פרויקט גמר בהנדסת תוכנה

חלופה – תכנון ותכנות מערכות בטלפונים ניידים תחת מערכת הפעלה Android

שם בית הספר – אורט בוים רוגוזין

שם האפליקציה – משחק דמקה אונליין

שם התלמיד – טל שמחייב

ת.ז התלמיד – 325820876

שם המורה/מנחה – טל פפרמן

יוני 2022 תש״פב

תוכן עניינים

מבוא

ספר זה יכיל את תיאור הפרויקט הגמר שבחרתי כ5 יחידות נוספות במדעי המחשב.
הספר מתאר את הפן הטכני של הפרויקט בנוסף עם הפן הידידותי למשתמש.

הנושא הנבחר:

האפליקציה נועדה עבור אנשים שירצו לשחק דמקה בזמנם הפנוי מול שחקנים אחרים שיפגשו באמצעות האפליקציה.

הרקע לאפליקציה הגיע ממחקר קצר באינטרנט: דמקה הוא משחק נפוץ מאוד שדורש תחכום, שימוש רב בטקטיקה וידע תאורטי רב (במיוחד בגרסה הבינלאומית). משחק הדמקה עוזר לפתח מחשבה לוגית מופשטת – הודות לפשטות הכללים ילדים יכולים לשחק בו ולהגיע לחשיבה מפותחת ואף חדה יותר. המשחק מהווה כלי חינוכי יעיל ותורם ליכולות הניתוח והחשיבה של השחקנים.

בדומה למשחק השחמט ישנן אליפויות דמקה, ספרי תאוריה שנכתבו למשחק וגם חידות הנסובות על המשחק. מסיבות אלו רציתי להכיר את המשחק באופן מעמיק יותר ולכן בחרתי בו כפרויקט גמר במדעי המחשב.

מטרת האפליקציה הינה להעניק גישה לאנשים שונים לשחק בחופשיות במשחק הדמקה, כך ששני שחקנים יוכלו לשחק בכל פעם. קהל היעד הוא לא מוגבל וכולם יכולים לשחק באפליקציה.

האפליקציה פותרת את המוגבלות של אנשים שלא מכירים אחד את השני, ובכל זאת הם יוכלו לשחק יחדיו. היא תהיה זמינה ותאפשר לשחק בה למשך שנים רבות קדימה.

סביבת פיתוח והרצה:

מערכת הפעלה: Windows

סביבת פיתוח: Android Studio

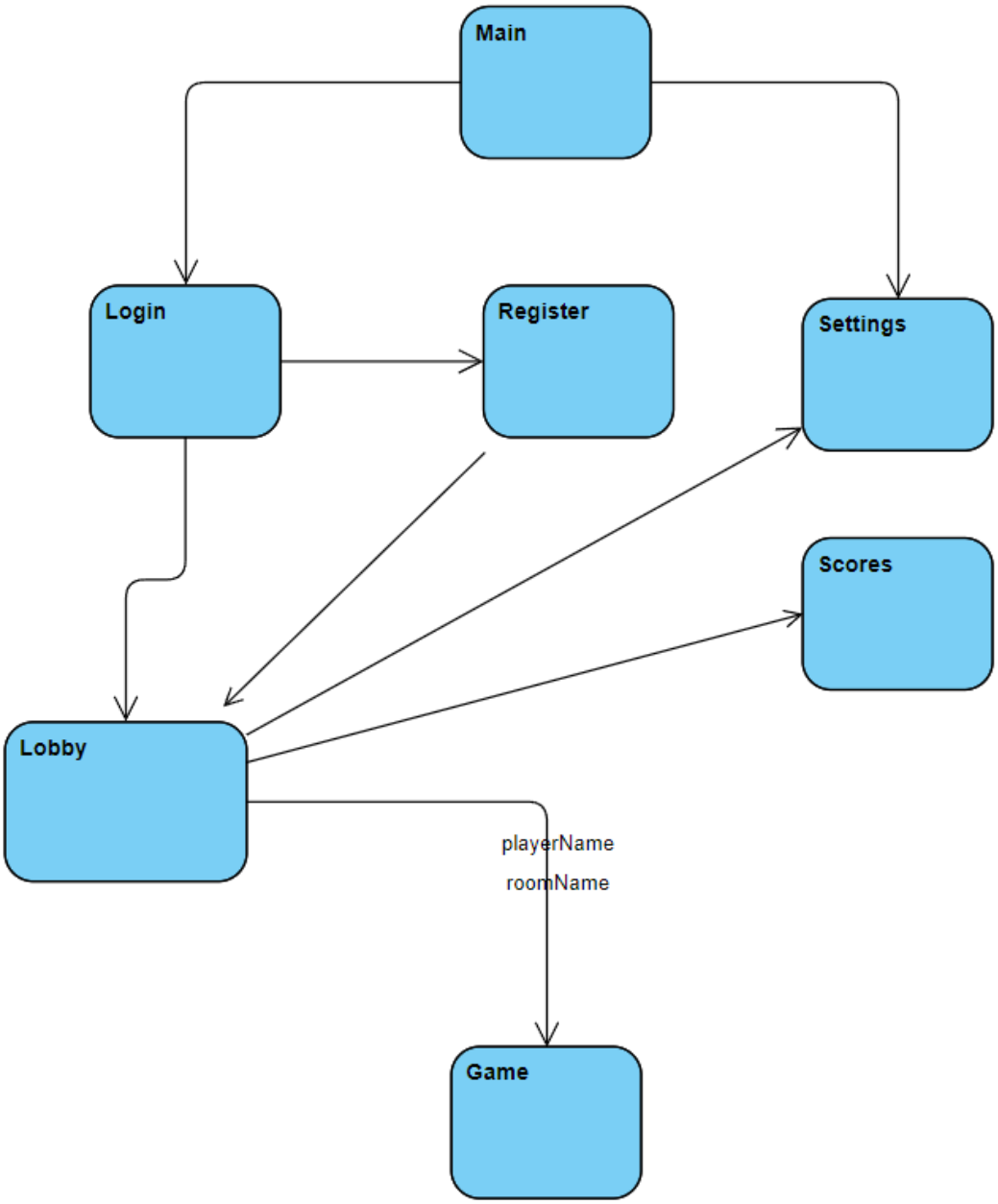
גרסת Android: 8.0

API: 26 ומעלה

הרצה: הורץ על ה emulator

מסד נתונים: Firebase

תרשים סכימטי



Main Activity

האקטיביטי הראשון שרץ כאשר פותחים את האפליקציה. אקטיביטי זה מכיל שני Buttons:

(1) העברה למסך ההתחברות.
(2) העברה להגדרות האפליקציה.

בנוסף, נעשית בדיקה באמצעות הAPI של FirebaseAuth (עם הפונקציה getCurrentUser())

אם המשתמש התחבר בעבר לאפליקציה. אם המשתמש אכן התחבר בעבר, האקטיביטי יפנה

אותו לאקטיביטי נוסף בשם Lobby, עם הפרטים שהוכנסו בעבר כך שלא יצטרך להתחבר שוב.
יתרה מזאת, בתחילת האקטיביטי נקראת פונקציה בשם removeFirestorePersistence(),

שבעזרתה מתבטל מנגנון הcache בFirestore. זאת על מנת שנוכל לקבל עדכונים בזמן אמת במהלך המשחק (במיוחד בתחילתו) ולא מה-cache.

Login Activity

האקטיביטי שאחראי על ההתחברות של המשתמש לאפליקציה. ניתן להגיע לאקטיביטי זה מ MainActivity.

הקלט שהאקטיביטי מקבל לצורך ההתחברות הוא אימייל וסיסמה.

אם זוהי הפעם הראשונה של המשתמש באפליקציה, יש לו אפשרות להירשם לאפליקציה בתחתית המסך (לחיצה על ה TextView שמכיל "New here? Create Account Here"), הוא יופנה לאקטיביטי Register ששם יוכל ליצור משתמש.

אם הפרטים שהמשתמש הכניס כקלט לאקטיביטי לא קיימים ב database, יופיע Toast מתאים שמציין כי הפרטים הללו שגויים.

לאחר התחברות מוצלחת, המשתמש מועבר לאקטיביטי Lobby.

Register Activity

האקטיביטי האחראי על רישום משתמש חדש לאפליקציה. פרטי הרישום הוא כדלקמן:

1. Username.
2. אימייל.
3. סיסמה.

כל הקלטים המבוקשים הם EditText, ויש גם אימות סיסמה (נועד לבדוק אם הסיסמה תואמת לסיסמה המקורית שהוכנסה)

לאחר רישום מוצלח, המשתמש מועבר לאקטיביטי Lobby.

Lobby Activity

אקטיביטי האחראי על הפגשת שחקנים המחוברים לאפליקציה והתחלת משחק. באקטיביטי קיים ListView המכיל את כל השחקנים המחוברים כרגע לשרת (מלבד השחקן עצמו). השרת אחראי על העברת נתונים והנגשת מידע בין טלפונים המחוברים לאפליקציה.

כאשר משתמש מועבר לאקטיביטי, הוא נחשב כ-אונליין בשרת וכל שאר המשתמשים האחרים יוכלו לראות אותו מחובר והוא יוכל לראות אותם (כל המשתמשים שמחוברים כרגע לשרת שמורים ברשימה בשם roomsList, וה ListView מתעדכן על פיו).

בנוסף, ניתן להתחיל משחק עם שחקן אשר מחובר לשרת באמצעות לחיצה על שמו בListView.

לאחר הלחיצה נשלחת הזמנה לשחקן המבוקש, והוא יכול לאשר או לדחות את הצעת המשחק של השחקן היריב.

אם השחקן המבוקש מאשר, מתחיל משחק עבור שני הצדדים והם מועברים לאקטיביטי Game בעזרת Intent, יחד עם המשתנים הבאים:

(1) playerName – הusername של השחקן הנוכחי (עבור בדיקות כמו אם הוא ה host של המשחק – אם הוא יצר את המשחק)
(2) roomName – כל המשחקים הם למעשה חדרים ששני שחקנים נמצאים בהם, אז המשתנה שומר את השם של החדר הנוכחי (של המשחק הנוכחי)

אם השחקן המבוקש דחה את ההצעה, לשחקן היריב קופץ Toast אשר מעדכן כי השחקן המבוקש דחה את ההצעה.

יתרה מזאת, השחקן ששלח את הצעת המשחק רשאי לבטל אותה. במקרה זה, לשחקן היריב קופץ Toast אשר מעדכן כי השחקן ששלח את הצעת המשחק ביטל אותה.

Game Activity

באקטיביטי זה מנוהל המשחק עצמו – אונליין. בפונקציה ()onCreate של האקטיביטי מתקבלים המשתנים playerName ו roomName המועברים אליו מאקטיביטי Lobby, והם מועברים לפונקציה ()initBoardAndDrawPieces שאחראית על אתחול לוח המשחק.

אקטיביטי זה משתמש במחלקה OnClickListenerForPieceMoves אשר אחראית לטפל בכל המהלכים שהשחקנים עושים, בשביל לממש את ההתנהגות כאשר שחקן לוחץ על אבן משחק כלשהי. בקצרה, המימוש בנוי מ-3 מחלקות:

(1) BlackPiece – מייצגת שחקן בעל צבע שחור.
(2) RedPiece – מייצגת שחקן בעל צבע אדום.
(3) KingPiece – מייצגת שחקן שהוא מלך.

כל אחת מהמחלקות המצוינות לעיל יורשות מהמחלקה Piece שמכילה את התנועה הבסיסית של כל האבנים – ()rightDiagonal ו- ()leftDiagonal. כל אחת מהמחלקות מגדירה את פונקציית ה ()move שלה (שמשתמשת בתנועה הבסיסית ()rightDiagonal ו- ()leftDiagonal) וכך האבנים זזות על גבי לוח המשחק.

המשחק מתחיל עם השחקן בעל הצבע השחור – ה host של המשחק.
בנוסף, שני השחקנים מאזינים אחד לשני דרך ה Firebase לעדכוני המהלכים של השחקן היריב.

כאשר מסתיים המשחק, לכל שחקן מוצג AlertDialog מתאים המציין את מנצח המשחק, עם כפתור חזרה ל Lobby.

Settings Activity

אקטיביטי האחראי על הגדרות האפליקציה.
<u>כרגע</u>, קיים בו את האופציה לשנות שלא יהיה רטט במכשיר כאשר משחק מתחיל.
תהליך שמירת ההגדרות מנוהל על ידי שימוש ב SharedPreferences, ונשמר תחת השם "settings". האופציה לשינוי הרטט נשמרת תחת ה key הבא: "vibrate".

ה key לעיל נקרא בתחילת הרצת האקטיביטי (בפונקציה ()onCreate) ונשמר לאחר שינוי ערכו על ידי המשתמש. ה key נקרא שוב באקטיביטי Lobby בשביל לבדוק אם המשתמש ביטל או השאיר את הרטט כאשר משחק מתחיל, והאקטיביטי פועל בהתאם.

מדריך למשתמש

רקע על המשחק:

משחק הדמקה ניתן למשחק לפי החוקים האמריקאים.

המשחק מיועד לשני שחקנים שלכל אחד מהם יש 12 אבני משחק בצבע ייחודי משלו על לוח בן 64 משבצות (8x8).

מטרת המשחק היא להוריד מהלוח ("לאכול") את כל האבנים של היריב, ובכך ולנצח את המשחק.

שחקן שנשאר ללא אבנים מוכרז כמפסיד.

סיום המשחק:

המשחק עלול להסתיים בניצחון של אחד הצדדים. ניצחון מושג אם מתקיים אחד מהתנאים הבאים:

- לשחקן היריב לא נותרו כלל כלים על הלוח (אבנים או מלכים)
- לשחקן היריב אין אפשרות לבצע מהלך מאחר שכליו חסומים

חוקי המשחק:

החוקים למשחק מורכבים מכמה שלבים: (1) אכילה, (2) תנועה, (3) יצירת מלכים.

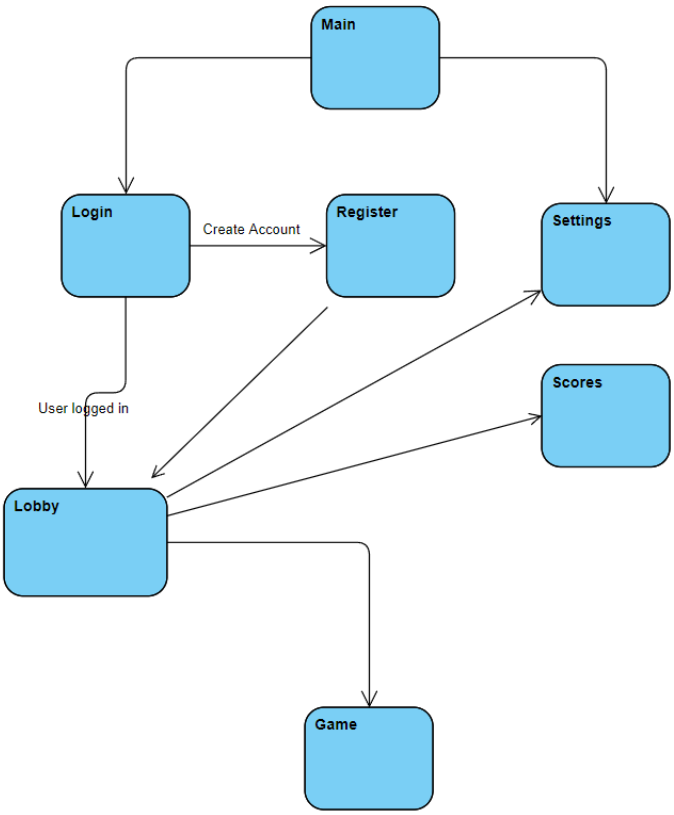
(1) אכילה (או דילוג) מתאפשרת כאשר אבן משחק מונחת במשבצת סמוכה לאבן היריב, ומעבר לאבן היריב יש מקום פנוי. בנוסף, כאשר מתאפשרת אכילה, **אין** חובה לבצע אותה. אכילה מבוצעת על ידי הנחת האבן במקום הפנוי שמעבר לאבן היריב והסרת אבן היריב מן הלוח.

(2) כל שחקן מניע בתורו אבן משחק באלכסון, ממשבצת שחורה אחת למשבצת שחורה סמוכה בכיוון היריב. על המשבצת להיות פנויה מכלים, כלומר לכל אבן יש שתי אפשרויות תנועה – לכיוון היריב ימינה ולכיוון היריב שמאלה – וכל אחת מהן עשויה להיות חסומה.

(3) כשאבן משחק מגיעה לשורה האחרונה, היא הופכת להיות "מלך", כאשר מייצגים מלך על ידי אבן משחק עם סימן של כתר עליה.
מלך, בניגוד לאבן רגילה, יכול לנוע משבצת אחת לכל הכיוונים באלכסון (כלומר גם אחורה).

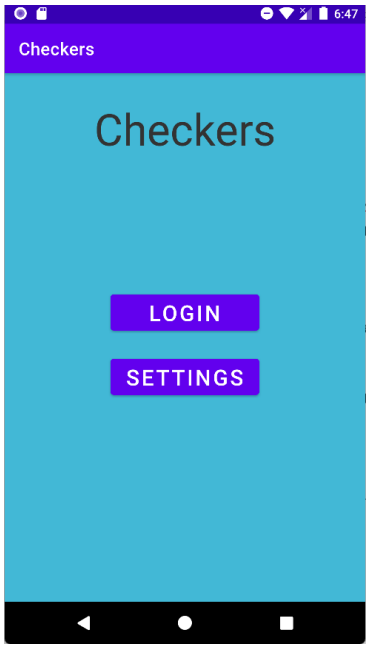
בנוסף, השחקן המתחיל הינו השחקן בעל האבנים השחורים.

תיאור מסכים



מסך פתיחה - Main

מסך הפתיחה שרואה המשתמש בפעם הראשונה שלו באפליקציה (MainActivity).
ממסך זה ניתן לנווט למסך ההתחברות ולמסך ההגדרות באמצעות כפתורים מתאימים.



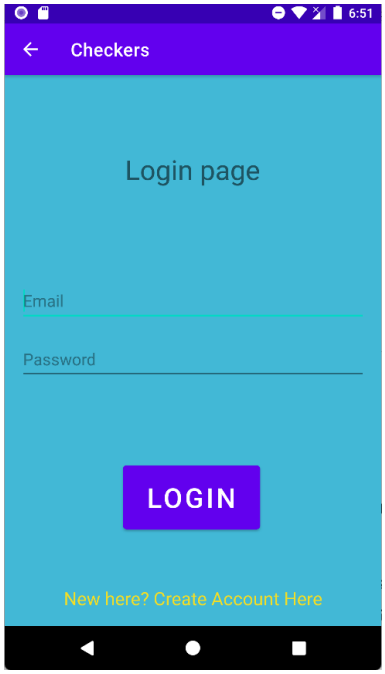
איור 1. מסך פתיחה

מסך ההתחברות - Login

במסך זה ניתן להתחבר לאפליקציה עם פרטי המשתמש הרלוונטיים : אימייל וסיסמה.
ניתן למלא פרטים אלו בתיבות הטקסט Email ו- Password.

אם זוהי הפעם הראשונה של המשתמש באפליקציה והוא לא רשום, ניתן לנווט למסך ההרשמה בתחתית המסך באמצעות כפתור.

בנוסף, על הסיסמה להיות לפחות 6 תווים, כמו במסך הרישום.



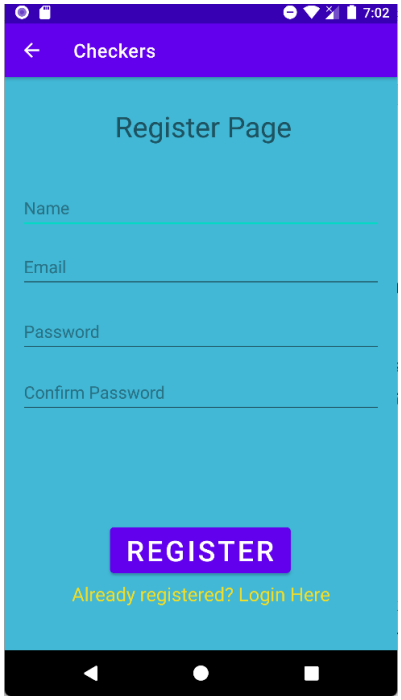
איור 2. מסך ההתחברות

מסך הרישום – Register

במסך זה אפשר להירשם לאפליקציה באמצעות מילוי פרטי הרישום הרלוונטיים : שם כינוי, אימייל וסיסמה.

ניתן למלא פרטי רישום אלו בתיבות הטקסט Name, Email, Password.
יתרה מזאת, קיימת תיבת טקסט בשם Confirm Password, שנודעה לבדיקת אימות הסיסמה.

על הסיסמה להכיל לפחות 6 תווים.



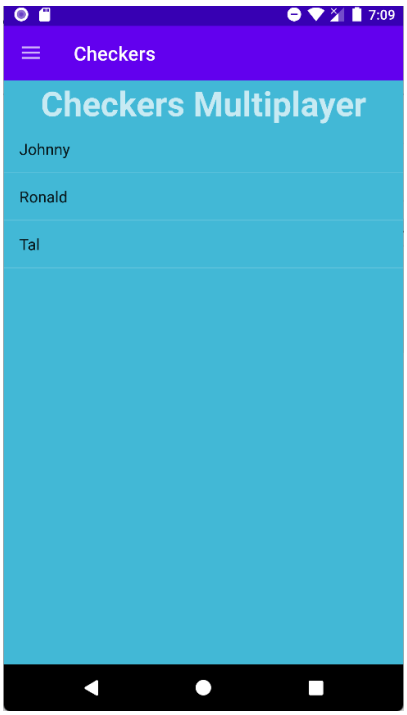
איור 3. מסך ההרשמה

מסך ההמתנה - Lobby

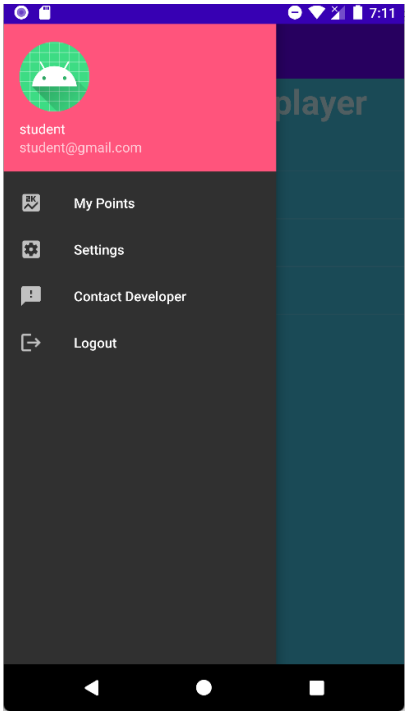
במסך זה ניתן להזמין שחקנים מחוברים למשחק דמקה.
ההזמנה מתבצעת על ידי לחיצה על שמם (אם הם מחוברים) ברשימת השחקנים המחוברים מתחת לכותרת "Checkers Multiplayer".

כאשר שחקן כלשהו הוזמן למשחק, מופיע לו AlertDialog המתאר את ההזמנה מאת השחקן ששלח אותה, והוא רשאי לאשר או לדחות את ההזמנה.
אם ידחה – לשחקן ששלח את ההזמנה יופיע AlertDialog שמעדכן כי השחקן שהוזמן דחה את ההזמנה.
אם יאשר – שני השחקנים ינווטו אל מסך המשחק (שכולל את לוח המשחק ואת כל אבני המשחק) ויוכלו לשחק בחופשיות.

בנוסף, ניתן לפתוח את התפריט לניווט משמאל בכותרת המסך (סימון של שלוש פסים בחלק העליון של המסך) ולראות את שם המשתמש והאימייל שלו, לנווט למסך ההגדרות (לדוגמה – בשביל לבטל את הרטט כאשר מתחיל משחק), למסך הניקוד, וגם לשלוח חוות דעת או לדווח על באג באפליקציה דרך SMS, או פשוט להתנתק מהאפליקציה..



