

FitNote



אפליקציה למעקב אחרי פעילויות ספורטיביות

|  |  |
| --- | --- |
| מגיש: | אלון בן דוד |
| ת.ז.: | 213761943 |
| המנחה: | ניצן |
| חלופה: | תכנות טלפונים ניידים |
| תאריך הגשה: | 20/05/2021 |

תוכן העניינים

[מבוא 5](#_Toc72220037)

[מדריך למתכנת 7](#_Toc72220038)

[אלגוריתמים מרכזיים 7](#_Toc72220039)

[השימוש במחלקת PictureFileHelper כדי לשמור תמונה בטלפון: 7](#_Toc72220040)

[השימוש בפעולה onActivityResult כדי לקבל מידע מאפליקציה אחרת בטלפון: 9](#_Toc72220041)

[שימוש במתאמים להצגת נתונים ברשימות: 10](#_Toc72220042)

[אתגרים מרכזיים 11](#_Toc72220043)

[בדיקת התראות באפליקציה: 11](#_Toc72220044)

[שמירת תמונות בטלפון בקבצים: 11](#_Toc72220045)

[הגדרת התרשים במסך הטבלה: 11](#_Toc72220046)

[תהליך בניית בסיס הנתונים עם העוזר: 12](#_Toc72220047)

[תכן 13](#_Toc72220048)

[עץ מודולים 13](#_Toc72220049)

[תרשים Use Case 13](#_Toc72220050)

[תרשים מחלקות 14](#_Toc72220051)

[Activities 15](#_Toc72220052)

[דוגמא לשליחת הודעה ב-WhatsApp 36](#_Toc72220053)

[FeedbackActivity 42](#_Toc72220054)

[Services 43](#_Toc72220055)

[Content Providers 45](#_Toc72220056)

[Broadcast Receivers 48](#_Toc72220057)

[שיקולי תכן (design) 51](#_Toc72220058)

[Resources 54](#_Toc72220059)

[קבצים 59](#_Toc72220060)

[SQLiteDataBase 59](#_Toc72220061)

[קובץ info פנימי בזיכרון הטלפון 59](#_Toc72220062)

[סכמת Database 60](#_Toc72220063)

[מדריך משתמש 62](#_Toc72220064)

[מטרת המערכת 62](#_Toc72220065)

[יכולות המערכת 65](#_Toc72220066)

[תפעול המערכת 66](#_Toc72220067)

[תרשים זרימה בין המסכים 66](#_Toc72220068)

[הרשאות 67](#_Toc72220069)

[דרישות מיוחדות ומגבלות 67](#_Toc72220070)

[רפלקציה 68](#_Toc72220071)

[ביבליוגראפיה 69](#_Toc72220072)

[נספחים 70](#_Toc72220073)

[תיעוד – קוד הפרויקט 70](#_Toc72220074)

[עוזר בניית מוסד הנתונים (DataBaseHelper): 70](#_Toc72220075)

[דוגמא למתאם(Adapter): 92](#_Toc72220076)

[מחלקת עזר לשמירה ולקיחת קבצי תמונות (PictureFileHelper): 95](#_Toc72220077)

[מחלקה להשמעת התראה בעזרת BroadcastReceiver (ReminderBroadcast): 97](#_Toc72220078)

[מחלקה להשמעת מוזיקה והפעלת טיימר בעזרת Service, MediaPlayer ו-CountDownTimer (MusicAndTimerService): 99](#_Toc72220079)

[מסמכים נוספים 102](#_Toc72220080)

[כל המחלקות והקבצים בפרויקט 114](#_Toc72220081)

[AndroidManifest 114](#_Toc72220082)

[Java Folder Classes 116](#_Toc72220083)

[AddExerciseActivity 116](#_Toc72220084)

[Admin 121](#_Toc72220085)

[ContactListAdapter 122](#_Toc72220086)

[ContactModel 124](#_Toc72220087)

[DataBaseHelper 125](#_Toc72220088)

[EditUserActivity 147](#_Toc72220089)

[ExecuteExerciseActivity 154](#_Toc72220090)

[Exercise 158](#_Toc72220091)

[ExerciseAdaptor 160](#_Toc72220092)

[FeedbackActivity 161](#_Toc72220093)

[InformationActivity 164](#_Toc72220094)

[MainScreenActivity 167](#_Toc72220095)

[ManagerUsersActivity 170](#_Toc72220096)

[MusicAndTimerService 175](#_Toc72220097)

[PictureFileHelper 178](#_Toc72220098)

[ProgramUserActivity 180](#_Toc72220099)

[RegisterActivity 187](#_Toc72220100)

[ReminderBroadcast 195](#_Toc72220101)

[SettingsActivity 197](#_Toc72220102)

[ShareActivity 209](#_Toc72220103)

[Song 214](#_Toc72220104)

[StatisticsActivity 215](#_Toc72220105)

[User 223](#_Toc72220106)

[UserExercise 226](#_Toc72220107)

[UserExerciseDoneAdapter 228](#_Toc72220108)

[UserExerciseNotDonteAdapter 230](#_Toc72220109)

[UserExercisesInnerJoinEx 233](#_Toc72220110)

[UsersAdapter 235](#_Toc72220111)

[ViewExercisesResultsActivity 238](#_Toc72220112)

[WhatsAppSendActivity 242](#_Toc72220113)

[res Folder 248](#_Toc72220114)

[drawable 248](#_Toc72220115)

[layout 264](#_Toc72220116)

[menu 312](#_Toc72220117)

[mipmap 314](#_Toc72220118)

[ic\_launcer file 314](#_Toc72220119)

[ic\_launcer\_round file 314](#_Toc72220120)

[raw 314](#_Toc72220121)

[values 315](#_Toc72220122)

[Grable Scripts 317](#_Toc72220123)

# מבוא

**הרקע לפרויקט:**

במקרים רבים בזמן האימונים שאני עושה אני מרגיש את הצורך להשתמש באפליקציה פשוטה אחת ויחידה שתאפשר לי לעקוב אחרי הביצועים שלי במגוון תרגילי ספורט. היה לי רצון שהאפליקציה כזאת תסייע ותעודד את המשתמש להתקדם באימונים שלו. המשתמש יוכל בקלות להבין את מצבו ביחס לביצועים קודמים ולהבחין במגמות של שיפור/האטה באופן ברור. זאת דרך גרפים עמודות ופרמטרים הקשורים לתרגיל. במעבר על חנות האפליקציות הבנתי שהרעיון הזה יכול לסייע לי ולרבים. חלק גדול מהאפליקציות שראיתי בנושא ניהול אימוני הכושר מסובכות ועמוסות מדי במידע ובפרטים.

בפרויקט זה החלטתי ליצור מקום נוח גמיש ומסודר שבו ניתן לעקוב אחרי מגוון פעילויות הגופניות שונות. המטרה היא לכתוב אפליקציה להכנת אימון כושר יעיל ולמעקב אחרי האימון. האפליקציה תשמש ככלי לארגון מספר תרגילים למשתמש לבצע. המשתמש יוכל לבצע את התרגילים מתי שהוא רוצה, באיזה אופן שהוא חושב לנכון והוא יקבל התראות לחזור לאימון רק בזמנים שהוא עצמו קובע. למשתמש באפליקציה שליטה מוחלטת על אופן האימון שלו.

**תהליך המחקר :**

התנסות באפליקציות קיימות, וברחבי הרשת. צפייה בסרטונים ב-YouTube.

**סקירת המצב הקיים בשוק:**

נכון לעכשיו בשוק האפליקציות עמוסות במידע ורובם הגדול מתמקד בחיזוק שרירים שקל לראות כמו חזה, כתפיים ורגליים. אפליקציות כאלה, בעיניי, הופכות את האימונים להרבה פחות מענים מפני שברור למשתמש שביצועם הוא רק למען שיפור המראה החיצוני. בנוסף, עומס המידע, הכפתורים ואפשרויות עלול להפוך את השימוש באפליקציה ללא נוח בכלל. לכן, אני סבור שיש לשנות מצב זה באמצעות פיתוח אפליקציה חדשה ידידותית יותר.

**אילו חידושים יש בפרויקט:**

הפרויקט מאפשר למשתמש לנהל אימון פשוט, ברור וגמיש מאוד ללא תלות בדבר. זאת בדרך שהוא רוצה לנהל אותו. המשתמש יוכל בקלות לעקוב אחרי ביצועיו, לראות מה מצבו ולהבחין במגמות של שיפור/האטה באופן ברור. כל זאת למען מגוון מטרות שיש למשתמש ולא בהכרח כדי להיראות "יפה" יותר.

**אתגרים מרכזיים:**

אתגר מרכזי אתו נתקלתי הוא שמירת מידע על מגוון תרגילים שונים באופן כללי כך שאוכל להתייחס לכולם כאל תרגילים במחלקה אחת בלבד. למשל, שחייה נמדדת במטרים ושכיבות שמיכה בכמות הירידות והעליות. על מנת להתמודד עם אתגר זה הגדרתי מאפיינים כלליים איתם אפשר לתאר כל סוג של תרגיל כושר וכך יהיה פשוט יותר לשמור מידע על התרגילים במוסד הנתונים ולעקוב אחר אימון המשתמש. הגדרת הטבלאות עם הדרישות שהיו לי הייתה מאתגרת. היה צורך להפריד בין תרגילים יבשים עם מידע בסיסי כמו תמונה ותיאור של התרגיל ובין תרגילי משתמש בהם יש גם את שם המשתמש, את ה-ID של התרגיל ומידע על ביצוע התרגיל (תאריך, זמן ביצוע, דירוג ומספר החזרות).

**הסיבות לבחירת הנושא:**

בחרתי לכתוב אפליקציה בנושא אימוני כושר מפני שהייתי סבור שיש לפתח אפליקציה פשוטה ואינטראקטיבית למתאמנים רבים שתספק מידע רב בקלות. עם הכוח הרב שיש בהתעסקות בטבלאות בבסיס נתונים ב-SQL הייתה לי היכולת להחזיק בטווח הארוך מגוון רחב של מידע בכל מיני סוגי תרגילים. באופן זה משתמשים יוכלו להבחין במגמות של השתפרות/האטה באופן ברור ומקיף על אימונים שהם עשו במשך שנים. בנוסף, אני מאמין שאפליקציה בנושא אימוני כושר דורשת מגוון רחב של כלים שיעזרו למתאמן לנתח את אימונו. יישומון שכזה דורש שימוש במגוון כלים שיש ב-android studio כמו טבלאות, רשימות, טיימר, BroadCastReceiver ו-ContentProvider על מנת לנתח את כל המידע על האמונים בצורה המקיפה והברורה ביותר. בחירת נושא זה עוזרת לי ללמוד על הרבה כלים חדשים ושימושיים רק מבניית אפליקציה אחת.

**מוטיבציה לעבודה:**

מפני שאפליקציית אימוני כושר יכולה לכלול כל כך הרבה פיצ'רים, במהלך העבודה לא הרגשתי פעם אחת שחסר מה להוסיף לאפליקציה. אפשר לראות ולנתח את אותם תוצאות האימונים בכמה דרכים (טבלאות, רשימות). כמו במרבית אפליקציות הכושר הקיימות, ניתן גם לשתף אחרים בהישגי התרגילים שבוצעו. תהליך האימון אינו סופי ותמיד אפשר לחזור ולהתאמן. לכן, אני רואה באפליקציה זו ארגז חול גדול למימוש מגוון כלים בסיסים ומתקדמים של android studio, ומסיבה זאת, לא היה רגע אחד בו לא הייתה לי מוטיבציה להוסיף, להרחיב, ללמוד ולשפר את האפליקציה שלי.

**על איזה צורך הפרויקט עונה? איזה פתרון הפרויקט הזה בא לתת?**

הפרויקט מהווה תוכנה המאפשרת למשתמש להכין אימון כושר יעיל לעצמו ולעקוב אחר הביצועים שלו באמצעות ביצוע התרגילים ושמירת הנתונים על הביצוע בתוכנה. למשתמש יש אפשרות להוסיף תרגילים לרשימת התרגילים לביצוע. הוא יכול לבצע את התרגילים באימון בכל סדר שירצה. סדר ביצוע התרגילים באימון דינמי כדי למנוע הגבלה על המתאמן. למשל, אם המתאמן נמצא בחדר כושר וההליכון תפוס ע"י מתאמן אחר, הוא יוכל לבחור תרגיל אחר באימון שלא דורש שימוש במתקן זה.

המידע עליו מתבצע המעקב באפליקציה הוא רק על תרגילים שבוצעו באופן מלא בתהליך הביצוע הכולל.

המעקב על ביצוע התרגילים מתבטא במסך הגרפים בו מוצג גרף עמודות דינמי המייצג את הנתונים על ביצוע התרגיל: זמן שלקח ביצוע התרגיל, תאריך ביצועו, דירוג המשתמש לקושי הביצוע ומספר החזרות בתרגיל.

