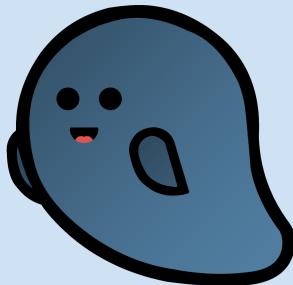


## תיק פרויקט - חלופה טלפוניים חכמים

נושא הפרויקט: **הערכת מטלות**



עבודת גמר תכנון ותוכנות מערכות **883599**

חלופה: **תוכנות טלפוניים חכמים**

מגיש: **אופק אלמוג**

תיקון שש שנתי פארק המדע

נס ציונה, תשפ"ג 2023



<b>4.</b>	<b>מבוא</b>
4.....	שם הפרויקט .....
4.....	תיאור קצר של הפרויקט .....
4.....	קהל יעד .....
4.....	הסיבות לבחירת הנושא .....
4.....	מטרות האפליקציה .....
5.....	יכולות האפליקציה .....
6.....	דרישות מיוחדות להפעלת האפליקציה .....
7.....	מכשירים עליהם האפליקציה נבדקה .....
7.....	תהליך מחקר .....
8.....	אתגרים מרכזים .....
<b>9.</b>	<b>מבנה / ארכיטקטורה</b>
9.....	תכנון ותיעוד מסכי הפרויקט .....
10.....	מסמך טעינה .....
12.....	מסמך הבית .....
17.....	מסמך הפרטים .....
20.....	מסמך חוספת פרטים .....
24.....	מסמך חוספת משימות .....
28.....	מסמך פרופיל .....
30.....	מסמך מעקב ביצוע משימה .....
32.....	מסמך צור קשר .....
34.....	תרשים מסcis - Screen flow diagram .....
36.....	תיאור מחלקות הפרויקט .....
40.....	קישור אונליין - Google Drive .....
<b>41.</b>	<b>ימוש הפרויקט</b>
42.....	Task .....
44.....	Reward .....
46.....	User .....
47.....	PickerDialog .....
49.....	GuiderDialog .....
50.....	ActivityGuideTracker .....
51.....	TimerService .....
52.....	CountdownThread .....
53.....	AirplaneModeReceiver .....
54.....	TaskAdapter .....
56.....	TaskViewHolder .....
57.....	RewardAdapter .....
59.....	RewardViewHolder .....
60.....	Utils .....
<b>61.</b>	<b>מדריך למשתמש</b>
62.....	מסמך התחרבות והרשמה .....



66.....	מיסר הבית (משימות)
69.....	מיסר הפרסים
<b>71.....</b>	<b>בוסט נתונים</b>
72.....	Shared preferences
73.....	קבלת משתמשים חדשים
74.....	מיסר הגדרות אישיות
75.....	Firebase
76.....	Authentication
77.....	Realtime Database
80.....	Storage
<b>81.....</b>	<b>רפלקטיה / סיכון אישי</b>
<b>83.....</b>	<b>ביבליוגרפיה</b>
<b>84.....</b>	<b>נספחים</b>
84.....	Appendix A: Project code in Github
85.....	Appendix B: Project implementation
85.....	Project Hierarchy
87.....	Code
265.....	Appendix C: UML diagram A3



## מבוא

### שם הפרויקט

Solobeast

### תיאור קצר של הפרויקט

האפליקציה עוזרת למשתמשים שלה לעשות מעקב אחר המשימות שעשו ובכך לTAGMAL את עצםם על העבודה הקשה שעשו. האפליקציה תתן למשתמש להערך מטלות על פי רמת קושי שהוא מגדיר וחושב לנכון. על פי זה יכול המשתמש TAGMAL את עצמו על העבודה הקשה שעשה עם תגמולים שהוא מגדיר.

האפליקציה מתוכננת כך שתתלווה את המשתמש במשך כל היום על גבי מכשירים שונים שברשותו (טלפון, טאבלט)

### קהל יעד

כל אדם שרוצה לקבל מענה על ה 'טו דו ליסטי' שקיים היום ולבצע מעקב סדר למשימות שביצע ולהגמל את עצמו בהתאם.

### הסיבות לבחירת הנושא

הסיבה המרכזית שבגלה בחרתי להכין את פרויקט זה, היא - אני. בכך השנתיים האחרונות הייתה עשו מעקב על משימות שלי, הייתה רושם על מחברת - מה התאריך היום, איזה משימות אני צריך לעשות, ומה חישבותן.

במהלך הזמן נהנתי מהתהליך זה אבל הוא התחיל ליאש כשהבנתי שאין מוטיבציה מאחורי זה - אף פעם לא קיבלתי מזה מהו והתחלתי להמעיט בשיטה זו.

הגעתי למסקנה שצריך למצוא מוטיבציה לגבי דבר שכזה ולכן התחלתי להציג פרסומים לכל שימושה שאעשה ומשמעותו דיגיטלי.

### מטרת האפליקציה

מטרת האפליקציה היא לעזור למשתמש לארגן את משימותיו, לעודד אותו בניתוח משוב וTAGMAL לאחר שסיים משימה, וכך לעזור לו לרצות לשאוף קידמה ולהתפתח בחומו האישיים.



## יכולות האפליקציה

- הצגת אнимציה בכניסה לאפליקציה
- יכולת סנכרון ועובדת על מספר מכשירים בו-זמנית
- הרשמה והתחברות עבור משתמש חדש
- זיהוי התחברות ראשונית וקבלת הסברים בתוך האפליקציה
- עבור מסכים באופן אוטומטי ללא התערבות המשתמש
- שמירת נתוני המשתמש בענן
- הוספה/מחיקת/עדכון משימה/פרס
- חישוב ניקוד לכל משימה ע"פ אלגוריהם
- טיימר כאשר משימה פועלת גם כשהמשתמש לא בהכרח על מסך האפליקציה
- התראות בזמן אמת של האפליקציה על מנת לתת מוטיבציה בזמן המשימה
- אם המשתמש נמצא על מצב טיסה כפופה את האפליקציה, ביציאה ממצב טיסה האפליקציה תסנכרן עם שרט הנתונים ותאפשר עבודה תקינה
- שליחת מייל לכותב האפליקציה על מנת לדוח על בעיות וכיו'
- עיצוב מסכים התומכים בתלפונים (מסך 6 אינץ' ומעלה) וטאבלטים



## דרישות מיוחדות להפעלת האפליקציה

### הרשאות

#### גישה לאינטרנט על מנת לגשת ל Firebase

```
<uses-permission  
    android:name="android.permission.INTERNET" />
```

#### גישה להתראות

```
<uses-permission  
    android:name="android.permission.POST_NOTIFICATIONS"/>
```

#### גישה לאחסון המכשיר בשבייל למשתמש לבחור תמונה פרופיל

```
<uses-permission  
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

#### גישה למצלמה (זמן ריצה)

```
<uses-permission  
    android:name="android.permission.CAMERA" />
```

#### דרישות הריצה של הפרויקט (build.gradle)

```
defaultConfig {  
  
    applicationId "com.example.solobeast"  
    minSdk 21  
    targetSdk 32  
    versionCode 1  
    versionName "1.0"  
  
    testInstrumentationRunner  
    "androidx.test.runner.AndroidJUnitRunner"  
}
```

ניתן לראות שגרסת ה-SDK המינימלית חייבת  
להיות 21

להרצה מיטבית של הפרויקט מומלץ לבדוק על  
גרסה +30



## מכשירים עליהם האפליקציה נבדקה

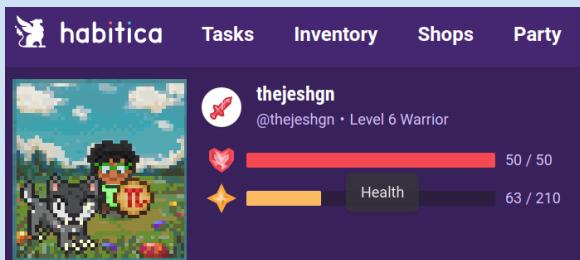
כדי לבדוק את תפקוד האפליקציה בחרוטי מכשירים שונים (חלקים מודרניים וחזקים וחלקם חלשים יותר) מבוססי אנדרואיד בגרסאות שונות.

האפליקציה מותקנת ללוות את המשתמש במהלך כל היום ולכן רציתי לוודא שהfonקציונליות מסתכרנת בין המכשירים גם בעת שימוש במקביל.

המערכת מבוססת גישה למסד נתונים בתקורת ip/tcp, הכוללת העלה והורדת תמונות. רציתי לוודא שבכל אמצעי התקורת של סלולר/וIFI ותקורת קווית זמני הגישה מתקיים באופן תקין ללא עיכוב המשתמש.  
מצורפת טבלה של המכשירים עם מערכות הפעלה וסוג הרשת עליהם נבדקה:

שם המכשיר	גרסת מערכת הפעלה	רשת
OnePlus 8	(API 33) אנדרואיד 13	Cellular/Wifi
Galaxy Tab A	(API 30) אנדרואיד 11	Wifi
OnePlus 6	(API 30) אנדרואיד 11	Cellular/Wifi
Emulator	(API 31) אנדרואיד 12	Ethernet/Wifi

## תהליך מחקר



בתחילת פיתוח הפרויקט, חקרתי וניסיתי לראות אם יש אפליקציות בשוק שהקונספט שלהם דומה לשלי, והייתה אפליקציה אחת צו בשם "Habitica" אשר הקונספט דומה אבל ככל שאתה מסיים יותר משימות הנקודות שאתה צובר הן הולכות לשיפור דמותך אוותאר שאתה מקבל כאשר אתה מתחילה לשחק באפליקציה.

**באפליקציה "Habitica"** אין יכולת להגדיר פרסים המותאמים אישית עבור המשתמש כך שהציג לא נמצא על פרודוקטיביות כמו באפליקציה שאני רציתי לעשות. כמו כן האפליקציה לא תומכת בעבודה על מספר מכשירים בו-זמנית, דבר שרציתי על מנת ללוות את המשתמש במהלך היום על גבי כל מכשיריו.



## אתגרים מרכזיים

האתגר שהיה המركזי ביותר עבורי הוא העבודה המתמשכת על הפרויקט, התחלתי את הממחקר והפיתוח של הפרויקט בתחילת אוגוסט. בכל פעם הייתה עלייה עם רעיונות חדשים ורצון להוסיף אותן לפרויקט והיו תקופות שלא עבדתי בשל עומס ואילוצים, אז לתחזק את זה היה לא פשוט.

הפייצר שהיה הכי מודע לפיתוח היה Service. תחילה הממחקר היה האerox ביותר והצריך הcyberה הרבה השקעה, לאחר מכן נתקלתי בעיות אשר מהן הייתה צריך להסביר את המשך העבודה בהתאם לארכיטקטורה ותכנון העבודה שהיא לי, כגון התראות למשתמש בהסתמך על service וכו'.



## מבנה / ארכיטקטורה

פרק זה יחולק ל-3 תת- פרקים:

- החלק הראשון מתאר את התכנון ותיעוד מסכי הפרויקט.
- החלק השני מתאר תרשימים זרימת המסכים.
- החלק השלישי מתאר את המחלקות בפרויקט והקשרים ביניהם.

הארQUITקטורה של הפרויקט תוכננה לתמוך בעבודה מקבילה במספר מכשירים בו זמנית.  
בתכנון של מסך הבית ומסך הפרסיטם, השתמשתי בהتمמשקות של Firebase  
(על מנת לעדכן/להסתنصرן עם מסד הנתונים באופן שוטף).

## תכנון ותיעוד מסכי הפרויקט

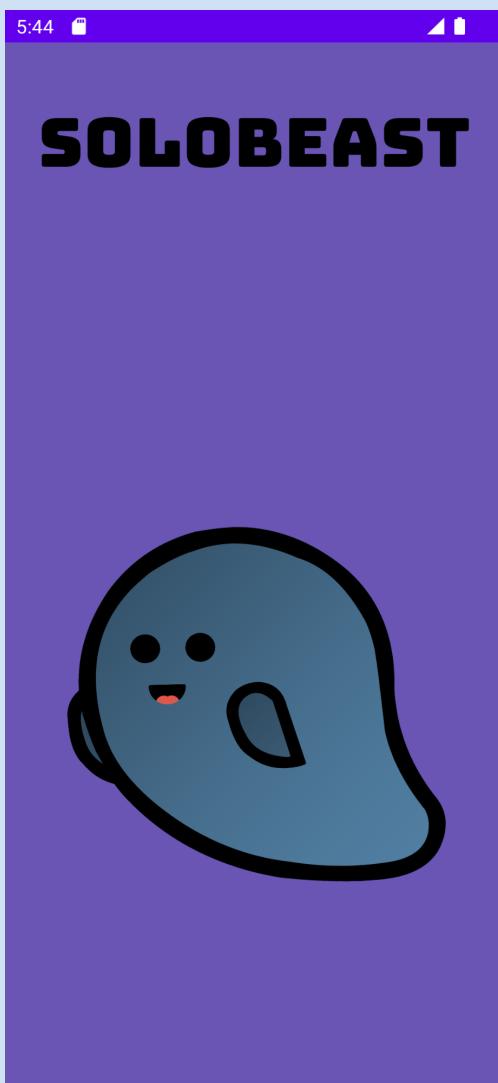
בחלק זה א塔אר את מסכי הפרויקט ותרשימים זרימותם  
אשר כל מסך כולל:

- שם המסך
- תפקיד המסך
- תיאור המסך
- תמונה מסך
- תפקידים אלמנטי התצוגה במסך



## מיסכי הפרויקט

### מסמך טעינה



זה הוא מסך הטעינה של האפליקציה, אשר יופיע בפתחת האפליקציה וככל>Anימציה של סיבוב הלוגו.

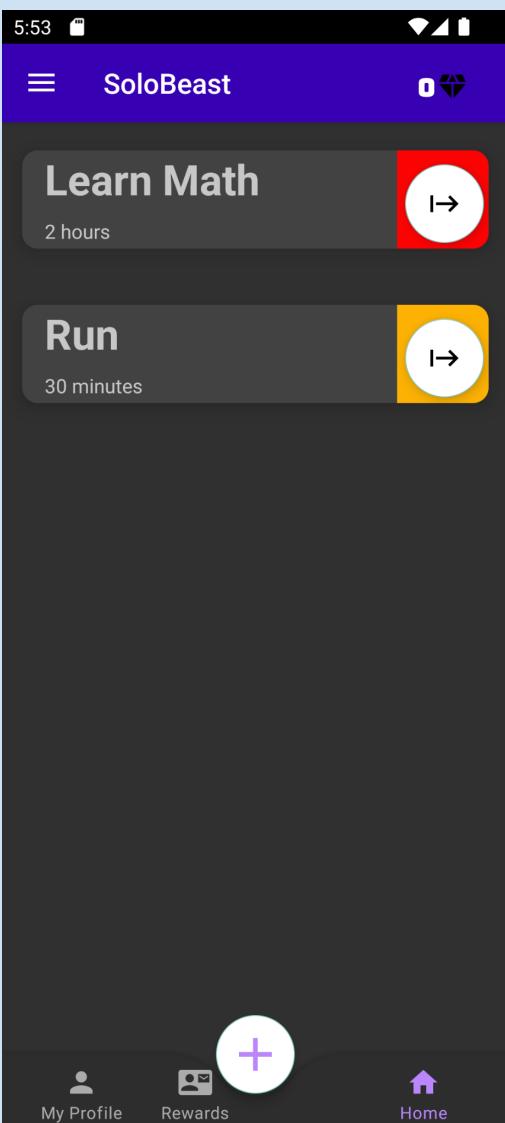
במסמך זה מופיע שם האפליקציה, והלוגו  
שעיצבתי לה 😊



שם הפעולה	הסביר
<code>void onCreate(Bundle savedInstanceState)</code>	הפעולה שפועלת בעת כניסה למסך - מתחילה את המספר - מבצעת קישור בין הרכיבים שבסך עיצוב לבין האובייקטים בקוד. - מפעילה את האנימציה כאשר האנימציה מסתיימת עוברים למסך הבית



## מסך הבית



זה הוא מסך הבית של האפליקציה, למשתמש רשום יוכל אליו ישר אחרי מסך הטעינה. במסך זה האפליקציה מציגה את כל המשימות של המשתמש.

כמויות נקודות (XP):  
בחלק העליון מצד ימין מוצג כמות הנקודות שהמשתמש צבר ביצוע מסוימות עד כה.

תפריט:  
בחלק העליון מצד שמאל ישנו סרגל כלים המאפשר פתיחת תפריט הכלל אפשריות:

- הגדרות
- onboarding
- יצירת קשר עם המפתח
- תרומה לפרויקט
- התנקות מהמשתמש המחויב

תצוגת ניווט:  
בחלק התחתון ישן כפתורים מעבר למסכים אחרים באפליקציה.

לחצן (+):  
אפשר הוספה משימה חדשה על ידי פתיחת מסך ייעודי



שם הפעולה	הסביר
<code>void onCreate(Bundle savedInstanceState)</code>	הפעולה שפועלת בעת כניסה למסך - מתחילה את המסר - מבצעת קישור בין הרכיבים שבמסך עיצוב לבין האובייקטים בקוד
<code>@Override void onUserDetailsReceived()</code>	פעולה FirebaseCallback, דורסת אותו במסמך הנ"ל. וכשהפעולה מקבלת עדכון מממד נתונים, מעדכנת את המסר בהתאם
<code>void init()</code>	פעולה שבה מתבצע הקישור בין הרכיבים. שבמסך עיצוב לבין האובייקטים בקוד. בנוסף מתחילה את עיצוב המסמך
<code>void navdrawer_init()</code>	פעולה המקשרת בין רכיב תצוגת הבניין לקוד, ומתחילה אותו
<code>@Override void onBackPressed()</code>	פעולה הדורשת את פעולה מחלוקת אב אשר מאפשרת מ-ComponentActivity, לסגור את בליחיצה על מקש החזר, התפריט.
<code>@Override public boolean onNavigationItemSelected(@NonNull MenuItem item)</code>	פעולה הדורשת את הממשק - NavigationView.OnNavigationItemSelectedListener אפשרות שליטה מלאה על הלחיצות של המשתמש כאשר לוחץ על תצוגת הבניין (navigation bar).
<code>void sign_out()</code>	פעולה המנתקת משתמש מחובר מהאפליקציה, יחוור למסך ההתחברות.



<code>@Override void onStart()</code>	פעולה המתעדכנת כאשר משתמש מתחבר/מתנתק ומודכנת את המספרים בהתאם.
<code>void replaceFragment(Fragment theInstance, Fragments enumVersion)</code>	פעולה המחליפה את תוכן המסך העיקרי באמצעות Fragment
<code>Fragments determineFragment()</code>	פעולה הקובעת על איזה Fragment אנו נמצאים ומחזירה את סוג המסך דרך enum שיצרתי במחלקה הנקרא (Fragments)
<code>@Override public void onResume()</code>	פעולה הזרוסת את מחלוקת האב FragmentActivity BroadcastReceiver ממכניסה את השימוש (registerReceiver)
<code>@Override public void onPause()</code>	פעולה הזרוסת את מחלוקת האב FragmentActivity BroadcastReceiver מוציאיה את השימוש (unregisterReceiver)
<code>static void updateXPtoUserFirebase(int xpToOperate, String operation)</code>	הפעולה מעדכנת את הnickod של המשתמש current_user-בבסיס הנתונים ובמקבלת את כמות הnickod שיש לעדכן (את סוג הפעולה (להוסף/להוריד פעולה זו פועלת ממספר מחלוקות בפרויקט



```
@Override  
public void onViewCreated(@NonNull View  
view, @Nullable Bundle  
savedInstanceState)
```

הפעולה שפועלת בעת כניסה למסך

- מתחילה את המספר
- מבצעת קישור בין הרכיבים

שבמסך עיצוב לבין האובייקטים  
בקוד.

```
@Override  
public View onCreateView(LayoutInflater  
inflater, ViewGroup container, Bundle  
savedInstanceState)
```

פעולה המקשרת בין מסך העיצוב למסך  
הראשי עליו אנו יושבים  
( MainActivity )

```
private void  
retrieveData(RecyclerViewFunctionalities  
recyclerViewFunctionalities)
```

הפעולה מדינה למסד הנתונים ומחכה  
לעדכון כלשהו, כשיש עדכון והפעולה  
תעדכן את העיצוב והנתונים של  
המשתמש בהתאם.



```
@Override  
public void onItemClick(int position)
```

הפעולה מגיבה ללחיצה של משתמש על  
משימה, מזמן את פעולה  
(moveToDetailedScreen)

```
@Override  
public boolean onItemLongClick(int  
position)
```

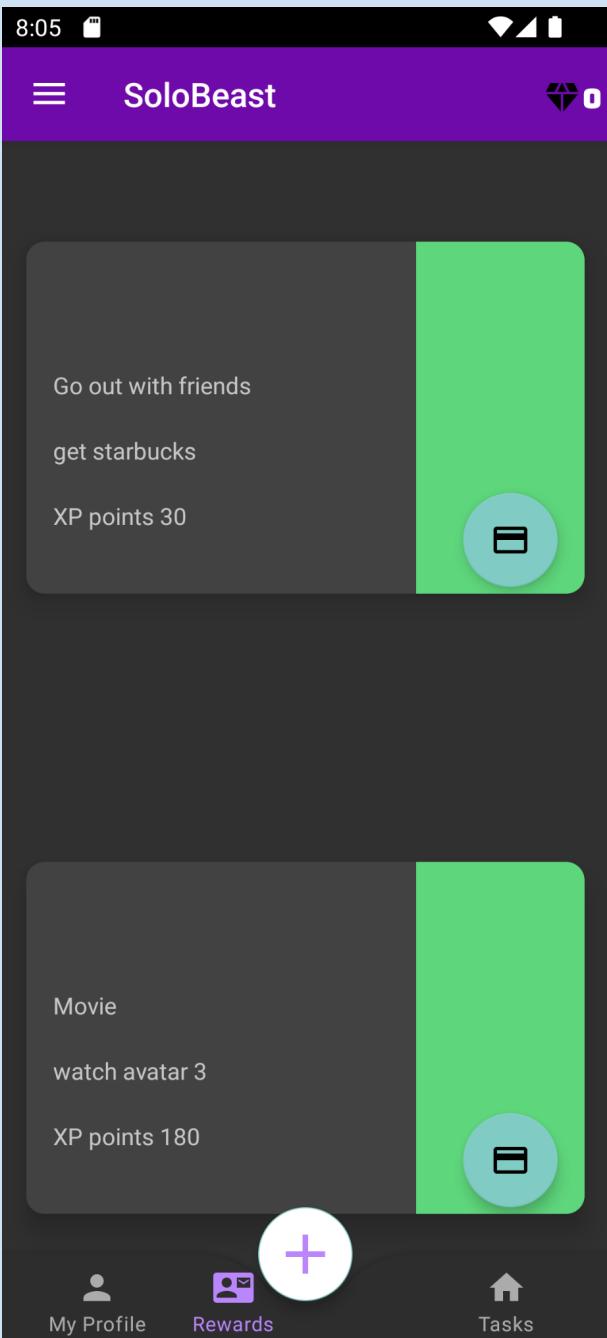
הפעולה מגיבה ללחיצה ארוכה של  
משתמש, מאפשרת מחיקה ועדכון של  
משימה בעזרת הצגת דיאלוג למשתמש

```
private void moveToDetailedScreen(Task  
task)
```

הפעולה מעבירה את המשתמש למסך  
אחר עם הערכים של המשימה הנוכחי



## מסך הפרסים



זה הוא מסך הפרסים של האפליקציה, המשמש יכול להכנס לכាបן ולצפות בפרסים שהגדיר. ברגע שיגיע למספר נקודות מסוימים, יוכל לרכוש לעצמו את הפרס שהגדיר.

נקודות נקודות (XP):  
בחלק העליון מצד ימין מוצג כמה נקודות שהמשתמש צבר ביצוע מסוימות עד כה.

תפריט:  
בחלק העליון מצד שמאל ישנו סרגל כלים המאפשרפתיתת תפריט הכלול אפשרויות:

- הגדרות
- onboarding
- יצירת קשר עם המפתח
- תרומה לפרויקט
- התנתקות מהמשתמש המחבר

תצוגת ניוט:  
בחלק התחתון ישנן כפתורים למעבר למסכים אחרים באפליקציה.

לחץ (+):  
אפשר הוספה שימוש חדשה על ידי פתיחת מסך ייעודי



שם הפעולה	הסבר
<pre>@Override public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)</pre>	הפעולה שפועלת בעת כניסה למסך - מתחילה את המספר - מבצעת קישור בין הרכיבים שבמספר עיצוב לבין האובייקטים בקוד.
<pre>@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)</pre>	פעולה המקשרת בין מסך העיצוב למסך הראשי עליו אנו יושבים (MainActivity)
<pre>private void retrieveData(RecyclerViewFunctionalities recyclerViewFunctionalities)</pre>	הפעולה מדינה למסך הנתונים ומחכה לעדכון כלשהו, כשייעדכו והפעולה תעדכן את העיצוב והנתונים של המשתמש בהתאם.



```
@Override  
public void onItemClick(int position)
```

הפעולה מוגיבה ללחיצה של  
משתמש על פרס, מזמנת את פעולה  
(moveToDetailedScreen)

```
@Override  
public boolean onItemLongClick(int  
position)
```

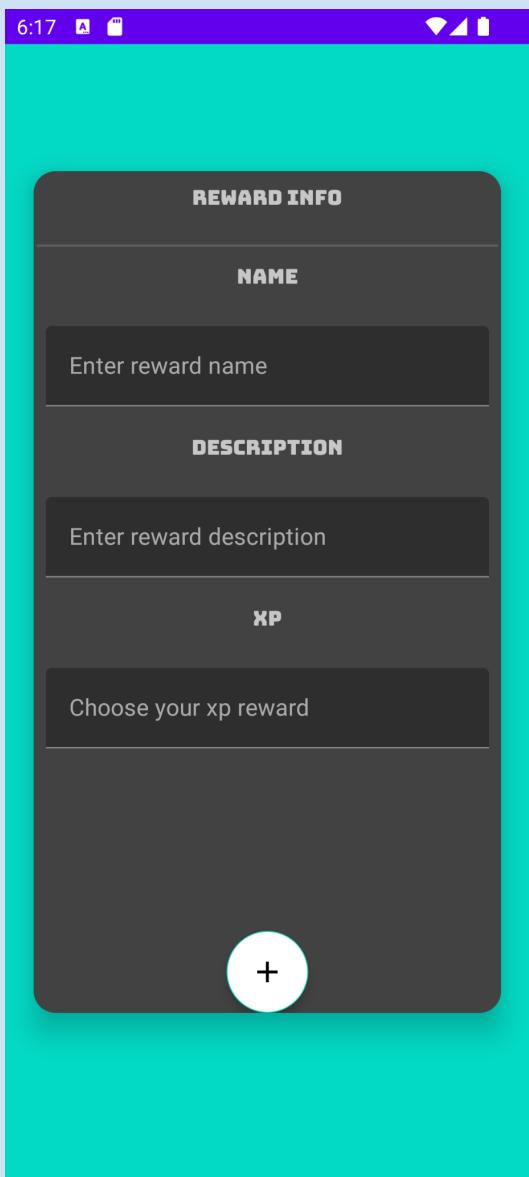
הפעולה מוגיבה ללחיצה ארוכה של  
משתמש, מאפשרת מחיקה ועדכון של  
פרס בעזרת הצגת דיאלוג למשתמש

```
private void  
moveToDetailedScreen(Reward reward)
```

הפעולה מעבירה את המשתמש למסקן  
אחר עם הערכים של הפרס הנוכחי



## מסך הוסף פרסים



זה הוא מסך הוסף הפרסים של האפליקציה, המשתמש יוכל להכנס לכאן ולהוסיף כרצונו פרסים.



שם הפעולה	הסביר
<code>@Override protected void onCreate(Bundle savedInstanceState)</code>	הפעולה שפועלת בעת כניסה למסך - מתחלה פרס חדש - מתחלה מאוזנים

<code>private void init()</code>	הפעולה מבצעת מספר דברים חשובים מיד לאחר onCreate() - מתחלה את המספר - מבצעת קישור בין הרכיבים שבמספר עיצוב לבין האובייקטים בקוד.
----------------------------------	---

<code>private void popXpSelection(View view)</code>	הפעולה מציגה דיאלוג המאפשר בחירת nikod למשתמש ומאזן לתשובה. לאחר בחירה מזין את הערך לעיצוב.
---	---

<code>private void determineEditOrAdd(String activityName)</code>	הפעולה קובעת האם המשתמש הגיע למסך כדי לעורך פרס או להוסיף אותו.
---	--



<pre>private void getValuesFromPrevActivityReward()</pre>	הפעולה מקבלת את ערכי הפרס מהמסך הקודם על מנת להציג אותם על המסך הנוכחי ( getIntent().getStringExtra() )
<pre>public int strToInt(String str)</pre>	פונקציה המחליפה string ל-int.
<pre>private void insertEditTextsValues()</pre>	הפעולה מעדכנת את ערכי תיבות הטקסט במידה והמשתמש הגיע למסך 'עדכן' פרס.
<pre>private void updateRewardToFirebase()</pre>	הפעולה מעדכנת את פרס הקיים במסד הנתונים
<pre>private void addRewardToFirebase()</pre>	



	הפעולה מוסיפה את הפרס למסד הנתונים
--	------------------------------------

`private void setAddIconDrawable()`

הפעולה מחליפה את האיקון ל'הוספה' במידה והמשתמש מנסה להוסיף ולא לעדכן.

`private void setUpdateIconDrawable()`

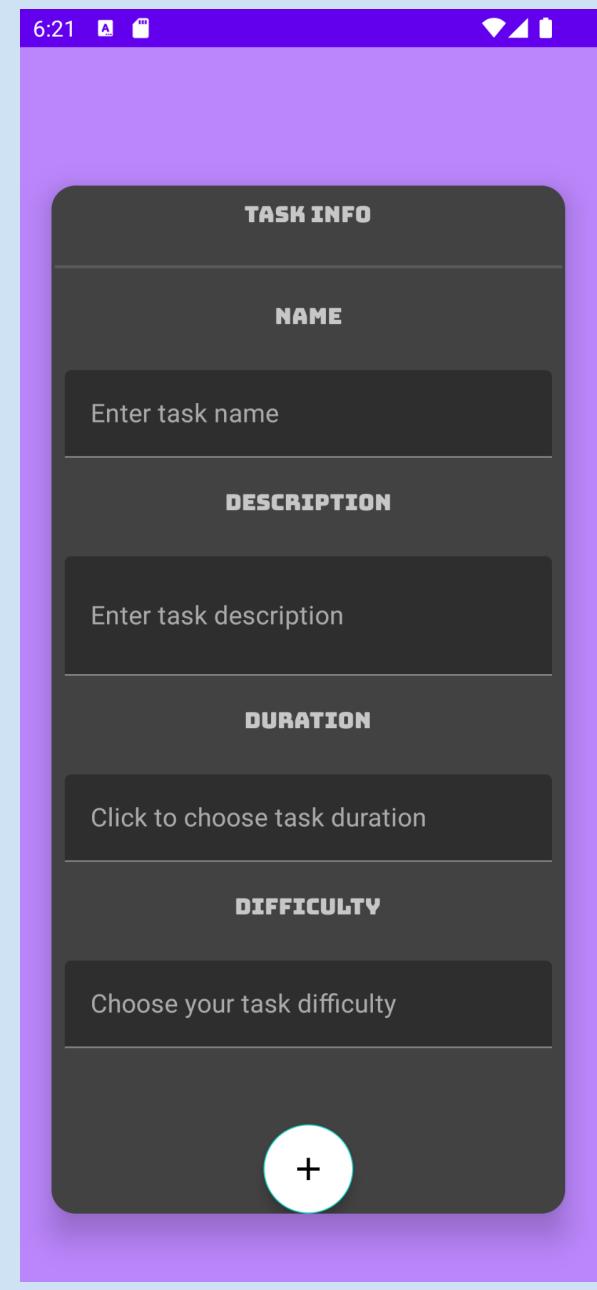
הפעולה מחליפה את האיקון לעדכון במידה והמשתמש מנסה לעדכן מנסה ולא להוסיף

`private void openEditTexts()`

הפעולה פותחת את התיבות טקסט לכתיבה ועריכה שלהן, לאחר לחיצה על כפתור העריכה



## מסך הוסף משימות



זה הוא מסך הוסף משימות של האפליקציה, המשתמש יוכל להכנס לכائנו ולהוסיף כרצונו משימות.



שם הפעולה	הסביר
<code>@Override protected void onCreate(Bundle savedInstanceState)</code>	הפעולה שפועלת בעת כניסה למסך - מתחלה מחדש חדשה - מתחלה מאוזנים

<code>public void popTimePicker(View view)</code>	הפעולה מציגה דיאログ המאפשר בחירת זמן למשימה
---	---

<code>public void popDiffSelection(View view)</code>	הפעולה מציגה דיאלוג המאפשר בחירת קווי למשימה
--	---

<code>private void init()</code>	הפעולה מבצעת מספר דברים חשובים מיד לאחר onCreate() - מתחלה את המסך - מבצעת קישור בין הרכיבים שבמסך עיצוב לבין האובייקטים בקוד.
----------------------------------	---



```
private void determineEditOrAdd(String activityName)
```

הפעולה קובעת האם המשתמש הגיע  
למסך כדי לערוך משימה או להוסיף  
אותה.

