



**שם בית הספר:** ישיבה תיכונית חספין

**שם פרויקט:**MapsBattle

**שם התלמיד:** אמיתי פלדהיים

**ת.ז התלמיד:** 330794041

**שם המנחה:** חגי סוויד

**שם החלופה:** תכנות טלפונים חכמים

**תאריך הגשה:** 13.6.2025

# **תוכן עניינים:**

תוכן

[**תוכן עניינים:** 2](#_Toc200559280)

[**מבוא** 3](#_Toc200559281)

[**הרקע לפרויקט** 3](#_Toc200559282)

[**תהליך המחקר** 4](#_Toc200559283)

[**אתגרים מרכזיים** 5](#_Toc200559284)

[**תיאור תחום הידע – פרק מילולי** 6](#_Toc200559285)

[**אובייקטים נחוצים:** 6](#_Toc200559286)

[**סוגי נתונים** 6](#_Toc200559287)

[**ייצוג מידע:** 6](#_Toc200559288)

[**תיאור פעולות על המידע:** 7](#_Toc200559289)

[**עץ תרשים זרימת מסכים** 8](#_Toc200559290)

[**ארכיטקטורת הפרויקט** 9](#_Toc200559291)

[**מימוש הפרויקט** 13](#_Toc200559292)

[**בסיס נתונים** 17](#_Toc200559293)

[**מדריך למשתמש:** 18](#_Toc200559294)

[**רפלקציה:** 19](#_Toc200559295)

[**ביביליוגרפיה** 19](#_Toc200559296)

[**נספחים** 20](#_Toc200559297)

[**קטעי הקוד** 20](#_Toc200559298)

[מחלקת Locations: 20](#_Toc200559299)

[מחלקת User\_pro 21](#_Toc200559300)

[מחלקת Game: 23](#_Toc200559301)

[activity\_login\_pro.xml 27](#_Toc200559302)

[LoginProActivity: 28](#_Toc200559303)

[activity\_preview 30](#_Toc200559304)

[PreviewActivity 31](#_Toc200559305)

[activity\_pro\_map.xml 32](#_Toc200559306)

[pro\_map 33](#_Toc200559307)

# **מבוא**

## **הרקע לפרויקט**

**שם הפרויקט:**

MapsBattle

**תיאור קצר של הפרויקט:**

הפרויקט מהווה אפליקציה שהיא משחק על ידיעת הארץ. באמצעות האפליקציה אתה יכול לראות עד כמה אתה בקיא בידיעת הארץ ובמיקומי הערים בעזרת זה שבכל פעם שאתה תמקם מיקום שיינתן לך אתה תוכל לראות כמה קרוב היית למקום האמיתי וכך להשתפר בידיעת הארץ.

**קהל היעד של הפרויקט:**

קהל היעד של האפליקציה הוא לכולם. כל מי שרוצה לראות כמה הוא בקיא בידיעת הארץ יכול להשתתף ולראות.

**הסיבות לבחירת הנושא:**

הסיבה שבגללה בחרתי לעשות את הפרויקט הזה היא שהפרויקט הוא על ידיעת הארץ שזה קודם כל דבר שאני מאוד אוהב ודבר שני אני חושב שזה דבר חשוב שאנשים ידעו איפה נמצאים הערים והיישובים במדינה שהם חיים בה.

## **תהליך המחקר**

**נקודות עיקריות במחקר לצורכי הפרויקט:**

במהלך העבודה על האפליקציה נדרשתי ללמוד תחומים רבים שלא הכרתי מראש. כדי להבין וליישם בצורה נכונה את הכלים והטכנולוגיות שבאפליקציה, נעזרתי באתרים ומקורות מידע שוני. אלו היו האתרים המרכזיים שהשתמשתי בהם בעבודה על הפרויקט:

MapBox – דרך האתר של MapBox למדתי איך שמים מפה באפליקציה ולמדתי איך משתמשים במפה בצורה הכי טובה.

StackOverflow – פורום של מפתחים מכל העולם שבו מצאתי תשובות לשאלות רבות שעלו לי במהלך העבודה, ובעיקר לשגיאות ובעיות שהיו לי. חיפשתי שם פתרונות קיימים של מפתחים שנתקלו באותם בעיות כמו שלי ולמדתי איך לגשת לקוד בצורה נכונה.

YouTube – צפיתי במדריכים שעזרו לי להבין איך לבנות מסכים, ליצור תרשים UML, לצרף מפה של MapBox לאפליקציה. הסרטונים עזרו לי ליישם תהליכים בקוד.

ChatGPT – עזר לי ללמוד דברים חדשים ששימשו אותי בעיצוב המסכים, הסבר של קודים, תיקון טעויות ונתינת קורדינאטות שאני אוכל להכניס לאפליקציה.

Classroom – נעזרתי בClassroom בחומרים שהמורה העלה לנו על כל מיני נושאים ךדוגמא, עיצוב מסכים, שימוש בFirebase ולבצע פעולות כמו התחברות, הרשמה והצגת רשימות.

קיימת אפליקציה דומה לאפליקציה אותה בניתי:

**מלך הארץ –** אפליקציה דומה שנותנת לך יישובים ואתה צריך למקם אותם על המפה ואתה מקבל נקודות לפי מרחק, בנוסף ניתן לשחק לפי אזורים(אזור הגולן, נגב וכו'..). את הרעיון לאפליקציה וחלק נרחב מהעיצוב לקחתי כהשראה מאפליקציה זאת.

