

תיכון גבעת שמואל

משחק זיכרון - סיימון

30/5/2020

מגיש : עמית אטליס

ת.ז. : 214391757

המנחה : אורי רוטנברג

חלופה : תכנות טלפונים ניידים – Android Studio

תאריך הגשה : 30/05/2020



תוכן העניינים

תוכן עניינים

תודות.....	3
מבוא	4
מדריך למתכנת.....	8
אלגוריתמים מרכזיים	8
אתגרים מרכזיים	9
תכן.....	10
עץ מודולים.....	10
Use Case תרשים	10
תרשים מחלקות.....	11
MODEL	11
Activities	13
Services	14
Threads.....	14
Content Providers.....	14
Broadcast Receivers.....	14
Resources	15
Drawables	16
קבצים.....	17
קבצי שמע	17
Database סכמת	18
פרוטוקול תקשורת.....	18
מדריך משתמש.....	19
מטרת המערכת	19
יכולות המערכת.....	19
תפעול המערכת	20
תרשים זרימה בין המסכים	20
מסך פתיחה.....	21
MainActivity(מסך הבית של האפליקציה)	22
EmailActivity(מסך ליצירת קשר עם מפתח האפליקציה)	23
LoginActivity(מסך התחברות עבור משתמש קיים)	24
RegisterActivitiy(מסך הרשמה עבור משתמש חדש)	25
UserActivity(מסך הבית לפני התחלת משחק (לאחר התחברות המשתמש))	26
SplashScreenGameActivity(מסך טעינה לפני התחלת משחק)	27
GameActivity(מסך המשחק)	28
הרשאות	29

דרישות מיוחדות ומגבלות	29
רפלקציה	30
ביבליוגרפיה	31
נספחים	32
תיעוד – קוד הפרויקט	32
מסמכים נוספים	45
כללי משחק (אופציונלי)	46

תודות

ברצוני להודות למורה שלי – אורי רוטנברג, שלמרות שהגיע אלינו רק בסוף, הוא עדיין הצליח ללמד אותי דברים חדשים שלא ידעתי לפני. הוא לימד אותי כיצד לעבוד בצורה נכונה ומסודרת יותר. לדוגמה, הוא לימד אותי כיצד לעבוד עם מחלקות מידע. בזכותו, הפרויקט כתוב בצורה מסודרת ונכונה יותר. כל זה לא היה קורה בלעדיו.

מבוא

רעיון מרכזי

אפליקציית משחק זיכרון - סיימון שמשפרת את הזיכרון.

משחק הסיימון הוא משחק זיכרון המוכר והפופולרי ביותר. צריך להפעיל המון זריזות, ראייה טובה שמיעה טובה וזיכרון מעולה. במשחק צריך לעקוב אחרי הצלילים שהסיימון מפיק. סיימון הוא משחק ממכר מאוד ומהנה מאוד. את המשחק הזה תשחקו רק בעזרת האצבעות והראש. האצבעות יהיה הכלי שיעביר את המסרים והראש יצטרך להיות תמיד דלוק ומוכן לפעולה.

סיימון הוא משחק זיכרון שבו נבחר באופן אקראי לחצן מסוים. עם בחירת הלחצן מושמע גם צליל ייחודי ללחצן. המשחק מתחיל כאשר נבחר באופן אקראי לחצן כלשהו. השחקן צריך ללחוץ על הלחצן שנבחר, כאשר בכל סיבוב נוסף מתווסף עוד צבע (עוד לחצן). זוהי צורה של למידה באמצעות חיקוי. המשחק מסתיים כאשר השחקן טועה ולוחץ על לחצן שלא הופיע ברצף, או כאשר הוא מצליח לחזור על הרצף הארוך ביותר שהוגדר למערכת.

מוטיבציה

האפליקציה נועדה לעזור לאותם אנשים הסובלים מבעיות זיכרון. האפליקציה עוזרת לשמירה על רצף, תפיסה חזותית וקשב וריכוז.

תיאור תוצאות מצופות

אני מקווה שהאפליקציה תעזור לשפר את הזיכרון לכל אדם הסובל מבעיות זיכרון. אני מצפה שהיא תעזור לאותם אנשים לטפל בבעיה שהיא מאוד נפוצה בימינו ואין הרבה אפשרויות מהנות לטפל בבעיה. אני שואף שכל אדם ייהנה בזמן השימוש באפליקציה ויוכל לשפר בדרך הטובה ביותר בבעיית הזיכרון שהוא סובל ממנה.

מצב קיים

כיום קיימים פתרונות רבים לאותם אנשים הסובלים מבעיות זיכרון. ישנם משחקי זיכרון שונים המסוגלים לעזור לאותם אנשים. כגון: חתחתול – משחק קלפים חברתי אשר דורש מספר משתתפים ובעל מספר חוקים. כרגע ללא גרסה וירטואלית. משחק זיכרון נוסף הוא

משחק הזיכרון – ישנם גרסאות רבות ומגוונות למשחק זה הן כמשחק קופסא והן גרסאות וירטואליות. ככל שישנם יותר קלפים כך עולה דרגת הקושי במשחק.

ישנם פתרונות נוספים לבעיה, ראה פירוט בהמשך.

בין כל הפתרונות המוצעים לבעיה יש גם את הפתרון שהוא המשחק סיימון. מחקרים רבים טוענים שהמשחק סיימון הוא בין הפתרונות היעילים ביותר לבעיית הזיכרון.

ניתוח צרכים

בעיית הזיכרון היא בין הבעיות השכיחות בעולם. ולכן, החברה צריכה פתרונות שונים בכדי לטפל בבעיה. אחד הפתרונות המוצעים הוא המשחק הזיכרון - סיימון. עקב זאת, החלטתי שהפרויקט שלי יהיה המשחק סיימון, אך משופר יותר, טוב יותר. במטרה שיעזור לכל אותם אנשים הסובלים מבעיות זיכרון. בכדי להשתמש באפליקציה המשתמש זקוק לטלפון חכם מסוג Android. מה גם על המשתמש להתחבר למערכת או ליצור משתמש חדש ולספק את הנתונים הבאים: שם מלא, אימייל, סיסמה ומספר טלפון.

סקר ספרות/סקר שוק

בכדי להכין את הפרויקט נדרש ממני להבין כיצד עובד המשחק - סיימון ומהו האלגוריתם שעומד מאחורי המשחק. בשביל זה נעזרתי באתר :

[https://he.wikipedia.org/wiki/%D7%A1%D7%99%D7%99%D7%9E%D7%95%D7%9F \(%D7%9E%D7%A9%D7%97%D7%A7\)](https://he.wikipedia.org/wiki/%D7%A1%D7%99%D7%99%D7%9E%D7%95%D7%9F (%D7%9E%D7%A9%D7%97%D7%A7))

בזכות האתר הצלחתי להבין את מהלך המשחק ואת הרעיון שעומד מאחורי המשחק.

מה גם, הורדתי מ Google Play את המשחק במטרה להבין בצורה טובה יותר איך עובד המשחק לאנדרואיד.

לאחר זמן מה אחרי ששיחקתי במשחק, הבנתי שאני צריך לכלול כמה דברים הכרחיים בפרויקט

הבנתי שבפרויקט צריך להיות שימוש במספר Activities שונים. ולכן, נעזרתי באתר .

<https://www.vogella.com/tutorials/AndroidIntent/article.html>

בזכות האתר הצלחתי לכלול באפליקציה מספר Activities שונים ולקשר ביניהם.

מה גם, ראיתי שצריך שימוש ב Dialogs באפליקציה. עקב זאת, נעזרתי באתר

<https://www.javatpoint.com/android-alert-dialog-example>

בזכות האתר הצלחתי לכלול באפליקציה Alert Dialogs.

נוסף לכך, שמתי לב שהאפליקציה חייבת לכלול שימוש באנימציות וסאונד. ולכן, נעזרתי בסרטונים הבאים:

<https://youtu.be/p9PHXRay3r4>

<https://youtu.be/9oj4f8721LM>

בזכות האתרים האלו הצלחתי להוסיף אנימציה וצלילי בעת לחיצת כפתור.

זאת ועוד, במטרה ליצור Firebase Database שישמור בתוכו את השיאים ואת המשתמשים ובמידת הצורך להראות נתונים אלו למשתמש, נעזרתי בסרטונים הבאים:

<https://youtu.be/tbh9YaWPKKs>

<https://youtu.be/TwHmrZxiPA8>

https://youtu.be/RiHGwJ_u27k

<https://youtu.be/pAhYEy6s9wQ>

<https://youtu.be/UMNeeMSUZlO>

סקירת שוק

כיום בשוק, ישנן כמה אפליקציות העובדות על אותו רעיון כמו האפליקציה שלי.

Memory Challenge (Simon) - קודם כל יש את המשחק

גרסה משופרת של המשחק הקלאסי "סיימון אומר", צריך ללחוץ על הכפתורים בסדר הנכון כל עוד אתה יכול. בודק את הזיכרון והרפלקסים שלך עם המשחק הפשוט הזה בשלושה מצבים שונים: * קלאסי: עקוב אחר הרצף ולחץ על הלחצנים בסדר הנכון.

הפוך: עליך לעקוב אחר הרצף בסדר הפוך! * מהירות: לחץ על הלחצנים בסדר הנכון במהירות האפשרית, ככל שתלחץ מהר יותר על הכפתור, כך תקבל יותר נקודות.

דירוג: 3.7 כוכבים מ-448 אנשים.

תמונה של המשחק:



משחק נוסף, הוא: משחק זיכרון - סיימון אמר

*משחק הקלסי סיימון אמר.

*משחק עם 3 רמות שנות: קל, בינוני וקשה.

*צלילים של פאנטום.

דירוג: 3.8 כוכבים מ-82 אנשים.