איור 4. רשימת שחקנים מחוברים



איור 5. תפריט במסך ההמתנה

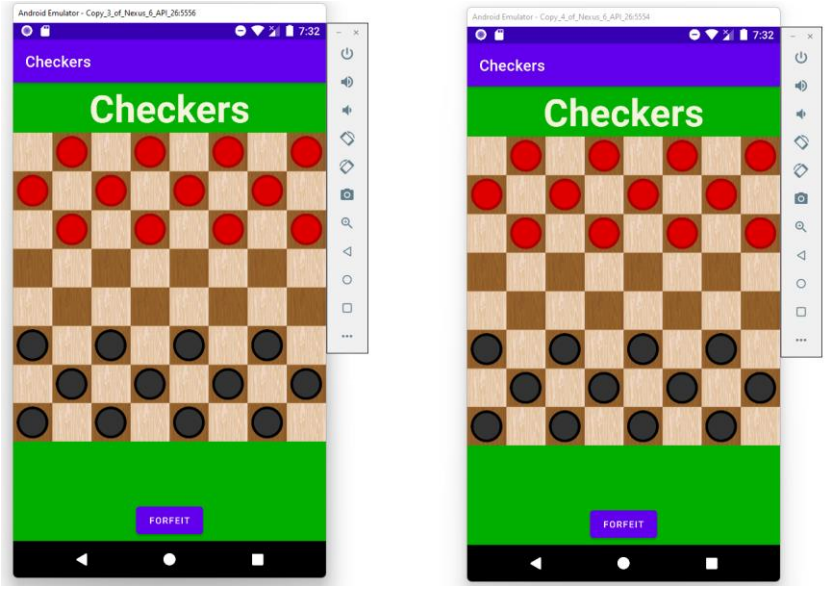
מסך המשחק - Game

במסך זה ניתן לשחק בחופשיות במשחק ואפשר לראות את הלוח ואת אבני המשחק של כל שחקן.

השחקן שהוזמן הוא שמתחיל (אבני המשחק בצבע שחור).

בנוסף, מופיעה תיבת טקסט בתחתית המסך שמתארת אם התור הנוכחי הוא שלך או של השחקן היריב. (הערה: זה לא מופיע באיור של המסך כי זה הוספה)

כאשר משחק מסתיים, מוצג AlertDialog המכריז על מנצח המשחק (עבור שני השחקנים) עם כפתור חזרה למסך ההמתנה.



איור 6. מסך המשחק

מסך ההגדרות - Settings

במסך זה ניתן לשנות את הגדרות האפליקציה, כגון ביטול האפשרות לרטט כאשר מתחיל משחק.
ההגדרות נשמרות גם לאחר סגירת האפליקציה באמצעות SharedPreferences.

איור 7. מסך ההגדרות

בסיס נתונים

Firebase Authentication

השימוש בdatabase לעיל הינו עבור שמירת פרטי המשתמשים שנרשמים לאפליקציה, והקריאה שלהם בעת ההתחברות.

ה Activities שמשתמשים בו הם :

Login Activity

במסך ההתחברות מתקבלים פרטי המשתמש בתיבות הטקסט Email ו Password.
לאחר מכן, מתבצעת בדיקה מול הdatabase אם הנתונים הללו קיימים (דרך הAPI של FirebaseAuth).
אם הנתונים קיימים ומתאימים למה שהמשתמש הכניס, הוא יועבר ל Lobby Activity.
אם הנתונים לא קיימים, יוצג Toast למשתמש שמעדכן כי הפרטי ההתחברות שגויים.
יתרה מזאת, אם האימייל שהמשתמש הכניס לתיבת הטקסט Email נרשם בעבר לבסיס הנתונים, יוצג Toast שמעדכן כי האימייל בשימוש.

Search by email address, phone number, or user UID | Add user

Identifier	Providers	Created	Signed In	User UID
fake@gmail.com		May 27, 2022	May 27, 2022	6pGM2okecyanqKA2xdi6gXqg6W63
tals06061@gmail.com		May 27, 2022	May 27, 2022	4hoHxXKnI3SPLF4ZbIVS2LB3JII2
someone@gmail.com		May 26, 2022	May 30, 2022	Ib4IQnFZFYgFyL0I4VCxnvsvnGp1
idk@gmail.com		May 26, 2022	May 27, 2022	dYDCYksC9iYj4tPvNkaUihFZZvO2
lol1@gmail.com		May 25, 2022	May 25, 2022	a4ipFxPntHQCHzIV8OA58y0CAxi1
student2@gmail.com		May 20, 2022	May 22, 2022	q2F2ByIZFwTFySSW3I9i2x107JE3
student@gmail.com		May 20, 2022	May 30, 2022	WbsfWdTsRoRq0hHsVI6Yt8ffLQ92
lol2@gmail.com		May 19, 2022	May 19, 2022	fyxAiZn1T3VWeoWr8PCGNeIQ55u2
ok@gmail.com		May 18, 2022	May 18, 2022	1NGyNoB6mERPcnFRacNWzYsVv...

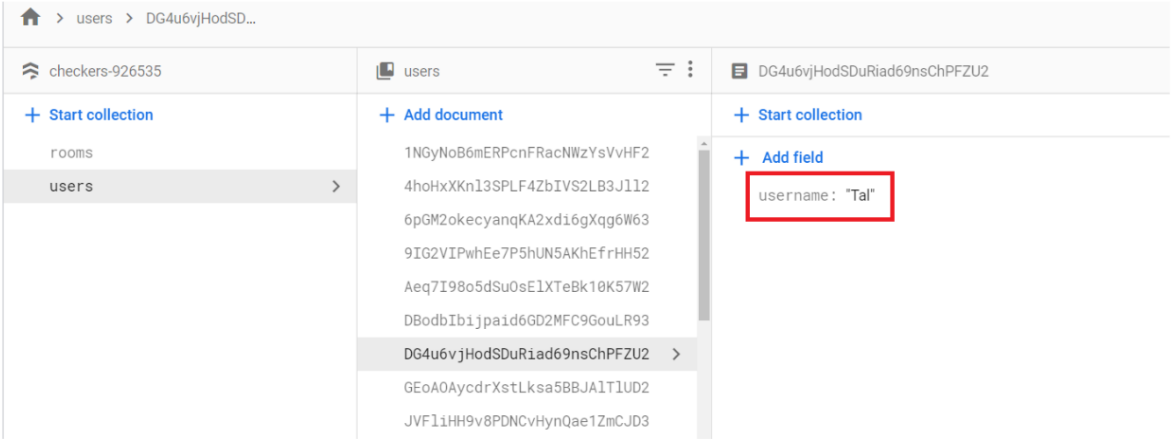
איור 8. רשימת המשתמשים הרשומים לבסיס הנתונים

Register Activity

במסך הרישום הפרטים הנשלחים לdatabase הינם : אימייל, סיסמה, ושם המשתמש (username).

אימייל המשתמש וסיסמתו הם הפרטים איתם הוא גם מתחבר, אך השדה ה username שלו נשמר בבסיס נתונים נוסף - Firestore.

השדה username נשמר תחת ה user id של המשתמש (שנוצר כאשר יוצרים משתמש בFirebaseAuth) בdocument, ומכיל שדה בשם username מסוג String שמכיל את ערך זה.



איור 9. שמירת השדה username בFirestore

Firestore

השימוש במבנה נתונים זה הינו עבור העברת נתונים הקשורים למשחק האונליין, ושמירת שדה ה username.

ה Activities שמשתמשים בו הם:

Lobby Activity

אקטיביטי זה אחראי על הפגשת שחקנים המחוברים לשרת ויצירת משחק.
כאשר משתמש מתחבר לאפליקציה, הוא נחשב מחובר לשרת ולכן גם נוצר חדר משחק בשמו (שדה ה username שבחר בהרשמה לאפליקציה).

חדר המשחק מכיל שני משתנים:

(1) host – משתנה מסוג מחרוזת ששומר את שם מנהל החדר. אשר נקבע לפי שמו של החדר.
(2) isInGame – משתנה מסוג Boolean שמכיל אמת או שקר אם החדר במשחק (תפוס או לא).

כאשר משתמש נוסף נכנס לחדר, הוא נחשב כ-"אורח" וקורים כמה דברים ברקע:

(1) הוא מתווסף לחדר בתור "guest" – משתנה מסוג מחרוזת ששומר את שמו של האורח שנכנס, ומשנה את ערכו של isInGame ל-true.
(2) לשחקן האורח מוצג AlertDialog שמעדכן כי ההזמנה נשלחה למנהל החדר, והוא מתחיל להאזין לdocument בשם hostUpdates.
(3) מנהל החדר (host) שם לב שערכו של isInGame התעדכן, ויוצר AlertDialog שמעדכן את השחקן כי שחקן אחר הצטרף לחדר ומאתגר אותו למשחק, ויש לו את האפשרות לאשר או לדחות את ההזמנה.

אם מנהל החדר אישר את ההזמנה:

(1) נוצר collection בשם hostUpdates אשר מנהל החדר יוצר בתוכו document בשם gameStatus ומעדכן שם משתנה בשם "startGame" שהינו מסוג Boolean ל-true.
(2) המשתמש מופנה לאקטיביטי חדש בשם GameActivity.
(3) המשתמש האורח שם לב שהערך של startGame התעדכן לtrue, ומריץ אקטיביטי חדש בשם GameActivity.

אם מנהל החדר דחה את ההזמנה:

(1) ערך המשתנה "startGame" בנתיב שצוין לעיל יתעדכן לfalse.
(2) המשתמש האורח שם לב שהערך של startGame התעדכן לfalse, ומציג Toast שמעדכן כי מנהל החדר דחה את ההזמנה.
(3) מנהל החדר מוחק את המשתנה "guest" מהחדר ומעדכן את המשתנה "isInGame" לfalse.

Game Activity

באקטיביטי זה מתנהל משחק הדמקה – אונליין.

כאשר משחק מתחיל, לנתיב החדר (נשמר במשתנה roomRef מסוג DocumentReference עבור שני השחקנים) מתווסף collection בשם gameplay ואליו מתווסף document בשם gameUpdates.

בdocument זה נוצר משתנה בשם "isBlackTurn" מסוג Boolean וערכו true (מאחר והשחקן השחור הוא המתחיל את המשחק).

משתנה זה נוצר על מנת לקבוע את תורו של השחקן הנוכחי – שחקן בעל האבנים השחורים הוא הhost של החדר, והאורח הוא השחקן בעל האבנים שאדומים. כל אחד מהשחקנים מאזין לנתיב ערך זה ומשחקים בתורם, ובסיום כל מהלך שלהם הם משנים את הערך בהתאם.

לאחר מכן, כל שחקן מאזין לתנועות האבנים של השחקן היריב שמעדכן את מיקום תנועותיו במיקום הבא:

(1) עבור ה-host: rooms/(roomName)/gameplay/hostMovesUpdates
(2) עבור ה-guest: rooms/(roomName)/gameplay/guestMovesUpdates

Variable Name	endAxis (String)	startAxis (String)	isKing (Boolean)	isJump (Boolean)	jumpedAxis (String)
Format	"x-y"	"x-y"	True/False	True/False	"x-y"

כל אחד מהשחקנים מעדכן כל תנועה שהוא עושה לפי הformat הבא:

כאשר שחקן שם לב שהוספו משתנים לנתיב עדכון התנועות של השחקן היריב – משמע השחקן היריב עשה תנועה, הוא מעדכן את המשחק אצלו גם כן (לוקאלית).

שימוש בקבצים

שימוש ב-SharedPreferences בשביל שמירת ההגדרות שהמשתמש בוחר.

ה Activities שמשתמשים בזה הם :

Settings Activity

באקטיביטי זה ניתן לשנות את הגדרות האפליקציה. לדוגמה, ניתן לבטל את הרטט עבור תחילת משחק. אופציה זו נשמרת תחת הkey בשם "vibrate" מסוג Boolean. ערך הdefault הינו true.

Game Activity

קריאת הגדרות המשתמש ולפעול לפיו. לדוגמה, אם האפשרות לרטט עבור תחילת משחק מבוטלת, האקטיביטי בודק זאת עם תנאי (של מה שנשמר בkey הזה בSharedPreferences) ולא עושה רטט בטלפון.

מדריך למפתח

פירוט Activities

Lobby Activity (איור 4)

אקטיביטי זה אחראי על הפגשת שחקנים מחוברים ויצירת משחקים בין זרים.

הסבר משתנים:

(1) toolbar – עבור דריסת ה Action bar שהוא הdefault (בשביל התפריט).

(2) drawer – הDrawer Layout החלופי שאיתו נדרס הAction bar וכך נוצר התפריט.

(3) mUsername – משתנה מסוג TextView שמחזיק הפניה לתיבת התצוגה שבה יופיע שם המשתמש.

(4) mEmail – משתנה מסוג TextView שמחזיק הפניה לתיבת התצוגה שבה יופיע האימייל של המשתמש.

(5) broadcastReceiver – אובייקט הbroadcastReceiver שאיתו משתמשים בשביל לקבל התראות כאשר אין חיבור לאינטרנט.

(6) hostUpdatesListener – אובייקט מסוג ListenerRegistration, נועד בשביל למחוק האזנה לנתיב העדכונים של המנהל שנעשה לFirestore על ידי המשתמשים (לדוגמה בעת יציאה מהאפליקציה).

(7) guestUpdatesListener - אובייקט מסוג ListenerRegistration, נועד בשביל למחוק האזנה לנתיב העדכונים של האורח שנעשה לFirestore על ידי המשתמשים (לדוגמה בעת יציאה מהאפליקציה).

(8) listView – המשתנה מסוג ListView ששומר את הview של רשימת המשתמשים המחוברים לשרת.

(9) hostUpdatesRef – משתנה שנועד להחזיר הפניה לdocument בFirestore.

(10) guestUpdatesRef – משתנה שנועד להחזיר הפניה לdocument בFirestore.

(11) fstore – משתנה שמחזיק instance לFirestore.

```
Lobby
+ toolbar: Toolbar
+ drawer: DrawerLayout
+ mUsername: TextView
+ mEmail: TextView
+ broadcastReceiver: BroadcastReceiver
+ hostUpdatesListener: ListenerRegistration
+ guestUpdatesListener: ListenerRegistration
+ listView: ListView
+ hostUpdatesRef: DocumentReference
+ guestUpdatesRef: DocumentReference
- fStore: FirebaseFirestore
+ onCreate(): void
+ onBackPressed(): void
+ onStop(): void
+ onResume(): void
+ onDestroy(): void
+ registerBroadcastListener(): void
+ unregisterBroadcastListener(): void
+ contactHandler(): void
+ listenForRoomUpdates(): void
+ initNavHeader(): void
+ disconnectUser(): void
+ connectUser(): void
- startGame(v: Vibrator)
- setListenerForGuestUpdates(): void
- setListenerForHostUpdates(): void
- gameInvitationHandler(): void
- getIsInGame(): boolean
```

כותרת הפעולה מתוך java	תיאור הפעולה
public void registerBroadcastListener()	הפעולה מפעילה את הbroadcastReceiver בכדי שיקבל עדכונים.
public void unregisterBroadcastListener()	הפעולה מבטלת את הbroadcastReceiver. (לדוגמה בעת יציאה מהאפליקציה)
public void contactHandler()	הפעולה פותחת את תוכנת הSMS בטלפון בשביל שהמשתמש ישלח הודעה למתכנת האפליקציה
private boolean getIsInGame()	הפעולה מחזירה את ערכו של isInGame שנמצא בFirestore בנתיב: rooms/(roomName)
public void listenForRoomUpdates()	הפעולה גורמת לטלפון להתחיל להאזין לעדכונים בחדר של המנהל (אם אורח הצטרף)
private void gameInvitationHandler()	הפעולה מטפלת בכל העניין של בקשת הזמנה למנהל, הצגת AlertDialog אצל האורח ואצל המנהל, וכו'.
public void handleGuestInGameInvitation (AlertDialog.Builder gameRequestDialogBuilder, String hostUsername, Vibrator vibrator)	הפעולה מטפלת בצד של האורח כאשר מתבצעת הזמנת משחק. זאת אומרת, היא מראה AlertDialog ומחכה לתגובת המנהל
public void handleHostInGameInvitation (AlertDialog.Builder gameRequestDialogBuilder,	הפעולה מטפלת בצד של המנהל כאשר מתבצעת הזמנת משחק. זאת אומרת, היא מראה AlertDialog של ההזמנה ובודקת בכל

רגע שהאורח לא מבטל את ההזמנה. אם הוא כן ביטל, מוצג AlertDialog בהתאם למנהל.	Map<String, Object> gameRequestData, Vibrator vibrator)
הפעולה גורמת לטלפון המנהל להתחיל להאזין לעדכוני האורח (אם ביטל את ההזמנה).	private void setListenerForGuestUpdates (String guestUsername, AlertDialog gameRequestDialog, DocumentReference guestUpdatesRef)
הפעולה גורמת לטלפון האורח להתחיל להאזין לעדכוני המנהל (אם דחה או אישר את ההזמנה).	private void setListenerForHostUpdates (String hostUsername, AlertDialog gameRequestDialog, DocumentReference hostUpdatesRef, Vibrator vibrator)
הפעולה בודקת לפי ההגדרות של המשתמש אם אפשרות הרטט כאשר משחק מתחיל היא דלוקה, ופועלת בהתאם. לאחר מכן, היא מנווטת ל Game Activity.	private void startGame (Vibrator v)
הפעולה מאתחלת את התפריט בצד שמאל למעלה (3 פסים), כך ששם המשתמש והאימייל שלו יופיע שם וגם כל הכפתורים למיניהם. כמו כן, הפעולה מחברת את המשתמש לשרת באמצעות קריאה ל()connectUser.	public void initNavHeader()
הפעולה מוחקת את חדר המשחק.	public void disconnectUser()
הפעולה יוצרת חדר חדש בשם המשתמש ומעדכנת את השדה roomName.	public void connectUser()
הפעולה סוגרת את התפריט אם נלחץ הכפתור "חזרה אחורה" בטלפון.	public void onBackPressed()
הפעולה נקראת כאשר האפליקציה לא נראית לעין יותר, והינה מוחקת את כל ההאזנות שנעשו בחדר (כגון guestUpdatesListener וכו').	public void onStop()
הפעולה נקראת כאשר המשתמש חוזר בחזרה לאפליקציה והיא מחברת אותו שוב לשרת.	public void onResume()
הפעולה נקראת כאשר האפליקציה נסגרת לחלוטין, והינה מנתקת את המשתמש מהשרת. (**חשוב לשים לב** כי onDestroy **לא נקרא תמיד.** לכן עד שאמצע פתרון טוב יותר, ההתנתקות תתבצע כך או על ידי המשתמש	public void onDestroy()

לוחץ ישירות על הכפתור Log out שתגרום להתנתקותו מהשרת)	

Game Activity (איור 6)

האקטיביטי בו מתנהל המשחק.

המשתנה היחיד הקיים בו (שאינו סטטי) הינו board, לוח המשחק הדו ממדי.

```
Game
+ board: Board
+ onCreate(): void
+ onBackPressed(): void
+ setOnClickForPieces():  void
+ initBoardAndDrawPieces(String playerName, String roomName): void
- listenDBForPieceMoves(DocumentReference playerMovesUpdatesRef, boolean isPieceBlack, Piece piece): ListenerRegistration
- initImageViews(): void
```

תיאור הפעולה	כותרת הפעולה מתוך java
הפעולה מאתחלת את המחלקה (לדוגמה קבלת המשתנים roomName ו playerName)	public void onCreate()
הפעולה נקראת כאשר האקטיביטי אינו מוצג למשתמש יותר. פעולה זו אחראית למחוק את כל הlisteners שאותחלו במהלך המשחק (הערה – זה לא מופיע בתרשים UML, כי זוהי הוספה).	public void onStop() {
הפעולה אחראית לאתחל את המערך הדו ממדי מסוג ImageView ששומר הפניות ישירות לכל משבצת בקובץ xml שאבן משחק יכולה להיות עליה.	private void initImageViews()
הפעולה יוצרת onClickListener על כל אחד מאבני המשחק, ומציבה בו את אובייקט מסוג OnClickListenerForPieceMoves	public void setOnClickForPieces()
הפעולה מאתחלת את לוח המשחק הדו ממדי כאשר האדומים ממוקמים מתחת לשורה ה2, והשחורים מעל השורה ה5. בנוסף, הפעולה גורמת לטלפון של המנהל להאזין לעדכוני התנועות של האורח, וגם להפך.	public void initBoardAndDrawPieces(String playerName, String roomName)

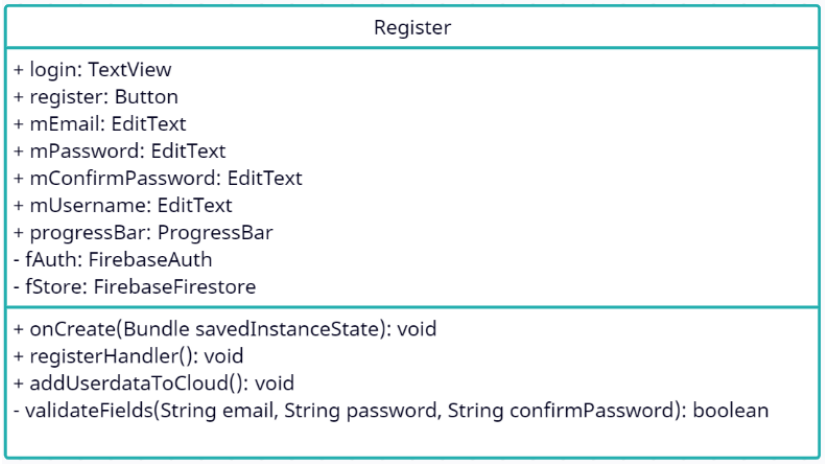
הפעולה גורמת לטלפון להאזין למיקום ה playerMovesUpdatesRef שקיבלה כפרמטר בdatabase.	Private ListenerRegistration listenDBForPieceMoves (DocumentReference playerMovesUpdatesRef, boolean isPieceBlack, Piece piece)
הפעולה יוצרת ומחזירה אובייקט מסוג Piece אשר מתווסף לboardArray, בכל פעם כאשר השחקן היריב משחק בתורו (הערה : הפעולה לא מופיע בתרשים UML, כי זוהי הוספה).	private Piece createAddedPiece (boolean isPieceBlack, boolean isKingDb, int endX, int endY)

Register Activity (איור 3)

האקטיביטי האחראי על רישום המשתמש לdatabase.

המשתנים במחלקה:

(1) login – משתנה מסוג TextView שכאשר לוחצים עליו, הוא מנווט את המשתמש לאקטיביטי Login.
(2) register – כפתור הרישום.
(3) mEmail – תיבת הטקסט של האימייל שהמשתמש הכניס.
(4) mPassword – תיבת הטקסט של הסיסמה שהמשתמש הכניס.
(5) mConfrimPassword – תיבת הטקסט של אימות הסיסמה.
(6) mUsername – תיבת הטקסט של שם המשתמש.
(7) progressBar – משתנה מסוג ProgressBar שמופעל כאשר הרישום מתבצע מול הdatabase.
(8) fAuth – instance של FirebaseAuth
(9) fStore – instance של FirebaseFirestore

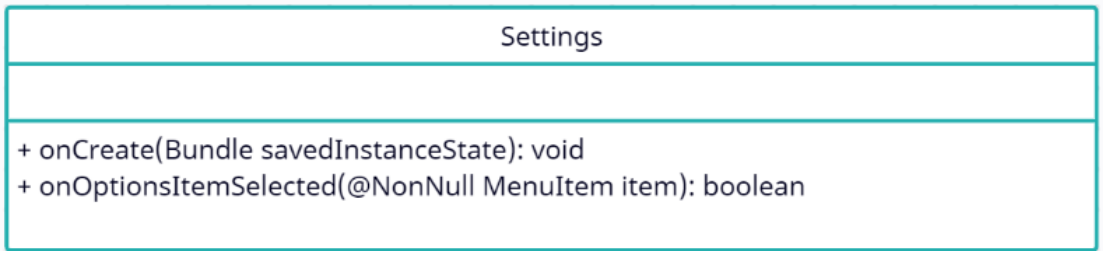


כותרת הפעולה מתוך java	תיאור הפעולה

אתחול המחלקה (לדוגמה מציאת הviews של כל אחד מהתיבות טקסט)	public void onCreate(Bundle savedInstanceState)
הפעולה האחראית לטיפול כאשר לוחצים על כפתור הregister.	public void registerHandler()
הוספת username המשתמש והסיסמה שבחר לdatabase.	public void addUserdataToCloud(String username, String email)
פונקציית האימות נתונים (לדוגמה – חייב להכניס לפחות 6 תווים בסיסמה). מחזירה אמת אם אימות קלט המשתמש עבר בהצלחה, אחרת מחזירה שקר.	private boolean validateFields(String email, String password, String confirmPassword)

Settings Activity (איור 7)

מסך ההגדרות. לדוגמה, ביטול האפשרות לרטט כאשר משחק מתחיל.



תיאור הפעולה	כותרת הפעולה מתוך java
אתחול המחלקה (לדוגמה – בדיקת ערכו השמור של vibrate בsharedPreferences)	public void onCreate(Bundle savedInstanceState)
פעולה הנקראת כאשר לוחצים על כפתור החץ הקטן למעלה בכותרת האקטיביטי. הפעולה חוזרת לאקטיביטי שפתח אותה.	public boolean onOptionsItemSelected(@NonNull MenuItem item)

Login (איור 2)

אקטיביטי ההתחברות.

המשתנים במחלקה:

(1) login – כפתור ההתחברות.