```
private void  
getValuesFromPrevActivityReward()
```

הפעולה מקבלת את ערכי המשימה  
מהמסך הקודם על מנת להציג אותם על  
מסך בעזרת  
(`getIntent().getStringExtra()`)

```
private void insertEditTextsValues()
```

הפעולה מעדכנת את ערכי תיבות  
הテקסט במידה והמשתמש הגיע למסך  
'עדכון' משימה.

```
private void updateTaskToFirebase()
```

הפעולה מעדכנת את משימה הקיימת  
במסד הנתונים



`private void addTaskToFirebase()`

הפעולה מוסיפה את המשימה למאד  
הנתונים

`private void setAddIconDrawable()`

הפעולה מחליפה את האיקון ל'הוספה'  
במידה והמשתמש מנסה להוסיף ולא  
לעדכן.

`private void setUpdateIconDrawable()`

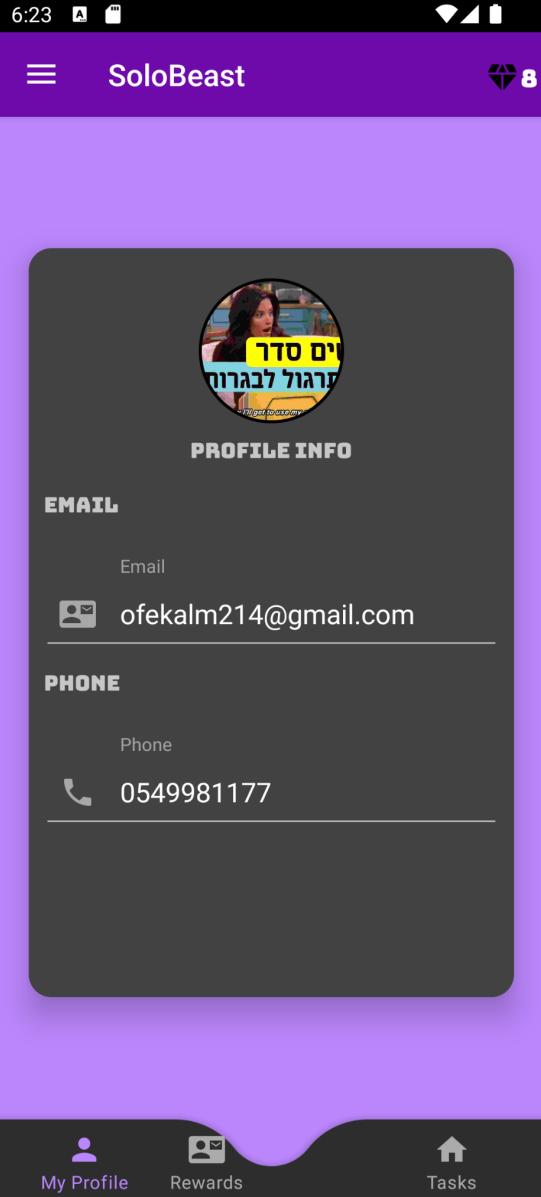
הפעולה מחליפה את האיקון לעדכון  
במידה והמשתמש מנסה לעדכן מנסה  
ולא להוסיף.

`private void openEditTexts()`

הפעולה פותחת את התיבות טקסט  
לכתיבה ועריכה שלhn, לאחר לחיצה על  
כפתור העריכה.



## מסך פרופיל



זה הוא מסך פרופיל של המשתמש, המשתמש יוכל להכנס לכיסו ולצפות בתמונה שהגדר ובספרטיו האישיים.

כמויות נקודות (XP): בחלק העליון מצד ימין מוצג כמות הנקודות שהמשתמש צבר ביצוע ממשימות עד כה.

תפריט: בחלק העליון מצד שמאל ישנו סרגל כלים המאפשרפתיתת תפריט הכלל אפשריות:

- הגדרות
- onboarding
- יצירת קשר עם המפתח
- תרומה לפרויקט
- התנהלות מהמשתמש המחבר

תצוגת ניוטן: בחלק התחתון ישנן כפתורים למעבר למפסים אחרים באפליקציה.



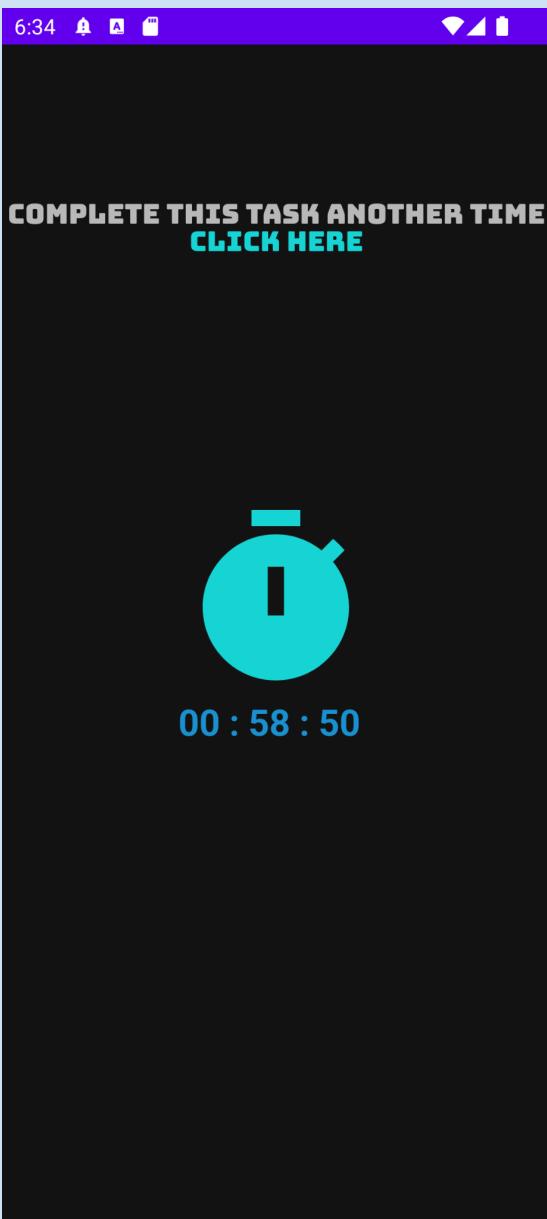
שם הפעולה	הסביר
<pre>@Override public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState)</pre>	הפעולה שפועלת בעת כניסה למסך - מתחילה את המספר - מבצעת קישור בין הרכיבים שבמסך עיצוב לבין האובייקטים בקוד.
<pre>@Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)</pre>	פעולה המקשרת בין מסך העיצוב למסך הראשי עליו אנו יושבים ( MainActivity )
<pre>public void getImageFromFireBase()</pre>	פעולה מקבלת את התמונה של המשתמש מממד הנתונים, מעדכנת אותה על המסך ברגע שנטענה.
<pre>public String getEmail()</pre>	פעולה המחזיר את האימייל של המשתמש כ-String



```
public void getUserDetails()
```

פעולה מקבלת ממseed הנתונים את הפרטים על המשתמש.

## מסמך מעקב ביצוע משימה



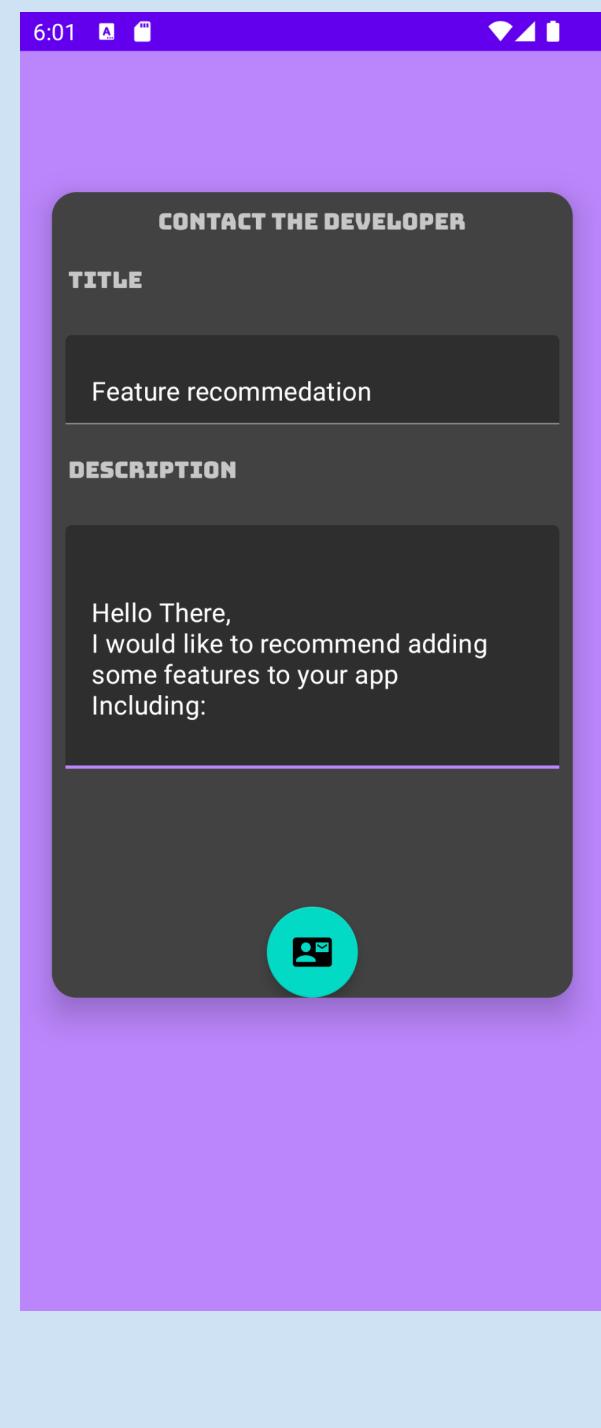
מסך זה מציג את הזמן הנותר לביצוע המשימה של המשתמש.  
בלחיצה על הטקסט לעילו, המשימה תושהה והמשתמש יחזור למסך הבית.



שם הפעולה	הסביר
<code>void onCreate(Bundle savedInstanceState)</code>	<p>הפעולה שפועלת בעת כניסה למסך</p> <ul style="list-style-type: none"> <li>- מתחילה את המספר</li> <li>- מבצעת קישור בין הרכיבים</li> <li>- שבמספר עיצוב לבין האובייקטים בקוד.</li> <li>- מפעילה את האנימציה</li> <li>- כאשר האנימציה מסתיימת עוברים למסך הבית</li> </ul>
<code>private int calculatePointsForTask(int taskTimeInMinutes, String difficulty)</code>	<p>פעולה המחשבת את כמות הנקודות על פי הזמן של המשימה והקושי שלה.</p>
<code>private void scheduleAnotification(long timeAsMilliseconds)</code>	<p>פעולה הקובעת זמני שליחת התראות למשתמש בעת עשיית המשימה לפחות מוגדרות מראש ולפי זמן המשימה של המשתמש ברגע שהזמן שנקבע יגיע, התראה תוצג על הטלפון של המשתמש.</p>
<code>private void sendNotificationToUser(String title, String msgContent)</code>	<p>פעולה המקבלת כותרת התראה והודעה ומתריעה למשתמש</p>
<code>private void init()</code>	<p>פעולה שבה מתבצע הקישור בין הרכיבים. שבמספר עיצוב לבין האובייקטים בקוד. בנוסף מתחילה את עיצוב המסך</p>
<code>private int indicateTimeAsSeconds(String time)</code>	<p>פעולה המחזירה את מספר שניות מהפורמט שמתקיים (HH:MM)</p>



## מסמך צור קשר



זה הוא מסך הצור קשר של האפליקציה, המשמש יכול להכנס לכאן לשלוח הודעה אל מפתח האפליקציה בנוגע לבאגים, תאיימות מכשירים, המלצות לשיפור ושיתופי פעולה.



בלחיצה על הפתור - צור קשר, מתאפשרות האופציות הבאות:

- העתקת הטקסט
- שיתוף לאפליקציות המותקנות במכשיר(Gmail, Whatsapp, Discord, Telegram)

\*בחירה שיתוף במייל, מוגדר מייל נמען כבחירה מחדל אל המפתח

Hello There,  
I would like to recommend adding some features...

Copy      Nearby

No recommended people to share with

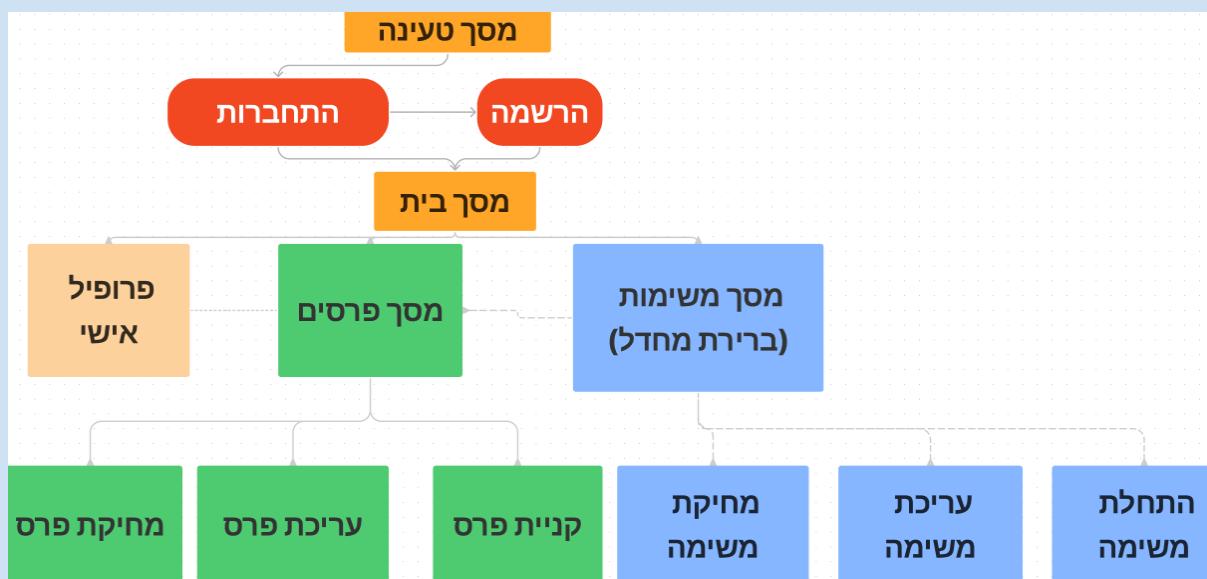
Gmail      Bluetooth      Drive  
Save to Dri...

שם הפעולה	הסביר
<code>void onCreate(Bundle savedInstanceState)</code>	הפעולה שפועלת בעת כניסה למסך <ul style="list-style-type: none"><li>- מתחילה את המספר</li><li>- מדינה לחיצה על כפתור, שנלחץ תפתח אופציה לשילוח המail שהוזן באפליקציה.</li></ul>
<code>void init()</code>	פעולה שבאה מתבצע הקישור בין הרכיבים. שבסך עיצוב לבין האובייקטים בקוד



## תרשים מסכימים - Screen flow diagram

מצורף תרשים מסכימים שמתאר את היררכיה המסכימים והמעברים ביניהם

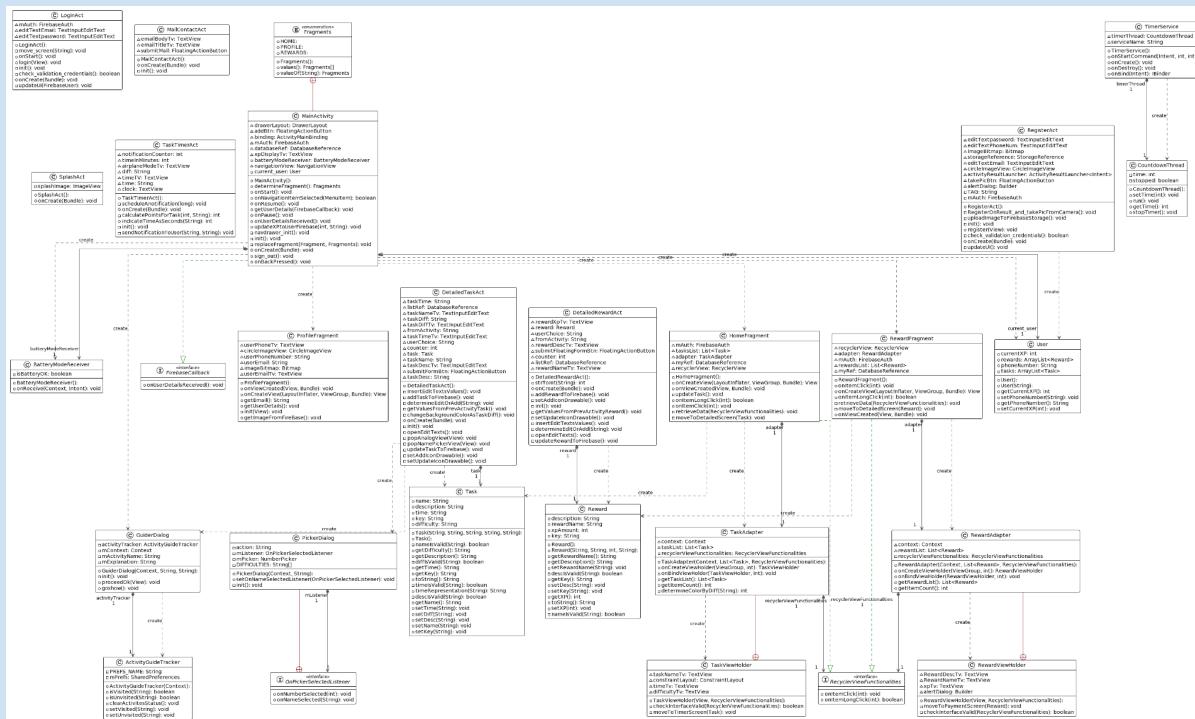






## תיאור מחלקות הפרויקט

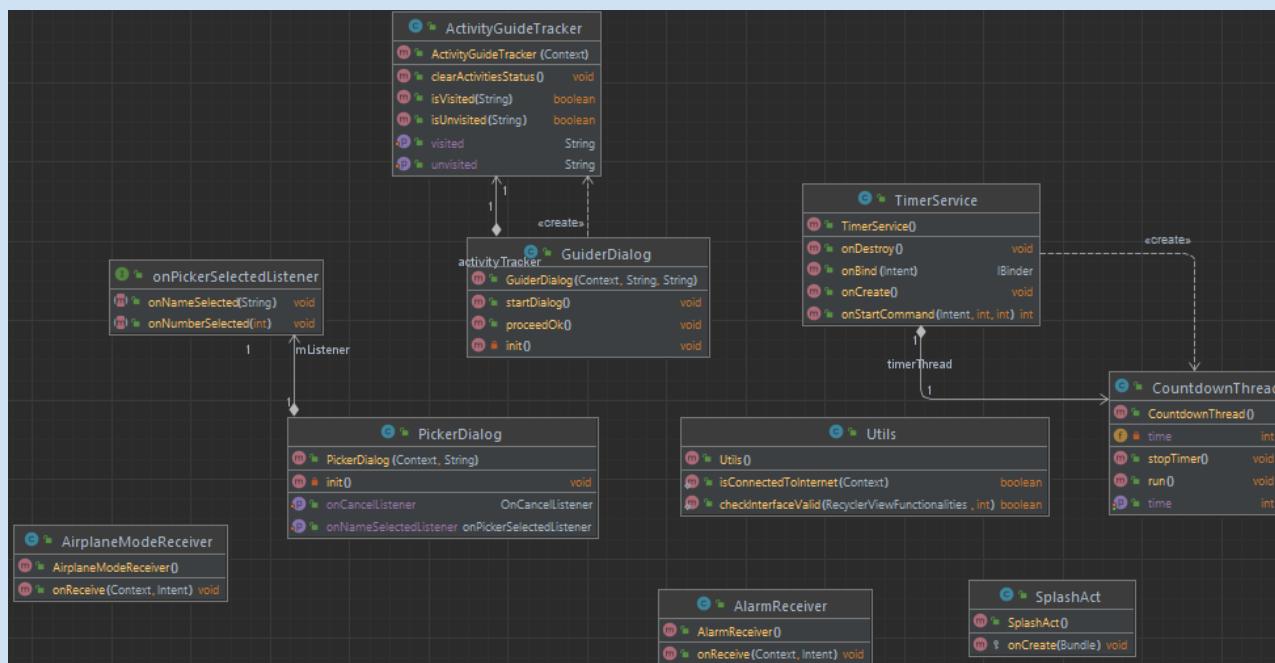
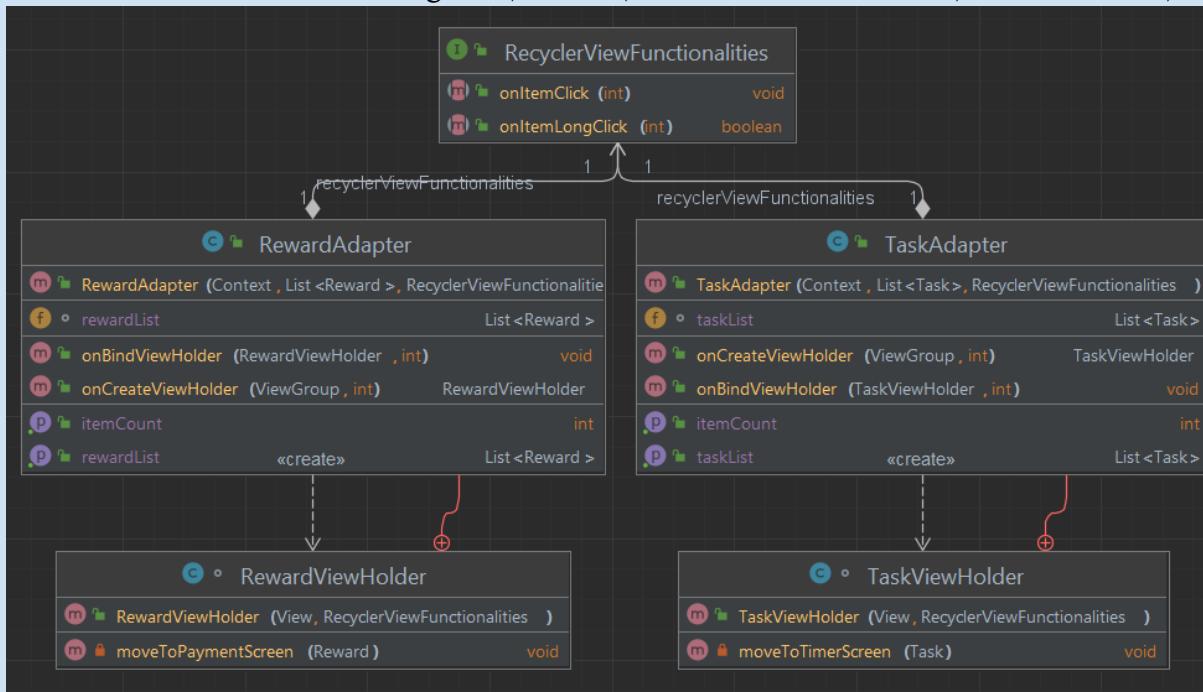
בחלק זה אפרט על התיאור של ה-UML של המחלקות והקשרים ביניהם



על מנת לראות בrzולוציה גבואה יותר - ראה **בשפח** בספר זה. (כניתן גם לראות אונגליוןaban)



## ניתן לראות גם בחלוקתם אשר מסודרים לפי חלקים הפרויקט (Packages)

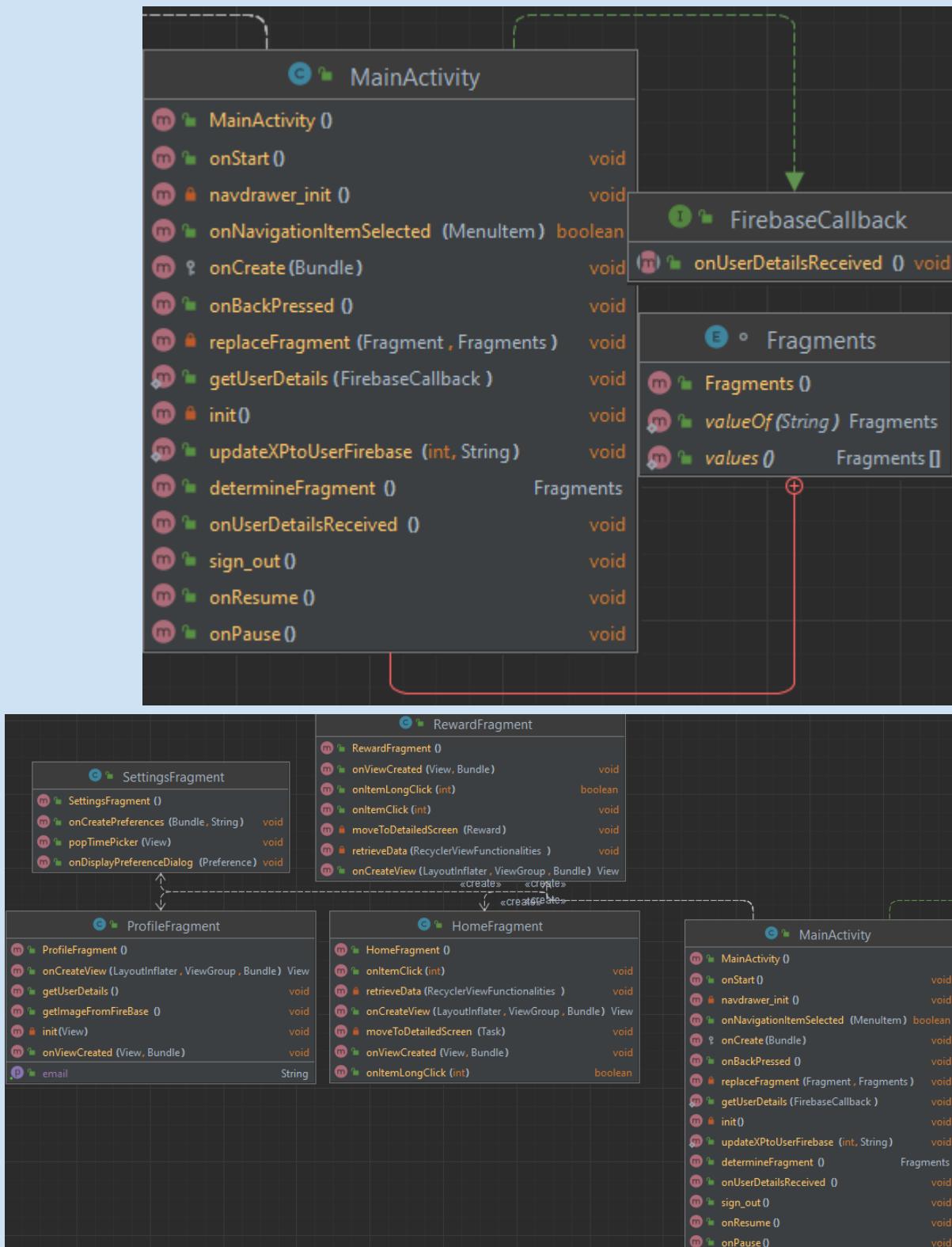




Task	Reward	User
m Task(String, String, String, String, String)	m Reward(String, String, int, String)	m User()
m Task()	m Reward()	m User(String)
f name String	f xpAmount int	f phoneNumber String
f time String	f rewardName String	f currentXP int
f difficulty String	f description String	m getCurrentXP() int
f key String	f key String	m setCurrentXP(int) void
f description String	m getKey() String	m setPhoneNumber(String) void
m setKey(String) void	m getDescription() String	m getPhoneNumber() String
m getKey() String	m setKey(String) void	m toString() String
m diffIsValid(String) boolean	m getXP() int	
m getDifficulty() String	m setXP(int) void	
m descIsValid(String) boolean	m setRewardName(String) void	
m getDescription() String	m getRewardName() String	
m getTime() String	m descIsValid(String) boolean	
m changeColorAsTaskDifficulty(String) String	m xpIsValid(int) boolean	
m setName(String) void	m toString() String	
m namesIsValid(String) boolean	m setDescription(String) void	
m toString() String	m namelsValid(String) boolean	
m setTime(String) void		
m getName() String		
m setDiff(String) void		
m timesIsValid(String) boolean		
m setDesc(String) void		
m timeRepresentation(String) String		
<hr/>		
LoginAct	RegisterAct	
m LoginAct()	m RegisterAct()	
m login(View) void	m askForUserPermissions() void	
m onCreate(Bundle) void	m check_validation_credentials() boolean	
m updateUI(FirebaseUser) void	m RegisterOnResult_and_takePicFromCamera() void	
m check_validation_credentials() boolean	m showImagePickerFromCamera() void	
m init() void	m updateUI() void	
m onStart() void	m init() void	
m move_screen(String) void	m drawableToBitmap(Drawable) Bitmap	
<hr/>		



 TaskTimerAct	
 TaskTimerAct ()	
 <code>indicateTimeAsSeconds (String)</code>	int
 <code>sendNotificationToUser(String, String)</code>	void
 <code>calculatePointsForTask(int, String)</code>	int
 <code>onCreate(Bundle)</code>	void
 <code>init()</code>	void
 <code>scheduleAnotification(long)</code>	void
 MailContactAct	
 MailContactAct ()	
 <code>onCreate(Bundle)</code>	void
 <code>init()</code>	void
 DetailedRewardAct	
 DetailedRewardAct()	
 <code>init()</code>	void
 <code>setUpdateIconDrawable()</code>	void
 <code>getValuesFromPrevActivityReward ()</code>	void
 <code>addRewardToFirebase()</code>	void
 <code>strToInt(String)</code>	int
 <code>updateRewardToFirebase()</code>	void
 <code>onCreate(Bundle)</code>	void
 <code>insertEditTextsValues()</code>	void
 <code>setAddIconDrawable()</code>	void
 <code>popXpSelection(View)</code>	void
 <code>openEditTexts()</code>	void
 <code>determineEditOrAdd(String)</code>	void
 DetailedTaskAct	
 DetailedTaskAct()	
 <code>popDiffSelection(View)</code>	void
 <code>onCreate(Bundle)</code>	void
 <code>init()</code>	void
 <code>popTimePicker (View)</code>	void
 <code>setAddIconDrawable()</code>	void
 <code>updateTaskToFirebase()</code>	void
 <code>addTaskToFirebase()</code>	void
 <code>openEditTexts()</code>	void
 <code>setUpdateIconDrawable()</code>	void
 <code>determineEditOrAdd(String)</code>	void
 <code>insertEditTextsValues()</code>	void
 <code>getValuesFromPrevActivityTask ()</code>	void



# **קישור אונליין - Google Drive**

להלן קישור לדיאגרמה על ידי אתר ייעודי (Drive) [כאן](#)



## מימוש הפרויקט

ב חלק זה א塔אר את מחלקות הפרויקט.

עבור כל מחלקה בפרויקט אתאר את:

- שם מחלקה.
- תפקיד המחלקה.
- הסבר כל תוכנות המחלקה. (תפקיד ותחום הכרה)
- הסבר של הפעולות במחלקה.



## מחלקות הפרויקט

### Task

תפקיד המחלקה הוא להגדיר את האובייקט "משימה". אשר כולל את המידע של המשימה, המשתנים שלה, והפונקציות שלה.

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם `private` על כן שינויים יכולים להתבצע רק בתחום המחלקה.
- עבור מקרים בהם נדרש גישה למשתנים ישנים `access modifiers` שהינם `public` על מנת לעשות שינויים לתוכנות.
- ישן פעולות של ולידציה אשר הן `private` ונתונות לשימוש רק על ידי תוכנות המחלקה.
- כל הפעולות הסטטיות מוגדרות כ-`public` וכך יש אופציה לעשות בהן שימוש במחלקות האחרות.

יש לשים לב שלמחלקה יש annotation חשוב בתחילתה:

`@IgnoreExtraProperties` -  
אשר עוזרת למסד הנתונים לקבל אינדיקציה לשומר רק את תוכנות המחלקה.

#### מאפיינים/תכונות

String time	מייצג את זמן המשימה
String name	מייצג את שם המשימה
String description	מייצג את תיאור המשימה
String key	עבור המשימה ( <code>unique</code> ) מייצג את המפתח לצורך מציאת המשימה מסד הנתונים

#### פעולות

<code>public Task()</code>	פעולה בונה דיפולטיבית עבור מסד הנתונים ( <code>Firebase</code> )
<code>public Task(String name, String time, String description, String difficulty, String key)</code>	פעולה בונה ברירת מחדל מקבלת את התכונות של המשימה
<code>boolean diffIsValid(String difficulty)</code>	פעולה הבודקת אם הקושי הנבחר תקין
<code>boolean descIsValid(String description)</code>	פעולה הבודקת אם התיאור שנכתב תקין במידה ו-ריך מחזיר שקר



boolean nameIsValid(String name)	פעולה הבודקת אם שם המשימה שנכתב תקין במידה ו-רייך מחזיר שקר
boolean timeIsValid(String time)	פעולה הבודקת אם הזמן שהוזן תקין במידה ו-רייך מחזיר שקר
static String timeRepresentation(String time)	(HH:MM) פעולה מקבלת זמן בפורמט ומ>Returns את הזמן בכתב  לדוגמה: 00:10 ייחזר: 10 דקות
static String changeColorAsTaskDifficulty(String diff)	פעולה מקבלת את קושי המשימה ומ>Returns לפיו את הצבע שלו



## Reward

תפקיד המחלקה הוא להגדיר את האובייקט "פרס". אשר כולל את המידע של הפרס, המשתנים שלו, והפונקציות שלו

### תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם private על כן שינוים יכולם להתבצע רק בתחום המחלקה.
- עבור מקרים בהם נדרש גישה למשתנים ישנים access modifiers שהינם public על מנת לבצע שינויים לתוכנות.
- ישן פעולות של וילדציה אשר הן private ונתונות לשימוש רק על ידי תכונות המחלקה.

יש לשים לב שלמחלקה יש annotation חשוב בתחילתה:

@IgnoreExtraProperties  
אשר עוזרת למסד הנתונים לקבל אינדיקציה לשמור רק את תכונות המחלקה.

מאפיינים	
מייצג את כמות הניקוד על מנת לקנות את הפרס	int xpAmount
מייצג את שם הפרס	String rewardName
מייצג את תיאור הפרס	String description
מייצג את המפתח (unique) עבור המשימה לצורך מציאת המשימה במסד הנתונים	String key

פעולות	
פעולה בונה דיפולטיבית עבור מסד הנתונים (Firebase)	()public Reward
פעולה בונה ברירת מחדל Reward מקבלת את התכונות של reward	public Reward(String name, String desc ,int xp, String key)
פעולה הבודקת אם התיאור שנכתב תקין במידה ו-ריק מחזיר שקר	(boolean descIsValid(String description
פעולה הבודקת אם שם המשימה שנכתב תקין במידה ו-ריק מחזיר שקר	(boolean nameIsValid(String name





## User

תפקיד המחלקה הוא להגדיר את האובייקט "User". אשר כוללת את המידע של המשתמש.

### תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם private על כן שינוייהם יכולים להתבצע רק בתחוםי המחלקה.
- עבור מקרים בהם נדרש גישה למשתנים ישנים access modifiers שהינם public על מנת לעשות שינויים לתוכנות.

מאפיינים	
מייצג את הטלפון של המשתמש	String phoneNumber
מייצג את כמות הנקודות שיש למשתמש	int currentXP

פעולות	
פעולה בונה דיפולטיבית עבור מסד הנתונים (Firebase)	()public User
פעולה בונה בריית מחרזת מקבלת את הטלפון של המשתמש ומdfsת את הנקודות	(public User(String phoneNum



## PickerDialog

תפקיד המחלקה הוא ליצור דיאלוג המותאם אישית לצורכי האפליקציה.

יורש ממחלקה:  
Dialog -

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם private על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
מייצג מערך עם קשיי המשימות	static final String[] DIFFICULTIES
מייצג הפעולה שנבחרה לביצוע	final String action
משמש לביצוע callback כאשר משתמש מזין ערך	onPickerSelectedListener mListener
מייצג אלמנט עיצוב לבחירת אופציה (לדוגמא מספר וכיו)	NumberPicker mPicker

פעולות	
פעולה בונה מקבלת את הפעולה אשר המשתמש בחר לבצע עם הדיאלוג. ומפעילה את פעולה init	public PickerDialog(Context context, String action)
פעולה הראשית של המחלקה, מחזיקה את כל הfonctionalities.  - מתחילה את המסך. - מבצעת את הקישור בין הרכיבים שבמסך העיצוב לבין הרכיבים בקדול. - מגדרה את ערכי האופציות mPicker על פי הaction שהמשתמש בחר. - מחזירה callback כשהמשתמש מאשר את האופציה שבחר.	void init()
פעולה המגדירה את callback. ברגע שהמשתמש בחר אופציה, היא תתעדכן	void setOnNameSelectedListener(onPickerSelectedListener listener)
פעולה המגדירה את הfonctionalities של	void setOnCancelListener(@Nullable



המשך העבודה כאשר משתמש לא בוחר  
אפשרה וambil

OnCancelListener listener)



## GuiderDialog

תפקיד המחלקה הוא ליצור דיאלוג המסביר על תוכן המסר למשתמש שראה אותה פעמי ואשונה.

יורש ממחלקה:  
Dialog -

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם `private` על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
Context mContext	מייצג את המסך עליו יראו את ההסבר
String mActivityName	מייצג את שם המסך עליו רשום ההסבר
String mExplanation	הסביר על המסך המוצג למשתמש
ActivityGuideTracker activityTracker	אובייקט המייצג מסד נתונים לוקאלי העוקב אחר איזה מסכים המשתמש ראה

פעולות	
public GuiderDialog(Context context, String activityName, String explanation)	פעולה בונה מקבלת את המסך עליו יוצג מסך הסביר, את השם של המסך עליו מסבירים, ואת הסביר עצמו.
void init()	הפעולה:  - מתחילה את המסך. - מציגה את הדיאלוג - מבצעת את הקישור בין הרכיבים שבמסך העיצוב לבין הרכיבים בקוד. - ברגע שימושו אישר שראה את הסביר סוגרת את הדיאלוג על ידי קרייה <code>proceedOk</code> לפעולה
void startDialog()	הפעולה שקוראים לה על מנת להתחיל את הפעולות של המחלקה - בזדמת במידה והמסך לא נצפה



	שחציג את המסך init תקרה לפעולה
void proceedOk()	הפעולה תסגור את מסך הדיאלוג



## ActivityGuideTracker

תפקיד המחלקה הוא לשמר סטטוס של צפיפות המ██ים בעזרת SharedPreferences

### תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם **private** על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
static final String PREFS_NAME	מייצג את השם של הקובץ עליו ישמרו הנתונים
SharedPreferences mPrefs	מייצג את הגישה לקובץ

פעולות	
public ActivityGuideTracker(Context context)	يוצר אובייקט של המחלקה, מתחילה את האפשרות להשתמש באובייקט הנוצר.
public boolean isVisited(String activityOrFragmentName)	מקבל את שם המ██ עליו יבודוק אם נצפה, במידה וכן יחזיר אמת אחרת שקר.
public boolean isUnvisited(String activityOrFragmentName)	מקבל את שם המ██ עליו יבודוק אם לא נצפה, במידה ולא יחזיר אמת אחרת שקר.
public void setVisited(String activityOrFragmentName)	מנגידיר מסך כנצפה
public void setUnvisited(String activityOrFragmentName)	מנגידיר מסך שלא נצפה
public void clearActivitiesStatus()	מנקה את הערכיהם מהקובץ של אובייקט המחלקה



## TimerService

תפקיד המחלקה הוא לחתם שירות שעבוד בركע של האפליקציה, משמשותה במידה והמשתמש לא נמצא על מסך האפליקציה באותו עת, חישובו של שירות זה עדין יעבדו. למרות שהאפליקציה לא מוצגת.

### תחומי ההכרה של המחלקה:

- כל המשתנים במחלקה הינם `private` על כן שינוים יכולים להתבצע רק בתחוםי המחלקה.

מאפיינים	
CountdownThread timerThread	מייצג את האובייקט ( <code>CountdownThread</code> ) עליו נעשית הלוגיקה של השירות.
String serviceName = "TimerServices"	מייצג את השם של השירות. מאותחל כברירת מחדל ל- ("TimerServices") ולא משתנה.

פעולות	
public TimerService()	פעולה בונה ברירת מחדל.
@Override public void onCreate()	פעולה המבוצעת כשפותחים שירות חדש.
@Override public int onStartCommand(Intent intent, int flags, int startId)	פעולה המבוצעת כאשר השירות מתחיל, מאותחל את האובייקטים ומתחילה את הטיימר.
@Override public IBinder onBind(Intent intent)	פעולה החייבת להיות בקוד הפרויקט בשבייל לעבור עם השירות, <b>תחריר</b> ( <code>UnsupportedOperationException</code> )
@Override public void onDestroy()	פעולה המבוצעת כאשר השירות הסתיים.



## CountdownThread

תפקיד המחלקה הוא להריץ תהליך על פי זמן מסוים שהמשתמש הזין המחלקה פועלת בעזרת עזרת מחלקה [TimerService](#)

ירוש ממחלקה:  Thread -

### תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם **private** על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
int time	מייצג את השם של הקובץ עליו ישמרו הנתונים
boolean stopped = false	אובייקט בולי אני אשר מייצג האם הטימר הסתיים, תחילתו תמיד מאוחר כשלך.

פעולות	
@Override public void run()	הפעולה מתחילה כאשר התהליכיון מתחילה, וסופרת את הזמן.
public int getTime()	מחזיר את תוכנת המחלקה (זמן)
public void setTime(int time)	מעדכן את הזמן שנשאר
public void stopTimer()	עצר את הטימר, בכך שמנגדיר את תוכנת המחלקה (boolean stopped) כ-אמת.



## AirplaneModeReceiver

תפקיד המחלקה הוא **ליידע את האפליקציה** במידה והמשתמש בחור לשום 'מצב טיסה' בזمان שהותם **באפליקציה**.

**יורש ממחלקה:**  
BroadcastReceiver -

תחום ההכרה של המחלקה:

- המשטנה במחלקה הינו ציבורו על מנת לאפשר גישה מכלל המחלקות הזקוקות לנตอน זה.

### מאפיינים

public static boolean isAirplaneMode

מייצג אם המשתמש על מצב טיסה או לא.  
במידה ולא נקבע עדרין ערך, יהיה null.

### פעולות

@Override  
public void onReceive(Context context,  
Intent intent)

פעולה המבצעת כאשר המשתמש הכניס  
את הטלפון למצב טיסה או ביטל אותו,  
הפעולה קובעת את מצב הטלפון ומעדכנת  
את הנתונים כדי שהאפליקציה יזיה תעבור  
 בהתאם



## TaskAdapter

תפקיד המחלקה הוא לחבר את הרשימת משימות (נתונים) למסך ( מבחינה עיצובית )

וירש ממחלקה:

- RecyclerView.Adapter<TaskAdapter.TaskViewHolder>

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם private על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
final RecyclerViewFunctionalities recyclerViewFunctionalities	מייצג אובייקט של ממש המאפשר Callback
Context context	מייצג קונטקט
List<Task> taskList	מייצג את רשיימת המשימות

פעולות	
public TaskAdapter(Context context, List<Task> taskList, RecyclerViewFunctionalities recyclerViewFunctionalities)	פעולה היוצרת אובייקט של המחלקה, מתחילה את האפשרות להשתמש באובייקט שנוצר
@NonNull @Override public TaskAdapter.TaskViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)	פעולה היוצרת את אובייקט העיצוב ודואגת לשיזף לו מקום על המסך .
@Override public void onBindViewHolder(@NonNull TaskAdapter.TaskViewHolder holder, int position)	פעולה המקשרת את נתונים המשימה לעיצוב .
@Override	פעולה המחזיר את מספר האיברים



public int getItemCount()	ברשימה
public List<Task> getTaskList()	פעולה המחזיר את הרשימה (תמונה מחלקה)



## TaskViewHolder

תפקיד המחלקה הוא לקשר את הפרס (נתונים) למסך ( מבחינה עיצובית )

ירוש ממחלקה:   
RecyclerView.ViewHolder -

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם `private` על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
TextView taskNameTv, difficultyTv, timeTv	אובייקטים המייצגים את נתונים המשימה שעל המסך .
ConstraintLayout constraintLayout	אובייקט המייצג מסך בתוך העיצוב, על מנת לשנות את צבע הקושי של המשימה.

פעולות	
public TaskViewHolder(@NonNull View itemView, RecyclerViewFunctionalities recyclerviewFunctionalities)	يُוצר أوببيكت شل المصلקה، ماتחל ات האפשרות להשתמש באובייקט הנוצר.
private void moveToTimerScreen(Task t)	פעולה המעבירת אל מסך הטימר בעת לחיצה על 'התחל משימה'.



## RewardAdapter

תפקיד המחלקה הוא לקשר את הרשימת פרסים (נתונים) למפרק ( מבחינה עיצובית)

יורש ממחלקה:

- RecyclerView.Adapter<RewardAdapter.RewardViewHolder>
- תחום ההכרה של המחלקה:
- כל המשתנים במחלקה הינם private על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
final RecyclerViewFunctionalities recyclerViewFunctionalities	מייצג אובייקט של ממש המאפשר Callback
Context context	מייצג קונטקסט
List<Reward> rewardList	מייצג את רשימת הפרסים

פעולות	
public RewardAdapter(Context context, List<Reward> rewardList, RecyclerViewFunctionalities recyclerViewFunctionalities)	פעולה היוצרת אובייקט של המחלקה, ማתחילה את האפשרות להשתמש באובייקט שנוצר
@NonNull @Override public RewardAdapter.RewardViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)	פעולה היוצרת את אובייקט העיצוב ודואגת לשים לו מקום על המסך.
@Override public void onBindViewHolder(@NonNull RewardAdapter.RewardViewHolder holder, int position)	פעולה המקשרת את נתונים הפרס לעיצוב.
@Override public int getItemCount()	פעולה המחזיר את מספר האיברים ברשימה