תמונה של המשחק:



מדריך למתכנת

אלגוריתמים מרכזיים

האלגוריתם המרכזי – נבחר צבע באופן אקראי. הצבע מוצמד ללחצן מסוים. עם בחירת הלחצן מושמע גם צליל ייחודי ללחצן, כמו כן, מופעלת גם אנימציה על הלחצן. השחקן צריך ללחוץ על הלחצן שנבחר, כאשר בכל סיבוב נוסף עוד צבע (עוד לחצן). המשחק מסתיים כאשר השחקן טועה ולוחץ על לחצן שלא הופיע ברצף.

לחיצת לחצן על ידי המערכת

ישנו מערך דינאמי - `ArrayList<COLOR> allColors` מסוג `COLOR`. `COLOR` הוא מסוג `enum` שמייצג את ארבעת הצבעים: `GREEN, RED, BLUE, YELLOW`. המערך שומר בתוכו את כל הצבעים שנלחצו על ידי המערכת עד כה וגם מוסיף כל פעם צבע רנדומלי לאחר שהמשתמש הצליח ללחוץ על כל הצבעים לפי הרצף הנכון. האלגוריתם ללחיצת לחצן על ידי המערכת עובד בכך שהוא עובר על המערך `allColors` באמצעות לולאת `for` ולפי הצבע מפעיל אנימציה שמדמה לחיצת כפתור וסאונד שמתאים לצבע, הכפתורים נלחצים בהפרש של שנייה אחד אחרי השני. במוד "classic" – הלחצנים נלחצים לפי הרצף שהופיע לפני וכל פעם מתווסף צבע נוסף לרצף. במוד "chaos" – הלחצנים נלחצים לפי רצף אקראי. (לא נשמר הרצף). ישנו חלק באלגוריתם שבמידה והשחקן משחק במוד "chaos", אז עוברים בלולאת `for` על המערך `allColors` ומשנים את הצבע שהיה במקום `i` במערך בצבע רנדומלי, וכך נוצר רצף חדש ואקראי.

בדיקה האם השחקן לחץ על הצבעים לפי הרצף הנכון

ישנו משתנה `int count` שהוא אחראי לשמירת מספר הלחצנים שנלחצו על ידי המשתמש והוא חלק מבדיקת הלחצן של המשתמש. כלומר, ברגע שהמשתמש לוחץ על לחצן בעל צבע מסוים, המשתנה `count` גדל באחד, כלומר גדל מספר הלחצנים שנלחצו על ידי המשתמש. לאחר מכן, יש אלגוריתם שבדוק האם הלחצן שנלחץ מתאים לצבע המסוים שברצף. אלגוריתם זה עובד בכך שהוא בודק האם הצבע שנמצא במקום `count` במערך `allColors` מתאים לצבע של הלחצן שהמשתמש לחץ.
`boolean gameOver = !allColors.get(count).equals(COLOR);`
 במידה והצבע של הלחצן שהמשתמש לחץ מתאים לצבע המסוים שברצף, `gameOver` יהיה שווה ל-`false`, והמשתמש יוכל להמשיך לשחק במשחק.

כאשר המשתמש הצליח לחזור על כל הרצף של הצבעים. כלומר, ברגע שמספר הלחצנים שנלחצו על ידי המשתמש (`count`) שווה לגודל המערך `allColors` (`allColors.size()`), כלומר שווה למספר הצבעים שנמצאים כרגע ברצף. התווסף צבע אקראי למערך `allColors` ויופעל האלגוריתם של לחיצת לחצן על ידי המערכת.

מסדי נתונים

הפרויקט יכלול בתוכו Firebase Database -Firestore Database ו Authentication Database. ה Database Authentication יהיה אחראי על שמירת המשתמשים וכל מה שקשור למשתמשים (איפוס סיסמה, התחברות למשתמש, יצירת משתמש ועוד...). ה Firestore Database יהיה אחראי על שמירת הפרטים של המשתמשים. כגון, שם מלא, אימייל ושיא אישי. במידת הצורך, יוכל המשתמש לראות מה Firestore את השיא של עצמו ושל 10 השחקנים הטובים ביותר. (Score Table)

אלגוריתם ליצירת טבלת שיאים

ישנה מחלקת מידע UserScore ששומרת בתכונותיה את השם ואת השיאים בשני המודים ("classic", "chaos") של כל משתמש.

בנוסף לכך, ישנה מחלקת מידע רבים UserScores שיורשת מ ArrayList ושומרת בתכונותיה שני ArrayList לכל מוד של משחק ("classic", "chaos") מסוג מחלקת המידע UserScore. בשני המערכים שמורים כל שמות השחקנים והשיאים שלהם. כדי ליצור את טבלת השיאים שבה מופיעים 10 השחקנים הטובים ביותר, השתמשתי באלגוריתם בתוך מחלקת המידע UserScores שממין את המערכים מהשיא הגבוה ביותר לשיא הנמוך ביותר. לאחר מכן, מה שנוותר הוא רק לרוץ בלולאת for על המערכים ולהציג את שמות השחקנים ואת השיאים שלהם. בכדי להציג את טבלת השיאים השתמשתי ב StringBuilder ששומר בתוכו מחרוזת של כל שמות השחקנים ואת השיאים שלהם ומציג אותם ב Alert Dialog.

אתגרים מרכזיים

האתגר המרכזי בפרויקט היה ליצור אלגוריתם שיממש את המשחק. האתגר הטכנולוגי בפרויקט הוא ליצור Firebase שישמור בתוכו את השיאים והפרטים של כל השחקנים ובמידת הצורך יראה נתונים אלו. קושי נוסף הוא ליצור דילי בין הלחיצות האקראיות של המערכת על הלחצנים.

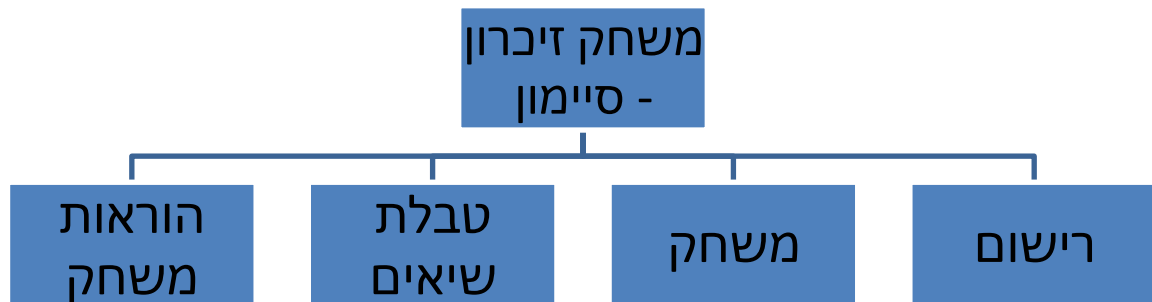
מה גם, אני צריך להשקיע מחשבה רבה במטרה לפתור תקלות נוספות בקוד. ולהשקיע זמן רב על העיצוב ועל הנגישות למשתמש.

כדי להצליח לבנות אב – טיפוס, עלי לעבוד שלב אחר שלב ובמקרה של אי הצלחה בתחום מסוים, עלי להשאיר את התקלה לסוף ולטפל בה אחרי שסיימתי את שאר הדברים. רק כך אצליח לסיים לבנות את האב – טיפוס. כלומר, אם אני אתקע על כל תקלה קטנה, אני בחיים לא אצליח להתקדם ולסיים את האב – טיפוס.

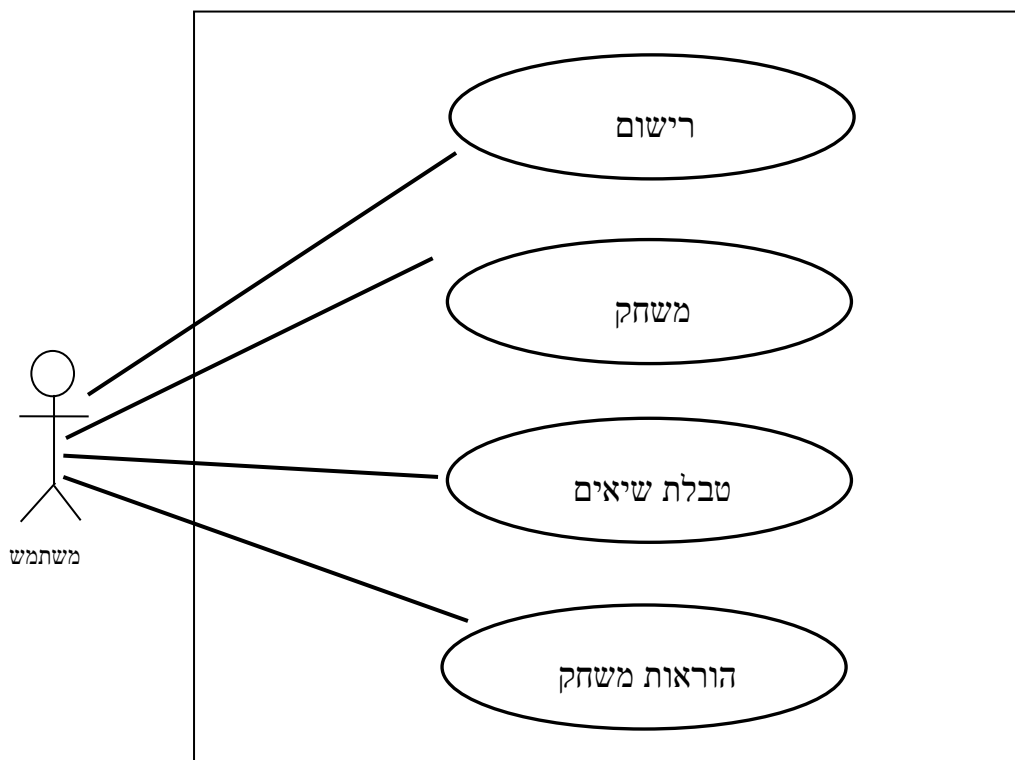
בכל תקלה שאני נתקע בזמן בניית האפליקציה, דבר ראשון אני פונה אל האינטרנט. אני מחפש פתרון לבעיה וברוב המקרים אני גם מוצא את הפתרון. לרוב, או שאני מוצא את הפתרונות ביוטיוב או באתרים שונים. כגון, Android Developers, StackOverflow ועוד.. אך, לא תמיד יש את כל התשובות באינטרנט. במקרה זה, אני פונה אל המורה שיעזור לי ויכוון אותי אל הפתרון. בדרך זו, הצלחתי להתגבר על כל הבעיות שנתקלתי בהן עד כה.

