || pos[3] + pos[4] + pos[5] == a || pos[6] + pos[7] + pos[8]
== a) {
        evaluation -= 6;
    }
    if(pos[0] + pos[3] + pos[6] == a || pos[1] + pos[4] + pos[7] == a || pos[2] + pos[5] + pos[8]
== a) {
        evaluation -= 6;
    }
    if(pos[0] + pos[4] + pos[8] == a || pos[2] + pos[4] + pos[6] == a) {
        evaluation -= 7;
    }
}
```

```
a = -1;
if((pos[0] + pos[1] == 2*a && pos[2] == -a) || (pos[1] + pos[2] == 2*a && pos[0] == -a) ||
(pos[0] + pos[2] == 2*a && pos[1] == -a)
    || (pos[3] + pos[4] == 2*a && pos[5] == -a) || (pos[3] + pos[5] == 2*a && pos[4] == -a)
|| (pos[5] + pos[4] == 2*a && pos[3] == -a)
    || (pos[6] + pos[7] == 2*a && pos[8] == -a) || (pos[6] + pos[8] == 2*a && pos[7] == -a)
|| (pos[7] + pos[8] == 2*a && pos[6] == -a)
    || (pos[0] + pos[3] == 2*a && pos[6] == -a) || (pos[0] + pos[6] == 2*a && pos[3] == -a)
|| (pos[3] + pos[6] == 2*a && pos[0] == -a)
    || (pos[1] + pos[4] == 2*a && pos[7] == -a) || (pos[1] + pos[7] == 2*a && pos[4] == -a)
|| (pos[4] + pos[7] == 2*a && pos[1] == -a)
    || (pos[2] + pos[5] == 2*a && pos[8] == -a) || (pos[2] + pos[8] == 2*a && pos[5] == -a)
|| (pos[5] + pos[8] == 2*a && pos[2] == -a)
    || (pos[0] + pos[4] == 2*a && pos[8] == -a) || (pos[0] + pos[8] == 2*a && pos[4] == -a)
|| (pos[4] + pos[8] == 2*a && pos[0] == -a)
    || (pos[2] + pos[4] == 2*a && pos[6] == -a) || (pos[2] + pos[6] == 2*a && pos[4] == -a)
|| (pos[4] + pos[6] == 2*a && pos[2] == -a)) {
    evaluation=-9;
}

a = -2;
if(pos[0] + pos[1] + pos[2] == a || pos[3] + pos[4] + pos[5] == a || pos[6] + pos[7] + pos[8]
== a) {
    evaluation += 6;
}
if(pos[0] + pos[3] + pos[6] == a || pos[1] + pos[4] + pos[7] == a || pos[2] + pos[5] + pos[8]
== a) {
    evaluation += 6;
}
if(pos[0] + pos[4] + pos[8] == a || pos[2] + pos[4] + pos[6] == a) {
    evaluation += 7;
}

a = 1;
if((pos[0] + pos[1] == 2*a && pos[2] == -a) || (pos[1] + pos[2] == 2*a && pos[0] == -a) ||
(pos[0] + pos[2] == 2*a && pos[1] == -a)
    || (pos[3] + pos[4] == 2*a && pos[5] == -a) || (pos[3] + pos[5] == 2*a && pos[4] == -a)
|| (pos[5] + pos[4] == 2*a && pos[3] == -a)
    || (pos[6] + pos[7] == 2*a && pos[8] == -a) || (pos[6] + pos[8] == 2*a && pos[7] == -a)
|| (pos[7] + pos[8] == 2*a && pos[6] == -a)
    || (pos[0] + pos[3] == 2*a && pos[6] == -a) || (pos[0] + pos[6] == 2*a && pos[3] == -a)
|| (pos[3] + pos[6] == 2*a && pos[0] == -a)
    || (pos[1] + pos[4] == 2*a && pos[7] == -a) || (pos[1] + pos[7] == 2*a && pos[4] == -a)
|| (pos[4] + pos[7] == 2*a && pos[1] == -a)
    || (pos[2] + pos[5] == 2*a && pos[8] == -a) || (pos[2] + pos[8] == 2*a && pos[5] == -a)
|| (pos[5] + pos[8] == 2*a && pos[2] == -a)
    || (pos[0] + pos[4] == 2*a && pos[8] == -a) || (pos[0] + pos[8] == 2*a && pos[4] == -a)
|| (pos[4] + pos[8] == 2*a && pos[0] == -a)
    || (pos[2] + pos[4] == 2*a && pos[6] == -a) || (pos[2] + pos[6] == 2*a && pos[4] == -a)
```



```
|| (pos[4] + pos[6] == 2*a && pos[2] == -a)) {
    evaluation+=9;
}

evaluation -= checkWinCondition(pos)*12;

return evaluation;
}

public static BoardLocation findBestMove(OuterBoard outerBoard) {
    int bestMove = -1;
    double[] bestScore = new double[9]; // Ensure bestScore is initialized as a double array of
size 9
    int[][] boards = convertOuterBoardToPosition(outerBoard);
    int boardToPlayOn = -1; // Default to free move
    if (!outerBoard.isFreeMove()) {
        Point lastMove = outerBoard.getLastMove();
        boardToPlayOn = lastMove.x * 3 + lastMove.y;
    }

    for (int i = 0; i < 9; i++) {
        bestScore[i] = Double.NEGATIVE_INFINITY;
    }