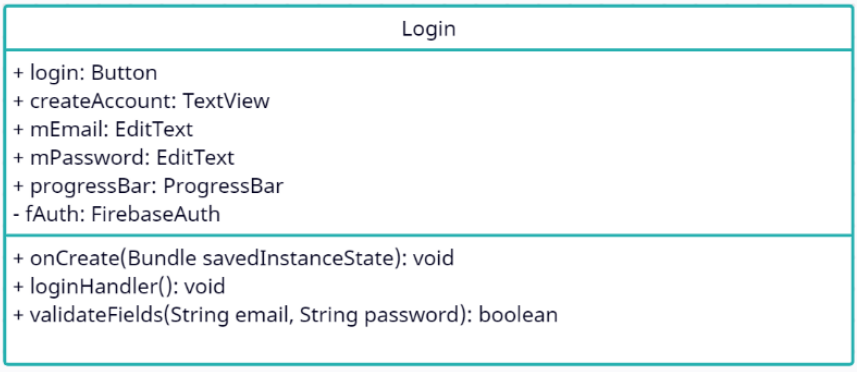
(2) createAccount – צג טקסט המפנה בעת לחיצה עליו אל האקטיביטי Register.

(3) mEmail – תיבת הטקסט של האימייל של המשתמש.

(4) mPassword – תיבת הטקסט של הסיסמה של המשתמש.

(5) progressBar – משתנה מסוג ProgressBar שמופעל כאשר המשתמש מתחבר לאפליקציה (לאחר הלחיצה על הכפתור login)

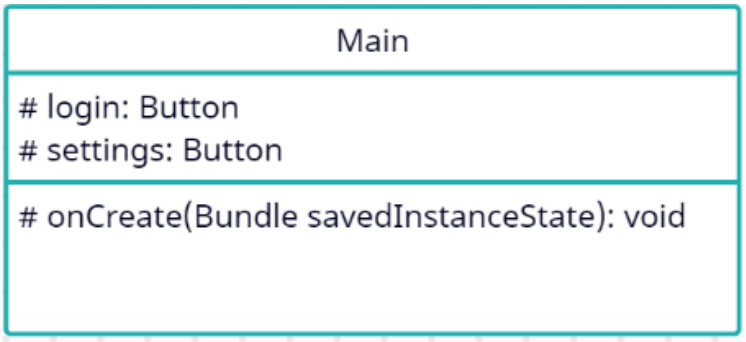
(6) instance – fAuth מסוג FirebaseAuth



תיאור הפעולה	כותרת הפעולה מתוך java
אתחול המחלקה. (לדוגמה יצירת הonClickListener לbuttons)	public void onCreate(Bundle savedInstanceState)
הפעולה שמטפלת במה שקורה כאשר המשתמש לחץ על הכפתור login.	public void loginHandler()
הפעולה שמאמתת את קלט המשתמש (לדוגמה אם הסיסמה היא לפחות בעלת 6 תווים). היא מחזירה אמת אם האימות הושלם בהצלחה, אחרת שקר.	private boolean validateFields(String email, String password)

Main Activity (איור 1)

אקטיביטי הפתיחה של האפליקציה. בעזרת אקטיביטי זה ניתן לנווט להגדרות האפליקציה ולמסך ההתחברות. למחלקה יש שני משתנים: כפתור login, וכפתור settings.



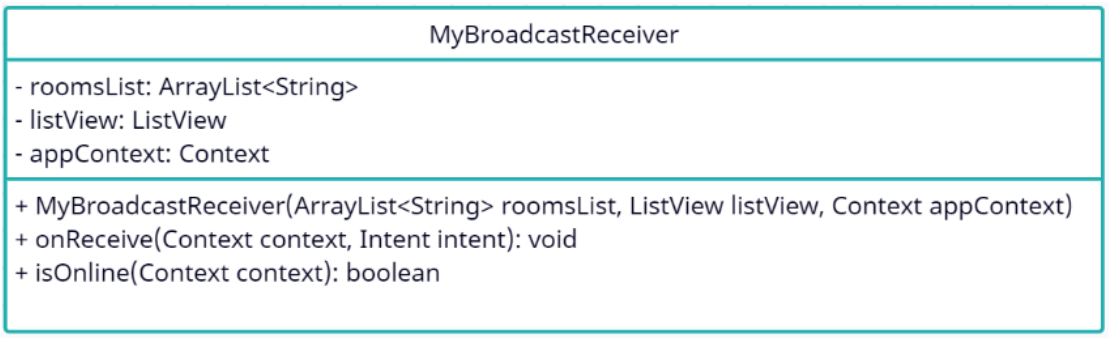
תיאור הפעולה	כותרת הפעולה מתוך java

אתחול המחלקה. פעולה זו מבצעת setOnClickListener על שני הכפתורים וכך הם מנווטים ל Activities.	protected void onCreate (Bundle savedInstanceState)
פעולה המחזירה הגדרות ל Firestore, שכוללות ביטול מנגנון הcache. ביטול מנגנון זה הינו הכרחי מאחר ואנו רוצים לקבל עדכונים בזמן אמת במהלך המשחק	public FirebaseFirestoreSettings removeFirestorePersistence () NOTE: this is not presented correctly in the UML, because of updating it.

פירוט עצמים

MyBroadcastReceiver

מחלקה היורשת מ BroadcastReceiver, ודורסת את הפונקציה onReceive() במטרה לקבל עדכון כאשר אין אינטרנט בטלפון, ולנתק את הקשר עם הdatabase.



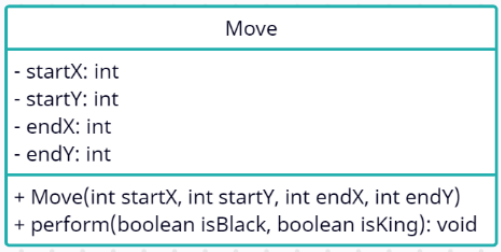
איור 11. תרשים UML – מחלקת MyBroadcastReceiver

תיאור הפעולה	כותרת הפעולה מתוך java
בנאי המחלקה. מקבל את הפרמטרים roomsList, ListView, appContext בהתאמה ומאתחל אותם. שדות אלו נועדו בשביל העברת פרמטרים לפונקציה updateListView() שנמצאת בonReceive()	public MyBroadcastReceiver (ArrayList<String> roomsList, ListView listView, Context appContext)
דריסת הפונקציה ב BroadcastReceiver. הפונקציה בודקת אם הטלפון מחובר לאינטרנט, ואם כן היא מעדכנת את רשימת השחקנים המחוברים לשרת. אם לא, היא מציגה Toast שמעדכן כי אין גישה לאינטרנט ומנתקת את עדכון רשימת השחקנים המחוברים לשרת.	@Override public void onReceive (Context context, Intent intent)
פונקציה שמחזירה true אם יש חיבור לאינטרנט, אחרת מחזירה false.	public boolean isOnline (Context context)

Move

מחלקה המגדירה מהלך בסיסי על המסך.

שימושה הינו במחלקה OnClickListenerForPieceMoves בתנועת האבנים על גבי המסך.



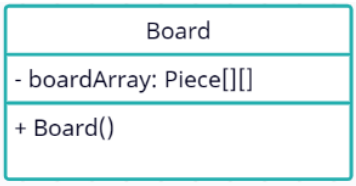
תיאור הפעולה	כותרת הפעולה מתוך java
בנאי המחלקה. מקבל int startX, int startY, int endX, int endY, ומאתחל אותם.	public Move (int startX, int startY, int endX, int endY)
פונקציה אשר עושה את הזזה על המסך. היא מחליפה את התמונה של האבן במיקום startX, startY למיקום endX, endY.	public void perform (boolean isBlack, boolean isKing)

Board

מחלקה שמתארת את לוח המשחק.

לוח המשחק הינו מערך דו ממדי של המחלקה Piece (הכלה).

שימושה הינו במחלקה OnClickListenerForPieceMoves עבור מערך כל לוח המשחק עליו מתבצעות התנועות.



תיאור הפעולה	כותרת הפעולה מתוך java
בנאי המחלקה. הבנאי מאתחל את המערך הדו ממדי boardArray מסוג Piece[][] לגודל SIZE	public Board ()

Piece

מחלקה המגדירה אבן משחק.

שימושה הינו במחלקה OnClickListenerForPieceMoves ביצירת אבן חדשה.

<u>הערה</u>: הפעולות isGameOver() וגם gameOver() הועברו למחלקה DBUtils (isGameOver() שונתה ל checkGameOver())

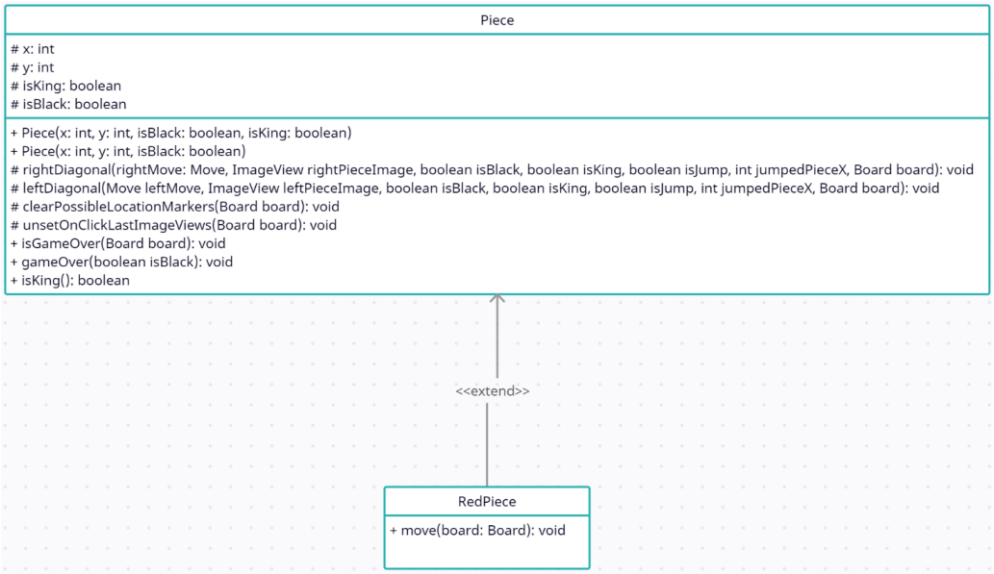
```
Piece
# x: int
# y: int
# isKing: boolean
# isBlack: boolean
+ Piece(int x, int y, boolean isBlack, boolean isKing)
+ Piece(int x, int y, boolean isBlack)
# rightDiagonal(Move rightMove, ImageView rightPieceImage, boolean isBlack, boolean isKing, boolean isJump, int jumpedPieceX, Board board): void
# leftDiagonal(Move leftMove, ImageView leftPieceImage, boolean isBlack, boolean isKing, boolean isJump, int jumpedPieceX, Board board): void
# clearPossibleLocationMarkers(Board board): void
# unsetOnClickLastImageViews(Board board): void
+ isGameOver(Board board): void
+ gameOver(boolean isBlack): void
+ isKing(): boolean
```

כותרת הפעולה מתוך java	תיאור הפעולה
public Piece (int x, int y, boolean isBlack, boolean isKing, TextView currentTurn) NOTE: this is not presented correctly in the UML, because of updating it.	בנאי המחלקה. אתחול ערכי האובייקט x, y, isBlack, isKing, currentTurn.
public Piece (int x, int y, boolean isBlack, TextView currentTurn) NOTE: this is not presented correctly in the UML, because of updating it.	בנאי נוסף במחלקה. מאתחל את ערכי האובייקט x, y, isBlack, currentTurn, ואת isKing ל-true.
protected void rightDiagonal (Move leftMove, ImageView leftPieceImage, boolean isBlack, boolean isJump, int jumpedPieceX, Board board) NOTE: this is not presented correctly in the UML, because of updating it.	פונקציית התזוזה עבור ה"אלכסון הימיני" של אבן משחק. "אלכסון ימיני" הינו האלכסון מצד ימין של אבן המשחק הנוכחית. עבור אבן שחורה, אכלסון זה יהיה בכיוון ימין <u>למעלה</u>. עבור אבן אדומה, אלכסון זה יהיה בכיוון ימין <u>למטה</u>. פונקציה זו הינה קוראת באופן רקורסיבי לפונקציה displayMoveOptionsAndMove.
protected void leftDiagonal (Move leftMove, ImageView leftPieceImage, boolean isBlack, boolean isJump, int jumpedPieceX, Board board)	פונקציית התזוזה עבור ה"אלכסון השמאלי" של אבן משחק. "אלכסון שמאלי" הינו האלכסון מצד שמאל של אבן המשחק הנוכחית. עבור אבן שחורה, אכלסון זה יהיה בכיוון שמאל <u>למעלה</u>.

עבור אבן אדומה, אלכסון זה יהיה בכיוון שמאל למטה. פונקציה זו הינה קוראת באופן רקורסיבי לפונקציה displayMoveOptionsAndMove.	NOTE: this is not presented correctly in the UML, because of updating it.
הפעולה אחראית לעדכון הboardArray לאחר תזוזה של אבן משחק. הפעולה נדרסת על ידי כל אחת מהמחלקות היורשות ממנה, ולכן הקוד שלה צפוי שלא להתבצע.	protected void updateBoardArray (Board board, int endX, int endY) NOTE: this is not presented correctly in the UML, because of updating it.
פעולה האחראית לבדוק אם ל Piece ישנם מהלכים לעשות או לא. מחזירה אמת אם הPiece יכול לזוז, אחרת מחזירה שקר. הפעולה נדרסת על ידי כל אחת מהמחלקות היורשות ממנה, ולכן הקוד שלה צפוי שלא להתבצע.	public boolean canMove (Board board) NOTE: this is not presented correctly in the UML, because of updating it.
כאשר לוחצים על אבן כלשהי היא מודגשת. פונקציה זו אחראית למחוק את ההדגשה כאשר לוחצים על אבן אחרת.	protected void clearPossibleLocationMarkers (Board board)
כאשר לוחצים על אבן כלשהי מוצגים למשתמש התנועות האפשריות שניתן לעשות עם אבן זו. פונקציה זו אחראית למחוק את תנועות אפשריות אלו אם נלחץ על אבן אחרת.	protected void unsetOnClickLastImageViews (Board board)

RedPiece

מחלקה זו מגדירה אבן משחק אדומה והיא יורשת מPiece.

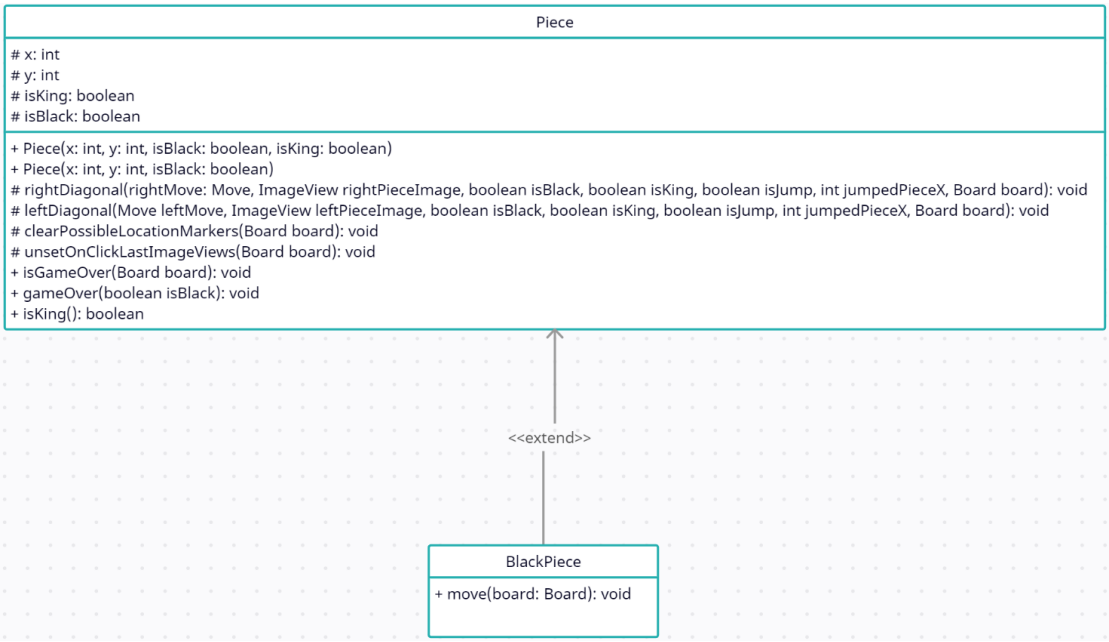


כותרת הפעולה מתוך java	תיאור הפעולה
public RedPiece (int x, int y, TextView currentTurn) NOTE: this is not presented correctly in the UML, because of updating the class.	בנאי המחלקה. השדות x, y, isBlack, isKing הינם מורשים מהמחלקה Piece והבנאי קורא לבנאי האב super()..
public void move (Board board)	תנועת אבן אדומה על גבי המסך. מתבצעת בדיקה לפני התנועה, בשביל לבדוק אם היא אפשרית על ידי בדיקת ה"אלכסון השמאלי" וה"אלכסון הימני" .
public boolean canMove (Board board) NOTE: this is not presented correctly in the UML, because of updating it.	פעולה האחראית לבדוק אם ל RedPiece ישנם מהלכים לעשות או לא. מחזירה אמת אם הRedPiece יכול לזוז, אחרת מחזירה שקר.
protected void updateBoardArray(Board board, int endX, int endY) NOTE: this is not presented correctly in the UML, because of updating it.	הפעולה אחראית לעדכון הboardArray לאחר תזוזה של אבן משחק אדומה.
private boolean isLeftDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating it.	פעולה הבודקת אם ה"אלכסון השמאלי" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isLeftJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating it.	פעולה הבודקת אם ה"אלכסון השמאלי - קפיצה" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isRightDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating it.	פעולה הבודקת אם ה"אלכסון הימיני" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isRightJumpDiagonalAvailable(Board board)	פעולה הבודקת אם ה"אלכסון הימיני - קפיצה" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.

NOTE: this is not presented correctly in the UML, because of updating it.	

BlackPiece

מחלקה זו מגדירה אבן משחק שחורה והיא יורשת מPiece.



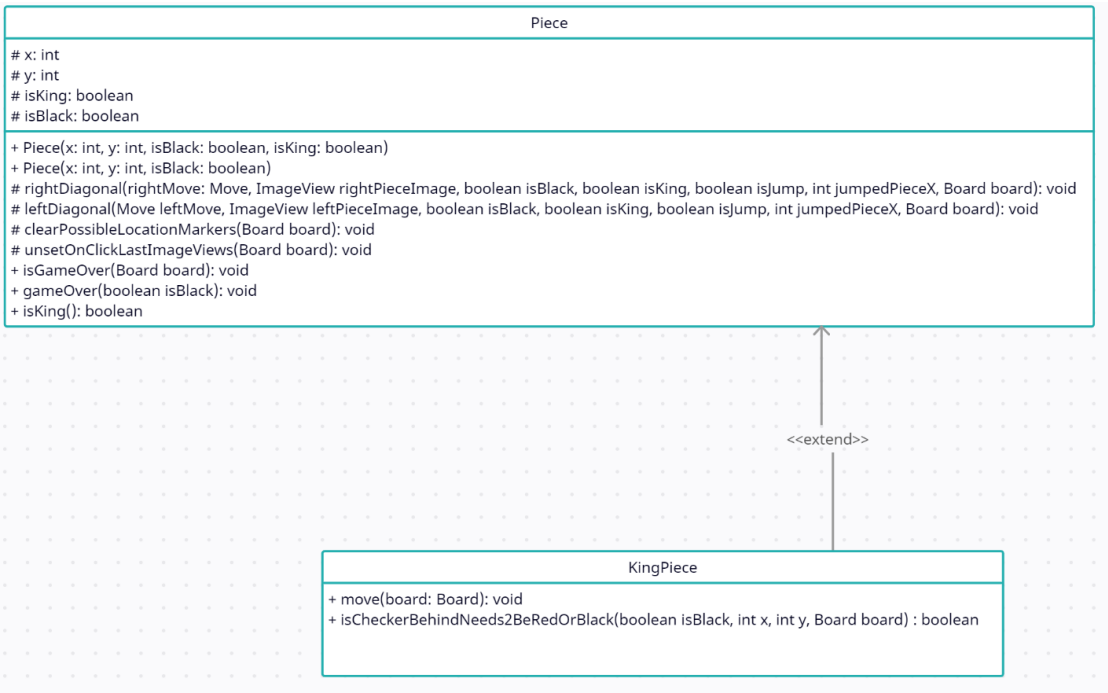
כותרת הפעולה מתוך java	תיאור הפעולה
public BlackPiece (int x, int y, TextView currentTurn) NOTE: this is not presented correctly in the UML, because of updating the class.	בנאי המחלקה. השדות x, y, isBlack, isKing הינם מורשים מהמחלקה Piece והבנאי קורא לבנאי האב ()super.
public void move (Board board)	תנועת אבן שחורה על גבי המסך. מתבצעת בדיקה לפני התנועה, בשביל לבדוק אם היא אפשרית על ידי בדיקת ה"אלכסון השמאלי" וה"אלכסון הימני" .
public boolean canMove (Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	פעולה האחראית לבדוק אם ל BlackPiece ישנם מהלכים לעשות או לא. מחזירה אמת אם הBlackPiece יכול לזוז, אחרת מחזירה שקר.
protected void updateBoardArray(Board board, int endX, int endY)	הפעולה אחראית לעדכון הboardArray לאחר תזוזה של אבן משחק שחורה.

NOTE: this is not presented correctly in the UML, because of updating the class.	
private boolean isLeftDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	פעולה הבודקת אם ה"אלכסון השמאלי" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isLeftJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	פעולה הבודקת אם ה"אלכסון השמאלי - קפיצה" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isRightDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	פעולה הבודקת אם ה"אלכסון הימיני" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.
private boolean isRightJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	פעולה הבודקת אם ה"אלכסון הימיני - קפיצה" פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.

KingPiece

מחלקה זו מגדירה אבן משחק שהיא מלך והיא יורשת מPiece.

<u>הערה</u>: שמה של הפעולה isCheckerBehindNeeds2BeRedOrBlack() הוחלף ל-canPieceBeEaten().



כותרת הפעולה מתוך java	תיאור הפעולה
public KingPiece (int x, int y, boolean isBlack, TextView currentTurn) NOTE: this is not presented correctly in the UML, because of updating the class.	בנאי המחלקה. השדות x, y, isBlack, הינם מורשים מהמחלקה Piece והבנאי קורא לבנאי האב super()..
public void move (Board board)	תנועת אבן שהיא מלך על גבי המסך. תנועת אבן זו היא מיוחדת, מכיוון שהיא שילוב התנועה של אבן שחורה ואבן אדומה.
private Boolean canPieceBeEaten (int x, int y, Board board) NOTE: this is not presented correctly in the UML, because of updating the class.	הפונקציה נועדה לבדוק אם אבן המשחק שבמיקום x, y יכולה להיאכל על ידי האבן הנוכחית (שצבעה שמור בשדה isBlack). לדוגמה, אם אבן המשחק הינה שחורה, אז האבן שאמורה להיאכל חייבת להיות אבון אדומה. לכן הפונקציה הזו בודקת זאת על פי הצבע של האבן הנוכחית. אם isBlack הוא true, הפונקציה תחזיר אמת אם יש במיקום x, y אבן אדומה (לפי board), אחרת תחזיר false.

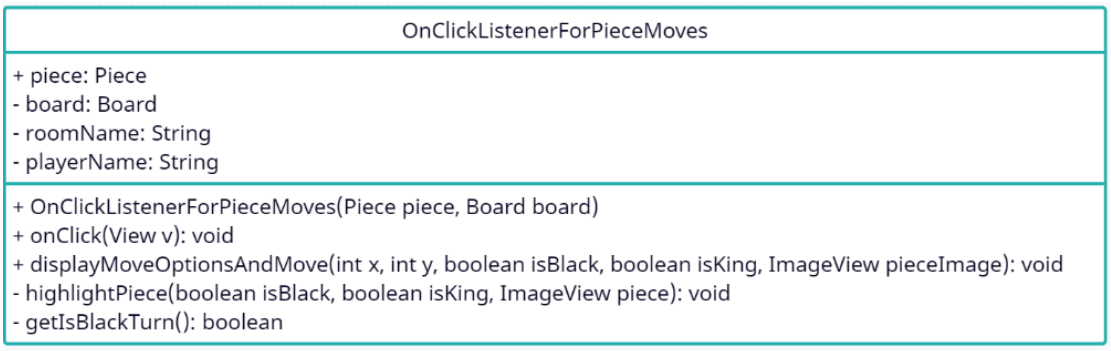
אם isBlack הוא false, הפונקציה תחזיר אמת אם יש במיקום x, y אבן שחורה (לפי board), אחרת תחזיר false.	
פעולה האחראית לבדוק אם ל KingPiece ישנם מהלכים לעשות או לא. מחזירה אמת אם הKingPiece יכול לזוז, אחרת מחזירה שקר.	public boolean canMove (Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
הפעולה אחראית לעדכון הboardArray לאחר תזוזה של אבן משחק המוגדרת כ-"מלך".	protected void updateBoardArray(Board board, int endX, int endY) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון השמאלי" של אבן שחורה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isBlackLeftDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון הימיני" של אבן שחורה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isBlackRightDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון השמאלי" של אבן אדומה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isRedLeftDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון הימיני" של אבן אדומה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isRedRightDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון השמאלי - קפיצה" של אבן שחורה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isBlackLeftJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון הימיני - קפיצה" של אבן שחורה פנוי	private boolean isBlackRightJumpDiagonalAvailable(Board board)

לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון השמאלי - קפיצה" של אבן אדומה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isRedLeftJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.
פעולה הבודקת אם ה"אלכסון ימיני - קפיצה" של אבן אדומה פנוי לתנועה. הפעולה מחזירה אמת אם כן, אחרת שקר.	private boolean isRedRightJumpDiagonalAvailable(Board board) NOTE: this is not presented correctly in the UML, because of updating the class.