נוסף על כך התוכנה מכילה את נתוני המשתמש: גיל, גובה, משקל. כך היא מחשבת את תוצאת ה-BMI שלו ומציגה בפניו את מצבו הנוכחי.

# מדריך למתכנת

## אלגוריתמים מרכזיים

### השימוש במחלקת PictureFileHelper כדי לשמור תמונה בטלפון:

באפליקציה יצרתי מחלקה שמטרתה לסייע לי בשמירת תמונות משתמשים.

אני משתמש באפליקציה בקובץ שממומש על ידי SharedPreferences בקובץ info בטלפון לשמירת תמונת משתמשים באפליקציה.

במחלקה זו יש שלושה פעולות:

פועלה ראשונה:

public static void writeFileToInternalStorage(Context context, Bitmap bitmap, String filename)  
{  
 SharedPreferences sp= context.getSharedPreferences("info",0);  
 int counter=sp.getInt("counter", 0);  
 try {  
 FileOutputStream os = ((Activity)context).openFileOutput(filename+counter, Context.*MODE\_PRIVATE*);  
 *//Here compress 50%, store the compressed data in os.*

bitmap.compress(Bitmap.CompressFormat.*PNG*,100,os);  
 counter++;  
 SharedPreferences.Editor editor=sp.edit();  
 editor.putInt("counter",counter);  
 editor.commit();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
}

פועלה שמאפשרת כתיבת קובץ של תמונת משתמש אותה אני רוצה לשמור בקובץ info בטלפון.

בשמירה אני דוחס את התמונה לפורמט PNG ב-100%, ואת הנתונים אאחסן ב-os.

שם התמונה יהיה שם המתשמש שמגיע עם הקריאה לפעולה זו + מספר מונה.

באופן הזה, גם אם למשתמש יש מספר תמונות משתמש באפליקציה (לא אפשרי כי ברישום יכול להיכנס רק פעם אחת משתמש עם שם מסוים) תיבחר זאת האחרונה (בהמשך יהיה ברור למה).

פעולה שנייה:

public static Bitmap readFileFromInternalStorage(Context context,String filename)  
{  
 Bitmap b=null;  
 try {  
 InputStream in = ((Activity)context).openFileInput(filename);  
 b= BitmapFactory.*decodeStream*(in);  
 in.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return b;  
}

פעולה לקריאת הקובץ info בזיכרון הטלפון ולאחר מכן שליפת התמונה הרצויה. בעזרת ה-**InputSteam** אני ניגש לתמונה עם השם שקיבלנו filename.

כמו שהסברתי קודם, גם אם יש כמה תמונות עם אותו שם משתמש. לכל תמונה משויך מספר ייחודי המעיד על מיקומו בקובץ info וגם בקריאה לפעולה עם שם המשתמש יוחזר התמונה האחרונה בקובץ info.

הפעולה מחזירה תמונה שהיא מצאה.

פעולה שלישית:

public static Bitmap getPic(Context context, String name)  
{  
 File mydir = context.getFilesDir();  
 File lister = mydir.getAbsoluteFile();  
 Bitmap bitmap=null;  
  
 for (String list : lister.list())  
 {  
 if(list.toString().contains(name)) {  
 *//Toast.makeText(context, list, Toast.LENGTH\_LONG).show();* bitmap = *readFileFromInternalStorage*(context, list);  
  
 }  
 }  
  
 return bitmap;  
}

הפעולה הזאת משיגה גישה לקובץ המכיל את התמונות ועוברת על כל התמונות. תמונה עם שם המתשמש שקיבלנו בקריאה לפעולה תישמר ב-Bitmap. לבסוף תמונה זו תוחזר למשתמש.

בעזרת מחלקה זו כשאר המתשמש נרשם ומצלם את עצמו תמונתו נשמרת בקובץ info בטלפון תחת שמו ומספר מונה. כך שבכל מסך שאני רוצה להציג את תמונת המשתמש אני לוקח את שמו מבסיס הנתונים ומחפש בעזרת הפעולה getPic(Context context, String name) את תמונת המשתמש ומראה אותה במסך.

### השימוש בפעולה onActivityResult כדי לקבל מידע מאפליקציה אחרת בטלפון:

כשהמשתמש נרשם לאפליקציה או משנה את הפרטים שלו, הוא יכול לצלם תמונה שתיהיה תמונת המתשמש שלו. כדי לצלם תמונת משתמש התוכנה מבצעת מספר צעדים:

1. ליצור Intent שמטרתו היא לצלם תמונה, ואליו אני מציינים את הפעולה שהוא יבצע:

*//the intents intention is to capture an image  
//we have to specify the action for the intent*Intent imageTakeIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);

1. עכשיו אפשר לבדוק אם בטלפון אכן יש אלפיקציה כלשהי לצילום תמונה שיכולה לבצע את הפעולה הנדרשת לכך. אם אין כזאת פעולה צילום התמונה לא יתרחש:

*//now we are able to check if there is any application  
//capable of handling this intent  
//otherwise your'e application will crush*if(imageTakeIntent.resolveActivity(getPackageManager()) != null)  
{  
 *//we have to user startActivityForResult method  
 //second parameter is the requestCode* startActivityForResult(imageTakeIntent, *REQUEST\_IMAGE\_CAPTURE*);  
}

1. אם יש אלפיקציה כזאת אני אקרא ל-onActivityResult שתחכה לתוצאה שתגיע מאפליקציה שיכולה לצלם. ב-AndroidManifest של האפליקציה יש uses-feature המתייחס לצילום התמונה:

<uses-feature  
 android:name="android.hardware.Camera"  
 android:required="true" />

1. בקבלת התוצאה מהאפליקציה בטלפון לביצוע הצילום, מתבצעת בדיקה נוספת של קוד הבקשה ובדיקת נכונות התוצאה. לאחר מכן נשמר ב-Bitmap התמונה שצולמה.

בעזרת הפעולה saveToInternalStorage(String username)

הנעזרת בפעולה PictureFileHelper.writeFileToInternalStorage עליה דיברתי קודם במחלקה PictureFileHelper האפליקציה שומרת את התמונה בקובץ info בטלפון.

*//in order to receive the result from the other application we  
//have to override a method called onActivityResult*@Override  
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
  
 *//we have to check if the requestCode is the same  
 //AND if the requestCode is ok* super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *REQUEST\_IMAGE\_CAPTURE* && resultCode == *RESULT\_OK*) {  
 *//here we have the data available on this intent  
 //now we can retrieve the data* Bundle extras = data.getExtras();  
 Bitmap imageBitmap = (Bitmap) extras.get("data");  
  
 *//now we can display the image on the image view* imgProfilePic.setImageBitmap(imageBitmap);  
  
 *//getting the image to save* bitmapToSave = imageBitmap;  
 }}

### שימוש במתאמים להצגת נתונים ברשימות:

באלפיקציה מימשתי מחלקות רבות למתאמים היורשים מהמתאם הבסיסי BaseAdapter על מנת להתאים את כל הרשימות לצורות שאני רוצה להציג בהם את הנתונים.

המחלקות האלו הם:

* ContactListAdapter להצגת אנשי קשר מסוג ContactModel ברשימה SearchView במסך ShareActivity.
* UsersAdapter להצגת משתמשים מסוג User ברשימה SearchView במסך ShareActivity.
* ExerciseAdaptor להצגת תרגילים מסוג Exercise ברשימה ListView במסך InformationActivity.
* UserExerciseDoneAdapter להצגת תוצאות תרגילים שבוצעו מסוג UserExercise ברשימה ListView במסך ViewExercisesResultsActivity.
* UserExerciseNotDoneAdapter להצגת תרגילי משתמש שלא בוצעו מסוג UserExercise ברשימה ListView במסך ProgramUserActivity.

תהליך יצירת מתאם:

1. כתיבת ה-XML המתאים לעיצוב האיבר ברשימה
2. יצירת מחלקה היורשת מ-BaseAdapter
3. יצירת המתאם במחלקה בה נמצאת הרשימה
4. שילוב הרשימה עם הנתונים והמתאם שיתאים את הנתונים לרשימה

בעזרת מחלקות אלו אני מנפח לכל איבר ברשימה עיצוב משלו המייצג באופן ברור ומקיף את כל המידע הרלוונטי לסוג נתון זה. כך המתשמש יכול להבין מהם התרגילים באלפיקציה, מהם הנתונים שלו ומה תוצאות הביצועים שלו בתרגילים שביצע.

## אתגרים מרכזיים

### בדיקת התראות באפליקציה:

באפליקציה במסך התרגילים של המתשמש (ProgramUserActivity) המשתמש יכול להפעיל התראה מתוזמנן לזמן שהוא רוצה שהיא תפעל. התראה זו תעודד את המתשמש לחזור להתאמן ותיקח אותו חזרה לאפליקציה. בעת הפעלת ההתראה לא היה ברור ממעבר על מסך ה-RUN אם ההתראה עומדת לפעול או לא בהצבת זמנים מסוימים. לאחר בדיקה על נושא ההתראות באינטרנט הבנתי שעלי לבדוק את תקינות התאריך ולוודא שהתאריך שאני משתמש בו ליצירת ההתראה דרך ReminderBroadcast הוא תקין והוא לאל לפני כרגע. אם התאריך הוא לפני עכשיו התוכנה לא תיצור את ה-AlarmManager ליצירת ההתראה בזמן הניתן ותישלח הודעה מתאימה במקום. אם הזמן תקין והוא אחרי עכשיו ה-AlarmManager יבנה.

*//checking that the time given is not before now*if (alarmStartTime <= System.*currentTimeMillis*()){  
  
 Toast.*makeText*(this, "This time is before now! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
  
}else{  
 *// Set Alarm  
 //AlarmManager.RTC\_WAKEUP - wakes up the device to fire the pending intent  
 //at the specified time* alarmManager.set(AlarmManager.*RTC\_WAKEUP*, alarmStartTime, pendingIntent);  
  
 Toast.*makeText*(this, "Set Done! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
}

### שמירת תמונות בטלפון בקבצים:

**לא ניתן לשמור תמונות ב-drawable ברגע שאני רוצה לשמור תמונות משתמשים**.

לאחר חקירה של פתורנות אפשריים גילית שאני יכול לעזר במחלקה נפרדת שתשמור את התמונה המצולמת בטלפון בקובץ בתוך הטלפון עצמו מחוץ לאפליקציה עצמה.

לאחר קליטת התמונה שצולמה באפליקציה נפרדת ב-onActivityResult, אני נעזר במחלקה PictureFileHelper לשמור את תמונת המתשמש עם שמו ומספר מונה. כך נשמר ביזכרון הטלפון בקובץ info תמונת המתשמש. אם נרצה לגשת לתמונה זו נעזר שוב במחלקה PictureFileHelper כאשר השם שנחפש הוא שם המתשמש מפני שתמונה נשמרת עם שם המתשמש.

### הגדרת התרשים במסך הטבלה:

הגדרת הצירים של הטבלה במסך הטבלה (StatisticsActivity) הייתה מאתגרת.

כדי להראות את התאריך של ביצוע התרגיל עם נתונים נוספים עבור כל תרגיל בנפרד לא יכולתי להתייחס לתאריך המלא בציר האיקס מפני שאז התרשים היה ארוך מדי.

לכן, כדי לפתור בעיה זו הגדרתי שני ספינרים לסינון המידע בתרשים: אחד לבחירת סוג התרגיל ואחד לבחירת השנה.

באופן זה מיקום בציר ה-x עבור כל איבר בתרשים מייצג את החודש בו בוצע התרגיל ומיקום בציר ה-y מציין את מספר החזרות בביצוע התרגיל.

### תהליך בניית בסיס הנתונים עם העוזר:

הדרישות לאפליקציה שלי דרשו בנייה של מספר טבלאות בבסיס הנתונים. התהליך של בניית בסיס הנתונים המתאים ביותר לאפליקציה שלי ולרעיונותי היה מאתגר.

לאחר מעבר על דרכי שמירת המדיע מצעתי שהשימוש במחלקה DataBaseHelper היורשת מ-SQLiteOpenHelper היא הדרך היעילה ביותר לשמירת מידע וניהולו.

כדי לשמור את המשתמשים של האפליקציה יצרתי טבלה בשם Users השומרת את פרטי המשתמשים.

כדי לשמור את התרגילים של האפליקציה יצרתי טבלה בשם Exercises השומרת את פרטי התרגילים.

כדי לשמור את תרגילים המתמשים יצרתי טבלה בשם UserExercises השומרת את שם המשתמש, ID של התרגיל שבוצע ונתונים נוספים על ביצוע התרגיל (במידה והתרגיל בוצע).

כדי לשמור את השירים של האפליקציה יצרתי טבלה בשם Songs השומרת את פרטי השירים שאני משמיע בעת ביצוע תרגילים במסך ביצוע התרגילים (ExecuteExxerciseActivity).