    RUNS = 0;
    //Calculates the remaining amount of empty squares
    int count = 0;
    for (int bt = 0; bt < boards.length; bt++) {
        if (checkWinCondition(boards[bt]) == 0) {
            for (int v : boards[bt]) {
                if (v == 0) {
                    count++;
                }
            }
        }
    }

    if (boardToPlayOn == -1 || checkWinCondition(boards[boardToPlayOn]) != 0) {
        MinimaxResult savedMm;
        System.out.println("Remaining: " + count);

        //This minimax doesn't actually play a move, it simply figures out which board you should
play on
        if (moves < 10) {
            // Use Math.min for depth, Double.NEGATIVE_INFINITY and Double.POSITIVE_INFINITY for
alpha/beta
            savedMm = miniMax(boards, -1, Math.min(4, count), Double.NEGATIVE_INFINITY,
```



```
Double.POSITIVE_INFINITY, true); //Putting math.min makes sure that minimax doesn't run when the
board is full
    } else if (moves < 18) {
        savedMm = miniMax(boards, -1, Math.min(5, count), Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, true);
    } else {
        savedMm = miniMax(boards, -1, Math.min(6, count), Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, true);
    }
    System.out.println(savedMm.tP);
    boardToPlayOn = savedMm.tP;
}

//Just makes a quick default move for if all else fails
for (int i = 0; i < 9; i++) {
    if (boards[boardToPlayOn][i] == 0) {
        bestMove = i;
        break;
    }
}

if (bestMove != -1) { //This condition should only be false if the board is full, but it's
here in case

    //Best score is an array which contains individual scores for each square, here we're just
changing them based on how good the move is on that one local board
    for (int a = 0; a < 9; a++) {
        if (boards[boardToPlayOn][a] == 0) {
            double score = evaluatePos(boards[boardToPlayOn], a) * 45; // Use double for score
            bestScore[a] = score;
        }
    }

    //And here we actually run minimax and add those values to the array
    if (checkWinCondition(boards[boardToPlayOn]) == 0) { // Check if the current board is
still playable
        for (int b = 0; b < 9; b++) {
            if (boards[boardToPlayOn][b] == 0) {
                boards[boardToPlayOn][b] = ai; // Make the move temporarily
                MinimaxResult savedMm; // Use the custom class
                //Notice the stacking, at the beginning of the game, the depth is much lower than
at the end
                if (moves < 20) {
                    savedMm = miniMax(boards, b, Math.min(5, count), Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, false);
                } else if (moves < 32) {
                    System.out.println("DEEP SEARCH");
                    savedMm = miniMax(boards, b, Math.min(6, count), Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, false);
                }
            }
        }
    }
}
```



```
    } else {
        System.out.println("ULTRA DEEP SEARCH");
        savedMm = miniMax(boards, b, Math.min(7, count), Double.NEGATIVE_INFINITY,
Double.POSITIVE_INFINITY, false);
    }
    double score2 = savedMm.mE;
    boards[boardToPlayOn][b] = 0; // Undo the move
    bestScore[b] += score2;
    //boardSel[b] = savedMm.tP;
    //console.log(score2);
}
}
}

//Chooses to play on the square with the highest evaluation in the bestScore array
// Translate for...in loop iterating over indices to a standard for loop
// Find the index of the maximum value in bestScore
bestMove = 0;
for (int i = 1; i < bestScore.length; i++) { // Iterate from the second element
    if (bestScore[i] > bestScore[bestMove]) {
        bestMove = i; // Update bestMove if a higher score is found
    }
}
moves += 1; // Increment moves by 2 for the AI's turn
return new BoardLocation(new Point(boardToPlayOn / 3, boardToPlayOn % 3), new
Point(bestMove / 3, bestMove % 3)); // Return the best move as a BoardLocation object
}
return null;
}
}
```



CpuGame.java

```
package com.emil_z.model;

import java.util.Objects;

/**
 * Represents a game where a human player competes against a CPU opponent.
 * Extends the {@link Game} class and manages player initialization and move logic.
 */
public class CpuGame extends Game {
    /**
     * Constructs a new CpuGame with the specified player IDs.
     *
     * @param localPlayerIdsFs the ID for the local (human) player
     * @param crossPlayerIdsFs the ID for the cross (CPU) player
     */
    public CpuGame(String localPlayerIdsFs, String crossPlayerIdsFs) {
        super();
        setStarted(true);
        player1.setName("Player 1");
        player1.setIdsFs(localPlayerIdsFs);
        player2.setName("CPU");
        player2.setIdsFs("CPU");
        this.crossPlayerIdsFs = this.currentPlayerIdsFs = crossPlayerIdsFs;
    }

