Brain Battle  
מגיש: יהודה אליסף

ת.ז. 328193602

מורה: חגי סוויד

חלופה: תכנות לטלפונים ניידים

תאריך הגשה: 12.6.2023

תוכן

[מבוא 3](#_Toc137146000)

[הרקע לפרויקט 3](#_Toc137146001)

[תהליך המחקר 3](#_Toc137146002)

[טכנולוגיות 3](#_Toc137146003)

[אתגרים מרכזיים 3](#_Toc137146004)

[תיאור תחום הידע 4](#_Toc137146005)

[אובייקטים נחוצים 4](#_Toc137146006)

[סוגי נתונים 4](#_Toc137146007)

[פעולות על המידע 4](#_Toc137146008)

[ארכיטקטורה 5](#_Toc137146009)

[מסכי הפרויקט 5](#_Toc137146010)

[מסך הרשמה והתחברות (מסך הפתיחה) 5](#_Toc137146011)

[מסך יצירת משחק 6](#_Toc137146012)

[מסך קוד המשחק 7](#_Toc137146013)

[מסך הצטרפות למשחק 8](#_Toc137146014)

[מסך המשחק 9](#_Toc137146015)

[מסך סיום משחק 10](#_Toc137146016)

[מסך ניקוד 11](#_Toc137146017)

[מסך הגדרות 12](#_Toc137146018)

[תרשים מסכים 13](#_Toc137146019)

[13](#_Toc137146020)

[13](#_Toc137146021)

[מחלקות הפרויקט 14](#_Toc137146022)

[מימוש הפרויקט 15](#_Toc137146023)

# מבוא

## הרקע לפרויקט

הפרויקט Brain Battle (קרב מוחות) הוא משחק טריוויה ל2 שחקנים. השחקנים מתחרים ביניהם אונליין, עונים על שאלות טריוויה שונות וצוברים נקודות. בסיום המשחק, השחקנים יכולים לראות סטטיסטיקות שונות כמו מספר התשובות הנכונות והשגויות, ואת מצב הניקוד שלהם ביחס לשחקנים אחרים.

המשחק מיועד לטווח גילאים רחב, משום שיש אפשרות לבחור את רמת הקושי הרצויה (כמובן שרמת הקושי משפיעה גם על הניקוד).להערכתי, תחום הגילאים המומלץ הוא 10 ומעלה.

בחרתי ליצור דווקא משחק כי רציתי ליהנות כמה שיותר מיצירת הפרויקט, ויצירת משחקים מוסיפה גורם של הנאה לפיתוח. בחרתי דווקא בטריוויה כי אני אוהב את המשחק, ונהנה לבחון את הידע הכללי שלי בתחומים שונים.

## תהליך המחקר

כיום, ישנן אפליקציות טריוויה רבות בשוק, גם ל2 שחקנים וגם למספרי שחקנים שונים. עם זאת, זה לא מפריע לי, כי המטרה העיקרית שלי בפיתוח במשחק היא להעמיק את הידע שלי בפיתוח אפליקציות.

## טכנולוגיות

במהלך יצירת הפרויקט השתמשתי במספר טכנולוגיות שאינן חלק מתוכנית הלימודים:

1. HTTP – פרוטוקול תקשורת שמיועד לשליחת וקבלת מידע לאתרי אינטרנט ומהם. השתמשתי בHTTP כדי להוריד שאלות טריוויה ממסד נתונים בשם [Open Trivia Database](https://opentdb.com/), במטרה להשיג כמות גדולה של שאלות בנושאים מגוונים ורמות קושי שונות. פרמטרי כמו מספר השאלות, רמת הקושי וכו' נשלחים בHTTP בעזרת פרמטרי GET.
2. JSON Deserialization – השאלות במסד הנתונים מתקבלות בפורמט JSON. כדי להפוך את השאלות שהתקבלו לאובייקט Question (אובייקט שיצרתי שמייצג שאלה, עליו אפרט בהמשך) יש צורך בדה-סריאליזציה של המידע, כלומר הפיכה שלו מJSON לאובייקט קוד.
3. Firebase – שירות של גוגל שמאפשר מספר פיצ'רים כמו ניהול ההתחברות וההרשמה של משתמשים, ושרת אחסון בשם Firestore שמאפשר לשמור מידע בענן. השתמשתי בFirestore בשביל לשמור את נתונים המשחקים ולסנכרן אותם בזמן אמת, ובשביל לשמור את רשימת המשתמשים ופרטים שלהם.

## אתגרים מרכזיים

היו לי לא מעט אתגרים בזמן העבודה על הפרויקט.

בגלל שהמשחק נערך אונליין בין 2 שחקנים, ואף אחד מהשחקנים לא יכול לשחק בלי השני, הייתי צריך להשתמש ב2 מכשירי פלאפון. לכן, יצרתי אמולטור שמדמה מכשיר פלאפון, והרצתי את האפליקציה גם בו וגם במכשיר הפיזי.

קושי נוסף שהתמודדתי איתו הוא נגרם מכך שהמשחק דורש חיבור אינטרנט רציף. אם באמצע המשחק אחד השחקנים מתנתק, האפליקציה יכולה להיתקע או לקרוס. כדי לפתור את הבעיה, הוספתי לכל תקשורת עם הFirestore פונקציית OnFailure שקובעת מה לעשות במידה וחיבור האינטרנט קרס (למשל: הצגת הודעת שגיאה למשתמש וחזרה למסך הראשי). בנוסף, ההכנסתי את בקשת הHTTP לבלוק try catch, ובמידה וקרתה שגיאה – הודעתי למשתמש שלא ניתן להוריד את השאלות.

# תיאור תחום הידע

## אובייקטים נחוצים

Game: אובייקט הכולל את כל המידע של המשחק. כולל בתוכו את 2 השחקנים ורשימה של שאלות (על מבנה השחקנים והשאלות אפרט בהמשך), ואת הID של המשחק.

User: מייצג משתמש. כולל בתוכו את המידע של המשתמש (שם משתמש, ID וסטטיסטיקות של ניקוד).

Player: מייצג שחקן במשחק. יורש מUser. מוסיף למשתמש רשימה של השאלות הנכונות והשגויות שהוא ענה במשחק.

Question: מייצג שאלה. כולל בתוכו את התשובות השגויות והתשובה הנכונה, קטגוריית השאלה ורמת הקושי שלה.

## סוגי נתונים

בתוך מחלקת Game, השאלות שמורות ברשימה של אובייקטים מסוג Question.

במחלקת Question, התשובות שמורות ברשימה של סטרינגים, והאינדקס של התשובה הנכונה שמור בint. הקטגוריה ורמת הקושי שמורות כל אחת בenum משל עצמה:

enum DifficultyLevel{

EASY,

MEDIUM,

HARD

}

enum Category{

ALL,

GENERAL\_KNOWLEDGE,

SCIENCE,

COMPUTER\_SCIENCE

{

## פעולות על המידע

Game: יצירת משחק, סיום משחק, גטרים וסטרים.