OnClickListenerForPieceMoves

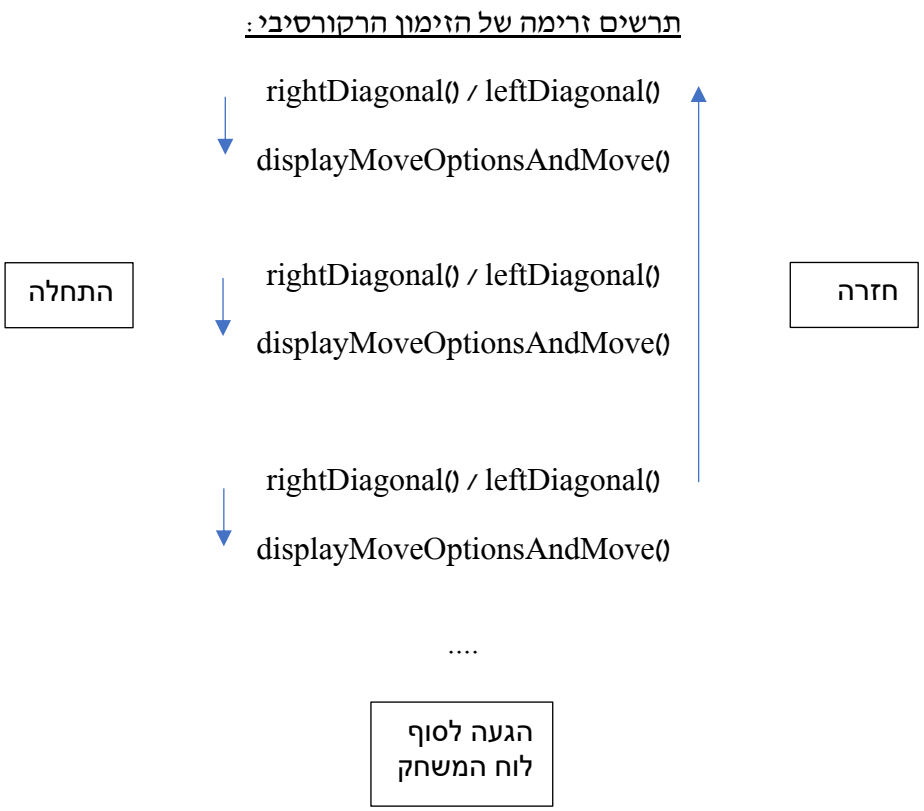
מחלקה זה אחראית על כל תנועות השחקנים (כאשר אבן כלשהי נלחצת במסך, הבנאי של המחלקה רץ). וזה מאחר שכל אבן משחק היא למעשה תמונה וב-Game Activity, הפונקציה setOnClickForPieces נקראת ושמה onClickListener עבור כל אבן משחק למחלקה הזאת.

הערה: הפונקציה getIsBlackTurn() הועברה למחלקה DBUtils.



תיאור הפעולה	כותרת הפעולה מתוך java
בנאי המחלקה. מאתחל את piece, ואת board.	public OnClickListenerForPieceMoves (Piece piece, Board board)
דריסת הפונקציה onClick בשביל מימוש הlistener. הפונקציה קוראת ל displayMoveOptionsAndMove.	@Override public void onClick(View v)
מטרת הפונקציה היא להציג את התנועות האפשריות בעת לחיצה על אבן משחק ולבצע תנועה אם המשתמש בחר בה.	public void displayMoveOptionsAndMove (int x, int y, boolean isBlack, boolean isKing, ImageView pieceImage)

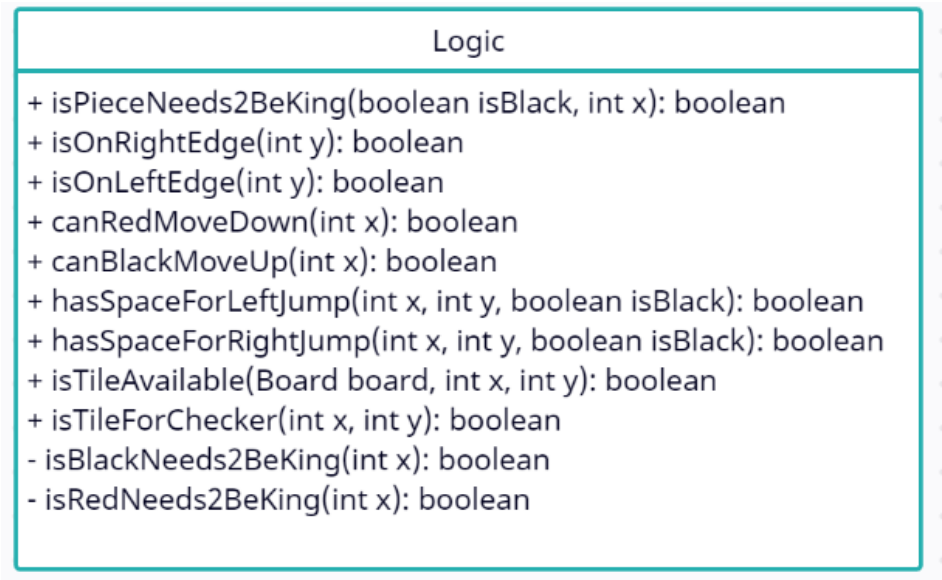
יש בפונקציה הזו זימון רקורסיבי ב rightDiagonal() וב leftDiagonal(), תרשים זרימה מוצג למטה.	
פונקציה האחראית על להדגיש אבן משחק כלשהי בעת לחיצה עליה.	private void highlightPiece (boolean isBlack, boolean isKing, ImageView piece)



פירוט מחלקות עזר

Logic

המחלקה מורכבת מפונקציות סטטיות שמטרתה לבדוק את הלוגיקה של המשחק (בהתאם לחוקי המשחק שצוינו במדריך למשתמש). מחלקה זו משמשת את המחלקה OnClickListenerForPieceMoves בבדיקת התנועות.



איור 10. תרשים UML – מחלקת Logic

תיאור הפעולה	כותרת הפעולה מתוך java
קוראת לפעולות isBlackNeeds2BeKing() ו isRedNeeds2BeKing(x) בהתאם לצבע האבן.	public static boolean isPieceNeeds2BeKing (boolean isBlack, int x)
אם ערך ה-y נמצא בקצה לוח המשחק הימני.	public static boolean isOnRightEdge (int y)
אם ערך ה-y נמצא בקצה לוח המשחק השמאלי.	public static boolean isOnLeftEdge (int y)
אם השחקן האדום ספציפית עם האבן במיקום ה-x הגיע לסוף לוח המשחק.	public static boolean canRedMoveDown (int x)
אם השחקן השחור ספציפית עם האבן במיקום ה-x הגיע לסוף לוח המשחק.	public static boolean canBlackMoveUp (int x)
אם המיקום x, y בלוח המשחק לא בקצה לוח המשחק ויש מקום לאכילה בצד שמאל – בהתאם לצבע השחקן.	public static boolean hasSpaceForLeftJump (int x, int y, boolean isBlack)
אם המיקום x, y בלוח המשחק לא בקצה לוח המשחק ויש מקום לאכילה בצד ימין – בהתאם לצבע השחקן.	public static boolean hasSpaceForRightJump (int x, int y, boolean isBlack)
אם המיקום x, y בלוח המשחק board לא תפוס על ידי שחקן אחר.	public static boolean isTileAvailable (Board board, int x, int y)

אם על המשבצת במיקום x, y יכולה להיות אבן משחק כלשהי.	public static boolean isTileForChecker (int x, int y)

DBUtils

מחלקה המורכבת מפונקציות סטטיות שמטרתה לעזור בכל הנלווה להתעסקות עם הdatabase.

מחלקה זו משמשת את כל המחלקות שצריכות אותה.

```
DBUtils
+ addDataToDatabase(Map<String, Object> data, DocumentReference docRef): void
+ isHost(): boolean
+ isWinner(): boolean
+ getGuestUsername(): String
+ updateBlackTurnInDb(boolean blackTurn, CollectionReference gameplayRef): void
+ uploadPieceLocationToDb(Move move, boolean isJump, int jumpX, int jumpY, boolean isKing): void
+ updateListview(ArrayList<String> roomsList, ListView listView, Context appContext): void
+ deleteAllDocumentsInCollection(CollectionReference collectionReference): void
+ checkGameOver(Board board, boolean isBlackTurn): void
+ gameOver(boolean didBlackWin): void
+ getIsBlackTurn(): boolean
- listenForFinishMessage(DocumentReference gameUpdates, String roomNameBak): void
- removeRoom(String roomNameBak, DocumentSnapshot snapshot): void
```

תיאור הפעולה	כותרת הפעולה מתוך java
פעולה העוטפת את הפעולות ()upadet ו()set שנמצאות בAPI של Firestore. הפעולה מעלה את הdata שקיבלה כפרמטר למיקום docRef בdatabase.	public static void addDataToDatabase (Map<String, Object> data, DocumentReference docRef)
הפעולה בודקת אם המשתמש הנוכחי הינו המנהל של החדר. הפעולה מחזירה אמת אם המשתמש הנוכחי הוא מנהל החדר, אחרת שקר.	public static boolean isHost()
הפעולה בודקת אם המשתמש הנוכחי הוא מנצח המשחק. הפעולה מחזירה אמת אם המשתמש הנוכחי הוא אכן מנצח המשחק (בעזרת הפרמטר nameOfWinner), אחרת שקר.	public static boolean isWinner (String nameOfWinner)
הפעולה אחראית להחזיר את שם המשתמש של האורח. הפעולה מחזירה את שמו של האורח בString.	public static String getGuestUsername ()
הפעולה מעדכנת את השדה "isBlackTurn" שנמצא בgameUpdates בתוך הdatabase עם הפרמטר blackTurn.	public static void updateBlackTurnInDb (boolean blackTurn)

public static void uploadPieceLocationToDb (Move move, boolean isJump, int jumpX, int jumpY, boolean isKing)	הפעולה מעלה לdatabase את המיקום של הPiece שזז על לוח המשחק, בכדי שהשחקן היריב יעדכן אצלו לוקאלית בלוח המשחק. פורמט ההעלאה מתואר בפירוט ב מיקום ההעלאה עבור המנהל הינו : hostMovesUpdates, ועבור האורח : guestMovesUpdates.
public static void updateListview (ArrayList<String> roomsList, ListView listView, Context appContext)	פעולה זו מעדכנת את הlistview של תצוגת השחקנים המחוברים לשרת.
public static void deleteAllDocumentsInCollection (CollectionReference collectionReference)	הפעולה מוחקת את כל הdocuments שקיימים בתוך הcollectionReference שקיבלה כפרמטר.
public static void checkGameOver (Board board, boolean isBlackTurn)	בדיקה אם המשחק אמור להיגמר או לא. פונקציה זו נקראת לאחר תזוזה של כל שחקן בשביל לבדוק אם אחרי תזוזה זו המשחק צריך להסתיים.
public static void gameOver (boolean didBlackWin)	פונקציה שאחראית לטפל בסיום משחק. הפונקציה מוחקת את כל הנתונים שנוצרו בdatabase בטרם המשחק, ומציגה AlertDialog עבור המשתמש עם כפתור חזרה לLobby.
private static void listenForFinishMessage (DocumentReference gameUpdates, String roomNameBak)	פונקציה שנקראת ממנצח המשחק, ואחראית להאזין להודעת ה"finish" מאת המפסיד של המשחק. כאשר ההודעה התקבלה, מנהל המשחק מוחק את כל החדר.
private static void removeRoom(String roomNameBak, DocumentSnapshot snapshot)	פונקציה האחראית למחוק את חדר המשחק (נקראת כאשר התקבלה ההודעה ב(listenForFinishMessage)
public static boolean getIsBlackTurn ()	פונקציה שמחזירה את ערך המשתנה isBlackTurn מה-database בנתיב : rooms/(roomName)/gameplay/gameUpdates

רפלקציה אישית

היעד ההתחלתי שהיה לי בפרויקט הוא לפתח משחק דמקה שכולם יוכלו להנות ממנו.
חשבתי איך אוכל להנות ממנו יחד עם כולם, וכך הגעתי לרעיון האונליין.
אני יכול להגיד בוודאות, שהיעד הושג.

למרות זאת, היו לי כמה קשיים במהלך הפרויקט שחוויתי, כגון למידת דברים חדשים, פיתוח משחק אונליין ראשון שלי, פיתוח אפליקציה לא פשוטה בAndroid Studio, וכו'.

אך עם הזמן, שמתי לב שלפתח משחק אונליין כמו דמקה, אינו כל כך מסובך, אלא עם מידת רצון מסוימת והתמדה, ניתן לפתח את המשחק אפילו בהנאה.

יתרה מזאת, למדתי המון על עצמי במהלך הפרויקט. לדוגמה, הבנתי את החוזקות שבי כמו אחריות, התמדה במטרה, רצינות, ועוד. אבל הכי חשוב - גיליתי שאני יודע לפתח משחק אונליין.

גיליתי שאני מתכנת טוב יותר ממה שחשבתי ולכן אני גאה בדרך שעברתי.

לא היו הרבה אנשים שעזרו לי בפרויקט חוץ מכמה שאלות שהיו לי אל המורה שלי בכיתה.

בנוסף, אני בוודאי אקח איתי כלים להמשך מכאן. כגון איך לכתוב אפליקציות ב Android Studio, איך לעבוד נכון עם Activities, ניהול הקוד בצורה טובה, וכו'.

אך בטוח אני אזכור את הmindset של פיתוח משחק אונליין כלשהו, ולהבא אדע לפתח את זה בצורה קלה יותר.

פיתוח עתידי עבור האפליקציה:

- הוספת האפשרות שהשחקן היריב יוכל להיכנע.
- הוספת האפשרות להצעת תיקו בין שני השחקנים.
- הוספה בהגדרות שיהיה אפשר לשנות את הthemes של המשחק.
- מסך הניקוד – Scores, שיכיל את רשומות הניצחונות וההפסדים של השחקן.

ביבליוגרפיה

אתרים

Android onDestroy() not called

remove firebase listener

activity

alertDialog

lifecycle

force listview item to be clicked

firebase always reading from cache

delete all subdocuments

get datasnapshot firestore

נספחים

BlackPiece.java

```
package com.example.checkers;

import static com.example.checkers.OnClickListenerForPieceMoves.lastUsedImageViews;

import android.widget.ImageView;
import android.widget.TextView;

/**
 * This class defines a black piece on the board.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class BlackPiece extends Piece {

    public BlackPiece(int x, int y, TextView currentTurn) {
        super(x, y, true, false, currentTurn);
    }

    /**
     * Check if the piece can move or not.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if black piece can move, false otherwise.
     */
    @Override
    public boolean canMove(Board board) {
        boolean left = isLeftDiagonalAvailable(board);
        boolean leftJump = isLeftJumpDiagonalAvailable(board);
        boolean right = isRightDiagonalAvailable(board);
        boolean rightJump = isRightJumpDiagonalAvailable(board);
```

```
    return left || leftJump || right || rightJump;
}

/**
 * Update the new piece in the board to be an object of BlackPiece.
 *
 * @param board The Board object that holds the current state of the game.
 * @param endX  The end X cord of the move.
 * @param endY  The end Y cord of the move.
 */
@Override
protected void updateBoardArray(Board board, int endX, int endY) {
    board.getBoardArray()[endX][endY] = new BlackPiece(endX, endY, currentTurn);
}

/**
 * Move the piece according to black logic.
 *
 * @param board The Board object that holds the current state of the game.
 */
public void move(Board board) {
    /* -------------------------- left diagonal -------------------------- */
    if (isLeftDiagonalAvailable(board)) {
        ImageView leftPieceImage = GameActivity.imageViewsTiles[x - 1][y - 1];
        lastUsedImageViews[0] = leftPieceImage;
        Move leftMove = new Move(x, y, x - 1, y - 1);
        leftDiagonal(leftMove, leftPieceImage, true, false, 0, board);
    }

    /* -------------------------- left-JUMP diagonal -------------------------- */
    if (isLeftJumpDiagonalAvailable(board)) {
        ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x - 2][y - 2];
        lastUsedImageViews[1] = leftJumpPieceImage;
        Move leftJumpMove = new Move(x, y, x - 2, y - 2);
```

```
        leftDiagonal(leftJumpMove, leftJumpPieceImage, true, true, x - 1, board);
    }

    /* ------------------------- right diagonal ------------------------- */
    if (isRightDiagonalAvailable(board)) {
        Move rightMove = new Move(x, y, x - 1, y + 1);
        ImageView rightPieceImage = GameActivity.imageViewsTiles[x - 1][y + 1];
        lastUsedImageViews[2] = rightPieceImage;
        rightDiagonal(rightMove, rightPieceImage, true, false, 0, board);
    }

    /* ------------------------- right-JUMP diagonal ------------------------- */
    if (isRightJumpDiagonalAvailable(board)) {
        ImageView rightJumpPieceImage = GameActivity.imageViewsTiles[x - 2][y + 2];
        lastUsedImageViews[3] = rightJumpPieceImage;
        Move rightJumpMove = new Move(x, y, x - 2, y + 2);
        rightDiagonal(rightJumpMove, rightJumpPieceImage, true, true, x - 1, board);
    }
}

/**
 * Check if left diagonal is available.
 *
 * @param board The Board object that holds the current state of the game.
 * @return true if diagonal is available, false otherwise.
 */
private boolean isLeftDiagonalAvailable(Board board) {
    return (Logic.canBlackMoveUp(x) && !Logic.isOnLeftEdge(y) &&
Logic.isTileAvailable(board, x - 1, y - 1) /* left tile */);
}

/**
 * Check if left-jump diagonal is available.
 *
```

```
    * @param board The Board object that holds the current state of the game.
    * @return true if diagonal is available, false otherwise.
    */
   private boolean isLeftJumpDiagonalAvailable(Board board) {
      return ((Logic.hasSpaceForLeftJump(x, y, true) && Logic.isTileAvailable(board, x - 2, y -
2) && !Logic.isTileAvailable(board, x - 1, y - 1) && !board.getBoardArray()[x - 1][y -
1].isBlack()));
   }

   /**
    * Check if right diagonal is available.
    *
    * @param board The Board object that holds the current state of the game.
    * @return true if diagonal is available, false otherwise.
    */
   private boolean isRightDiagonalAvailable(Board board) {
      return (Logic.canBlackMoveUp(x) && !Logic.isOnRightEdge(y) &&
Logic.isTileAvailable(board, x - 1, y + 1) /* right tile */);
   }

   /**
    * Check if right-jump diagonal is available.
    *
    * @param board The Board object that holds the current state of the game.
    * @return true if diagonal is available, false otherwise.
    */
   private boolean isRightJumpDiagonalAvailable(Board board) {
      return (Logic.hasSpaceForRightJump(x, y, true) && Logic.isTileAvailable(board, x - 2, y +
2) && !Logic.isTileAvailable(board, x - 1, y + 1) && !board.getBoardArray()[x - 1][y +
1].isBlack());
   }


}
```

Board.java

```
package com.example.checkers;

/**
 * This class defines the board of the game.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class Board {

   public static final int SIZE = 8; // Board must be square-shaped (default is 8x8)
   /**
    * The actual board that is a 2D array of Piece object.
    */
   private Piece[][] boardArray; // 2D board

   public Board() {
      this.boardArray = new Piece[Board.SIZE][Board.SIZE];
   }


   public Piece[][] getBoardArray() {
      return this.boardArray;
   }

   public void setBoardArray(Piece[][] boardArray) {
      this.boardArray = boardArray;
   }


}
```