```
public List<Reward> getRewardList()
```

פועלה המחזיר את הרשימה (תכוונת  
מחלקה)



## RewardViewHolder

תפקיד המחלקה הוא לקשר את הפרס (נתונים) למפרק ( מבחינה עיצובית)

יורש ממחלקה:  
RecyclerView.ViewHolder -

תחום ההכרה של המחלקה:

- כל המשתנים במחלקה הינם **private** על כן שינוים יכולם להתבצע רק בתחוםי המחלקה.

מאפיינים	
TextView RewardNameTv, RewardDescTv, xpTv	אובייקטים המייצגים את הנתוני הפרס על המסך
AlertDialog.Builder alertDialog	אובייקט המייצג דיאלוג על מנת לאשר רכישת פרס

פעולות	
public RewardViewHolder(@NonNull View itemView, RecyclerViewFunctionalities recyclerViewFunctionalities)	يُוצר أوبี้كت شل المصلكة، ماتחל את האפשרות להשתמש באובייקט הנוצר.
private void moveToPaymentScreen(Reward r)	פעולה המבצעת רכישת פרס, כוללת הצגת דיאלוג ובמידה והמשתמש מאשר, מעדכנת את פרטיו



## Utils

תפקיד המחלקה הוא לשמר פעולות סטטיות שימושים בהם שימוש במקומות אחרים בפרויקט.

### תחום ההכרה של המחלקה:

- תחום ההכרה הינו מחלקות הפרויקט ומכל מחלקה יש אפשרות גישה לפעולות המחלקה.

### מאפיינים

אין

### פעולות

public static boolean isConnectedToInternet(Context context)	בודק אם המשתמש יכול לגלוש באינטרנט. מחזיר אמת אם כן, אחרת שקר.
public static boolean checkInterfaceValid(RecyclerViewFunctionalities recyclerViewFunctionalities, int adapterPos)	בודק אם הממשק תקין במידה וכן יחזיר אמת. אחרת שקר.



## מדריך למשתמש

בחלק זה, נסקרו את האפליקציה וنعוזר למשתמש המתחילה להצליח להתנהל כמו שצריך  
באפליקציה 😊.

עבור כל מסך בפרויקט אתה את:

- שם המסך.
- תפקיד המסך.
- הסבר על כל האובייקטים הנמצאים על המסך, ופעולותיהם.



## מסך התחברות והרשמה

כל משתמש באפליקציה, חדש/משתמש שעד עכשו עשה את פעילותו על מכשיר אחר. יש דרך להתחיל להשתמש באפליקציה כהכלמה.  
משתמש שכבר רשום, יתחבר לאפליקציה במסך התחברות.  
משתמש חדש, ירשם לאפליקציה במסך הרשמה.  
כל אחד מהמסכים אסביר קצת

מסך התחברות	
 <b>HELLO THERE, WELCOME BACK</b> Sign In to continue <div style="border: 2px solid red; padding: 5px; margin-top: 10px;"> <input style="width: 100%; height: 30px; border: none; background-color: #f0f0f0; font-size: 14px; padding-left: 10px; margin-bottom: 5px;" type="text"/> <span style="font-size: 14px;">Email</span> </div> <div style="border: 2px solid red; padding: 5px; margin-top: 10px;"> <input style="width: 100%; height: 30px; border: none; background-color: #f0f0f0; font-size: 14px; padding-left: 10px; margin-bottom: 5px;" type="password"/> <span style="font-size: 14px;">Password</span> <span style="float: right; font-size: 14px;">(Show)</span> </div> <div style="background-color: #4a86e8; color: white; text-align: center; width: 100%; height: 30px; line-height: 30px; border: none; border-radius: 5px; margin-top: 10px;"> <span style="font-size: 14px;">GO</span> </div> <hr style="margin-top: 10px;"/> <div style="text-align: center; margin-top: 10px;"> <span style="font-size: 14px;">OR</span> </div> <div style="border: 2px solid red; padding: 5px; margin-top: 10px;"> <span style="font-size: 14px;">Not signed up already? <a href="#" style="color: red;">Register</a></span> </div>	<p>מסך התחברות של האפליקציה מיועד לכל המשתמשים שנרשמו בעבר.</p> <p>בהתחברות הראשונית מהמכשיר המיועד לא יהיה צורך לעוד התחברות נוספת במכשיר זה והמשתמש לא מתנתק באופן ידני מהאפליקציה.</p> <p><b>חלקי המסך:</b></p> <ul style="list-style-type: none"> <li>- (1) בחלק זה המשתמש מזין את פרטיו האישיים כתוב.</li> <li>(איימיל וסיסמא)</li> <li>- (2) כאשר המשתמש סיים להזין את פרטיו, ילחץ על כפתור וועבר אל מסך הבית של האפליקציה.</li> <li>- (3) במידה ואין משתמש רשום באפליקציה, בלחיצה על ה(<b>Register</b>) יועבר למסך הרשמה.</li> </ul>



**מסך הרשמה**

**HELLO THERE,  
WELCOME**

Register to continue

(1)

(2)

(3)

Email

Password

Phone Number

**LET ME IN**

(4)

OR

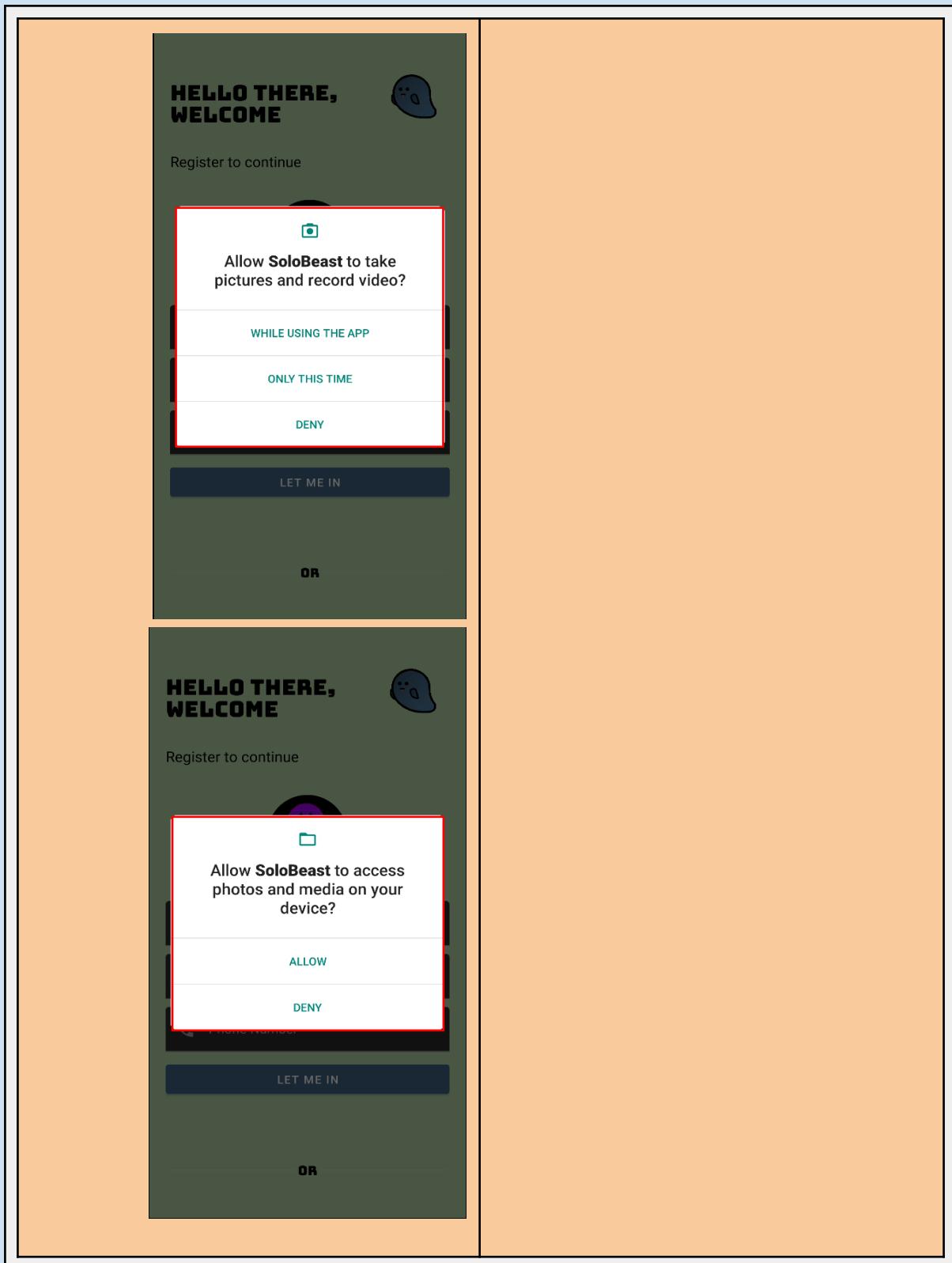
Already have a user? [Login](#) (5)

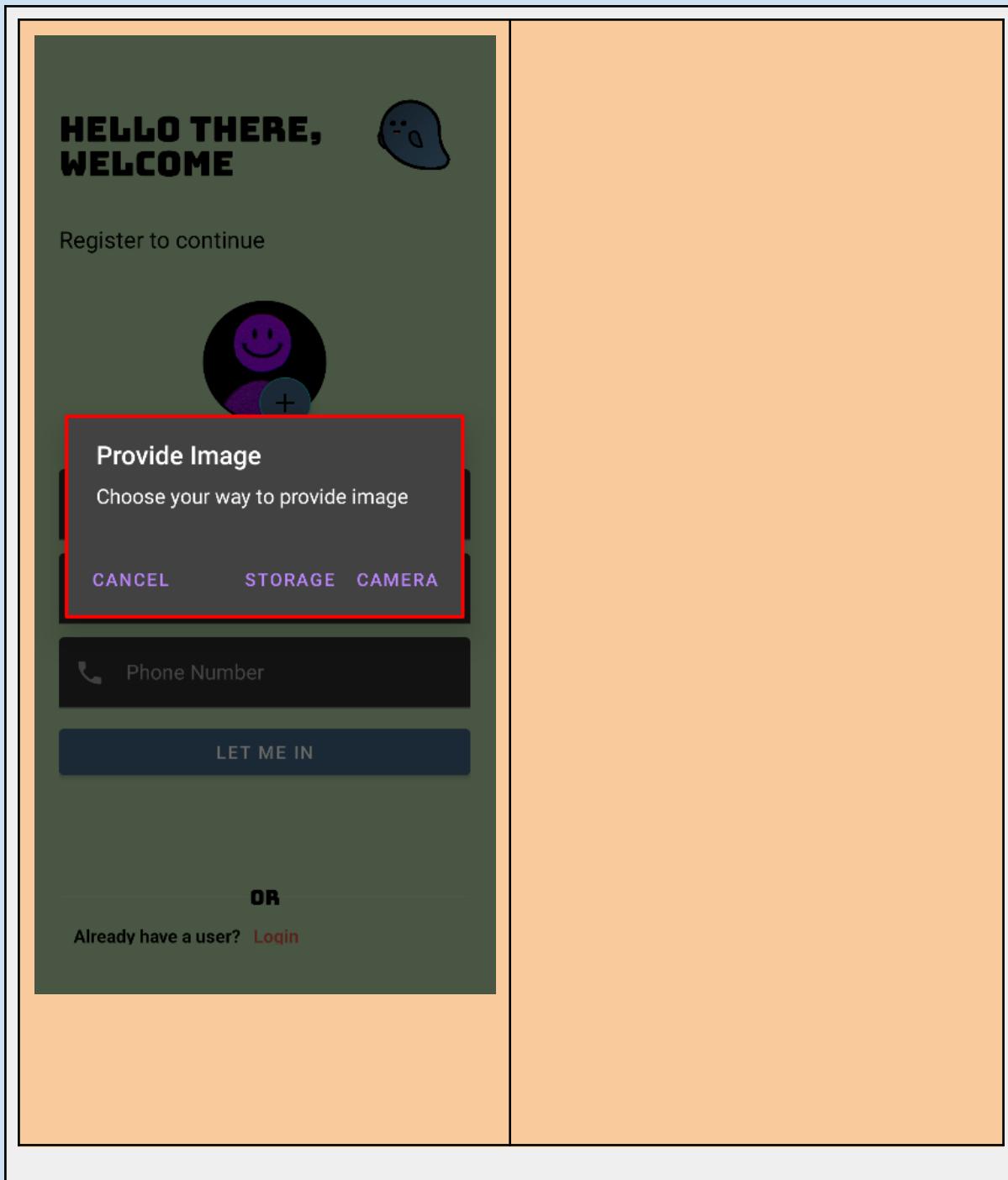
מסך ההרשמה של האפליקציה מיועד למשתמשים חדשים שרצו להשתמש באפליקציה, במסך זה ייצרו לעצמם משתמש.

בכונסה למסך, משתמש שהתקין את האפליקציה בפעם הראשונה קיבל דיאלוג שיצטרך לאשר על מנת לבחור/לצלם תמונה. נתינת הרשות לאפליקציה הון חלק בלתי נפרד מהאפליקציה. ראה צילומי מסך בהמשך.

חלקי המסך:

- (1) בחולק זה יהיה אפשר לראות את התמונה אשר המשתמש בחר כתמונה פרופיל.
- (2) אם המשתמש בוחר להוסיף תמונה פרופיל, ילחץ על ה(+) ויוצג דיאלוג המאפשר לו לבחור את שיטת ליקחת התמונה. ראה צילום מסך בהמשך.
- (3) בחולק זה המשתמש מזין את פרטיו האישיים כתוב.
- (4) כאשר המשתמש סיים להזין את אימייל וסיסמה וטלפון נייד
- (5) כאשר המשתמש ילחץ על כפתוריו יועבר אל מסך הבית של האפליקציה.
- (5) במידה והמשתמש רוצה להתחבר עם משתמש רשום באפליקציה, בלחיצה על ה(Login) יועבר למסך ההתחברות.

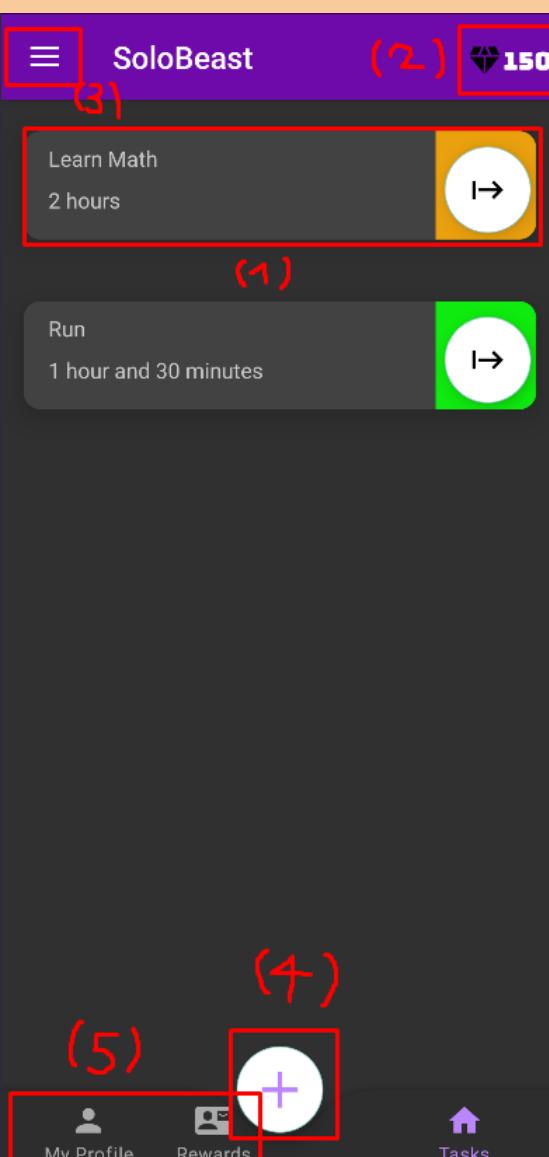






## מסך הבית (משימות)

מסך הבית הוא המסך אשר כל משתמש נכנס אליו כאשר יהיה מחובר לאפליקציה, באופן אוטומטי.  
במסך הבית יש אפשרות לצפות במשימות שהגדרת, לבצע אותן (יוסבר בהמשך מסך מעקב משימות)

מסך הבית	
 <p>The screenshot shows the SoloBeast app interface. At the top, there's a purple header bar with the SoloBeast logo and three dots. Below it, a task card for "Learn Math" is highlighted with a red box and labeled (1). The card shows "2 hours" and a yellow "Start" button. To its right, another task card for "Run" is shown with a green "Start" button and the text "1 hour and 30 minutes". At the bottom, there's a navigation bar with icons for "My Profile" (highlighted with a red box and labeled (5)), "Rewards" (highlighted with a red box and labeled (4)), and "Tasks" (labeled (7)).</p>	<p>זה המסך ברירת המחדל של האפליקציה.</p> <p>חלקי המסך:</p> <ul style="list-style-type: none"><li>(1) משימה שהמשתמש הגדר. (ירוק = קל, צהוב = בינוני, אדום = קשה)</li><li>(2) מד הנקודות, כרגע המשתמש צבר 150 נקודות.</li><li>(3) תפריט האפליקציה, ברוגע המשתמש צבר 150 נקודות.</li><li>(4) הוספת משימה. ייפתח מסך המיועד לכך <u>מסך הוספת משימה</u>.</li><li>(5) בליצצה על אחד מהכפתורים (Rewards / My Profile) המשתמש יעבור למסך שלחץ בהתאם.</li></ul>



The screenshot shows the SoloBeast mobile application interface. The top navigation bar is purple with the SoloBeast logo and a blue blob icon. The main content area has a light orange background.

**SoloBeast**

150

Settings

OnBoarding

Communicate

Contact Dev team

Donate

Authentication

Logout

Tasks

The interface includes a sidebar with various menu items and a central area for tasks.



## דיאלוגי הסבר



**YOU'RE NEW HERE.  
LET ME GUIDE YOU  
THROUGH THE  
PAGES OF THE APP,  
WHAT'S BEHIND  
ME IS THE FIRST  
AND THE MAIN  
SCREEN. YOU CAN  
ADD TASKS AT THE  
PLUS BUTTON AT  
THE BOTTOM OF  
THE SCREEN AND  
ALSO NAVIGATE TO  
ANOTHER SCREENS.**

I UNDERSTOOD

במהלך שהייה באפליקציה, בכניסה הראשונית של משתמש חדש לאפליקציה קיבל הסבירים על כל מסך שיכנס אליו בהתאם למסך (מצורפת תמונה דוגמה המוצגת במסך הבית)



## מסך הפרסים

המשתמש יוכל להכנס לכאן ולצפות ברכזונו בפרסים שהגדיר.  
ברגע שיגיע למספר נקודות מסוימים, יוכל לרכוש לעצמו את הפרס שהגדיר.

### הסביר מסך

חלקי המסך:

- (1) הפרס עצמו, בלחיצה על הריבוע תינוקן אופצייה לעריכת הפרס (ראה: [הוספת פרס](#))
- (2) רכישת ומימוש הפרס, בלחיצה יוצג דיאלוג קנייה (ראה תמונה)
- (3) פרטי הפרס (cotורת הפרס, תיאור והניקוד שצורך כדי לקנות אותו)
- (4) הוספת פרס בלחיצה



The screenshot shows the SoloBeast app interface. At the top, there's a purple header bar with the app name "SoloBeast" and a "150" badge. Below the header, there are two reward cards:

- Get starbucks with a friend**
- Get coffee and a cookie**

Both cards mention "XP points 100" and feature a green "Buy" button with a credit card icon.

A central modal dialog is displayed, asking for confirmation to proceed with a reward payment. The dialog has a red border and contains the following text:  
"Buy a reward"  
"Are you sure you want to proceed  
with this reward payment?"  
With "CANCEL" and "YES" buttons at the bottom.

At the bottom of the screen, there are three navigation icons: "My Profile" (person icon), "Rewards" (person with plus icon), and "Tasks" (house icon).



## בסיס נתונים

פרק זה מחלק ל 2 תתי פרקים:

Shared Preferences - (שמירת נתונים לوكאלית)

Firebase - (שמירת נתונים בענן)



## Shared preferences

בפרויקט יש שני שימושים חשובים לשימרת נתונים באופן לוקאלי

- קבלת משתמשים חדשים לאפליקציה עם הסברים, לנוחות המשתמש (ActivityGuideTracker)
- מסך הגדרות אישיות



## קבלת משתמשים חדשים

כאשר משתמש חדש נכנס לאפליקציה, הוא מגיע ישירות לדף ההתחברות - מכיוון שאין לו משתמש הוא יכנס לדף ההרשמה, כשיגיע לשם במידה ועל זיכרונו המכשיר של האפליקציה היו נתונים של משתמש אחר (איזה מסכים הוא ביקר) הם יאופסו.

XML File for saving the data example