User: יצירת משתמש, מחיקת משתמש, גטרים וסטרים.

Player: יצירת שחקן, חישוב של כמות התשובות הנכונות/שגויות במשחק, מתודת equals(), גטרים וסטרים.

Question: יצירת שאלה, גטרים וסטרים.

# ארכיטקטורה

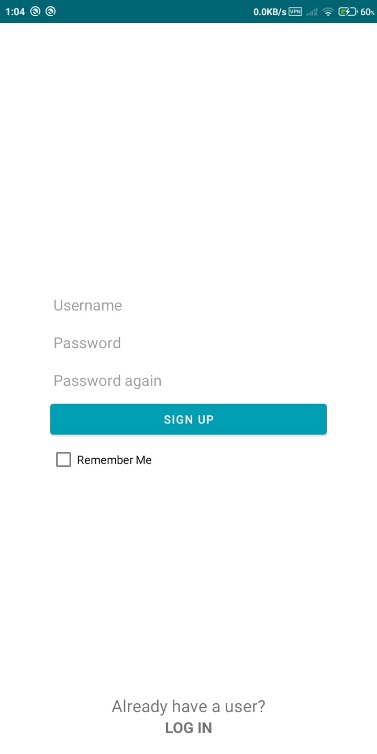
## מסכי הפרויקט

### מסך הרשמה והתחברות (מסך הפתיחה)

במצב התחברות, המסך מציג 2 EditTextים בשביל שם משתמש וסיסמא. במצב הרשמה, המסך מציג EditText נוסף שמיועד להכנסת הסיסמא שוב.

Button

מעבר בין המצבים מתבצע באמצעות לחיצה על הlabel שבתחתית המסך (SIGN UP/LOG IN).

תמונה שמכילה טקסט, צילום מסך, תוכנה, מערכת הפעלה

התיאור נוצר באופן אוטומטי

Button

הרשמה

בודק אם השם משתמש והסיסמא תקינים, ואם כן יוצר משתמש ומעביר למסך יצירת משחק

Button

התחברות

בודק בפיירבייס אם השם משתמש והסיסמא נכונים, ואם כן מעביר למסך יצירת משחק

EditText

הכנסת סיסמא פעם נוספת

EditTexts

הכנסת שם משתמש וסיסמא

CheckBox

קובע האם לשמור את שם המשתמש והסיסמא בהתחברות הבאה

TextView (Clickable)

מעבר ממצב הרשמה למצב התחברות ולהיפך

### מסך יצירת משחק

מאפשר למשתמש ליצור משחק חדש, ולבחור את הקטגוריה, רמת הקושי ומספר השאלות.

אם בפתיחת האפליקציה המשתמש כבר מחובר, זה יהיה מסך הפתיחה.

Buttons

בחירת הקטגוריה של המשחק מבין הבאות:

כל הקטגוריות, ידע כללי, מדע, מדעי המחשב

מסך זה הוא Fragment בתוך הActivity של בתפריט הראשי,

מעבר לFragmentים אחרים מתבצע באמצעות לחיצה על כפתורי הניווט למטה

או החלקה ימינה ושמאלה.

תמונה שמכילה טקסט, צילום מסך, תוכנה, דף אינטרנט

התיאור נוצר באופן אוטומטי

Buttons

בחירת רמת הקושי של המשחק:

קל, בינוני או קשה

TextView

הצגת כמות השאלות:

מינימום 2, מקסימום 20.

Buttons

בחירת כמות השאלות:

כפתור להגדלה וכפתור להקטנה של מספר השאלות

Buttons

בחירת רמת הקושי של המשחק:

קל, בינוני או קשה

Button

מעבר למסך ניקוד

Button

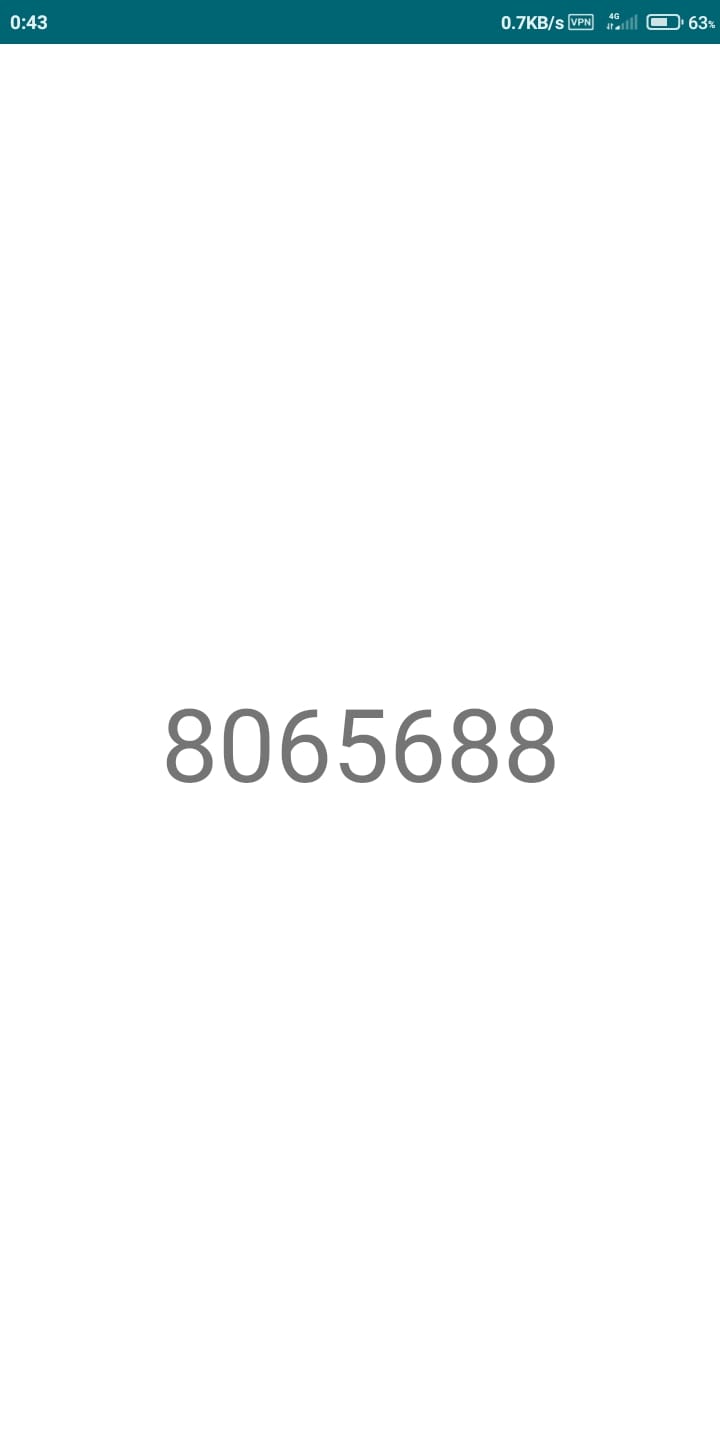
מעבר למסך הצטרפות למשחק

Button

מעבר למסך הגדרות

## מסך קוד המשחק

לאחר יצירת המשחק, יופיע מסך ובו קוד אותו צריך המשתמש השני להכניס כדי להתחבר.