}

DBUtils

```
package com.example.checkers;


import static com.example.checkers.GameActivity.gameOverListener;
import static com.example.checkers.GameActivity.guestMovesUpdatesListener;
import static com.example.checkers.GameActivity.hostMovesUpdatesListener;
import static com.example.checkers.LobbyActivity.ROOMSPATH;
import static com.example.checkers.LobbyActivity.playerName;
import static com.example.checkers.LobbyActivity.roomListener;
import static com.example.checkers.LobbyActivity.roomName;
import static com.example.checkers.LobbyActivity.roomRef;
import static com.example.checkers.LobbyActivity.roomsUpdaterViewListener;
import static com.example.checkers.OnClickListenerForPieceMoves.appContext;
import static com.example.checkers.OnClickListenerForPieceMoves.gameplayRef;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.util.Log;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.Task;
```

```
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * This class contains static helper functions for a database API.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class DBUtils {
   public static final String TAG = "DatabaseUtils";

   /**
    * wrapper function for update and set functions in the Firestore API.
    *
    * @param data   The data to be uploaded to the database - type of a Map
    * @param docRef The Document Reference to the location of the uploaded data.
    */
   public static void addDataToDatabase(Map<String, Object> data, DocumentReference
docRef) {
      docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
         @Override
```

```
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()) {
                DocumentSnapshot document = task.getResult();
                if (document.exists()) {
                    docRef.update(data);
                } else {
                    docRef.set(data);
                }
            } else {
                Log.d(TAG, "Failed with: ", task.getException());
            }
        }
    });
}

/**
 * Checks if local playerName is equal to the host name. This is done by comparing
playerName and roomName.
 *
 * @return True if the player is the host of the room.
 */
public static boolean isHost() {
    return playerName.equals(roomName);
}

/**
 * Checks if the winner is the current player. This is done by comparing playerName and
nameOfWinner.
 *
 * @param nameOfWinner The name of the winner.
 * @return True if playerName is the winner of the game, else otherwise.
 */
public static boolean isWinner(String nameOfWinner) {
    return playerName.equals(nameOfWinner);
```

```
    }

    /**
     * Gets the "guest" field in the database which is saved in the current room (pointed by
roomRef). returns "*GUEST*" if it couldn't retrieve the field (almost never happens).
     *
     * @return The field "guest", which is saved in the database in roomRef location (as a
String).
     */
    public static String getGuestUsername() {
        Task<DocumentSnapshot> getGuest = roomRef.get();
        while (!getGuest.isComplete()) {
            System.out.println("waiting for guestUsername");
        }
        if (getGuest.isSuccessful()) {
            DocumentSnapshot guestUsernameDoc = getGuest.getResult();
            return (String) guestUsernameDoc.get("guest");
        } else
            Log.d(TAG, "Error getting document: ", getGuest.getException());

        return "*GUEST*"; // couldn't get the guest username
    }

    /**
     * Uploads the given blackTurn parameter to the location pointed by gameplayRef.
     *
     * @param blackTurn The current turn to update in the database.
     */
    public static void updateBlackTurnInDb(boolean blackTurn) {
        Map<String, Object> gameUpdates = new HashMap<>();
        gameUpdates.put("isBlackTurn", blackTurn);
        addDataToDatabase(gameUpdates, gameplayRef.document("gameUpdates"));
    }
```

```
/**
 * Uploads the piece location based on the format:
 * Name    | startAxis -----  endAxis  -----   isKing    -----   isJump    -----   jumpedAxis |
 * Value   |  "X-Y"   -----   "X-Y"   -----   True/False -----   True/False  -----    "X-Y"   |
 *
 * @param move   The Move object that represents the move that was made.
 * @param isJump A boolean indicating if a jump has occurred.
 * @param jumpX  If there was a jump (based on isJump), this will contain the X cord of it.
 * @param jumpY  If there was a jump (based on isJump), this will contain the Y cord of it.
 * @param isKing A boolean indicating whether this piece is a king piece or not.
 */
public static void uploadPieceLocationToDb(Move move, boolean isJump, int jumpX, int
jumpY, boolean isKing) {
    DocumentReference documentReference;
    if (isHost())
        documentReference = gameplayRef.document("hostMovesUpdates"); // for host
updates
    else
        documentReference = gameplayRef.document("guestMovesUpdates"); // for guest
updates
    Map<String, Object> updates = new HashMap<>();
    String startAxis = move.getStartX() + "-" + move.getStartY();
    String endAxis = move.getEndX() + "-" + move.getEndY();
    updates.put("startAxis", startAxis);
    updates.put("endAxis", endAxis);
    updates.put("isKing", isKing);
    updates.put("isJump", isJump);
    if (isJump)
        updates.put("jumpedAxis", jumpX + "-" + jumpY);
    addDataToDatabase(updates, documentReference);
}

/**
 * Updates the list view when a player joins the server.
```

```
 *
 * @param roomsList  A reference to the roomsList object in Lobby activity.
 * @param listView   A reference to the listView object in Lobby activity.
 * @param appContext The Application Context.
 */
 public static void updateListview(ArrayList<String> roomsList, ListView listView, Context
appContext) {
    CollectionReference roomsRef =
FirebaseFirestore.getInstance().collection(ROOMSPATH);
    roomsUpdaterViewListener = roomsRef.addSnapshotListener(new
EventListener<QuerySnapshot>() {
        @Override
        public void onEvent(@Nullable QuerySnapshot value, @Nullable
FirebaseFirestoreException error) {
            if (error != null) {
                Log.w(TAG, "Listen failed.", error);
                return;
            }
            roomsList.clear();
            for (QueryDocumentSnapshot doc : value) {
                if (!doc.getId().equals(playerName)) {
                    roomsList.add(doc.getId());
                }

            }
            ArrayAdapter<String> adapter = new ArrayAdapter<>(appContext,
android.R.layout.simple_list_item_1, roomsList);
            listView.setAdapter(adapter);
        }
    });
 }

 /**
  * Deletes all documents in a given collection location.
```

```
    * This is done by looping through the entire collection and removing each document by
getting its reference.
    *
    * @param collectionReference The Reference to the collection to be deleted.
    */
   public static void deleteAllDocumentsInCollection(CollectionReference
collectionReference) {
      collectionReference.get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
         @Override
         public void onComplete(@NonNull Task<QuerySnapshot> task) {
            List<DocumentSnapshot> myListOfDocuments = task.getResult().getDocuments();
            for (DocumentSnapshot doc : myListOfDocuments) {
               doc.getReference().delete().addOnFailureListener(new OnFailureListener() {
                  @Override
                  public void onFailure(@NonNull Exception e) {
                     Log.d(TAG, "ERROR!!! Failed to delete gameplay documents!", e);
                  }
               });
            }

         }
      });
   }

   /**
    * Checks if the game is over by one of the following conditions:
    * if a player either has no more checkers or cannot make a move on their turn.
    *
    * @param board       The Board object that contains the current state of the game.
    * @param isBlackTurn A boolean indicating the current turn.
    */
   public static void checkGameOver(Board board, boolean isBlackTurn) {
      int redPieces = 0;
```

```
    int blackPieces = 0;
    boolean canBlackMove = false;
    boolean canRedMove = false;

    for (int i = 0; i < Board.SIZE; i++)
        for (int j = 0; j < Board.SIZE; j++) {
            if (board.getBoardArray()[i][j] != null) {
                if (board.getBoardArray()[i][j].isBlack()) {
                    blackPieces++;
                    if (board.getBoardArray()[i][j].canMove(board)) {
                        canBlackMove = true;
                    }
                } else {
                    redPieces++;
                    if (board.getBoardArray()[i][j].canMove(board)) {
                        canRedMove = true;
                    }
                }

            }
        }
    // black won
    if (redPieces == 0 || (!canRedMove && !isBlackTurn)) {
        // show locally on black's phone that he won
        gameOver(true);
    }

    // red won
    else if (blackPieces == 0 || (!canBlackMove && isBlackTurn)) {
        // show locally on red's phone that he won
        gameOver(false);
    }

}
```

```
/**
 * Handles the game over event by showing a dialog of the winner and a "Return to the lobby" button, as well as removing the game room.
 * The removal of the room is done by the following steps:
 * 1) The winner starts to listen for "finish" field in gameUpdates (Boolean variable).
 * 2) The loser does all his clean-ups (such as setting the roomRef variable and changing roomName), and then uploads "finish" to gameUpdates location.
 * 3) The winner gets the "finish" message from the loser and then removes the room completely (by calling deleteAllDocumentsInCollection())
 *
 * @param didBlackWin A boolean indicating the winner of the game.
 */
public static void gameOver(boolean didBlackWin) {

    boolean host = isHost();
    // backup of the roomName. useful when a guest wins and does his cleanup, which includes resetting roomName to his playerName, thus losing the actual room name.
    String roomNameBak = roomName;
    String winner;
    DocumentReference gameUpdates = FirebaseFirestore.getInstance().collection(LobbyActivity.ROOMSPATH).document(roomName).collection("gameplay").document("gameUpdates");

    AlertDialog.Builder builder = new AlertDialog.Builder(appContext, AlertDialog.THEME_HOLO_LIGHT);
    builder.setCancelable(false);
    builder.setTitle("Game is Over!");
    builder.setPositiveButton("Return Back To The Lobby", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            appContext.startActivity(new Intent(appContext, LobbyActivity.class));
            ((Activity) appContext).finish(); // finish GameActivity
```

```
        }
    });

    if (didBlackWin) {
        // show popup that the host won (roomName = hostname)
        String hostUsername = roomName;
        winner = hostUsername; // winner is host
        builder.setMessage(hostUsername + " has won the game! he is probably better.");
    } else {
        // show popup that the guest won (getGuestUsername())
        String guestUsername;
        if (host) // on the host phone (he doesn't have the guest's username, so he has to get
it from db
            guestUsername = getGuestUsername();
        else // on the guest phone (the local username is stored in playerName)
            guestUsername = playerName;

        winner = guestUsername; // winner is guest
        builder.setMessage(guestUsername + " has won the game! he is probably better.");
    }
    AlertDialog gameFinishedDialog;
    gameFinishedDialog = builder.create();
    gameFinishedDialog.show();

    if (isWinner(winner)) {
        // start listening for "finish" message from the loser.
        listenForFinishMessage(gameUpdates, roomNameBak);
    }


    if (!host) // for the guest
    {

        // change roomName back to guest's name
```

```
            roomName = playerName;
            roomRef =
FirebaseFirestore.getInstance().collection(ROOMSPATH).document(roomName);

            // remove room listener for guest
            roomListener.remove();
        }

        if (!isWinner(winner)) // if i'm the loser
        {
            // upload "finish" message
            Map<String, Object> updates = new HashMap<>();
            updates.put("finish", true);
            addDataToDatabase(updates, gameUpdates);
        }

        // clean-up stuff
        if (hostMovesUpdatesListener != null)
            hostMovesUpdatesListener.remove();
        if (guestMovesUpdatesListener != null)
            guestMovesUpdatesListener.remove();
    }

    /**
     * Listens for "finish" field in gameUpdates location, and removes the room when receives
it.
     *
     * @param gameUpdates The Document Reference to the gameUpdates location in the
database.
     * @param roomNameBak The backup String of the original roomName variable (useful
when the guest is the winner).
     */
    private static void listenForFinishMessage(DocumentReference gameUpdates, String
roomNameBak) {
```

```
    gameOverListener = gameUpdates.addSnapshotListener(new
EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException error) {
            if (error != null) {
                Log.w(TAG, "Listen failed.", error);
                return;
            }
            if (snapshot != null && snapshot.exists()) {
                removeRoom(roomNameBak, snapshot);
            }
        }
    });
  }

  /**
   * Removes the room completely if snapshot contains the "finish" field.
   *
   * @param roomNameBak The backup String of the original roomName variable (useful
when the guest is the winner).
   * @param snapshot    The DocumentSnapshot object that might contain the "finish"
message, which we got in the onEvent in listenForFinishMessage().
   */
  private static void removeRoom(String roomNameBak, DocumentSnapshot snapshot) {
    Boolean isFinish = (Boolean) snapshot.get("finish");
    if (isFinish != null) // loser is finished, remove the room.
    {
        Log.d(TAG, "GOT finish MESSAGE, REMOVING ROOM");
        Map<String, Object> updates = new HashMap<>();
        updates.put("guest", FieldValue.delete()); // mark "guest" field as deletable on the
database (remove it)
        updates.put("isInGame", false); // update isInGame to false
```

```
        addDataToDatabase(updates,
FirebaseFirestore.getInstance().collection(ROOMSPATH).document(roomNameBak));

        deleteAllDocumentsInCollection(gameplayRef); // remove all gameplay documents
that the host and guest created (cleaning-up)
    }
  }

  /**
   * Gets the "isBlackTurn" field in the gameUpdates location pointed by gameplayRef.
   *
   * @return The value of isBlackTurn in the database (a Boolean variable)
   */
  public static boolean getIsBlackTurn() {
    Task<DocumentSnapshot> getTurn = gameplayRef.document("gameUpdates").get();
    while (!getTurn.isComplete()) {
      System.out.println("waiting for getIsBlackTurn");
    }
    if (getTurn.isSuccessful()) {
      DocumentSnapshot isBlackTurnResult = getTurn.getResult();
      Boolean val = (Boolean) isBlackTurnResult.get("isBlackTurn");
      if (val != null)
        return val;
    }
    Log.d(TAG, "Error getting document: ", getTurn.getException());
    throw new IllegalStateException("couldn't get isBlackTurn from db");
  }
}
```

GameActivity

```
package com.example.checkers;
```

```
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.widget.ImageView;
import android.widget.TextView;

import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.ListenerRegistration;

import static com.example.checkers.DBUtils.addDataToDatabase;
import static com.example.checkers.DBUtils.checkGameOver;
import static com.example.checkers.DBUtils.isHost;
import static com.example.checkers.LobbyActivity.roomName;
import static com.example.checkers.LobbyActivity.roomRef;

import java.util.HashMap;
import java.util.Map;

/**
 * This class is responsible to handle game events, such as piece moves, game ending and
more.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class GameActivity extends AppCompatActivity {

   /**
```

```
 * all the squares which contain the actual pieces (reference from the xml).
 */
public static final ImageView[][] imageViewsTiles = new
ImageView[Board.SIZE][Board.SIZE];
public static final String TAG = "GameActivity";
public static ListenerRegistration guestMovesUpdatesListener;
public static ListenerRegistration hostMovesUpdatesListener;
public static ListenerRegistration gameOverListener;
public TextView currentTurn;
public Board board;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_game);

    currentTurn = findViewById(R.id.currentTurn);

    initImageViews();

    board = new Board();
    initBoardAndDrawPieces(); // init board as well as drawing the black and red pieces on it

    // set initial value for isBlackTurn (host starts as black)
    FirebaseFirestore fStore = FirebaseFirestore.getInstance();
    DocumentReference gameUpdatesRef =
fStore.collection(LobbyActivity.ROOMSPATH).document(roomName).collection("gameplay")
.document("gameUpdates");
    Map<String, Object> data = new HashMap<>();
    data.put("isBlackTurn", true); // sets isBlackTurn to true, so that Black player starts first.
    addDataToDatabase(data, gameUpdatesRef);

    if (isHost()) {
        currentTurn.setText(R.string.your_turn);
```

```
        // set a listener for red's moves (guest moves) and move the red pieces accordingly
        DocumentReference guestMovesUpdatesRef =
roomRef.collection("gameplay").document("guestMovesUpdates");
        guestMovesUpdatesListener = listenDBForPieceMoves(guestMovesUpdatesRef, false);
    } else {
        currentTurn.setText(R.string.not_your_turn);

        // set a listener for black's moves (host pieces) and move the black pieces accordingly
        DocumentReference hostMovesUpdatesRef =
roomRef.collection("gameplay").document("hostMovesUpdates");
        hostMovesUpdatesListener = listenDBForPieceMoves(hostMovesUpdatesRef, true);
    }

    setOnClickForPieces();



  }

  /**
   * Set onClick listeners for all pieces with OnClickListenerForPieceMoves.
   */
  public void setOnClickForPieces() {
    for (int x = 0; x < Board.SIZE; x++) {
        for (int y = 0; y < Board.SIZE; y++) {
            Piece currPiece = board.getBoardArray()[x][y];
            if (currPiece != null) {
                imageViewsTiles[x][y].setOnClickListener(new
OnClickListenerForPieceMoves(currPiece, board, currentTurn));
            }

        }
    }
  }
```

```
/**
 * Responsible for initializing and drawing the pieces on the board
 */
public void initBoardAndDrawPieces() {
    for (int x = 0; x < Board.SIZE; x++) {
        for (int y = 0; y < Board.SIZE; y++) {
            // red pieces
            if (x <= 2 && Logic.isTileForChecker(x, y)) {
                imageViewsTiles[x][y].setImageResource(R.drawable.red_piece);
                board.getBoardArray()[x][y] = new RedPiece(x, y, currentTurn);
            }

            // black pieces
            if (x >= 5 && Logic.isTileForChecker(x, y)) {
                imageViewsTiles[x][y].setImageResource(R.drawable.black_piece);
                board.getBoardArray()[x][y] = new BlackPiece(x, y, currentTurn);
            }

        }
    }
}

/**
 * @param playerMovesUpdatesRef The DocumentReference to the player moves updates
location, e.g guestMovesUpdatesRef and hostMovesUpdatesRef
 * @param isPieceBlack        A boolean indicating if the current piece is black or not.
 * @return the ListenerRegistration object to be removed in the onStop() callback.
 */
// listen to playerMovesUpdatesRef in the host and in the guest
private ListenerRegistration listenDBForPieceMoves(DocumentReference
playerMovesUpdatesRef, boolean isPieceBlack) {

    return playerMovesUpdatesRef.addSnapshotListener(new
```

```
EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException error) {
            if (error != null) {
                Log.w(TAG, "Listen failed.", error);
                return;
            }
            if (snapshot != null && snapshot.exists()) {
                String endAxis = (String) snapshot.get("endAxis"); // parsing the axis in the
format: "X-Y"
                String startAxis = (String) snapshot.get("startAxis"); // parsing the axis in the
format: "X-Y"
                Boolean isJump = (Boolean) snapshot.get("isJump");
                Boolean isKingDb = (Boolean) snapshot.get("isKing");
                if (endAxis != null && startAxis != null && isKingDb != null) {
                    // parser
                    int startX = Integer.parseInt(startAxis.split("-")[0]);
                    int startY = Integer.parseInt(startAxis.split("-")[1]);
                    int endX = Integer.parseInt(endAxis.split("-")[0]);
                    int endY = Integer.parseInt(endAxis.split("-")[1]);
                    Move move = new Move(startX, startY, endX, endY);
                    move.perform(isPieceBlack, isKingDb);

                    // updating boardArray
                    board.getBoardArray()[endX][endY] = createAddedPiece(isPieceBlack, isKingDb,
endX, endY);
                    board.getBoardArray()[startX][startY] = null; // remove old piece

                    // marking the start position (for the user)

imageViewsTiles[startX][startY].setImageResource(R.drawable.possible_location_marker);

                    if (isJump != null) {
```

```
                    if (isJump) { // if true: there was a jump, remove the jumped piece
                        String jumpedAxis = (String) snapshot.get("jumpedAxis"); // // parsing the
axis in the format: "X-Y"
                        if (jumpedAxis != null) {
                            int jumpedX = Integer.parseInt(jumpedAxis.split("-")[0]);
                            int jumpedY = Integer.parseInt(jumpedAxis.split("-")[1]);


GameActivity.imageViewsTiles[jumpedX][jumpedY].setImageResource(android.R.color.trans
parent);
                            GameActivity.imageViewsTiles[jumpedX][jumpedY].setClickable(false);
                            board.getBoardArray()[jumpedX][jumpedY] = null;
                        } else
                            Log.d(TAG, "Couldn't get jumpedAxis");
                    }
                }
                currentTurn.setText(R.string.your_turn);
                checkGameOver(board, !isPieceBlack);
            }
        }
    }
  });
}

/**
 * @param isPieceBlack A boolean indicating if the current piece is black or not.
 * @param isKingDb     A boolean field from the database indicating if the piece is king or
not.
 * @param endX         An integer that holds the end X cord of the move.
 * @param endY         An integer that holds the end Y cord of the move.
 * @return The newly created piece to be added to the boardArray hold in the Board
object.
 */
private Piece createAddedPiece(boolean isPieceBlack, boolean isKingDb, int endX, int
```

```
endY) {
    Piece addedPiece;
    if (isKingDb)
        addedPiece = new KingPiece(endX, endY, isPieceBlack, currentTurn);
    else if (isPieceBlack)
        addedPiece = new BlackPiece(endX, endY, currentTurn);
    else
        addedPiece = new RedPiece(endX, endY, currentTurn);
    return addedPiece;
  }

  /**
   * Init every imageview from the xml, to move the pieces around the board.
   */
  private void initImageViews() {

    imageViewsTiles[0][1] = findViewById(R.id.circle01);
    imageViewsTiles[0][3] = findViewById(R.id.circle03);
    imageViewsTiles[0][5] = findViewById(R.id.circle05);
    imageViewsTiles[0][7] = findViewById(R.id.circle07);
    imageViewsTiles[1][0] = findViewById(R.id.circle10);
    imageViewsTiles[1][2] = findViewById(R.id.circle12);
    imageViewsTiles[1][4] = findViewById(R.id.circle14);
    imageViewsTiles[1][6] = findViewById(R.id.circle16);
    imageViewsTiles[2][1] = findViewById(R.id.circle21);
    imageViewsTiles[2][3] = findViewById(R.id.circle23);
    imageViewsTiles[2][5] = findViewById(R.id.circle25);
    imageViewsTiles[2][7] = findViewById(R.id.circle27);
    imageViewsTiles[3][0] = findViewById(R.id.circle30);
    imageViewsTiles[3][2] = findViewById(R.id.circle32);
    imageViewsTiles[3][4] = findViewById(R.id.circle34);
    imageViewsTiles[3][6] = findViewById(R.id.circle36);
    imageViewsTiles[4][1] = findViewById(R.id.circle41);
    imageViewsTiles[4][3] = findViewById(R.id.circle43);
```

```
        imageViewsTiles[4][5] = findViewById(R.id.circle45);
        imageViewsTiles[4][7] = findViewById(R.id.circle47);
        imageViewsTiles[5][0] = findViewById(R.id.circle50);
        imageViewsTiles[5][2] = findViewById(R.id.circle52);
        imageViewsTiles[5][4] = findViewById(R.id.circle54);
        imageViewsTiles[5][6] = findViewById(R.id.circle56);
        imageViewsTiles[6][1] = findViewById(R.id.circle61);
        imageViewsTiles[6][3] = findViewById(R.id.circle63);
        imageViewsTiles[6][5] = findViewById(R.id.circle65);
        imageViewsTiles[6][7] = findViewById(R.id.circle67);
        imageViewsTiles[7][0] = findViewById(R.id.circle70);
        imageViewsTiles[7][2] = findViewById(R.id.circle72);
        imageViewsTiles[7][4] = findViewById(R.id.circle74);
        imageViewsTiles[7][6] = findViewById(R.id.circle76);
    }

    @Override
    public void onBackPressed() {
        // back button is not allowed here.
    }

    /**
     * When the activity is in the "STOP" state (onStop() is called), do clean-ups by removing
the gameOverListener, as well as host and guest moves listener.
     */
    @Override
    public void onStop() {
        if (gameOverListener != null)
            gameOverListener.remove();
        if (hostMovesUpdatesListener != null)
            hostMovesUpdatesListener.remove();
        if (guestMovesUpdatesListener != null)
            guestMovesUpdatesListener.remove();
        super.onStop();
```

```
    }
}
```

KingPiece

```
package com.example.checkers;

import static com.example.checkers.OnClickListenerForPieceMoves.lastUsedImageViews;

import android.widget.ImageView;
import android.widget.TextView;

/**
 * This class defines a king piece.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class KingPiece extends Piece {

    public KingPiece(int x, int y, boolean isBlack, TextView currentTurn) {
        super(x, y, isBlack, true, currentTurn);
    }

    /**
     * Check if the piece can move or not.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if black piece can move, false otherwise.
     */
    @Override
    public boolean canMove(Board board) {

        boolean blackLeftDiagonal = isBlackLeftDiagonalAvailable(board);
        boolean blackRightDiagonal = isBlackRightDiagonalAvailable(board);
```

```
    boolean redLeftDiagonal = isRedLeftDiagonalAvailable(board);
    boolean redRightDiagonal = isRedRightDiagonalAvailable(board);

    boolean blackLeftJumpDiagonal = isBlackLeftJumpDiagonalAvailable(board);
    boolean blackRightJumpDiagonal = isBlackRightJumpDiagonalAvailable(board);
    boolean redLeftJumpDiagonal = isRedLeftJumpDiagonalAvailable(board);
    boolean redRightJumpDiagonal = isRedRightJumpDiagonalAvailable(board);

    return (blackLeftDiagonal || blackRightDiagonal || redLeftDiagonal || redRightDiagonal
||
            blackLeftJumpDiagonal || blackRightJumpDiagonal || redLeftJumpDiagonal ||
redRightJumpDiagonal);
  }


  /**
   * Update the new piece in the board to be an object of KingPiece.
   *
   * @param board The Board object that holds the current state of the game.
   * @param endX  The end X cord of the move.
   * @param endY  The end Y cord of the move.
   */
  @Override
  protected void updateBoardArray(Board board, int endX, int endY) {
    board.getBoardArray()[endX][endY] = new KingPiece(endX, endY, isBlack, currentTurn);
  }

  /**
   * Move according to king piece logic.
   *
   * @param board The Board object that holds the current state of the game.
   */
  public void move(Board board) {
    /* ------------------------- left diagonal BLACK ------------------------- */
```

```
if (isBlackLeftDiagonalAvailable(board)) {
   ImageView leftPieceImage = GameActivity.imageViewsTiles[x - 1][y - 1];
   lastUsedImageViews[0] = leftPieceImage;
   Move leftMove = new Move(x, y, x - 1, y - 1);
   leftDiagonal(leftMove, leftPieceImage, isBlack, false, 0, board);
}

/* ------------------------- right diagonal BLACK ------------------------- */
if (isBlackRightDiagonalAvailable(board)) {
   Move rightMove = new Move(x, y, x - 1, y + 1);
   ImageView rightPieceImage = GameActivity.imageViewsTiles[x - 1][y + 1];
   lastUsedImageViews[1] = rightPieceImage;
   rightDiagonal(rightMove, rightPieceImage, isBlack, false, 0, board);
}


/* ------------------------- left diagonal RED ------------------------- */
if (isRedLeftDiagonalAvailable(board)) {
   Move leftMove = new Move(x, y, x + 1, y - 1);
   ImageView leftPieceImage = GameActivity.imageViewsTiles[x + 1][y - 1];
   lastUsedImageViews[2] = leftPieceImage;
   leftDiagonal(leftMove, leftPieceImage, isBlack, false, 0, board);
}

/* ------------------------- right diagonal RED ------------------------- */
if (isRedRightDiagonalAvailable(board)) {
   Move rightMove = new Move(x, y, x + 1, y + 1);
   ImageView rightPieceImage = GameActivity.imageViewsTiles[x + 1][y + 1];
   lastUsedImageViews[3] = rightPieceImage;
   rightDiagonal(rightMove, rightPieceImage, isBlack, false, 0, board);
}


// JUMP - CHECKS
```

```
    /* -------------------------- left-JUMP diagonal BLACK -------------------------- */
    if (isBlackLeftJumpDiagonalAvailable(board)) {
        ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x - 2][y - 2];
        lastUsedImageViews[4] = leftJumpPieceImage;
        Move leftJumpMove = new Move(x, y, x - 2, y - 2);
        leftDiagonal(leftJumpMove, leftJumpPieceImage, isBlack, true, x - 1, board);
    }
    /* -------------------------- right-JUMP diagonal BLACK -------------------------- */
    if (isBlackRightJumpDiagonalAvailable(board)) {
        ImageView rightJumpPieceImage = GameActivity.imageViewsTiles[x - 2][y + 2];
        lastUsedImageViews[5] = rightJumpPieceImage;
        Move rightJumpMove = new Move(x, y, x - 2, y + 2);
        rightDiagonal(rightJumpMove, rightJumpPieceImage, isBlack, true, x - 1, board);
    }
    /* -------------------------- left-JUMP diagonal RED -------------------------- */
    if (isRedLeftJumpDiagonalAvailable(board)) {
        ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x + 2][y - 2];
        lastUsedImageViews[6] = leftJumpPieceImage;
        Move leftJumpMove = new Move(x, y, x + 2, y - 2);
        leftDiagonal(leftJumpMove, leftJumpPieceImage, isBlack, true, x + 1, board);
    }
    /* -------------------------- right-JUMP diagonal RED -------------------------- */
    if (isRedRightJumpDiagonalAvailable(board)) {
        ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x + 2][y + 2];
        lastUsedImageViews[7] = leftJumpPieceImage;
        Move leftJumpMove = new Move(x, y, x + 2, y + 2);
        rightDiagonal(leftJumpMove, leftJumpPieceImage, isBlack, true, x + 1, board);
    }
}

/**
 * Check if the black-left diagonal is available.
```

```
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if the diagonal is available, false otherwise.
     */
    private boolean isBlackLeftDiagonalAvailable(Board board) {
        return (Logic.canBlackMoveUp(x) && !Logic.isOnLeftEdge(y) &&
Logic.isTileAvailable(board, x - 1, y - 1) /* left tile */);
    }

    /**
     * Check if the black-right diagonal is available.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if the diagonal is available, false otherwise.
     */
    private boolean isBlackRightDiagonalAvailable(Board board) {
        return (Logic.canBlackMoveUp(x) && !Logic.isOnRightEdge(y) &&
Logic.isTileAvailable(board, x - 1, y + 1) /* right tile */);
    }

    /**
     * Check if the red-left diagonal is available.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if the diagonal is available, false otherwise.
     */
    private boolean isRedLeftDiagonalAvailable(Board board) {
        return (Logic.canRedMoveDown(x) && !Logic.isOnLeftEdge(y) &&
Logic.isTileAvailable(board, x + 1, y - 1) /* left tile */);
    }

    /**
     * Check if the red-right diagonal is available.
     *
```

```
 * @param board The Board object that holds the current state of the game.
 * @return true if the diagonal is available, false otherwise.
 */
private boolean isRedRightDiagonalAvailable(Board board) {
    return (Logic.canRedMoveDown(x) && !Logic.isOnRightEdge(y) &&
Logic.isTileAvailable(board, x + 1, y + 1) /* right tile */);
}


/**
 * Check if the black-left-jump diagonal is available.
 *
 * @param board The Board object that holds the current state of the game.
 * @return true if the diagonal is available, false otherwise.
 */
private boolean isBlackLeftJumpDiagonalAvailable(Board board) {
    return (Logic.hasSpaceForLeftJump(x, y, true) && Logic.isTileAvailable(board, x - 2, y - 2)
&& !Logic.isTileAvailable(board, x - 1, y - 1) && canPieceBeEaten(x - 1, y - 1, board) /* if the
piece to be eaten is black color (because we are red in this condition) */);
}

/**
 * Check if the black-right-jump diagonal is available.
 *
 * @param board The Board object that holds the current state of the game.
 * @return true if the diagonal is available, false otherwise.
 */
private boolean isBlackRightJumpDiagonalAvailable(Board board) {
    return (Logic.hasSpaceForRightJump(x, y, true) && Logic.isTileAvailable(board, x - 2, y +
2) && !Logic.isTileAvailable(board, x - 1, y + 1) && canPieceBeEaten(x - 1, y + 1, board) /* if
the piece to be eaten is black color (because we are red in this condition) */);
}

/**
```

```
 * Check if the red-left-jump diagonal is available.
 *
 * @param board The Board object that holds the current state of the game.
 * @return true if the diagonal is available, false otherwise.
 */
private boolean isRedLeftJumpDiagonalAvailable(Board board) {
    return (Logic.hasSpaceForLeftJump(x, y, false) && Logic.isTileAvailable(board, x + 2, y -
2) && !Logic.isTileAvailable(board, x + 1, y - 1) && canPieceBeEaten(x + 1, y - 1, board) /* if
the piece to be eaten is black color (because we are red in this condition) */);
}

/**
 * Check if the red-right-jump diagonal is available.
 *
 * @param board The Board object that holds the current state of the game.
 * @return true if the diagonal is available, false otherwise.
 */
private boolean isRedRightJumpDiagonalAvailable(Board board) {
    return (Logic.hasSpaceForRightJump(x, y, false) && Logic.isTileAvailable(board, x + 2, y +
2) && !Logic.isTileAvailable(board, x + 1, y + 1) && canPieceBeEaten(x + 1, y + 1, board) /* if
the piece to be eaten is black color (because we are red in this condition) */);
}

/**
 * Check if the piece at the location (x,y) in the board can be eatan.
 * Only in the eating-checks we do, we need to check for black that the eaten piece is really
a RED COLOR piece (so we can't eat our own color), and the same for a red piece.
 *
 * @param x     The x cord of the eatan piece
 * @param y     The y cord of the eatan piece
 * @param board The Board object that holds the current state of the game.
 * @return true if the piece can be eaten, false otherwise.
 */
private boolean canPieceBeEaten(int x, int y, Board board) {
```

```
    if (isBlack)
        return !board.getBoardArray()[x][y].isBlack(); // check if there is red piece in front of me
    return board.getBoardArray()[x][y].isBlack(); // else, check if there is black piece in front of me
  }
}
```