## תכן

## עץ מודולים

## תרשים Use Case

Register

EditUsers

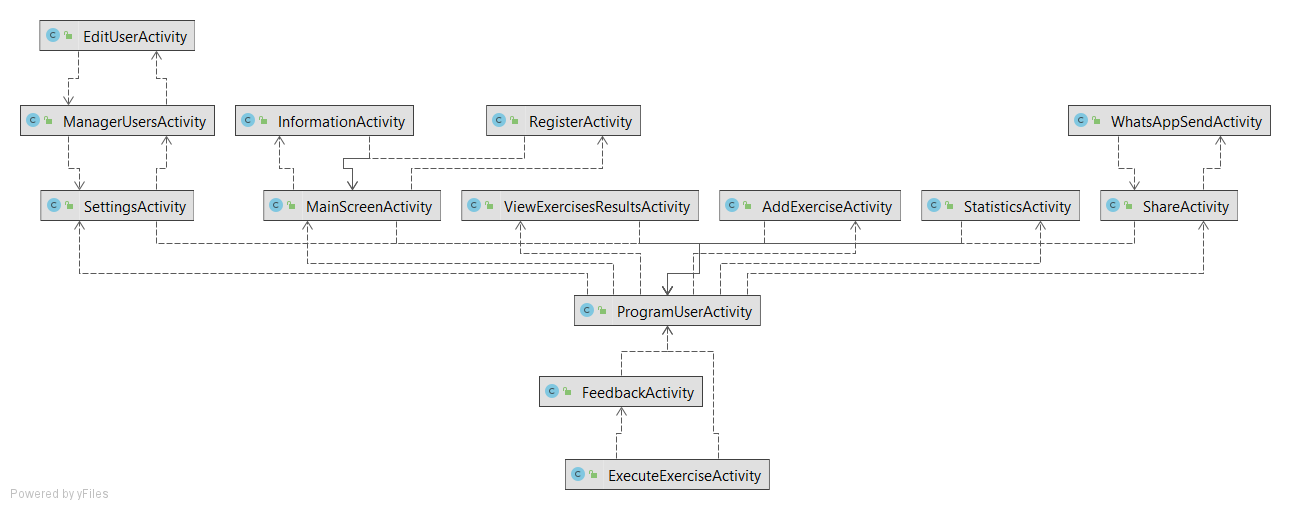
משתמש

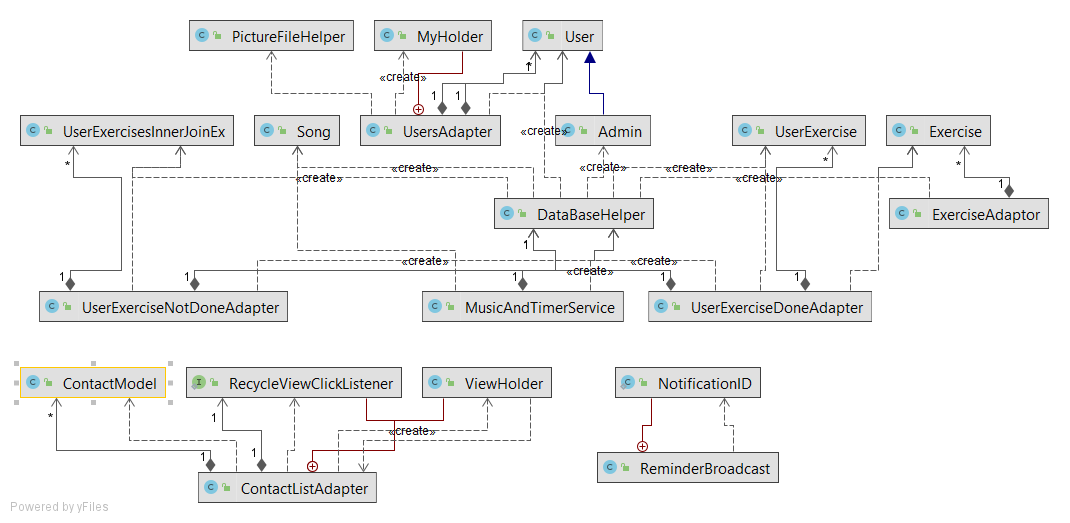
מנהל

UserExercises

Exercises

## תרשים מחלקות

****תרשים UML של מסכים (Activitys):

תרשים UMLשל שאר המחלקות באפליקציה (Adapter-ים, Service, BroadcastReciever ועוד):

### Activities

#### MainScreenActivity

**מסך הפתיחה**: במסך זה מתממש המסך הראשון והראשי של האפליקציה.

לחיצה על כפתור הרישום מעבירה את המשתמש למסך הרישום **RegisterActivity**.

לחיצה על כפתור הכניסה מעבירה את המשתמש למסך התרגילים של המשתמש **ProgramUserActivity**.

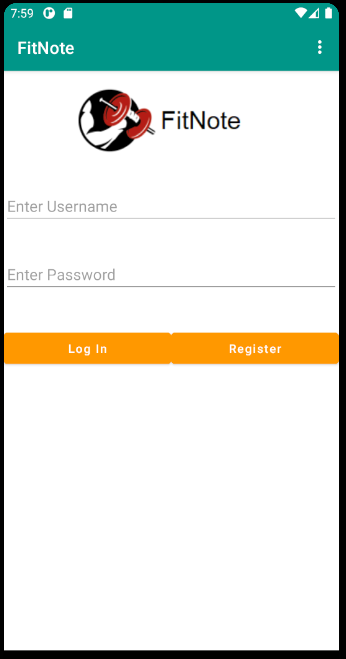
כולל את הרכיבים הבאים:

**EditText, Button, ImageView, Menu, Dialog**

צילום מסך ראשי:

**Menu**

תפריט של מסך ראשי



**ImageView**

לוגו של האפליקציה FITNOTE

**Button**

לקליטת לחיצת משתמש:

כפתור אחד למסך הרישום והשני לכניסה לאפליקציה כמשתמש בעל הנתונים שהוכנסו

**EditText**

לקליטת פרטי משתמש:

לקליטת שם משתמש וסיסמא

תפריט של מסך ראשי כולל אפשרות ללכת למסך מידע על האפליקציה ואפשרות לצאת מהאפלציה דרך מענה על Dialog של הסכמה לצאת מהאפליקציה.

צילום מסך ראשי עם הופעה של התפריט וה-dialog:

#### C:\Users\d-830\AppData\Local\Microsoft\Windows\INetCache\Content.Word\MainActivity_menu1.pngC:\Users\d-830\Desktop\Screens\MainActivity_menu2_exit_dialog.PNG

**Menu**

**Item 1**

about:

אופציה למעבר למסך המידע על האפליקציה **InformationActivity**

**Dialog**

לאפשרות יציאה מהאפליקציה: דרך אישור היציאה ב-OK או ביטול הפעולה ב-CANCEL

**Menu**

**Item 2**

exit:

אופציה ליציאה מהאפליקציה

#### InformationActivity

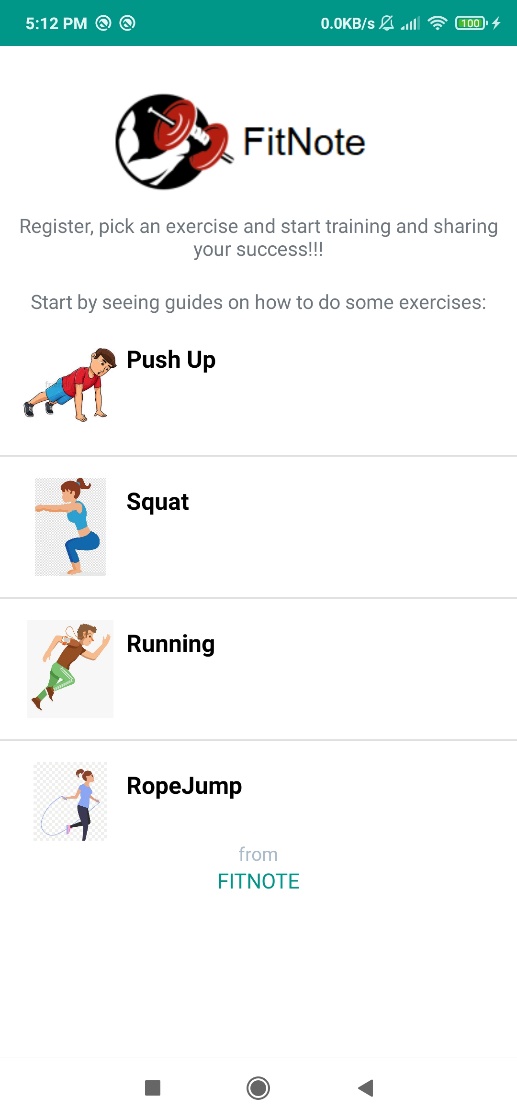
**מסך המידע**: מסך זה כולל הסבר קצר על מה אפשר לעשות באפליקציה.

ברשימה למטה (**ListView**) מופיעים כל התרגילים שיש באפליקציה כך שגם משתמשים שרק התקינו את האפליקציה יוכלו ללחוץ על כל תרגיל ולהבין איזה תרגילים קיימים בתוכנה זו.

כולל את הרכיבים הבאים:

**ImageView, TextView, ListView**

צילום של מסך המידע:



**ImageView**

לוגו של האפליקציה FITNOTE

**TextView**

לכותרת קטנה למטה עם שם האפליקציה

**ListView**

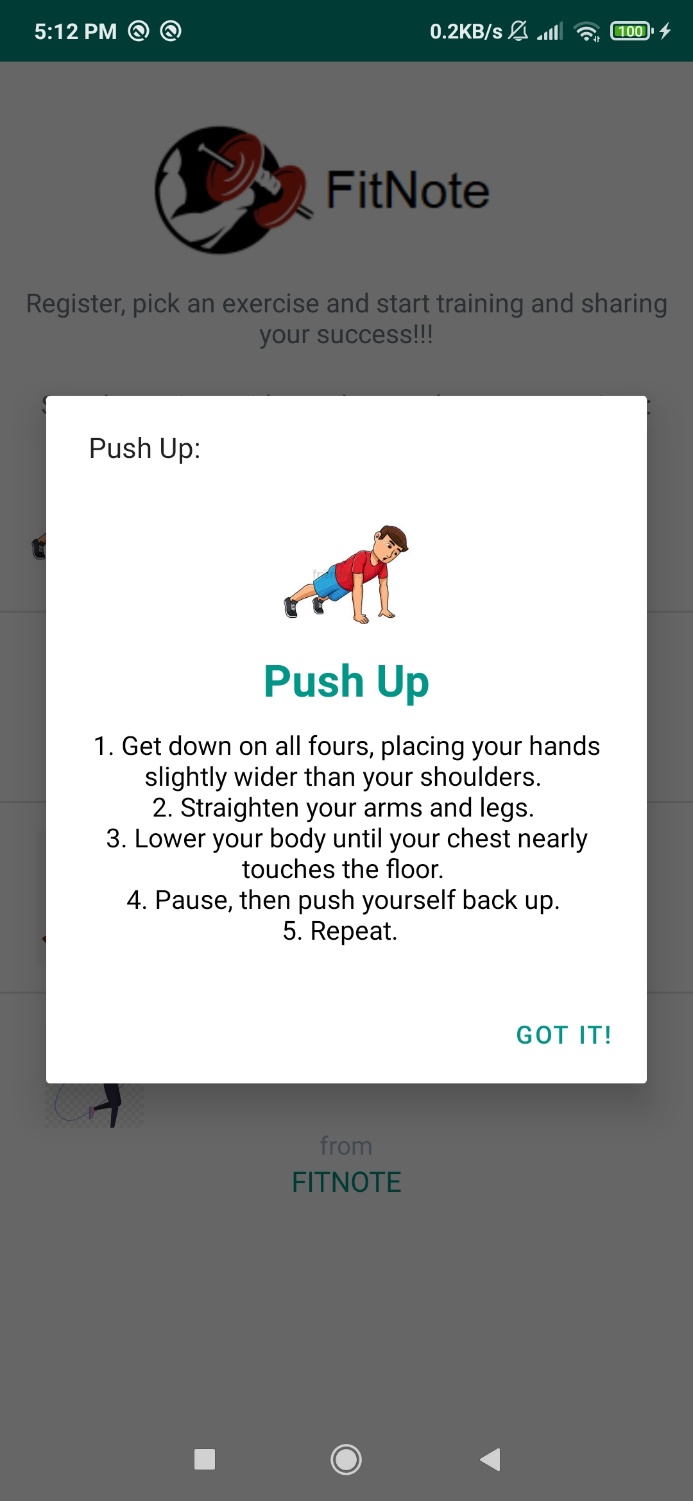
רשימה של כל סוגי התרגילים (Exercises):

הרשימה נעזרת במתאם **Adapter** מיוחד הנקרא **ExerciseAdaptor**

**TextView**

להסבר קצר על האפליקציה

צילום מסך המידע עם הופעה של ה-dialog:



**AlertDialog**

כולל את הרכיבים הבאים:

**ImageView, TextView**

ב-GOT IT! מבטלים את תיבת הדו שיח

**AlertDialog**

כולל את הרכיבים הבאים:

**ImageView, TextView**

ב-YES מאשרים את הוספת התרגיל וב-CANCEL מבטלים את הוספת התרגיל ונשארים באותו מסך

**AlertDialog**

כולל את הרכיבים הבאים:

**ImageView, TextView**

ב-YES מאשרים את הוספת התרגיל וב-CANCEL מבטלים את הוספת התרגיל ונשארים באותו מסך

#### RegisterActivity

**מסך הרישום**: במסך זה מתממש תהליך הרישום של האפליקציה למשתמשים חדשים.

**בתהליך הרישום נמנעת האפשרות להירשם עם אותו שם משתמש יותר מפעם אחת!**

לחיצה על אחד הכפתורים למטה מחזירה את המשתמש למסך הכניסה **MainScreenActivity**.