תכנ

עץ מודולים



Use Case תרשים



תרשים מחלקות**MODEL**

User
<pre>private final String name; private final String email; private int bestScoreClassic; private int bestScoreChaos;</pre>
<pre>User(String name, String email, int bestScoreClassic, int bestScoreChaos) setBestScoreClassic(int bestScoreClassic): void setBestScoreChaos(int bestScoreChaos): void getName(): String getEmail(): String getBestScoreClassic(): int getBestScoreChaos(): int</pre>

UserScores extends ArrayList
<pre>private ArrayList<UserScore> bestScoreClassicArray = new ArrayList<>(); private ArrayList<UserScore> bestScoreChaosArray = new ArrayList<>();</pre>
<pre>UserScores() getNameByIndexClassic(int index): String getNameByIndexChaos(int index): String getScoreByIndexClassic(int index): int getScoreByIndexChaos(int index): int getTopNameClassic(): String getTopNameChaos(): String getTopScoreClassic(): String getTopScoreChaos(): int getSizeClassicArray(): int getSizeChaosArray(): int add(UserScore userScore): void clear(): void sort(): void</pre>

UserScore
<pre>private String name; private int scoreClassic; private int scoreChaos;</pre>
<pre>UserScore(String name, int scoreClassic, int scoreChaos) getName(): String getScoreClassic(): int getScoreChaos(): int</pre>

NotificationReceiver extends BroadcastReceiver
none
onReceive(Context context, Intent intent): void

InternetConnection
private final Context context;
<pre>InternetConnection(Context context) isConnected(): boolean</pre>

SharedPreferencesRememberMe
<pre>- private final SharedPreferences sharedPref; - private final String key = "rememberMe";</pre>
<pre>- public SharedPreferencesRememberMe(Context context) - public edit(boolean value); void - public contains(); boolean - public getBoolean(); boolean</pre>

Activities

SplashScreenStartUpActivity - משמשת כמסך הפתיחה לאפליקציה. זהו המסך הראשון שהמשתמש רואה בעת כניסה לאפליקציה.

MainActivity - מממשת את התפריט הראשי של האפליקציה. MainActivity משמשת כמסך הבית של האפליקציה. בתוך מסך זה ניתן לעשות כמה פעולות:

- להתחבר למשתמש קיים.
- ליצור משתמש חדש.
- לעיין בהוראות המשחק.
- לעיין בטבלת השיאים.
- ליצור קשר עם המפתח של האפליקציה באמצעות אימייל.

מה גם, ניתן לראות באמצע המסך את השחקן עם הניקוד הגבוה ביותר בכל מוד (Winners Of The Day).

EmailActivity - משמשת כמסך ליצירת קשר עם מפתח האפליקציה באמצעות שליחת מייל.

LoginActivity - משמשת כמסך התחברות עבור משתמש קיים. רק לאחר התחברות, יוכל המשתמש להתחיל לשחק. כאשר המשתמש התחבר פעם אחת, הוא לא יצטרך להתחבר בשנית כאשר יכנס חזרה לאפליקציה.

RegisterActivity - משמשת כמסך עבור יצירת משתמש חדש. לאחר יצירת משתמש, המשתמש יועבר למסך UserActivity, ומשם יוכל להתחיל לשחק.

UserActivity - משמשת כמסך בית לאחר התחברות משתמש למערכת. לאחר שהמשתמש התחבר למשתמש קיים או יצר משתמש חדש, הוא יועבר למסך זה ומפה יוכל להתחיל לשחק. במסך זה ניתן לעשות כמה פעולות:

- לעיין בטבלת השיאים.
- לעיין בהוראות המשחק.
- להתנתק מהמשתמש ולחזור למסך הבית של האפליקציה – MainActivity.
- להתחיל לשחק במשחק או במוד "chaos" או במוד "classic" לפי בחירת השחקן.
- לבחור זמן ביום שבאותו זמן ישלח למשתמש התראה שתזכיר לו לשחק במשחק ולשפר את הניקוד והזיכרון שלו.

SplashScreenGameActivity - מסך זה משמש כמסך שמופיע לפני התחלת משחק. מסך זה נותן לשחקן מספר שניות להתכונן לפני תחילת המשחק.

GameActivity - משמשת כמסך המשחק. במסך זה ניתן לשחק במשחק עצמו - סיימון. לאחר שהמשתמש נפסל במשחק, הוא יכול לבחור בין כמה אופציות שונות:

- לחזור למסך הבית לפני התחלת משחק (לאחר התחברות משתמש למערכת) (UserActivity).
- לשחק שוב.

- לשתף את הניקוד שהשחקן הגיע אליו באמצעות פלטפורמות שונות. כגון: WhatsApp.
- מה גם, המשתמש יכול לראות את הניקוד שהגיע אליו ואת השיא הנוכחי שלו.

Services

באפליקציה ישנו Service שאחראי להפעלת קבצי שמע - mp3 באמצעות MediaPlayer. לדוגמה, במסך המשחק - GameActivity, ישנם 4 כפתורים שבעת לחיצה משמיעים סאונד. מה גם, כאשר השחקן נפסל במשחק, מושמע סאונד.

Threads

ממומש ב-Service.

FinalAnimationThread

מנהל את האנימציה במשחק. מופעל ע"י ה-GameActivity. מתקשר עם ה-GameActivity ע"י ה-Handler הממומש ע"י המחלקה AnimationHandler.

Content Providers

לא בשימוש.

Broadcast Receivers

באפליקציה ישנו שימוש ב- Broadcast Receivers במטרה לשלוח למשתמש התראה. מחלקת NotificationReceiver יורשת מ- BroadcastReceiver.

Resources

Layouts

activity_main.xml – מתאימה למחלקת MainActivity, ומציגה את מסך הבית של האפליקציה (לפני התחברות למשתמש קיים).

activity_email.xml – מתאימה למחלקת EmailActivity, ומציגה את מסך יצירת הקשר עם ה-developer של האפליקציה.

activity_login.xml – מתאימה למחלקת LoginActivity, ומציגה את מסך ההתחברות למשתמש קיים.

activity_register.xml – מתאימה למחלקת RegisterActivity, ומציגה את מסך יצירת משתמש חדש.

activity_user.xml – מתאימה למחלקת UserActivity, ומציגה את מסך הבית לפני התחלת המשחק (לאחר התחברות למשתמש קיים).

activity_game.xml – מתאימה למחלקת GameActivity, ומציגה את מסך המשחק.

activity_splash_screen_start_up.xml – מתאימה למחלקת SplashScreenStartUpActivity, ומציגה את מסך הפתיחה של האפליקציה. מסך זה הוא המסך הראשון שהמשתמש רואה בעת כניסה לאפליקציה.

activity_splash_screen_game.xml – מתאימה למחלקת SplashScreenGameActivity, ומציגה את המסך שמופיע לפני התחלת משחק. מסך זה מאפשר למשתמש להתכונן לפני התחלת המשחק.

Menus

באפליקציה ישנו תפריט - menu_user.xml.

מחלקת UserActivity משתמשת בתפריט זה.

Menu_user.xml

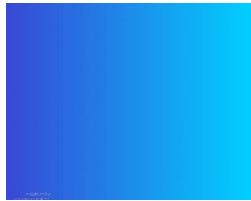
בתוך ה menu_user.xml ישנם 2 items : How To Play ו- Alarm.

How To Play – בעת לחיצה על item זה, נפתח למשתמש דיאלוג שמסביר על חוקי המשחק.

Alarm – בעת לחיצה על item זה, נפתח למשתמש דיאלוג שבו יוכל לבחור זמן ביום, שבזמן זה ישלח למשתמש התראה שתזכיר לו לשחק במשחק במטרה לשפר את הזיכרון ואת הניקוד שלו.

Drawables

רקעים למסכים באפליקציה



רקעים לכפתורים



icons לכפתורים



לוגו האפליקציה



תמונות נוספות הנמצאות בשימוש באפליקציה



קבצים

קבצי שמע

- Sound1.mp3
- Sound2.mp3
- Sound3.mp3
- Sound4.mp3
- Losegamesound.mp3

קבצי mp3, הנשמרים בתיקיית raw, ונעשה בהם שימוש על ידי מחלקת GameActivitiy.java :

GameActivity.java

- Sound1.mp3
- Sound2.mp3
- Sound3.mp3
- Sound4.mp3
- Losegamesound.mp3

הפעולות בהן נעשה שימוש בקובץ

קבצי השמע sound1,2,3,4.mp3 מופעלים כאשר המשתמש לוחץ על כפתורי המשחק.
קובץ השמע losegamesound.mp3 מופעל כאשר השחקן נפסל במשחק.