LobbyActivity

```
package com.example.checkers;

import static com.example.checkers.DBUtils.addDataToDatabase;
import static com.example.checkers.DBUtils.isHost;
import static com.example.checkers.DBUtils.getGuestUsername;
import static com.example.checkers.DBUtils.updateListview;
import static com.example.checkers.SettingsActivity.SETTINGS_PREFS;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.app.AlertDialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.VibrationEffect;
import android.os.Vibrator;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
```

```
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FieldValue;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.firestore.FirebaseFirestoreSettings;
import com.google.firebase.firestore.ListenerRegistration;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

/**
 * This class handles the interaction in the Lobby activity, such as sending a game invite.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class LobbyActivity extends AppCompatActivity {
   public static final String TAG = "WaitingRoom";
   public static final String ROOMSPATH = "rooms";
   public static final int PhoneNumOfDeveloper = 99999999;
   public Toolbar toolbar;
   public DrawerLayout drawer;
   public TextView mUsername;
   public TextView mEmail;

   public BroadcastReceiver broadcastReceiver;
   public static DocumentReference roomRef;
   public static ListenerRegistration roomListener;
   public ListenerRegistration hostUpdatesListener;
   public ListenerRegistration guestUpdatesListener;
   public static ListenerRegistration roomsUpdaterViewListener;
   public ListView listView;
   public static String playerName;
   public static String roomName;

   private FirebaseFirestore fStore;
```

```
    public DocumentReference hostUpdatesRef;
    public DocumentReference guestUpdatesRef;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_lobby);

        listView = findViewById(R.id.listViewPlayers);
        ArrayList<String> roomsList = new ArrayList<>();
        fStore = FirebaseFirestore.getInstance();
        broadcastReceiver = new MyBroadcastReceiver(roomsList, listView,
getApplicationContext());
        registerBroadcastListener();


        initNavHeader();
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                // join an existing room and add yourself as guest
                String otherPlayerName = roomsList.get(position);
                if (!otherPlayerName.equals(playerName)) // if the player is not challenging himself
                {
                    roomName = otherPlayerName;
                    roomRef = fStore.collection(ROOMSPATH).document(roomName);
                    Map<String, Object> userData = new HashMap<>();
                    userData.put("guest", playerName);
                    userData.put("isInGame", true);
                    addDataToDatabase(userData, roomRef);
                    listenForRoomUpdates();
                } else // if the player IS challenging himself, just tell him he can't.
                    Toast.makeText(getApplicationContext(), "Challenge... yourself? :)",
Toast.LENGTH_LONG).show();
            }
        });

        updateListview(roomsList, listView, getApplicationContext());
    }


    /**
     * Registers the broadcast listener.
     */
    public void registerBroadcastListener() {
        // According to the
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N)
            registerReceiver(broadcastReceiver, new
```

```
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION));

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)
        registerReceiver(broadcastReceiver, new
IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION));
  }

  /**
   * Unregisters the broadcast listener.
   */
  public void unregisterBroadcastListener() {
    if (broadcastReceiver != null)
        unregisterReceiver(broadcastReceiver);
  }


  /**
   * Contact button handler function. Creates an Intent object with an ACTION_SENDTO
(SMS app), to the developer's phone.
   * The user can send feedback to the developer and report issues and possible add-ons.
   */
  public void contactHandler() {
    Uri uri = Uri.parse("smsto:" + PhoneNumOfDeveloper);
    Intent intent = new Intent(Intent.ACTION_SENDTO, uri);
    intent.putExtra("sms_body", "Your feedback here");
    try {
        startActivity(intent);
    } catch (android.content.ActivityNotFoundException e) {
        Toast.makeText(getApplicationContext(), "SMS FAILED, please try again later",
Toast.LENGTH_SHORT).show();
    }
  }

  /**
   * Gets the "isInGame" field in the room document, which can be accessed by using
roomName.
   *
   * @return "isInGame" field value (Boolean) in the current room path. Throws an
IllegalStateException if couldn't retrieve the field.
   */
  // get isInGame value from db
  private boolean getIsInGame() {
    Task<DocumentSnapshot> getInGame =
fStore.collection(ROOMSPATH).document(roomName).get();
    while (!getInGame.isComplete()) {
        System.out.println("waiting for isInGame value");
    }
```

```
        if (getInGame.isSuccessful()) {
            DocumentSnapshot isInGameVal = getInGame.getResult();
            Boolean val = (Boolean) isInGameVal.get("isInGame");
            if (val != null)
                return val;
        }
        Log.d(TAG, "Error getting document: ", getInGame.getException());
        throw new IllegalStateException("couldn't get isBlackTurn from db");
    }

    /**
     * This function is called when a room gets updated, and checks if isInGame value on the
database has changed (means a guest joined the room).
     * if it has then it redirects execution to gameInvitationHandler(), else does nothing.
     */
    public void listenForRoomUpdates() {
        roomListener = roomRef.addSnapshotListener(new
EventListener<DocumentSnapshot>() {
            @Override
            public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException error) {
                if (error != null) {
                    Log.w(TAG, "Listen failed.", error);
                    return;
                }
                Log.d(TAG, "listenForRoomsUpdates");
                if (snapshot != null && snapshot.exists()) {
                    Boolean isInGame = (Boolean) snapshot.get("isInGame");
                    if (isInGame != null && isInGame) // if a guest joined
                    {
                        gameInvitationHandler();
                    }
                }
            }
        });
    }

    /**
     * Handles a game invite (when a guest joins someone's room) by calling the
handleHostInGameInvitation() and handleGuestInGameInvitation().
     */
    private void gameInvitationHandler() {
        String hostUsername = roomName;
        Map<String, Object> gameRequestData = new HashMap<>();
        AlertDialog.Builder gameRequestDialogBuilder = new
AlertDialog.Builder(LobbyActivity.this, AlertDialog.THEME_HOLO_LIGHT);
        gameRequestDialogBuilder.setCancelable(false);
```

```
    Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
    hostUpdatesRef = roomRef.collection("hostUpdates").document("gameStatus");
    guestUpdatesRef = roomRef.collection("guestUpdates").document("gameStatus");

    if (isHost()) // for the host
        handleHostInGameInvitation(gameRequestDialogBuilder, gameRequestData,
vibrator);

    else // for the guest
        handleGuestInGameInvitation(gameRequestDialogBuilder, hostUsername, vibrator);
  }

  /**
   * Handle guest side when a game invite occurs, by showing the invite dialog and start
listening to host response.
   *
   * @param gameRequestDialogBuilder The AlertDialog builder to build the "Challenging"
dialog in the guest's phone.
   * @param hostUsername          The host username value (of type String)
   * @param vibrator              The vibrator object to pass to startGame(), if the host accepts
the invite.
   */
  public void handleGuestInGameInvitation(AlertDialog.Builder gameRequestDialogBuilder,
String hostUsername, Vibrator vibrator) {
    gameRequestDialogBuilder.setMessage("Challenging " + hostUsername + "...");
    gameRequestDialogBuilder.setTitle("Challenge Sent");
    gameRequestDialogBuilder.setNegativeButton("CANCEL", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            Log.d(TAG, "GUEST CANCELED: SEND 'CANCELED' MESSAGE TO HOST");

            Map<String, Object> cancelRequestData = new HashMap<>();
            cancelRequestData.put("canceled", true);
            addDataToDatabase(cancelRequestData, guestUpdatesRef);

            // remove room listener for guest
            roomListener.remove();

            // change roomName back to guest's name because he canceled
            roomName = playerName;
            roomRef = fStore.collection(ROOMSPATH).document(roomName);

            // remove the guest from the room because he canceled - this is managed in
setListenerForGuestUpdates()

        }
    });
```

```
    AlertDialog gameRequestDialog = gameRequestDialogBuilder.create();

    gameRequestDialog.show();

    // listen for host response
    setListenerForHostUpdates(hostUsername, gameRequestDialog, hostUpdatesRef,
vibrator);
  }

  /**
   * Handle host side when a game invite occurs, by showing the invite dialog and start
listening for guest cancellation.
   *
   * @param gameRequestDialogBuilder The AlertDialog builder to build the "Challenging"
dialog in the host's phone.
   * @param gameRequestData        The host username value (of type String)
   * @param vibrator               The vibrator object to pass to startGame(), if the host accepts
the invite.
   */
  public void handleHostInGameInvitation(AlertDialog.Builder gameRequestDialogBuilder,
Map<String, Object> gameRequestData, Vibrator vibrator) {
    String guestUsername = getGuestUsername();

    gameRequestDialogBuilder.setMessage(guestUsername + " has challenged you to a
game!");
    gameRequestDialogBuilder.setTitle("You've Been Challenged");
    gameRequestDialogBuilder.setPositiveButton("ACCEPT", new
DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
        Log.d(TAG, "START GAME FOR HOST AND GUEST!");

        gameRequestData.put("startGame", true);

        addDataToDatabase(gameRequestData, hostUpdatesRef);

        // LET'S PLAY!
        startGame(vibrator);
      }
    });
    gameRequestDialogBuilder.setNegativeButton("DECLINE", new
DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
        Log.d(TAG, "HOST DECLINED: SEND 'DECLINED' MESSAGE TO GUEST");

        gameRequestData.put("startGame", false);
```

```
            addDataToDatabase(gameRequestData, hostUpdatesRef);

            // remove guest now because host doesn't want to play with him
            Map<String, Object> updates = new HashMap<>();
            updates.put("guest", FieldValue.delete()); // mark "guest" field as deletable on the
database (remove it)
            updates.put("isInGame", false); // update isInGame to false
            addDataToDatabase(updates, roomRef);


        }
    });

    AlertDialog gameRequestDialog = gameRequestDialogBuilder.create();

    gameRequestDialog.show();

    // listen for guest updates (e.g maybe he canceled the request)
    setListenerForGuestUpdates(guestUsername, gameRequestDialog, guestUpdatesRef);

  }


  /**
   * Host will call this function to listen for guest updates e.g if guest cancelled their
invitation.
   *
   * @param guestUsername     The guest's username (String)
   * @param gameRequestDialog The AlertDialog object of the game request. used when
host declines the request and then the dialog is dismissed.
   * @param guestUpdatesRef   The DocumentReference of the location to the guest
updates.
   */
  private void setListenerForGuestUpdates(String guestUsername, AlertDialog
gameRequestDialog, DocumentReference guestUpdatesRef) {
    guestUpdatesListener = guestUpdatesRef.addSnapshotListener(new
EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException error) {
            if (error != null) {
                Log.w(TAG, "Listen failed.", error);
                return;
            }
            if (snapshot != null && snapshot.exists()) {
                Boolean isCanceled = (Boolean) snapshot.get("canceled");
                if (isCanceled != null)
                    if (isCanceled) { // guest canceled
```

```
                    gameRequestDialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Sorry, " + guestUsername + "
cancelled their invitation.", Toast.LENGTH_SHORT).show();

                    // remove gameStatus document in guestUpdates
                    guestUpdatesRef.delete();

                    // remove the guest from the room because he canceled
                    Map<String, Object> updates = new HashMap<>();
                    updates.put("guest", FieldValue.delete()); // mark "guest" field as deletable
on the database (removes it)
                    updates.put("isInGame", false); // update isInGame to false
                    addDataToDatabase(updates, roomRef);
                }
            }
        }
    });
  }

  /**
   * Guest will call this function to listen for host updates when inviting him to a game. The
guest will start a game if the host accepts the invite.
   *
   * @param hostUsername      The host username (a String).
   * @param gameRequestDialog The AlertDialog object of the game request. used when
host accepts the request and then the dialog is dismissed and a game is started.
   * @param hostUpdatesRef    The DocumentReference to the hostUpdates location in the
database.
   * @param vibrator          The vibrator object, used when a game starts (if it is enabled in
the Settings).
   */
  private void setListenerForHostUpdates(String hostUsername, AlertDialog
gameRequestDialog, DocumentReference hostUpdatesRef, Vibrator vibrator) {
    hostUpdatesListener = hostUpdatesRef.addSnapshotListener(new
EventListener<DocumentSnapshot>() {
        @Override
        public void onEvent(@Nullable DocumentSnapshot snapshot, @Nullable
FirebaseFirestoreException error) {
            if (error != null) {
                Log.w(TAG, "Listen failed.", error);
                return;
            }
            if (snapshot != null && snapshot.exists()) {
                Boolean startGame = (Boolean) snapshot.get("startGame");
                if (startGame != null)
                    if (startGame) { // host confirmed = LET'S PLAY!
                        gameRequestDialog.dismiss();
                        startGame(vibrator);
```

```
                } else { // host declined
                    gameRequestDialog.dismiss();
                    Toast.makeText(getApplicationContext(), "Sorry, " + hostUsername + "
declined your invitation", Toast.LENGTH_SHORT).show();

                    // remove gameStatus document in hostUpdates
                    hostUpdatesRef.delete();

                    // change roomName back to guest's name
                    roomName = playerName;
                    roomRef = fStore.collection(ROOMSPATH).document(roomName);

                    // remove room listener for guest
                    roomListener.remove();
                }
            }
        }
    });
}

/**
 * Start the game. this will be also a vibration if the user didn't disable the option in the
settings.
 *
 * @param vibrator The vibrator object, used when a game is started.
 */
private void startGame(Vibrator vibrator) {
    SharedPreferences settingsPrefs = getSharedPreferences(SETTINGS_PREFS,
MODE_PRIVATE);
    boolean isVibrate = settingsPrefs.getBoolean("vibrate", true);

    if (isVibrate) {
        // Vibrate for 500 milliseconds
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            vibrator.vibrate(VibrationEffect.createOneShot(500,
VibrationEffect.DEFAULT_AMPLITUDE));
        } else {
            //deprecated in API 26
            vibrator.vibrate(500);
        }
    }

    if (!isHost()) // delete the guest's room when starting a game
    {
        DocumentReference guestRoomRef =
fStore.collection(ROOMSPATH).document(playerName);
        guestRoomRef.delete();
    }
```

```
        Intent intent = new Intent(getApplicationContext(), GameActivity.class);
        startActivity(intent);
    }


    /**
     * init the menu and connect the user to the database when finished.
     */
    public void initNavHeader() {
        toolbar = findViewById(R.id.toolBar);
        setSupportActionBar(toolbar);

        drawer = findViewById(R.id.drawer_layout);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawer, toolbar,
R.string.nav_open, R.string.nav_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView nav_view = findViewById(R.id.nav_view);
        nav_view.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item) {
                drawer.closeDrawer(GravityCompat.START); // close the menu
                switch (item.getItemId()) {
                    case R.id.nav_settings:
                        startActivity(new Intent(getApplicationContext(), SettingsActivity.class));
                        break;
                    case R.id.nav_scores:
                        startActivity(new Intent(getApplicationContext(), ScoresActivity.class));
                        break;
                    case R.id.nav_logout:
                        FirebaseAuth.getInstance().signOut();
                        listView.setAdapter(null);
                        disconnectUser();
                        startActivity(new Intent(getApplicationContext(), MainActivity.class));
                        finish();
                        break;
                    case R.id.nav_feedback:
                        contactHandler();
                        break;

                    default:
                        throw new IllegalStateException("Unexpected value: " + item.getItemId());
                }
                return false; // return false means unchecked state
```

```
        }
    });

    View headerView = nav_view.getHeaderView(0);
    mUsername = headerView.findViewById(R.id.textviewUsername); // referring to the
header style
    mEmail = headerView.findViewById(R.id.textviewEmail); // referring to the header style

    FirebaseAuth fAuth = FirebaseAuth.getInstance();
    FirebaseFirestore fStore = FirebaseFirestore.getInstance();

    String uid = Objects.requireNonNull(fAuth.getCurrentUser()).getUid(); // can't be null
cuz we're already in WaitingRoom...
    DocumentReference userRef = fStore.collection("users").document(uid);

    userRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>()
{
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()) {
                playerName = task.getResult().getString("username");
                mUsername.setText(playerName);
                Log.d(TAG, "set username field in the navigation header to: " + playerName);

                connectUser();

            } else
                Log.d(TAG, "get() failed with: " + task.getException());
        }
    });
    mEmail.setText(Objects.requireNonNull(fAuth.getCurrentUser()).getEmail()); // again,
can't be null...
  }

  /**
   * Disconnect the user from the database. This removes the room only if the user is not in a
game.
   */
  public void disconnectUser() {
    if (isHost())
        roomRef.delete();

  }

  /**
   * Create a room in the database, named playerName.
   * This adds the fields "host" which saves the player's name, and "isInGame" which
indicates if the room is taken (in a game)
```

```
 */
public void connectUser() {
    if (playerName != null) {
        roomName = playerName;
        roomRef = fStore.collection(ROOMSPATH).document(roomName);
        Map<String, Object> userData = new HashMap<>();
        userData.put("host", playerName);
        userData.put("isInGame", false);
        addDataToDatabase(userData, roomRef);
        listenForRoomUpdates();
    }

}

/**
 * When pressing the back button in the phone, the menu will collapse.
 */
@Override
public void onBackPressed() {
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    }
}

/**
 * When the activity is in the "STOP" state (onStop() is called), do clean-ups by removing
the listeners, as well as guest and host updates.
 */
@Override
public void onStop() {
    if (roomListener != null)
        roomListener.remove();
    if (guestUpdatesListener != null)
        guestUpdatesListener.remove();
    if (hostUpdatesListener != null)
        hostUpdatesListener.remove();
    if (roomsUpdaterViewListener != null)
        roomsUpdaterViewListener.remove();
    if (guestUpdatesRef != null)
        guestUpdatesRef.delete();
    if (hostUpdatesRef != null)
        hostUpdatesRef.delete();

    super.onStop();
}

/**
 * When a application is being destroyed, unregister the broadcast listener and disconnect
```

```
the user from the database.
    * It is important to mention that this callback is not called always.
    *
    * @see <a href="https://stackoverflow.com/questions/4449955/activity-ondestroy-never-called"> onDestroy is never called </a>
    */
   @Override
   public void onDestroy() {
      unregisterBroadcastListener();
      disconnectUser();

      super.onDestroy();

   }
}
```

Logic

```
package com.example.checkers;

/**
 * This class contains static helper functions for the Logic implementation in the game.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class Logic {

   /**
    * Wrapper function for isBlackNeeds2BeKing and isRedNeeds2BeKing.
    *
    * @param isBlack The piece color.
    * @param x       The given x cord.
    * @return true or false according if the piece needs to be king or not.
    */
   public static boolean isPieceNeeds2BeKing(boolean isBlack, int x) {
      if (isBlack)
         return isBlackNeeds2BeKing(x);
      return isRedNeeds2BeKing(x);
   }

   /**
    * Check if red is at the bottom of the board.
    *
    * @param x The given x cord.
    * @return true if the x cord is in the end for black, thus he needs to become king, false otherwise.
```

```
 */
private static boolean isBlackNeeds2BeKing(int x) {
    return x == 0;
}

/**
 * Check if red is at the top of the board.
 *
 * @param x The given x cord.
 * @return true if the x cord is in the end for red, thus he needs to become king, false
otherwise.
 */
private static boolean isRedNeeds2BeKing(int x) {
    return x == 7;
}

/**
 * Check if the given y cord is on right edge.
 *
 * @param y The given y cord.
 * @return true if the y cord is on the right edge of the board, false otherwise.
 */
public static boolean isOnRightEdge(int y) {
    return y == 7;
}

/**
 * Check if the given y cord is on left edge.
 *
 * @param y The given y cord.
 * @return true if the y cord is on the left edge of the board, false otherwise.
 */
public static boolean isOnLeftEdge(int y) {
    return y == 0;
}

/**
 * Check if red has space to move.
 *
 * @param x The given x cord.
 * @return true if the x cord is not in the end of the board for red, false otherwise.
 */
public static boolean canRedMoveDown(int x) {
    return x + 1 <= 7; // if red reached the end (already king)
}

/**
 * Check if black has space to move.
```

```
 *
 * @param x The given x cord.
 * @return true if the x cord is not in the end of the board for black, false otherwise.
 */
public static boolean canBlackMoveUp(int x) {
   return x - 1 >= 0; // if black reached the end (already king)
}

/**
 * Check if not on edge and has space to jump.
 *
 * @param x      The given x cord.
 * @param y      The given y cord.
 * @param isBlack The color of the piece.
 * @return true if piece is not on the edge for left jump, false otherwise.
 */
public static boolean hasSpaceForLeftJump(int x, int y, boolean isBlack) {
   if (isBlack)
      return x - 2 >= 0 && y - 2 >= 0;
   return x + 2 <= 7 && y - 2 >= 0;
}

/**
 * Check if not on edge and has space to jump.
 *
 * @param x      The given x cord.
 * @param y      The given y cord.
 * @param isBlack The color of the piece.
 * @return true if piece is not on the edge for right jump, false otherwise.
 */
public static boolean hasSpaceForRightJump(int x, int y, boolean isBlack) {
   if (isBlack)
      return x - 2 >= 0 && y + 2 <= 7;
   return x + 2 <= 7 && y + 2 <= 7;
}

/**
 * Check if the given x and y axis are taken on the board by another piece.
 *
 * @param board The Board object that holds the current state of the game.
 * @param x     The given x cord.
 * @param y     The given y cord.
 * @return true if the tile is not taken by another piece in the board, false otherwise.
 */
public static boolean isTileAvailable(Board board, int x, int y) {
   return board.getBoardArray()[x][y] == null;
}
```

```
    /**
     * Check if the given tile is darkwood colored or not (darkwood colored tile means that a
checker can be placed on it).
     *
     * @param x the x axis of the tile on the board.
     * @param y the y axis of the tile on the baord.
     * @return True if a checker can be placed on a given tile (represented by x and y axis),
false otherwise.
     */
    public static boolean isTileForChecker(int x, int y) {
        return (x + y) % 2 == 1; // this is true for every tile that a checker can be on (darkwood
colored)
    }

}
```