    /**
     * Makes a move at the specified board location, updates the current player,
     * and checks for game completion.
     *
     * @param location the {@link BoardLocation} where the move is made
     */
    public void makeMove(BoardLocation location) {
        super.makeMove(location);
        outerBoard.getBoard(location.getOuter()).isFinished();
        currentPlayerIdsFs = Objects.equals(currentPlayerIdsFs, player1.getIdFs()) ?
            player2.getIdFs() :
            player1.getIdFs();
        if (outerBoard.isGameOver()) {
            isFinished = true;
            winnerIdsFs = String.valueOf(outerBoard.getWinner());
        }
    }
}
```



Game.java

```
package com.emil_z.model;

import com.emil_z.model.BASE.BaseEntity;
import com.google.firebase.firestore.Exclude;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

/**
 * Represents a game session, managing players, moves, and board state.
 * Extends {@link BaseEntity} and implements {@link Serializable}.
 */
public class Game extends BaseEntity implements Serializable {
    protected Player player1;
    protected Player player2;
    protected String crossPlayerIds;
    protected String currentPlayerIds;
    protected String winnerIds;
    protected boolean isStarted;
    protected boolean isFinished;
    protected List<BoardLocation> moves;
    @Exclude
    protected OuterBoard outerBoard;

    /**
     * Default constructor. Initializes players, board, and state.
     */
    public Game() {
        this.player1 = new Player();
        this.player2 = new Player();
        this.winnerIds = null;
        this.isStarted = false;
        this.isFinished = false;
        this.moves = new ArrayList<>();
        this.outerBoard = new OuterBoard();
    }

    /**
     * Copy constructor. Creates a new Game instance by copying another.
     * @param game the Game instance to copy
     */
    public Game(Game game) {
        this.player1 = new Player(game.player1);
        this.player2 = new Player(game.player2);
        this.crossPlayerIds = game.crossPlayerIds;
        this.currentPlayerIds = game.currentPlayerIds;
        this.winnerIds = game.winnerIds;
    }
}
```



```
this.isStarted = game.isStarted;
this.isFinished = game.isFinished;
this.moves = new ArrayList<>(game.moves);
this.outerBoard = new OuterBoard();
}

/**
 * Gets the first player.
 * @return player1
 */
public Player getPlayer1() {
    return player1;
}

/**
 * Sets the first player.
 * @param player1 the player to set
 */
public void setPlayer1(Player player1) {
    this.player1 = player1;
}

/**
 * Gets the second player.
 * @return player2
 */
public Player getPlayer2() {
    return player2;
}

/**
 * Gets the Firestore ID of the cross player.
 * @return crossPlayerIdFs
 */
public String getCrossPlayerIdFs() {
    return crossPlayerIdFs;
}

/**
 * Gets the Firestore ID of the current player.
 * @return currentPlayerIdFs
 */
public String getCurrentPlayerIdFs() {
    return currentPlayerIdFs;
}

/**
 * Gets the Firestore ID of the winner.
 * @return winnerIdFs
 */
```



```
    */
    public String getWinnerIds() {
        return winnerIds;
    }

    /**
     * Sets the Firestore ID of the winner.
     * @param winnerIds the ID to set
     */
    public void setWinnerIds(String winnerIds) {
        this.winnerIds = winnerIds;
    }

    /**
     * Checks if the game has started.
     * @return true if started, false otherwise
     */
    public boolean isStarted() {
        return isStarted;
    }

    /**
     * Sets the started state of the game.
     * @param started true if started, false otherwise
     */
    public void setStarted(boolean started) {
        isStarted = started;
    }

    /**
     * Checks if the game has finished.
     * @return true if finished, false otherwise
     */
    public boolean isFinished() {
        return isFinished;
    }

    /**
     * Sets the finished state of the game.
     * @param finished true if finished, false otherwise
     */
    public void setFinished(boolean finished) {
        isFinished = finished;
    }

    /**
     * Gets the list of moves made in the game.
     * @return list of moves
     */
    public List<BoardLocation> getMoves() {
```

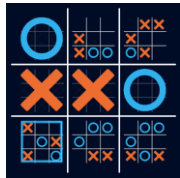



```
        return moves;
    }

    /**
     * Gets the outer board representing the game state.
     * Excluded from Firestore serialization.
     * @return outerBoard
     */
    @Exclude
    public OuterBoard getOuterBoard() {
        return outerBoard;
    }

    /**
     * Checks if a move at the given location is legal.
     * @param location the board location to check
     * @return true if the move is legal, false otherwise
     */
    public boolean isLegal(BoardLocation location) {
        return outerBoard.isLegal(location);
    }

    /**
     * Makes a move at the specified location and records it.
     * @param location the board location where the move is made
     */
    public void makeMove(BoardLocation location) {
        outerBoard.makeMove(location);
        moves.add(location);
    }
}
```



Games.java

```
package com.emil_z.model;

import com.emil_z.model.BASE.BaseList;

/**
 * Represents a list of {@link Game} objects.
 * Extends {@link BaseList} to provide list operations specific to games.
 */
public class Games extends BaseList<Game, Games> {
}
```



GameType.java

```
package com.emil_z.model;