```
<map>
    <boolean name="DetailedTaskAct" value="true" />
    <boolean name="MainActivity" value="true" />
</map>
```

כאשר הערך הוא "true" זה אומר שההסבר במסך נצפה.



## מיסק הגדירות אישיות

ההגדרות אשר נשמרו במסמך ההגדירות מפורטות להלן:

- ערכות נושא בהירה/כהה.**
- אפשרות להפעלת התראות יומיות.** (אפשרות לבחירת שעה רצוייה)
- אפשרות להפעיל/לכבות קבלת הסבר על מסכים למשתמש**



## Firebase

עם שירות של Firebase אני משתמש במהלך כל הפרויקט  
במערכות להלן:

- Firebase Authentication (חיבור והרשמה של משתמשים)
- Firebase Realtime Database (שמירת נתונים על משתמשים באופן דינامي בזמן אמת)
- Firebase Storage (שמירת תמונות פרופיל של משתמשים)



## Authentication

מהרցע שהמשתמש נרשם לאפליקציה, עם אימייל וסיסמה (שנשמרים ב Authentication ) הוא מקבל **Uid** (מספר זהה) שמאפשר לי להתנהל אליו עם עובודה במערכות האחירות בפיירבייס דוגמה למשתמש הנרשם בעזרת המערכת:

Identifier	Providers	Created	Signed In	User UID	⋮
ofekalm214@gmail.com	✉	Apr 30, 2023	May 13, 2023	jlgNegCr9RcFKeZxyH8ksOeCYaB3	⋮

אפשר לראות מתי המשתמש נוצר, מה הפעם האחרון שהתחבר בהצלחה לאפליקציה והנתון השכיח ביותר - מספר הזיהוי שלו (המורכב מ 28 אותיות ומספרים, פיירבייס משתמשים בטכניות קריפטוגרפיה על מנת לאבטח את נתוני המשתמש).

סביר קצת על טכניקת הגיבוב שיש אצל פרויקט, לטכנית קוראים SCRYPT והיא תוכנה לשימוש במערכות גודלות כמו Firebase ככח שניסיון הפיצוח של הסיסמה ייקח משאבי זיכרון רבים וזמן, השיטה חוזרת על עצמה 8 פעמים ככח שהסיכון לפצח הוא קטן בהרבה מאשר חזרה חד פעמית. בתמונה מטה אפשר לראות את הגדרות השיטה לפי פיירבייס.

Password hash parameters	
<p>These parameters are sensitive information to the user account security. Be sure to keep them private.</p>	
The information below can be used to migrate password users.	
hash_config { algorithm: SCRYPT, base64_signer_key: rVtQv9oEx05hLkKkuoAucNpJwsCYA9EnmRiw9mQGkCRs55ziNx8Z0FfuNxAnA0YG6FaUdETPGkArj7Lif1Xew==, base64_salt_separator: Bw==, rounds: 8, mem_cost: 14, }	⋮



## Realtime Database

Database schema

לצורך ההסבר, השארתי שני משתמשים להציגים עליהם את המבנה של הנתונים המשתמשים בזمن אמת

אפשר לראות את המספר המזהה של המשתמש שראינו קודם מוקודם ב Authentication (ljgNegCr9RcFKeZxyH8ksOeCYaB3)

```
graph TD; Root[ ]; Root --> Users[Users]; Users --> User1[ljgNegCr9RcFKeZxyH8ksOeCYaB3]; Users --> User2[n1CAj5ndLhS1EdaD7XV8rPiXqM73]
```

כל הסתעפות מענף "Users" מהוות משתמש אחר.

פירוט מידע על משתמש

```
graph TD; Root[ ]; Root --> User1[1jgNegCr9RcFKeZxyH8ksOeCYaB3]; User1 --> Rewards[Rewards]; User1 --> Tasks[Tasks]; User1 --> currentXP[currentXP: 18]; User1 --> phoneNumber[phoneNumber: "0549981177"]
```

לכל משתמש בסיס נתונים הדינامي נשמרים:

- הnickod שלו
- מס' טלפון
- רשימה של שירותי
- רשימה של פרסים



פירוט על משימה	
<pre>Users   -- 1jgNegCr9RcFKeZxyH8ks0eCYaB3     -- Rewards     -- Tasks       -- -NUh07-GwquCfrUGkjFX       -- -NU10zpJKfF8v50YNZVg     currentXP: 18     phoneNumber: "0549981177"</pre>	בתמונה אפשר לראות שלמשתמש יש שתי משימות כרגע שהגדרן
<pre>-NUh07-GwquCfrUGkjFX   description: "learn bubble sorting"   difficulty: "EASY"   key: "-NUh07-GwquCfrUGkjFX"   name: "learn cs"   time: "01:01"</pre>	כל משימה שנשמרת מקבלת מספר זהה בגודל קבוע של 20 תווים והוא מיוצר על ידי פירוביס בשימוש בפעולה push כך נראה ה-schema של משימה:  - שם - זמן (HH:MM) - תיאור - קורי - מספר זהוי זהה למספר זהוי של המשימה, על מנת לבצע עדכון משימה



פירוט על משימה	
<pre>Users   1jgNegCr9RcFKeZxyH8ks0eCYaB3     Rewards       -NUIJ30BuV9GUh5vsf1D       Tasks       currentXP: 18       phoneNumber: "0549981177"</pre>	בתמונה אפשר לראות של משתמש יש פרס אחד ברגע שהגדר
<pre>-NUIJ30BuV9GUh5vsf1D   description: "Buy stuff"   key: "-NUIJ30BuV9GUh5vsf1D"   rewardName: "Shopping"   xpAmount: 31</pre>	כל פרס שנשמר מקבל מספר זהה בגודל קבוע של 20 תווים והוא מיוצר על ידי פיריביז push בשימוש בפעולה : כך נראה ה-schema של פרס: <ul style="list-style-type: none"><li>- שם</li><li>- תיאור</li><li>- נקודות לקנייה</li><li>- מספר זהוי זהה למספר זהוי של הפרס, על מנת לבצע עדכון לפרס</li></ul>



## Storage

### Storage structure

לצורך שמירת התמונות של כל המשתמשים, יצרתי בעזרת מערכת Storage תיקייה בשם "profile-images"

profile-images/

Folder

### פירוט מידע על תמונות של המשתמשים

Name	Size	Type	Last modified
dBbmCI2CQVd9Lh19ZjepQGkEvvth2	7.23 KB	image/jpeg	May 12, 2023
u22PMsrznOAXBe92lqEQB0Mqxh1	12.3 KB	image/jpeg	May 12, 2023

כפי שנאמר בחלק של Authentication לכל משתמש יש מספר מזזה, ובאופן זהה נשמרות התמונות. כל תמונה נקראת בשם של המספר המזזה של המשתמש כדי לא לאייצר התנגשויות.

התמונה נשמרת כJPEG ובアイcot יחסית נמוכה על מנת להבטיח שהתמונה תעלה גם בתנאים פחות טוביים מהתנאי הפיתוח לדוגמה, טלפון עם רשות חלשה.

הורדת האיקות של התמונה עוזרת לזמן היעבוד של האפליקציה.

וגם להבטיח שהשרת קיבל את התמונה ויטפל בה כמו שצריך (במידה והריזולוציה גבוהה מדי התמונה לא תתקבל)



## רפלקציה / סיקום אישי

במהלך העבודה הרחבותי והעמוקתי את הידע שלי בנושאים הקשורים באנדרואיד (פיתוח אפליקציות) ואנדרואיד כמערכת הפעלה, כגון עבודה עם `multi-threaded` וכו'

הרעילון יהיה לי לאפליקציה נרkersם בשלב מוקדם מאוד והבנתי שזאת הולכת להיות האפליקציה שאבנה. אבל המעבר מדמיון למימוש הרגיש קצר כמו חלום בהתחלה, ישבתי עם דפי A4 וציירתי דמוי-MSCIM, איך המסד נתונים הולך להתנהל, איך מידע נשמר לכל משתמש ובאיilo מבנה נתונים המכיעים למטרה הסופית.

המטרה הראשונית הייתה השיג את מה שהייתי עושה על דף נייר (ניהול משימות ותגמול) ולהעביר את זה למשהו דיגיטלי כדי לפתור את הצורך האישי שלי. ובין הקשיים הראשוניים שהיו הם איך לעשות את האפליקציה רבת משתמשים ובעזרת לא מעט סקיצות הגעתו לסקימה של מסד הנתונים. וככה זה המשיך להתפתח, מشرطוט לשרטוט.

במבט לאחר, הידע של האפליקציה הושג. אבל היו לא מעט קשיים בדרך עליהם ארחים:

- הקשיים העיקריים שהורגש מבחןתי היו העובדה הקיימת הלא רציפה על הפרויקט, התחלתי אותו בתחלת אוגוסט וכל פעם הייתה מתקבל שוב מוטיבציה לימים אחדים מוסיף כמה שינויים ועוזב את זה לעוד שבועיים. כל פעם חזרתי לסייעת קוד גדול יותר ולשמור ולהציג אותה היה מאוד לא קל להשתלט כל פעם מחדש.
  - קשיים נוספים עברתי במהלך הפרויקט, כשהייתי יושב לעבוד על פיצ'ר חדש לרוב נתקלתי בלי כמעט בעיות.
- שעות של עבודה, תסכול, חוסר מצב רוח. להתפס - וחזר חלילה. ואז מגיע הרגע המיויחל של הצלחה, וואו! איך התרגשות, אדם שלא רגש לא יבין על מה אני מדבר
- surfet רגשות לטוב, הרגשת ניצחון

**סיפרתי **בнтегрии аисиим** על הפיצ'ר הכי קשה שמיימשתי, שהוא Service ניסיתי לפתור את הבעיה שהיה איתנו במשך 14 שעות, 12 שעות ברצף ושתים ביום שאחריו. זה היה רק אני ומהסך 2,073,600 פיקסלים שעמדו לרשותי. והרגשות שצפו בי לאחר מכן מזדיינים.**



על הקשיים שהיו התגברתי בכמה דרכי:

- ראייתי שהעבודה הלא רציפה מקשה עליי ועל המוטיבציה להמשיך להתעסך בפרויקט בקצב הדורש אז לקחתי פרקי זמן קצרים (3-4 ימים לכל היותר) ומשרין את כל היום לעבודה על הפרויקט.
- במידה ושמתי לב שלוקח לי יותר מדי זמן לשבת על אישתי בעיה, היית מנסה לשחרר אחרי כמה שעות. הפנמתי שלתת מנוחה ולהסתכל על זה יותר מאוחר יכול לסלול זווית ראייה אחרת על אותו קונספט, ולרוב זה עבד.

היו כמה אנשים במהלך העבודה שתרמו לי להתקדם:

- זיו יהלום, המורה ורכז המגמה. שהיה שם בשביili וביקש לקבל דוח מצב כל פרק זמן קצר, וסייע בהכוונה ובייעוץ.
- אבא שלי, שি�שב איתי לא מעט כדי לתת הרבה הארות/הערות ובאמת דחף אותי להצלחה והיה הכל אובייקטיבי שיש למרות הקربה המשפחתייה.
- חברים שלי, שישבטי איתם לעבוד על הפרויקט ביחד וננתנו אחד לשני מוטיבציה להתקדם. ואם לא עבדתי בזמן האחרון על הפרויקט הם הצליחו להזכיר לי שצורך להתקדם.

תובנות שהגעתי אליהן במהלך העבודה:

- במהלך העבודה למדתי על עצמי שאני נהנה מלחתעסך עם פיתוח צד שרת.
- נתקלתי בנושאים שלא הכרתי והיה כיף להרחב את ההבנה גם אם זה לא בהכרח קשור לנושא הספציפי שהתעסקתי בו באותו רגע.
- עבודה עם חברים יכולה להיות כיפית ומחייבת אבל צריך לדעת להפריד ולעבוד בלבד.

פיתוח עתידי המתוכנן להתבצע באפליקציה:

- הוספת תתי מישימות להשלמת משימה.
- לדוגמה: משימה - **ללמוד לבחן במתמטיקה (כללי מדי)** על מנת להתפרק יותר בתוכן ולתת יותר רצון להתחילה משימה. אף אחד לא רוצה ללמידה במשך 12 שעות ברצף בשבייל לקבל 100 בבחן. אבל תכונן וצעדים קטנים על מנת להגיע למטרה יותר ריאלי.
- אינטגרציה לספרטיפי בשבייל לחתת למשתמש לבחור פלייליסט שהוא רוצה לשם כשהוא עושה את המשימה, ככה יש לו את האפשרות להתפרק ללא הסחות דעת.
- הוספת ויזagitים למסך הבית המציגים את כמות הנקודות של המשתמש ואייזה פרסומים הציב לעצמו, כדי לחתת למשתמש תמרץ להכנס להשלים את משימותיו.



## ביבליוגרפיה

פליליסט שיצרתי במהלך העבודה של סרטונים מעניינים ו שימושיים שמצאתי

[https://www.youtube.com/@android\\_knowledge](https://www.youtube.com/@android_knowledge)

ערוץ היוטיוב - אנדרואיד לבגרות

של אסף זמיר appschool אתר

<https://www.youtube.com/@codinginflow>

<https://stackoverflow.com>

<https://www.geeksforgeeks.org>

<https://www.javatpoint.com/android-tutorial>

<https://developer.android.com/courses/fundamentals-training/overview-v2>



## נספחים

### Appendix A: Project code in Github

קוד הפרויקט מנוהל, מתועד ומתחזק דרך מערכת הגרסאות git אשר נמצאת באתר Github. מערכת ניהול הגרסאות מאפשרת שימרת גרסאות תוכן כדי ניהול ותחזוק הפרויקט בשלביו השונים.

ניתן לצפות ולהוריד את קוד הפרויקט בקישור הבא: [SoloBeast on Github](#)

תקציר המידע מפורט להלן:

The screenshot shows the GitHub README.md page for the 'AndroidSoloBeast' repository. The page has a dark theme. At the top, there's a file icon and the text 'readme.md'. Below that is the title 'AndroidSoloBeast'. Under the title, there's a 'Description' section with the text 'Task rewarding app for your assessments on day to day life'. Then there's a 'Table of Contents' section with a bulleted list of links: Technologies Used, Features, Screenshots, Setup, Usage, Project Status, Acknowledgements, and Contact. At the bottom, there's a 'Technologies Used' section with a bulleted list: XML&Java and Firebase.

readme.md

# AnroidSoloBeast

## Description

Task rewarding app for your assessments on day to day life

## Table of Contents

- [Technologies Used](#)
- [Features](#)
- [Screenshots](#)
- [Setup](#)
- [Usage](#)
- [Project Status](#)
- [Acknowledgements](#)
- [Contact](#)

## Technologies Used

- [XML&Java.](#)
- [Firebase.](#)



## Appendix B: Project implementation

בנספח זה ניתן לראות מבני הפרויקט

הפרויקט מורכב מ-21 מחלקות ו-3 ממשקים להלן:

Project Hierarchy	
Java Packages	
+---Adapters	
RecyclerViewFunctionalities.java	
RewardAdapter.java	
TaskAdapter.java	
+---Extras	
ActivityGuideTracker.java	
GuiderDialog.java	
PickerDialog.java	
SplashAct.java	
+---Receivers	
AirplaneModeReceiver.java	
\---Services	
CountdownThread.java	
TimerService.java	
+---Objects	
Reward.java	
Task.java	
User.java	
\---ui	
DetailedRewardAct.java	
DetailedTaskAct.java	
MailContactAct.java	
TaskTimerAct.java	
+---Auth	
LoginAct.java	
RegisterAct.java	
\---Home	
FirebaseCallback.java	
MainActivity.java	
\---Fragments	
HomeFragment.java	
ProfileFragment.java	
RewardFragment.java	



## XML

```
+---anim
|     splash_rainbow.xml
|
+---drawable
|     baseline_diamond_24.xml
|     baseline_help_24.xml
|     baseline_local_fire_department_24.xml
|     baseline_notification_important.xml
|     baseline_payment.xml
|     baseline_phone.xml
|     baseline_task_24.xml
|     baseline_timer.xml
|     baseline_timer_128.xml
|     ic_baseline_accessibility_new.xml
|     ic_baseline_add.xml
|     ic_baseline_contact_mail.xml
|     ic_baseline_edit.xml
|     ic_baseline_home.xml
|     ic_baseline_logout.xml
|     ic_baseline_menu.xml
|     ic_baseline_monetization_on.xml
|     ic_baseline_password.xml
|     ic_baseline_person.xml
|     ic_baseline_search.xml
|     ic_baseline_settings.xml
|     ic_baseline_start.xml
|     ic_baseline_update.xml
|     ic_launcher_background.xml
|     img.png
|     logo_splash_screen.png
|     profile_picture1.jpg
|     try_two.png
|
+---drawable-v24
|     ic_launcher_foreground.xml
|
+---font
|     bungee.ttf
|
+---layout
|     activity_detailed_reward.xml
|     activity_detailed_task.xml
|     activity_login.xml
|     activity_mail_contact.xml
|     activity_main.xml
|     activity_register.xml
|     activity_splash.xml
|     activity_task_timer.xml
|     custom_guider_dialog.xml
|     custom_picker_dialog.xml
|     fragment_home.xml
|     fragment_profile.xml
|     fragment_reward.xml
|     navigation_header.xml
|     reward_item.xml
|     task_item.xml
|
+---menu
|     bottom_bar_navigation.xml
|     navigation_menu.xml
|
+---values
|     colors.xml
|     ic_launcher_background.xml
|     strings.xml
|     styles.xml
|     themes.xml
|
+---values-night
|     themes.xml
```



## Code



## קבצי הפרויקט

**Objects**

**Task**



```
package com.example.solobeast.Objects;

import com.google.firebaseio.database.IgnoreExtraProperties;

import java.util.Arrays;
import java.util.List;

/**
Task class represents a task with a name, description, difficulty, and estimated time to
complete it.
This class is used in the context of a Firebase database.
@author Ofek Almog
*/
@IgnoreExtraProperties
//FOR FIREBASE\\
public class Task {

    /** The estimated time to complete the task, represented as a string. */
    private String time;

    /**The name of the task, represented as a string.*/
    private String name;

    /**The description of the task, represented as a string.*/
    private String description;

    /**The difficulty of the task, represented as a string.*/
    private String difficulty;

    /** key used for identification in the database. */
    private String key;

    /**
     Default constructor for Firebase.
     */
    public Task(){
        //DEFAULT FOR FIREBASE.\\
    }

    /**
     Constructor for Task class.
     @param name the name of the task
     @param time the estimated time to complete the task
     @param description the description of the task
     @param difficulty the difficulty level of the task
     @param key the key associated with the task in Firebase
}
```



```
/*
public Task(String name, String time, String description, String difficulty, String key) {
    if(timeIsValid(time)){ this.time = time;}else{this.time = "01:00";}
    if(nameIsValid(name)){ this.name = name;}else{this.name = "Default name";}
    if(descIsValid(description)){this.description = description;}else{this.description =
"Default description because none was provided";}
    if(diffIsValid(difficulty)){this.difficulty = difficulty;}else {this.difficulty =
"EASY";}
    this.key = key;
}

/***
Check if the difficulty level provided is valid.
@param difficulty the difficulty level
@return true if the difficulty is EASY, MEDIUM, or HARD; false otherwise
*/
private boolean diffIsValid(String difficulty) {
    if(difficulty.equals("EASY") || difficulty.equals("MEDIUM") || difficulty.equals("HARD"))
        return true;
    return false;
}

/***
Check if the task description provided is valid.
@param description the task description
@return true if the description is not empty; false otherwise
*/
private boolean descIsValid(String description) {
    if(description.isEmpty()) return false;
    return true;
}

/***
Check if the task name provided is valid.
@param name the task name
@return true if the name is not empty; false otherwise
*/
private boolean nameIsValid(String name) {
    if(name.isEmpty()) return false;
    return true;
}

/***
Check if the estimated time provided is valid.
@param time the estimated time
@return true if the time is not empty; false otherwise
*/
```



```
/*
private boolean timeIsValid(String time) {
    if(time.isEmpty()) return false;
    return true;
}

/***
Get the color associated with the difficulty level.
@param diff the difficulty level
@return the color as a hexadecimal string
*/
public static String changeColorAsTaskDifficulty(String diff){
    String greenHex = "#0eeb0e";
    String orangeHex = "#eba10e";
    String redHex = "#db200b";

    switch (diff){
        case "EASY":
            return greenHex;
        case "MEDIUM":
            return orangeHex;
        case "HARD":
            return redHex;
    }
    return "";
}

/***
 * Converts a time string in the format "hh:mm" into a human-readable representation of hours
and minutes.
 * @param hourAndMinute the time string in the format "hh:mm"
 * @return a human-readable representation of hours and minutes, such as "1 hour and 30
minutes"
*/
public static String timeRepresentation(String hourAndMinute) {
    String[] parts = hourAndMinute.split(":");
    int hours = Integer.parseInt(parts[0]);
    int minutes = Integer.parseInt(parts[1]);

    String hourString = hours == 1 ? "hour" : "hours";
    String minuteString = minutes == 1 ? "minute" : "minutes";

    if (hours == 0 && minutes == 1) {
        return "1 minute";
    } else if (hours == 0) {
        return minutes + " " + minuteString;
    }
}
```



```
        } else if (minutes == 0) {
            return hours + " " + hourString;
        } else {
            return hours + " " + hourString + " and " + minutes + " " + minuteString;
        }
    }

/**
 * Returns the name of the task.
 * @return the name of the task.
 */
public String getName() {
    return this.name;
}

/**
 * Sets the name of the task.
 * @param name the name of the task.
 */
public void setName(String name) {
    this.name = name;
}

/**
 * Sets the time required for the task.
 * @param time the time required for the task.
 */
public void setTime(String time) {
    this.time = time;
}

/**
 * Returns the time required for the task.
 * @return the time required for the task.
 */
public String getTime() {
    return this.time;
}

/**
 * Returns the difficulty level of the task.
 * @return the difficulty level of the task.
 */
public String getDifficulty() {
    return this.difficulty;
}
```



```
/**  
 * Returns the description of the task.  
 * @return the description of the task.  
 */  
public String getDescription() {  
    return this.description;  
}  
  
/**  
 * Returns the key of the task.  
 * @return the key of the task.  
 */  
public String getKey() {  
    return this.key;  
}  
  
/**  
 * Sets the description of the task.  
 * @param description the description of the task.  
 */  
public void setDesc(String description) {  
    this.description = description;  
}  
  
/**  
 * Sets the difficulty level of the task.  
 * @param diff the difficulty level of the task.  
 */  
public void setDiff(String diff) {  
    this.difficulty = diff;  
}  
  
/**  
 * Sets the key of the task.  
 * @param key the key of the task.  
 */  
public void setKey(String key) {  
    this.key = key;  
}  
  
/**  
 * Returns a string representation of the task, including its time, name, description, and  
difficulty level.  
 * @return a string representation of the task.  
 */
```



```
    @Override
public String toString() {
    return "Task{" +
        "time='" + this.time + '\'' +
        ", name='" + this.name + '\'' +
        ", description='" + this.description + '\'' +
        ", difficulty='" + this.difficulty + '\'' +
        '}';
}
```



## Reward



```
package com.example.solobeast.Objects;

import com.google.firebaseio.database.IgnoreExtraProperties;

/**
Represents a reward that can be earned by a user.
@author Ofek Almog
*/
@IgnoreExtraProperties
public class Reward {

    /** amount of experience points earned by completing the reward */
    private int xpAmount;

    /** name of the reward */
    private String rewardName;

    /** description of the reward */
    private String description;

    /** key used for identification in the database */
    private String key;

    /**
     Default constructor for Firebase.
     */
    public Reward(){
        //DEFAULT FOR FIREBASE.\\
    }

    /**
     Constructor to create a new Reward object.
     @param name The name of the reward.
     @param desc The description of the reward.
     @param xp The amount of XP to be rewarded.
     @param key The key of the reward.
     */
    public Reward(String name, String desc ,int xp,String key){
        if(nameIsValid(name)) {this.rewardName = name;}else {this.rewardName = "Default name";}
        if(descIsValid(desc)){this.description = desc;}else {this.description = "Default
description cause none was provided";}
        if (xpIsValid(xp)){this.xpAmount = xp;}else{this.xpAmount = 1;}
        this.key = key;
    }

    /**

```



```
Checks if a reward description is valid.  
@param description description of the reward  
@return true if the description is not empty, false otherwise  
*/  
private boolean descIsValid(String description) {  
    if(description.isEmpty()) return false;  
    return true;  
}  
  
/**  
Checks if an experience point value is valid.  
@param xp amount of experience points earned by completing the reward  
@return true if the experience point value is not 0, false otherwise  
*/  
private boolean xpIsValid(int xp){  
    boolean a;  
    a = (xp != 0) ? true : false;  
    return a;  
}  
  
/**  
Checks if a reward name is valid.  
@param name name of the reward  
@return true if the name is not empty, false otherwise  
*/  
private boolean nameIsValid(String name) {  
    if(name.isEmpty()) return false;  
    return true;  
}  
  
/**  
Returns the name of the reward.  
@return name of the reward  
*/  
public String getRewardName() {  
    return this.rewardName;  
}  
  
/**  
Sets the name of the reward.  
@param name name of the reward  
*/  
public void setRewardName(String name) {  
    this.rewardName = name;  
}
```



```
/**  
 Sets the amount of experience points earned by completing the reward.  
 @param xpAmount amount of experience points earned by completing the reward  
 */  
public void setXP(int xpAmount){ this.xpAmount = xpAmount;}  
  
/**  
 Returns the amount of experience points earned by completing the reward.  
 @return amount of experience points earned by completing the reward  
 */  
public int getXP() {  
     return this.xpAmount;  
}  
  
/**  
 Sets the description of the reward.  
 @param description description of the reward  
 */  
public void setDescription(String description) {  
    this.description = description;  
}  
  
/**  
 Returns the description of the reward.  
 @return the description of the reward as a string.  
 */  
public String getDescription() {  
    return this.description;  
}  
  
/**  
 Returns the key of the reward.  
 @return the key of the reward as a string.  
 */  
public String getKey() {  
    return this.key;  
}  
  
/**  
 Sets the key of the reward.  
 @param key the new key to set for the reward.  
 */  
public void setKey(String key) {  
    this.key = key;  
}
```



```
/**  
 * Returns a string representation of the reward object.  
 * @return a string representation of the reward object.  
 */  
@Override  
public String toString() {  
    return "Reward{" +  
        ", name='" + this.rewardName + '\'' +  
        ", description='" + this.description + '\'' +  
        ", xp='" + this.xpAmount + '\'' +  
        '}';  
}  
}
```



## User



```
package com.example.solobeast.Objects;

/**
 * Represents a user
 * @author Ofek Almog
 */
public class User {

    /** The phone number of the user*/
    private String phoneNumber;

    /** The current XP amount of the user*/
    private int currentXP;

    /**
     * Default constructor for Firebase.
     */
    public User(){

    }

    /**
     * Constructor for creating a user with a given phone number.
     * @param phoneNum the phone number of the user
     */
    public User(String phoneNum) {
        this.phoneNumber = phoneNum;
        this.currentXP = 0;
    }

    /**
     * Returns the current XP of the user.
     * @return the current XP
     */
    public int getCurrentXP() {
        return currentXP;
    }

    /**
     * Sets the current XP of the user.
     * @param currentXP the current XP to set
     */
    public void setCurrentXP(int currentXP) {
        this.currentXP = currentXP;
    }
}
```



```
/**  
 * Sets the phone number of the user.  
 * @param phoneNumber the phone number to set  
 */  
public void setPhoneNumber(String phoneNumber) {  
    this.phoneNumber = phoneNumber;  
}  
  
/**  
 * Returns the phone number of the user.  
 * @return the phone number  
 */  
public String getPhoneNumber() {  
    return phoneNumber;  
}  
  
/**  
 * Returns a string representation of the User object.  
 * @return a string representation of the User object.  
 */  
@Override  
public String toString() {  
    return "User{" +  
        "phoneNumber='" + this.phoneNumber + '\'' +  
        ", currentXP=" + this.currentXP +  
        '}';  
}  
}
```



**ui**

**ui.Auth**

**RegisterAct**



```
package com.example.solobeast.ui.Auth;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.content.FileProvider;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.drawable.BitmapDrawable;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.text.InputFilter;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.example.solobeast.BuildConfig;
import com.example.solobeast.Extras.ActivityGuideTracker;
import com.example.solobeast.Objects.User;
import com.example.solobeast.R;
import com.example.solobeast.ui.Home.MainActivity;
import com.google.android.gms.tasks.Task;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.FirebaseNetworkException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
import com.google.firebase.auth.FirebaseAuthUserCollisionException;
import com.google.firebase.auth.FirebaseAuthWeakPasswordException;
import com.google.firebaseio.database.FirebaseDatabase;
import com.google.firebaseio.storage.FirebaseStorage;
import com.google.firebaseio.storage.StorageReference;

import java.io.ByteArrayOutputStream;
```



```
import             java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

import de.hdodenhof.circleimageview.CircleImageView;

/**
The RegisterAct class represents the activity that allows the user to register
for a new account. It provides UI elements to capture user inputs such as email,
password, phone number, and profile picture. It uses Firebase Authentication and
Firebase Storage to authenticate the user and store the profile picture.
The class includes methods to validate user inputs, select an image from the camera
or the device's gallery, and register the user in Firebase. It also includes methods
to handle the result of the image selection, including handling camera permissions,
creating an image file, and setting the profile picture ImageView.
This class extends the AppCompatActivity class and overrides its onCreate method
to initialize the UI elements and Firebase objects. It also implements other helper
methods to perform specific tasks such as registering for activity results, converting
a drawable to a bitmap, and resetting the activity guide tracker for new users.