סקירת המצב הקיים בשוק:

כמו שאמרתי, האפליקציה שלי היא לא ייחודית כי יש את אפליקצית מלך הארץ שהאפליקציה שלי דומה לאפליקציה הזאת. הייחודיות שלי באפליקציה היא שבזמן המשחק יש אצלי ברקע מוזיקה ובאפליקציה מלך הארץ אין מוזיקה, חוץ מזה האפליקציות דומות.

## **אתגרים מרכזיים**

הפרויקט ופיתוחו גרם לי להתמודד עם כמה אתגרים עיקריים שביניהם: עבודה עם Firebase ועבודה עם MapBox.

1. **עבודה עם Firebase:**

לפני תחילת הפרויקט, לא היה לי הרבה ניסיון בעבודה עם מסדי נתונים, ובפרט לא עם Firebase. נדרשתי להבין וללמוד כיצד שומרים מידע אונליין בענן, ואיך מקשרים בין ממשק המשתמש לבין הנתונים האלו – כל זה באמצעות קוד.

במהלך העבודה למדתי לבצע מספר פעולות עיקריות באמצעות Firebase:

יצירת משתמשים חדשים, ושמירת נתוני המשתמש במסד תחת העננים locations או users. התחברות משתמש קיים, באמצעות Sign in בעזרת אימייל וסיסמא. שליפת נתונים כדי להציג רשימה של המשתמשים והניקוד שלהם. עדכון נתונים כמו לעדכן את הניקוד אם הניקוד החדש יותר גבוה.

על מנת לבצע את כל הפעולות האלו, ולוודא שהקוד מתקשר באופן חלק עם מסד הנתונים, היה עליי ללמוד ולחקור על Firebase, להבין כיצד עובד מסד הנתונים ואילו פעולות ניתן לבצע בעזרתו.

1. **עבודה עם MapBox**

כשהתחלתי את הפרויקט לא ידעתי איך לצרף מפה ואיך להשתמש בה

במהלך העבודה למדתי לבצע מספר פעולות עיקריות באמצעות MapBox:

למדתי קודם כל איך לצרף מפה לאפליקציה כדי שכל המסך יהיה המפה. למדתי איך להוציא את הקורדינאטות שנמצאות במרכז המסך כדי שאני אוכל לבדוק את המרחק ביניהם לנקודה אחרת.

# **תיאור תחום הידע – פרק מילולי**

**אובייקטים נחוצים, ייצוגם ופעולות על המידע שלהם**

## **אובייקטים נחוצים:**

1. **User(שחקן במשחק):**

אובייקט שמייצג את השחקן שמשחק במשחק

* Email – כתובת דוא"ל של המשתמש. טיפוס: String.
* Nickname–שם של המשתמש שייגזר מהמייל(מה שכתוב לפני ה-@).טיפוס: String
* Score – הניקוד שהשחקן משיג במהלך המשחק. טיפוס: int.
* documentId – הID של המשתמש בFirebase. טיפוס: DocumentReference.

1. **Location(מיקומי הערים)**

* Name – שם העיר כפי שמופיע בFirebase. טיפוס: String.
* Longitude – קו האורך של הערים כפי שמופיע בFirebase. טיפוס: double
* Latitude – קו הרוחב של הערים כפי שמופיע בFirebase. טיפוס: double
* locations – מערך של כל הערים שנמשך מהFirebase. טיפוס: Location[]

1. **Game(מחלקה של המשחק)**

* User – שחקן שמשחק במשחק. טיפוס: User

## **סוגי נתונים**

* int – מייצג מספרים(ניקוד)
* String – מייצג טקסט(מייל, כינוי, שם של העיר)
* DocumentReference – מייצג ID של המשתמש בFirebase
* Double – מייצג מספריים עשרוניים(קווי אורך ורוחב של הערים)
* User – מייצג אובייקט של שחקן

## **ייצוג מידע:**

* **משתמש שחקן(User)**

אובייקט עם כתובת מייל, כינוי, ניקוד וID שמתקבל אוטומטית מהFirebase.

מאפשר לשמור ולנהל את פרטי המשתמש המשחק בצורה נוחה ונגישה.

* **משתנה עיר(Location)**

אובייקט שמוגדר בFirebase עם שם של העיר, קו הרוחב של העיר וקו האורך של העיר ומערך של כל הערים כפי שמופיעים בFirebase. מאפשר לנהל את ולהשתמש בערים בצורה קלה ולנהל אותם.

## **תיאור פעולות על המידע:**

1. **הוספה:**

הוספת שחקן חדש, הזנת הפרטים שתוארו למעלה, יצירת מזהים ייחודים ויצירת אובייקט חדש ושמירה שלו תחת הענף המתאים בFirebase.

1. **עדכון:**

עדכון של הניקוד לשחקן, עדכון של שחקן לתוך הרשימה של השחקנים והניקוד שלהם.

1. **קריאה:**

שליפת נתוני המשתמש מהFirebase על מנת לעדכן אותם ולנהל אותם.

# **עץ תרשים זרימת מסכים**

LoginProActivity

PreciewActivity

pro\_map

# **ארכיטקטורת הפרויקט**

שם המסך: LoginProActivity

מסך כניסה של האפליקציה, מאפשר כניסה והרשמה לאפליקציה, המשתמש בוחר עם להירשם או להיכנס במידה ונכנס בעבר, כאשר המשתמש נכנס במרווח זמן קצר יחסית אז הוא לא צריך לעשות שוב כניסה אלא האפליקציה ישר עוברת למסך הבא.