סכמת Database

<u>טבלת לקוחות</u>				
תיאור	מפתח זר	טיפוס הנתונים	שם השדה	
מזהה		integer	id	
השיא של המשתמש למוד "chaos"		integer	bestScoreChaos	
השיא של המשתמש למוד "classic"		integer	bestScoreClassic	
האימייל של המשתמש		String	email	
השם של המשתמש		String	name	

פרוטוקול תקשורת

לא בשימוש

מדריך משתמש

מטרת המערכת

אפליקציית משחק זיכרון - סיימון שמשפרת את הזיכרון.

משחק הסיימון הוא משחק זיכרון המוכר והפופולרי ביותר. צריך להפעיל המון זריזות, ראייה טובה שמיעה טובה וזיכרון מעולה. במשחק צריך לעקוב אחרי הצלילים שהסיימון מפיק. סיימון הוא משחק ממכר מאוד ומהנה מאוד. את המשחק הזה תשחקו רק בעזרת האצבעות והראש. האצבעות יהיה הכלי שיעביר את המסרים והראש יצטרך להיות תמיד דלוק ומוכן לפעולה.

סיימון הוא משחק זיכרון שבו נבחר באופן אקראי לחצן מסוים. עם בחירת הלחצן מושמע גם צליל ייחודי ללחצן. המשחק מתחיל כאשר נבחר באופן אקראי לחצן כלשהו. השחקן צריך ללחוץ על הלחצן שנבחר, כאשר בכל סיבוב נוסף מתווסף עוד צבע (עוד לחצן). זוהי צורה של למידה באמצעות חיקוי. המשחק מסתיים כאשר השחקן טועה ולוחץ על לחצן שלא הופיע ברצף, או כאשר הוא מצליח לחזור על הרצף הארוך ביותר שהוגדר למערכת.

יכולות המערכת

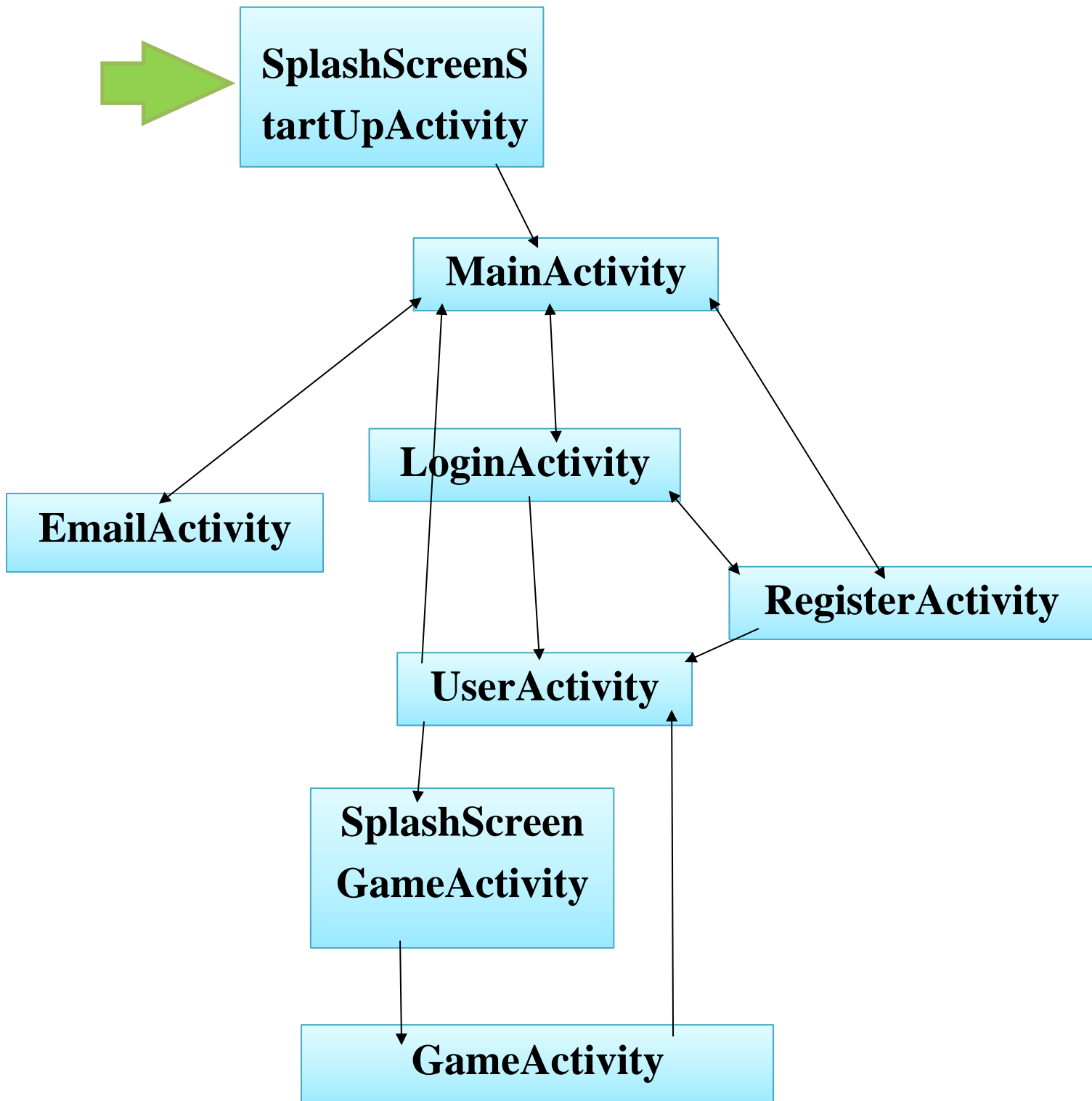
תיאור האופציות שנותנת המערכת למשתמש (Feature List).

- התחברות למשתמש קיים.
- יצירת משתמש חדש.
- איפוס סיסמה במידת הצורך.
- "Remember Me" - זכירת המשתמש על ידי המערכת, כך שלא יצטרך להתחבר בשנית.
- להציג ולהסתיר את הסיסמה על ידי לחיצת כפתור.
- יציאה ממשתמש מחובר.
- עיון בהוראות המשחק.
- עיון בטבלת השיאים של המשחק.
- עיון בשחקן עם הניקוד הגבוה ביותר בכל מוד של משחק.
- יצירת קשר עם המפתח של האפליקציה ע"י שליחת אימייל.
- קביעת זמן ביום שבו ישלח למשתמש התראה שתזכיר לו לחזור לשחק במשחק במטרה לשפר את הזיכרון ואת הניקוד שלו.
- אפשרות למשחק בשני מודים שונים: "classic" ו "chaos"
- עיון בניקוד הנוכחי של השחקן באמצע ובסיום המשחק.
- עיון בשיא של השחקן עצמו בכל מוד של משחק.
- שיתוף הניקוד של השחקן.

תפעול המערכת

תרשים זרימה בין המסכים

מתאר את אופן הניווט בין המסכים השונים



מסך פתיחה

מסך זה משמש כמסך הפתיחה לאפליקציה. זהו המסך הראשון שהמשתמש רואה בעת כניסה לאפליקציה. במסך זה מופעלת אנימציה גם לתמונה וגם לטקסט שמתחת לתמונה. באנימציה זו ניתן לראות את הלוגו של המשחק יורד מלמעלה למרכז המסך, והטקסט שמתחת ללוגו עולה מלמטה.

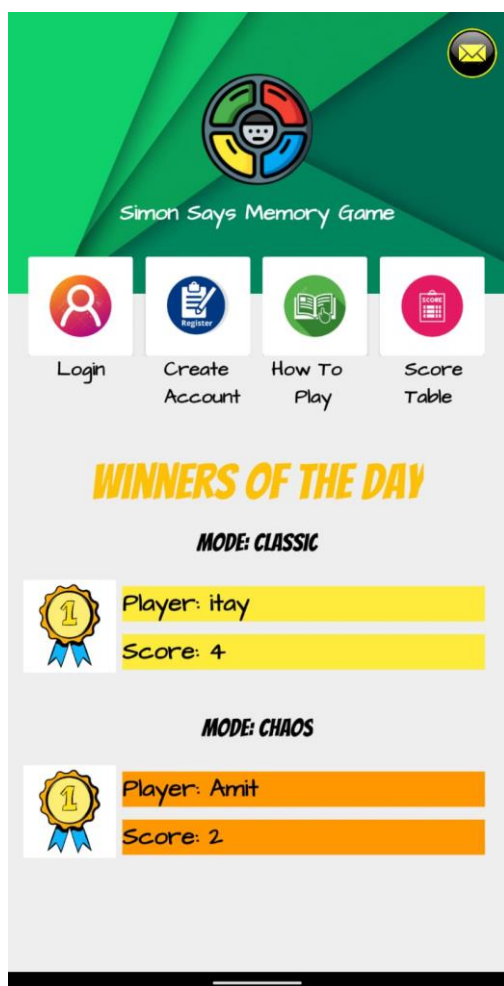


מסך הבית של האפליקציה (MainActivity)

מממשת את התפריט הראשי של האפליקציה. MainActivity משמשת כמסך הבית של האפליקציה. בתוך ה MainActivity ניתן לעשות כמה פעולות :

- להתחבר למשתמש קיים באמצעות לחיצה על כפתור "Login". בעת לחיצה על כפתור זה, יועבר המשתמש למסך - התחברות עבור משתמש קיים (LoginActivity).
- ליצור משתמש חדש באמצעות לחיצה על כפתור "Create Account". בעת לחיצה על כפתור זה, יועבר המשתמש למסך - הרשמה עבור משתמש חדש (RegisterActivity).
- לעיין בהוראות המשחק באמצעות לחיצה על כפתור "How To Play". בעת לחיצה על כפתור זה, יפתח למשתמש דיאלוג שבו יהיה כתוב את הוראות המשחק.
- לעיין בטבלת השיאים באמצעות לחיצה על כפתור "Score Table". בעת לחיצה על כפתור זה, יפתח למשתמש דיאלוג שבו יהיה את טבלת השיאים. בטבלה יהיו 10 השחקנים עם הניקוד הגבוה ביותר.
- ליצור קשר עם מפתח האפליקציה ע"י שליחת מייל, באמצעות לחיצה על הכפתור שנמצא בצד ימין למעלה עם icon של אימייל. בעת לחיצה על כפתור זה, יועבר המשתמש למסך – יצירת קשר עם מפתח האפליקציה (EmailActivity).