@author Ofek Almog
*/
public class RegisterAct extends AppCompatActivity {

    /**
     Input fields for email, password, and phone number.
    */
    TextInputEditText editTextEmail, editTextPassword, editTextPhoneNum;

    /**
     The Firebase authentication object.
    */
    private FirebaseAuth mAuth;

    /**
     The Firebase storage reference object for storing the user's profile picture.
    */
    private StorageReference storageReference;

    /**
     The FAB button for taking a picture.
    */
    private FloatingActionButton takePicBtn;
```



```
/**  
 * The ImageView for the profile picture.  
 */  
private CircleImageView circleImageView;  
  
/**  
 * The bitmap object for the profile picture.  
 */  
private Bitmap imageBitmap;  
  
/**  
 * The alert dialog for selecting an image source.  
 */  
private AlertDialog.Builder alertDialog;  
  
/**  
 * The URI for the image file.  
 */  
private Uri imageUri;  
  
/**  
 * The activity result launcher for taking a picture.  
 */  
private ActivityResultLauncher<Uri> takePictureLauncher;  
  
/**  
 * The activity result launcher for selecting an image from the gallery.  
 */  
private ActivityResultLauncher<String> pickImageLauncher;  
  
private final String TAG = "AuthData"; //Used for logging.  
  
/**  
 * Initializes the UI elements and Firebase objects.  
 * Also initializes camera and local storage methods.  
 */  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_register);  
  
    // Reset activity guide tracker for new users  
    ActivityGuideTracker agt = new ActivityGuideTracker(this);  
    agt.clearActivitiesStatus();
```



```
//FirebaseAuth object
mAuth = FirebaseAuth.getInstance();

init();

askForUserPermissions();

}

/***
Initializes the input fields, FAB button, ImageView, and alert dialog.

Also sets the default profile picture and sets a listener for the "move to login" text.
*/
private void init(){
    Drawable d = getDrawable(R.drawable.profile_picture1);
    imageBitmap = drawableToBitmap(d);

    editTextEmail = findViewById(R.id.editTextTextEmailAddress_Register);
    editTextPassword = findViewById(R.id.editTextTextPassword_Register);
    circleImageView = findViewById(R.id.profile_image);
    takePicBtn = findViewById(R.id.profile_pic_fab);

    editTextPhoneNum = findViewById(R.id.editTextTextPhoneNum_Register);
    InputFilter[] lengthFilter = new InputFilter []{new InputFilter.LengthFilter(9)}; //Restrict to 10 characters, for israeli numbers only
    editTextPhoneNum.setFilters(lengthFilter);

    TextView moveToLogin = findViewById(R.id.move_screen);
    alertDialog = new AlertDialog.Builder(this);

    takePicBtn.setOnClickListener(view -> {
        //Show the user the selection dialog.
        alertDialog
            .setMessage("Choose your way to provide image")
            .setTitle("Provide Image")
            .setCancelable(false);

        alertDialog.setPositiveButton("Camera", (dialogInterface, i) -> {
            dialogInterface.cancel();
            showImagePickerFromCamera();
        });
        alertDialog.setNegativeButton("Storage", (dialogInterface, i) -> {
```



```
        dialogInterface.cancel();
        showImagePickerFromGallery();
    });
    alertDialog.setNeutralButton("Cancel", (dialogInterface, i) -> {
        dialogInterface.dismiss();
    });
    AlertDialog alert = alertDialog.create();
    alert.show();
});

moveToLogin.setOnClickListener(view -> startActivity(new Intent(RegisterAct.this,
LoginAct.class)));
}

/**
 * This method checks if all the registration credentials are valid.
 * It checks if the email and password fields are not empty and if the phone
 * number is exactly 10 digits long.
 *
 * @return true if all the credentials are valid, false otherwise.
 */
private boolean check_validation_credentials() {
    if(editTextEmail.getText().length() == 0){
        editTextEmail.setError("You haven't typed any credentials");
        editTextEmail.requestFocus();
        return false;
    }
    if(editTextPassword.getText().length() == 0){
        editTextPassword.setError("You haven't typed any credentials");
        editTextPassword.requestFocus();
        return false;
    }
    if (editTextPhoneNum.getText().length() != 9){
        editTextPhoneNum.setError("Phone number should be 9 digits");
        editTextPhoneNum.requestFocus();
        return false;
    }

    return true;
}

/**
 * This method shows a dialog to allow the user to choose to take a picture with the device's
camera
 * and starts an activity to capture the image. It uses the ActivityResultLauncher API to
handle
```



```
*           the result of the activity.  
*/  
private void showImagePickerFromCamera() {  
    try {  
        // Create an image file to store the camera photo  
        File photoFile = createImageFile();  
        if (photoFile != null) {  
            // Create a file URI to pass to the camera app  
            imageUri = FileProvider.getUriForFile(this, BuildConfig.APPLICATION_ID +  
".provider", photoFile);  
  
            // Launch the camera app  
            takePictureLauncher.launch(imageUri);  
        }  
    } catch (IOException ex) {  
        ex.printStackTrace();  
        Toast.makeText(this, "Error occurred while creating the photo file",  
Toast.LENGTH_SHORT).show();  
    }  
}  
  
/**  
 * This method shows a dialog to allow the user to choose an image from the device's gallery  
 * and starts an activity to pick the image. It uses the ActivityResultLauncher API to handle  
 * the result of the activity.  
 */  
private void showImagePickerFromGallery() {  
    pickImageLauncher.launch("image/*");  
}  
  
/**  
 Sets up the take picture and pick image launchers to handle the user taking a photo  
 with the camera or selecting an image from their device's gallery.  
 If the user takes a photo with the camera, the image is displayed in the CircleImageView and  
 stored as a bitmap.  
 If the user cancels or fails the camera operation, the gallery is opened for image  
 selection.  
 If the user selects an image from the gallery, the image is displayed in the CircleImageView  
 and  
 stored as a bitmap.  
 If the user cancels the image selection from gallery, the gallery is opened again for image  
 selection until the user chooses.  
 */  
private void RegisterOnResult_and_takePicFromCamera(){  
    try {
```



```
        takePictureLauncher = registerForActivityResult(new
ActivityResultContracts.TakePicture(), result -> {
    if (result) {
        // Camera photo was taken successfully
        circleImageView.setImageURI(imageUri);
        imageBitmap = BitmapFactory.decodeFile(imageUri.getPath());
    } else {
        // Camera operation was canceled or failed
        alertDialog.show();
        //let user re-select
    }
});
}catch (IllegalStateException e) {
    // Handle the exception
    e.printStackTrace(); // or use a logging mechanism
    // Perform any necessary cleanup or error handling
}

try {
    pickImageLauncher = registerForActivityResult(new
ActivityResultContracts.GetContent(), result -> {
        if (result != null) {
            // Image was picked successfully from gallery
            circleImageView.setImageURI(result);
            try {
                imageBitmap =
BitmapFactory.decodeStream(getContentResolver().openInputStream(result));
            } catch (FileNotFoundException e) {
                throw new RuntimeException(e);
            }
        } else {
            // Image picking was canceled
            alertDialog.show();
            //let user re-select
        }
    });
}catch (IllegalStateException e) {
    // Handle the exception
    e.printStackTrace(); // or use a logging mechanism
    // Perform any necessary cleanup or error handling
}

}
/**
```



```
*           This method converts a Drawable object to a Bitmap object.  
*  
* @param drawable the Drawable object to convert.  
* @return a Bitmap object representing the converted Drawable object.  
*/  
public static Bitmap drawableToBitmap(Drawable drawable) {  
    if (drawable instanceof BitmapDrawable) {  
        return ((BitmapDrawable) drawable).getBitmap();  
    }  
  
    Bitmap bitmap = Bitmap.createBitmap(drawable.getIntrinsicWidth(),  
drawable.getIntrinsicHeight(), Bitmap.Config.ARGB_8888);  
    Canvas canvas = new Canvas(bitmap);  
    drawable.setBounds(0, 0, canvas.getWidth(), canvas.getHeight());  
    drawable.draw(canvas);  
  
    return bitmap;  
}  
  
/**  
 * This method creates a unique image file name based on the current timestamp.  
 *  
 * @return a File object representing the created image file.  
 * @throws IOException if the file creation fails.  
 */  
private File createImageFile() throws IOException {  
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());  
    String imageFileName = "JPEG_" + timeStamp + "_";  
    File storageDir = getExternalFilesDir(Environment.DIRECTORY_PICTURES);  
    return File.createTempFile(imageFileName, ".jpg", storageDir);  
}  
  
/**
```

Registers a new user with Firebase Authentication and adds the user's information to the Firebase Realtime Database.

Validates the input credentials entered by the user, and displays an error message for any invalid input.

If registration is successful, the user's profile picture is uploaded to Firebase Storage, and the user is taken to the main screen of the app.

```
@param view The view that triggers the registration process, typically a button.  
*/  
public void registerNewUser(View view) {
```



```
String userPhone = editTextPhoneNum.getText().toString();
User my_user = new User(
    userPhone
);
String userEmail = editTextEmail.getText().toString();
String userPassword = editTextPassword.getText().toString();
if(check_validation_credentials()){
    mAuth.createUserWithEmailAndPassword(userEmail,userPassword)
        .addOnCompleteListener(RegisterAct.this, task -> {
            if (task.isSuccessful()) {
                Log.d(TAG, "createUserWithEmail:success");

                FirebaseDatabase.getInstance().getReference("Users")
                    .child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                    .setValue(my_user).addOnCompleteListener(task1 -> {
                        if (task1.isSuccessful()) {
                            Log.d(TAG, "AddToDatabase:success");
                            Toast.makeText(getApplicationContext(), "YES",
Toast.LENGTH_LONG).show();
                        } else {
                            Log.d(TAG, "AddToDatabase:failure");
                            Toast.makeText(getApplicationContext(), "NO",
Toast.LENGTH_LONG).show();
                        }
                    });
            }

            uploadImageToFirebaseStorage(imageBitmap);

            updateUI();
        } else {
            // If sign in fails, display a message to the user.
            Log.w(TAG, "createUserWithEmail:failure", task.getException());

            try {
                throw task.getException();
            } catch(FirebaseAuthWeakPasswordException e) {
                editTextPassword.setError("Password is too weak");
                editTextPassword.requestFocus();
            } catch(FirebaseAuthInvalidCredentialsException e) {
                editTextEmail.setError("Email is invalid");
                editTextEmail.requestFocus();
            }catch(FirebaseAuthUserCollisionException e) {
                editTextEmail.setError("This email is in use");
                editTextEmail.requestFocus();
            }catch (FirebaseNetworkException e){
                Toast.makeText(getApplicationContext(),"Please Check Your

```



```
        internet connection",Toast.LENGTH_LONG).show();
    }
    catch(Exception e) {
        Toast.makeText(getApplicationContext(),"Some of the fields are
not valid",Toast.LENGTH_LONG).show();
    }
}
});
```

}

/\*

Uploads the given bitmap image to Firebase Storage, compressing it and converting it to bytes first.

The image is saved in the "profile-images" folder with the filename as the current user's ID.

A Toast message is displayed if the upload is successful.

```
@param bitmap The Bitmap image to upload to Firebase Storage
*/
private void uploadImageToFirebaseStorage(Bitmap bitmap){
    storageReference = FirebaseStorage.getInstance().getReference();
    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.JPEG, 5, stream);
    byte[] data = stream.toByteArray();
    //\\\
    StorageReference imageRef = storageReference.child("profile-images/" +
FirebaseAuth.getInstance().getCurrentUser().getUid());
    Task<Uri> urlTask = imageRef.putBytes(data).continueWithTask(task -> {
        if (!task.isSuccessful()) {
            throw task.getException();
        }
        if (task.isSuccessful()) {
            Toast.makeText(this,"Image uploaded to database",Toast.LENGTH_LONG).show();
        } else {
            // Handle failures
            // ...
        }
        return imageRef.getDownloadUrl();
    });
}
```



```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == 100) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED
                && grantResults[1] == PackageManager.PERMISSION_GRANTED
                && grantResults[2] == PackageManager.PERMISSION_GRANTED) {
                // Permissions are granted
                // Your code logic here
                RegisterOnResult_and_takePicFromCamera();
            } else {
                // Permissions are denied
                // Handle the denied permission case
                Toast.makeText(this,"Please accept the camera and storage permission",
Toast.LENGTH_LONG).show();
            }
        }
    }

    public void askForUserPermissions(){
        if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED
            || ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED
            || ContextCompat.checkSelfPermission(this,
Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
            // Permissions are not granted, request them
            ActivityCompat.requestPermissions(this, new String[]{
                Manifest.permission.CAMERA,
                Manifest.permission.WRITE_EXTERNAL_STORAGE,
                Manifest.permission.READ_EXTERNAL_STORAGE
            }, 100);
        } else {
            // Permissions are already granted

            // Register activity result launchers and take a picture from camera
            RegisterOnResult_and_takePicFromCamera();
        }
    }

    /**
     * Redirects the user to the main activity screen when successful registration occurs.
     */
    private void updateUI() {
```



```
        Intent i = new Intent(this, MainActivity.class);
startActivity(i);
}

}
```



**ui**

**ui.Auth**

**RegisterAct : XML**



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.Auth.RegisterAct"
    android:orientation="vertical"
    android:background="@color/light_green"
    android:padding="20dp"
    >

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="20dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <ImageView
        android:id="@+id/logo_image"
        android:layout_width="88dp"
        android:layout_height="75dp"

        android:src="@drawable/try_two_png"
        android:transitionName="logo_image"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/logo_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:fontFamily="@font/bungee"
        android:text="Hello there, Welcome"
        android:textColor="#000"
        android:textSize="27sp"
        android:transitionName="logo_text"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
```



```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/slogan_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:text="Register to continue"
    android:textSize="18sp"
    android:textColor="@color/black"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logo_name" />

<de.hdodenhof.circleimageview.CircleImageView
    android:id="@+id/profile_image"
    android:layout_width="96dp"
    android:layout_height="96dp"
    android:layout_marginStart="112dp"
    android:layout_marginTop="36dp"
    android:layout_marginEnd="112dp"
    android:src="@drawable/profile_picture1"
    app:civ_border_color="#FF000000"
    app:civ_border_width="2dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.49"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/slogan_name" />

<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/profile_pic_fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="56dp"
    android:layout_marginEnd="8dp"
    android:backgroundTint="@color/grey_blue"
    android:src="@drawable/ic_baseline_add"

    app:fabSize="mini"
    app:layout_constraintEnd_toEndOf="@+id/profile_image"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="@+id/profile_image"
    app:layout_constraintTop_toTopOf="@+id/profile_image" />

<LinearLayout
    android:id="@+id/linearLayout"
```



```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="204dp"
        android:orientation="vertical"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/logo_name">

    <com.google.android.material.textfield.TextInputLayout
        style="Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        app:startIconDrawable="@drawable/ic_baseline_contact_mail"
        android:layout_height="match_parent"
        android:hint="Email">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/editTextTextEmailAddress_Register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="48dp" />
</com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        style="Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        app:startIconDrawable="@drawable/ic_baseline_password"
        android:layout_marginTop="10dp"
        app:passwordToggleEnabled="true">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/editTextTextPassword_Register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:minHeight="48dp" />
</com.google.android.material.textfield.TextInputLayout>

    <com.google.android.material.textfield.TextInputLayout
        style="Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:startIconDrawable="@drawable/baseline_phone"
        app:prefixText="+972"
        android:hint="Phone Number"
```



```
        android:layout_marginTop="10dp"
    >

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/editTextTextPhoneNum_Register"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="phone"
        android:minHeight="48dp" />
</com.google.android.material.textfield.TextInputLayout>

<Button
    android:id="@+id/submit_form"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="5dp"
    android:backgroundTint="@color/grey_blue"
    android:onClick="registerNewUser"
    android:text="LET ME IN"
    android:textColor="#fff" />

</LinearLayout>

<LinearLayout
    android:id="@+id/or_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="64dp"
    android:gravity="center"
    android:orientation="horizontal"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/linearLayout">

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:layout_margin="2dp"
        android:layout_weight="1"
        android:background="?android:attr/listDivider"

        android:backgroundTint="#FFFFFF" />

    <TextView
        android:layout_width="wrap_content"
```



```
        android:layout_height="wrap_content"
        android:fontFamily="@font/bungee"
        android:text="or"
        android:textColor="@color/black"
        android:textSize="16sp" />

<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_margin="2dp"
    android:layout_weight="1"
    android:background="?android:attr/listDivider"
    android:backgroundTint="#FFFFFF" />

</LinearLayout>

<TextView
    android:id="@+id/move_screen"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/go_to_login"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.078"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/or_view" />

</androidx.constraintlayout.widget.ConstraintLayout>

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



אופק אלמוג - SoloBeast



**ui**

**ui.Auth**

**LoginAct**



```
package com.example.solobeast.ui.Auth;

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

import com.example.solobeast.ui.MainActivity;
import com.example.solobeast.R;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.RuntimeExecutionException;
import com.google.android.material.textfield.TextInputEditText;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseNetworkException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthInvalidCredentialsException;
import com.google.firebase.auth.FirebaseAuthInvalidUserException;
import com.google.firebase.auth.FirebaseUser;

/**
 * This class handles the login process for the user. It provides a simple interface for the user
 * to input their login credentials, validates their input and attempts to sign in the user using
 * FirebaseAuth. If the sign-in is successful, the user is redirected to the MainActivity.
 * Otherwise,
 * appropriate error messages are displayed to the user based on the type of error encountered.
 * @author Ofek Almog
 */
public class LoginAct extends AppCompatActivity {

    /** Firebase Authentication instance */
    FirebaseAuth mAuth;

    /** EditText field for email */
    TextInputEditText editTextEmail;

    /** EditText field for password */
    TextInputEditText editTextpassword;
```



```
/**  
 * Initializes the activity layout and Firebase Authentication instance.  
 * Also, it calls the init method to initialize the layout views.  
 *  
 * @param savedInstanceState An instance of Bundle class to restore the activity to a  
 previous state if necessary.  
 */  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_login);  
  
    mAuth = FirebaseAuth.getInstance();  
  
    init();  
}  
  
/**  
 * Initializes the views and listeners for the activity.  
 */  
private void init(){  
    editTextEmail = (TextInputEditText) findViewById(R.id.editTextTextEmailAddress_Register);  
    editTextpassword = (TextInputEditText) findViewById(R.id.editTextTextPassword_Register);  
    TextView moveToRegister = findViewById(R.id.move_screen);  
  
    moveToRegister.setOnClickListener(view -> startActivity(new Intent(LoginAct.this,  
RegisterAct.class)));  
}  
  
/**  
 * Validates user input for email and password fields.  
 * @return boolean true if input is valid, false otherwise.  
 */  
private boolean check_validation_credentials() {  
    if(editTextEmail.getText().length() ==0){  
        editTextEmail.setError("You haven't typed any credentials");  
        editTextEmail.requestFocus();  
        return false;  
    }  
    if(editTextpassword.getText().length() ==0){  
        editTextpassword.setError("You haven't typed any credentials");  
        editTextpassword.requestFocus();  
        return false;  
    }  
    return true;  
}
```



```
}
```

```
/**  
 * Attempts to log the user in using the provided email and password.  
 * @param view the view that triggered the method.  
 */  
public void login(View view){  
    boolean validation_credentials_are_valid= check_validation_credentials();  
    if(validation_credentials_are_valid) {  
        Log.d("AuthData", "Username: " + editTextEmail.getText().toString() + "\n" +  
"Password: " + editTextpassword.getText().toString());  
  
        mAuth.signInWithEmailAndPassword(editTextEmail.getText().toString(),  
editTextpassword.getText().toString()).addOnCompleteListener(new  
OnCompleteListener<AuthResult>() {  
    @Override  
    public void onComplete(@NonNull Task<AuthResult> task) {  
        if (task.isSuccessful()) {  
            Log.d("AuthData", "signInWithCredential:success");  
            FirebaseUser user = mAuth.getCurrentUser();  
            updateUI(user);  
        } else {  
            Log.w("AuthData", "signInWithCredential:failure", task.getException());  
        }  
    }  
}).addOnFailureListener(e -> {  
    if(e instanceof FirebaseAuthInvalidUserException){  
        Toast.makeText(LoginAct.this, "This User Not Found , Create A New Account",  
Toast.LENGTH_SHORT).show();  
    }  
    if(e instanceof FirebaseAuthInvalidCredentialsException){  
        Toast.makeText(LoginAct.this, "Check your credentials",  
Toast.LENGTH_SHORT).show();  
    }  
    if(e instanceof FirebaseNetworkException){  
        Toast.makeText(getApplicationContext(), "Please Check Your internet  
connection", Toast.LENGTH_SHORT).show();  
    }  
});  
  
}  
}  
  
/**  
 * Moves the user to the main activity screen and passes the email as an extra parameter.  
 */
```



```
        @param email the email of the user
*/
private void move_screen(String email){
    Intent i = new Intent(LoginAct.this, MainActivity.class);
    String Identifier = "username_of_user";
    i.putExtra(email,Identifier);
    startActivity(i);
}

/**
 Called when the activity is starting. Checks if the user is signed in and updates the UI
accordingly.
*/
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.

    FirebaseAuth currentUser = mAuth.getCurrentUser();
    if(currentUser != null){
        updateUI(currentUser);
    }
}

/**
Updates the user interface after authentication.
@param currentUser the currently authenticated user
*/
private void updateUI(FirebaseUser currentUser) {

    Log.d("AuthData","Email is " + currentUser.getEmail());
    move_screen(currentUser.getEmail());
}
}
```



**ui**

**ui.Auth**

**LoginAct : 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"
    tools:context=".ui.Auth.LoginAct"
    android:orientation="vertical"
    android:background="@color/light_green"
    android:padding="20dp"
    >

<ImageView
    android:id="@+id/logo_image"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:transitionName="logo_image"
    android:layout_gravity="center"
    android:src="@drawable/try_two_png"/>
<TextView
    android:id="@+id/logo_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello there, Welcome Back"

    android:textSize="40sp"
    android:transitionName="logo_text"
    android:fontFamily="@font/bungee"
    android:textColor="#000"/>
<TextView
    android:textColor="#000"
    android:id="@+id/slogan_name"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Sign In to continue"
    android:textSize="18sp"/>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:layout_marginBottom="20dp"
    android:orientation="vertical">
    <com.google.android.material.textfield.TextInputLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:startIconDrawable="@drawable/ic_baseline_contact_mail"
```



```
        android:hint="Email"
        style="Widget.MaterialComponents.TextInputLayout.OutlinedBox">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:minHeight="48dp"
        android:id="@+id/editTextTextEmailAddress_Register"
        />
</com.google.android.material.textfield.TextInputLayout>
<com.google.android.material.textfield.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:startIconDrawable="@drawable/ic_baseline_password"
    android:hint="Password"
    app:passwordToggleEnabled="true"
    android:layout_marginTop="10dp"
    style="Widget.MaterialComponents.TextInputLayout.OutlinedBox">

    <com.google.android.material.textfield.TextInputEditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editTextTextPassword_Register"
        android:minHeight="48dp"
        android:inputType="textPassword" />
</com.google.android.material.textfield.TextInputLayout>
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="GO"
    android:id="@+id/submit_form"
    android:onClick="login"
    android:backgroundTint="@color/grey_blue"
    android:textColor="#ffffff"
    android:layout_marginTop="5dp"
    android:layout_marginBottom="5dp"/>

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center"
    android:orientation="horizontal">

    <View
```



```
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:layout_margin="2dp"
        android:layout_weight="1"
        android:background="?android:attr/listDivider"

        android:backgroundTint="#FFFFFF" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:fontFamily="@font/bungee"
    android:text="or"
    android:textColor="@color/black"
    android:textSize="16sp" />

<View
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_margin="2dp"
    android:layout_weight="1"
    android:background="?android:attr/listDivider"
    android:backgroundTint="#FFFFFF" />

</LinearLayout>
<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="@string/go_to_sign_up"
    android:layout_marginTop="10dp"
    android:id="@+id/move_screen"
    />
</LinearLayout>
```





**ui**

**ui.Home**

**MainActivity**



```
package com.example.solobeast.ui.Home;

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

import android.content.Intent;
import android.content.IntentFilter;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.widget.SearchView;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.example.solobeast.Extras.GuiderDialog;
import com.example.solobeast.Extras.Receivers.AirplaneModeReceiver;
import com.example.solobeast.Objects.User;
import com.example.solobeast.ui.DetailedRewardAct;
import com.example.solobeast.ui.DetailedTaskAct;
import com.example.solobeast.ui.Home.Fragments.CalendarFragment;
import com.example.solobeast.ui.Home.Fragments.HomeFragment;
import com.example.solobeast.ui.Home.Fragments.ProfileFragment;
import com.example.solobeast.R;
import com.example.solobeast.ui.Home.Fragments.RewardFragment;
import com.example.solobeast.databinding.ActivityMainBinding;
import com.example.solobeast.ui.Auth.LoginAct;
import com.example.solobeast.ui.Home.Fragments.SettingsFragment;
import com.example.solobeast.ui.MailContactAct;
import com.google.android.gms.tasks.Task;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
```



```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

/**
 * MainActivity is the main screen of the app, containing a navigation drawer with various
options,
 * a bottom navigation bar, and a floating action button for adding tasks or rewards based on the
fragments. It also
 * displays user XP, allows users to sign out, and initializes various fragments.
 * @author Ofek Almog
 */
public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener, FirebaseCallback{

    /** ActivityMainBinding is a generated class that provides easy access to all views in the
layout. */
    ActivityMainBinding binding;

    /** AirplaneModeReceiver is a broadcast receiver for detecting airplane mode changes. */
    AirplaneModeReceiver airplaneModeReceiver;

    /** current_user stores the current User object. */
    public static User current_user;

    /** drawerLayout is the layout for the navigation drawer. */
    DrawerLayout drawerLayout;

    /** navigationView is the navigation drawer itself. */
    NavigationView navigationView;

    /** addBtn is the floating action button for adding tasks or rewards. */
    FloatingActionButton addBtn;

    /** mAuth is the instance of the FirebaseAuth class for authentication. */
    FirebaseAuth mAuth;

    /** xpDisplayTv is the TextView that displays the user's XP. */
    TextView xpDisplayTv;

    /** Fragments is an enum representing the three possible fragments on the main screen. */
    enum Fragments{
        HOME,
        REWARDS,
        PROFILE,
        SETTINGS,
```



## CALENDAR

```
}
```

```
SearchView searchView;
```

```
/**
```

```
 * onCreate is called when the activity is starting. It initializes various components and
```

```
 * sets listeners for the navigation drawer, bottom navigation bar, and floating action
```

```
button.
```

```
*
```

```
 * @param savedInstanceState If the activity is being re-initialized after previously being
```

```
 * saved then this Bundle contains the data it most recently supplied in {@link
```

```
#onSaveInstanceState(Bundle)}.
```

```
*/
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    current_user = new User();
```

```
    getUserDetails(this);
```

```
    init();
```

```
    GuiderDialog guiderDialog = new GuiderDialog(this,"MainActivity","Hello there,\nYou're
```

```
new here. Let me guide you through the pages of the app, What's behind me is the first and the
```

```
main screen. You can add tasks at the plus button at the bottom of the screen and also navigate
```

```
to another screens.");
```

```
    guiderDialog.startDialog();
```

```
    airplaneModeReceiver = new AirplaneModeReceiver();
```

```
//        if(AirplaneModeReceiver.isAirplaneMode){
```

```
//
```

```
//    }
```

```
    addBtn.setOnClickListener(view -> {
```

```
        if(determineFragment() == Fragments.HOME){
```

```
            Intent i = new Intent(getApplicationContext(), DetailedTaskAct.class);
```

```
            i.putExtra("from_intent","Add");
```

```
            startActivity(i);
```

```
        }
```

```
        if(determineFragment() == Fragments.REWARDS){
```

```
            Intent i = new Intent(getApplicationContext(), DetailedRewardAct.class);
```

```
            i.putExtra("from_intent","Add");
```

```
            startActivity(i);
```

```
        }
```

```
        if (determineFragment() == Fragments.PROFILE){
```

```
            Toast.makeText(this,"",Toast.LENGTH_LONG).show();
```

```
        }
```

```
        if (determineFragment() == Fragments.SETTINGS){
```

```
            Toast.makeText(this,"",Toast.LENGTH_LONG).show();
```



```
        }

    });

mAuth = FirebaseAuth.getInstance();

binding.bottomNavigationView.setOnItemSelectedListener(item -> {
    switch (item.getItemId()) {
        case R.id.homescreen:
            replaceFragment(new HomeFragment(),Fragments.HOME);
            addBtn.setVisibility(View.VISIBLE);
            break;
        case R.id.profile:
            replaceFragment(new ProfileFragment(),Fragments.PROFILE);
            addBtn.setVisibility(View.GONE);
            break;
        case R.id.reward:
            replaceFragment(new RewardFragment(),Fragments.REWARDS);
            addBtn.setVisibility(View.VISIBLE);
            break;
        case R.id.calendar:
            replaceFragment(new CalendarFragment(),Fragments.CALENDAR);
    }
    return true;
});

navdrawer_init();
}

/**
 * Initializes the activity by inflating the layout, setting the content view,
 * replacing the fragment, setting up the bottom navigation view and setting the XP display.
 */
private void init(){
    binding = ActivityMainBinding.inflate(getLayoutInflater());
    setContentView(binding.getRoot());

    replaceFragment(new HomeFragment(),Fragments.HOME);
    binding.bottomNavigationView.setSelectedItemId(R.id.homescreen);
    binding.bottomNavigationView.setBackground(null);
    addBtn = (FloatingActionButton) binding.addButtonFab.findViewById(R.id.add_button_fab);
    xpDisplayTv = findViewById(R.id.xp_main_display);
    xpDisplayTv.setText("XP");
    binding.xpMainDisplay.setContentDescription("You have "+current_user.getCurrentXP()+"xp");
    //xpDisplayTv.setOnLongClickListener(this);
}
```



```
/***
 * Initializes the navigation drawer by setting up the toolbar, drawer layout, toggle,
 * and navigation view.
 */
private void navdrawer_init(){
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    drawerLayout = findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this, drawerLayout, toolbar,
        R.string.action_nav_open, R.string.action_nav_close);
    drawerLayout.addDrawerListener(toggle);
    toggle.syncState();

    navigationView = findViewById(R.id.navigation_view);
    navigationView.setNavigationItemSelectedListener(this);

}

/***
 * Overrides the default behavior of the back button press to close the navigation drawer if it
 * is open,
 * otherwise it calls the default behavior of the back button press.
 */
@Override
public void onBackPressed() {
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

/***
 * This method is called when a menu item in the Navigation Drawer is selected.
 * It handles the item selection based on the item ID and performs the corresponding action.
 * @param item The menu item that was selected in the Navigation Drawer
 */
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {
    switch(item.getItemId()){
        case R.id.nav_settings:
            Log.i("MoveScreen", "MoveToSettingsScreen:Success");
            //Intent moveToSettings = new Intent(this, SettingsActivity.class);
            //startActivity(moveToSettings);
    }
}
```



```
        replaceFragment(new SettingsFragment(), Fragments.SETTINGS);
        addBtn.setVisibility(View.GONE);
        //Toast.makeText(this,"Default settings is enabled in the current
version",Toast.LENGTH_LONG).show();
        break;
    case R.id.nav_onboarding:
        Log.i("MoveScreen", "MoveToOnBoardScreen:Success");
        Toast.makeText(this, "Welcome! every new user gets instruction in every screen ",Toast.LENGTH_SHORT).show();
        break;
    case R.id.nav_contact:
        Log.i("MoveScreen", "MoveToContact Screen:Success");
        Intent moveToMailContact = new Intent(this, MailContactAct.class);
        startActivity(moveToMailContact);
        break;
    case R.id.nav_donate:
        Log.i("MoveScreen", "MoveToDonateScreen:Success");
        Intent browserIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.paypal.com/donate/?hosted_button_id=G7LXD4YAGLL5Y"));
        startActivity(browserIntent);
        break;
    case R.id.nav_auth:
        Log.i("AuthData", "LogOut:Success");
        sign_out();
        startActivity(new Intent(this, LoginAct.class));
        break;
    }
    drawerLayout.closeDrawer(GravityCompat.START);
    return true;
}

/**
 Signs out the current user by calling the signOut() method of FirebaseAuth instance.
 */
public void sign_out() {
    mAuth.signOut();
}

/**
 Called when the activity is becoming visible to the user. This method checks if a user is
signed in
 and updates the UI accordingly. If the user is not signed in, it starts the LoginActivity.
@see LoginAct
*/
@Override
public void onStart() {
```



```
        super.onStart();
// Check if user is signed in (non-null) and update UI accordingly.

FirebaseUser currentUser = mAuth.getCurrentUser();
if(currentUser == null){
    startActivity(new Intent(this,LoginAct.class));
}
}

/**
 Replaces the current fragment in the FrameLayout container with the specified fragment
instance.
@param theInstance the fragment instance to replace the current fragment with
@param enumVersion an enumeration representing the type of fragment to be replaced
*/
private void replaceFragment(Fragment theInstance, Fragments enumVersion) {
    String onWhichFragment = "";
    int theFrag = enumVersion.ordinal();
    switch (theFrag){
        case 0:
            onWhichFragment = "HOME";
            break;
        case 1:
            onWhichFragment = "REWARDS";
            break;
        case 2:
            onWhichFragment = "PROFILE";
            break;
        case 3:
            onWhichFragment = "SETTINGS";
            break;
        case 4:
            onWhichFragment = "CALENDAR";
            break;
    }
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.frame_layout, theInstance, onWhichFragment);
    fragmentTransaction.commit();
}

/**
 Determines the current visible fragment by checking if the rewards or profile fragment is
visible,
if not, returns the HOME fragment.
@return An enum value of the currently visible fragment.
```



```
/*
public Fragments determineFragment(){
    Fragment rewardsFrag =
(Fragment) getSupportFragmentManager().findFragmentByTag("REWARDS");
    Fragment profileFrag =
(Fragment) getSupportFragmentManager().findFragmentByTag("PROFILE");
    Fragment settingsFrag =
(Fragment) getSupportFragmentManager().findFragmentByTag("SETTINGS");
    if(settingsFrag != null && settingsFrag.isVisible()){
        return Fragments.SETTINGS;
    }
    if(rewardsFrag != null && rewardsFrag.isVisible()){
        return Fragments.REWARDS;
    }
    else if(profileFrag != null && profileFrag.isVisible()){
        return Fragments.PROFILE;
    }
    return Fragments.HOME;
}

/**
 This method is called when the activity is resumed from a paused state.
 It registers an instance of AirplaneModeReceiver to receive broadcast intents for airplane
 mode changes.
 */
@Override
public void onResume() {
    super.onResume();
    registerReceiver(airplaneModeReceiver, new
IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED));
}

/**
 This method is called when the activity is going into the background and unregistering the
airplane mode broadcast receiver to avoid leaks
 or unnecessary processing while the activity is not visible.
 */
@Override
public void onPause() {
    super.onPause();
    unregisterReceiver(airplaneModeReceiver);
}

/**
 Retrieves user details from the Firebase Realtime Database using the Firebase Authentication
user ID
```



```
and           updates the {@link #current_user} object accordingly.  
 @param callback an interface that provides a callback method for when the user details have  
been received  
 */  
 public static void getUserDetails(FirebaseCallback callback){  
     DatabaseReference ref =  
FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getInstance().getUid());  
     Task<DataSnapshot> task = ref.get();  
     task.addOnSuccessListener(dataSnapshot -> {  
         User user_fb = dataSnapshot.getValue(User.class);  
         current_user.setCurrentXP(user_fb.getCurrentXP());  
         current_user.setPhoneNumber(user_fb.getPhoneNumber());  
         Log.d("AuthData","got all user details" + current_user.getCurrentXP());  
         callback.onUserDetailsReceived();  
     }).addOnFailureListener(e -> {  
         // Handle any errors here  
         Log.d("AuthData","The operation is not good\nCause: \n" +e);  
     });  
 }  
  
/**  
 * Updates the current XP value of the current user in Firebase database.  
 *  
 * @param xpToOperate the amount of XP to add or subtract from the current XP value.  
 * @param operation the operation to perform on the current XP value: "IncreaseVal" to add XP  
or any other string to subtract XP.  
 */  
 public static void updateXPtoUserFirebase(int xpToOperate, String operation) {  
     //set value in firebase  
     DatabaseReference ref =  
FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getInstance().getUid()).  
child("currentXP");  
  
     int current = MainActivity.current_user.getCurrentXP();  
     int CurrentValue;  
     if(operation.equals("IncreaseVal")){  
         CurrentValue = current + xpToOperate;  
     }  
     else {  
         CurrentValue = current - xpToOperate;  
     }  
  
     ref.setValue(CurrentValue);  
     MainActivity.current_user.setCurrentXP(CurrentValue);  
 }
```



```
/**  
 * onUserDetailsReceived is a method from the FirebaseCallback interface that sets the  
 * text of xpDisplayTv to the user's current XP.  
 */  
  
@Override  
public void onUserDetailsReceived() {  
    xpDisplayTv.setText(""+ current_user.getCurrentXP());  
}  
}
```