Button

כפתור הרשמה וכניסה, משתנה בהתאם לבחירה בSpinner

EditText

קלט הסיסמה

EditText

קלט האימייל

Spinner

מאפשר בחירה בין כניסה להרשמה

TextView

שם המשחק

שם המסך: PreviewActivity

מסך שמאפשר לראות את רשימת השחקנים והניקוד שלהם, משם מתחילים את המשחק.



Button

כפתור תחילת המשחק

RecycleView

רשימה של המשתמשים והניקוד שלהם(עוד לא מומש)

שם המסך: pro\_map

מסך המשחק

TextView

מראה את העיר שצריך למקם



Button

כפתור אישור של המיקום

ImageView

סמן מיקום

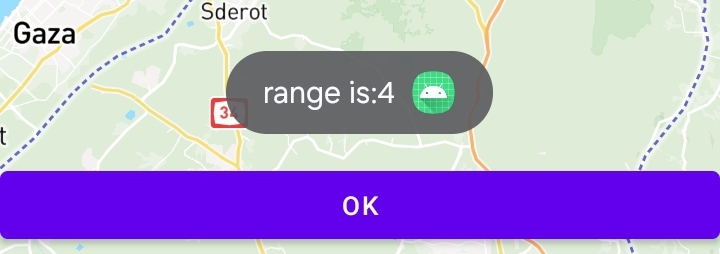
MapView

מפת המשחק

TextView

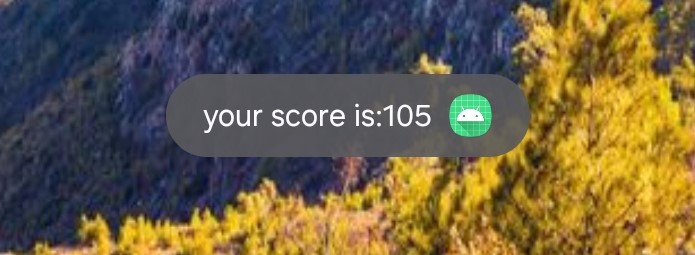
מראה את הניקוד

המשך מסך:



Toast

מראה למשתמש את המרחק מהעיר



Toast

מראה למשתמש את הניקוד שלו לאחר שנגמר המשחק

# **מימוש הפרויקט**

שם המחלקה: Game

מטרת המחלקה: מחלקה שבתוכה נמצאות הפעולות של המשחק

תכונות ופעולות המחלקה:

|  |  |
| --- | --- |
| תכונה | הסבר |
| ArrayList<Locations>locations | מערך מקומות מטיפוס Locations |
| Locations myLocation | משתנה מטיפוס Locations |
| int score | משתנה מטיפוס int |
| double distance | משתנה מטיפוס double |
| User\_pro myUser | משתנה מטיפוס User\_pro |
| פעולה | הסבר |
| FillingTheLocationsArray() | פעולה שממלאת את מערך הערים מהFirebase ומפעילה את פעולת randomcity |
| randomCity() | פעולה שמקבלת עיר רנדומלית ומעדכנת את הTextview בהתאם |
| calculateDistance(Point from, Point to) | פעולה שמחשבת את המרחק בין 2 נקודות |
| showCordinatesAndDistance(Point point) | פעולה שמציגה את המרחק מהעיר בקילומטרים |
| getCords() | פעולה שמקבלת את מיקום המשתמש |
| getRandomIndex(ArrayList<Locations>locations) | פעולה שמקבלת עיר רנדומלית ממערך הערים |
| calculateScore() | פעולה שמחשבת את הניקוד של השחקן בהתאם למרחק שלו |

שם המחלקה: User\_pro

מטרת המחלקה: מחלקה שבתוכה מוגדר המשתמש והפעולות שקשורות לו

תכונות ופעולות המחלקה:

|  |  |
| --- | --- |
| תכונה | הסבר |
| String UID | משתנה מטיפוס String |
| int score | משתנה מטיפוס int |
| DocumentReference documentId | משתנה מטיפוס DocumentReference |
| פעולה | הסבר |
| checkIfUserExist() | פעולה שבודקת אם המשתמש קיים בFirebase או לא, אם הוא לא קיים היא מוסיפה אותו ואם קיים היא מעדכנת את הdocument שלו |

שם המחלקה: Locations

מטרת המחלקה: מחלקה שבתוכה מוגדר הLocations והפעולות שקשורות לו

תכונות ופעולות המחלקה:

|  |  |
| --- | --- |
| תכונה | הסבר |
| String name | משתנה מטיפוס String |
| double latitiude | משתנה מטיפוס double |
| double longitude | משתנה מטיפוס double |
| פעולה | הסבר |
|  |  |

שם המחלקה: LoginProActivity

מטרת המחלקה: מסך הכניסה של האפליקציה, באמצעותו נרשמים\נכנסים למשחק.