LoginActivity

```
package com.example.checkers;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

/**
```

```
 * This class manages the LoginActivity in the application.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class LoginActivity extends AppCompatActivity {

    public Button login;
    public TextView createAccount;
    public EditText mEmail;
    public EditText mPassword;
    public ProgressBar progressBar;
    private FirebaseAuth fAuth; // used to authenticate the user

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        login = findViewById(R.id.auth);
        createAccount = findViewById(R.id.textViewCreateAccount);
        mEmail = findViewById(R.id.editTextEmail);
        mPassword = findViewById(R.id.editTextPassword);

        fAuth = FirebaseAuth.getInstance();
        progressBar = findViewById(R.id.progressBar);
        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                loginHandler();
            }
        });

        createAccount.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            // Just switch to the Register activity for creating a new account
            Intent intent = new Intent(getApplicationContext(), RegisterActivity.class);
            startActivity(intent);
            finish();
        }
    });
}

/**
 * Handle the login button press, and check in the database if the given input is valid.
 * If the input is valid then log in, else show error on text boxes.
 */
public void loginHandler() {
    String email = mEmail.getText().toString().trim(); // remove spaces
    String password = mPassword.getText().toString().trim();

    // check user input (e.g make sure that the user entered a password)
    if (!validateFields(email, password))
        return;

    progressBar.setVisibility(View.VISIBLE);

    // authenticate the user
    login.setEnabled(false);

    fAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            Context appContext = getApplicationContext();
            if (task.isSuccessful()) {
                Toast.makeText(appContext, "Logged in Successfully.",
```

```
Toast.LENGTH_SHORT).show();
                startActivity(new Intent(appContext, LobbyActivity.class));
                finish();
            } else {
                Exception exception = task.getException();
                if (exception != null)
                    Toast.makeText(appContext, "Error! " + exception.getMessage(),
Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(appContext, "Error: Couldn't log in.",
Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.INVISIBLE);
            }
            login.setEnabled(true);
        }
    });
  }

  /**
   * Check if user input in different fields such as email and password are valid or not (e.g
password length must be atleast 6 characters).
   *
   * @param email    The string representation of the email entered.
   * @param password The string representation of the password entered.
   * @return true if user input is OK, else otherwise.
   */
  private boolean validateFields(String email, String password) {
    boolean isValid = true;

    if (email.isEmpty()) {
        mEmail.setError("Email is Required.");
        isValid = false;
    }
    if (password.isEmpty()) {
```

```
            mPassword.setError("Password is Required.");
            isValid = false;
        } else if (password.length() < 6) {
            mPassword.setError("Password Must be >= 6 Characters");
            isValid = false;
        }
        return isValid;
    }

}
```

MainActivity

```
package com.example.checkers;

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

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreSettings;

/**
 * This class manages the MainActivity in the application.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class MainActivity extends AppCompatActivity {

    protected Button login;
    protected Button settings;

    /**
     * Handle the login and settings button presses, and redirect to the corresponding activity.
     *
     * @param savedInstanceState The saved instance bundle from the last run (if is not null)
which is passed to the super.
     */
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // disable Firestore cache
        FirebaseFirestore.getInstance().setFirestoreSettings(removeFirestorePersistence());

        if (FirebaseAuth.getInstance().getCurrentUser() != null) // if user already logged in
        {
            System.out.println("user already logged in, redirecting to WaitingRoom");
            startActivity(new Intent(getApplicationContext(), LobbyActivity.class));
        }

        login = findViewById(R.id.login);
        settings = findViewById(R.id.settings);

        login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                startActivity(intent);
            }
        });

        settings.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getApplicationContext(), SettingsActivity.class);
                startActivity(intent);
            }
        });
    }

    /**
     * Remove the Firestore persistence, thus disabling getting data from cache (because we
need realtime updates during a game).
     *
     * @return The FirebaseFirestoreSettings object, to change the Firestore settings and
disable cache.
     */
    public FirebaseFirestoreSettings removeFirestorePersistence() {
        return new FirebaseFirestoreSettings.Builder()
                .setPersistenceEnabled(false)
                .build();
    }
}
```

Move

```
package com.example.checkers;

/**
 * This class defines a simple move on the board.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class Move {
   private int startX, startY, endX, endY;

   public Move(int startX, int startY, int endX, int endY) {

      this.startX = startX;
      this.startY = startY;
      this.endX = endX;
      this.endY = endY;
   }

   /**
    * Replace the ImageViews and do the transformation between the pieces - remove the old
piece from the screen and add the new piece.
    *
    * @param isBlack The color the moved piece.
    * @param isKing  A boolean indicating if the piece is a KingPiece or not.
    */
   public void perform(boolean isBlack, boolean isKing) {

GameActivity.imageViewsTiles[this.startX][this.startY].setImageResource(android.R.color.tra
nsparent);
      GameActivity.imageViewsTiles[this.startX][this.startY].setClickable(false);
      if (isBlack)
```

```
        if (isKing) {
GameActivity.imageViewsTiles[this.endX][this.endY].setImageResource(R.drawable.black_king);
GameActivity.imageViewsTiles[this.endX][this.endY].setTag(R.drawable.black_king);
        } else {
GameActivity.imageViewsTiles[this.endX][this.endY].setImageResource(R.drawable.black_piece);
GameActivity.imageViewsTiles[this.endX][this.endY].setTag(R.drawable.black_piece);
        }

      else if (isKing) {
GameActivity.imageViewsTiles[this.endX][this.endY].setImageResource(R.drawable.red_king);
GameActivity.imageViewsTiles[this.endX][this.endY].setTag(R.drawable.red_king_highlighted);
      } else {
GameActivity.imageViewsTiles[this.endX][this.endY].setImageResource(R.drawable.red_piece);
         GameActivity.imageViewsTiles[this.endX][this.endY].setTag(R.drawable.red_piece);
      }

   }


   public int getStartX() {
      return startX;
   }
```

```
    public void setStartX(int startX) {
        this.startX = startX;
    }

    public int getStartY() {
        return this.startY;
    }

    public void setStartY(int startY) {
        this.startY = startY;
    }

    public int getEndX() {
        return this.endX;
    }

    public void setEndX(int endX) {
        this.endX = endX;
    }

    public int getEndY() {
        return this.endY;
    }

    public void setEndY(int endY) {
        this.endY = endY;
    }

}
```

MyBroadcastReceiver

```
package com.example.checkers;
```

```
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.widget.ListView;
import android.widget.Toast;

import static com.example.checkers.DBUtils.updateListview;
import static com.example.checkers.LobbyActivity.roomsUpdaterViewListener;

import java.util.ArrayList;

/**
 * This class handles network changes on the user's phone.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class MyBroadcastReceiver extends android.content.BroadcastReceiver {

   private final ArrayList<String> roomsList;
   private final ListView listView;
   private final Context appContext;

   public MyBroadcastReceiver(ArrayList<String> roomsList, ListView listView, Context
appContext) {
      // for updateListView when no internet.
      this.roomsList = roomsList;
      this.listView = listView;
      this.appContext = appContext;
   }

   /**
    * A callback function for network changes.
    * Check if the phone is connected to the internet, and if it's not connected, then show a
Toast and stop listening to new players joining the server.
    *
    * @param context The context of the activity.
    * @param intent  The intent object used.
    */
   @Override
   public void onReceive(Context context, Intent intent) {
      // an Intent broadcast.
      if (isOnline(context)) {
         updateListview(this.roomsList, this.listView, this.appContext);
      } else {
         Toast.makeText(context, "Lost Internet Connection.", Toast.LENGTH_SHORT).show();
         roomsUpdaterViewListener.remove(); // removing the listener to avoid network
```

```
issues
        }

    }

    /**
     * Check if there is an internet connection or not.
     *
     * @param context The context of the activity.
     * @return true if there is an internet connection, false otherwise.
     */
    public boolean isOnline(Context context) {

        ConnectivityManager cm = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo netInfo = cm.getActiveNetworkInfo();
        //should check null because in airplane mode it will be null
        return (netInfo != null && netInfo.isConnected());
    }
}
```

OnClickListenerForPieceMoves

```
package com.example.checkers;

import static com.example.checkers.DBUtils.getIsBlackTurn;
import static com.example.checkers.DBUtils.isHost;
import static com.example.checkers.LobbyActivity.roomRef;

import android.content.Context;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import com.google.firebase.firestore.CollectionReference;

/**
 * This class implements an onClickListener and controls the movements of the pieces.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class OnClickListenerForPieceMoves implements View.OnClickListener {

    public static final String TAG = "MyListenerForPieceMoves";
    public static ImageView[] lastUsedImageViews; // for removing the setOnClickListeners
that we set and the player did not choose, so there will not be hanging listeners.
```

```
    public static Context appContext; // for showing dialogs
    public Piece piece;
    public TextView currentTurn;
    private final Board board;
    public static CollectionReference gameplayRef;


    public OnClickListenerForPieceMoves(Piece piece, Board board, TextView currentTurn) {
        this.piece = piece;
        this.board = board;
        this.currentTurn = currentTurn;
        gameplayRef = roomRef.collection("gameplay");
        appContext = null;
        lastUsedImageViews = new ImageView[10];
    }

    /**
     * A callback function for when a piece has been clicked on.
     *
     * @param v The view that has been clicked on.
     */
    @Override
    public void onClick(View v) {
        displayMoveOptionsAndMove(this.piece.getX(), this.piece.getY(), this.piece.isBlack(),
this.piece.isKing(), (ImageView) v);
    }

    /**
     * Handles the callback for when a piece has been clicked. It highlights the clicked piece
and calls the move() functions accordingly to the piece type (King, Red or Black).
     *
     * @param x        The given x cord of the piece.
     * @param y        The given y cord of the piece.
     * @param isBlack    The color of the piece.
     * @param isKing     A boolean indicating if the piece is a king piece or not.
     * @param pieceImage The piece ImageView.
     */
    public void displayMoveOptionsAndMove(int x, int y, boolean isBlack, boolean isKing,
ImageView pieceImage) {

        appContext = pieceImage.getContext();
        this.piece.clearPossibleLocationMarkers(board);
        this.piece.unsetOnClickLastImageViews(board);

        boolean isBlackTurn = getIsBlackTurn();

        if (isHost()) // for the host (for black)
        {
```

```
        if (isBlack && isBlackTurn) {
            highlightPiece(true, isKing, pieceImage);
            if (!isKing) {
                // move black
                ((BlackPiece) this.piece).move(board);
            } else {
                this.piece = new KingPiece(x, y, true, currentTurn);
                ((KingPiece) this.piece).move(board);
            }
        }

    } else // for the guest (for red)
    {
        if (!isBlack && !isBlackTurn) {

            highlightPiece(false, isKing, pieceImage);
            if (!isKing) {

                ((RedPiece) this.piece).move(board);
            } else {
                this.piece = new KingPiece(x, y, false, currentTurn);
                ((KingPiece) this.piece).move(board);
            }
        }

    }
}


/**
 * Highlight the piece changing its ImageResource to a highlighted piece.
 *
 * @param isBlack The color of the piece.
 * @param isKing  A boolean indicating if the piece is a king piece or not.
 * @param piece   The piece ImageView to highlight.
 */
private void highlightPiece(boolean isBlack, boolean isKing, ImageView piece) {
    if (isBlack) {
        if (isKing) {
            piece.setImageResource(R.drawable.black_king_highlighted);
            piece.setTag(R.drawable.black_king_highlighted);
        } else {
            piece.setImageResource(R.drawable.black_piece_highlighted);
            piece.setTag(R.drawable.black_piece_highlighted);
        }

    } else {
        if (isKing) {
```

```
            piece.setImageResource(R.drawable.red_king_highlighted);
            piece.setTag(R.drawable.red_king_highlighted);
        } else {
            piece.setImageResource(R.drawable.red_piece_highlighted);
            piece.setTag(R.drawable.red_piece_highlighted);
        }

    }
  }


}
```

Piece

```
package com.example.checkers;



import static com.example.checkers.DBUtils.checkGameOver;
import static com.example.checkers.DBUtils.updateBlackTurnInDb;
import static com.example.checkers.DBUtils.uploadPieceLocationToDb;
import static com.example.checkers.OnClickListenerForPieceMoves.gameplayRef;
import static com.example.checkers.OnClickListenerForPieceMoves.lastUsedImageViews;

import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

/**
 * This class defines a piece in the game.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class Piece {

  protected int x;
  protected int y;
```

```
    protected boolean isKing;
    protected boolean isBlack; // color-wise
    protected final TextView currentTurn;

    public Piece(int x, int y, boolean isBlack, boolean isKing, TextView currentTurn) {
        this.x = x;
        this.y = y;
        this.isBlack = isBlack;
        this.isKing = isKing;
        this.currentTurn = currentTurn;
    }

    public Piece(int x, int y, boolean isBlack, TextView currentTurn) {
        this.x = x;
        this.y = y;
        this.isBlack = isBlack;
        this.isKing = false;
        this.currentTurn = currentTurn;
    }

    /**
     * Check if the piece can be moved or not.
     * This function is overridden by all the subclasses of Piece, because their canMove() is
specific to each subclass.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if the piece can move, false otherwise.
     */
    public boolean canMove(Board board) {
        if (this.isKing) // if King piece : can king move?
            return ((KingPiece) this).canMove(board);
        else if (this.isBlack) // else if Black piece : can black move?
            return ((BlackPiece) this).canMove(board);
        else // else Red piece : can red move?
```

```
        return ((RedPiece) this).canMove(board);
    }

    /**
     * Show a right diagonal move, and do it if the user wishes to.
     *
     * @param rightMove      A Move object representing the move to be made.
     * @param rightPieceImage The ImageView of the right piece.
     * @param isBlack        The color of the piece.
     * @param isJump         A boolean indicating if there is a jump.
     * @param jumpedPieceX   If there is a jump, this will hold the x cord of it.
     * @param board          The Board object that holds the current state of the game.
     */
    protected void rightDiagonal(Move rightMove, ImageView rightPieceImage, boolean
isBlack, boolean isJump, int jumpedPieceX, Board board) {
        rightPieceImage.setImageResource(R.drawable.possible_location_marker);
        rightPieceImage.setClickable(true);
        rightPieceImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int endX = rightMove.getEndX();
                int endY = rightMove.getEndY();
                int startX = rightMove.getStartX();
                int startY = rightMove.getStartY();

                // updating boardArray
                updateBoardArray(board, endX, endY);
                board.getBoardArray()[startX][startY] = null; // remove old piece
                int jumpedPieceY = startY + 1;
                if (isJump) {
                    // delete the jumped piece

GameActivity.imageViewsTiles[jumpedPieceX][jumpedPieceY].setImageResource(android.R.
color.transparent);
```

```
            GameActivity.imageViewsTiles[jumpedPieceX][jumpedPieceY].setClickable(false);
            board.getBoardArray()[jumpedPieceX][jumpedPieceY] = null;
        }
        clearPossibleLocationMarkers(board);
        unsetOnClickLastImageViews(board);

        // check if needs to be king
        if (Logic.isPieceNeeds2BeKing(isBlack, endX))
            board.getBoardArray()[endX][endY].setKing();

        rightMove.perform(isBlack, board.getBoardArray()[endX][endY].isKing());

        // set onClick for the new piece (location)
        rightPieceImage.setClickable(true);
        rightPieceImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                OnClickListenerForPieceMoves onClickListenerForPieceMoves = new
OnClickListenerForPieceMoves(board.getBoardArray()[endX][endY], board, currentTurn);
                onClickListenerForPieceMoves.displayMoveOptionsAndMove(endX, endY,
isBlack, board.getBoardArray()[endX][endY].isKing(), rightPieceImage); // recursively show
more move options
            }
        });

        // updating next turn - passing the turn to the other player
        updateBlackTurnInDb(!isBlack);
        currentTurn.setText(R.string.not_your_turn);

        // upload new piece location to db
        uploadPieceLocationToDb(rightMove, isJump, jumpedPieceX, jumpedPieceY,
board.getBoardArray()[endX][endY].isKing());

        checkGameOver(board, !isBlack);
```

```
        }
    });
  }

  /**
   * Show a left diagonal move, and do it if the user wishes to.
   *
   * @param leftMove      A Move object representing the move to be made.
   * @param leftPieceImage The ImageView of the right piece.
   * @param isBlack       The color of the piece.
   * @param isJump        A boolean indicating if there is a jump.
   * @param jumpedPieceX  If there is a jump, this will hold the x cord of it.
   * @param board         The Board object that holds the current state of the game.
   */
  protected void leftDiagonal(Move leftMove, ImageView leftPieceImage, boolean isBlack,
boolean isJump, int jumpedPieceX, Board board) {
    leftPieceImage.setImageResource(R.drawable.possible_location_marker);
    leftPieceImage.setClickable(true);
    leftPieceImage.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        int endX = leftMove.getEndX();
        int endY = leftMove.getEndY();
        int startX = leftMove.getStartX();
        int startY = leftMove.getStartY();

        // updating boardArray
        updateBoardArray(board, endX, endY);
        board.getBoardArray()[startX][startY] = null; // remove old piece
        int jumpedPieceY = startY - 1;
        if (isJump) {
          // delete the jumped piece

GameActivity.imageViewsTiles[jumpedPieceX][jumpedPieceY].setImageResource(android.R.
```

```
color.transparent);
            GameActivity.imageViewsTiles[jumpedPieceX][jumpedPieceY].setClickable(false);
            board.getBoardArray()[jumpedPieceX][jumpedPieceY] = null;
        }
        clearPossibleLocationMarkers(board);
        unsetOnClickLastImageViews(board);

        // check if needs to be king
        if (Logic.isPieceNeeds2BeKing(isBlack, endX))
            board.getBoardArray()[endX][endY].setKing();

        leftMove.perform(isBlack, board.getBoardArray()[endX][endY].isKing());

        // set onClick for the new piece (location)
        leftPieceImage.setClickable(true);
        leftPieceImage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                OnClickListenerForPieceMoves onClickListenerForPieceMoves = new
OnClickListenerForPieceMoves(board.getBoardArray()[endX][endY], board, currentTurn);
                onClickListenerForPieceMoves.displayMoveOptionsAndMove(endX, endY,
isBlack, board.getBoardArray()[endX][endY].isKing(), leftPieceImage); // recursively show
more move options
            }
        });

        // updating next turn - passing the turn to the other player
        updateBlackTurnInDb(!isBlack);
        currentTurn.setText(R.string.not_your_turn);

        // upload new piece location to db
        uploadPieceLocationToDb(leftMove, isJump, jumpedPieceX, jumpedPieceY,
board.getBoardArray()[endX][endY].isKing());
```

```
            checkGameOver(board, !isBlack);
        }
    });
}

/**
 * Update the board with the new piece.
 * This function is overridden by all the subclasses of Piece, because their board-update
when a piece moves is different.
 *
 * @param board The Board object that holds the current state of the game.
 * @param endX  The end x cord of the move.
 * @param endY  The end y cord of the move.
 */
protected void updateBoardArray(Board board, int endX, int endY) {
    board.getBoardArray()[endX][endY] = new Piece(endX, endY, isBlack, currentTurn);
}


/**
 * Clear possible location markers ImageViews on the board (it appears when a player
clicks on a piece) when a player clicked a different piece.
 *
 * @param board The Board object that holds the current state of the game.
 */
protected void clearPossibleLocationMarkers(Board board) {
    for (int i = 0; i < Board.SIZE; i++) {
        for (int j = 0; j < Board.SIZE; j++) {
            if (Logic.isTileForChecker(i, j)) {
                if (board.getBoardArray()[i][j] != null) {
                    Integer tag = (Integer) GameActivity.imageViewsTiles[i][j].getTag();
                    if (tag != null) {
                        if (board.getBoardArray()[i][j].isBlack()) {
                            if (tag == R.drawable.black_piece_highlighted) {
```

```
GameActivity.imageViewsTiles[i][j].setImageResource(R.drawable.black_piece);
                    } else if (tag == R.drawable.black_king_highlighted) // king
                    {

GameActivity.imageViewsTiles[i][j].setImageResource(R.drawable.black_king);
                    }

                  } else {
                    if (tag == R.drawable.red_piece_highlighted) {

GameActivity.imageViewsTiles[i][j].setImageResource(R.drawable.red_piece);
                    } else if (tag == R.drawable.red_king_highlighted)// king
                    {

GameActivity.imageViewsTiles[i][j].setImageResource(R.drawable.red_king);
                    }



                  }
                }
              } else // remove possible_loc_marker

GameActivity.imageViewsTiles[i][j].setImageResource(android.R.color.transparent);
            }
          }
        }
      }

    /**
     * Responsible for removing the setOnClickListeners that we set and the player did not
choose to go, so there will not be hanging listeners.
     *
     * @param board The Board object that holds the current state of the game.
```

```
 */
protected void unsetOnClickLastImageViews(Board board) {
   // how to make sure that at the place of the image there isn't also a checkers piece:
   // 1. get id of image; extract X and Y axis from it
   // 2. compare those X and Y in boardArray
   // 3. if a piece is found at that location: do not unset it!
   // 4. else: unset it.
   for (ImageView image : lastUsedImageViews) {
      if (image != null) {
         // get id of image and extract axis
         String idStr = image.getResources().getResourceEntryName(image.getId());
         String axis = idStr.substring(idStr.length() - 2);
         int x = Character.getNumericValue(axis.charAt(0));
         int y = Character.getNumericValue(axis.charAt(1));



         if (board.getBoardArray()[x][y] == null) {
            image.setClickable(false);
            image.setOnClickListener(null);
         }
      }
   }
}



public boolean isKing() {
   return this.isKing;
}

public void setKing() {
   this.isKing = true;
}

public int getX() {
```

```
        return this.x;
    }

    public void setX(int x) {
        this.x = x;
    }

    public int getY() {
        return this.y;
    }

    public void setY(int y) {
        this.y = y;
    }

    public boolean isBlack() {
        return this.isBlack;
    }

    public void setBlack(boolean black) {
        isBlack = black;
    }
}
```