מה גם, ניתן לראות באמצע המסך את השחקן עם הניקוד הגבוה ביותר בכל מוד של משחק. (Winners Of The Day)



מסך ליצירת קשר עם מפתח האפליקציה (EmailActivity)

מסך זה משמש כמסך ליצירת קשר עם מפתח האפליקציה באמצעות שליחת מייל. במידה והמשתמש מעוניין ליצור קשר עם המפתח מכל סיבה כלשהי, מה שעליו לעשות הוא למלא את הנושא של הבקשה ואת התוכן שלה, ולסיום ללחוץ על הכפתור "Compose An Email". לאחר שלחץ על הכפתור, יפתח למשתמש דיאלוג שבו יבחר פלטפורמת Email שממנה ישלח המייל למפתח. מה גם, למשתמש יש את האפשרות לחזור למסך הבית (MainActivity) באמצעות לחיצה על הכפתור עם icon של חץ אחורה.

מסך התחברות עבור משתמש קיים (LoginActivity)

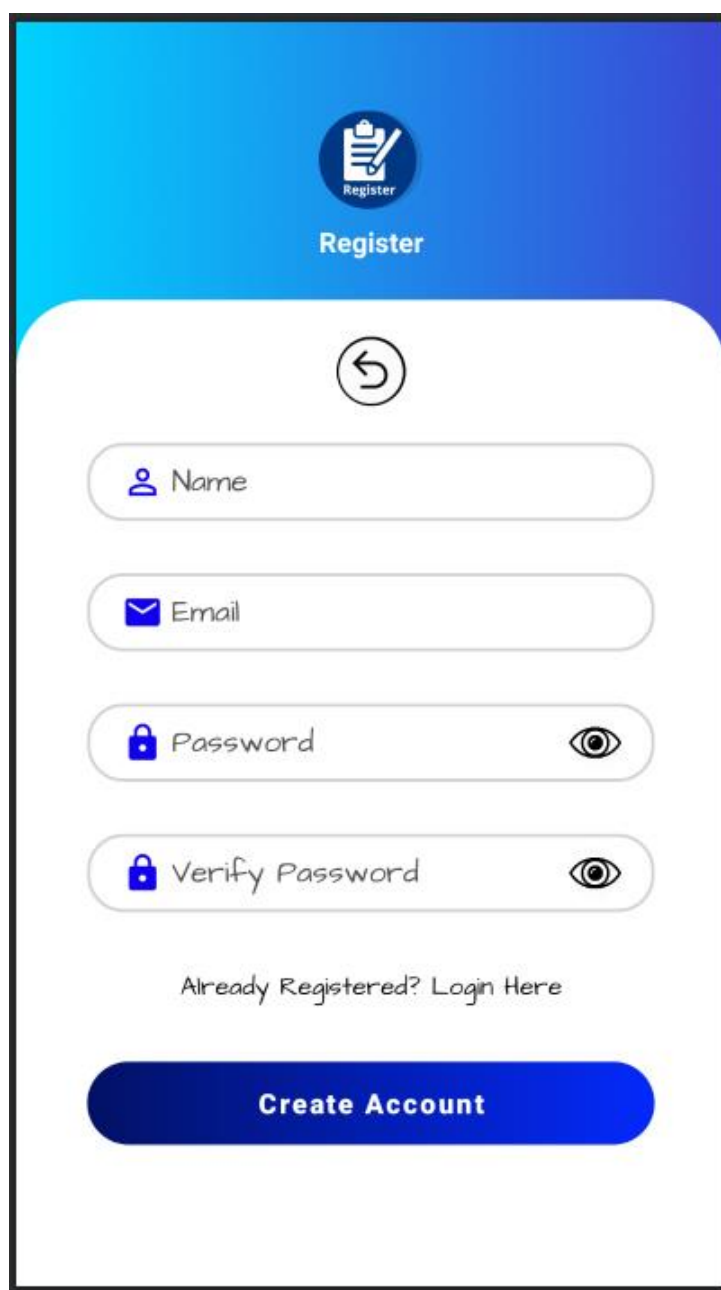
מסך זה משמש כמסך התחברות עבור משתמש קיים. במידה והמשתמש אינו זוכר את סיסמתו, יוכל ללחוץ על "Forgot Password? Reset Here" וישלח אימייל לכתובת המייל שלו לאיפוס סיסמה. מה גם, במידה והמשתמש אינו רשום, יוכל לעבור למסך RegisterActivity – יצירת משתמש חדש, באמצעות לחיצה על "New Here? Create Account". זאת ועוד, יוכל המשתמש להציג ולהסתיר את הסיסמה באמצעות לחיצה על הכפתור עם icon של עין.

רק לאחר התחברות, יוכל המשתמש להתחיל לשחק. כאשר המשתמש התחבר פעם אחת, הוא לא יצטרך להתחבר בשנית כאשר יכנס חזרה לאפליקציה. מה גם, למשתמש יש את האפשרות לחזור למסך הבית (MainActivity) באמצעות לחיצה על הכפתור עם icon של חץ אחורה.

The screenshot shows the LoginActivity interface. It features a red background with a white login form. At the top of the form is a back arrow icon. Below it are input fields for Email and Password. The Password field has an eye icon to toggle visibility. Below the password field is a 'Remember Me' checkbox. At the bottom of the form are two links: 'Forgot Password? Reset Here' and 'New Here? Create Account'. A large red 'Login' button is at the very bottom.

מסך הרשמה עבור משתמש חדש (RegisterActiviy)

משמשת כמסך עבור יצירת משתמש חדש. לאחר יצירת משתמש, המשתמש יועבר למסך UserActiviy, ומשם יוכל להתחיל לשחק. מה גם, במידה והמשתמש רשום, יוכל לעבור למסך LoginActivity – התחברות למשתמש קיים, באמצעות לחיצה על "Already Registered? Login Here". מה גם, למשתמש יש את האפשרות לחזור למסך הבית (MainActivity) באמצעות לחיצה על הכפתור עם icon של חץ אחורה. זאת ועוד, יוכל המשתמש להציג ולהסתיר את הסיסמה באמצעות לחיצה על הכפתור עם icon של עין.



The image shows a mobile app registration screen. At the top, there is a blue header with a circular icon containing a clipboard and the word "Register". Below the header is a white rounded rectangle containing the registration form. At the top of this white area is a back arrow icon. The form consists of four input fields: "Name" with a person icon, "Email" with an envelope icon, "Password" with a lock icon and an eye icon to toggle visibility, and "Verify Password" with a lock icon and an eye icon. Below these fields is the text "Already Registered? Login Here". At the bottom of the white area is a large blue button with the text "Create Account".

מסך הבית לפני התחלת משחק (לאחר התחברות המשתמש) (UserActivity)

מסך זה משמש כמסך הבית לאחר התחברות המשתמש למערכת. לאחר שהמשתמש התחבר למשתמש קיים או יצר משתמש חדש, הוא יועבר למסך זה ומפה יוכל להתחיל לשחק. במסך זה ניתן לעשות כמה פעולות:

- לעיין בטבלת השיאים באמצעות לחיצה על הכפתור שנמצא בצד ימין למטה עם icon של טבלת שיאים בצבעי זהב ושחור.
- לעיין בהוראות המשחק באמצעות לחיצה על ה item שנמצא בתפריט עם icon של ספר בצבע ירוק.
- להתנתק מהמשתמש ולחזור למסך הבית של האפליקציה – MainActivity באמצעות לחיצה על הכפתור שנמצא בצד שמאל למטה עם icon של כיבוי בצבע אדום.
- להתחיל לשחק במשחק או במוד "chaos" או במוד "classic" לפי בחירת השחקן, באמצעות לחיצה על הכפתור שנמצא במרכז המסך עם icon של חץ לבן בתוך עיגול שחור.
- לבחור זמן ביום שבאותו זמן ישלח למשתמש התראה שתזכיר לו לשחק במשחק ולשפר את הניקוד והזיכרון שלו, באמצעות לחיצה על ה item שנמצא בתפריט עם icon של פעמון בצבעי צהוב ושחור.



תפריט user

בתוך מסך הבית לפני התחלת משחק (UserActivity) מופיע התפריט - menu_user.xml .
 בתוך התפריט ישנם 2 items : How To Play ו- Alarm .
How To Play – בעת לחיצה על item זה , עם icon של ספר בצבע ירוק, נפתח למשתמש דיאלוג שמסביר על חוקי המשחק.
Alarm – בעת לחיצה על item זה , עם icon של פעמון בצבעי צהוב ושחור, נפתח למשתמש דיאלוג שבו יוכל לבחור זמן ביום, שבזמן זה ישלח למשתמש התראה שתזכיר לו לשחק במשחק במטרה לשפר את הזיכרון ואת הניקוד שלו.

**מסך טעינה לפני התחלת משחק (SplashScreenGameActivity)**

מסך זה משמש כמסך שמופיע לפני התחלת משחק. מסך זה נותן לשחקן מספר שניות להתכונן לפני תחילת המשחק. במסך זה מופעלת אנימציה לטקסט. באנימציה זו ניתן לראות את הטקסט – "READY?" מופיע לכמה שניות ואז נעלם. אחריו מופיע הטקסט "GO!" לכמה שניות ונעלם, ורק לאחר מכן מתחיל המשחק.