תכונות המחלקה:

|  |  |  |
| --- | --- | --- |
| שם התכונה | טיפוס | הסבר |
| tvNameOfGame | TextView | מציג את שם המשחק |
| spLogin | Spinner | מאפשר בחירה בין הרשמה לכניסה, הכפתור משתנה בהתאם לבחירה |
| etEmail | EditText | שדה לכתיבת האימייל |
| etPassword | EditText | שדה לכתיבת סיסמא |
| btSubmit | Button | כפתור כניסה או הרשמה, משתנה בהתאם לבחירה בSpinner |

פעולות המחלקה:

|  |  |
| --- | --- |
| פעולה | הסבר |
| onCreate | קישור למסך וכניסה אליו, מקשר לFirebase ומתחיל את המשחק |
| startGame | פעולה שמתחילה את המשחק |
| onClick(btSubmit) | אם ההרשמה עובדת שולח Toast ומתחיל את המשחק ואם לא שולח Toast שההרשמה נכשלה |
| ItemSelected(spLogin) | משנה את הטקסט שעל הכפתור בהתאם לבחירה בספינר |

שם המחלקה: PreviewActivity

מטרת המחלקה: מסך שלפני המשחק, ניתן לראות בו רשימה של המששתפים והניקוד שלהם(עוד לא בוצע)

תכונות המחלקה:

|  |  |  |
| --- | --- | --- |
| שם התכונה | טיפוס | הסבר |
| spUsers | RecycleView | מראה את הרשימה של המשתמשים והניקוד שלהם(עוד לא בוצע) |
| btJoinGame | Button | כפתור שבלחיצה עליו אתה עובר למסך של המשחק |

פעולות המחלקה:

|  |  |
| --- | --- |
| פעולה | הסבר |
| onCreate | קישור למסך וכניסה אליו |
| checkMap | מתחיל את המשחק |
| onClick(btJoinGame) | מפעיל את פעולת checkMap |

שם המחלקה: pro\_map

מטרת המחלקה: המסך שבו המשחק קורה

תכונות המחלקה:

|  |  |  |
| --- | --- | --- |
| שם התכונה | טיפוס | הסבר |
| mapView | MapView | המפה שנמצאת על המסך |
| tvNameOfCity | TextView | מראה את שם העיר שהמשתמש צריך למקם על המפה |
| tvCityScore | TextView | מראה את הניקוד של השחקן |
| ivredMarker | ImageView | תמונה של סמן שנמצאת במרכז המסך |
| btmyButton | Button | כפתור שהשחקן לוחץ עליו שהוא חושב שמיקם במקום הנכון |

פעולות המחלקה:

|  |  |
| --- | --- |
| הסבר | שם הפעולה |
| מפעיל את פעולת checkIfUserExist ואת פעולת FillingTheLocationsArray | onCreate |
| הOnClick של הכפתור עושה שאחרי שאני לוחץ על OK אני מקבל Toast של המרחק, מפעיל את הפעולה של הגרלת עיר, מעדכן את העיר בTextView ומחשב את הניקוד | btmyButton.setOnClickListener |

## **בסיס נתונים**

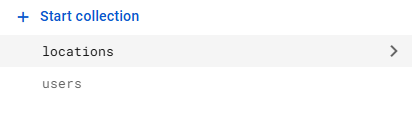
בפרויקט זה השתמשתי בFirebase, בסיס נתונים שיתופי על פי צרכי הפרויקט. בFirebase השתמשתי בשתי תכונות:

1. Authentication
2. Firestore Database

בAuthentication נשמרים האימייל והסיסמה של המשתמש באופן ייחודי(UID מונע כפילויות).

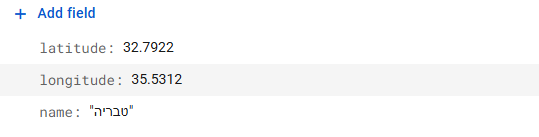
בFirestore Database נשמרים יותר פרטים.

* הcollections המתבקשים בפרויקט:



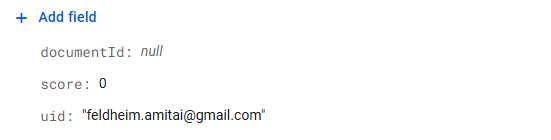
* **פירוט על כל collection:**

locations: רשימה שמכילה את כל הערים



* latitude – קו הרוחב של העיר
* longitude – קו האורך של העיר
* name – שם העיר

users: רשימה שמכילה את כל המשתמשים והפרטים שלהם



* documentId – מזהה ייחודי שנוצר אוטומטית לכל משתמש
* score – התוצאה של השחקן, מתעדכן בהתאם למשחק(עוד לא מומש)
* uid – האימייל של השחקן, גוזר את תחילת המילה כדי לצרף לרשימת תוצאות(עוד לא מומש)

# **מדריך למשתמש:**

מסך הכניסה, אם זו כניסה ראשונה שלך אז יש למלא את הפרטים(מייל וסיסמא) ולאחר מכן ללחוץ על הרשמה, אם זו לא הכניסה הראשונה וכבר נכנסת בעבר, יש ללחוץ על איפה שכתוב Item 1 ולבחור באופציה sign up, למלא את הפרטים(מייל וסיסמא) וללחוץ על כניסה.

לאחר מכן אתה תראה רשימה של משתמשים והניקוד שלהם(עוד לא מומש). מעל הרשימה יש כפתור START תלחץ עליו על מנת להתחיל לשחק.

עכשיו אתה מגיע למסך המשחק ובו תראה מפה, למעלה תראה את הכיתוב "הניקוד שלך הוא:" ואת הכיתוב "העיר היא:" ואז שם של עיר, עליך להזיז את המפה עד שהסמן יוצב על המיקום שאתה חושב ששם נמצאת העיר ולאחר שמיקמת עליך ללחוץ על הכפתור ok שלמטה, ברגע שתלחץ אתה תראה הודעה שתראה לך כמה היית רחוק מהמיקום בקילומטרים ובמקביל למעלה הניקוד שלך יתעדכן בהתאם למרחק שלך מהעיר(ככול שתמקם יותר קרוב הניקוד יהיה יותר גבוה). עכשיו אתה תקבל עיר חדשה ועליך להמשיך ככה במשך 3 סיבובים. לאחר הפעם השלישית אתה תחזור למסך הקודם ובמקביל תקבל הודעה למטה שתראה לך כמה ניקוד קיבלת במשחק.