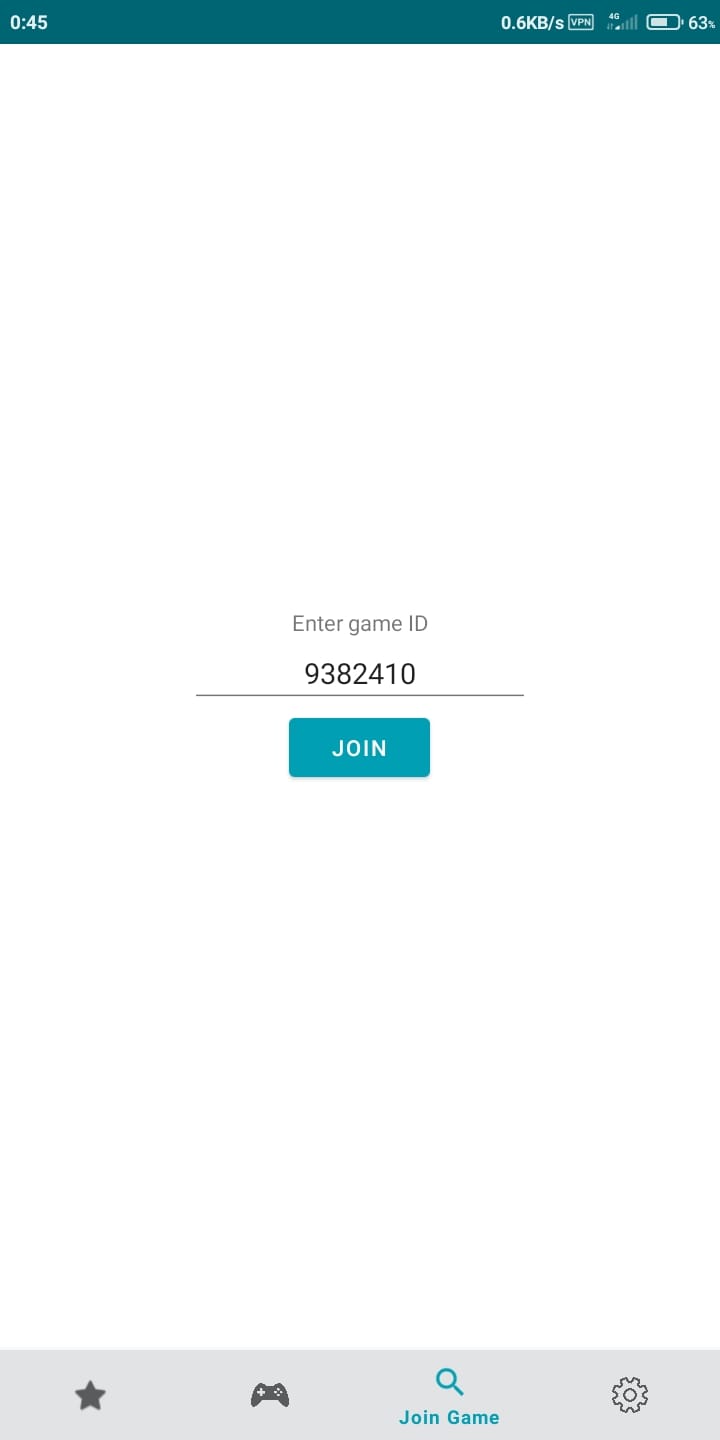


TextView

הצגת קוד המשחק

## מסך הצטרפות למשחק

אחרי שהשחקן הראשון יצר את המשחק, השחקן השני יכניס במסך זה את קוד המשחק כדי להצטרך אליו.



Button

הצטרפות למשחק

EditText

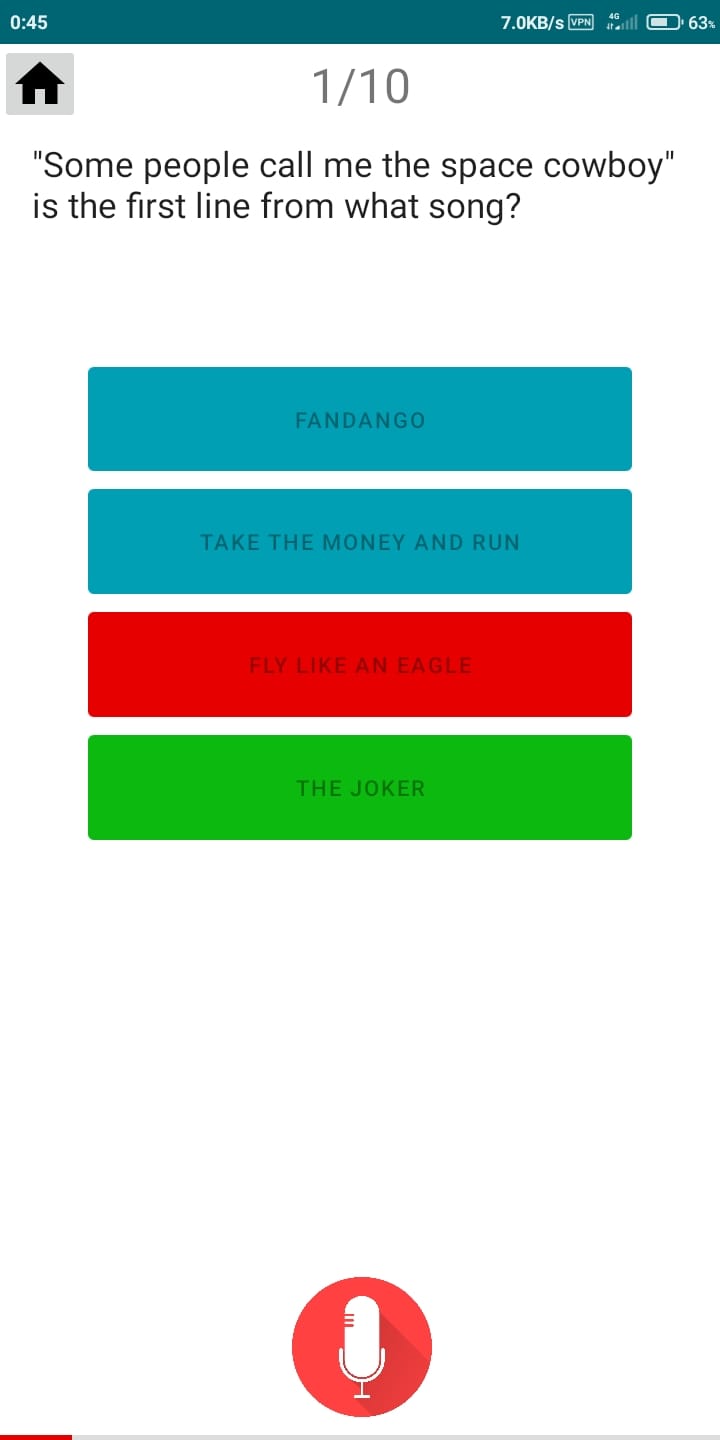
הכנסת קוד המשחק

## מסך המשחק

כששני השחקנים מחוברים, הם מועברים למסך זה, בו המשחק עצמו מתנהל.