/**
 * Enum representing the different types of games available.
 */
public enum GameType {
    CPU,
    LOCAL,
    ONLINE,
    REPLAY
}
```



InnerBoard.java

```
package com.emil_z.model;

import android.graphics.Point;

import com.emil_z.model.BASE.BaseEntity;

import java.io.Serializable;

/**
 * Represents a 3x3 inner board for the game .
 * Handles board state, winner detection, and move legality.
 */
public class InnerBoard extends BaseEntity implements Serializable {
    private final char[][] board;
    private char winner;
    private boolean isFinished;

    /**
     * Constructs a new, empty InnerBoard.
     */
    public InnerBoard() {
        board = new char[3][3];
        isFinished = false;
        winner = 0;
    }

    /**
     * Copy constructor. Creates a deep copy of the given InnerBoard.
     * @param innerBoard The InnerBoard to copy.
     */
    public InnerBoard(InnerBoard innerBoard) {
        board = new char[3][3];
        isFinished = innerBoard.isFinished;
        winner = innerBoard.winner;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = innerBoard.getCell(new Point(i, j));
            }
        }
    }

    /**
     * Gets the winner of the board.
     * @return The winner character ('X', 'O', 'T' for tie, or 0 if none).
     */
    public char getWinner() {
        return winner;
    }
}
```



```
/**
 * Checks if the board is finished (win or tie).
 * Updates the winner and finished state if necessary.
 * @return true if the board is finished, false otherwise.
 */
public boolean isFinished() {
    if (isFinished) return true;
    for (int i = 0; i < 3; i++) {
        if (board[i][0] != 0 && board[i][0] == board[i][1] && board[i][0] == board[i][2]) {
            winner = board[i][0];
            isFinished = true;
            return true;
        }
        if (board[0][i] != 0 && board[0][i] == board[1][i] && board[0][i] == board[2][i]) {
            winner = board[0][i];
            isFinished = true;
            return true;
        }
    }
    if (board[0][0] != 0 && board[0][0] == board[1][1] && board[0][0] == board[2][2] ||
        board[0][2] != 0 && board[0][2] == board[1][1] && board[0][2] == board[2][0]) {
        winner = board[1][1];
        isFinished = true;
        return true;
    }
    if (isTie()) {
        isFinished = true;
        winner = 'T';
        return true;
    }
    return false;
}

/**
 * Gets the value of a cell at the given point.
 * @param inner The point (x, y) to get the cell from.
 * @return The character in the cell, or 0 if empty.
 */
public char getCell(Point inner) {
    return board[inner.x][inner.y];
}

/**
 * Checks if the board is a tie (all cells filled, no winner).
 * @return true if the board is a tie, false otherwise.
 */
public boolean isTie() {
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
```



```
        if (board[row][col] == 0) {
            return false;
        }
    }
}
return true;
}

/**
 * Checks if a move at the given point is legal (cell empty and board not finished).
 * @param inner The point (x, y) to check.
 * @return true if the move is legal, false otherwise.
 */
public boolean isLegal(Point inner) {
    return board[inner.x][inner.y] == 0 && !isFinished;
}

/**
 * Makes a move for the current player at the given point.
 * @param inner The point (x, y) to place the move.
 * @param currentPlayer The character representing the current player.
 */
public void makeMove(Point inner, char currentPlayer) {
    board[inner.x][inner.y] = currentPlayer;
}
}
```



LocalGame.java

```
package com.emil_z.model;

/**
 * Represents a local game instance, extending the base Game class.
 * Handles initialization and move logic for a local two-player game.
 */
public class LocalGame extends Game {
    /**
     * Constructs a new LocalGame with the given local player ID.
     * Both players are assigned the same ID and default names.
     *
     * @param localPlayerIdFs The ID to assign to both players.
     */
    public LocalGame(String localPlayerIdFs) {
        super();
        setStarted(true);
        player1.setName("Player 1");
        player2.setName("Player 2");
        player1.setIdFs(localPlayerIdFs);
        player2.setIdFs(localPlayerIdFs);
        currentPlayerIdFs = crossPlayerIdFs = player1.getIdFs();
    }

    /**
     * Makes a move at the specified board location.
     * Updates the game state and checks for game over conditions.
     *
     * @param location The location on the board where the move is made.
     */
    public void makeMove(BoardLocation location) {
        super.makeMove(location);
        outerBoard.getBoard(location.getOuter()).isFinished();
        if (outerBoard.isGameOver()) {
            isFinished = true;
            winnerIdFs = String.valueOf(outerBoard.getWinner());
        }
    }
}
```



OnlineGame.java

```
package com.emil_z.model;

import com.google.firebase.Timestamp;

import java.util.Objects;

/**
 * Represents an online game session, extending the base Game class.
 * Handles player management, game start, and move logic for online play.
 */
public class OnlineGame extends Game {
    private Timestamp startedAt;

    /**
     * Default constructor for OnlineGame.
     * Initializes the base Game.
     */
    public OnlineGame() {
        super();
    }

    /**
     * Constructs an OnlineGame with the specified player as player1.
     * @param player1 The first player in the game.
     */
    public OnlineGame(Player player1) {
        super();
        this.player1 = player1;
    }