עכשיו תוכל לראות את עצמך ברשימה ולראות כמה ניקוד עשית ביחס לשחקנים אחרים.

אם ברצונך לשחק שוב ולשפר את הניקוד תוכל ללחוץ על START ולשחק שוב.

בהצלחה!

# **רפלקציה:**

הסיבה שבגללה בחרתי לעשות את האפליקציה הזאת היא שהאפליקציה עוסקת בידיעת הארץ שזה קודם כל דבר שאני מאוד אוהב ודבר שני אני חושב שזה דבר חשוב שאנשים ידעו איפה נמצאים הערים והיישובים במדינה שהם חיים בה.

בהתחלה פחדתי כי אמרתי לעצמי איך אני אצליח לבנות אפליקציה? איך יהיה לי זמן לזה עם כל הלימודים? לא היה לי ניסיון עם דברים כאלה חוץ ממה שלמדנו בשיעורים שהיו יחסית בתחילת שנה, אז התחלתי לחזור על כל הנושאים שוב, איך לבנות מסכים ולעצב אותם, איך לעבור בין מסך למסך? איך לעשות Spinner?

חוץ מזה בשביל שהאפליקציה תעבוד כמו שאני רוצה הייתי צריך ללמוד דברים חדשים, גם דברים כללים כמו איך לעבוד עם Firebase, להאזין לאירועים במסד הנתונים ולנהל את זרימת המשתמש. בנוסף, הייתי צריך ללמוד דברים ספציפיים כמו איך אני בונה מפה ואיך אני עובד איתה. במהלך הדרך למדתי לחשוב ולתכנן ממשק משתמש ברור, חשבתי איך האפליקציה תראה הכי מזמינה שהמשתמשים ייהנו להשתמש בה ובעקבות זה עשיתי תמונה של נוף בתור רקע, והוספתי כל מיני דברים קטנים שיהפכו את החוויה של המשתמש ליותר טובה.

ככל שהתקדמתי בפרויקט, הבנתי שזה יותר מורכב וקשה ממה שחשבתי, נתקלתי בבעיות שונות שהייתי צריך לשבת עליהם 15-20 דקות כדי לחקור באינטרנט ולגלות מה הבעיה ואיך אני מתקן אותה.

במהלך העבודה למדתי ששאילת שאלות זה דבר חשוב ולא רק לנסות לפתור דברים בעצמי. הבנתי שאני צריך לפתור כל בעיה לאט לאט, קודם לזהות איפה הבעיה, לנסות להבין איך אני מתקן אותה בעזרת המורה\האינטרנט ואז לתקן אותה.

באגים שהייתי מתקן בעתיד: יש באג שאם אתה ממקם ומקבל לדוגמא 50 נקודות ואחרי זה אתה לוחץ עוד פעם בלי להזיז את המפה אז אתה מקבל אותו ניקוד ללא קשר למרחק.

משימות עתידיות: יצירת הRecycleView שיראה את המשתמשים והניקוד שלהם, עדכון הscore בFirebase בהתאם לניקוד שהוא מקבל במשחק, לעשות שכאשר השחקן עושה שיא נקודות זה יחליף את הניקוד בfirestore.

לסיכום, הפרויקט הזה היה תהליך מאתגר אבל גם משמעותי, זה היה חוויה של למידה והתמודדות עם קשיים. אני מאוד מרוצה ממה שהצלחתי לבנות, והרגשתי שצמחתי ברמה המקצועית וברמה האישית ובסוף בסוף למרות כל הקשיים זה היה חוויה שנהנתי ממנה.

# **ביביליוגרפיה**

* אתר Stackoverflow
* ChatGPT
* Youtube
* MapBox
* Classroom

# **נספחים**

## **קטעי הקוד**

### מחלקת Locations:

package com.example.mathprojectamitai2;  
  
public class Locations {  
private String name;  
private double latitiude;  
private double longitude;  
  
  
public Locations(String name, double latitiude, double longitude) {  
this.name = name;  
this.latitiude = latitiude;  
this.longitude = longitude;  
}  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public double getLatitiude() {  
 return latitiude;  
 }  
  
 public void setLatitiude(double latitiude) {  
 this.latitiude = latitiude;  
 }  
  
 public double getLongitude() {  
 return longitude;  
 }  
  
 public void setLongitude(double longitude) {  
 this.longitude = longitude;  
 }  
}

### מחלקת User\_pro

public class User\_pro {  
  
 private String UID;  
  
 private int score;  
 //private User\_pro myUser;  
  
 private DocumentReference documentId;  
  
 public User\_pro(String UID, int score, DocumentReference documentId) {  
 this.UID = UID;  
 this.score = score;  
 this.documentId = documentId;  
 }  
  
 public User\_pro() {  
 }  
  
  
  
 public User\_pro(String UID) {  
 this.UID = UID;  
 }  
  
 public String getUID() {  
 return UID;  
 }  
  
 public void setUID(String UID) {  
 this.UID = UID;  
 }  
  
 public int getScore() {  
 return score;  
 }  
  
 public void setScore(int score) {  
 this.score = score;  
 }  
  
 public DocumentReference getDocumentId() {  
 return documentId;  
 }  
  
 public void setDocumentId(DocumentReference documentId) {  
 this.documentId = documentId;  
 }  
  
 static User\_pro *user*;  
  
 public void checkIfUserExist() {  
  
 *user*=this;  
 FirebaseFirestore.*getInstance*().collection("users").whereEqualTo("uid", this.getUID()).get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {  
 @Override  
 public void onSuccess(QuerySnapshot queryDocumentSnapshots) {  
 if(queryDocumentSnapshots.isEmpty()) {  
 FirebaseFirestore.*getInstance*().collection("users").add(*user*).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {  
 @Override  
 public void onSuccess(DocumentReference documentReference) {  
 int n=0;  
 documentId = documentReference;  
 }  
 });  
  