Button

חזרה לתפריט הראשי



ImageButton

הקלטת תשובה

המשתמש צריך ללחוץ על הכפתור ולומר את מספר התשובה

(one, two, three, four)

הפיצ'ר עדיין לא ממומש

Canvas

הצגת ההתקדמות של היריב והאם הוא צדק או טעה (ירוק/אדום בהתאמה)

TextView

הצגת השאלה

TextView

הצגת מספר השאלה הנוכחית וסך השאלות

Buttons

כפתורי תשובות

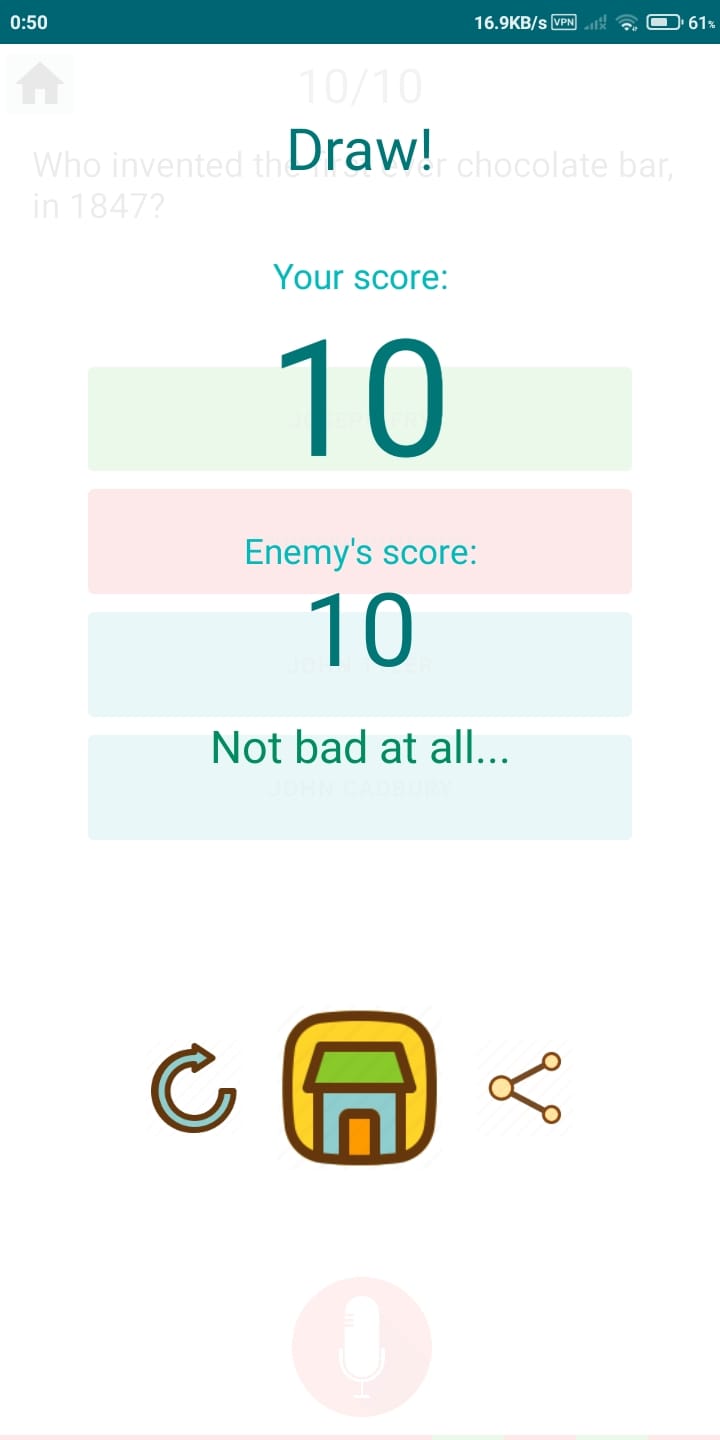
המשתמשים צריכים ללחוץ על התשובה הנכונה. לאחר מכן, הכפתור ייצבע באדום או ירוק בהתאם לתשובה

## מסך סיום משחק

בסוף המשחק, יופיע מסך זה, המציג את הניקוד ואת המנצח במשחק.

TextView

הצגת השחקן המנצח



ImageButton

שיתוף תוצאות משחק

ImageButton

חזרה למסך הבית

ImageButton

יצירת משחק חדש עם אותם פרמטרים (כמות שאלות וכו')

TextView

הצגת הניקוד של היריב

TextView

הצגת הניקוד שלך

## מסך ניקוד

במסך זה המשתמש יכול לראות את מצב הנקודות שלו ושל אחרים, וסטטיסטיקות נוספות.

TextView

סה"כ תשובות נכונות

TextView

סך הניקוד שלך

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

התיאור נוצר באופן אוטומטי

RecyclerView

הצגת הניקוד שלך מול משתמשים אחרים

ProgressBar

הצגת אחוז התשובות הנכונות מבין כל התשובות

TextView

סה"כ תשובות שגויות

## מסך הגדרות

במסך זה אפשר לערוך את הגדרות האפליקציה והגדרות המשתמש.

TextView

שם המשתמש

תמונה שמכילה טקסט, צילום מסך, תוכנה, גופן

התיאור נוצר באופן אוטומטי

ImageButton

שינוי שם המשתמש

ImageButton

מחיקת המשתמש

ProgressBar + ImageButton

שינוי עוצמת מוזיקת הרקע

שם המשתמש

## תרשים מסכים

## 

LoginActivity

MainMenuActivity

GameIdFragment

GameActivity

EndGameFragment

NewGameFragment

JoinGameFragment

SettingsFragment

ScoreFragment

## מחלקות הפרויקט



;;

;;

;;

# מימוש הפרויקט

## מחלקות מסכים (Fragment/Activity)

### LoginActivity

הערה: כל התכונות מוגדרות כprivate אלא אם נכתב אחרת.