**ui**

**ui.Home**

**MainActivity : XML**



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer_layout"
    android:fitsSystemWindows="true"
    tools:openDrawer="start"
    tools:context=".ui.Home.MainActivity">

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

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <com.google.android.material.appbar.AppBarLayout
                android:id="@+id/appBarLayout"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:background="#000000"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent">

                <androidx.constraintlayout.widget.ConstraintLayout
                    android:layout_width="match_parent"
                    android:layout_height="match_parent">

                    <androidx.appcompat.widget.Toolbar
                        android:id="@+id/toolbar"
                        android:layout_width="match_parent"
                        android:layout_height="?attr/actionBarSize"
                        android:background="@color/dark_purple"
                        android:minHeight="?attr/actionBarSize"
                        android:theme="?attr/actionBarTheme"
                        app:layout_constraintEnd_toEndOf="parent"
                        app:layout_constraintStart_toStartOf="parent"
                        app:layout_constraintTop_toTopOf="parent"
                        app:navigationIcon="@drawable/ic_baseline_menu"
                        app:title="@string/app_name" />
                
```



```
<!--           <ImageView-->
<!--           android:id="@+id/imageView"-->
<!--           android:layout_width="28dp"-->
<!--           android:layout_height="25dp"-->
<!--           android:layout_marginEnd="24dp"-->
<!--           android:backgroundTint="#000000"-->
<!--
app:layout_constraintBottom_toBottomOf="parent"-->
    <!--
app:layout_constraintEnd_toEndOf="@+id/toolbar"-->
    <!--           app:layout_constraintTop_toTopOf="parent"-->
    <!--           app:srcCompat="@drawable/ic_baseline_search"
/>-->
<TextView
    android:id="@+id/xp_main_display"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="4dp"
    android:drawableLeft="@drawable/baseline_diamond_24"
    android:fontFamily="@font/bungee"
    android:text="XP"
    android:textColor="#FFFFFF"
    android:textSize="16sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

</com.google.android.material.appbar.AppBarLayout>
<!-- main content goes here -->
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/frame_layout"
    android:layout_marginTop="56dp"/>

<!-- main content ends-->
</androidx.constraintlayout.widget.ConstraintLayout>

</LinearLayout>

<androidx.coordinatorlayout.widget.CoordinatorLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```



```
<com.google.android.material.bottomappbar.BottomAppBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/bottomAppBar"
    android:layout_gravity="bottom"
    android:theme="@style/Theme.MaterialComponents"
    android:background="@color/white"
    app:fabCradleMargin="3dp"
    app:fabCradleRoundedCornerRadius="50dp">
    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/bottomNavigationView"
        android:layout_marginEnd="20dp"
        app:labelVisibilityMode="labeled"
        android:background="@android:color/transparent"
        app:menu="@menu/bottom_bar_navigation"
        android:contentDescription="this is the bottom bar"/>
</com.google.android.material.bottomappbar.BottomAppBar>
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/add_button_fab"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/white"
    android:src="@drawable/ic_baseline_add"
    app:layout_anchor="@+id/bottomAppBar"
    app:maxImageSize="40dp"
    app:tint="@color/purple_200"/>

</androidx.coordinatorlayout.widget.CoordinatorLayout>
<com.google.android.material.navigation.NavigationView
    android:id="@+id/navigation_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    app:headerLayout="@layout/navigation_header"
    app:menu="@menu/navigation_menu" />
</androidx.drawerlayout.widget.DrawerLayout>
```

**ui****ui.Home****FirebaseCallback**

```
package com.example.solobeast.ui.Home;

/**
Interface definition for a callback to be invoked when Firebase database operations complete.
@author Ofek Almog
*/
public interface FirebaseCallback {

    /**
     Called when user details are received from the Firebase database.
     */
    void onUserDetailsReceived();
}
```



**ui**

**ui.Home.Fragments**

**HomeFragment**



```
package com.example.solobeast.ui.Home.Fragments;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.example.solobeast.Adapters.RecyclerFunctionalities;
import com.example.solobeast.Adapters.RewardAdapter;
import com.example.solobeast.Adapters.TaskAdapter;
import com.example.solobeast.Extras.Utils;
import com.example.solobeast.Objects.Task;
import com.example.solobeast.R;
import com.example.solobeast.ui.DetailedTaskAct;
import com.example.solobeast.ui.Home.MainActivity;
import com.google.firebase.FirebaseNetworkException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.database.DataSnapshot;
import com.google.firebaseio.database.DatabaseError;
import com.google.firebaseio.database.DatabaseReference;
import com.google.firebaseio.database.FirebaseDatabase;
import com.google.firebaseio.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

/**
A fragment for displaying and managing the user's task list. The 'HomePage'
Implements the RecyclerFunctionalities interface to handle RecyclerView actions.
@author Ofek Almog
*/
public class HomeFragment extends Fragment implements RecyclerFunctionalities {

    /** List of tasks */
```



```
public static List<Task> tasksList;

/** Adapter for the RecyclerView */
TaskAdapter adapter;

/** RecyclerView for displaying the tasks */
RecyclerView recyclerView;

/** Reference to the Firebase Realtime Database */
DatabaseReference myRef;

/** Firebase Authentication instance */
FirebaseAuth mAuth;

/** 
 Called immediately after onCreateView has returned, but before any saved state has been
 restored in to the view.
 Initializes the RecyclerView and retrieves the user's tasks from the database.
 @param view The View returned by onCreateView.
 @param savedInstanceState If non-null, this fragment is being re-constructed from a previous
 saved state as given here.
 */
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    recyclerView = view.findViewById(R.id.task_recycler_view);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new GridLayoutManager(getContext(), 1));

    mAuth = FirebaseAuth.getInstance();

    // Set the reference to the Firebase Realtime Database
    myRef =
        FirebaseDatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-rtdb.firebaseio.com/");
    myRef = myRef.child("Users/")
        + FirebaseAuth.getInstance().getCurrentUser().getUid()
        + "/Tasks";

    //\\
    if(Utils.isConnectedToInternet(getContext()))
        this.retrieveData(this);
    else{
        AlertDialog.Builder alertDialog;
        alertDialog = new AlertDialog.Builder(getContext());
    }
}
```



```
AlertDialog
        .setMessage("You're not connected to internet, we can't proceed.")
        .setTitle("Internet Connection")
        .setCancelable(false)
        .setIcon(R.drawable.baseline_airplanemode_active_24);

    alertDialog.setPositiveButton("I WILL TURN IT OFF", (dialogInterface, i) -> {
        dialogInterface.cancel();
        this.retrieveData(this);
    });

    alertDialog.setNeutralButton("OK", (dialogInterface, i) ->{
        dialogInterface.cancel();
    });
    AlertDialog alert = alertDialog.create();
    alert.show();

}

}

/***
Retrieves the user's tasks from the Firebase Realtime Database and updates the RecyclerView.
@param recyclerViewFunctionalities An instance of the RecyclerViewFunctionalities interface.
*/
private void retrieveData(RecyclerViewFunctionalities recyclerViewFunctionalities) {
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            tasksList = new ArrayList<Task>();
            for(DataSnapshot data : snapshot.getChildren()){
                Task t = data.getValue(Task.class);
                tasksList.add(t);
            }
            adapter = new TaskAdapter(getContext(),tasksList,recyclerViewFunctionalities);
            recyclerView.setAdapter(adapter);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
        }
    });
}

}

/***
```



Called to have the fragment instantiate its user interface view.

Inflates the layout for this fragment and returns the inflated View object.

@param inflator The LayoutInflater object that can be used to inflate any views in the fragment.

@param container If non-null, this is the parent view that the fragment's UI should be attached to.

@param savedInstanceState If non-null, this fragment is being re-constructed from a previous saved state as given here.

@return The View for the fragment's UI, or null.

\*/

**@Override**

```
public View onCreateView(LayoutInflater inflator, ViewGroup container,
                        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflator.inflate(R.layout.fragment_home, container, false);
}
```

/\*\*

Method to move to the detailed screen of a selected task.

@param task The task to be shown in the detailed screen.

\*/

**private void moveToDetailedScreen(Task task){**

```
    Log.d("MoveScreen","Moving to detailed screen");
    Intent i = new Intent(getActivity(), DetailedTaskAct.class);
    Log.d("ObjectValues",task.toString());
    i.putExtra("selected_task_name",task.getName());
    i.putExtra("selected_task_time",task.getTime());
    i.putExtra("selected_task_desc",task.getDescription());
    i.putExtra("selected_task_diff",task.getDifficulty());
    i.putExtra("selected_task_key",task.getKey());
    i.putExtra("from_intent","Edit");
    startActivity(i);
}
```

/\*\*

Handles the action when an item in the RecyclerView is clicked.

Starts the DetailedTaskAct activity with the information of the selected task.

@param position The position of the selected item in the RecyclerView.

\*/

**@Override**

```
public void onItemClick(int position) {
```

```
    Task t = new Task(
        tasksList.get(position).getName(),
        tasksList.get(position).getTime(),
        tasksList.get(position).getDescription(),
        tasksList.get(position).getDifficulty(),
```



```
        tasksList.get(position).getKey()
    );
    moveToDetailedScreen(t);
}

/**
 Handles the long-click event of an item in the RecyclerView list.
 Shows an AlertDialog with a confirmation message for task deletion, and deletes the task
 from the database
 and updates the adapter on "Yes" button click.
 @param position the position of the item in the RecyclerView list.
 @return true if the event was consumed, false otherwise.
 */
@Override
public boolean onItemLongClick(int position) {
    AlertDialog.Builder alertDialog;
    alertDialog = new AlertDialog.Builder(getContext());

    alertDialog
        .setMessage("Are you sure you want to delete this task?")
        .setTitle("Delete Task");

    alertDialog.setPositiveButton("Yes", (dialogInterface, i) -> {
        dialogInterface.cancel();

        DatabaseReference myRef = FirebaseDatabase.getInstance().getReference(
            "Users/"
                + FirebaseAuth.getInstance().getCurrentUser().getUid()
                + "/Tasks"
        );
        myRef = myRef.child(tasksList.get(position).getKey());
        myRef.removeValue();
        adapter.notifyItemRemoved(position);

    });
    alertDialog.setNegativeButton("Cancel", (dialogInterface, i) -> {
        dialogInterface.cancel();
        Toast.makeText(getContext(), "Event was cancelled
successfully", Toast.LENGTH_LONG).show();
    });
    AlertDialog alert = alertDialog.create();
    alert.show();
    return true;
    //REMOVE FROM DB.
}
}
```



## ui

### ui.Home.Fragments

#### HomeFragment : 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.Home.Fragments.HomeFragment">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/task_recycler_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:layout_editor_absoluteX="0dp"
            tools:layout_editor_absoluteY="0dp"
            tools:listitem="@layout/task_item" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```



**ui**

**ui.Home.Fragments**

**RewardFragment**



```
package com.example.solobeast.ui.Home.Fragments;

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

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.example.solobeast.Adapters.RecyclerViewFunctionalities;
import com.example.solobeast.Adapters.RewardAdapter;
import com.example.solobeast.Objects.Reward;
import com.example.solobeast.R;
import com.example.solobeast.ui.DetailedRewardAct;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

/**
 * This class represents a Fragment for displaying a list of rewards. It implements the
 * RecyclerViewFunctionalities interface
 * to handle item click and long click events. It uses a RecyclerView to display the rewards list
 * and a RewardAdapter to populate
 * the RecyclerView with reward items. It also uses Firebase Realtime Database to retrieve and
 * manipulate data related to rewards.
 * @author Ofek Almog
 */
public class RewardFragment extends Fragment implements RecyclerViewFunctionalities {

    /**
     * A list of rewards to be displayed in the RecyclerView.
    
```



```
/*
List<Reward> rewardsList;

/**
An adapter for the RecyclerView.
*/
RewardAdapter adapter;

/**
The RecyclerView for displaying the rewards list.
*/
RecyclerView recyclerView;

/**
A reference to the Firebase Realtime Database.
*/
DatabaseReference myRef;

/**
An instance of FirebaseAuth to retrieve information about the currently logged in user.
*/
FirebaseAuth mAuth;

/**
Called immediately after onCreateView has returned, but before any saved state has been restored in to the view.
Initializes the RecyclerView and retrieves the user's rewards from the database.
@param view The View returned by onCreateView.
@param savedInstanceState If non-null, this fragment is being re-constructed from a previous saved state as given here.
*/
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    recyclerView = view.findViewById(R.id.reward_recycler_view);
    recyclerView.setHasFixedSize(true);
    recyclerView.setLayoutManager(new GridLayoutManager(getContext(), 1));

    mAuth = FirebaseAuth.getInstance();

    // Set the reference to the Firebase Realtime Database
    myRef =
    FirebaseDatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-rtdb.firebaseio.com/");
    myRef = myRef.child("Users/")
}
```



```
+ FirebaseAuth.getInstance().getCurrentUser().getUid()
+ "/Rewards");

this.retrieveData(this);

}

/***
Retrieves data from Firebase Realtime Database and populates the rewards list.
@param recyclerViewFunctionalities An instance of RecyclerViewFunctionalities to handle
RecyclerView click events.
*/
private void retrieveData(RecyclerViewFunctionalities recyclerViewFunctionalities) {
    myRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            rewardsList = new ArrayList<Reward>();
            for(DataSnapshot data : snapshot.getChildren()){
                Reward r = data.getValue(Reward.class);
                rewardsList.add(r);
            }
            adapter = new
RewardAdapter(getContext(),rewardsList,recyclerViewFunctionalities);
            recyclerView.setAdapter(adapter);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {
            Log.d("AuthData","retrieveData - on reward is cancelled");
        }
    });
}

/***
Called to have the fragment instantiate its user interface view.
Inflates the layout for this fragment and returns the inflated View object.
@param inflater The LayoutInflater object that can be used to inflate any views in the
fragment.
@param container If non-null, this is the parent view that the fragment's UI should be
attached to.
@param savedInstanceState If non-null, this fragment is being re-constructed from a previous
saved state as given here.
@return The View for the fragment's UI, or null.
*/
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
```



```
        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_reward, container, false);
}

/**
 Navigates to the DetailedRewardAct activity with the details of the selected reward.
@param reward The selected reward to display its details.
*/
private void moveToDetailedScreen(Reward reward){
    Log.d("MoveScreen","Moving to detailed screen");
    Intent i = new Intent(getActivity(), DetailedRewardAct.class);
    Log.d("ObjectValues",reward.toString());
    i.putExtra("selected_reward_name",reward.getRewardName());
    i.putExtra("selected_reward_desc",reward.getDescription());
    i.putExtra("selected_reward_xp",reward.getXP());
    i.putExtra("selected_reward_key",reward.getKey());
    i.putExtra("from_intent","Edit");
    startActivity(i);
}

/**
 This method is called when an item in the RecyclerView is clicked. It creates a new Reward
object using the data
 from the clicked item and passes it to the {@link #moveToDetailedScreen(Reward)} method to
start a new activity to display
the details of the selected reward.
@param position the position of the clicked item in the RecyclerView
*/
@Override
public void onItemClick(int position) {
    Reward t = new Reward(
        rewardsList.get(position).getRewardName(),
        rewardsList.get(position).getDescription(),
        rewardsList.get(position).getXP(),
        rewardsList.get(position).getKey()
    );
    moveToDetailedScreen(t);
}

/**
 Called when a reward item in the list is long clicked. Prompts the user to confirm if they
want to delete
the selected reward item. If the user confirms the deletion, the reward is removed from the
database and the
corresponding item is removed from the rewards list displayed on the screen.

```



```
    @param position The position of the reward item that was long clicked in  
the rewards list.  
    @return Returns true to indicate that the long click event has been consumed and should not  
be further handled.  
 */  
@Override  
public boolean onItemLongClick(int position) {  
    AlertDialog.Builder alertDialog;  
    alertDialog = new AlertDialog.Builder(getContext());  
  
    alertDialog  
        .setMessage("Are you sure you want to delete this reward?")  
        .setTitle("Delete Reward");  
  
    alertDialog.setPositiveButton("Yes", (dialogInterface, i) -> {  
        dialogInterface.cancel();  
  
        DatabaseReference myRef = FirebaseDatabase.getInstance().getReference(  
            "Users/"  
                + FirebaseAuth.getInstance().getCurrentUser().getUid()  
                +"/Rewards"  
        );  
        myRef = myRef.child(rewardsList.get(position).getKey());  
        myRef.removeValue();  
        adapter.notifyItemRemoved(position);  
  
    });  
    alertDialog.setNegativeButton("Cancel", (dialogInterface, i) -> {  
        dialogInterface.cancel();  
        Toast.makeText(getContext(), "Event was cancelled  
successfully", Toast.LENGTH_LONG).show();  
    });  
    AlertDialog alert = alertDialog.create();  
    alert.show();  
  
    return true;  
    //REMOVE FROM DB.  
}  
}
```

**ui****ui.Home.Fragments****RewardFragment : 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.Home.Fragments.RewardFragment">
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/reward_recycler_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            tools:layout_editor_absoluteX="0dp"
            tools:layout_editor_absoluteY="0dp"
            tools:listitem="@layout/reward_item" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</FrameLayout>
```



**ui**

**ui.Home.Fragments**

**ProfileFragment**



```
package com.example.solobeast.ui.Home.Fragments;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import com.example.solobeast.Objects.User;
import com.example.solobeast.R;
import com.example.solobeast.ui.Auth.RegisterAct;
import com.example.solobeast.ui.Home.MainActivity;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

import de.hdodenhof.circleimageview.CircleImageView;

/**
ProfileFragment is a fragment that displays the user's profile details, including their image,
email, and phone number.
@author Ofek Almog
*/
public class ProfileFragment extends Fragment {

    /** The user's profile image */
    Bitmap imageBitmap;

    /** The view to display the profile image */
    CircleImageView circleImageView;
```



```
/** The user's email address */
String userEmail;

/** The views to display the user's email and phone */
TextView userEmailTv, userPhoneTv;

/** The user's phone number */
String userPhoneNumber;

/**
 Called immediately after onCreateView() has returned, but before any saved state has been
 restored in to the view.
 @param view The View returned by onCreateView()
 @param savedInstanceState If the fragment is being re-created from a previous saved state,
 this is the state.
 */
@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    getUserDetails();
    init(view);

}

/**
 Initializes the views in this fragment.
 @param view The root view of the fragment
 */
private void init(View view){
    circleImageView = view.findViewById(R.id.profile_circle_image);
    getImageFromFireBase();

    userEmail = getEmail();
    userEmailTv = view.findViewById(R.id.task_name_et);
    userEmailTv.setText(userEmail);

    userPhoneTv = view.findViewById(R.id.phone_textview_profile_from_fb);
}

/**
 Called to create the view hierarchy associated with this fragment.
 @param inflater The LayoutInflator object that can be used to inflate any views in the
 fragment
 @param container The parent view that the fragment's UI should be attached to
 @param savedInstanceState This fragment is being re-constructed from a previous saved state
 as given here
}
```



```
        @return Return the View for the fragment's UI
*/
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_profile, container, false);
}

/**
 * Gets the user's profile image from Firebase Storage and sets it in the {@link
 #circleImageView} view.
 */
public void getImageFromFireBase(){
    StorageReference mImageRef =
FirebaseStorage.getInstance().getReference().child("profile-images/" +
FirebaseAuth.getInstance().getCurrentUser().getUid());
    final long ONE_MEGABYTE = 1024 * 1024;
    mImageRef.getBytes(ONE_MEGABYTE).addOnSuccessListener(bytes -> {
        imageBitmap = BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
        if(imageBitmap !=null)
            circleImageView.setImageBitmap(imageBitmap);
        //Toast.makeText(getActivity(), "successfully loaded your profile credentials",
Toast.LENGTH_LONG).show();
    }).addOnFailureListener(exception -> Toast.makeText(getContext(), "fail",
Toast.LENGTH_LONG).show());
}

/**
 * Gets the user's email address from FirebaseAuth.
 * @return the user's email address
 */
public String getEmail(){
    return FirebaseAuth.getInstance().getCurrentUser().getEmail();
}

/**
 * Gets the user's phone number from Firebase Database and sets it in the {@link #userPhoneTv}
TextView.
 */
public void getUserDetails(){
    DatabaseReference ref =
FirebaseDatabase.getInstance().getReference("Users").child(FirebaseAuth.getInstance().getUid()).child("phoneNumber");
    Task<DataSnapshot> task = ref.get();
    task.addOnSuccessListener(dataSnapshot -> {
```



```
        String str_p_num = dataSnapshot.getValue(String.class);
        userPhoneNumber = str_p_num;
        userPhoneTv.setText(userPhoneNumber);
        Log.d("AuthData","lol that's worked actually");
    }).addOnFailureListener(e -> {
    // Handle any errors here
    Log.d("AuthData","The operation is not good\nCause: \n" +e);
});

}
```



**ui**

**ui.Home.Fragments**

**ProfileFragment : XML**



```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".ui.Home.Fragments.ProfileFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/purple_200"
        android:orientation="vertical"
        >

        <androidx.cardview.widget.CardView
            android:id="@+id/card_view"
            android:layout_width="match_parent"
            android:layout_height="550dp"
            android:layout_marginVertical="60dp"
            app:cardCornerRadius="15dp"

            app:cardElevation="15dp"
            app:cardUseCompatPadding="true">

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

                <de.hdodenhof.circleimageview.CircleImageView
                    android:id="@+id/profile_circle_image"
                    android:layout_width="96dp"
                    android:layout_height="96dp"
                    android:layout_marginTop="20dp"
                    android:layout_gravity="center"
                    android:src="@drawable/profile_picture1"
                    app:civ_border_color="#FF000000"
                    app:civ_border_width="2dp" />

            <!-- <com.google.android.material.floatingactionbutton.FloatingActionButton-->
            <!--     android:id="@+id/profile_pic_fab"-->
            <!--     android:layout_width="wrap_content"-->
            <!--     android:layout_height="wrap_content"-->
            <!--     android:layout_marginTop="56dp"-->
```



```
<!-- android:layout_marginEnd="8dp"-->
<!-- android:src="@drawable/ic_baseline_add"-->
<!-- app:fabSize="mini"-->
<!-- app:layout_constraintEnd_toEndOf="@+id/profile_image"-->
<!-- app:layout_constraintHorizontal_bias="1.0"-->
<!-- app:layout_constraintStart_toStartOf="@+id/profile_image"-->
<!-- app:layout_constraintTop_toTopOf="@+id/profile_image" /-->

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Profile Info"
    android:fontFamily="@font/bungee"
    android:layout_gravity="center"/>
<TextView
    android:id="@+id/textview_email_profile"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:fontFamily="@font/bungee"
    android:text="Email" />

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

    android:id="@+id/textInputLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    app:startIconDrawable="@drawable/ic_baseline_contact_mail"
    android:hint="Email">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/task_name_et"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        tools:ignore="SpeakableTextPresentCheck" />

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



```
    android:id="@+id/textview_phone_profile"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginLeft="10dp"
    android:fontFamily="@font/bungee"
    android:text="Phone" />

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

    android:id="@+id/textInputLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    app:startIconDrawable="@drawable/baseline_phone"
    android:hint="Phone">

    <com.google.android.material.textfield.TextInputEditText

        android:id="@+id/phone_textview_profile_from_fb"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        tools:ignore="SpeakableTextPresentCheck" />

</com.google.android.material.textfield.TextInputLayout>
<TextView-->
    android:layout_marginTop="10dp"-->
    android:id="@+id/task_desc_1"-->
    android:layout_width="wrap_content"-->
    android:layout_height="wrap_content"-->
    android:layout_marginLeft="10dp"-->
    android:fontFamily="@font/bungee"-->
    android:text="Description" />-->

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

    android:id="@+id/textInputLayout1"-->
    android:layout_width="match_parent"-->
    android:layout_height="wrap_content"-->
    android:layout_marginTop="16dp"-->
    android:clickable="false"-->
```



```
<!-- android:hint="Description">-->

<!-- <com.google.android.material.textfield.TextInputEditText-->
<!-- android:id="@+id/task_desc_et"-->
<!-- android:layout_width="304dp"-->
<!-- android:layout_height="55dp"-->
<!-- android:layout_gravity="center"-->
<!-- android:focusable="false"-->
<!-- android:focusableInTouchMode="false"-->
<!-- tools:ignore="SpeakableTextPresentCheck" />-->

<!-- </com.google.android.material.textfield.TextInputLayout>-->
</LinearLayout>
<!-- <com.google.android.material.floatingactionbutton.FloatingActionButton-->
<!-- android:id="@+id/fab_edit_add_task"-->
<!-- android:layout_width="wrap_content"-->
<!-- android:layout_height="wrap_content"-->
<!-- android:layout_gravity="bottom|center"-->

<!-- app:srcCompat="@drawable/ic_baseline_edit"-->
<!-- android:layout_alignParentRight="true"-->
<!-- android:layout_alignParentEnd="true" />-->
</androidx.cardview.widget.CardView>

</LinearLayout>
</FrameLayout>
```



**ui**

**ui.Home.Fragments**

**SettingsFragment**



```
package com.example.solobeast.ui.Home.Fragments;

import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

import androidx.preference.EditTextPreference;
import androidx.preference.MultiSelectListPreference;
import androidx.preference.Preference;
import androidx.preference.PreferenceFragmentCompat;

import com.example.solobeast.R;

import java.util.Locale;

public class SettingsFragment extends PreferenceFragmentCompat {
    EditTextPreference editTextPreference;
    MultiSelectListPreference multiSelectListPreference;
    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
        setPreferencesFromResource(R.xml.root_preferences, rootKey);

        editTextPreference = findPreference("time_selector");
        multiSelectListPreference = findPreference("activities_multiselect_list_preference");
        multiSelectListPreference.setEntries(R.array.activities);

        multiSelectListPreference.setEntryValues(R.array.activities_values);
    }

    /**
     * Opens the time picker dialog.
     * @param view The view that triggered the method.
     */
    public void popTimePicker(View view) {
        TimePickerDialog.OnTimeSetListener onTimeSetListener = (timePicker, selectedHour,
selectedMinute) -> editTextPreference.setText(String.format(Locale.getDefault(),
"%02d:%02d", selectedHour, selectedMinute));
        //TimePickerDialog.OnCancelListener onCancelListener = (timePicker) ->
editTextPreference.performClick();

        TimePickerDialog timePickerDialog = new TimePickerDialog(getContext(), onTimeSetListener,
1, 0, true);
        //timePickerDialog.setOnCancelListener(onCancelListener);
        timePickerDialog.setTitle("Select Task Duration (HH:MM)");
        timePickerDialog.show();
    }
}
```



}

```
@Override
public void onDisplayPreferenceDialog(Preference preference) {
    //super.onDisplayPreferenceDialog(preference);
    switch (preference.getKey()){
        case "time_selector":
            popTimePicker(getView());
            break;
        case "activities_multiselect_list_preference":
            super.onDisplayPreferenceDialog(preference);
            break;
    }
}
```