 }else  
for(DocumentSnapshot documentSnapshot : queryDocumentSnapshots) {  
 if (documentSnapshot.exists()) {  
 documentId = documentSnapshot.getReference();  
 }  
 }  
  
 }  
 });  
 }

### מחלקת Game:

public class Game {  
 private String Uid;  
  
 MapView mapView;  
  
  
 private int previousIndex = -1;  
  
 Locations myLocation;  
  
 private double distance;  
  
 private Point cityPoint = Point.*fromLngLat*(35.2137, 31.7683);  
  
 private ArrayList<Locations> locations;  
  
  
  
 public Game(String uid, MapView mapView, int previousIndex, Locations myLocation, double distance, Point cityPoint, ArrayList<Locations> locations) {  
 Uid = uid;  
 this.mapView = mapView;  
 this.previousIndex = previousIndex;  
 this.myLocation = myLocation;  
 this.distance = distance;  
 this.cityPoint = cityPoint;  
 this.locations = locations;  
 }  
  
  
  
 public String getUid() {  
 return Uid;  
 }  
 public void setUid(String uid) {  
 Uid = uid;  
 }  
 public MapView getMapView() {  
 return mapView;  
 }  
 public void setMapView(MapView mapView) {  
 this.mapView = mapView;  
 }  
 public int getPreviousIndex() {  
 return previousIndex;  
 }  
 public void setPreviousIndex(int previousIndex) {  
 this.previousIndex = previousIndex;  
 }  
 public Locations getMyLocation() {  
 return myLocation;  
 }  
 public void setMyLocation(Locations myLocation) {  
 this.myLocation = myLocation;  
 }  
 public double getDistance() {  
 return distance;  
 }  
 public void setDistance(double distance) {  
 this.distance = distance;  
 }  
 public Point getCityPoint() {  
 return cityPoint;  
 }  
 public void setCityPoint(Point cityPoint) {  
 this.cityPoint = cityPoint;  
 }  
 public ArrayList<Locations> getLocations() {  
 return locations;  
 }  
 public void setLocations(ArrayList<Locations> locations) {  
 this.locations = locations;  
 }  
  
 */\*\*  
 \* את מערך המקומות ממלא   
 \*/* public void FillingTheLocationsArray() {  
 FirebaseFirestore.*getInstance*().collection("locations").get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {  
 @Override  
 public void onSuccess(QuerySnapshot queryDocumentSnapshots) {  
 for(DocumentSnapshot documentSnapshot : queryDocumentSnapshots){  
 if(documentSnapshot.exists()){  
 String name = documentSnapshot.getString("name");  
 double latitiude = documentSnapshot.getDouble("latitude");  
 double longitude = documentSnapshot.getDouble("longitude");  
  
 Locations location1 = new Locations(name, latitiude, longitude);  
 locations.add(location1);  
 }  
 }  
  
 randomCity();  
 }  
 });  
 }  
  
 */\*\*  
 \* עיר רנדומלית ומעדכן את הטקסט מקבל   
 \*/* public void randomCity(){  
 int n=getRndomIndex(locations, previousIndex);   
 previousIndex = n;   
 myLocation = locations.get(n);  
 Log.*d*("random",n+"");  
 cityPoint= Point.*fromLngLat*(locations.get(n).getLongitude(), locations.get(n).getLatitiude());  
  
  
 }  
 */\*\*  
 \*המרחק חישוב   
 \* @param from  
 \* @param to  
 \* @return  
 \*/* private double calculateDistance(Point from, Point to) {  
 return TurfMeasurement.*distance*(from, to);  
 }  
 */\*\*  
 \* המרחק מהעיר הצגת   
 \* @param point  
 \*/* private void showCoordinatesAndDistance(Point point) {  
 distance = calculateDistance(point, cityPoint); //חישוב מרחק והצבה בדיסטנס  
 String coordinates =  
 "\nDistance to city: " + String.*format*("%.2f", distance) + " km";  
 Log.*d*("tag", coordinates);  
  
 }  
  
 */\*\*  
 \* המשתמש קבלת מיקום   
 \* @return  
 \*/* public Point getCords(){  
 double latitude = mapView.getMapboxMap().getCameraState().getCenter().latitude();  
 double longitude = mapView.getMapboxMap().getCameraState().getCenter().longitude();  
 String message = "Latitude: " + latitude + ", Longitude: " + longitude;  
 Log.*d*("TAG", message);  
 Point point = Point.*fromLngLat*(longitude, latitude);  
 return point;  
 }  
  