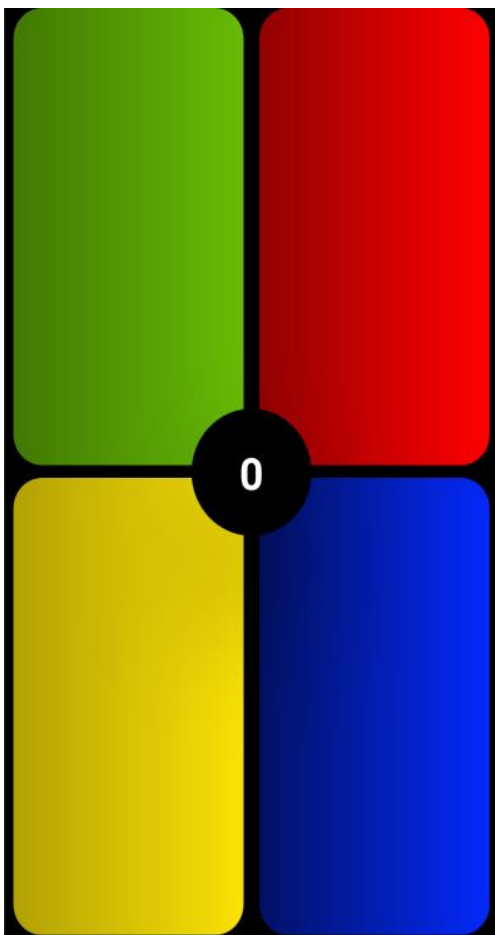


מסך המשחק (GameActivity)

מסך זה משמש כמסך המשחק. במסך זה ניתן לשחק במשחק עצמו - סיימון. באמצע המסך ישנו עיגול שחור שמראה את הניקוד הנוכחי של השחקן. לאחר שהשתמש נפסל במשחק, הוא יכול לבחור בין כמה אופציות שונות:

- לחזור למסך הבית לאחר התחברות משתמש למערכת – MainActivity, באמצעות הכפתור "MAIN MENU"
- לשחק שוב, באמצעות לחיצה על הכפתור "PLAY AGAIN"
- לשתף את הניקוד שהשחקן הגיע אליו באמצעות פלטפורמות שונות. כגון: WhatsApp, על ידי לחיצה על הכפתור "SHARE SCORE"

מה גם, המשתמש יכול לראות את הניקוד שהגיע אליו ואת השיא הנוכחי שלו.



הרשאות

- android.permission.INTERNET – נדרשת כדי לאפשר תמיכה באינטרנט.
- android.permission.ACCESS_WIFI_STATE – נדרשת כדי לאפשר גישה להגדרות WIFI של המשתמש.
- android.permission.ACCESS_NETWORK_STATE – נדרשת כדי לאפשר גישה למצב החיבור האינטרנטי של המשתמש. (האם הוא מחובר לאינטרנט או לא).
- android.permission.SET_ALARM – נדרשת כדי לאפשר שליחת התראות למשתמש.

דרישות מיוחדות ומגבלות

המערכת מחייבת תקשורת WIFI. כלומר, המערכת מחייבת את המשתמש לחיבור לאינטרנט.

גירסת Android מינימלית

Android 5.0 גירסת API 21

מכשירים עליהם נבדקה המערכת

Android 11 Google Pixel 2 XL המריץ

רפלקציה

מאז שאני זוכר את עצמי תמיד התעניינתי באפליקציות. מאז ומתמיד, רציתי לדעת כיצד מכינים אפליקציות והיה לי חלום להכין אפליקציה בעצמי. כשהגעתי לתיכון נפתחה בפני ההזדמנות להגשים את החלום. בחרתי ללמוד במגמת מחשבים וסייבר ששם למדנו לתכנת ב-JAVA ובאנדרואיד סטודיו- פיתוח אפליקציות לאנדרואיד. מרוב שהנושא עניין אותי, החלטתי להמשיך ללמוד במקביל גם בבית. פיתחתי אפליקציות בסיסות ולאט לאט עליתי ברמה. כשהגעתי לרגע בו הייתי צריך לבחור נושא לפרויקט גמר בהנדסת תוכנה, ישר ידעתי מה לבחור. בחרתי להכין את המשחק זיכרון – סיימון מכמה סיבות. דבר ראשון, בילדות שיחקתי בו כל הזמן ומאוד אהבתי את המשחק. מה גם, רציתי לעזור לאותם אנשים הסובלים מבעיות זיכרון, באמצעות המשחק. המשחק ידוע כאמצעי לשיפור הזיכרון, והוא יכול לעזור לאותם אנשים אלו.

לעבוד על הפרויקט היה בשבילי חוויה מדהימה, שלא אשכח אותה. נהניתי מכל רגע, ואין בי ספק שבעתיד אמשיך לעסוק בתחום זה.

למדתי המון מהעבודה על הפרויקט. לא רק, שהעבודה על הפרויקט הקנתה לי ידע נרחב בפיתוח אפליקציות לאנדרואיד. אלא גם, למדתי כיצד לחקור את האינטרנט במטרה להתגבר על הקשיים שנתקלתי בהם במהלך הפרויקט.

הכלים שאקח איתי להמשך הם את הידע שרכשתי בפיתוח אפליקציות לאנדרואיד ואת היכולת להתגבר על כל בעיה שאתקל בה בהמשך באופן עצמאי.

הקושי הגדול ביותר עבורי היה שלא היה לי שום ידע קודם על פיתוח אפליקציות לאנדרואיד. ולכן, עוד לפני שהתחלתי ללמוד במגמה, החלטתי ללמוד קודם כל בבית את הבסיס על פיתוח אפליקציות לאנדרואיד. לאחר מכן, כשהגעתי כבר למגמה היה לי בסיס שאיתו יכולתי לעבוד ולהתקדם בנושא.

אני מאמין כי העבודה תרמה לי המון, ושאמשיך לעסוק בנושא זה גם בשירות הצבאי ובחיים שאחרי. לו והיה לי יותר זמן לעבוד על הפרויקט, הייתי משדרג את האפליקציה. לדוגמה, הייתי מפתח אפשרות לשחקן לשחק גם בלי חיבור לאינטרנט. אך אני בטוח שברגע שיתפנה לי זמן, אני אמשיך לעבוד על האפליקציה ולהוציא ממנה את המיטב.

ביבליוגרפיה

- [/https://stackoverflow.com](https://stackoverflow.com) - Stack Overflow
- [/https://www.fxp.co.il](https://www.fxp.co.il) - Fxp
- [/https://github.com](https://github.com) - GitHub
- <https://developer.android.com/docs> - Android Developers
- [/https://www.geeksforgeeks.org](https://www.geeksforgeeks.org) - Geeks For Geeks
- [/https://www.javatpoint.com](https://www.javatpoint.com) - Java Point

נספחים

תיעוד – קוד הפרויקט

`public void createScoreTable()`

אלגוריתם היוצר את טבלת השיאים. אלגוריתם זה עובד בצורה הבאה :

- הצגת דיאלוג טעינה עד קבלת כל הנתונים מFirestore ויצירת טבלת השיאים.
- קבלת הנתונים של השחקן מ Firestore והכנסת נתונים אלו לתוך אובייקט מסוג User ששומר בתוכו את הנתונים.
- קבלת השם והשיא של השחקן בכל מוד של משחק , מהאובייקט user, והכנסת נתונים אלו לתוך אובייקט UserScore ששומר בתוכו את השם והשיא של השחקן בכל מוד.
- הכנסת האובייקט userScore לתוך המערכים של האובייקט מסוג UserScores ששומר בתוכו שני מערכים : `bestScoreClassicArray` ו `ArrayList<UserScore>` `bestScoreChaosArray` לכל מוד של משחק. מערכים אלו שומרים את השם ואת השיא של השחקן.
- מיון המערכים לפי הניקוד הגבוה ביותר.
- זימון הפונקציה `Top10()` שמרכיבה את תוכן טבלת השיאים – `msg`.
- יישום תוכן טבלת השיאים - `msg` לתוך הדיאלוג שמציג את טבלת השיאים.
- איפוס התוכן (`msg`) וביטול דיאלוג הטעינה.

במחלקה `UserActivity` ניתן גם לראות את השיא האישי של השחקן בכל מוד של משחק ובאיזה מקום הוא נמצא בטבלת השיאים. דבר זה נעשה על ידי זימון הפונקציה `showScore()`.

(ראה תמונה של הקוד למטה)

```

// יוצר את טבלת השיאים
public void createScoreTable(){
    progressBarDialog();// מציג דיאלוג טעינה

    // Firestore collection בשם "Users" שנמצאת בתוך
    // בתוך Users נמצאים Documents שכל אחד מהם שומר את ה ID של כל משתמש שמחובר למערכת, וכל ID שומר את הפרטים על המשתמש
    CollectionReference docRef = fStore.collection( collectionPath: "Users");
    docRef.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {

                // מאפס את המערכים
                userScores.clear();

                // collection Users ב documents שנמצאים ב
                for (QueryDocumentSnapshot document : Objects.requireNonNull(task.getResult())) {

                    // מקבל מה document את האובייקט של User
                    User user = document.toObject(User.class);

                    String name = user.getName();
                    int bestScoreClassic = user.getBestScoreClassic();
                    int bestScoreChaos = user.getBestScoreChaos();

                    UserScore userScore = new UserScore(name, bestScoreClassic, bestScoreChaos);
                    userScores.add(userScore);
                }

                // ממין את המערך bestScoreClassicArray מהערך (score) הגדול לקטן
                userScores.sort();

                if(btnScoreClicked){
                    Top10();
                    dialog.setMessage(msg);
                    msg.setLength(0);//(msg) השיאים
                    progressBarDialog.dismiss();// הטעינה
                    btnScoreClicked = false;
                }
                else{
                    showScore();
                }
            }
        }
    });
}

```

public void showScore()

אלגוריתם המציג את השיא האישי של השחקן בכל מוד של משחק ובאיזה מקום הוא נמצא בטבלת השיאים. אלגוריתם זה עובד בצורה הבאה :

- קבלת הנתונים של השחקן מ Firestore והכנסת נתונים אלו לתוך אובייקט מסוג User ששומר בתוכו את הנתונים.
- קבלת השיא של השחקן בכל מוד של משחק , מהאובייקט user.
- שימוש בפונקציות `getPlaceClassic()` ו `getPlaceChaos()` במטרה למצוא באיזה מקום נמצא השחקן בטבלת השיאים.
- הרכבת תוכן טבלת השיאים – msg בשיא האישי של השחקן בכל מוד של משחק ובאיזה מקום הוא נמצא בטבלת השיאים.
- יישום תוכן טבלת השיאים - msg לתוך הדיאלוג שמציג את טבלת השיאים.
- איפוס התוכן (msg) וביטול דיאלוג הטעינה.