**ui**

**ui.Home.Fragments**

**SettingsFragment : XML**



```
<PreferenceScreen xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:android="http://schemas.android.com/apk/res/android">

    <PreferenceCategory app:title="@string/notifications_settings">
        <CheckBoxPreference
            app:key="want_notifications"
            app:title="Want to get daily notifications?"
            app:summaryOff="mark the checkbox in-order to enable the notifications"
            app:summaryOn="un-check the checkbox in-order to disable daily notifications"
        />
        <EditTextPreference
            app:dependency="want_notifications"
            app:key="time_selector"
            app:title="Select Time for notifications"
            app:useSimpleSummaryProvider="true"
            app:summary="Tap to select time"
        />
    </PreferenceCategory>
    <PreferenceCategory app:title="Theme">

        <SwitchPreferenceCompat
            app:key="dark_theme"
            app:title="Dark theme"
            app:summaryOff="Tap the switch to enable dark theme"
            app:summaryOn="Tap the switch to enable light theme"
            app:defaultValue="true"/>

    </PreferenceCategory>
    <PreferenceCategory app:title="Onboadring">
        <!-- <MultiSelectListPreference-->
        <!--     app:key="activities_multiselect_list_preference"-->
        <!--     app:title="Select your guides you want to get"-->
        <!--     app:summary="Select your explanation to activities"-->
        <!--     android:entries="@array/activities"-->
        <!--     android:entryValues="@array/activities_values"-->
        <!--     app:dialogTitle="Choose activities"-->
        <!--     app:dialogMessage="Select one or more activities"-->
        <!--     app:positiveButtonText="OK"-->
        <!--     app:negativeButtonText="Cancel"/>-->
        <MultiSelectListPreference
            android:key="activities_multiselect_list_preference"
            android:title="Example MultiSelectListPreference"
            android:summary="Select your favorite colors"

            android:dialogTitle="Choose colors"
            android:dialogMessage="Select one or more colors"
    </PreferenceCategory>
</PreferenceScreen>
```



```
        android:positiveButtonText="Ok"  
        android:negativeButtonText="Cancel"/>  
</PreferenceCategory>  
  
</PreferenceScreen>
```





ui

**DetailedRewardAct**



```
package com.example.solobeast.ui;

import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import com.example.solobeast.Extras.PickerDialog;
import com.example.solobeast.Extras.onPickerSelectedListener;
import com.example.solobeast.Objects.Reward;
import com.example.solobeast.R;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.database.DatabaseReference;
import com.google.firebaseio.database.FirebaseDatabase;

/**
 * This activity class is responsible for displaying the details of a reward, including its name,
 * description and experience points. It also provides functionality to add a new reward or
 * update an existing one.
 * @author Ofek Almog
 */
public class DetailedRewardAct extends AppCompatActivity {

    /**
     * The TextInputEditTexts for the reward name, description, difficulty and experience points.
     */
    TextView rewardNameTv, rewardDescTv,rewardXpTv;

    /**
     * The reward being edited or added.
     */
    Reward reward;

    /**
     * The FloatingActionButton used to submit the form.
     */
    FloatingActionButton submitFloatingFormBtn;

    /**
     * A counter used to determine if the form has been submitted before. a UI updating solution.
     */
    int counter = 0;
```



```
/**  
 * The name of the activity that started this one.  
 */  
String fromActivity;  
  
/**  
 * Indicates whether the user is adding a new reward or editing an existing one.  
 */  
String userChoice;  
  
/**  
 * A reference to the Firebase Realtime Database.  
 */  
DatabaseReference listRef;  
  
/**  
 * Called when the activity is starting. Sets up the views and initializes the reward.  
 * @param savedInstanceState The saved state of the activity, if any.  
 */  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_detailed_reward);  
  
    init();  
    reward = new Reward();  
    determineEditOrAdd(fromActivity);  
  
    rewardXpTv.setOnFocusChangeListener((view, hasFocus) -> {  
        if(hasFocus)  
            popXpSelection(view);  
  
    });  
    submitFloatingFormBtn.setOnClickListener(view -> {  
        counter++;  
        if(counter == 1){  
            openEditTexts();  
            setUpdateIconDrawable();  
        }  
        else if(counter >= 2){  
            if(userChoice.equals("AddReward")){  
                addRewardToFirebase();  
                finish();  
            }  
        }  
    });  
}  
}
```



```
        else{
            updateRewardToFirebase();
            finish();
        }
    });
}

/**
 * Displays a number picker dialog allowing the user to select an experience point value for
 * the reward.
 * The selected value is displayed in the experience point TextInputEditText.
 * @param view The view that called this method, typically the experience point
 * TextInputEditText.
 */
private void popXpSelection(View view){
    // Instantiate the PickerDialog
    PickerDialog pickerDialog = new PickerDialog(this, "numbers");
    DialogInterface.OnCancelListener onCancelListener = (picker) ->
rewardDescTv.requestFocus();
    pickerDialog.setOnCancelListener(onCancelListener);

    // Set a listener to be notified when a name is selected
    pickerDialog.setOnNameSelectedListener(new onPickerSelectedListener() {
        @Override
        public void onNameSelected(String name) {
            // Display the selected name

        }

        @Override
        public void onNumberSelected(int num) {
            rewardXpTv.setText(String.valueOf(num));
        }
    });
}

// Show the dialog
pickerDialog.show();
}

/**
 * Initializes the activity by setting up views and retrieving data from the incoming intent.
 */
private void init(){
    if(getIntent() != null){
        fromActivity = getIntent().getStringExtra("from_intent");
    }
}
```



```
        }

    //put values of edittexts in variables
    rewardNameTv = findViewById(R.id.reward_name_et);
    rewardDescTv = findViewById(R.id.reward_desc_et);
    rewardXpTv = findViewById(R.id.reward_et_xp1);
    submitFloatingFormBtn = findViewById(R.id.fab_edit_add_reward_1);

}

/***
Determines whether the user is editing or adding a reward based on the name of the activity.
If the activity name is "Edit", it sets edit to true and calls the {@link
#getValuesFromPrevActivityReward()}
method to get the previous reward values, and {@link #insertEditTextsValues()} method to
insert the values
into the edit texts.
If the activity name is not "Edit", it sets edit to false and calls the {@link
#setAddIconDrawable()}
method to set the add icon drawable to the submit button, {@link #openEditTexts()} method to
open the edit texts,
and increments the counter by one.
@param activityName the name of the activity, either "Edit" or "Add"
*/
private void determineEditOrAdd(String activityName){
    boolean edit = false;
    if(activityName.equals("Edit")) edit = true;
    if(edit){
        userChoice = "EditReward";
        getValuesFromPrevActivityReward();
        insertEditTextsValues();

    }
    else{
        userChoice = "AddReward";
        setAddIconDrawable();
        openEditTexts();
    }
}

/***
This method retrieves the values of the selected reward from the previous activity
and sets them to the corresponding fields in the current activity's task object.
*/
private void getValuesFromPrevActivityReward() {
    reward.setRewardName(getIntent().getStringExtra("selected_reward_name"));
    reward.setDescription(getIntent().getStringExtra("selected_reward_desc"));
    reward.setXP(getIntent().getIntExtra("selected_reward_xp",0));
}
```



```
        reward.setKey(getIntent().getStringExtra("selected_reward_key"));  
    }  
  
    /**  
     * Converts a string to an integer. If the input string is empty or null, returns the integer  
     * 1.  
     * @param str the string to be converted to an integer.  
     * @return the integer representation of the input string, or 1 if the input string is empty or  
     * null.  
     */  
    public int strToInt(String str){  
        if(!str.equals(""))  
            return Integer.parseInt(str);  
        else {  
            return 1;  
        }  
    }  
  
    /**  
     * This method sets the reward data to the corresponding fields in the current activity's  
     * reward object.  
     * This method should be called after the {@link #insertEditTextsValues()} method.  
     */  
    private void insertEditTextsValues() {  
        rewardNameTv.setText(reward.getRewardName());  
        rewardDescTv.setText(reward.getDescription());  
        rewardXpTv.setText(String.valueOf(reward.getXP()));  
    }  
  
    /**  
     * This method updates the selected task in the Firebase Realtime Database.  
     * It creates a reference to the database and updates the corresponding reward node  
     * with the new values entered by the user in the EditText fields.  
     */  
    private void updateRewardToFirebase() {  
        listRef =  
Fireatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-firebase.com/");  
        listRef = listRef.child("Users/"  
            + FirebaseAuth.getInstance().getCurrentUser().getUid()  
            +"/Rewards"  
        );  
        System.out.println("---- the key is --- "+reward.getKey());  
        listRef = listRef.child(reward.getKey());  
        Reward t = new Reward(  
    }
```



```
    rewardNameTv.getEditableText().toString()
    ,rewardDescTv.getEditableText().toString()
    ,Integer.parseInt(rewardXpTv.getEditableText().toString())
    ,reward.getKey()
);
listRef.setValue(t);
finish();
}

/**
Adds a new reward to Firebase by creating a new reference in the Firebase database
with the reward details provided by the user. The newly created reward is associated
with the currently authenticated user.
*/
private void addRewardToFirebase() {
    listRef =
FirebaseDatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-rtbd.firebaseio.com/");
    listRef = listRef.child("Users/"
        + FirebaseAuth.getInstance().getCurrentUser().getUid()
        +"/Rewards");

    String key = listRef.push().getKey();

    listRef = listRef.child(key);
    Reward t = new Reward(
        rewardNameTv.getEditableText().toString()
        , rewardDescTv.getEditableText().toString()
        ,strToInt(rewardXpTv.getEditableText().toString())
        ,key
    );
    listRef.setValue(t);
    finish();
}

/**
Sets the add icon to the add button in the current activity.
*/
private void setAddIconDrawable() {
    if(submitFloatingFormBtn != null) {
        submitFloatingFormBtn.setImageResource(R.drawable.ic_baseline_add);
    }
}

/**
Sets the update icon to the update button in the current activity.

```



```
/*
private void setUpdateIconDrawable() {
    if(submitFloatingFormBtn != null) {
        submitFloatingFormBtn.setImageResource(R.drawable.ic_baseline_update);
    }
}

/**
 * Opens the edit texts in the current activity, allowing the user to modify the values of the
fields.
 */
private void openEditTexts(){
    rewardNameTv.setFocusable(true);
    rewardNameTv.setFocusableInTouchMode(true);

    rewardDescTv.setFocusable(true);
    rewardDescTv.setFocusableInTouchMode(true);

    rewardXpTv.setFocusable(true);
    rewardXpTv.setFocusableInTouchMode(true);
}
}
```



## DetailedRewardAct : 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:background="@color/teal_200"
    android:orientation="vertical"
    tools:context=".ui.DetailedRewardAct">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="630dp"
        android:layout_marginVertical="60dp"
        app:cardCornerRadius="15dp"

        app:cardElevation="15dp"
        app:cardUseCompatPadding="true">

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

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Reward Info"
                android:fontFamily="@font/bungee"
                android:layout_gravity="center"/>
            <LinearLayout
                android:id="@+id/or_view"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="12dp"
                android:gravity="center"
                android:orientation="horizontal"
                >
                <View
                    android:layout_width="match_parent"
                    android:layout_height="2dp"
                    android:layout_margin="2dp"
                    android:layout_weight="1"
                    android:background="?android:attr/listDivider"
                >
            
```



```
        android:backgroundTint="#FFFFFF" />

    </LinearLayout>
    <TextView
        android:id="@+id/reward_name1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/bungee"
        android:text="Name" />

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

        android:id="@+id/textInputLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:clickable="false"
        android:hint="Enter reward name">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/reward_name_et"
            android:layout_width="304dp"
            android:layout_height="55dp"
            android:layout_gravity="center"
            android:focusable="false"
            android:focusableInTouchMode="false"
            tools:ignore="SpeakableTextPresentCheck" />

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

    <TextView
        android:layout_marginTop="10dp"
        android:id="@+id/reward_desc_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/bungee"
        android:text="Description" />

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

        android:id="@+id/textInputLayout1"
```



```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:clickable="false"
        android:hint="Enter reward description">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/reward_desc_et"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        tools:ignore="SpeakableTextPresentCheck" />

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

<TextView
    android:id="@+id/reward_xp1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_gravity="center"
    android:fontFamily="@font/bungee"
    android:text="XP" />

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

    android:id="@+id/textInputLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    android:hint="Choose your xp reward">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/reward_et_xp1"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:textColorHint="#BCAAA4"
        tools:ignore="SpeakableTextPresentCheck" />
```



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

</LinearLayout>
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_edit_add_reward_1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|center"
    app:srcCompat="@drawable/ic_baseline_edit"
    android:backgroundTint="@color/white"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />
</androidx.cardview.widget.CardView>

</LinearLayout>
```



ui

**DetailedTaskAct**



```
package com.example.solobeast.ui;

import androidx.appcompat.app.AppCompatActivity;

import android.app.TimePickerDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.util.Log;
import android.view.View;

import com.example.solobeast.Extras.GuiderDialog;
import com.example.solobeast.Extras.PickerDialog;
import com.example.solobeast.Extras.onPickerSelectedListener;
import com.example.solobeast.Objects.Task;
import com.example.solobeast.R;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.database.DatabaseReference;
import com.google.firebaseio.database.FirebaseDatabase;

import java.util.Locale;

/**
 * This activity class is responsible for displaying the details of a task, including its name,
 * description, difficulty, and duration. It also provides functionality to add a new task or
 * update an existing one.
 * @author Ofek Almog
 */
public class DetailedTaskAct extends AppCompatActivity {

    /**
     * The TextInputEdits for the task name, description, difficulty and duration.
     */
    TextInputEditText taskNameTv,taskDescTv,taskDiffTv,taskTimeTv;

    /**
     * The task being edited or added.
     */
    Task task;

    /**
     * The FloatingActionButton used to submit the form.
     */
    FloatingActionButton submitFormBtn;
```



```
/**  
 * A counter used to determine if the form has been submitted before. a UI updating solution.  
 */  
int counter = 0;  
  
/**  
 * The name of the activity that started this one.  
 */  
String fromActivity;  
  
/**  
 * Indicates whether the user is adding a new task or editing an existing one.  
 */  
String userChoice;  
  
/**  
 * A reference to the Firebase Realtime Database.  
 */  
DatabaseReference listRef;  
  
/**  
 * Called when the activity is starting. Sets up the views and initializes the task.  
 * @param savedInstanceState The saved state of the activity, if any.  
 */  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_detailed_task);  
  
    init();  
    task = new Task();  
    determineEditOrAdd(fromActivity);  
    GuiderDialog gd = new GuiderDialog(this, "DetailedTaskAct", "This is a explanation for you  
😊");  
    gd.startDialog();  
    taskTimeTv.setOnFocusChangeListener((view, hasFocus) -> {  
        if(hasFocus)  
            popTimePicker(view);  
    });  
  
    taskDiffTv.setOnFocusChangeListener((view, hasFocus) -> {  
        if (hasFocus)  
            popDiffSelection(view);  
    });  
}
```



```
        submitFormBtn.setOnClickListener(view -> {
    ++counter;
    if(counter == 1){
        openEditTexts();
        setUpdateIconDrawable();
    }
    else if(counter >= 2){
        if(userChoice.equals("AddTask")){
            addTaskToFirebase();
            finish();
        }
        else{
            updateTaskToFirebase();
            finish();
        }
    }
});
}

/**
 * Opens the time picker dialog.
 * @param view The view that triggered the method.
 */
public void popTimePicker(View view) {
    TimePickerDialog.OnTimeSetListener onTimeSetListener = (timePicker, selectedHour,
selectedMinute) -> taskTimeTv.setText(String.format(Locale.getDefault(),
"%02d:%02d", selectedHour, selectedMinute));
    TimePickerDialog.OnCancelListener onCancelListener = (timePicker) ->
taskDescTv.requestFocus();

    TimePickerDialog timePickerDialog = new TimePickerDialog(this, onTimeSetListener, 1,0,
true);
    timePickerDialog.setOnCancelListener(onCancelListener);
    timePickerDialog.setTitle("Select Task Duration (HH:MM)");
    timePickerDialog.show();
}

/**
 * Opens the difficulty picker dialog.
 * @param view The view that triggered the method.
 */
public void popDiffSelection(View view){
    // Instantiate the PickerDialog
    PickerDialog pickerDialog = new PickerDialog(this, "diff");

    pickerDialog.setOnNameSelectedListener(new onPickerSelectedListener() {
```



```
    @Override
    public void onNameSelected(String name) {
        // Display the selected name
        taskDiffTv.setText(name);
    }

    @Override
    public void onNumberSelected(int num) {
        // This method is not used in this example
    }
});

pickerDialog.show();
}

/**
 * Initializes the activity by setting up views and retrieving data from the incoming intent.
 */
private void init(){
    if(getIntent() != null){
        fromActivity = getIntent().getStringExtra("from_intent");
    }
    taskNameTv = findViewById(R.id.task_name_et);
    taskDescTv = findViewById(R.id.task_desc_et);
    taskDiffTv = findViewById(R.id.task_diff_et);
    taskTimeTv = findViewById(R.id.task_time_et);
    submitFormBtn = findViewById(R.id.fab_edit_add_task);
}

/**
 * Determines whether the user is editing or adding a task based on the name of the activity.
 * If the activity name is "Edit", it sets edit to true and calls the {@link #getValuesFromPrevActivityTask()}
 * method to get the previous task values, and {@link #insertEditTextsValues()} method to insert the values
 * into the edit texts.
 * If the activity name is not "Edit", it sets edit to false and calls the {@link #setAddIconDrawable()}
 * method to set the add icon drawable to the submit button, {@link #openEditTexts()} method to open the edit texts,
 * and increments the counter by one.
 * @param activityName the name of the activity, either "Edit" or "Add"
 */
private void determineEditOrAdd(String activityName){
    boolean edit = false;
    if(activityName.equals("Edit")) edit = true;
```



```
        if(edit){
    userChoice = "EditTask";
    getValuesFromPrevActivityTask();
    insertEditTextsValues();
}
else{
    userChoice = "AddTask";

    setAddIconDrawable();
    openEditTexts();
    ++counter;
}
}

/***
This method retrieves the values of the selected task from the previous activity
and sets them to the corresponding fields in the current activity's task object.
*/
private void getValuesFromPrevActivityTask() {
    task.setName(getIntent().getStringExtra("selected_task_name"));
    task.setTime(getIntent().getStringExtra("selected_task_time"));
    task.setDesc(getIntent().getStringExtra("selected_task_desc"));
    task.setDiff(getIntent().getStringExtra("selected_task_diff"));
    task.setKey(getIntent().getStringExtra("selected_task_key"));
}

/***
This method sets the task data to the corresponding fields in the current activity's task
object.
This method should be called after the {@link #insertEditTextsValues()} method.
*/
private void insertEditTextsValues() {
    taskNameTv.setText(task.getName());
    taskTimeTv.setText(task.getTime());
    taskDescTv.setText(task.getDescription());
    taskDiffTv.setText(task.getDifficulty());
}

/***
 * This method updates the selected task in the Firebase Realtime Database.
 * It creates a reference to the database and updates the corresponding task node
 * with the new values entered by the user in the EditText fields.
 */
private void updateTaskToFirebase() {
    listRef =
FirebaseDatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-rtdb.firebaseio.com/tasks.json")
    .child(task.getKey())
    .setValue(task);
}
}
```



```
        aseio.com/");
listRef = listRef.child("Users/"
    + FirebaseAuth.getInstance().getCurrentUser().getUid()
    +"/Tasks"
);
listRef = listRef.child(task.getKey());
Task t = new Task(
    taskNameTv.getEditableText().toString()
    ,taskTimeTv.getEditableText().toString()
    ,taskDescTv.getEditableText().toString()
    ,taskDiffTv.getEditableText().toString()
    ,task.getKey()
);
listRef.setValue(t);
}

/**
 * Adds a new task to Firebase by creating a new reference in the Firebase database
 * with the task details provided by the user. The newly created task is associated
 * with the currently authenticated user.
 */
private void addTaskToFirebase() {
    listRef =
    FirebaseDatabase.getInstance().getReferenceFromUrl("https://solobeast-android-default-rtbd.firebaseio
aseio.com/");
    listRef = listRef.child("Users/"
        + FirebaseAuth.getInstance().getCurrentUser().getUid()
        +"/Tasks");

    String key = listRef.push().getKey();

    listRef = listRef.child(key);
    Task t = new Task(
        taskNameTv.getEditableText().toString()
        ,taskTimeTv.getEditableText().toString()
        ,taskDescTv.getEditableText().toString()
        ,taskDiffTv.getEditableText().toString()
        ,key
    );
    listRef.setValue(t);
    finish();
}

/**
 * Sets the add icon to the submit button in the current activity.
 */
```



```
private void setAddIconDrawable() {
    submitFormBtn.setImageResource(R.drawable.ic_baseline_add);
}

/**
 * Sets the update icon to the submit button in the current activity.
 */
private void setUpdateIconDrawable() {
    submitFormBtn.setImageResource(R.drawable.ic_baseline_update);
}

/**
 * Opens the edit texts in the current activity, allowing the user to modify the values of the
 * fields.
 */
private void openEditTexts(){
    taskNameTv.setFocusable(true);
    taskNameTv.setFocusableInTouchMode(true);

    taskDescTv.setFocusable(true);
    taskDescTv.setFocusableInTouchMode(true);

    taskTimeTv.setFocusable(true);
    taskTimeTv.setFocusableInTouchMode(true);

    taskDiffTv.setFocusable(true);
    taskDiffTv.setFocusableInTouchMode(true);
}
}
```



ui

## DetailedTaskAct : 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:background="@color/purple_200"
    android:orientation="vertical"
    tools:context=".ui.DetailedTaskAct">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="700dp"
        android:layout_marginVertical="60dp"
        app:cardCornerRadius="15dp"

        app:cardElevation="15dp"
        app:cardUseCompatPadding="true">

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

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Task Info"
                android:fontFamily="@font/bungee"
                android:layout_gravity="center"/>
            <LinearLayout
                android:id="@+id/or_view"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="12dp"
                android:gravity="center"
                android:orientation="horizontal"
                >
                <View
                    android:layout_width="match_parent"
                    android:layout_height="2dp"
                    android:layout_margin="2dp"
                    android:layout_weight="1"
                    android:background="?android:attr/listDivider"
                >
            
```



```
        android:backgroundTint="#FFFFFF" />

    </LinearLayout>
    <TextView
        android:id="@+id/task_name1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:fontFamily="@font/bungee"
        android:layout_marginTop="10dp"
        android:layout_gravity="center"
        android:text="Name" />

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

    android:id="@+id/textInputLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    android:hint="Enter task name">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/task_name_et"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        tools:ignore="SpeakableTextPresentCheck" />

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

    <TextView
        android:layout_marginTop="10dp"
        android:id="@+id/task_desc_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:fontFamily="@font/bungee"
        android:text="Description" />

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



```
        android:id="@+id/textInputLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:clickable="false"
        android:hint="Enter task description">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/task_desc_et"
        android:layout_width="304dp"
        android:layout_height="75dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        tools:ignore="SpeakableTextPresentCheck" />

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

<TextView
    android:id="@+id/task_time1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_gravity="center"
    android:fontFamily="@font/bungee"
    android:text="Duration" />

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

    android:id="@+id/textInputLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    android:hint="Click to choose task duration">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/task_time_et"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:textColorHint="#BCAAA4"
        tools:ignore="SpeakableTextPresentCheck" />
```



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

<TextView
    android:id="@+id/task_diff1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_gravity="center"
    android:fontFamily="@font/bungee"
    android:text="Difficulty" />

<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/textInputLayout3"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:clickable="false"
    android:hint="Choose your task difficulty">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/task_diff_et"
        android:layout_width="304dp"
        android:layout_height="55dp"
        android:layout_gravity="center"
        android:focusable="false"
        android:focusableInTouchMode="false"
        android:textColorHint="#BCAAA4"
        tools:ignore="SpeakableTextPresentCheck" />

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

</LinearLayout>
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/fab_edit_add_task"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="bottom|center"
    android:backgroundTint="@color/white"
    app:srcCompat="@drawable/ic_baseline_edit"
    android:layout_alignParentRight="true"
```



```
        android:layout_alignParentEnd="true" />
</androidx.cardview.widget.CardView>

</LinearLayout>
```



ui

**TaskTimerAct**



```
package com.example.solobeast.ui;

import static com.example.solobeast.ui.Home.MainActivity.updateXPToUserFirebase;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.NotificationCompat;
import androidx.core.app.NotificationManagerCompat;

import android.app.ActivityManager;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.provider.Settings;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import com.example.solobeast.Extras.Services.CountdownThread;
import com.example.solobeast.Extras.Services.TimerService;
import com.example.solobeast.R;
import com.example.solobeast.ui.Home.Fragments.ProfileFragment;
import com.example.solobeast.ui.Home.MainActivity;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebaseio.database.DataSnapshot;
import com.google.firebaseio.database.DatabaseReference;
import com.google.firebaseio.database.FirebaseDatabase;

import java.util.Arrays;
import java.util.List;
import java.util.Locale;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.TimeUnit;

/**
 * The TaskTimerAct class represents an activity that sets a timer for a task.
 * It extends the AppCompatActivity class.
 * @author Ofek Almog
 */
```



```
*/  
public class TaskTimerAct extends AppCompatActivity {  
  
    /** The time string for the task. */  
    String time;  
  
    /** The TextView for displaying the clock. */  
    TextView clock;  
  
    /** The TextView for displaying the task time. */  
    TextView timeTV;  
  
    /** The notification counter. */  
    static int notificationCounter;  
  
    /** The time difficulty. */  
    static String diff;  
  
    /** The time in minutes. */  
    static int timeInMinutes;  
  
    /**  
     * Initializes the activity.  
     */  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_task_timer);  
        init();  
  
        // Convert time string to seconds  
        int timerSeconds = indicateTimeAsSeconds(time);  
        timeInMinutes = timerSeconds / 60;  
  
        // Calculate XP for task  
        int xpToAdd = calculatePointsForTask(timeInMinutes,diff);  
        sendNotificationToUser("This task will give you", "{ " + xpToAdd + " } xp" );  
  
        long timerAsMiliseconds = timerSeconds * 1000;  
        scheduleAnotification(timerAsMiliseconds);  
  
        // Start the timer  
        CountDownTimer timer= new CountDownTimer(timerAsMiliseconds, 1000){  
            @Override  
            public void onTick(long l) {
```



```
// Convert duration to hours, minutes, and seconds
long durationMillis = 1;
long hours = TimeUnit.MILLISECONDS.toHours(durationMillis);
long minutes = TimeUnit.MILLISECONDS.toMinutes(durationMillis) -
TimeUnit.HOURS.toMinutes(hours);
long seconds = TimeUnit.MILLISECONDS.toSeconds(durationMillis) -
TimeUnit.HOURS.toSeconds(hours) - TimeUnit.MINUTES.toSeconds(minutes);

// Display duration as a string in the clock TextView

String durationString = String.format("%02d : %02d : %02d", hours, minutes,
seconds);

clock.setText(durationString);
}

@Override
public void onFinish() {
    // Display XP earned and update Firebase

    int xpToAdd = calculatePointsForTask(timeInMinutes,diff);
    sendNotificationToUser("You did it!", "You're rewarded yourself with more { "+
xpToAdd + " } xp" );
    updateXPtoUserFirebase(xpToAdd,"IncreaseVal");
    finish();
}
}.start();

// Start the TimerService
Intent taskTimer = new Intent(this, TimerService.class);
taskTimer.putExtra("task_time",timerSeconds);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    startService(taskTimer);
}

// Set up onClickListener for cancel button
timeTV.setOnClickListener(view -> {
    stopService(taskTimer);
    timer.cancel();
    sendNotificationToUser("You cancelled the task","Waiting for you to come back... ");
    finish();
});

}

/**
```



```
*           Calculates XP earned for a task.
* @param taskTimeInMinutes The time taken to complete the task in minutes.
* @param difficulty The difficulty of the task.
* @return The XP earned for the task.
*/
private int calculatePointsForTask(int taskTimeInMinutes, String difficulty) {
    int basePoints = 0;
    switch (difficulty) {
        case "EASY":
            basePoints = 10;
            break;
        case "MEDIUM":
            basePoints = 20;
            break;
        case "HARD":
            basePoints = 30;
            break;
        default:
            break;
    }

    // Calculate points based on time taken to complete task
    double timeInHours = taskTimeInMinutes / 60.0;
    double points = basePoints * timeInHours;
    if ((int)points == 0){
        return 1;
    }
    // Ensure points are capped at 200
    return Math.min(200, (int) points);
}

/**
 Schedule notifications for specific time intervals before the end of the timer.
 If the time left is less than a specific interval, a notification will not be scheduled for
 that interval.
@param timeAsMilliseconds The remaining time on the timer in milliseconds.
*/
private void scheduleAnotification(long timeAsMilliseconds){
    if (timeAsMilliseconds > 2*4*900000){ //2-hour
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                sendNotificationToUser("KEEP GOING!","2 hours left.");
                Log.d("Notifications","scheduled a notification for 2 hours mark");
            }
        });
    }
}
```



```
        }, timeAsMilliseconds -2*4*900000);

    // Start the timer
}

if (timeAsMilliseconds >4*900000){ //1-hour
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
        @Override
        public void run() {
            sendNotificationToUser("KEEP GOING!","1 hour left.");
            Log.w("Notifications","scheduled a notification for 1 hours mark");

        }
    }, timeAsMilliseconds - 4*900000);

    // Start the timer
}

if (timeAsMilliseconds >2*900000){ //30-min
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
        @Override
        public void run() {
            sendNotificationToUser("YOU'RE ALMOST DONE!","30 minutes left.");
            Log.d("Notifications","scheduled a notification for 30 min mark");
        }
    }, timeAsMilliseconds - 2*900000);

    // Start the timer
}

if (timeAsMilliseconds > 900000){ //15-min
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
        @Override
        public void run() {
            sendNotificationToUser("YOU'RE ALMOST DONE!","15 minutes left.");
            Log.d("Notifications","scheduled a notification for 15 min mark");

        }
    }, timeAsMilliseconds - 900000);

    // Start the timer
}

if (timeAsMilliseconds > 300000){ //5-min
    Timer timer = new Timer();
    timer.schedule(new TimerTask() {
        @Override
```



```
        public void run() {
            sendNotificationToUser("YOU'RE ALMOST DONE!","5 minutes left.");
            Log.d("Notifications","scheduled a notification for 5 minutes mark");
        }
    }, timeAsMilliseconds - 300000);

    // Start the timer
}

/***
 * Sends a notification to the user with the specified title and message content.
 *
 * The method creates a notification with the specified title and message content,
 * and sends it to the user. The notification is shown with the app's icon and the current
 * timestamp.
 * When the user taps the notification, the app is launched and the main activity is
 * displayed.
 *
 * @param title the title of the notification.
 * @param msgContent the message content of the notification.
 */
private void sendNotificationToUser(String title, String msgContent) {
    //phase 1
    int icon = R.drawable.baseline_notification_important;
    long when = System.currentTimeMillis();

    //phase 2
    Intent intent = new Intent(TaskTimerAct.this, TaskTimerAct.class);
    PendingIntent pendingIntent = PendingIntent.getActivity(TaskTimerAct.this, 0, intent,
PendingIntent.FLAG_IMMUTABLE | PendingIntent.FLAG_UPDATE_CURRENT);
    NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

    NotificationCompat.Builder builder = new
NotificationCompat.Builder(getApplicationContext(), "M_CH_ID");
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        String channelId = "TimerNotifications";
        NotificationChannel channel = new NotificationChannel(channelId,
                "human readable title :)",
                NotificationManager.IMPORTANCE_DEFAULT);
        notificationManager.createNotificationChannel(channel);
        builder.setChannelId(channelId);
    }
    //phase 3
```



```
        Notification notification = builder.setContentIntent(pendingIntent)
            .setSmallIcon(icon).setWhen(when)
            .setAutoCancel(true).setContentTitle(title)
            .setContentText(msgContent).build();

    notificationManager.notify(notificationCounter, notification);
    notificationCounter++;
    // Show the notification to the user

}

/***
Initializes the activity by finding the necessary views and retrieving
data from the Intent passed to the activity.
Sets up the initial values for the timer and notification counter.
*/
private void init(){
    timeTV = findViewById(R.id.timer_tv);
    //airplaneModeTv = findViewById(R.id.airplane_mode_tv);
    clock = findViewById(R.id.text_view_timer);
    notificationCounter = 0;
    if(getIntent() != null){
        time= getIntent().getStringExtra("selected_task_time");
        diff= getIntent().getStringExtra("selected_task_diff");
        Log.d("Value","the time is: \t"+time);
    }
}