 */\*\*  
 \* קבלת עיר רנדומלית ובודק שלא תוגרל אותה העיר פעמיים   
 \*  
 \* @param locations  
 \* @param previousIndex  
 \* @return  
 \*/* public int getRndomIndex(ArrayList<Locations> locations, int previousIndex){  
 Random random = new Random();  
 int newIndex;  
  
 if (locations.size() <= 1)  
 return 0;  
 do {  
 newIndex = random.nextInt(locations.size());  
 } while (newIndex == previousIndex);  
  
 return newIndex;  
 }  
  
 */\*\*  
 \* נקודות חישוב   
 \* @param distance  
 \* @param score  
 \* @return  
 \*/* public int calculateScore(double distance, int score){  
 int score1 = score;  
 int range = (int) Math.*round*(distance);  
 if(range>=0 && range<=5)  
 score1 = 50;  
 else if (range>5 && range<=10)  
 score1 = 45;  
 else if (range>10 && range<=15)  
 score1 = 40;  
 else if (range>15 && range<=20)  
 score1 = 35;  
 else if (range>20 && range<=25)  
 score1 = 30;  
 else if (range>25 && range<=30)  
 score1 = 25;  
 else if (range>30 && range<=35)  
 score1 = 20;  
 else if (range>35 && range<=40)  
 score1 = 15;  
 else if (range>40 && range<=45)  
 score1 = 10;  
 else if (range>45 && range<=50)  
 score1 = 5;  
 else  
 score1 = 0;  
  
 return score1;  
  
 }

### activity\_login\_pro.xml

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 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:orientation="vertical"  
 android:gravity="center"  
 tools:context=".LoginProActivity"  
 android:background="@drawable/appbackground"  
 >  
 <TextView  
 android:id="@+id/tvNameOfGame"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:text="MapsBattle"  
 android:textSize="50dp"  
 android:textAlignment="center"  
 android:textColor="#FF0707"  
 android:fontFamily="sans-serif-medium"  
 />  
 <Spinner  
 android:id="@+id/spLogin"  
 android:layout\_width="150dp"  
 android:layout\_height="25dp"  
 android:background="#673AB7"  
 android:textAlignment="center"  
 android:layout\_marginBottom="30dp"  
 android:layout\_marginTop="30dp"  
 />  
 <EditText  
 android:id="@+id/etEmail"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:hint="מייל"  
 android:gravity="center"  
 android:textSize="40dp"  
 android:textColorHint="@color/white"  
 />  
 <EditText  
 android:id="@+id/etPassowrd"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:hint="סיסמא"  
 android:gravity="center"  
 android:textSize="40dp"  
 android:textColorHint="@color/white"  
 />  
 <Button  
 android:id="@+id/btSubmit"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_marginTop="10dp"  
 android:text="הרשמה"  
 android:textSize="20dp"  
 android:backgroundTint="#673AB7"  
 />  
</LinearLayout>

### LoginProActivity:

public class LoginProActivity extends AppCompatActivity {  
 private TextView tvNameOfGame;  
 private EditText etEmail;  
 private EditText etPassowrd;  
 private Button btSubmit;  
 private Spinner spLogin;  
 private FirebaseAuth auth;  
 */\*\*  
 \* למסך כניסה   
 \* @param savedInstanceState  
 \*/* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_login\_pro*);  
 initview();  
 auth = FirebaseAuth.*getInstance*();  
 if(auth.getCurrentUser()!=null){  
 startGame();  
 }  
 }  
 */\*\*  
 \* התחלת המשחק   
 \*/* private void startGame(){  
 Intent inn = new Intent(this, PreviewActivity.class);  
 inn.putExtra("userName",auth.getCurrentUser().getEmail().toString());  
 startActivity(inn);  
 }  
 boolean isRegister = true;  
 public void initview(){  
 tvNameOfGame = findViewById(R.id.*tvNameOfGame*);  
 etEmail = findViewById(R.id.*etEmail*);  
 etPassowrd = findViewById(R.id.*etPassowrd*);  
 btSubmit = findViewById(R.id.*btSubmit*);  
 spLogin = findViewById(R.id.*spLogin*);  
 btSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 //אימות והרשמה  
 if(isRegister == true){  
 auth.createUserWithEmailAndPassword(etEmail.getText().toString(),etPassowrd.getText().toString()).addOnCompleteListener(LoginProActivity.this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if(task.isSuccessful()){  
 Toast.*makeText*(LoginProActivity.this, "Registion success.", Toast.*LENGTH\_SHORT*).show();  
 startGame();  
 }else{  
 Toast.*makeText*(LoginProActivity.this, "Registion failed.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
 if(isRegister == false){  
 auth.signInWithEmailAndPassword(etEmail.getText().toString(), etPassowrd.getText().toString()).addOnCompleteListener(LoginProActivity.this, new OnCompleteListener<AuthResult>() {  
 @Override  
 public void onComplete(@NonNull Task<AuthResult> task) {  
 if(task.isSuccessful()){  
 Toast.*makeText*(LoginProActivity.this, "Authentication success.", Toast.*LENGTH\_SHORT*).show();  
 startGame();  
 }else{  
 Toast.*makeText*(LoginProActivity.this, "Authentication failed.", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 }  
 }  
 });  
 spLogin.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int position, long l) {  
 String item = adapterView.getItemAtPosition(position).toString();  
 Toast.*makeText*(LoginProActivity.this, "sellected: " + item, Toast.*LENGTH\_LONG*).show();  
  
 if(item.equals("sign up")) {  
 isRegister=true;  
 btSubmit.setText("הרשמה");  
 }if (item.equals("Log in")) {  
 isRegister = false;  
 btSubmit.setText("כניסה");  
 }  
 }  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
 }  
 });  
 ArrayList<String> arrayList = new ArrayList<>();  
 arrayList.add("sign up");  
 arrayList.add("Log in");  
 ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.*simple\_spinner\_item*, arrayList);  
 adapter.setDropDownViewResource(android.R.layout.*select\_dialog\_singlechoice*);  
 spLogin.setAdapter(adapter);  
 }  
}