    /**
     * Constructs an OnlineGame from an existing Game instance.
     * @param game The Game instance to copy or reference.
     */
    public OnlineGame(Game game) {
    }

    /**
     * Gets the timestamp when the game started.
     * @return The start timestamp.
     */
    public Timestamp getStartedAt() {
        return startedAt;
    }

    /**
     * Sets the timestamp for when the game started.
     * @param startedAt The start timestamp to set.
     */
}
```




```
*/  
public void setStartedAt(Timestamp startedAt) {  
    this.startedAt = startedAt;  
}  
  
/**  
 * Initializes the outer board for the joiner when the game starts.  
 */  
public void startGameForJoiner() {  
    outerBoard = new OuterBoard();  
}  
  
/**  
 * Makes a move at the specified board location.  
 * Switches the current player and checks for game over conditions.  
 * @param location The location on the board where the move is made.  
 */  
public void makeMove(BoardLocation location) {  
    super.makeMove(location);  
    outerBoard.getBoard(location.getOuter()).isFinished();  
    currentPlayerIds = Objects.equals(currentPlayerIds, player1.getIdFs()) ?  
        player2.getIdFs() :  
        player1.getIdFs();  
    if (outerBoard.isGameOver()) {  
        isFinished = true;  
        winnerIds = outerBoard.getWinner() == 'T' ? "T" :  
            Objects.equals(currentPlayerIds, player2.getIdFs()) ? player1.getIdFs() :  
            player2.getIdFs();  
    }  
}  
}
```



OuterBoard.java

```
package com.emil_z.model;

import android.graphics.Point;

/**
 * Represents the outer board in an ultimate tic-tac-toe game.
 * Manages the state of the game, including the current player, winner,
 * free move status, and the last move made.
 */
public class OuterBoard {
    private final InnerBoard[] [] board;
    private char currentPlayer;
    private char winner;
    private boolean freeMove;
    private Point lastMove;

    /**
     * Constructs a new OuterBoard, initializing all inner boards and setting the starting player.
     */
    public OuterBoard() {
        board = new InnerBoard[3][3];
        winner = 0;
        currentPlayer = 'X';
        freeMove = true;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = new InnerBoard();
            }
        }
    }

    /**
     * Returns the inner board at the specified outer board location.
     * @param outer The coordinates of the inner board.
     * @return The InnerBoard at the given location.
     */
    public InnerBoard getBoard(Point outer) {
        return board[outer.x][outer.y];
    }

    /**
     * Gets the current player ('X' or 'O').
     * @return The current player.
     */
    public char getCurrentPlayer() {
        return currentPlayer;
    }
}
```



```
/**
 * Gets the winner of the game.
 * @return The winner ('X', 'O', 'T' for tie, or 0 if no winner yet).
 */
public char getWinner() {
    return winner;
}

/**
 * Checks if the next move can be made on any board.
 * @return True if a free move is allowed, false otherwise.
 */
public boolean isFreeMove() {
    return freeMove;
}

/**
 * Gets the last move made.
 * @return The last move as a Point.
 */
public Point getLastMove() {
    return lastMove;
}

/**
 * Checks if the game is a tie (all inner boards are finished).
 * @return True if the game is a tie, false otherwise.
 */
public boolean isTie() {
    for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 3; col++) {
            if (!board[row][col].isFinished()) {
                return false;
            }
        }
    }
    return true;
}

/**
 * Checks if the game is over (win or tie).
 * Updates the winner if the game is over.
 * @return True if the game is over, false otherwise.
 */
public boolean isGameOver() {
    for (int i = 0; i < 3; i++) {
        if (board[i][0].getWinner() != 0 && board[i][0].getWinner() == board[i][1].getWinner() &&
board[i][0].getWinner() == board[i][2].getWinner()) {
            winner = board[i][0].getWinner();
            return true;
        }
    }
}
```



```
    }
    if (board[0][i].getWinner() != 0 && board[0][i].getWinner() == board[1][i].getWinner() &&
board[0][i].getWinner() == board[2][i].getWinner()) {
        winner = board[0][i].getWinner();
        return true;
    }
}
if (board[0][0].getWinner() != 0 && board[0][0].getWinner() == board[1][1].getWinner() &&
board[0][0].getWinner() == board[2][2].getWinner() ||
    board[0][2].getWinner() != 0 && board[0][2].getWinner() == board[1][1].getWinner() &&
board[0][2].getWinner() == board[2][0].getWinner()) {
    winner = board[1][1].getWinner();
    return true;
}
if (isTie()) {
    winner = 'T';
    return true;
}
return false;
}

/**
 * Checks if a move at the given location is legal.
 * @param location The board location to check.
 * @return True if the move is legal, false otherwise.
 */
public boolean isLegal(BoardLocation location) {
    InnerBoard innerBoard = board[location.getOuter().x][location.getOuter().y];
    return freeMove
        ? innerBoard.isLegal(location.getInner())
        : location.getOuter().equals(lastMove) && innerBoard.isLegal(location.getInner());
}