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| static final int MIN\_PASSWORD\_LENGTH = 6 | קבוע: מספר הספרות המינימלי בסיסמא |
| Button loginButton | כפתור התחברות/יצירת משתמש |
| TextView toggleLoginModeLbl | כותרת המעבר ממצב התחברות למצב הרשמה |
| TextView toggleLoginModeLink | לייבל לחיץ (Clickable) למעבר ממצב התחברות למצב הרשמה ובחזרה |
| EditText usernameTxt | שורה להכנסת שם המשתמש |
| EditText passwordTxt | שורה להכנסת הסיסמא |
| EditText passwordAgainTxt | שורה להכנסת הסיסמא שוב (במצב הרשמה) כדי לוודא שאין שגיאת הקלדה |
| CheckBox rememberMeCb | קובע האם לזכור את שם המשתמש והסיסמא להתחברות הבאה ולהתחבר באופן אוטומטי |
| TextView loginStatusLbl | מודיע אם ההתחברות/הרשמה הצליחה. אם לא כותב מה השגיאה |
| Mode mode | המצב הנוכחי – התחברות או הרשמה  enum Mode {  LOGIN,  SIGNUP  { |

פעולות עיקריות

התחברות:

private void login(String username, String password) {

**בדיקה אם שם המשתמש והסיסמא תקינים**

**(אורך תקין, אין תווים לא חוקיים וכו')**

if(!validateUsernameAnsPassword(username, password))

return;

**אם הם תקינים - התחברות**

**פנייה לפיירבייס לבקשת התחברות. בסוף ההתחברות נקראת פעולת onComplete**

FirebaseAuth.getInstance().signInWithEmailAndPassword(username, password).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

if(task.isSuccessful()){

**התחברות הצליחה**

**מעבר לתפריט הראשי (מסך יצירת משחק)**

startMainMenuActivity();

//TODO: if user isn't listed in "users" list, then create user and add it

//can happen if user was signed in and had an connection error before he was added to list

}

Else

**התחברות נכשלה**

**צריך להודיע למשתמש מה הייתה השגיאה**

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

הרשמה:

private void signup(String username, String password, String passwordAgain) {

**בדיקה אם 2 הסיסמאות שהוכנסו זהות**

if(!password.equals(passwordAgain)){

**אם הן שונות, מודיעים למשתמש ויוצאים מהפעולה**

loginStatusLbl.setText("Passwords doesn't match");

passwordTxt.setText("");

passwordAgainTxt.setText("");

setWrongColors(passwordTxt);

setWrongColors(passwordAgainTxt);

validateUsernameAnsPassword(username, password);

return;

}

**בודקים אם שם המשתמש והסיסמא תקינים**

**אם לא, יוצאים מהפעולה**

if(!validateUsernameAnsPassword(username, password))

return;

String email = username;

**פנייה לפיירבייס לבקשת יצירת משתמש**

FirebaseAuth.getInstance().createUserWithEmailAndPassword(email, password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {

**אחרי ביצוע הבקשה בודקים אם ההרשמה הצליחה**

if(task.isSuccessful()){

//add user to users list

**אם היא הצליחה, מוסיפים את המשתמש לרשימת המשתמשים**

User user = new User(username, FirebaseAuth.getInstance().getUid(), 0, 0, 0);

FirebaseFirestore.getInstance(). collection(GameActivity.USERS\_COLLECTION\_PATH).document(email).set(user).

addOnCompleteListener(new OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful())

**אחרי שמוסיפים את המשתמש לרשימת המשתמשים, עוברים לתפריט הראשי**

startMainMenuActivity();

else

**אם הרשמת המשתמש/ההוספה שלו לרשימת המשתמשים נכשלה, מודיעים למשתמש מה הייתה השגיאה**

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

else

loginStatusLbl.setText(task.getException().getMessage());

}

});

}

### MainMenuActivity

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| GestureDetectorCompat swipeDetector | רכיב שמזהה החלקות על המסך. אם המשתמש החליק ימינה או שמאלה לאורך מרחק מסויים, עוברים מסך ימינה/שמאלה בהתאמה |
| static final int MIN\_SWIPE\_LENGTH = 150 | המרחק המינימלי של החלקה אופקית בשביל לעבור מסך. אם המשתמש החליק על המסך למרחק קטן יותר, לא יקרה כלום |
| BottomNavigationView navigationView | תפריט בתחתית המסך |
| NewGameFragment newGameFragment | ארבעת הפרגמנטים אליהם ניתן לעבור בתפריט  ביצירת המסך, הפרגמנט המוצג הוא  NewGameFragment |
| JoinGameFragment joinGameFragment |
| ScoreFragment scoreFragment |
| SettingsFragment settingFragment |

פעולות

מעבר בין מסכים בהחלקה:

public boolean onFling(@NonNull MotionEvent e1, @NonNull MotionEvent e2, float velocityX, float velocityY) {

**מציאת הפרגמנט שמוצג עכשיו**

Fragment currentFragment = getSupportFragmentManager().findFragmentById(R.id.mainFragmentContainer);

FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();

**בדיקה אם ההחלקה מספיק גדולה בשביל לעבור מסך**

if(Math.abs(e2.getX() - e1.getX()) > MIN\_SWIPE\_LENGTH){

//swipe

**בדיקה אם ההחלקה הייתה ימינה או שמאלה**

if(e2.getX() > e1.getX()){

//right swipe

**בהחלקה ימינה: מעבר מסך אחד שמאלה**

if(currentFragment instanceof JoinGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, newGameFragment).commit();

navigationView.setSelectedItemId(R.id.newGameFragment);

}

else if(currentFragment instanceof NewGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, scoreFragment).commit();

navigationView.setSelectedItemId(R.id.scoreFragment);

}

else if(currentFragment instanceof SettingsFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, joinGameFragment).commit();

navigationView.setSelectedItemId(R.id.joinGameFragment);

}

}

else{

**בהחלקה שמאלה: מעבר מסך אחד ימינה**

//left swipe

if(currentFragment instanceof ScoreFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, newGameFragment).commit();

navigationView.setSelectedItemId(R.id.newGameFragment);

}

else if(currentFragment instanceof NewGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, joinGameFragment).commit();

navigationView.setSelectedItemId(R.id.joinGameFragment);

}

else if(currentFragment instanceof JoinGameFragment){

fragmentTransaction.replace(R.id.mainFragmentContainer, settingFragment).commit();

navigationView.setSelectedItemId(R.id.settingsFragment);

}

}

}

**החזרה: הפעולה הסתיימה בהצלחה**

return true;

}

הערה: מבחינה תכנותית, נכון יותר לשים את הFragmentים במערך. בכתיבת הקוד, היו בהתחלה רק 2 מסכים, אח"כ 3 ובסוף 4, ולכן עשיתי בדרך הזו.

### NewGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| static final float UNSELECTED\_BUTTON\_ALPHA = (float) 0.4 | אם בוחרים כפתור (למשל: כפתור של רמת קושי קלה), כל הכפתורים האחרים שאיתו (רמת קושי בינונית וקשה) נעשים שקופים קצת. זו רמת השקיפות שלהם. |
| static final int MIN\_QUESTION\_COUNT = 2 | מספר השאלות המינימלי |
| static final int MAX\_QUESTION\_COUNT = 15 | מספר השאלות המקסימלי |
| Button allCategoriesBtn | כפתור בחירת קטגוריה: כל הקטגוריות |
| Button generalKnowledgeCategoryBtn | כפתור בחירת קטגוריה: ידע כללי |
| Button scienceCategoryBtn | כפתור בחירת קטגוריה: מדע |
| Button computerScienceCategoryBtn | כפתור בחירת קטגוריה: מדעי המחשב |
| Category category | הקטגוריה שנבחרה  enum Category{  ALL,  GENERAL\_KNOWLEDGE,  SCIENCE,  COMPUTER\_SCIENCE  { |
| Button easyDifficultyLevelBtn | כפתור בחירת רמת קושי: קל |
| Button mediumDifficultyLevelBtn | כפתור בחירת רמת קושי: בינוני |
| Button hardDifficultyLevelBtn | כפתור בחירת רמת קושי: קשה |
| DifficultyLevel difficultyLevel | רמת הקושי שנבחרה  enum DifficultyLevel{  EASY,  MEDIUM,  HARD  { |
| Button questionCountDecBtn | הקטנת כמות השאלות למשחק ב1 |
| Button questionCountIncBtn | הגדלת כמות השאלות למשחק ב1 |
| TextView questionCountValueLbl | הצגת מספר השאלות שנבחר למשתמש |
| int questionCount | מספר השאלות שנבחר |
| Button playBtn | כפתור התחלת המשחק |

פעולות

הפעולה שנקראת כשנלחץ כפתור במסך:

public void onClick(View v) {

**בדיקה איזה כפתור נלחץ**

switch (v.getId()){

**אם זה אחד מכפתורי בחירת רמת הקושי, בוחרים את רמת הקושי הזו.**

**קוראת לפעולה setDifficultyLevel, שבהמשך אפרט מה היא עושה.**

case R.id.easyDifficultyLevelBtn:

setDifficultyLevel(EASY);

break;

case R.id.mediumDifficultyLevelBtn:

setDifficultyLevel(MEDIUM);

break;

case R.id.hardDifficultyLevelBtn:

setDifficultyLevel(HARD);

break;

**אם זה אחד מכפתורי בחירת הקטגוריה: מסמנים את הקטגוריה הזו.**

case R.id.allCategoriesBtn:

setCategory(ALL);

break;

case R.id.generalKnowledgeCategoryBtn:

setCategory(GENERAL\_KNOWLEDGE);

break;

case R.id.scienceCategoryBtn:

setCategory(SCIENCE);

break;

case R.id.computerScienceCategoryBtn:

setCategory(COMPUTER\_SCIENCE);

break;

**אם נלחץ כפתור הוספת/הורדת שאלה, מוסיפים או מורידים שאלה בהתאמה.**

case R.id.questionCountDecBtn:

questionCountDec();

break;

case R.id.questionCountIncBtn:

questionCountInc();

break;

**אם נלחץ כפתור התחלת משחק, עוברים למסך יצירת משחק**

case R.id.playBtn:

**משתמשים בIntent כדי להעביר למסך את רמת הקושי, מספר השאלות והקטגוריה**

Intent intent = new Intent(getActivity(), GameActivity.class);

int extras[] = new int[3];

extras[QUESTIONS\_COUNT\_INDEX] = questionCount;

extras[DIFFICULTY\_LEVEL\_INDEX] = difficultyLevel.ordinal();

extras[CATEGORY\_INDEX] = category.ordinal();

intent.putExtra(GameActivity.NEW\_GAME\_EXTRAS, extras);

intent.putExtra(IS\_NEW\_GAME\_EXTRA, true);

**מתחילים את GameActivity**

startActivity(intent);

getActivity().finish();

}

{

פעולה שקובעת את רמת הקושי לרמת הקושי שנבחרה:

void setDifficultyLevel(DifficultyLevel newDifficultyLevel){

**מאפסים את אחוז השקיפות של כל כפתורי קביעת רמת הקושי**

easyDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

mediumDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

hardDifficultyLevelBtn.setAlpha(UNSELECTED\_BUTTON\_ALPHA);

**בודקים מה רמת הקושי שנבחרה מסמנים את הכפתור שלה**

**וקובעים את רמת הקושי להיות זו שנבחרה.**

difficultyLevel = newDifficultyLevel;

switch (newDifficultyLevel){

case EASY:

easyDifficultyLevelBtn.setAlpha(1);

break;

case MEDIUM:

mediumDifficultyLevelBtn.setAlpha(1);

break;

case HARD:

hardDifficultyLevelBtn.setAlpha(1);

break;

}

{

}

### JoinGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| public static final int GAME\_ID\_LENGTH = 7 | אורך הID של משחק  מוגדר כpublic כי גם ביצירת משחק צריך לדעת את האורך של הID |
| EditText gameIdTxt | שורה בה כותבים את הID |
| Button joinGameBtn | הצטרפות למשחק |

פעולות

הצטרפות למשחק

private void joinGame(){

String id = gameIdTxt.getText().toString();

int intId;

try{

**הפיכת הID לint.**

intId = Integer.parseInt(id);

**אם אי אפשר להפוך לint, הID לא חוקי (ID יכול לכלול רק מספרים).**

if(id.length() != GAME\_ID\_LENGTH)

**אם אי אפשר להפוך לint, הID לא חוקי (ID יכול לכלול רק מספרים).**

throw new Exception();

}

**ID לא חוקי. מודיעים למשתמש שהID לא חוקי.**

catch (Exception e){

Toast.makeText(getContext(), "Invalid ID entered!", Toast.LENGTH\_SHORT).show();

gameIdTxt.setText("");

return;

}

**אם הID חוקי, עוברים למסך המשחק.**

**בintent, מודיעים שהמשתמש הצטרף למשחק (ולא יצר אותו),**

**ומעבירים את הID של המשחק.**

Intent intent = new Intent(getActivity(), GameActivity.class);

intent.putExtra(GameActivity.IS\_NEW\_GAME\_EXTRA, false);

intent.putExtra(GameActivity.GAME\_ID\_EXTRA, intId);

startActivity(intent);

}

{

### ScoreFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| RecyclerView scoreRv | רשימת המשתמשים |
| CircularProgressBar successPercentagePb | סרגל התקדמות מעגלי, שמציג כמה אחוז מהתשובות נכונות וכמה שגויות |
| TextView totalScoreLbl | לייבל שמציג את הנקודות של השחקן |
| TextView totalCorrectLbl | לייבל שמציג את סה"כ התשובות הנכונות |
| TextView totalWrongLbl | לייבל שמציג את סה"כ התשובות השגויות |
| TextView successPercentageLbl | לייבל שמציג את אחוז התשובות הנכונות |
| ArrayList<User> users | רשימת משתמשים |
| User currentUser | המשתמש הנוכחי |