### activity\_preview

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
 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:orientation="vertical"  
 android:gravity="center"  
 tools:context=".PreviewActivity"  
 android:background="@drawable/appbackground"  
 >  
  
  
 <Button  
 android:id="@+id/btJoinGame"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="60dp"  
 android:text="START"  
  
 android:textSize="30dp"  
 />  
</LinearLayout>

### PreviewActivity

public class PreviewActivity extends AppCompatActivity {  
 private Button btJoinGame;  
 */\*\*  
 \* למסך כניסה   
 \* @param savedInstanceState  
 \*/* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_preview*);  
 initview();  
 }  
 */\*\*  
 \* התחלת המשחק   
 \*/* private void checkMap(){  
 Intent inn = new Intent(this, pro\_map.class);  
 startActivity(inn);  
 }  
 public void initview(){  
 btJoinGame = findViewById(R.id.*btJoinGame*);  
 btJoinGame.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 checkMap();  
 }  
 });  
 }

### activity\_pro\_map.xml

<?xml version="1.0" encoding="utf-8"?>  
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/main"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 tools:context=".pro\_map">  
  
 <com.mapbox.maps.MapView  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 xmlns:mapbox="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/mapView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 mapbox:mapbox\_cameraTargetLat="32.31"  
 mapbox:mapbox\_cameraTargetLng="35.06"  
 mapbox:mapbox\_cameraZoom="9.0"  
 />  
 <TextView  
 android:id="@+id/tvNameOfCity"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:layout\_marginTop="75dp"  
 android:text="העיר היא: "  
 android:textSize="30sp"  
 android:textAlignment="center"  
 android:background="@color/purple\_500"  
 />  
 <TextView  
 android:id="@+id/tvCityScore"  
 android:layout\_width="match\_parent"  
 android:layout\_height="50dp"  
 android:text="הניקוד שלך הוא: "  
 android:textSize="30sp"  
 android:textAlignment="center"  
 android:layout\_marginTop="130dp"  
 android:background="@color/purple\_500"  
 />  
 <ImageView  
 android:id="@+id/ivredMarker"  
 android:layout\_width="50dp"  
 android:layout\_height="50dp"  
 android:layout\_gravity="center"  
 android:src="@drawable/red\_marker"  
 />  
 <Button  
 android:id="@+id/btmyButton"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="OK"  
 android:layout\_gravity="bottom"  
 android:layout\_marginBottom="50dp"  
 />  
</FrameLayout>

### pro\_map

public class pro\_map extends AppCompatActivity {  
   
 MapView mapView;  
 private TextView tvNameOfCity;  
 private Button btmyButton;  
 Locations myLocation;  
 private int score;  
 private int counter;  
 private TextView tvCityScore;  
 private double distance;  
 private User\_pro user\_pro;  
 private Game game;  
 private int previousIndex = -1;  
 private ArrayList<Locations> locations = new ArrayList<>();  
 private Point cityPoint = Point.*fromLngLat*(35.2137, 31.7683);  
   
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_pro\_map*);  
 initview();  
 String email = FirebaseAuth.*getInstance*().getCurrentUser().getEmail();  
 user\_pro = new User\_pro();  
 user\_pro.checkIfUserExist();  
 game = new Game(user\_pro.getUID(), mapView, previousIndex, myLocation, distance, cityPoint, locations);  
  
   
 game.FillingTheLocationsArray();  
 mapView.setOnTouchListener(new View.OnTouchListener() {  
 @Override  
 public boolean onTouch(View view, MotionEvent motionEvent) {  
 if (motionEvent.getAction() == MotionEvent.*ACTION\_UP*) {  
 Point centerPoint = mapView.getMapboxMap().getCameraState().getCenter();  
 game.getCords();//קבלת מיקום המשתמש  
 //game.showCoordinatesAndDistance(centerPoint); }  
 return false;  
 }  
 });  
 if (mapView != null) {  
 mapView.getMapboxMap().loadStyleUri(Style.*MAPBOX\_STREETS*, new Style.OnStyleLoaded() {  
 @Override  
 public void onStyleLoaded(@NonNull Style style) {  
 }  
 });  
 }  
 }  
 public void initview() {  
 mapView = findViewById(R.id.*mapView*);  
 tvNameOfCity = findViewById(R.id.*tvNameOfCity*);  
 btmyButton = findViewById(R.id.*btmyButton*);  
 tvCityScore = findViewById(R.id.*tvCityScore*);  
 btmyButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 int range = (int) Math.*round*(distance);  
 Toast.*makeText*(pro\_map.this, "range is:" + range, Toast.*LENGTH\_SHORT*).show();  
 game.randomCity();  
 tvNameOfCity.setText("העיר היא: " + myLocation.getName());  
 int x = game.calculateScore(distance, score); score += x;  
 tvCityScore.setText("הניקוד שלך הוא: " + score);  
 counter++;  
 if (counter > 2) {  
 if (user\_pro.getDocumentId() != null) {  
 user\_pro.getDocumentId().update("score", score)  
 .addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void unused) {  
 Log.*d*("Firestore", "Score updated successfully");  
 Toast.*makeText*(pro\_map.this, "your score is:" + score, Toast.*LENGTH\_SHORT*).show();  
 }  
 })  
 .addOnFailureListener(new OnFailureListener() {  
 @Override  
 public void onFailure(@NonNull Exception e) {  
 Log.*e*("Firestore", "Failed to update score", e);  
 }  
 });  
 }  
 Intent inn = new Intent(pro\_map.this, PreviewActivity.class);  
 startActivity(inn);  
 }  
 }  
 });  
 }  
}