כולל את הרכיבים הבאים:

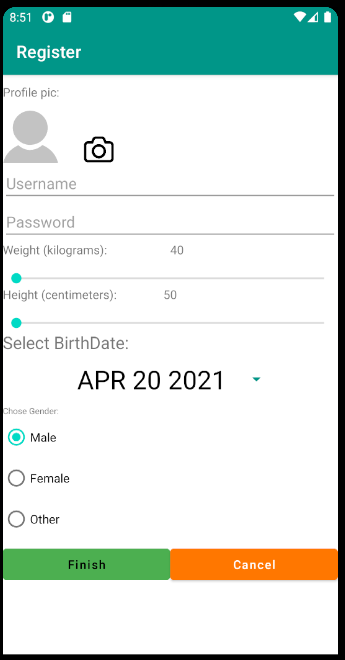
**ImageView**

תמונה של משתמש:

מראה את תמונת המשתמש לאחר קליטת התמונה

**ImageView, Button, EditText, SeekBar, TextView, RadioGroup, RadioButton**

צילום מסך ראשי:



**ImageView**

תמונה של מצלמה: לחיצה עליה פותחת פעולה של לקיחת תמונה עם המצלמה בטלפון.

אם אין אישור לגשת למצלמה לא תהיה אפשרות לצלם בטלפון,

ובהכנסת פרטי המשתמש יוכנס תמונת הפרופיל הברירת מחדל. אם המשתמש כן צילם את עצמו תוכנס לבסיס הנתונים התמונה שהוא צילם

**RadioGroup**

שמאגד כמה **RadioButton-ים**:

כפתורים אלו קולטים את מין המשתמש

**TextView**

להסבר איזה מידע לוקחים בכפתור

**TextView**

לתיאור המידע הנקלט מה-**SeekBar**

**SeekBar**

לקליטת משקל וגובה של המשתמש

**Button**

כפתור לקליטת לחיצה של המשתמש:

בלחיצה מופיעה ה-**DatePickerDialog**

שהוא לוקח את תאריך לידתו של המשתמש.

התאריך מועבר מה-**DatePicker** לטקסט בכפתור ואז מהטקסט נקלח המידע לבסיס הנתונים

**Button**

כפתורים לסיום הרישום וכפתור לביטול הרישום: בלחיצה על שניהם חוזרים למסך הראשי

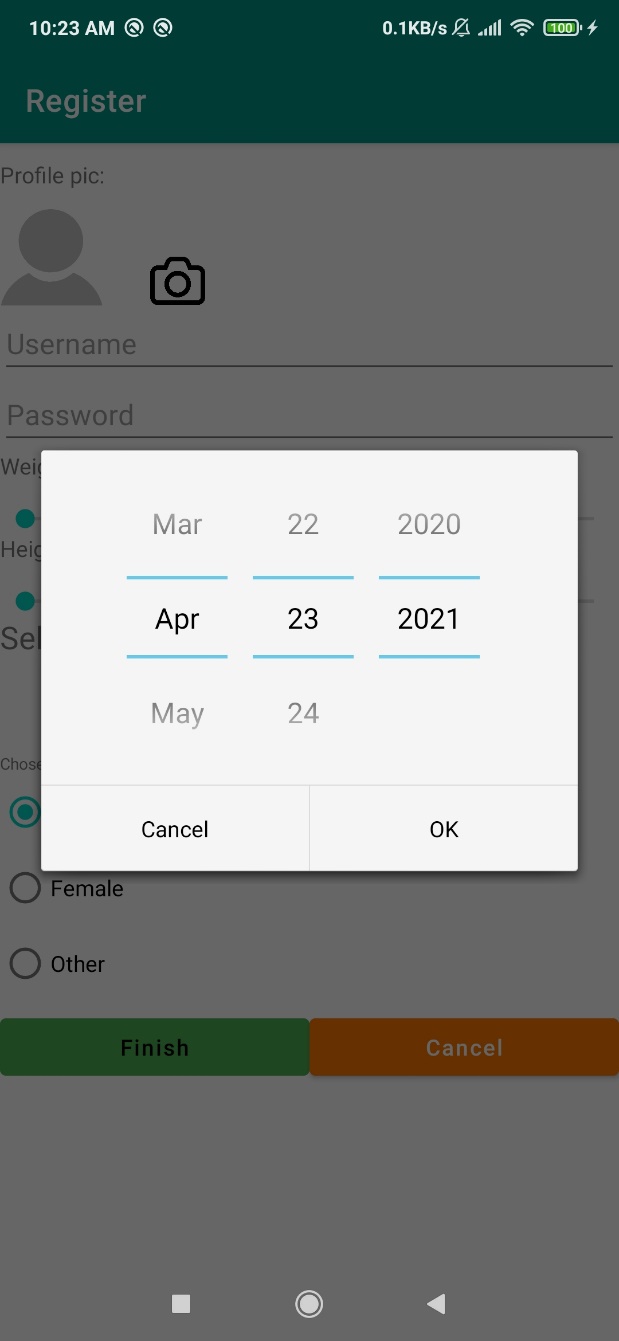
**EditText**

לקליטת פרטי משתמש חדש:

לקליטת שם משתמש וסיסמא

בלחיצה על כפתור התאריך במסך הקודם מופיע ה-**DatePickerDialog** שהוא לוקח את תאריך לידתו של המשתמש.

צילום של ה- **DatePickerDialog**:



**DatePickerDialog**

לבחירת תאריך הלידה החדש:

OK לאישור תאריך זה. אז התאריך יועבר לכפתור (**Button**) ב-- **AlertDialog**.

CANCEL לביטול בחירת התאריך וחזרה ל- **AlertDialog**.

#### ProgramUserActivity

**מסך התרגילים של המשתמש**: במסך זה מופיעים כל התרגילים (UserExercises) של המשתמש אותם הוא עוד לא ביצע.

למעלה בתפריט מופיעים ארבעה אייקונים למסכים שונים. יש טקסט קבלת פנים למשתמש הנכנס ומוצג תמונת הפרופיל של המשתמש. ברשימה יופיעו התרגילים (UserExercises)אותם המשתמש עוד לא ביצע אך הוסיף אותם לרשימת התרגילים שהוא יבצע (כלומר, אלו תרגילים בלי תוצאות). למטה יש אפשרות להוסיף עוד תרגילים ואפשרות להכין התראה לזמן מסוים שתעודד את המשתמש לחזור להתאמן.

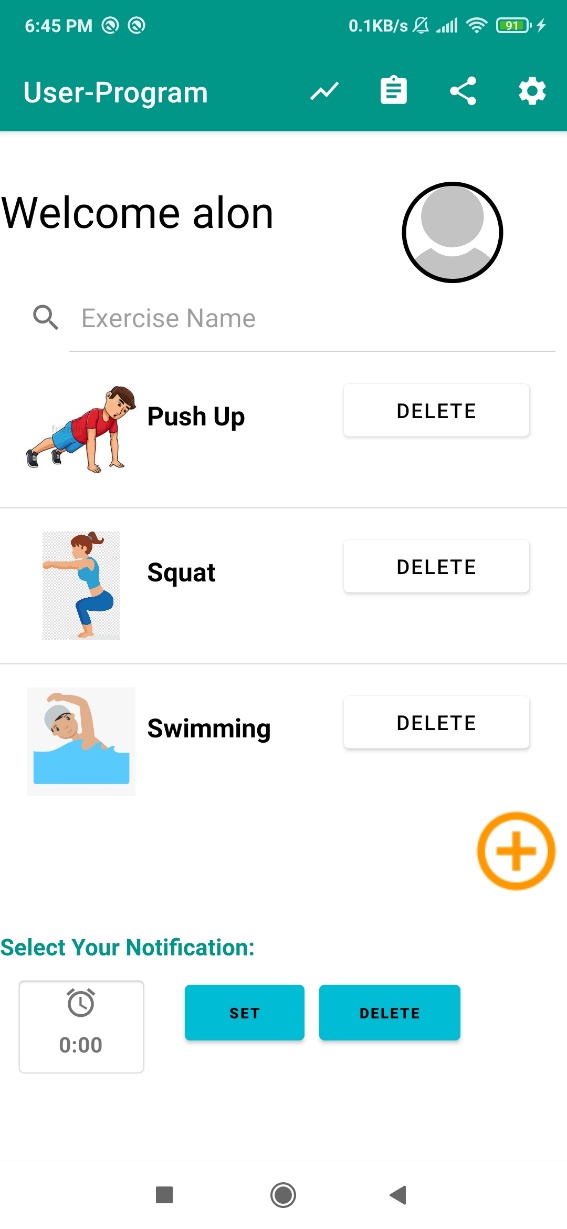
לחיצה על כפתור הפלוס לוקחת את המשתמש למסך הוספת התרגילים **AddExerciseActivity**.

כולל את הרכיבים הבאים:

ספרייה של CircleImageView: <https://github.com/hdodenhof/CircleImageView>

**TextView, Button, ImageView, CircleImageView (from a library on GitHub), ListView, Menu, SearchView**

צילום של מסך התרגילים של המשתמש:



**SearchView**

איבר לכתיבת טקסט: לפיו מסננים תרגילים ברשימת תרגילי המשתמש

**CircleImageView**

תמונה של משתמש:

מראה את תמונת המשתמש

**Menu**

תפריט של מסך התרגילים של המשתמש

**TextView**

טקסט קבלת פנים עם שם המשתמש שנכנס

**ListView**

רשימה של **כל תרגילי המשתמש שעדין לא בוצעו**(UserExercise):

הרשימה נעזרת במתאם **Adapter** מיוחד הנקרא **UserExerciseNotDoneAdaptor**

**TextView**

עם תמונה של שעון שבלחיצה **עליו מופיעה TimePickerDialog:**

בעת בחירת הזמן יתעדכן הטקסט להראות את הזמן שנבחר בפורמט "h:mm a"

**Button**

כפתורים להגדרה (SET) וביטול (CANCEL) של AlarmManager:

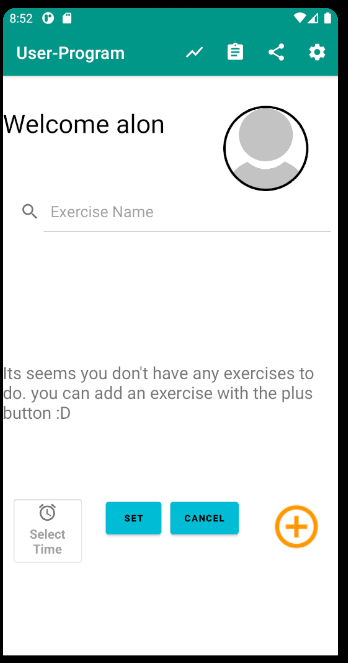
כשיגיע הזמן שהגדיר המשתמש להפעלת ההתראה, יאזין ה- ReminderBroadcast ל-Intent עם ההודעה של המשתמש להתראה ושמו

**Button**

כפתור למעבר למסך AddExerciseActivity

* אם למשתמש אין תרגילים לביצוע או שנכנס משתמש חדש, רשימת התרגילים תהיה ריקה ובמקום זאת יופיע טקסט שיסביר איך להוסיף תרגילים לביצוע. כלומר, הטקסט יופיע (יהיה visible) אם ה-**ListView** יהיה ריק, והרשימה תהיה ריקה אם ה-**DataBaseHelper** לא ימצא אף תרגיל שהמשתמש לא ביצע.