פעולות

**יצירת הפרגמנט וקבלת נתוני המשתמש מFireStore**

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

**קביעת השדות של המחלקה לערך שלהם בקובץ הlayout**

// Inflate the layout for this fragment

View v = inflater.inflate(R.layout.fragment\_score, container, false);

scoreRv = v.findViewById(R.id.scoreRv);

successPercentagePb = v.findViewById(R.id.successPercentagePb);

totalScoreLbl = v.findViewById(R.id.totalScoreLbl);

totalCorrectLbl = v.findViewById(R.id.totalCorrectLbl);

totalWrongLbl = v.findViewById(R.id.totalWrongLbl);

successPercentageLbl = v.findViewById(R.id.successPercentageLbl);

**הצגת פרגמנט טעינה**

Fragment loadingFragment = new LoadingFragment();

getChildFragmentManager().beginTransaction().replace(R.id.scoreLayout, loadingFragment).commit();

**הורדת רשימת המשתמשים מפיירסטור**

//fetch user list

FirebaseFirestore.getInstance().

collection(GameActivity.USERS\_COLLECTION\_PATH).get().addOnFailureListener(new OnFailureListener() {

@Override

**במקרה שהורדת הרשימה נכשלה – הצגת הודעת שגיאה למשתמש**

**וחזרה למסך יצירת משחק**

public void onFailure(@NonNull Exception e) {

Toast.makeText(getContext(), "Connection error!", Toast.LENGTH\_SHORT).show();

//back to main manu

FragmentTransaction ft = getChildFragmentManager().beginTransaction();

ft.replace(R.id.mainFragmentContainer, new NewGameFragment()).commit();

}

}).addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {

@Override

public void onSuccess(QuerySnapshot queryDocumentSnapshots) {

**במקרה שההורדה הצליחה – אתחול רשימת המשתמשים לרשימה שהתקבלה.**

ArrayList<User> fetchedUserList = new ArrayList<>();

for(DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments())

if(documentSnapshot.exists())

fetchedUserList.add(documentSnapshot.toObject(User.class));

users = fetchedUserList;

**מציאת המשתמש הנוכחי לצורך הצגת הסטטיסטיקות שלו**

//find current user

for(User user : users)

if(user.getUid().equals(FirebaseAuth.getInstance().getUid()))

currentUser = user;

if(currentUser == null)

**אם המשתמש הנוכחי לא קיים (לא צריך לקרות) – מציגים הודעת שגיאה.**

Toast.makeText(getContext(), "Current user not found!", Toast.LENGTH\_SHORT).show();

**הסתרת מסך הטעינה** getChildFragmentManager().beginTransaction().hide(loadingFragment).commit();

**קריאה לפעולה שמציגה את הסטטיסטיקות על המסך (אפרט עליה בהמשך)**

showScore();

}

});

return v;

}

הצגת הסטטיסטיקות על המסך:

private void showScore() {

**הצגת הניקוד, מספר התשובות הנכונות והשגויות של המשתמש הנוכחי**

totalScoreLbl.setText(Integer.toString(currentUser.getScore()));

totalCorrectLbl.setText(Integer.toString(currentUser.getTotalCorrect()));

totalWrongLbl.setText(Integer.toString(currentUser.getTotalWrong()));

**הצגת אחוז התשובות הנכונות**

//show correct percentage progress bar

successPercentagePb.setProgressBarColor(MyColor.CORRECT\_GREEN);

successPercentagePb.setProgressBarWidth(15);

successPercentagePb.setBackgroundProgressBarColor(MyColor.WRONG\_RED);

successPercentagePb.setBackgroundProgressBarWidth(10);

int totalAnswers = currentUser.getTotalCorrect() + currentUser.getTotalWrong();

int progress;

if(totalAnswers > 0)

progress = (100 \* currentUser.getTotalCorrect()) / totalAnswers;

else

//avoid division by zero

progress = 0;

successPercentagePb.setProgressWithAnimation(progress, (long)1000);

successPercentageLbl.setText(Integer.toString(progress) + “%”);

**הצגת רשימת המשתמשים והניקוד שלהם**

//show user list

users.sort((o1, o2) -> (int)(o2.getScore() - o1.getScore()));

ScoreListAdapter scoreListAdapter = new ScoreListAdapter(getContext(), users);

scoreRv.setAdapter(scoreListAdapter);

scoreRv.setLayoutManager(new LinearLayoutManager(getContext()));

{

### GameIdFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| TextView startGameGameIdLbl | מציג למשתמש את הID של המשחק, כדי שהשחקן השני ידע להתחבר אליו |

פעולות

אתחול המסך:

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

View view = inflater.inflate(R.layout.fragment\_game\_id, container, false);

**מציאת הלייבל שמציגה את הID**

startGameGameIdLbl = view.findViewById(R.id.startGameGameIdLbl);

**מציאת הID של המשחק**

GameViewModel gameViewModel = new ViewModelProvider(getActivity()).get(GameViewModel.class);

int gameId = gameViewModel.getGame().getId();

**הצגת הID למשתמש**

startGameGameIdLbl.setText(Integer.toString(gameId));

return view;

{

### GameActivity

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
| public static final String GAMES\_COLLECTION\_PATH = "games" | השם של אוסף המשחקים בפיירסטור |
| public static final String USERS\_COLLECTION\_PATH = "users" | השם של אוסף המשתמשים בפיירסטור |
| public static final String NEW\_GAME\_EXTRAS = "extras" | שם הextra בintent שכולל פרמטרים של המשחק החדש – קטגוריה, רמת קושי ומספר שאלות, כמערך של int. |
| public static final String IS\_NEW\_GAME\_EXTRA = "isNewGame" | שם הextra בintent שאומר למחלקה האם צריך ליצור משחק חדש (כלומר – האם המשתמש הזה יוצר את המשחק או מתחבר למשחק קיים). |
| public static final String GAME\_ID\_EXTRA = "gameId" | שם הextra בintent שכולל את הID של המשחק (בהתחברות למשחק קיים) |
| public static final int QUESTIONS\_COUNT\_INDEX = 0 | בAsyncTask, הפרמטרים מתקבלים במערך שלIntegerים.  אלו האינדקסים של פרמטרים שונים בGetQuestionsAsync.  בהמשך אראה את אופן השימוש בהם. |
| public static final int DIFFICULTY\_LEVEL\_INDEX = 1 |
| public static final int CATEGORY\_INDEX = 2 |
| TextView currentQuestionLbl | לייבל שמציגה את מספר השאלה הנוכחית מתוך כל השאלות |
| TextView questionLbl | לייבל שמציגה את גוף השאלה |
| Button[] answerButtons | 4 כפתורים שעליהם לוחצים כדי לבחור בתשובה |
| ImageButton homeImgBtn | כפתור חזרה למסך הבית |
| Canvas pbCanvas | קנבס שמציג את ההתקדמות של היריב, ואת התשובות הנכונות והשגויות שלו |
| ImageView progressImg | מקום בו נמצא הקנבס הנ"ל |
| Bitmap progressBitmap | ביטמאפ שנצרך בשביל הקנבס (בהמשך אראה למה צריך את שלושתם) |
| ImageButton recordImgBtn | כפתור הקלטת תשובה (עדיין לא ממומש) |
| GameViewModel gameVM | רכיב ששומר נתונים של המשחק |
| Fragment loadingFragment | פרגמנט שמראה את טעינת המסך |
| Fragment gameIdFragment | פרגמנט שמציג את הID של המשחק לפני ההתחלה, כדי שהמשתמש השני יוכל להתחבר |
| FirebaseFirestore firestore | האובייקט המרכזי של פיירסטור שמאפשר חיבור לDB. |