/**
 * Makes a move at the specified location, updates the game state, and switches the current
player.
 * @param location The location to make the move.
 */
public void makeMove(BoardLocation location) {
    board[location.getOuter().x][location.getOuter().y].makeMove(location.getInner(),
currentPlayer);
    lastMove = location.getInner();
    freeMove = board[location.getInner().x][location.getInner().y].isFinished();
    currentPlayer = currentPlayer == 'X' ? 'O' : 'X';
}
}
```



Player.java

```
package com.emil_z.model;

import android.graphics.Bitmap;

import com.emil_z.helper.BitMapHelper;
import com.emil_z.model.BASE.BaseEntity;
import com.google.firebase.firestore.Exclude;

/**
 * Represents a player in the application, including their name, ELO rating, and profile picture.
 * Extends {@link BaseEntity} for common entity properties.
 */
public class Player extends BaseEntity {
    private String name;
    private float elo;
    @Exclude
    private String picture;

    /**
     * Default constructor for Player.
     */
    public Player() {
    }

    /**
     * Constructs a Player from a User object.
     * @param user The User to convert to a Player.
     */
    public Player(User user) {
        this.idFs = user.getIdFs();
        this.name = user.getUsername();
        this.elo = user.getElo();
        this.picture = user.getPicture();
    }

    /**
     * Copy constructor. Creates a new Player by copying another Player's properties.
     * @param player The Player to copy.
     */
    public Player(Player player) {
        this.idFs = player.idFs;
        this.name = player.name;
        this.elo = player.elo;
        this.picture = player.picture;
    }

    /**
     * Gets the player's name.
     */
}
```



```
* @return The player's name.
*/
public String getName() {
    return name;
}

/**
 * Sets the player's name.
 * @param name The new name for the player.
 */
public void setName(String name) {
    this.name = name;
}

/**
 * Gets the player's ELO rating.
 * @return The player's ELO rating.
 */
public float getElo() {
    return elo;
}

/**
 * Sets the player's profile picture as a Base64-encoded string.
 * Excluded from Firestore serialization.
 * @param picture The Base64-encoded profile picture.
 */
@Exclude
public void setPicture(String picture) {
    this.picture = picture;
}

/**
 * Decodes and returns the player's profile picture as a Bitmap.
 * Excluded from Firestore serialization.
 * @return The profile picture as a Bitmap, or null if decoding fails.
 */
@Exclude
public Bitmap getPictureBitmap() {
    return BitmapHelper.decodeBase64(picture);
}

/**
 * Compares this player's ELO rating to another player's ELO rating.
 * @param player The other player to compare to.
 * @return A negative integer, zero, or a positive integer as this player's ELO
 *         is less than, equal to, or greater than the specified player's ELO.
 */
public int compareElo(Player player) {
    return Float.compare(this.elo, player.elo);
}
```



```
}  
}
```

User.java

```
package com.emil_z.model;  
  
import android.graphics.Bitmap;  
  
import com.emil_z.helper.BitMapHelper;  
import com.emil_z.model.BASE.BaseEntity;  
import com.google.firebase.firestore.Exclude;  
  
import java.io.Serializable;  
  
/**  
 * Represents a user in the application, including username, hashed password,  
 * profile picture, and ELO rating.  
 * Extends {@link BaseEntity} and implements {@link Serializable}.  
 */  
public class User extends BaseEntity implements Serializable {  
    private String username;  
    private String hashedPassword;  
    private String picture;  
    private float elo;  
  
    /**  
     * Default constructor for User.  
     */  
    public User() {  
    }  
  
    /**  
     * Constructs a User with the specified username, password, and picture.  
     * Sets the default ELO rating to 1200.  
     * @param username The user's username.  
     * @param password The user's hashed password.  
     * @param picture The user's profile picture as a Base64-encoded string.  
     */  
    public User(String username, String password, String picture) {  
        this(username, password, 1200, picture);  
    }  
  
    /**  
     * Constructs a User with the specified username, password, ELO rating, and picture.  
     * @param username The user's username.  
     * @param password The user's hashed password.  
     * @param elo The user's ELO rating.  
     * @param picture The user's profile picture as a Base64-encoded string.  
     */  
}
```



```
public User(String username, String password, float elo, String picture) {
    this.username = username;
    this.password = password;
    this.elo = elo;
    this.picture = picture;
}

/**
 * Gets the user's username.
 * @return The username.
 */
public String getUsername() {
    return username;
}

/**
 * Gets the user's hashed password.
 * @return The hashed password.
 */
public String getHashedPassword() {
    return hashedPassword;
}

/**
 * Gets the user's profile picture as a Base64-encoded string.
 * @return The profile picture.
 */
public String getPicture() {
    return picture;
}

/**
 * Sets the user's profile picture as a Base64-encoded string.
 * @param picture The new profile picture.
 */
public void setPicture(String picture) {
    this.picture = picture;
}

/**
 * Gets the user's ELO rating.
 * @return The ELO rating.
 */
public float getElo() {
    return elo;
}

/**
 * Decodes and returns the user's profile picture as a Bitmap.
 * Excluded from Firestore serialization.
 */
```