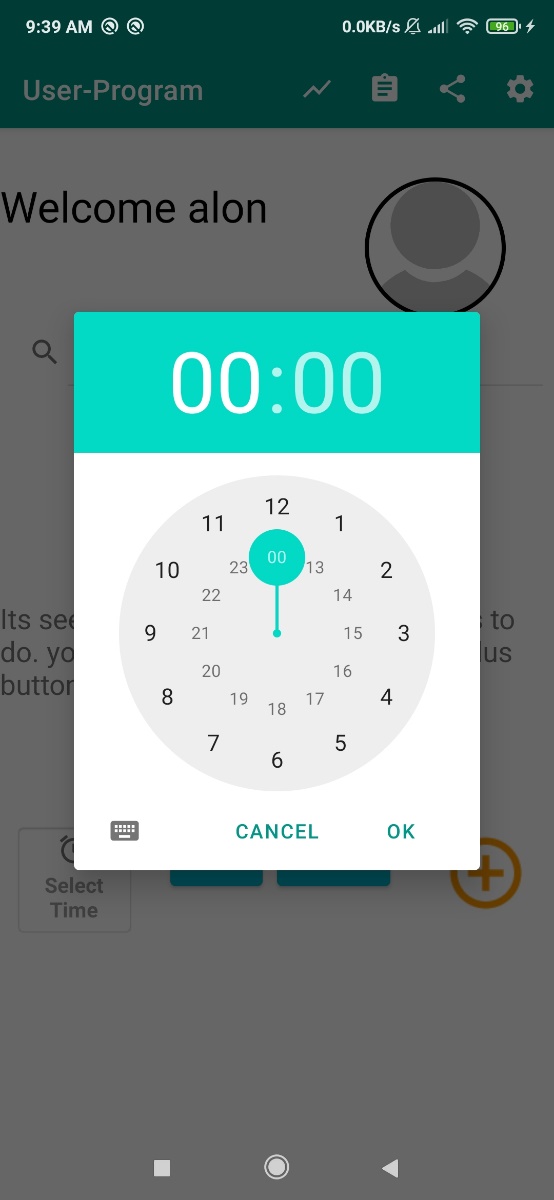
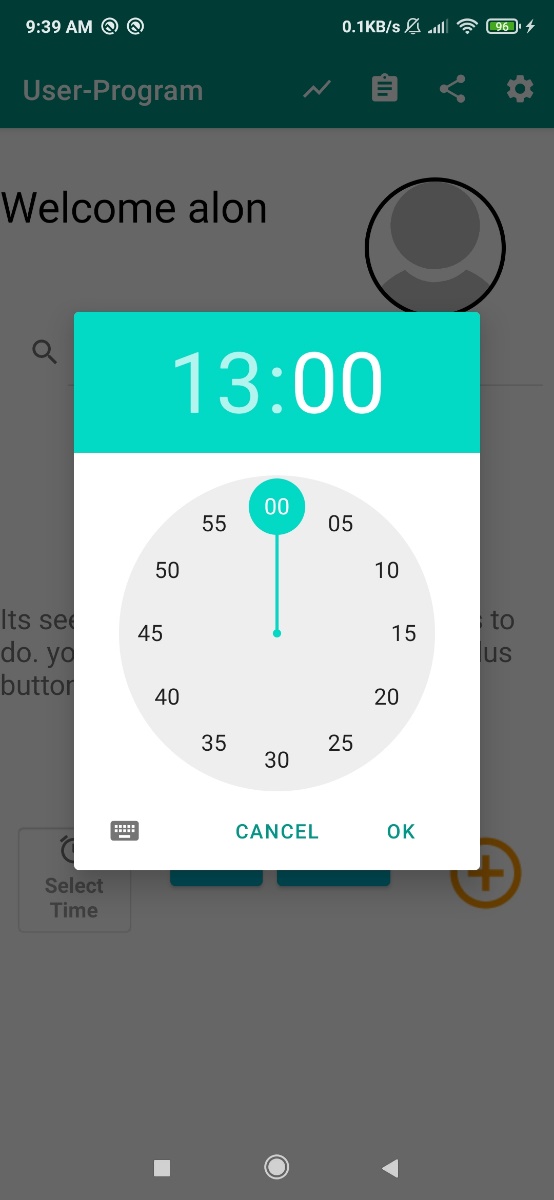
צילום של מסך התרגילים של המשתמש כאשר אין תרגילים ברשימה:



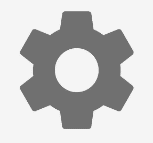
* אם המשתמש לוחץ על ה-**TextView** מלטה עם תמונת השעון, מופיעה **TimePickerDialog** בו אפשר לבחור את הזמן (שעות ודקות) להופעה של **Notification** דרך המחלקה **ReminderBroadcast**. לאחר בחירת הזמן ב-**TimePickerDialog**, הוא יופיע ב- **TextView**.

בלחיצה על כפתור ההגדרה (SET) יבנה ה-**AlarmManager** בו יש **PendingIntent** לזמן שהגדיר המשתמש ב- **TextView**.

צילום ש ה- **TimePickerDialog**:



* התפריט של מסך התרגילים של המשתמש כולל מספר אפשרויות:

1. לעבור למסך ההגדרות **SettingsActivity** בו המשתמש יכול לשנות את כל הפרטים שלו חוץ משמו.
2. לעבור למסך שיתוף המידע על תוצאות התרגילים **ShareActivity** לאיש קשר לבחירת המשתמש.
3. לעבור למסך **ViewExercisesResultsActivity** בו המשתמש יכול לראות את תוצאות התרגילים שביצע ברשימה.
4. לעבור למסך הטבלה **StatisticsActivity** בו המשתמש יכול לראות את תוצאות התרגילים בטבלה אחת עם מספר פילטרים.

* כל איבר ברשימה נבנה לפי העיצוב (layout) הבא:

לחיצה על איבר ברשימה מעביר את המשתמש למסך ביצוע התרגילים **ExecuteExerciseActivity**.

העיצוב כולל את הרכיבים הבאים:

**ImageView, TextView, Button**

**User\_exercise\_layout**:



**Button**

כפתור למחיקת התרגיל:

לחיצה על כפתור זה מוחקת את התרגיל הזה מבסיס הנתונים בעזרת ה-ID של ה-UserExercise

**ImageView**

תמונת התרגיל:

מראה תמונה של כל UserExercise שלא בוצע עדין ברשימת **ListView**

**TextView**

טקסט הכולל את שם התרגיל

#### SettingsActivity

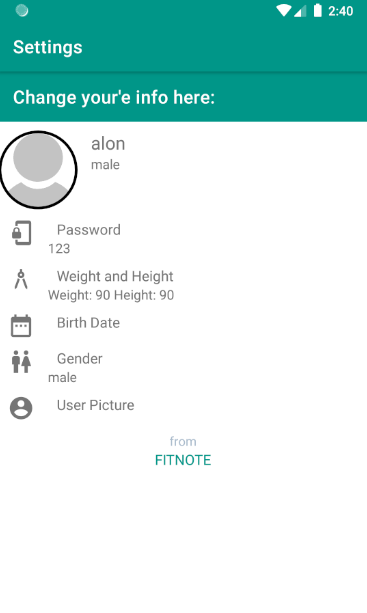
**מסך ההגדרות**: במסך זה המשתמש יכול לשנות את כל פרטי המשתמש שלו (סיסמה, משקל, גובה, תאריך לידה, מין ותמונת פרופיל) אך לא את שמו מפני ששמו הוא מפתח בבסיס הנתונים למציאת משתמש (**בתהליך הרישום נמנעת האפשרות להירשם עם אותו שם משתמש יותר מפעם אחת!**). בלחיצה על כל אחד מה-LinearLayout-ים מופיעה אפשרות לשינוי נתון מסוים על המשתמש ב-**AlertDialog**. ב-**AlertDialog** מופיע הפריסה (layout) שמאפשר את שינוי פרטי המשתמש.

ספרייה של CircleImageView: <https://github.com/hdodenhof/CircleImageView>

כולל את הרכיבים הבאים:

**Toolbar, ScrollView, CircleImageView (from a library on GitHub), TextView, View, ImageView**

צילום של מסך ההגדרות אם המשתמש הנכנס הוא רגיל (לא מנהל ולא סופר מנהל):

****

**Toolbar**

משמש ככותרת להסבר המסך

**CircleImageView**

תמונה של משתמש:

מראה את תמונת המשתמש

**ImageView**

להראות את **סוג המידע** בתמונה

**TextView**

טקסט לתיאור **שם המשתמש ומינו**

**TextView**

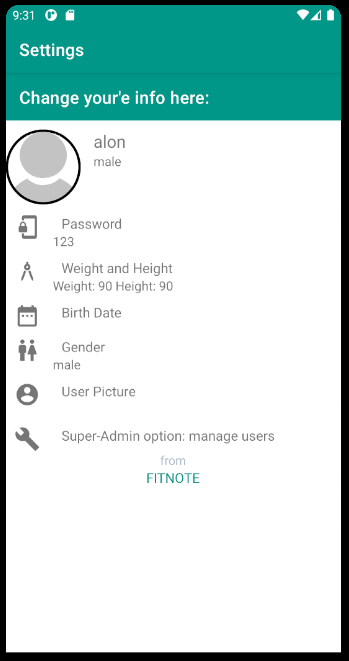
**טקסט חותמת** לשם האפליקציה

**TextView**

טקסט לתיאור **סוג המידע** וטקסט להראות **מה הוא כעת**

#### ה-LinearLayout למטה יופיע אם ורק אם נכנס משתמש שהוא או מנהל (Admin) או סופר מנהל (Super-Admin). לחיצה על ה-LinearLayout עם המידע על דרגת המשתמש והתמונה של המפתח ברגים לוקחת את המשתמש למסך ManagerUserAcitivity.

#### צילום של מסך ההגדרות אם נכנס אליו מנהל או סופר מנהל:



**Linear Layout**

עם תמונה וטקסט

הטקסט מראה את דרגת המשתמש

אם המשתמש רואה את איבר זה הוא או מנהל (Admin) או סופר מנהל(Super-Admin).

**Toolbar**

משמש ככותרת להסבר המסך

**TextView**

טקסט לתיאור **שם המשתמש ומינו**

**Toolbar**

משמש ככותרת להסבר המסך

**TextView**

טקסט לתיאור **סוג המידע** וטקסט להראות **מה הוא כעת**

**ImageView**

להראות את **סוג המידע** בתמונה

**TextView**

**טקסט חותמת** לשם האפליקציה

#### על לחיצת כל שורה בהגדרות מופיעה AlertDialog שאיתו אפשר לשנות פרט מסוים בנתונים של המשתמש. כל פרטי המשתמש ניתנים לשינוי חוץ משמו כי זהו מפתח בבסיס הנתונים:

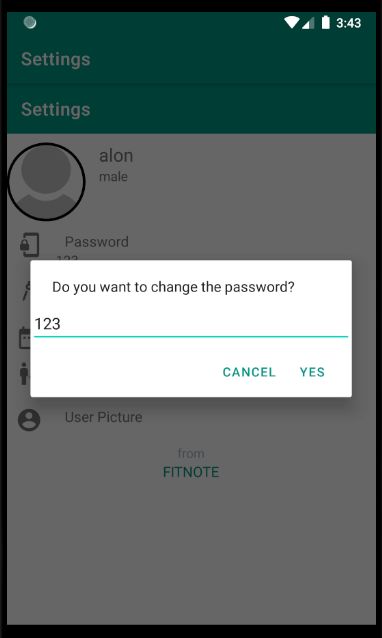
שינוי סיסמה:

בחירת הנתונים ב- **AlertDialog**מעדכנת את פרטי המשתמש עם הנתון החדש. במקרה זה הסיסמה מתעדכנת.

כולל את הרכיבים הבאים:

**EditText**

צילום של ה- **AlertDialog**:



**AlertDialog**

YES לאישור העדכון של הסיסמה עם מה שכתוב ב- **EditText**.

CANCEL לביטול התהליך וסגירת ה- **AlertDialog**.

**EditText**

להכנסת סיסמה חדשה

**TextView**

**טקסט חותמת** לשם האפליקציה

**TextView**

**טקסט חותמת** לשם האפליקציה

שינוי משקל וגובה:

בחירת הנתונים ב- **AlertDialog**מעדכנת את פרטי המשתמש עם הנתון החדש. במקרה זה המשקל והגובה מתעדכנים.

כולל את הרכיבים הבאים:

**TextView, SeekBar**

צילום של ה- **AlertDialog**:



**AlertDialog**

YES לאישור העדכון של המשקל והגובה עם מה שכתוב ב- **TextView**.

CANCEL לביטול התהליך וסגירת ה- **AlertDialog**.

**SeekBar**-ים

לקליטת משקל וגובה של המשתמש

**TextView**

טקסטיםלהסבר מה **סוג המשתנה**, **היחידות** המידה שלו ו**הכמות** שלו כעת לפי ה-**SeekBar**

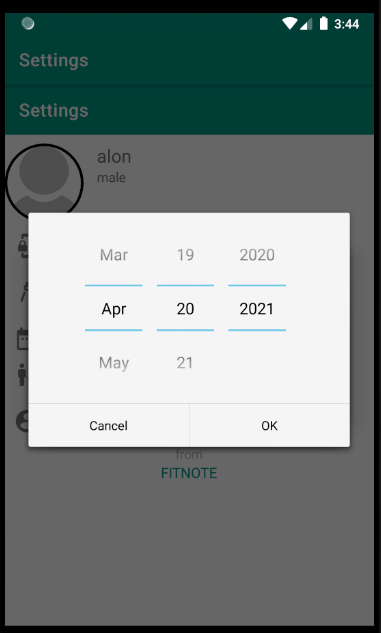
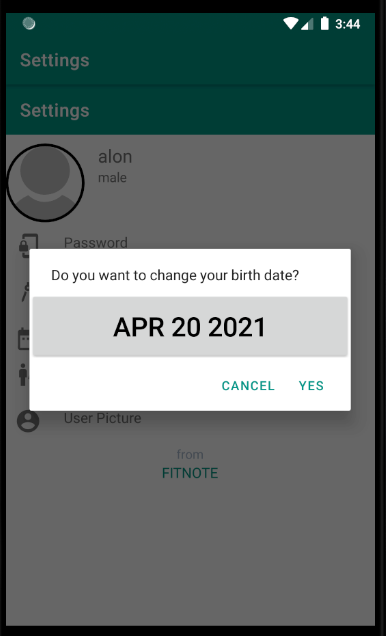
שינוי תאריך לידה:

בחירת הנתונים ב- **AlertDialog**מעדכנת את פרטי המשתמש עם הנתון החדש. במקרה זה תאריך הלידה מתעדכן.