(ראה תמונה של הקוד למטה)

```
// מראה את השיא האישי של השחקן בכל מוד של משחק
public void showScore(){
    if(fAuth.getCurrentUser() != null){ //בודק האם יש משתמש שמחובר עכשיו למערכת

        // ה ID של המשתמש המחובר למערכת
        userId = fAuth.getCurrentUser().getUid();

        // מקבל את הdocument של המשתמש (לפי ה ID של המשתמש) המחובר כרגע למערכת
        DocumentReference docRef = fStore.collection(collectionPath: "Users").document(userId);

        docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document != null) {

                        // מקבל מהdocument את האובייקט של User
                        User user = document.toObject(User.class);

                        int bestScoreClassic = Objects.requireNonNull(user).getBestScoreClassic(); // null ל שווה ל
                        int bestScoreChaos = Objects.requireNonNull(user).getBestScoreChaos(); // null ל שווה ל

                        msg.append("Mode: Classic \n").append("Best Score: ").append(bestScoreClassic).append("\n");
                        msg.append("Place: ").append(getPlaceClassic()).append("\n \n");

                        msg.append("Mode: Chaos \n").append("Best Score: ").append(bestScoreChaos).append("\n");
                        msg.append("Place: ").append(getPlaceChaos()).append("\n");

                        dialog.setMessage(msg);
                        msg.setLength(0);
                        progressDialog.dismiss();

                    } else {
                        Log.d(tag: "LOGGER", msg: "No such document");
                    }
                } else {
                    Log.d(tag: "LOGGER", msg: "get failed with ", task.getException());
                }
            }
        });
    }
}
```

public int getPlaceChaos() ו public int getPlaceClassic()

אלגוריתמים אלו מחזירים באיזה מקום השחקן נמצא בטבלת השיאים בכל מוד של המשחק. אלגוריתמים אלו עובדים בכך שהם עוברים בלולאת for על המערכים ArrayList<UserScore> bestScoreClassicArray ו bestScoreChaosArray (ששומרים בתוכם את השם ואת השיא של השחקן לאחר מיון לפי הניקוד הגבוה ביותר), ומחזירים את האינדקס כאשר השם של שחקן באינדקס מסוים במערך יהיה שווה לשם של המשתמש שמחובר כרגע למערכת.

```
// מחזיר באיזה מקום (במוד "classic") נמצא המשתמש המחובר כרגע למערכת
public int getPlaceClassic() {
    int index = 0;

    for (int i = 0; i < userScores.getSizeClassicArray(); i++) {
        if(userScores.getNameByIndexClassic(i).equals(fullName)){
            index = i;
            break;
        }
    }
    return index + 1;
}

// מחזיר באיזה מקום (במוד "chaos") נמצא המשתמש המחובר כרגע למערכת
public int getPlaceChaos() {
    int index = 0;

    for (int i = 0; i < userScores.getSizeChaosArray(); i++) {
        if(userScores.getNameByIndexChaos(i).equals(fullName)){
            index = i;
            break;
        }
    }
    return index + 1;
}
```

public void top10()

אלגוריתם היוצר את תוכן טבלת השיאים. אלגוריתם זה מציג למשתמש את 10 השחקנים עם הניקוד הגבוה ביותר. אלגוריתם זה עובד בצורה הבאה :

- בודק באיזו טבלת שיאים המשתמש מעוניין לעיין לפי מוד של משחק.
- בדיקת גודל המערך ששומר בתוכו את השם והשיא של כל משתמש.
- במידה וגודל המערך גדול מ10, אלגוריתם זה ירוץ בלולאת for מ0 עד 10(לא כולל) ויחבר לmsg (מייצג את תוכן טבלת השיאים) את המקום, השם והשיא של השחקן באותו מוד של המשחק שנמצאים בתוך המערך.
- במידה וגודל המערך קטן מ10, אלגוריתם זה ירוץ בלולאת for על המערך ויחבר לmsg (מייצג את תוכן טבלת השיאים) את המקום, השם והשיא של השחקן באותו מוד של המשחק שנמצאים בתוך המערך.

(ראה תמונה של הקוד למטה)

```
// יוצר את התוכן של טבלת השיאים
public void top10(){
    if (gameMode == null) {
        msg.append("");
    } else {
        switch (gameMode) {
            case CLASSIC:

                if(userScores.getSizeClassicArray() > 10){
                    for (int i = 0; i < 10; i++) {
                        msg.append(i + 1).append(" ").append(userScores.getNameByIndexClassic(i)).
                            append("\n Best Score: ").append(userScores.getScoreByIndexClassic(i)).append("\n \n");
                    }
                } else {
                    for (int i = 0; i < userScores.getSizeClassicArray(); i++) {
                        msg.append(i + 1).append(" ").append(userScores.getNameByIndexClassic(i)).
                            append("\n Best Score: ").append(userScores.getScoreByIndexClassic(i)).append("\n \n");
                    }
                }
                break;
            case CHAOS:

                if(userScores.getSizeChaosArray() > 10){
                    for (int i = 0; i < 10; i++) {
                        msg.append(i + 1).append(" ").append(userScores.getNameByIndexChaos(i)).
                            append("\n Best Score: ").append(userScores.getScoreByIndexChaos(i)).append("\n \n");
                    }
                } else {
                    for (int i = 0; i < userScores.getSizeChaosArray(); i++) {
                        msg.append(i + 1).append(" ").append(userScores.getNameByIndexChaos(i)).
                            append("\n Best Score: ").append(userScores.getScoreByIndexChaos(i)).append("\n \n");
                    }
                }
                break;
        }
    }
}
}
```

public void game()

אלגוריתם ללחיצת לחצן על ידי המערכת. אלגוריתם זה עובד בצורה הבאה :

ישנו מערך דינאמי - `ArrayList<COLOR> allColors` מסוג `COLOR`. `COLOR` הוא מסוג `enum` שמייצג את ארבעת הצבעים : `GREEN, RED, BLUE, YELLOW`.

המערך שומר בתוכו את כל הצבעים שנלחצו על ידי המערכת עד כה וגם מוסיף כל פעם צבע רנדומלי לאחר שהמשתמש הצליח ללחוץ על כל הצבעים לפי הרצף הנכון. האלגוריתם ללחיצת לחצן על ידי המערכת עובד בכך שהוא עובר על המערך `allColors` באמצעות לולאת `for` ולפי הצבע מפעיל אנימציה שמדמה לחיצת כפתור וסאונד שמתאים לצבע, הכפתורים נלחצים בהפרש של שנייה אחד אחרי השני.

במוד "classic" – הלחצנים נלחצים לפי הרצף שהופיע לפני וכל פעם מתווסף צבע נוסף לרצף.

במוד "chaos" – הלחצנים נלחצים לפי רצף אקראי. (לא נשמר הרצף). ישנו חלק באלגוריתם שבמידה והשחקן משחק במוד "chaos", אז עוברים בלולאת `for` על המערך `allColors` ומשנים את הצבע שהיה במקום `i` במערך בצבע רנדומלי, וכך נוצר רצף חדש ואקראי.

(ראה תמונה של הקוד למטה)


```

public void game() {
    disableButtons();

    count = 0;
    int random;

    random = rand.nextInt( bound: 4); // מספר רנדומלי מ-0 עד 4 (לא כולל)
    allColors.add(COLOR.values()[random]); //allColors מוסיף צבע רנדומלי למערך

    //game mode: chaos
    // מעדכן במערך allColors צבעים רנדומליים. הסדר שונה כל סיבוב. משנה את רצף הצבעים.
    if(gameMode.equals("chaos")){
        for (int i = 0; i < allColors.size(); i++) {
            random = rand.nextInt( bound: 4); // מספר רנדומלי מ-0 עד 4 (לא כולל)
            allColors.set(i, COLOR.values()[random]); // משנה את הצבע שהיה במקום i במערך בצבע רנדומלי
        }
    }
}

```

```

// עובר על מערך הצבעים allColors ו"לוחץ"
// (מפעיל את האנימציה של הלחיצה וסאונד מתאים לכל צבע)
// לפי רצף הצבעים שנמצא במערך בהפרש של שנייה לכל לחיצה
for (int i = 0; i < allColors.size(); i++) {
    int finalI = i;
    myHandler.postDelayed(new Runnable() {

        @Override
        public void run() {
            switch (allColors.get(finalI)) {
                case GREEN:
                    ClickGreen();
                    break;
                case RED:
                    ClickRed();
                    break;
                case BLUE:
                    ClickBlue();
                    break;
                case YELLOW:
                    ClickYellow();
                    break;
            }
        }
    }, delayMillis: 1000 * i); // דיוילי של שנייה בין כל לחיצה של כפתור
}

// אחרי שהכפתור האחרון נלחץ על ידי המערכת,
// המשתמש יוכל ללחוץ על הכפתורים (לאחר 900 מאיות)
myHandler.postDelayed(new Runnable() {

    @Override
    public void run() {
        enableButtons();
    }
}, delayMillis: 900 * allColors.size());
}

```

public void onClick(View v)

אלגוריתם שבודק האם השחקן לחץ על הצבעים לפי הרצף הנכון. אלגוריתם זה עובד בצורה הבאה:

ישנו משתנה `int count` שהוא אחראי לשמירת מספר הלחצנים שנלחצו על ידי המשתמש והוא חלק מבדיקת הלחצן של המשתמש. כלומר, ברגע שהמשתמש לוחץ על לחצן בעל צבע מסוים, המשתנה `count` גדל באחד, כלומר גדל מספר הלחצנים שנלחצו על ידי המשתמש. לאחר מכן, יש אלגוריתם שבודק האם הלחצן שנלחץ מתאים לצבע המסוים שברצף. אלגוריתם זה עובד בכך שהוא בודק האם הצבע שנמצא במקום `count` במערך `allColors` מתאים לצבע של הלחצן שהמשתמש לחץ.