```
    * @return The profile picture as a Bitmap, or null if decoding fails.
    */
    @Exclude
    public Bitmap getPictureBitmap() {
        return BitmapHelper.decodeBase64 (picture);
    }
}
```

Users.java

```
package com.emil_z.model;

import com.emil_z.model.BASE.BaseList;

/**
 * Represents a list of User objects.
 * Extends {@link BaseList} with User as the element type and Users as the list type.
 */
public class Users extends BaseList<User, Users> {
}
```



AppMonitorService.java

```
package com.emil_z.ultimate_tic_tac_toe.SERVICES;

import android.app.Application;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;

import androidx.core.app.NotificationCompat;

import com.emil_z.repository.OnlineGamesRepository;

/**
 * Service to monitor the app's game state and handle foreground notifications
 * for ongoing games. Manages game state transitions and ensures proper cleanup
 * when the app is removed from recent tasks.
 */
public class AppMonitorService extends Service {
    private static final String EXTRA_IN_GAME = "in_game_activity";
    private static final String EXTRA_GAME_ID_FS = "game_id_fs";
    private static final String EXTRA_PLAYER1_ID_FS = "player1_id_fs";
    private static final String EXTRA_PLAYER2_ID_FS = "player2_id_fs";
    private static final String EXTRA_IS_PLAYER1 = "is_player1";

    private static boolean inGameActivity = false;
    private static String gameIdFs;
    private static String player1IdFs;
    private static String player2IdFs;
    private static boolean isPlayer1;

    private OnlineGamesRepository repository;

    /**
     * Starts the AppMonitorService as a foreground service with the provided game state.
     *
     * @param context    The context to use for starting the service.
     * @param inGame     Whether the user is currently in a game.
     * @param gameIdFs   The IdFs of the current game.
     * @param player1IdFs The IdFs of player 1.
     * @param player2IdFs The IdFs of player 2.
     * @param isPlayer1  Whether the current user is player 1.
     */
    public static void startService(Context context, boolean inGame, String gameIdFs,
                                   String player1IdFs, String player2IdFs, boolean isPlayer1) {
        Intent intent = new Intent(context, AppMonitorService.class)
```



```
.putExtra(EXTRA_IN_GAME, inGame)
.putExtra(EXTRA_PLAYER1_ID_FS, player1IdFs)
.putExtra(EXTRA_PLAYER2_ID_FS, player2IdFs)
.putExtra(EXTRA_GAME_ID_FS, gameIdFs)
.putExtra(EXTRA_IS_PLAYER1, isPlayer1);
context.startForegroundService(intent);
}

/**
 * Updates the static game state fields.
 *
 * @param inGame      Whether the user is currently in a game.
 * @param gameIdFs    The IdFs of the current game.
 * @param player1IdFs The IdFs of player 1.
 * @param player2IdFs The IdFs of player 2.
 * @param isPlayer1   Whether the current user is player 1.
 */
public static void updateGameState(boolean inGame, String gameIdFs,
                                   String player1IdFs, String player2IdFs, boolean isPlayer1) {
    inGameActivity = inGame;
    AppMonitorService.gameIdFs = gameIdFs;
    AppMonitorService.player1IdFs = player1IdFs;
    AppMonitorService.player2IdFs = player2IdFs;
    AppMonitorService.isPlayer1 = isPlayer1;
}

/**
 * Notifies the service that the user has closed the game activity.
 * Updates the game state to reflect that the user is no longer in a game.
 *
 * @param context The context to use for starting the service.
 */
public static void userClosedActivity(Context context) {
    Intent intent = new Intent(context, AppMonitorService.class)
        .putExtra(EXTRA_IN_GAME, false)
        .putExtra(EXTRA_GAME_ID_FS, gameIdFs)
        .putExtra(EXTRA_PLAYER1_ID_FS, player1IdFs)
        .putExtra(EXTRA_PLAYER2_ID_FS, player2IdFs)
        .putExtra(EXTRA_IS_PLAYER1, isPlayer1);
    context.startService(intent);
}

/**
 * Initializes the repository when the service is created.
 */
@Override
public void onCreate() {
    super.onCreate();
    repository = new OnlineGamesRepository((Application) getApplicationContext());
}
```



```
/**
 * Not used, as this is not a bound service.
 *
 * @param intent The intent that was used to bind to this service.
 * @return Always returns null.
 */
@Override
public IBinder onBind(Intent intent) {
    return null;
}

/**
 * Handles the start command for the service, updates the game state from the intent,
 * and starts the service in the foreground with a notification.
 *
 * @param intent The intent supplied to startService(Intent).
 * @param flags Additional data about this start request.
 * @param startId A unique integer representing this specific request to start.
 * @return The mode in which to continue running.
 */
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    if (intent != null) {
        inGameActivity = intent.getBooleanExtra(EXTRA_IN_GAME, false);
        gameIdFs = intent.getStringExtra(EXTRA_GAME_ID_FS);
        player1IdFs = intent.getStringExtra(EXTRA_PLAYER1_ID_FS);
        player2IdFs = intent.getStringExtra(EXTRA_PLAYER2_ID_FS);
        isPlayer1 = intent.getBooleanExtra(EXTRA_IS_PLAYER1, false);
    }
    startForeground(1001, createNotification());
    return START_STICKY;
}