כולל את הרכיבים הבאים:

**Button**

צילום של ה- **AlertDialog**וה- **DatePickerDialog**:



**AlertDialog**

YES לאישור העדכון של תאריך הלידה עם מה שכתוב ב- **Button**.

CANCEL לביטול התהליך וסגירת ה- **AlertDialog**.

**DatePickerDialog**

לבחירת תאריך הלידה החדש:

OK לאישור תאריך זה. אז התאריך יועבר לכפתור () ב-- **AlertDialog**.

CANCEL לביטול בחירת התאריך וחזרה ל- **AlertDialog**.

**Button**

כפתור עם תאריך של היום:

לחיצה עליו מפעילה **DatePickerDialog**. איתו בוחרים את תאריך הלידה החדש.

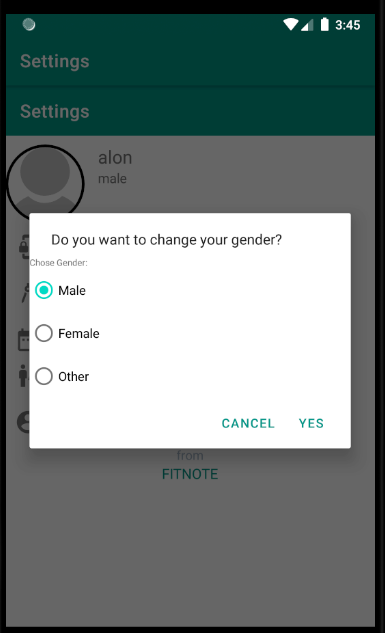
שינוי מין:

בחירת הנתונים ב- **AlertDialog**מעדכנת את פרטי המשתמש עם הנתון החדש. במקרה זה מין המשתמש מתעדכן.

כולל את הרכיבים הבאים:

**TextView, RadioGroup, RadioButton**

צילום של ה- **AlertDialog**:



**AlertDialog**

YES לאישור העדכון של מין המשתמש עם מה שכתוב ב- **RadioButton**הנבחר.

CANCEL לביטול התהליך וסגירת ה- **AlertDialog**.

**TextView**

כותרת להסבר

**RadioGroup**

שמאגד כמה **RadioButton-ים**:

כפתורים אלו קולטים את מין המשתמש

שינוי תמונת משתמש:

צילום של ה- **AlertDialog**:

בחירת הנתונים ב- **AlertDialog**מעדכנת את פרטי המשתמש עם הנתון החדש. במקרה זה תמונת המשתמש מתעדכנת.

כולל את הרכיבים הבאים:

**TextView, ImageView**

צילום של ה- **AlertDialog**:



**AlertDialog**

YES לאישור העדכון של תמונת המשתמש עם מה שצולם.

CANCEL לביטול התהליך וסגירת ה- **AlertDialog**.

**ImageView**

תמונה של מצלמה: לחיצה עליה פותחת פעולה של לקיחת תמונה עם המצלמה בטלפון.

אם אין אישור לגשת למצלמה לא תהיה אפשרות לצלם בטלפון,

ובהכנסת פרטי המשתמש יוכנס תמונת הפרופיל הברירת מחדל. אם המשתמש כן צילם את עצמו תוכנס לבסיס הנתונים התמונה שהוא צילם

**ImageView**

תמונה של משתמש:

מראה את תמונת המשתמש לאחר קליטת התמונה

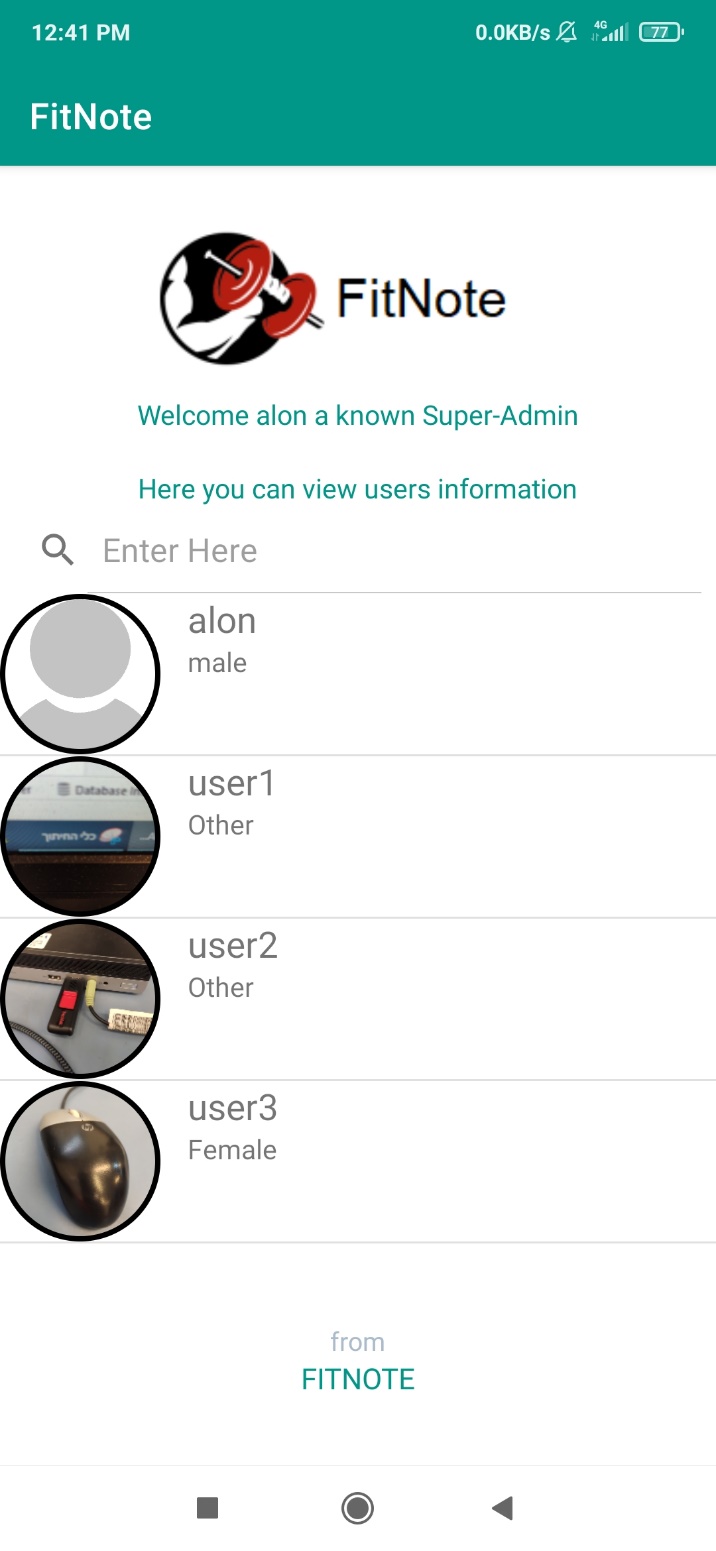
#### ManagerUsersActivity

**מסך ניהול המשתמשים**: למסך זה, לפי ההגבלה במסך ההגדרות, יכול להיכנס רק משתמש שאינו רגיל. מסך זה מציג את שם המשתמש שנכנס ודרגתו. אם המשתמש הוא מנהל רגיל (**Admin**) אז הוא יוכל לראות את המשתמשים ברשימה בלבד. אם המשתמש הוא סופר מנהל (**Super-Admin**) הוא יוכל ללחוץ על אחד המשתמשים ברשימה (חוץ מעצמו) ולהגיע למסך (**EditUserActivity**) בו הוא יוכל לשנות את פרטי המשתמש (מלבד שם ותמונה) ואף למחוק את המשתמש הנבחר.

כולל את הרכיבים הבאים:

**TextView, ImageView, SearchView, ListView**

צילום של מסך ניהול המשתמשים:



**ListView**

רשימה של **כל המשתמשים בבסיס הנתונים** (Users):

לחיצה על כל משתמש מלבד המתשמש עצמו שהתחבר מעבירה את המשתמש למסך עיצוב המשתמשים **EditUserActivity** בו הוא יכול לשנות את פרטי המשתמש הנבחר

**TextView**

**טקסט** לכותרת הסבר ולהראות את שם המשתמש ואת דרגתו

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים משתמשים ברשימת המשתמשים

**TextView**

**טקסט חותמת** לשם האפליקציה

**ImageView**

לוגו של אפליקציה

#### EditUserActivity

#### מסך עיצוב המשתמש: במסך זה המשתמש שנכנס יכול לשנות את כל פרטי המשתמש מלבד שמו ותמונת הפרופיל שלו. בנוסף, המשתמש יכול למחוק את זה שהוא בחר. מחיקה, עדכון או ביטול בלחיצה על הכפתורים DELETE UPDATE ו-CANCEL בהתאמה מחזירה את המשתמש למסך ניהול המשתמשים.

כולל את הרכיבים הבאים:

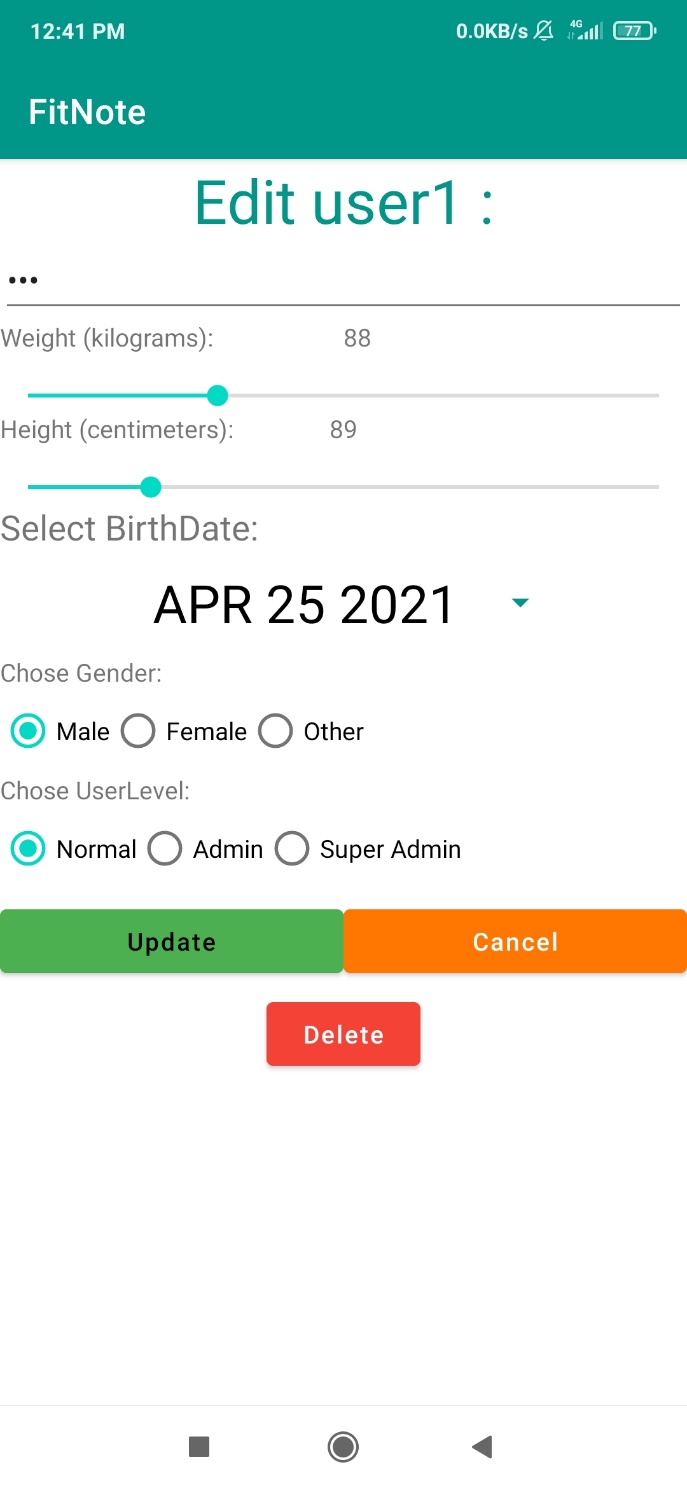
**TextView, EditText, SeekBar, Button, RadioGroup, RadioButton**

צילום של מסך עיצוב המשתמש:

**EditText**

לקליטת פרט משתמש חדש:

לקליטת סיסמא



**ShareActivity**

**TextView**

להסבר ולהראות את שם המשתמש הנבחר

**Button**

כפתור לקליטת לחיצה של המשתמש:

בלחיצה מופיעה ה-**DatePickerDialog**

שהוא לוקח את תאריך לידתו של המשתמש.