`gameOver = !allColors.get(count).equals(COLOR); boolean`

במידה והצבע של הלחצן שהמשתמש לחץ מתאים לצבע המסוים שברצף, `gameOver` יהיה שווה ל-`false`, והמשתמש יוכל להמשיך לשחק במשחק.

```
public void onClick(View v) {
    if(v.equals(btnGreen)) {
        btnGreen.startAnimation(animation);
        greenSound.start();
        gameOver = !allColors.get(count).equals(COLOR.GREEN); //בודק האם הצבע שנמצא במקום count במערך allColors מתאים לצבע של הלחצן שהמשתמש לחץ
        count++;
        buttonClicked = true;
    }

    else if(v.equals(btnRed)) {
        btnRed.startAnimation(animation);
        redSound.start();
        gameOver = !allColors.get(count).equals(COLOR.RED); //בודק האם הצבע שנמצא במקום count במערך allColors מתאים לצבע של הלחצן שהמשתמש לחץ
        count++;
        buttonClicked = true;
    }

    else if(v.equals(btnBlue)) {
        btnBlue.startAnimation(animation);
        blueSound.start();
        gameOver = !allColors.get(count).equals(COLOR.BLUE); //בודק האם הצבע שנמצא במקום count במערך allColors מתאים לצבע של הלחצן שהמשתמש לחץ
        count++;
        buttonClicked = true;
    }

    else if(v.equals(btnYellow)) {
        btnYellow.startAnimation(animation);
        yellowSound.start();
        gameOver = !allColors.get(count).equals(COLOR.YELLOW); //בודק האם הצבע שנמצא במקום count במערך allColors מתאים לצבע של הלחצן שהמשתמש לחץ
        count++;
        buttonClicked = true;
    }
}
```

public void isGameOver()

אלגוריתם שרץ אחרי שהשחקן לחץ על אחד מכפתורי המשחק, ומטרתו לבדוק האם השחקן יכול להמשיך לשחק, או שהוא נפסל והמשחק נגמר. אלגוריתם זה עובד בצורה הבאה:

במידה והשחקן נפסל, הוא לא יוכל להמשיך לשחק ולכן, תופעל מנגינת הפסד ותזומן הפונקציה `score()`. במידה והשחקן לא נפסל והצליח לחזור על כל הרצף של הצבעים, כלומר, מספר הלחצנים שנלחצו על ידי השחקן (`count`) שווה לגודל המערך `allColors` (`allColors.size()`), כלומר שווה למספר הצבעים שנמצאים כרגע ברצף, אז מתווסף צבע אקראי נוסף למערך `allColors` ומופעל האלגוריתם של לחיצת לחצן על ידי המערכת.

```
// בודק האם השחקן יכול להמשיך לשחק, או שהוא נפסל והמשחק נגמר.
public void isGameOver(){
    if (gameOver) {
        lostGameSound.start();
        score();
    } else {
        if (count == allColors.size()) { // בודק האם המשתמש חזר על כל הרצף של הצבעים
            currentScore++; // מעלה בנקודה אחת את הניקוד הנוכחי של השחקן
            tvScore.setText(String.valueOf(currentScore));
            disableButtons();

            // מריץ את פונקציית ה game () לאחר 1.2 שניות
            myHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    game();
                }
            }, delayMillis: 1200); // 1000 = 1 second
        }
    }
}
```

public void score()

אלגוריתם המעדכן את השיא של השחקן במידת הצורך. אלגוריתם זה רץ לאחר שהמשתמש נפסל במשחק. האלגוריתם בודק האם הניקוד הנוכחי שהשחקן הגיע אליו במשחק גדול מהשיא שלו, במידה וכן, האלגוריתם מעדכן את השיא הנוכחי של המשתמש בFirestore באמצעות זימון הפונקציות `updateBestScoreClassic()` או `updateBestScoreChaos()` לפי המוד של המשחק.

```
// מעדכן את השיא של המשתמש במידת הצורך
// הפונקציה בודקת האם הניקוד הנוכחי שהמשתמש הגיע אליו במשחק גדול מהשיא שלו,
// במידה וכן, הפונקציה מעדכנת את השיא הנוכחי של המשתמש בFirestore
public void score(){
    progressBarDialog();
    docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
        @Override
        public void onComplete(@NonNull Task<DocumentSnapshot> task) {
            if (task.isSuccessful()) {

                DocumentSnapshot document = task.getResult();

                if (document != null) {

                    //מקבל מהdocument את האובייקט של User
                    user = document.toObject(User.class);

                    switch (gameMode){
                        case "classic":
                            bestScoreClassic = Objects.requireNonNull(user).getBestScoreClassic();
                            if(currentScore > bestScoreClassic){//ישאח הנוכחית גדולה מהשיא של השחקן
                                updateBestScoreClassic();
                            }
                            break;
                        case "chaos":
                            bestScoreChaos = Objects.requireNonNull(user).getBestScoreChaos();
                            if(currentScore > bestScoreChaos){//ישאח הנוכחית גדולה מהשיא של השחקן
                                updateBestScoreChaos();
                            }
                            break;
                    }

                    progressDialog.dismiss();
                    openLoseDialog();

                } else {
                    Log.d( tag: "LOGGER", msg: "No such document");
                }
            } else {
                Log.d( tag: "LOGGER", msg: "get failed with ", task.getException());
            }
        }
    });
}
```

public void updateBestScoreChaos() ו public void updateBestScoreClassic()

אלגוריתמים אלו מעדכנים את השיא הנוכחי של המשתמש ב-Firebase לפי מוד של משחק. אלגוריתמים אלו עובדים בצורה הבאה:

הם משנים את הערך של השיא באובייקט user הנוכחי ומעדכנים את DocumentReference (הdocument) של המשתמש (לפי ה ID של המשתמש) המחובר כרגע למערכת) עם האובייקט המעודכן.

```
// "classic" מוד של השחקן במוד
public void updateBestScoreClassic(){
    // משנה את הערך של השיא באובייקט user הנוכחי ומעדכן את DocumentReference עם האובייקט המעודכן
    user.setBestScoreClassic(currentScore);
    docRef.set(user);
}

// "chaos" מוד של השחקן במוד
public void updateBestScoreChaos(){
    // משנה את הערך של השיא באובייקט user הנוכחי ומעדכן את DocumentReference עם האובייקט המעודכן
    user.setBestScoreChaos(currentScore);
    docRef.set(user);
}
```

מסמכים נוספים

כל מסמך שכתבתם, או מצאתם במקור מידע כל שהוא שעזר לכם :

- בגיבוש הרעיון.
- בפיתוח התוכנה.
- בתיבת הקוד.
- בבדיקות התקינות.

חלק זה אופציונאלי

כללי משחק (אופציונלי)

באנגלית

The principle of the game is simple: the player has to memorize the series of illuminated keys and reproduce it.

The purpose of the game is to reproduce the longest series of colors / sounds randomly generated by the Simon.

In each round a new key is added to the series and the game becomes increasingly difficult because the player's memory is more and more solicited.

1- At the beginning of the game, one of the 4 keys lights up randomly producing

2- The player has to press the same key.

3- Next, the Simon turns back the same light on and a second one, again randomly.

4- The player has to reproduce this chain of light using his memory.

5- And so on... In each round a new key is added to the series and the game becomes all the more difficult as the player's memory is put to the test.

6- If the player doesn't make any mistake, the game goes on, so it is an endless game!

בעברית

עקרון המשחק הוא פשוט : על השחקן לשנן את סדרת המקשים המוארים ולשכפל אותה. מטרת המשחק היא להעתיק את הסדרה הארוכה ביותר של צבעים / צלילים שנוצרו באופן אקראי על ידי סיימון.

בכל סיבוב נוסף מפתח חדש לסדרה והמשחק הופך להיות קשה יותר ויותר מכיוון שזיכרון השחקן מתבקש יותר ויותר.

1- בתחילת המשחק, אחד מ-4 המקשים נדלק באופן אקראי.

2- השחקן צריך ללחוץ על אותו מקש.

3 - לאחר מכן, שמעון מדליק את אותו האור לאחר ושני, שוב באופן אקראי.

4- השחקן צריך לשחזר את שרשרת האור הזו באמצעות הזיכרון שלו.

5- וכך הלאה ... בכל סיבוב נוסף מפתח חדש לסדרה והמשחק נעשה קשה ככל שמבחנים את זיכרון השחקן.

6-אם השחקן לא עושה טעות, המשחק נמשך, כך שזה משחק אינסופי!