/**
 * Called when the app is removed from recent tasks.
 * If a game is in progress, attempts to exit the game directly.
 *
 * @param rootIntent The intent that was used to remove the task.
 */
@Override
public void onTaskRemoved(Intent rootIntent) {
    super.onTaskRemoved(rootIntent);
    if (inGameActivity && gameIdFs != null) {
        repository.exitGameDirect(gameIdFs, player1IdFs, player2IdFs, isPlayer1)
            .addOnSuccessListener(a -> {
            })
            .addOnFailureListener(e -> {
            });
    }
}
```



```
stopSelf();
}

/**
 * Creates and returns a notification for the foreground service.
 *
 * @return The notification to display.
 */
private Notification createNotification() {
    NotificationManager manager = getSystemService(NotificationManager.class);
    manager.createNotificationChannel(new NotificationChannel(
        "app_monitor_channel", "App Monitor", NotificationManager.IMPORTANCE_LOW));

    return new NotificationCompat.Builder(this, "app_monitor_channel")
        .setContentTitle("Game in progress")
        .setSmallIcon(android.R.drawable.ic_dialog_info)
        .setPriority(NotificationCompat.PRIORITY_LOW)
        .build();
}
}
```



Main.py (Cloud Functions)

```
import random

# The Cloud Functions for Firebase SDK to create Cloud Functions and set up triggers.
from http import HTTPStatus
import json
from firebase_functions import https_fn, firestore_fn

# The Firebase Admin SDK to access Cloud Firestore.
from firebase_admin import initialize_app, firestore
import google.cloud.firestore
from google.cloud.firestore_v1.transaction import transactional, Transaction
from google.cloud.firestore_v1.document import DocumentReference

app = initialize_app()

@https_fn.on_call()
def join_game(req: https_fn.CallableRequest) -> https_fn.Response:
    game_id = req.data["game_id"]
    player = json.loads(req.data["player"])
    firestore_client: google.cloud.firestore.Client = firestore.client()

    transaction = firestore_client.transaction()
    game_doc = firestore_client.document("Games", game_id)

    @transactional
    def update_game(transaction: Transaction,
                    game_doc: DocumentReference,
                    player: dict[str, str | int]) -> bool:
        game_snapshot = game_doc.get(transaction=transaction)
        #logger.log(msg=game_snapshot.to_dict().get("player2").get("name") is None,
        level=logging.INFO)
        if game_snapshot.exists and game_snapshot.to_dict().get("player2").get("name") is None:
            first_id_fs = [game_snapshot.to_dict().get("player1").get("idFs"), player["idFs"]]
            first_id_fs = random.choice(first_id_fs)
            transaction.update(game_doc, {"player2": {"idFs": player["idFs"], "name":
player["name"], "elo": int(player["elo"])}, "currentPlayerIdFs": first_id_fs, "crossPlayerIdFs":
first_id_fs})
            return True
        return False

    if update_game(transaction, game_doc, player):
        return {"status": HTTPStatus.OK}
    else:
        raise https_fn.HttpsError(
            code=https_fn.FunctionsErrorCode.ABORTED,
            message="Game already started or player 2 already joined",
        )
```



```
@https_fn.on_call()
def finish_game(req: https_fn.CallableRequest) -> https_fn.Response:
    player_1_id = req.data["player_1_id"]
    player_2_id = req.data["player_2_id"]
    score = req.data["score"]
    firestore_client: google.cloud.firestore.Client = firestore.client()

    transaction = firestore_client.transaction()
    player_1_doc = firestore_client.document("Users", player_1_id)
    player_2_doc = firestore_client.document("Users", player_2_id)

    @transactional
    def update_user(transaction: Transaction,
                    player_1_doc: DocumentReference,
                    player_2_doc: DocumentReference) -> bool:
        player_1_snapshot = player_1_doc.get(transaction=transaction)
        player_2_snapshot = player_2_doc.get(transaction=transaction)
        if player_1_snapshot.exists and player_2_snapshot.exists:
            elo_change = calculate_elo(player_1_snapshot.to_dict().get("elo"),
            player_2_snapshot.to_dict().get("elo"), score)
            transaction.update(player_1_doc, {"elo": player_1_snapshot.to_dict().get("elo") +
            elo_change})
            transaction.update(player_2_doc, {"elo": player_2_snapshot.to_dict().get("elo") -
            elo_change})
            return True
        return False
    if update_user(transaction, player_1_doc, player_2_doc):
        return {"status": HTTPStatus.OK}
    else:
        raise https_fn.HttpsError(
            code=https_fn.FunctionsErrorCode.ABORTED,
            message="Someone intervened",
        )

def calculate_elo(player_1_elo: float, player_2_elo: float, score: int) -> float:
    difference = player_2_elo - player_1_elo
    expected = 1 / (10**((difference) / 400) + 1)
    k_factor = 20
    return float(k_factor * (score - expected))
```