RedPiece

```
package com.example.checkers;

import static com.example.checkers.OnClickListenerForPieceMoves.lastUsedImageViews;

import android.widget.ImageView;
import android.widget.TextView;

/**
 * This class defines a red piece.
```

```
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class RedPiece extends Piece {

   public RedPiece(int x, int y, TextView currentTurn) {
      super(x, y, false, false, currentTurn);
   }

   /**
    * Check if the piece can move or not.
    *
    * @param board The Board object that holds the current state of the game.
    * @return true if red piece can move, false otherwise.
    */
   @Override
   public boolean canMove(Board board) {
      boolean left = isLeftDiagonalAvailable(board);
      boolean leftJump = isLeftJumpDiagonalAvailable(board);
      boolean right = isRightDiagonalAvailable(board);
      boolean rightJump = isRightJumpDiagonalAvailable(board);

      return left || leftJump || right || rightJump;
   }

   /**
    * Update the new piece in the board to be an object of RedPiece.
    *
    * @param board The Board object that holds the current state of the game.
    * @param endX  The end X cord of the move.
    * @param endY  The end Y cord of the move.
    */
   @Override
```

```
protected void updateBoardArray(Board board, int endX, int endY) {
    board.getBoardArray()[endX][endY] = new RedPiece(endX, endY, currentTurn);
}

/**
 * Move the piece according to red logic.
 *
 * @param board The Board object that holds the current state of the game.
 */
public void move(Board board) {
    /* ------------------------- left diagonal ------------------------- */
    if (isLeftDiagonalAvailable(board)) {
        Move leftMove = new Move(x, y, x + 1, y - 1);
        ImageView leftPieceImage = GameActivity.imageViewsTiles[x + 1][y - 1];
        lastUsedImageViews[4] = leftPieceImage;
        leftDiagonal(leftMove, leftPieceImage, false, false, 0, board);
    }

    /* ------------------------- left-JUMP diagonal ------------------------- */

    if (isLeftJumpDiagonalAvailable(board)) {
        ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x + 2][y - 2];
        lastUsedImageViews[5] = leftJumpPieceImage;
        Move leftJumpMove = new Move(x, y, x + 2, y - 2);
        leftDiagonal(leftJumpMove, leftJumpPieceImage, false, true, x + 1, board);
    }

    /* ------------------------- right diagonal ------------------------- */
    if (isRightDiagonalAvailable(board)) {
        Move rightMove = new Move(x, y, x + 1, y + 1);
        ImageView rightPieceImage = GameActivity.imageViewsTiles[x + 1][y + 1];
        lastUsedImageViews[6] = rightPieceImage;
        rightDiagonal(rightMove, rightPieceImage, false, false, 0, board);
    }
```

```
        /* ------------------------- right-JUMP diagonal ------------------------- */
        if (isRightJumpDiagonalAvailable(board)) {
            ImageView leftJumpPieceImage = GameActivity.imageViewsTiles[x + 2][y + 2];
            lastUsedImageViews[7] = leftJumpPieceImage;
            Move leftJumpMove = new Move(x, y, x + 2, y + 2);
            rightDiagonal(leftJumpMove, leftJumpPieceImage, false, true, x + 1, board);
        }
    }

    /**
     * Check if left diagonal is available.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if diagonal is available, false otherwise.
     */
    private boolean isLeftDiagonalAvailable(Board board) {
        return (Logic.canRedMoveDown(x) && !Logic.isOnLeftEdge(y) &&
Logic.isTileAvailable(board, x + 1, y - 1) /* left tile */);
    }

    /**
     * Check if left-jump diagonal is available.
     *
     * @param board The Board object that holds the current state of the game.
     * @return true if diagonal is available, false otherwise.
     */
    private boolean isLeftJumpDiagonalAvailable(Board board) {
        return (Logic.hasSpaceForLeftJump(x, y, false) && Logic.isTileAvailable(board, x + 2, y -
2) && !Logic.isTileAvailable(board, x + 1, y - 1) && board.getBoardArray()[x + 1][y -
1].isBlack());
    }

    /**
```

```
    * Check if right diagonal is available.
    *
    * @param board The Board object that holds the current state of the game.
    * @return true if diagonal is available, false otherwise.
    */
   private boolean isRightDiagonalAvailable(Board board) {
      return (Logic.canRedMoveDown(x) && !Logic.isOnRightEdge(y) &&
Logic.isTileAvailable(board, x + 1, y + 1) /* right tile */);
   }

   /**
    * Check if right-jump diagonal is available.
    *
    * @param board The Board object that holds the current state of the game.
    * @return true if diagonal is available, false otherwise.
    */
   private boolean isRightJumpDiagonalAvailable(Board board) {
      return (Logic.hasSpaceForRightJump(x, y, false) && Logic.isTileAvailable(board, x + 2, y +
2) && !Logic.isTileAvailable(board, x + 1, y + 1) && board.getBoardArray()[x + 1][y +
1].isBlack());
   }

}
```

RegisterActivity

```
package com.example.checkers;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

/**
 * This class handles the RegisterActivity operations.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class RegisterActivity extends AppCompatActivity {
   public static final String TAG = "Register";
   public TextView login;
   public Button register;
   public EditText mEmail;
   public EditText mPassword;
   public EditText mConfirmPassword;
   public EditText mUsername;
```

```
public ProgressBar progressBar;
private FirebaseAuth fAuth;
private FirebaseFirestore fStore; // database

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);

    login = findViewById(R.id.textViewLogin);
    register = findViewById(R.id.ButtonRegister);
    mEmail = findViewById(R.id.editTextEmail);
    mPassword = findViewById(R.id.editTextPassword);
    mConfirmPassword = findViewById(R.id.editTextConfirmPassword);
    mUsername = findViewById(R.id.editTextName);

    progressBar = findViewById(R.id.progressBar);
    fAuth = FirebaseAuth.getInstance();
    fStore = FirebaseFirestore.getInstance();

    login.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // Redirect to Login Page
            startActivity(new Intent(getApplicationContext(), LoginActivity.class));
            finish();
        }
    });

    register.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            registerHandler();
        }
```

```
        });
    }

    /**
     * Handle the register button press, and check if the user input is valid.
     * If the input is valid then register, else show error on text boxes.
     */
    public void registerHandler() {
        String email = mEmail.getText().toString().trim(); // remove spaces
        String password = mPassword.getText().toString().trim();
        String confirmPassword = mConfirmPassword.getText().toString().trim();
        String username = mUsername.getText().toString().trim();

        // check user input (e.g make sure that the user entered a password)
        if (!validateFields(email, password, confirmPassword))
            return;

        progressBar.setVisibility(View.VISIBLE); // show the loading state

        // validation checks passed, now register the user in the database

        register.setEnabled(false); // disable any button presses when registering user
        fAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                Context appContext = getApplicationContext();
                if (task.isSuccessful()) {
                    Toast.makeText(appContext, "User created.", Toast.LENGTH_SHORT).show();
                    addUserdataToCloud(username);
                    startActivity(new Intent(appContext, LobbyActivity.class));
                    finish();
                } else {
                    Exception exception = task.getException();
```

```
                if (exception != null)
                    Toast.makeText(appContext, "Error! " + exception.getMessage(),
Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(appContext, "Error: Couldn't create user.",
Toast.LENGTH_SHORT).show();
                progressBar.setVisibility(View.INVISIBLE);
            }
            register.setEnabled(true); // enable the button once finished registering
        }
    });
}

/**
 * Add username to the Firestore database to the "users" collection.
 *
 * @param username A String representation of the username field.
 */
public void addUserdataToCloud(String username) {
    // Create a new user with its username and email
    Map<String, Object> userData = new HashMap<>();
    userData.put("username", username);

    String uid = Objects.requireNonNull(fAuth.getCurrentUser()).getUid(); // impossible to
get nullptr exception because this code snippet will only be run if the user is successfully
created in fAuth
    DocumentReference documentReference = fStore.collection("users").document(uid);

    // Add a new document with the unique UID of the user
    documentReference.set(userData).addOnSuccessListener(new
OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void unused) {
            Log.d(TAG, "OnSuccess: user profile is created for uid: " + uid);
```

```
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                Log.w(TAG, "Error adding document", e);
            }
        });
    }

    /**
     * Check if user input in different fields such as email and password are valid or not (e.g
password length must be atleast 6 characters).
     *
     * @param email    The string representation of the email entered.
     * @param password The string representation of the password entered.
     * @return true if user input is OK, else otherwise.
     */
    private boolean validateFields(String email, String password, String confirmPassword) {
        boolean isValid = true;
        if (email.isEmpty()) {
            mEmail.setError("Email is Required.");
            isValid = false;
        }
        if (password.isEmpty()) {
            mPassword.setError("Password is Required.");
            isValid = false;
        } else if (password.length() < 6) {
            mPassword.setError("Password Must be >= 6 Characters");
            isValid = false;
        }
        if (!confirmPassword.equals(password)) {
            if (confirmPassword.isEmpty())
                mConfirmPassword.setError("Please Confirm your Password.");
            else
```

```
            mConfirmPassword.setError("Passwords Don't Match");
        isValid = false;
    }
    return isValid;
  }



}
```

SettingsActivity

```
package com.example.checkers;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.MenuItem;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SwitchCompat;

/**
 * This class handles the SettingsActivity operations.
 *
 * @author Tal Simhayev
 * @version 1.0
 */
public class SettingsActivity extends AppCompatActivity {

  public static final String SETTINGS_PREFS = "settings";

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_settings);

    SwitchCompat vibrator = findViewById(R.id.vibrateID);

    SharedPreferences settingsPrefs = getSharedPreferences(SETTINGS_PREFS,
MODE_PRIVATE);
    boolean isVibrate = settingsPrefs.getBoolean("vibrate", true);
    vibrator.setChecked(isVibrate);
```

```
        vibrator.setOnCheckedChangeListener((buttonView, isChecked) -> {
            SharedPreferences.Editor editor = getSharedPreferences(SETTINGS_PREFS,
MODE_PRIVATE).edit();
            editor.putBoolean("vibrate", isChecked);
            editor.apply();
        });

        ActionBar actionBar = getSupportActionBar();
        if (actionBar != null)
            actionBar.setDisplayHomeAsUpEnabled(true);
        else
            System.out.println("*********Error: actionBar is null! Can't set back arrow.");

    }

    /**
     * This function returns to the caller activity, and finishes this activity.
     *
     * @param item The item in the menu (when pressing the back button).
     * @return true if returning to MainActivity, else the super function's value.
     */
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.getItemId() == android.R.id.home) {
            this.finish();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

ScoresActivity

```
package com.example.checkers;


import androidx.appcompat.app.AppCompatActivity;


import android.os.Bundle;


/**
 * This class handles the ScoresActivity operations.
 *
 * @author Tal Simhayev
```

```
 * @version 1.0
 */
public class ScoresActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scores);
    }
}
```

XML

activity_game.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/checkersPortrait"
    android:layout_width="match_parent"
    android:layout_height="650dp"
    android:background="@color/green"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="20dp"
        android:layout_weight="1"
        android:orientation="horizontal">


        <TextView
            android:id="@+id/welcome"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight=".8"
            android:gravity="center_horizontal|center_vertical"
            android:text="Checkers"
            android:textColor="@color/lightwhite"
            android:textSize="50sp"
            android:textStyle="bold" />
```

```
</LinearLayout>

<!-- The actual board layout (matrix 8x8) -->
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="360dp"
    android:layout_weight="1"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/square00"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/lightwood" />

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square01"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/darkwood" />

            <!-- This ImageView (circle) will be for example a red or black piece -->
            <ImageView
                android:id="@+id/circle01"
                android:layout_width="42dp"
                android:layout_height="42dp"
                android:layout_gravity="center"
                android:layout_weight="1" />
        </FrameLayout>

        <ImageView
            android:id="@+id/square02"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
```

```
    android:layout_weight="1"
    android:background="@drawable/lightwood" />


<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle03"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square04"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />


<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square05"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
```

```
            android:id="@+id/circle05"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
        </FrameLayout>

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square06"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1"
                android:background="@drawable/lightwood" />


        </FrameLayout>

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square07"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/darkwood" />

            <ImageView
                android:id="@+id/circle07"
                android:layout_width="42dp"
                android:layout_height="42dp"
                android:layout_gravity="center"
                android:layout_weight="1" />
        </FrameLayout>

    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
```

```
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square10"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle10"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
    </FrameLayout>


    <ImageView
      android:id="@+id/square11"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle12"
```

```
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
    </FrameLayout>


    <ImageView
      android:id="@+id/square13"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square14"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle14"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />

    </FrameLayout>

    <ImageView
      android:id="@+id/square15"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">
```

```
        <ImageView
            android:id="@+id/square16"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle16"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
    </FrameLayout>

    <ImageView
        android:id="@+id/square17"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="@drawable/lightwood" />
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">


    <ImageView
        android:id="@+id/square20"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="@drawable/lightwood" />

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/square21"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle21"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square22"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />


<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square23"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle23"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square24"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />
```

```
<FrameLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_gravity="center"
   android:layout_weight="1">

   <ImageView
      android:id="@+id/square25"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

   <ImageView
      android:id="@+id/circle25"
      android:layout_width="42dp"
      android:layout_height="42dp"
      android:layout_gravity="center"
      android:layout_weight="1" />
</FrameLayout>

<ImageView
   android:id="@+id/square26"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_weight="1"
   android:background="@drawable/lightwood" />


<FrameLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_gravity="center"
   android:layout_weight="1">

   <ImageView
      android:id="@+id/square27"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

   <ImageView
      android:id="@+id/circle27"
      android:layout_width="42dp"
      android:layout_height="42dp"
```

```
            android:layout_gravity="center"
            android:layout_weight="1" />
        </FrameLayout>
    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="horizontal">

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square30"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/darkwood" />

            <ImageView
                android:id="@+id/circle30"
                android:layout_width="42dp"
                android:layout_height="42dp"
                android:layout_gravity="center"
                android:layout_weight="1" />
        </FrameLayout>


        <ImageView
            android:id="@+id/square31"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/lightwood" />

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square32"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle32"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square33"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />

<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square34"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle34"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square35"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
```

```
        android:background="@drawable/lightwood" />

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/square36"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle36"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
    </FrameLayout>

    <ImageView
        android:id="@+id/square37"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="@drawable/lightwood" />


</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">


    <ImageView
        android:id="@+id/square40"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="@drawable/lightwood" />
```

```
<FrameLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_gravity="center"
   android:layout_weight="1">

   <ImageView
      android:id="@+id/square41"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

   <ImageView
      android:id="@+id/circle41"
      android:layout_width="42dp"
      android:layout_height="42dp"
      android:layout_gravity="center"
      android:layout_weight="1" />
</FrameLayout>


<ImageView
   android:id="@+id/square42"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_weight="1"
   android:background="@drawable/lightwood" />


<FrameLayout
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:layout_gravity="center"
   android:layout_weight="1">

   <ImageView
      android:id="@+id/square43"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

   <ImageView
      android:id="@+id/circle43"
      android:layout_width="42dp"
      android:layout_height="42dp"
      android:layout_gravity="center"
```

```
        android:layout_weight="1" />
    </FrameLayout>

    <ImageView
      android:id="@+id/square44"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />


    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square45"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle45"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
    </FrameLayout>


    <ImageView
      android:id="@+id/square46"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />


    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
```

```
            android:id="@+id/square47"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle47"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
    </FrameLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/square50"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle50"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
    </FrameLayout>


    <ImageView
        android:id="@+id/square51"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
```

```
    android:background="@drawable/lightwood" />

  <FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
      android:id="@+id/square52"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

    <ImageView
      android:id="@+id/circle52"
      android:layout_width="42dp"
      android:layout_height="42dp"
      android:layout_gravity="center"
      android:layout_weight="1" />
  </FrameLayout>


  <ImageView
    android:id="@+id/square53"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />

  <FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
      android:id="@+id/square54"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_weight="1"
      android:background="@drawable/darkwood" />

    <ImageView
      android:id="@+id/circle54"
      android:layout_width="42dp"
      android:layout_height="42dp"
```

```
            android:layout_gravity="center"
            android:layout_weight="1" />
        </FrameLayout>


        <ImageView
            android:id="@+id/square55"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/lightwood" />

        <FrameLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_gravity="center"
            android:layout_weight="1">

            <ImageView
                android:id="@+id/square56"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:background="@drawable/darkwood" />

            <ImageView
                android:id="@+id/circle56"
                android:layout_width="42dp"
                android:layout_height="42dp"
                android:layout_gravity="center"
                android:layout_weight="1" />
        </FrameLayout>


        <ImageView
            android:id="@+id/square57"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/lightwood" />


    </LinearLayout>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
```

```
    android:orientation="horizontal">


    <ImageView
      android:id="@+id/square60"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />


    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square61"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle61"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
    </FrameLayout>


    <ImageView
      android:id="@+id/square62"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />


    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
```

```
        android:id="@+id/square63"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle63"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square64"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />

<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square65"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle65"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square66"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
        android:layout_weight="1"
        android:background="@drawable/lightwood" />


    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/square67"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle67"
            android:layout_width="42dp"
            android:layout_height="42dp"
            android:layout_gravity="center"
            android:layout_weight="1" />
    </FrameLayout>
</LinearLayout>

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:layout_weight="1"
    android:orientation="horizontal">

    <FrameLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/square70"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/darkwood" />

        <ImageView
            android:id="@+id/circle70"
            android:layout_width="42dp"
```

```
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
    </FrameLayout>

    <ImageView
      android:id="@+id/square71"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">

      <ImageView
        android:id="@+id/square72"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

      <ImageView
        android:id="@+id/circle72"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />

    </FrameLayout>


    <ImageView
      android:id="@+id/square73"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_weight="1"
      android:background="@drawable/lightwood" />

    <FrameLayout
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:layout_gravity="center"
      android:layout_weight="1">
```

```
    <ImageView
        android:id="@+id/square74"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle74"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square75"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1"
    android:background="@drawable/lightwood" />

<FrameLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center"
    android:layout_weight="1">

    <ImageView
        android:id="@+id/square76"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:background="@drawable/darkwood" />

    <ImageView
        android:id="@+id/circle76"
        android:layout_width="42dp"
        android:layout_height="42dp"
        android:layout_gravity="center"
        android:layout_weight="1" />
</FrameLayout>


<ImageView
    android:id="@+id/square77"
    android:layout_width="fill_parent"
```

```
            android:layout_height="fill_parent"
            android:layout_weight="1"
            android:background="@drawable/lightwood" />


        </LinearLayout>

    </LinearLayout>

    <TextView
        android:id="@+id/currentTurn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="center"
        android:textColor="@color/lightwhite"
        android:textSize="20sp" />

    <Button
        android:id="@+id/forfeit_button"
        android:layout_width="wrap_content"
        android:layout_height="48dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="1dp"
        android:layout_marginBottom="50dp"
        android:gravity="center_horizontal|center_vertical"
        android:text="Forfeit"
        android:textSize="12sp"
        android:visibility="invisible" />

</LinearLayout>
```

activity_lobby.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
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/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    android:fitsSystemWindows="true"
```

```
tools:context=".LobbyActivity">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolBar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/purple_500"
        android:minHeight="?attr/actionBarSize" />


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="@string/checkers_multiplayer"
        android:textSize="35sp"
        android:textStyle="bold" />

    <ListView
        android:id="@+id/listViewPlayers"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>

<com.google.android.material.navigation.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
```

```
    app:headerLayout="@layout/nav_header"
    app:menu="@menu/navigation_menu" />


</androidx.drawerlayout.widget.DrawerLayout>
```

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:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/background"
  android:gravity="center_horizontal"
  android:orientation="vertical"
  tools:context=".LoginActivity">

  <Button
    android:id="@+id/auth"
    android:layout_width="150dp"
    android:layout_height="82dp"
    android:layout_marginBottom="100dp"
    android:text="@string/login_text"
    android:textSize="30sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.494"
    app:layout_constraintStart_toStartOf="parent" />

  <EditText
    android:id="@+id/editTextEmail"
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginEnd="16dp"
    android:layout_marginBottom="150dp"
    android:ems="10"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:minHeight="48dp"
    app:layout_constraintBottom_toTopOf="@+id/auth"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent" />

<EditText
    android:id="@+id/editTextPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="Password"
    android:inputType="textPassword"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextEmail" />

<TextView
    android:id="@+id/textViewLoginPage"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="100dp"
```

```
        android:text="Login page"
        android:textSize="30sp"
        app:layout_constraintBottom_toTopOf="@+id/editTextEmail"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/textViewCreateAccount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="56dp"
        android:minHeight="48dp"
        android:text="New here? Create Account Here"
        android:textColor="#F3DE27"
        android:textSize="20sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/auth" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:indeterminateTint="#FF0000"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/auth"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextPassword" />

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

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/background"
   android:gravity="center_horizontal"
   android:orientation="vertical"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="30dp"
      android:text="@string/app_name"
      android:textColor="@color/black"
      android:textSize="50sp" />

   <Button
      android:id="@+id/login"
      android:layout_width="170dp"
      android:layout_height="wrap_content"
      android:layout_marginTop="150dp"
      android:text="@string/login_text"
      android:textSize="25sp" />

   <Button
      android:id="@+id/settings"
      android:layout_width="170dp"
      android:layout_height="wrap_content"
      android:layout_marginTop="20dp"
      android:text="@string/settings_text"
      android:textSize="25sp" />
</LinearLayout>
```

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:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/background"
   android:gravity="center_horizontal"
   android:orientation="vertical"
   tools:context=".RegisterActivity">


   <TextView
      android:id="@+id/textViewRegisterPage"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="32dp"
      android:text="Register Page"
      android:textSize="30sp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.511"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

   <EditText
      android:id="@+id/editTextName"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_marginStart="16dp"
      android:layout_marginEnd="16dp"
      android:ems="10"
```

```
    android:hint="Name"
    android:inputType="textPersonName"
    android:minHeight="48dp"
    app:layout_constraintBottom_toTopOf="@+id/editTextEmail"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewRegisterPage"
    app:layout_constraintVertical_bias="0.75" />

<EditText
    android:id="@+id/editTextEmail"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="104dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="Email"
    android:inputType="textEmailAddress"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textViewRegisterPage" />

<EditText
    android:id="@+id/editTextPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="20dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
```

```
    android:hint="Password"
    android:inputType="textPassword"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextEmail" />

<EditText
    android:id="@+id/editTextConfirmPassword"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:ems="10"
    android:hint="Confirm Password"
    android:inputType="textPassword"
    android:minHeight="48dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/editTextPassword" />

<Button
    android:id="@+id/ButtonRegister"
    android:layout_width="wrap_content"
    android:layout_height="0dp"
    android:layout_marginTop="112dp"
    android:text="Register"
    android:textSize="30sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/editTextConfirmPassword" />

    <TextView
        android:id="@+id/textViewLogin"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:text="Already registered? Login Here"
        android:textColor="#F3DE27"
        android:textSize="20sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ButtonRegister"
        app:layout_constraintVertical_bias="0.096" />

    <ProgressBar
        android:id="@+id/progressBar"
        style="?android:attr/progressBarStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="32dp"
        android:indeterminateTint="#FF0000"
        android:visibility="invisible"
        app:layout_constraintBottom_toTopOf="@+id/ButtonRegister"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editTextConfirmPassword"
        app:layout_constraintVertical_bias="1.0" />

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

activity_scores.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="@color/background"
   android:gravity="center_horizontal"
   android:orientation="vertical"
   tools:context=".ScoresActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/scores"
      android:textSize="50sp" />

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="50dp"
      android:textSize="30sp" />

</LinearLayout>
```

activity_settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:gravity="center_horizontal"
   android:background="@color/background"
```

```
    tools:context=".SettingsActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/settings_text"
        android:textSize="50sp"/>

    <androidx.appcompat.widget.SwitchCompat
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginStart="7dp"
        android:text="@string/vibrate_when_a_game_starts"
        android:textSize="17sp"
        android:id="@+id/vibrateID"/>
</LinearLayout>
```

nav_header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:gravity="bottom"
    android:padding="16dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark"
    android:background="#ff547c"
    android:layout_width="match_parent"
    android:layout_height="176dp">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher"
        android:contentDescription="TODO" />
```

```
    <TextView
        android:id="@+id/textviewUsername"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="8dp"
        android:text="username"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:id="@+id/textviewEmail"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="email"/>

</LinearLayout>
```

navigation_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:showIn="navigation_view">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_scores"
            android:icon="@drawable/ic_points"
            android:title="@string/my_points" />

        <item
            android:id="@+id/nav_settings"
            android:icon="@drawable/ic_settings"
            android:title="@string/settings" />

        <item
            android:id="@+id/nav_feedback"
            android:icon="@drawable/ic_feedback"
            android:title="@string/contact_developer" />

        <item
            android:id="@+id/nav_logout"
            android:icon="@drawable/ic_logout"
            android:title="@string/logout" />
```

```
    </group>
</menu>
```

strings.xml

```
<resources>
    <string name="app_name">Checkers</string>
    <string name="login_text">Login</string>
    <string name="settings_text">Settings</string>
    <string name="waiting_room">Waiting Room</string>
    <string name="scores">Scores</string>
    <!--to toggle the open close button
      of the navigation drawer-->
    <string name="nav_open">Open</string>
    <string name="nav_close">Close</string>
    <string name="online_players">Online players</string>
    <string name="my_points">My Points</string>
    <string name="settings">Settings</string>
    <string name="logout">Logout</string>
    <string name="contact_developer">Contact Developer</string>
    <string name="checkers_multiplayer">Checkers Multiplayer</string>
    <string name="vibrate_when_a_game_starts">Vibrate when a game starts</string>
    <string name="your_turn">It\'s your turn to play!</string>
    <string name="not_your_turn">Opponents turn.</string>
    <string name="email">Email</string>
    <string name="password">Password</string>
    <string name="login_page">Login page</string>
    <string name="new_here_create_account_here">New here? Create Account
Here</string>
    <string name="register_page">Register Page</string>
    <string name="name">Name</string>
    <string name="confirm_password">Confirm Password</string>
    <string name="register">Register</string>
    <string name="already_registered_login_here">Already registered? Login Here</string>
    <string name="username">username</string>
    <string name="android_icon">android_icon</string>
</resources>
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">

    <!-- Base application theme. -->

    <style name="Theme.Checkers"

parent="Theme.MaterialComponents.DayNight.DarkActionBar">

        <!-- Primary brand color. -->

        <item name="colorPrimary">@color/purple_500</item>``
```

```
        <item name="colorPrimaryVariant">@color/purple_700</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>

    <style name="Theme.AppCompat.NoActionBar">
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>
</resources>
```