/***
Converts a string in the format "hh:mm" to the total number of seconds represented by the
time.
@param time A string representing a time in the format "hh:mm".
@return An integer representing the total number of seconds represented by the input time.
*/
private int indicateTimeAsSeconds(String time){
    int timerTime = 0;
    List<String> arrayList = Arrays.asList(time.split(":"));
    String hours = arrayList.get(0);
    String minutes = arrayList.get(1);
    timerTime = Integer.parseInt(minutes) * 60 + Integer.parseInt(hours) * 3600;
    return timerTime;
}

}
```



אופק אלמוג - SoloBeast



**ui**

**TaskTimerAct : XML**



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

    <TextView
        android:id="@+id/timer_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="88dp"
        android:textSize="18sp"
        android:fontFamily="@font/bungee"
        android:text="@string/task_text_stop"
        android:gravity="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/text_view_timer"
        android:textStyle="bold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="148dp"
        android:drawableTop="@drawable/baseline_timer_128"
        android:textColor="#1991D0"
        android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/timer_tv" />

    <!-- <TextView-->
    <!--     android:id="@+id/airplane_mode_tv"-->
    <!--     android:layout_width="wrap_content"-->
    <!--     android:layout_height="wrap_content"-->
    <!--     android:layout_margin="10dp"-->
    <!--     android:fontFamily="@font/bungee"-->
    <!--     android:text="@string/let_user_decide_airplane_mode"-->
    <!--     android:textSize="18sp"-->
    <!--     app:layout_constraintBottom_toBottomOf="parent"-->
```



```
<!-- app:layout_constraintEnd_toEndOf="parent"-->
<!-- app:layout_constraintStart_toStartOf="parent"-->
<!-- app:layout_constraintTop_toBottomOf="@+id/text_view_timer" />-->

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



## Extras

### ActivityGuideTracker



```
package com.example.solobeast.Extras;

import android.content.Context;
import android.content.SharedPreferences;

/**
 * A utility class for tracking which activities or fragments have been visited
 * by the user in a mobile app.
 * @author Ofek Almog
 */
public class ActivityGuideTracker {

    /**
     * The name of the shared preferences file where the activity/fragment visit status is saved.
     */
    private static final String PREFS_NAME = "activity_tracker_prefs";

    /**
     * The shared preferences object used to store and retrieve activity/fragment visit status.
     */
    private SharedPreferences mPrefs;

    /**
     * Constructs a new ActivityGuideTracker instance.
     *
     * @param context The application context.
     */
    public ActivityGuideTracker(Context context) {
        mPrefs = context.getSharedPreferences(PREFS_NAME, Context.MODE_PRIVATE);
    }

    /**
     * Returns true if the given activity or fragment has been visited by the user before,
     * false otherwise.
     *
     * @param activityOrFragmentName The name of the activity or fragment to check.
     * @return True if the activity or fragment has been visited before, false otherwise.
     */
    public boolean isVisited(String activityOrFragmentName) {
        return mPrefs.getBoolean(activityOrFragmentName, false);
    }

    /**
     * Sets the visited status of the given activity or fragment to true.
     *
     * @param activityOrFragmentName The name of the activity or fragment to mark as visited.
     */
}
```



```
/*
public void setVisited(String activityOrFragmentName) {
    mPrefs.edit().putBoolean(activityOrFragmentName, true).apply();
}

/***
 * Returns true if the given activity or fragment has not been visited by the user yet,
 * false otherwise.
 * @param activityOrFragmentName The name of the activity or fragment to check.
 * @return True if the activity or fragment has not been visited yet, false otherwise.
 */
public boolean isUnvisited(String activityOrFragmentName) {
    return mPrefs.getBoolean(activityOrFragmentName, false);
}

/***
 * Sets the visited status of the given activity or fragment to false.
 * @param activityOrFragmentName The name of the activity or fragment to mark as unvisited.
 */
public void setUnvisited(String activityOrFragmentName) {
    mPrefs.edit().putBoolean(activityOrFragmentName, false).apply();
}

/***
 * Clear all saved activity/fragment visit status from the shared preferences.
 */
public void clearActivitiesStatus() {
    mPrefs.edit().clear().apply();
}

}
```



## Extras

### GuiderDialog



```
package com.example.solobeast.Extras;

import android.app.Dialog;
import android.content.Context;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import com.example.solobeast.R;

/**
A custom dialog class used to display guide messages for activities to the user. {@link
ActivityGuideTracker}
@author Ofek Almog
*/
public class GuiderDialog extends Dialog {

    /**The context of the dialog*/
    private Context mContext;

    /** The name of the activity associated with the dialog */
    private String mActivityName;

    /** The explanation message to display in the dialog */
    private String mExplanation;

    /** The activity tracker instance */
    private ActivityGuideTracker activityTracker;

    /**
     * Constructs a new GuiderDialog instance.
     * @param context The application context.
     * @param activityName The name of the activity to display the guide for.
     * @param explanation The message to display to the user.
     */
    public GuiderDialog(Context context, String activityName, String explanation) {
        super(context);
        mContext = context;
        mActivityName = activityName;
        mExplanation = explanation;
    }

    /**
     * Initializes the dialog view and shows the dialog.
     */
    private void init() {
```



```
        show();
    setContentView(R.layout.custom_guider_dialog);
    TextView explanationTv = findViewById(R.id.explanation_text);
    explanationTv.setText(mExplanation);
    Button okBtn = findViewById(R.id.proceed_guide_btn);

    okBtn.setOnClickListener(view ->
        proceedOk());
}

/**
 Starts the dialog for the given activity if it has not been visited before.
 */
public void startDialog() {
    activityTracker = new ActivityGuideTracker(mContext);
    if (!activityTracker.isVisited(mActivityName)) {
        init();
        activityTracker.setVisited(mActivityName);
    }
}

/**
 Dismisses the dialog when the "OK" button is clicked.
 */
public void proceedOk(){
    dismiss();
}
}
```



## Extras

### GuiderDialog : XML



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:cardview="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/grey_blue"
    >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="16dp"
        >

        <ImageView
            android:id="@+id/image_view"
            android:layout_width="150dp"
            android:layout_height="150dp"
            android:layout_gravity="center"
            android:src="@drawable/img" />
        <androidx.core.widget.NestedScrollView
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                >
                <TextView
                    android:id="@+id/explanation_text"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="50dp"
                    android:textColor="#FAB007"
                    android:fontFamily="@font/bungee"
                    android:text="Your explanation text " />
                <Button
                    android:id="@+id/proceed_guide_btn"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:backgroundTint="#E85757"
                    android:layout_marginTop="50dp"
                    android:text="I Understood" />
            
        
    

```



```
</LinearLayout>

</androidx.core.widget.NestedScrollView>

</LinearLayout>

</androidx.cardview.widget.CardView>
```



## Extras

### onPickerSelectedListener

```
package com.example.solobeast.Extras;

/**
This interface defines a callback to be invoked when a name or a number is selected from a
picker.
@author Ofek Almog
*/
public interface onPickerSelectedListener {

    /**
     Called when a name is selected from the picker.
     @param name The name selected from the picker.
     */
    void onNameSelected(String name);

    /**
     Called when a number is selected from the picker.
     @param num The number selected from the picker.
     */
    void onNumberSelected(int num);
}
```



## Extras

### PickerDialog



```
package com.example.solobeast.Extras;

import android.app.Dialog;
import android.content.Context;
import android.widget.Button;
import android.widget.NumberPicker;

import androidx.annotation.Nullable;

import com.example.solobeast.R;

/**
A custom dialog for picking a name or number from a list of items.
@author Ofek Almog
*/
public class PickerDialog extends Dialog {
    private static final String[] DIFFICULTIES = {"EASY", "MEDIUM", "HARD"};
    private final String action;
    private onPickerSelectedListener mListener;
    private NumberPicker mPicker;

    /**
     Constructor for creating a new instance of the PickerDialog.
     @param context The Context in which the dialog should be created.
     @param action The action to perform when an item is selected. Can be "diff" for selecting
difficulty level or "number" for selecting a number.
    */
    public PickerDialog(Context context, String action) {
        super(context);
        this.action = action;
        init();
    }

    /**
     Initializes the dialog by setting its layout and listeners based on the selected action.
    */
    private void init() {
        setContentView(R.layout.custom_picker_dialog);
        Button okButton = findViewById(R.id.ok_btn);

        mPicker = findViewById(R.id.number_picker_1);
        if(action.equals("diff")){
            mPicker.setMinValue(0);
            mPicker.setMaxValue(DIFFICULTIES.length - 1);
            mPicker.setDisplayedValues(DIFFICULTIES);
        }
    }
}
```



```
        okButton.setOnClickListener(v -> {
            if (mListener != null) {
                mListener.onNameSelected(DIFFICULTIES[mPicker.getValue()]);
            }
            dismiss();
        });
    }
    else{
        mPicker.setMinValue(0);
        mPicker.setMaxValue(200);

        okButton.setOnClickListener(v -> {
            if (mListener != null) {
                mListener.onNumberSelected(mPicker.getValue());
            }
            dismiss();
        });
    }

}

/***
Set a listener to be notified when an item is selected.
@param listener The listener to set.
*/
public void setOnNameSelectedListener(onPickerSelectedListener listener) {
    mListener = listener;
}

/***
Sets a listener to be invoked when the dialog is canceled.
@param listener The {@link OnCancelListener} to set, or null to remove the currently set
listener.
*/
@Override
public void setOnCancelListener(@Nullable OnCancelListener listener) {
    super.setOnCancelListener(listener);
}

}
```



## Extras

### PickerDialog : XML

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="200dp"
    android:layout_height="350dp"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:background="#5044e4"
    android:id="@+id/custom_guide_dialog">

    <NumberPicker
        android:id="@+id/number_picker_1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="16dp"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ok_btn"
        android:text="Click to proceed"
        android:layout_gravity="center"
        android:layout_marginTop="150dp"
        />

</androidx.cardview.widget.CardView>
```



## Extras

### SplashAct



```
package com.example.solobeast.Extras;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

import com.example.solobeast.R;
import com.example.solobeast.ui.Auth.LoginAct;

/**
The SplashAct is an activity that shows a splash screen with an animation and then starts the
main activity.
@author Ofek Almog
*/
public class SplashAct extends AppCompatActivity {

    /**
     * The ImageView displaying the splash screen animation.
     */
    private ImageView splashImage;

    /**
     * Called when the activity is starting.
     */
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        splashImage = findViewById(R.id.splash_image_imgv);

        Animation animation = AnimationUtils.loadAnimation(this, R.anim.splash_rainbow);
        splashImage.startAnimation(animation);

        // Wait for the animation to finish and start the main activity
        animation.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
            }
        });
    }
}
```



```
    @Override
    public void onAnimationEnd(Animation animation) {
        Intent intent = new Intent(SplashAct.this, LoginAct.class);
        startActivity(intent);
        finish();
    }

    @Override
    public void onAnimationRepeat(Animation animation) {
    }
});
```



## Extras

### SplashAct : 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:background="#6A55B5"
    tools:context=".Extras.SplashAct">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SoloBeast"
        android:textSize="50sp"
        android:fontFamily="@font/bungee"
        android:layout_marginTop="10dp"
        android:textColor="@color/black"
        android:layout_gravity="center"/>

    <ImageView
        android:id="@+id/splash_image_imgv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"

        android:layout_marginTop="50dp"
        android:src="@drawable/try_two_png" />
</LinearLayout>
```



## Extras

## Utils



```
package com.example.solobeast.Extras;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

import androidx.recyclerview.widget.RecyclerView;

import com.example.solobeast.Adapters.RecyclerViewFunctionalities;
import androidx.recyclerview.widget.RecyclerView;

/**
 * Utility class with various helper methods.
 * @author Ofek Almog
 */
public class Utils {

    /**
     * Checks if the device is connected to the internet.
     *
     * @param context the application or activity context.
     * @return true if the device is connected to the internet, false otherwise.
     */
    public static boolean isConnectedToInternet(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
        return activeNetworkInfo != null && activeNetworkInfo.isConnected();
    }

    /**
     * Method to check if the RecyclerViewFunctionalities interface is valid and the item
     * position is not RecyclerView.NO_POSITION.
     * @param recyclerViewFunctionalities The interface implementing the RecyclerView
     * functionalities.
     * @param adapterPos The position of the item in the list.
     * @return True if the interface is valid and the position is not RecyclerView.NO_POSITION,
     * false otherwise.
     */
    public static boolean checkInterfaceValid(RecyclerViewFunctionalities
recyclerViewFunctionalities, int adapterPos){
        if(recyclerViewFunctionalities != null){
            int pos = adapterPos;
            if(pos != RecyclerView.NO_POSITION){
                return true;
            }
        }
    }
}
```



```
        }  
    return false;  
}  
}
```



## Extras.Receivers

### AirplaneModeReceiver

```
package com.example.solobeast.Extras.Receivers;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class AirplaneModeReceiver extends BroadcastReceiver {
    public static boolean isAirplaneMode;
    @Override
    public void onReceive(Context context, Intent intent) {
        boolean isAirplaneModeOn = intent.getBooleanExtra("state", false);
        isAirplaneMode = isAirplaneModeOn;
        if (isAirplaneModeOn) {
            // Airplane mode is turned on
            Toast.makeText(context.getApplicationContext(),"Airplane is turned
on.",Toast.LENGTH_LONG).show();
        } else {
            // Airplane mode is turned off
            Toast.makeText(context.getApplicationContext(),"Airplane is turned
off.",Toast.LENGTH_LONG).show();
        }
    }
}
```



## Extras.Services

### CountdownThread



```
package com.example.solobeast.Extras.Services;

import android.util.Log;

/**
This class represents a thread that runs a countdown timer functionality.
It counts down from a specified time in seconds and stops when the timer reaches zero or is
stopped manually.
@author Ofek Almog
*/
public class CountdownThread extends Thread{

    /**
     The remaining time on the timer.
     */
    private int time;

    /**
     A boolean flag indicating whether the timer has been stopped.
     */
    private boolean stopped = false;

    /**
     The method that runs when the thread is started.
     It counts down from the specified time in seconds and stops when the timer reaches zero or
is stopped manually.
     */
    @Override
    public void run() {
        super.run();
        this.time = getTime();
        while(this.time > 0 && !stopped){
            try {
                Thread.sleep(1000);
                Log.d("services","timer : " + time);
                time--;
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        Log.i("services","TimerStopped:Success");
    }

    /**
     Returns the remaining time in seconds of the countdown timer.
     @return The remaining time in seconds of the countdown timer.
    
```



```
/*
public int getTime(){return this.time;}

/**
Sets the remaining time in seconds of the countdown timer.
@param time The remaining time in seconds of the countdown timer to set.
*/
public void setTime(int time){
    this.time = time;
}

/**
Stops the countdown timer manually.
*/
public void stopTimer() {
    this.stopped = true;
}
}
```

## Extras.Services

### TimerService



```
package com.example.solobeast.Extras.Services;

import android.app.Service;
import android.content.Intent;
import android.os.Build;
import android.os.IBinder;
import android.util.Log;

import com.example.solobeast.ui.Home.MainActivity;

/**
This class represents a service that runs in the background and implements a {@link
#timerThread} functionality.
@see CountdownThread
*/
public class TimerService extends Service {

    /** The thread that runs the countdown timer. */
    CountdownThread timerThread;

    /** The name of the service. */
    String serviceName = "TimerServices";

    /**
     Default constructor.
     */
    public TimerService() {
    }

    /**
     Called when the service is created.
     */
    @Override
    public void onCreate() {
        super.onCreate();
    }

    /**
     Called when the service is started.

     @param intent The intent that was used to start the service.
     @param flags Additional data about this start request.
     @param startId A unique integer representing this specific request to start.
     @return The return value indicates what semantics the system should use for the service's
current started state.
    */
}
```



```
    @Override
public int onStartCommand(Intent intent, int flags, int startId) {
    if(timerThread == null){
        timerThread = new CountdownThread();
        timerThread.setTime(intent.getIntExtra("task_time",0));
        timerThread.start();
    }
    Log.i("Value","STARTED");

    return START_NOT_STICKY;
}

/**
Called when a client explicitly starts the service by calling Context.startService(Intent).
@param intent The Intent supplied to Context.startService(Intent), as given.
@return IBinder interface for communicating with the service, or null if binding is not
supported.
*/
@Override
public IBinder onBind(Intent intent) {
    // TODO: Return the communication channel to the service.
    throw new UnsupportedOperationException("Not yet implemented");
}

/**
Called when the service is destroyed.
*/
@Override
public void onDestroy() {
    super.onDestroy();
    Log.i("Value","FINIHED");
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.N){
        Intent stopServiceIntent = new Intent(this, MainActivity.class);
        stopService(stopServiceIntent);
    }
    timerThread.stopTimer();
    timerThread = null;
}
}
```



## Adapters

### RecyclerViewFunctionalities

```
package com.example.solobeast.Adapters;

/**
An interface defining common functionalities for RecyclerView items.
*/
public interface RecyclerViewFunctionalities {

    /**
     Invoked when an item in the RecyclerView is clicked.
     @param position the position of the clicked item in the RecyclerView
     */
    void onItemClick(int position);

    /**
     Invoked when an item in the RecyclerView is long-clicked.
     @param position the position of the long-clicked item in the RecyclerView
     @return true if the long-click event is consumed and false otherwise
     */
    boolean onItemLongClick(int position);
}
```



## Adapters

### RewardAdapter



```
package com.example.solobeast.Adapters;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.provider.MediaStore;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;

import com.example.solobeast.Extras.Utils;
import com.example.solobeast.Objects.Reward;
import com.example.solobeast.R;
import com.example.solobeast.ui.Home.MainActivity;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.List;

/**
The RewardAdapter class is an adapter for the RecyclerView widget that handles a list of Reward objects.
It provides a view for each Reward item in the list, and it binds data from the Reward objects to the corresponding views.
It also listens to user interactions, such as clicking or long pressing an item in the list, and it triggers the appropriate actions.
@author Ofek Almog
*/
public class RewardAdapter extends RecyclerView.Adapter<RewardAdapter.RewardViewHolder>{

    /**
     An instance of the RecyclerViewFunctionalities interface that provides callbacks for the adapter's actions.
     */
    private final RecyclerViewFunctionalities recyclerViewFunctionalities;

    /**
     The context of the adapter, used for inflating the item views.
     */
    Context context;
```



```
/**  
 * The list of Reward objects that this adapter is handling.  
 */  
List<Reward> rewardList;  
  
/**  
 * Creates a new instance of the RewardAdapter class.  
 * @param context The context of the adapter.  
 * @param rewardList The list of Reward objects to be handled by the adapter.  
 * @param recyclerViewFunctionalities An instance of the RecyclerViewFunctionalities interface  
 * that provides callbacks for the adapter's actions.  
 */  
public RewardAdapter(Context context, List<Reward> rewardList, RecyclerViewFunctionalities  
recyclerViewFunctionalities){  
    this.context = context;  
    this.rewardList = rewardList;  
    this.recyclerViewFunctionalities = recyclerViewFunctionalities;  
}  
  
/**  
 * Creates a new RewardViewHolder instance by inflating the item view from the specified  
 * layout.  
 * @param parent The parent ViewGroup in which the item view will be contained.  
 * @param viewType The type of the view to be inflated.  
 * @return The new RewardViewHolder instance.  
 */  
@NonNull  
@Override  
public RewardAdapter.RewardViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int  
viewType) {  
    LayoutInflator inflater = LayoutInflator.from(context);  
  
    View view = inflater.inflate(R.layout.reward_item,null);  
    return new RewardViewHolder(view, recyclerViewFunctionalities);  
}  
  
/**  
 * Binds the data from the Reward object at the specified position in the list to the  
 * corresponding views in the holder.  
 * @param holder The RewardViewHolder that holds the views to be bound.  
 * @param position The position of the Reward object in the list.  
 */  
@Override  
public void onBindViewHolder(@NonNull RewardAdapter.RewardViewHolder holder, int position) {  
    Reward r = rewardList.get(position);  
}
```



```
        holder.RewardNameTv.setText(r.getRewardName());
        holder.RewardDescTv.setText(r.getDescription());
        holder.xpTv.setText("XP points " + r.getXP());
    }

    /**
     * Gets the number of items in the list.
     * @return The number of Reward objects in the list.
     */
    @Override
    public int getItemCount() {
        return rewardList.size();
    }

    /**
     * Returns the list of Reward objects handled by this adapter.
     * @return The list of Reward objects.
     */
    public List<Reward> getRewardList(){return this.rewardList;}

    /**
     * The RewardViewHolder class represents a view holder for the RecyclerView items in the
     * RewardAdapter.
     * It holds references to the views in the item layout, and it sets listeners for user
     * interactions.
     */
    class RewardViewHolder extends RecyclerView.ViewHolder{

        /**
         * The TextViews that displays the properties of the reward.
         */
        TextView RewardNameTv,RewardDescTv,xpTv;

        /**
         * The alert dialog used to confirm a reward purchase.
         */
        AlertDialog.Builder alertDialog;

        /**
         * Constructs a new RewardViewHolder object.
         * @param itemView The View object corresponding to the reward item layout.
         * @param recyclerViewFunctionalities The interface implementing the RecyclerView
         * functionalities.
         */
        public RewardViewHolder(@NonNull View itemView, RecyclerViewFunctionalities
recyclerViewFunctionalities) {
```



```
super(itemView);

//find all views by id in reward_item layout.
RewardNameTv = itemView.findViewById(R.id.RewardName);
RewardDescTv = itemView.findViewById(R.id.desc_item_holder);
xpTv = itemView.findViewById(R.id.xp_reward_needed);
AlertDialog = new AlertDialog.Builder(context);
FloatingActionButton startTaskFAB = itemView.findViewById(R.id.fab_buy_reward);

//Set listener for reward purchase button.
startTaskFAB.setOnClickListener(v -> {
    int pos = getBindingAdapterPosition();
    Reward r= getRewardList().get(pos);
    moveToPaymentScreen(r);
});

//Set listener for item click.
itemView.setOnClickListener(view -> {
    int pos = getBindingAdapterPosition();

if(Utils.checkInterfaceValid(recyclerViewFunctionalities,getBindingAdapterPosition()))
recyclerViewFunctionalities.onItemClick(pos);
});

//Set listener for item long click.
itemView.setOnLongClickListener(view -> {
    int pos = getBindingAdapterPosition();

if(Utils.checkInterfaceValid(recyclerViewFunctionalities,getBindingAdapterPosition()))
recyclerViewFunctionalities.onItemLongClick(pos);
    return true;
});

/***
 * Method to confirm a reward purchase and handle the transaction.
 * @param r The Reward object to be purchased.
 */
private void moveToPaymentScreen(Reward r) {

    alertDialog
        .setMessage("Are you sure you want to proceed with this reward payment?")
        .setCancelable(false)
        .setTitle("Buy a reward");

    alertDialog.setPositiveButton("Yes", (dialogInterface, i) -> {
```



```
        dialogInterface.cancel();
    if (r.getXP() <= MainActivity.current_user.getCurrentXP()){
        Toast.makeText(context,"You just bought the reward.
Congrats!",Toast.LENGTH_LONG).show();
        MainActivity.updateXPtoUserFirebase(r.getXP(),"DecreaseVal");
    }
    else {
        Toast.makeText(context,"You don't have enough xp.
😢",Toast.LENGTH_LONG).show();
    }

    if (context instanceof Activity) {
        ((Activity) context).finish();
    }
});
alertDialog.setNegativeButton("Cancel", (dialogInterface, i) -> {
    dialogInterface.cancel();
    Toast.makeText(context,"Event was cancelled
successfully",Toast.LENGTH_LONG).show();
});
AlertDialog alert = alertDialog.create();
alert.show();
}

}
```



## Adapters

### RewardAdapter : 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:id="@+id/reward_item"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view_reward"
        android:layout_width="match_parent"
        android:layout_height="250dp"
        android:layout_marginVertical="60dp"
        app:cardCornerRadius="11dp"
        app:cardElevation="11dp"
        app:cardUseCompatPadding="true">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            >

            <TextView
                android:id="@+id/RewardName"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="76dp"
                android:text="RewardName:"
                app:layout_constraintStart_toStartOf="parent"
                android:ellipsize="end"
                android:maxLines="1"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/desc_item_holder"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="20dp"
                android:text="Description:"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/RewardName" />

            <TextView
                android:id="@+id/xp_reward_needed"
```



```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="20dp"
        android:text="xp:"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/desc_item_holder" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="100dp"
    android:layout_height="match_parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:background="#5ed67c"
    >

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab_buy_reward"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:src="@drawable/baseline_payment"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.636"
        app:layout_constraintStart_toStartOf="parent"
        android:contentDescription="fab_buy_reward"/>
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>

</LinearLayout>
```



## Adapters

### TaskAdapter



```
package com.example.solobeast.Adapters;
import static com.example.solobeast.Objects.Task.*;

import com.example.solobeast.Extras.Utils;
import com.example.solobeast.Objects.Task;
import com.example.solobeast.R;
import com.example.solobeast.ui.TaskTimerAct;
import com.google.android.material.floatingactionbutton.FloatingActionButton;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

/**
The TaskAdapter class is an adapter for the RecyclerView widget that handles a list of Task objects.
It provides a view for each Task item in the list, and it binds data from the Task objects to the corresponding views.
It also listens to user interactions, such as clicking or long pressing an item in the list, and it triggers the appropriate actions.
@author Ofek Almog
*/
public class TaskAdapter extends RecyclerView.Adapter<TaskAdapter.TaskViewHolder>{

    /**
     An instance of the RecyclerViewFunctionalities interface that provides callbacks for the adapter's actions.
     */
    private final RecyclerViewFunctionalities recyclerViewFunctionalities;

    /**
     The context of the adapter, used for inflating the item views.
     */
    Context context;

    /**

```



```
The list of Task objects that this adapter is handling.  
*/  
List<Task> taskList;  
  
/**  
 * Creates a new instance of the TaskAdapter class.  
 * @param context The context of the adapter.  
 * @param taskList The list of Task objects to be handled by the adapter.  
 * @param recyclerViewFunctionalities An instance of the RecyclerViewFunctionalities interface  
 * that provides callbacks for the adapter's actions.  
 */  
public TaskAdapter(Context context, List<Task> taskList, RecyclerViewFunctionalities  
recyclerViewFunctionalities){  
    this.context = context;  
    this.taskList = taskList;  
    this.recyclerViewFunctionalities = recyclerViewFunctionalities;  
}  
  
/**  
 * Creates a new TaskViewHolder instance by inflating the item view from the specified layout.  
 * @param parent The parent ViewGroup in which the item view will be contained.  
 * @param viewType The type of the view to be inflated.  
 * @return The new RewardViewHolder instance.  
 */  
@NonNull  
@Override  
public TaskAdapter.TaskViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType)  
{  
    LayoutInflator inflater = LayoutInflator.from(context);  
    View view = inflater.inflate(R.layout.task_item,null);  
    return new TaskViewHolder(view, recyclerViewFunctionalities);  
}  
  
/**  
 * Binds the data from the Task object at the specified position in the list to the  
 * corresponding views in the holder.  
 * @param holder The RewardViewHolder that holds the views to be bound.  
 * @param position The position of the Reward object in the list.  
 */  
@Override  
public void onBindViewHolder(@NonNull TaskAdapter.TaskViewHolder holder, int position) {  
    Task task = taskList.get(position);  
  
    holder.taskNameTv.setText(task.getName());  
    holder.taskNameTv.setContentDescription("Task name is " + task.getName());  
    holder.difficultyTv.setText(task.getDifficulty());  
}
```



```
holder.constraintLayout.setBackgroundColor(Color.parseColor(changeColorAsTaskDifficulty(task.getDifficulty())));
    holder.timeTv.setText(timeRepresentation(task.getTime()));
    holder.timeTv.setContentDescription(task.getTime());
}

/**
 * Gets the number of items in the list.
 * @return The number of Task objects in the list.
 */
@Override
public int getItemCount() {
    return taskList.size();
}

/**
 * Returns the list of Task objects handled by this adapter.
 * @return The list of Task objects.
 */
public List<Task> getTaskList(){return this.taskList;}

/**
 * The TaskViewHolder class represents a view holder for the RecyclerView items in the
 * TaskAdapter.
 * It holds references to the views in the item layout, and it sets listeners for user
 * interactions.
 */
class TaskViewHolder extends RecyclerView.ViewHolder {

    /**
     * The TextViews that displays the properties of the task.
     */
    TextView taskNameTv, difficultyTv, timeTv;

    /**
     * The ConstraintLayout field represents the layout of the task item in the TaskAdapter's
     * RecyclerView.
     */
    ConstraintLayout constraintLayout;

    /**
     * Constructs a new TaskViewHolder object.
     *
     * @param itemView           The View object corresponding to the task item
     * layout.
     */
}
```



```
* @param recyclerViewFunctionalities The interface implementing the
RecyclerView functionalities.
*/
public TaskViewHolder(@NonNull View itemView, RecyclerViewFunctionalities
recyclerViewFunctionalities) {
    super(itemView);
    //find all views by id in task_item layout.
    taskNameTv = itemView.findViewById(R.id.taskName);
    difficultyTv = itemView.findViewById(R.id.diff);
    timeTv = itemView.findViewById(R.id.time);
    constraintLayout = itemView.findViewById(R.id.constraintLayout2);

    FloatingActionButton startTaskFAB = itemView.findViewById(R.id.fab_start_task);

    //Set listener for start task button.
    startTaskFAB.setOnClickListener(v -> {
        int pos = getBindingAdapterPosition();
        Task t = getTaskList().get(pos);
        moveToTimerScreen(t);
    });

    //Set listener for item click.
    itemView.setOnClickListener(view -> {
        int pos = getBindingAdapterPosition();
        if (Utils.checkInterfaceValid(recyclerViewFunctionalities,
getBindingAdapterPosition()))
            recyclerViewFunctionalities.onItemClick(pos);
    });

    //Set listener for item long click.
    itemView.setOnLongClickListener(view -> {
        int pos = getBindingAdapterPosition();
        if (Utils.checkInterfaceValid(recyclerViewFunctionalities,
getBindingAdapterPosition()))
            recyclerViewFunctionalities.onItemLongClick(pos);
        return true;
    });
}

/**
 * This method is used to move to the task timer screen for a given task.
 * It creates a new intent with the selected task's name, description, time, and
difficulty
 * as extras and starts the TaskTimerAct activity with that intent.
 *
```



```
* @param t The task to start the timer for.  
*/  
private void moveToTimerScreen(Task t) {  
    Intent i = new Intent(context, TaskTimerAct.class);  
    i.putExtra("selected_task_name", t.getName());  
    i.putExtra("selected_task_desc", t.getDescription());  
    i.putExtra("selected_task_time", t.getTime());  
    i.putExtra("selected_task_diff", t.getDifficulty());  
    context.startActivity(i);  
}  
}  
}
```



## Adapters

### TaskAdapter : 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:id="@+id/task_name"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:contentDescription="this is a task">

    <androidx.cardview.widget.CardView
        android:id="@+id/card_view"
        android:layout_width="match_parent"
        android:layout_height="110dp"
        app:cardCornerRadius="11dp"
        app:cardElevation="11dp"
        app:cardUseCompatPadding="true">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            >

            <TextView
                android:id="@+id/taskName"
                android:layout_width="120dp"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="8dp"
                android:ellipsize="end"
                android:maxLines="2"
                android:text="TaskName:"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/diff"
                android:visibility="gone"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginStart="44dp"
                android:layout_marginTop="12dp"
                android:drawableLeft="@drawable/baseline_local_fire_department_24"
                android:ellipsize="end"
                android:fontFamily="@font/bungee"
                android:maxLines="3"
                android:text="Difficulty:"/>

```



```
        app:layout_constraintStart_toEndOf="@+id/taskName"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="8dp"
        android:ellipsize="end"
        android:maxLines="3"
        android:text="Time:"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/taskName" />

<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/constraintLayout2"
    android:layout_width="65dp"
    android:layout_height="match_parent"
    android:background="@color/purple_200"
    app:layout_constraintEnd_toEndOf="parent">

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab_start_task"
        android:backgroundTint="@color/white"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:contentDescription="fab_start_task"
        android:src="@drawable/ic_baseline_start"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.636"
        app:layout_constraintStart_toStartOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>

</LinearLayout>
```



## Appendix C: UML diagram A3