התאריך מועבר מה-**DatePicker** לטקסט בכפתור ואז מהטקסט נקלח המידע לבסיס הנתונים

**TextView**

להסבר איזה מידע לוקחים בכפתור

**TextView**

לתיאור המידע הנקלט מה-**SeekBar**

**SeekBar**

לקליטת משקל וגובה של המשתמש

**RadioGroup**

שמאגד כמה **RadioButton-ים**:

כפתורים אלו קולטים את מין המשתמש

**Button**

כפתורים לעדכון הפרטים החדשים בבסיס הנתונים למשתמש הנבחר, למחיקתו ולביטול התהליך.

**מסך שיתוף המידע**: במסך זה המשתמש יכול, במידה ואישר לאפליקציה לגשת לאנשי הקשר שלו, לבחור איש קשר אליו הוא רוצה לשלוח הודעה עם **סיכום על כל תוצאות תרגיליו** **בשבוע האחרון** ועם הודעה נוספת שהוא כותב בהתחלה.

לחיצה על אישר קשר ברשימה למטה מעבירה את המשתמש למסך **WhatsAppSendActivity** בו שולחים את ההודעה למספר הטלפון הנבחר.

כולל את הרכיבים הבאים:

**TextView, RecyclerView** **SearchView,**

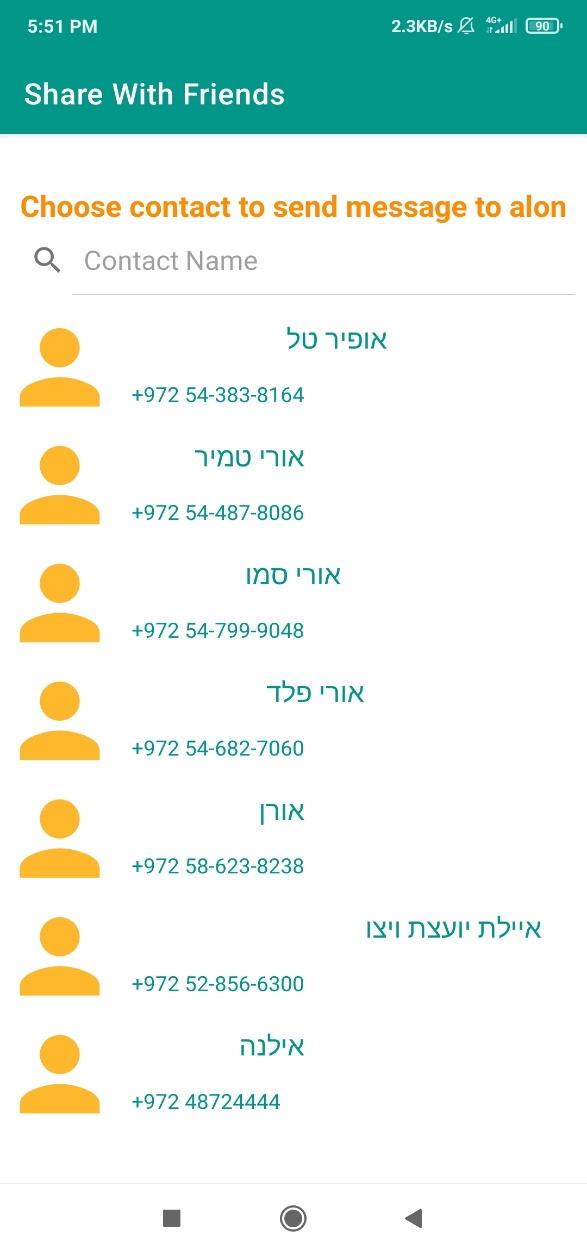
**TextView**

טקסט להסבר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

צילום של מסך שיתוף המידע:



**RecyclerView**

רשימה להצגת אנשי הקשר בטלפון של המשתמש

#### WhatsAppSendActivity

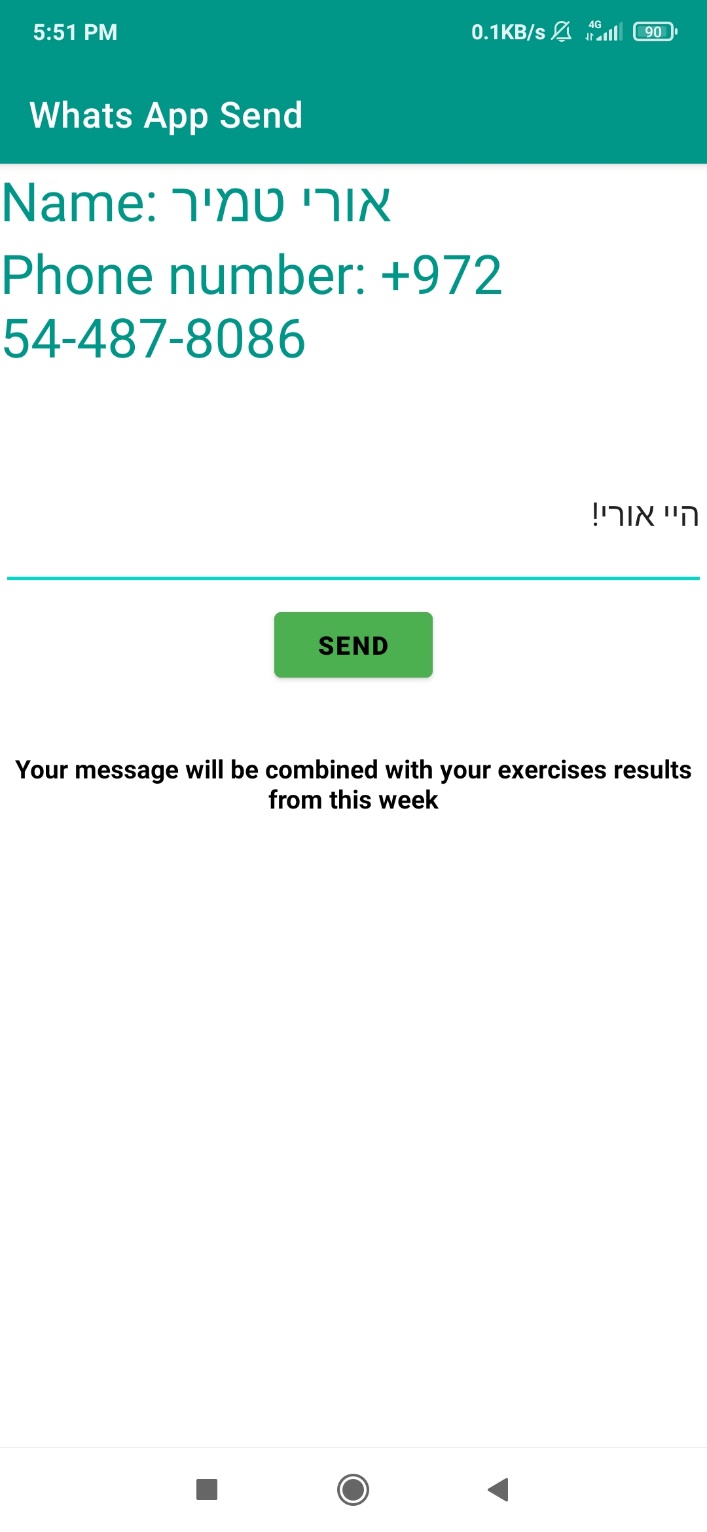
**מסך שליחת הודעה לאיש קשר באווטצאפ**: במסך זה המשתמש כותב הודעה הנשלחת לאיש הקשר אותו בחר במסך **ShareActivity**. להודעה הנשלחת מתווסף טקסט אותו יכול לכתוב המשתמש ב-**EditText**.

תהיה אפשרות לשלוח הודעה באווטצאפ רק עם האפליקציה WhatsApp מותקנת בטלפון.

כולל את הרכיבים הבאים:

**TextView, EditText, Button**

צילום של מסך שליחת הודעה לאיש קשר באווטצאפ:



**TextView**

טקסט להסבר

**TextView**

טקסט להסבר

**TextView**

טקסטים להצגת שם ומספר איש הקשר שנבחר במסך הקודם

**EditText**

לכתיבת הודעה לאיש הקשר ב-WhatsApp

**Button**

כפתור לשליחת ההודעה

**EditText**

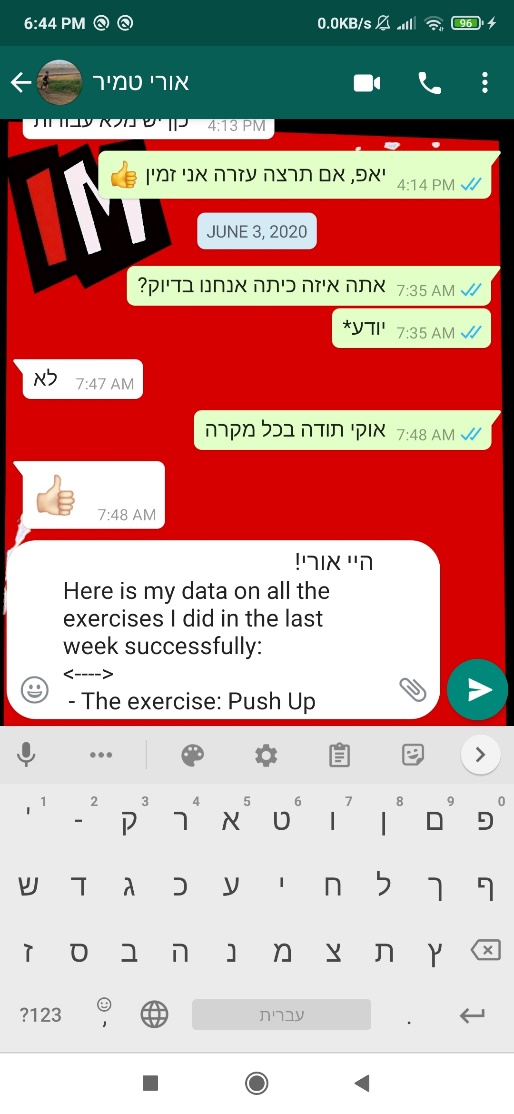
לכתיבת הודעה לאיש הקשר ב-WhatsApp

**TextView**

טקסט להסבר איזה נתונים יופיעו בהודעה

### דוגמא לשליחת הודעה ב-WhatsApp

צילום מסך של שליחת הודעה באווטצאפ הכוללת הודעה אישית של המשתמש ונתונים על תוצאות התרגילים מהאפליקציה:



היי אורי!

Here is my data on all the exercises I did in the last week successfully:

<---->

 - The exercise: Push Up

 - The date in which the exercise was done:APR 21 2021

 - The time it took to perform the exercise: 00 : 00 : 10

 - The rating alon gave to the exercise: Easy-Medium

 - alon did the exercise 47times

<---->

 - The exercise: HammerCurls

 - The date in which the exercise was done:APR 21 2021

 - The time it took to perform the exercise: 00 : 00 : 13

 - The rating alon gave to the exercise: Medium

 - alon did the exercise 30times

#### ViewExercisesResultsActivity

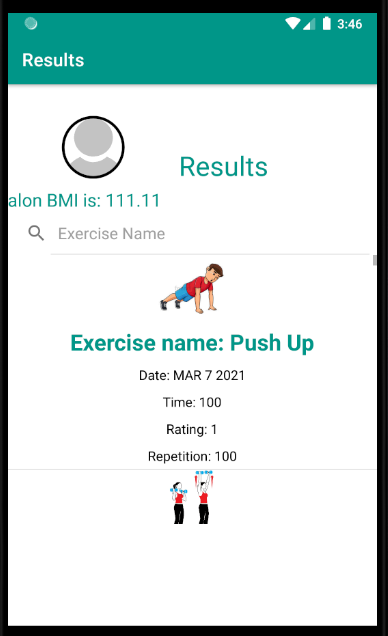
**מסך נתונים על התרגילים שבוצעו ברשימה**: במסך זה מופיעים כל התרגילים שביצע המשתמש והנתונים עליהם ברשימה. למעלה מופיעה ה-BMI של המשתמש שמחושב לפי הגובה של המשתמש ומשקלו(BMI זה משקל חלקי הגובה במטרים בריבוע).