פעולות

הורדת השאלות מהשרת, ביצירת משחק (משתמש בAsyncTask, מחלקה שמאפשרת להריץ פעולות ברקע תוך כדי ריצת הת'רד הראשי):

private class GetQuestionsAsync extends AsyncTask<Integer, Integer, ArrayList<Question>> {

@Override

//params: question count, difficulty level, category

protected ArrayList<Question> doInBackground(Integer... integers) {

**יצירת אובייקט של המחלקה שמורידה את השאלות וקבלת השאלות ממנו.**

**פרמטרים כמו קטגוריה וכו' מתקבלים בפרמטר integers.**

HttpQuestionFetcher questionFetcher = new HttpQuestionFetcher();

return questionFetcher.getQuestions(integers[QUESTIONS\_COUNT\_INDEX], integers[DIFFICULTY\_LEVEL\_INDEX], integers[CATEGORY\_INDEX]);

}

**פעולה שרצה לאחר קבלת השאלות מהשרת**

@Override

protected void onPostExecute(ArrayList<Question> questions) {

**בדיקה אם התקבלו שאלות. אם לא, הצגת הודעת שגיאה וחזרה למסך הראשי.**

if (questions == null || questions.size() == 0) {

Toast.makeText(getBaseContext(), "Couldn't fetch questions!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

} else {

**אם התקבלו שאלות בהצלחה, שומרים אותן בViewModel.**

gameVM.setQuestions(questions);

**שליחת המשחק שנוצר לפיירסטור. פעולה זו גם מציגה את GameIdFragment עד שיתחבר המשתמש.**

sendGameToFirestore();

}

}

{

קבלת הGame מהשרת, במצב התחברות למשחק:

private class JoinGameAsync extends AsyncTask<Integer, Integer, Void>{

@Override

//params: id

**הפעולה מקבלת כפרמטר את הID של המשחק.**

protected Void doInBackground(Integer... integers) {

**שמירת השחקן הנוכחי, כי הורדת המשחק מפיירסטור תמחק את המשתמש הנוכחי מGame, כי הוא עדיין לא נמצא בGame שבDB.**

Player myPlayer = gameVM.getMyPlayer(); //getting the game from firestore overrides player2 to null

**הפיכת הID לסטרינג**

int id = integers[0];

String strId = Integer.toString(id);

**הורדה של המשחק מפיירסטור**

firestore.collection(GAMES\_COLLECTION\_PATH).document(strId).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {

@Override

public void onSuccess(DocumentSnapshot documentSnapshot) {

//game loaded successfully

if(documentSnapshot.exists()){

**אם ההורדה הצליחה, שמירת המשחק בviewModel:**

Game game = documentSnapshot.toObject(Game.class);

gameVM.setGame(game);

gameVM.enableGameSyncWithFirestore(GameActivity.this);

gameVM.setMyPlayer(myPlayer);

hideLoadingFragment(getSupportFragmentManager(), loadingFragment);

**הצגת השאלה הראשונה.**

showCurrentQuestion();

}

else{

**אם המשחק לא קיים, הצגת הודעת שגיאה וחזרה לתפריט הראשי**

Toast.makeText(GameActivity.this, "Wrong game ID!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

}

}

**אם ההורדה נכשלה, הצגת הודעת שגיאה וחזרה לתפריט הראשי**

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(GameActivity.this, "Connection error!", Toast.LENGTH\_SHORT).show();

backToMainMenu();

}

});

return null;

}

{

בכל פעם שהמשתמש בוחר תשובה, מציגים אם היא נכונה למשך שניה,

ואחר כך מציגים את השאלה הבאה:

private void sendAnswer(int answerIndex, Button answerButton) {

**בדיקה אם התשובה שנענתה נכונה**

int currentQuestionIndex = gameVM.getMyPlayer().getCurrentQuestionIndex();

oolean isCorrect = gameVM.getQuestions().get(currentQuestionIndex).correctAnswer == answerIndex;

**הוספה לרשימת התשובות הנכונות/שגויות של השחקן את התשובה שהוא ענה.**

ArrayList<Boolean> isCorrectList = gameVM.getMyPlayer().getIsCorrectList();

isCorrectList.add(isCorrect);

gameVM.setMyIsCorrectList(isCorrectList);

if (isCorrect) {

**אם התשובה נכונה, מציגיםם אותה בירוק**

answerButton.setBackgroundColor(MyColor.CORRECT\_GREEN);

} else {

**אם לא, מציגים אותה באדום.**

answerButton.setBackgroundColor(MyColor.WRONG\_RED);

answerButtons[gameVM.getQuestions().get(currentQuestionIndex).correctAnswer].setBackgroundColor(MyColor.CORRECT\_GREEN);

}

**מציירים האם התשובה נכונה/שגויה על הקנבס בתחתית המסך.**

**(כרגע מציג את ההתקדמות של השחקן הנוכחי, בהמשך אשנה כך שיציג את ההתקדמות של היריב).**

drawIsCorrectOnProgressBar(isCorrect, currentQuestionIndex);

**הגדלת אינדקס השאלה הנוכחית ב1.**

gameVM.setMyCurrentQuestionIndex(currentQuestionIndex + 1);

**כיבוי של כפתורי התשובה בזמן הצגת התשובה הנכונה.**

for (Button answerBtn : answerButtons)

answerBtn.setEnabled(false);

recordImgBtn.setEnabled(false);

**לחכות שניה אחת לפני הצגת השאלה הבאה**

new Handler().postDelayed(new Runnable() {

@Override

public void run() {

**אם אין עוד שאלות (נגמר המשחק), מפעילים AsyncTask שיציג את מסך סיום המשחק**

if (currentQuestionIndex == gameVM.getQuestions().size() – 1) {

//game ended

EndGameAsync endGameAsync = new EndGameAsync();

endGameAsync.execute();

**אם יש עוד שאלות, מציגים את השאלה הבאה.**

} else {

showCurrentQuestion();

}

}

}, 1000);

}

### EndGameFragment

|  |  |
| --- | --- |
| **תכונה** | **הסבר** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## מחלקות עזר

### Game

### Question

### User

### Player

### HttpQuestionFetcher

### GameViewModel

### ScoreListAdapter

## קבצי תצורה

## בסיסי נתונים