ספרייה של CircleImageView: <https://github.com/hdodenhof/CircleImageView>

כולל את הרכיבים הבאים:

**CircleImageView (from a library on GitHub), TextView, SearchView, ListView**

צילום של מסך נתונים על התרגילים שבוצעו ברשימה:



**CircleImageView**

תמונה של משתמש:

מראה את תמונת המשתמש

**TextView**

טקסט לכותרת וטקסט להראות את ה-BMI של המשתמש

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים את התרגילים ברשימת התרגילים

**ListView**

רשימה של **כל תרגילי המשתמש שבוצעו**(UserExercise):

הרשימה נעזרת במתאם **Adapter** מיוחד הנקרא **UserExerciseDoneAdaptor**

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

**SearchView**

איבר לכתיבת טקסט: לפיו מסננים אנשי קשר ברשימת אנשי הקשר

#### StatisticsActivity

**מסך הטבלה**: במסך זה מופיעה טבלת chart בה יש נתונים על כל התרגילים שביצע המשתמש.

* צבע העמודות מראה את הדירוג של המתאמן על דרגת הקושי של התרגיל.
* מיקום בציר ה-x מציין את החודש שבו בוצע התרגיל.
* גובה ביחס לציר ה-y מציין את מספר החזרות שביצע המתאמן בתרגיל זה.

\*מעל הטבלה יש פילטרים להראות סוג מסוים של תרגילים ובשנה מסוימת.

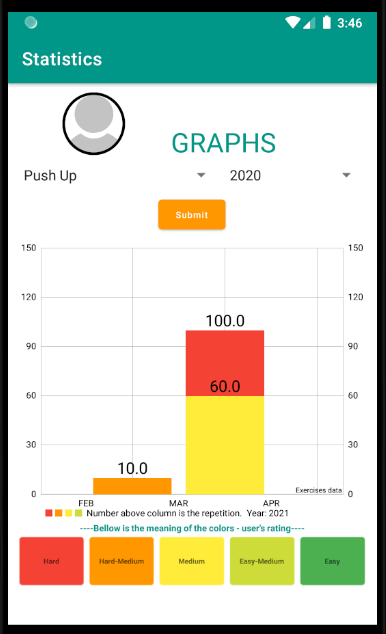
ספרייה של MPAndroidChart: <https://github.com/PhilJay/MPAndroidChart-Realm>

ספרייה של CircleImageView: <https://github.com/hdodenhof/CircleImageView>

כולל את הרכיבים הבאים:

**CircleImageView (from a library on GitHub), Spinner, Button, BarChart (from a library on GitHub), TextView**

צילום של מסך הטבלה:



**TextView**

טקסט לכותרת

**CircleImageView**

תמונה של משתמש:

מראה את תמונת המשתמש

**Button**

כפתור לעדכון פילטרים למידע המופיע בטבלה

**Spinner**

להכנסת שנה וסוג תרגיל להראות בטבלה

**BarChart**

להצגת מידע על תוצאות תרגילי המשתמש באופן ויזואלי

**TextView**

טקסטים להסבר על הטבלה

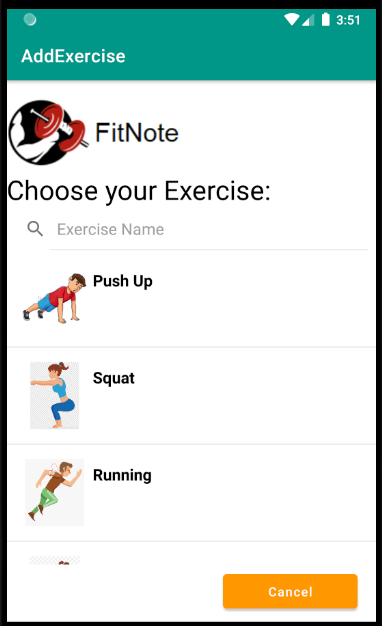
#### AddExerciseActivity

**מסך הוספת התרגילים**: במסך זה המשתמש בוחר את התרגיל שהוא רוצה לבצע. לחיצה על התרגיל תפעיל **AlertDialog** בו יופיע הסבר על מה התרגיל שהמשתמש רוצה להוסיף. אישור במסך הדו שיח יוסיף את התרגיל לרשימת התרגילים שהמשתמש עוד לא ביצע. ביטול במסך הדו שיח יוריד את ה-**AlertDialog** וישאיר את המשתמש באותו מסך.

כולל את הרכיבים הבאים:

**ListView, TextView, SearchView, Button**

צילום של מסך הוספת התרגילים:



**TextView**

כותרת הסבר

**ImageView**

לוגו האפליקציה

**Button**

כפתור לביטול התהליך וחזרה למסך **ProgramUserActivity**

**SearchView**

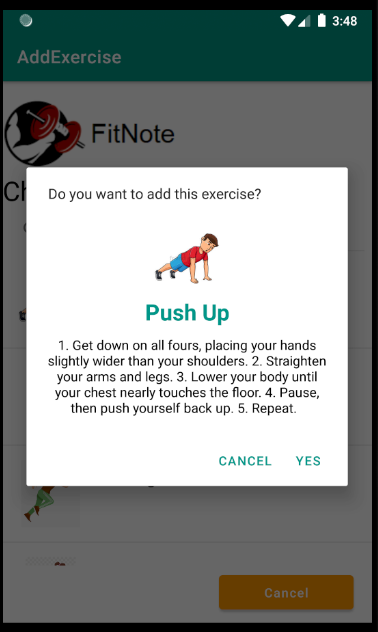
איבר לכתיבת טקסט: לפיו מסננים תרגילים ברשימת התרגילים

**ListView**

רשימה של כל סוגי התרגילים (Exercises):

הרשימה נעזרת במתאם **Adapter** מיוחד הנקרא **ExerciseAdaptor**

צילום מסך הוספת התרגילים עם הופעה של ה-dialog:



**AlertDialog**

כולל את הרכיבים הבאים:

**ImageView, TextView**

ב-YES מאשרים את הוספת התרגיל וב-CANCEL מבטלים את הוספת התרגיל ונשארים באותו מסך

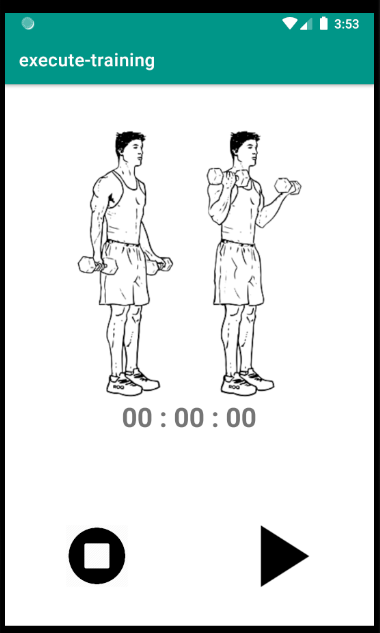
#### ExecuteExerciseActivity

**מסך ביצוע התרגילים**: במסך זה המשתמש מבצע את התרגיל שהוא בחר מקודם במסך **ProgramUserActivity**. לחיצה על התמונה PLAY מימין למטה מפעיל את שירות (**Service**) המוזיקה והטיימר **MusicAndTimerService**. לחיצה על התמונה PLAY פעם אחת הופכת את המשולש לשני קווים מקבילים שזה בעצם כפתור ה-PAUSE לחיצה על כפתור ה-PAUSE משהה את השירות והופכת את הכפתור בחזרה לכפתור ה-PLAY. לחיצה על התמונה STOP משמאל למטה עוצרת את השירות (**Service**) לגמרי ומעבירה את המשתמש למסך **FeedbackActivity**.

כולל את הרכיבים הבאים:

**ImageView, TextView**

צילום של מסך ביצוע התרגילים:



**TextView**

להראות את הזמן בטיימר המופעל בלחיצה על כפתור ההפעלה מימין למטה

**ImageView**

להראות את תמונת התרגיל שמבצע המשתמש

**ImageView**

כפתורים להפעלת המוזיקה והטיימר, השהייתו והפסקתו. לחיצה על כפתור ההפעלה הופכת אותו לכפתור השהייה. לחיצה על כפתור העצירה מעבירה את המשתמש למסך המשוב **FeedbackActivity**

### FeedbackActivity

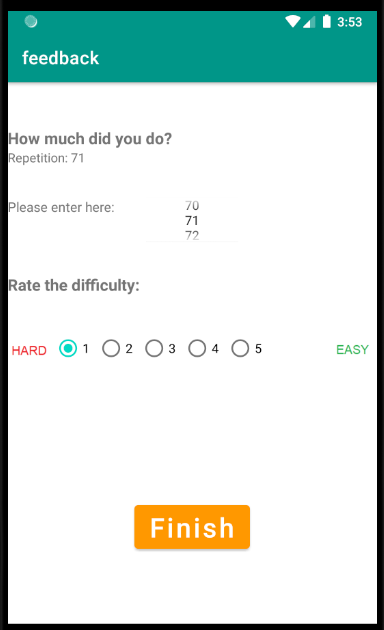
**מסך משוב על ביצוע התרגיל**: במסך זה המשתמש כותב כמה ביצועים הוא ביצע בתרגיל במסך הקודם **ExecuteExerciseActivity**, והוא מדרג את רמת הקושי של התרגיל שהוא ביצע. את הזמן שלקח לבצע את התרגיל, את ה- userExerciseIDשל התרגיל מהמסך הקודם, את מספר הביצועים ואת דירוג הקושי לוקחים ביחד ומעדכנים את תרגיל המשתמש בבסיס הנתונים עם הנתונים החדשים. **באופן זה התרגיל הופך מתרגיל שהמשתמש לא ביצע לכזה שהוא כן ביצע**.

לחיצה על כפתור ה-FINISH מסיים את תהליך ביצוע התרגיל והמשוב עליו. לאחר מכן מעודכן תרגיל המשתמש עם המידע החדש על ביצוע התרגיל והמשתמש חוזר למסך **ProgramUserActivity**.

כולל את הרכיבים הבאים:

**TextView, NumberPicker, ImageView, RadioGroup, RadioButton, Button**

צילום מסך המשוב על ביצוע התרגילים:



**Button**

כפתורים לסיום ביצוע משוב על התרגיל: בלחיצה על הכפתור חוזרים למסך **ProgramUserActivity**

**RadioGroup**

שמאגד כמה **RadioButton-ים**:

כפתורים אלו קולטים את דרגת הקושי

**ImageView**

תמונה להראות את הסקלה של בחירת דרגת הקושי של התרגיל שהמשתמש ביצע

1-קשה עד 2-קל

**NumberPicker**

לבחירת מספר הביצועים בתרגיל שביצע המשתמש

**TextView**

כותרות להסבר

וכותרת להראות את מספר הביצועים שבחר המשתמש ב-**NumberPicker**

### Services

הגדרת השירות (Service):

Service הוא שירות שנועד לביצוע פעולות ארוכות טווח. טווח של זמן בו יש סיכוי שהמשתמש יעבור לאפליקציה אחרת. כשנעבור מסך ה-Thread (תהליכון ברקע) יופסק. כלומר כשנסגור את ה-activity ה-thread יופסק. לעומת זאת, ה-Service ימשיך לפעול גם כשנעבור בין מסכים. נוכל לעצור את ה-Service רק בסגירה מלאה של האפליקציה או בהפסקת השירות. לשירות יש תוחלת חיים משל עצמו. הוא מוגבל ביכולת שלו לבוא באינטראקציה עם רכיבים (components מבחינת UI). רוב השירותים הם started.

שימו ל-StartService: שמיעת מוזיקה ברקע.



#### MusicAndTimerService

שירות שמפעיל מוזיקה וטיימר ברקע בעת ביצוע התרגיל במסך ExecuteExerciseActivity. המחלקה יורשת ממחלקת service שהיא מובנת בתוך ה-android.

תכונות:

1. שלוש שדות במחלקה:

* dataBaseHelper לגישה לבסיס הנתונים באפליקציה ללקיחת השירים שננגן
* mediaPlayer לניגון השירים שניקח מבסיס הנתונים
* countDownTimer לעדכון ה-count כל שנייה
* count לשמירת זמן ביצוע התרגיל בשניות

private DataBaseHelper dataBaseHelper;  
  
private MediaPlayer mediaPlayer;  
  
private CountDownTimer countDownTimer;

1. מימוש פעולות הורשה

* onCreate פעולה שנקראת בעת מימוש השירות שבה משיגים שיר רנדומלי מהשירים הנלקחים מבסיס הנתונים ומממשים את הטיימר ואת נגן המוזיקה
* onStart פעולה שנקראת בעת הפעלת השירות. ב-Intent מועבר String שמסביר אם הפעולה המבוצעת על ידי השירות צריכה להיות הפעלת המוזיקה והטיימר (PLAY)או לשהות אותם (PAUSE)
* onDestroy פעולה הנקראת בעת עצירת השירות והשמדתו. הפעולה מבטלת את הטיימר ועוצרת את המוזיקה

1. הרשמת השירות ב-Manifest

<service android:name=".MusicAndTimerService" />

1. הפעלת השרות על ידי ה-Intent

לנגינת השיר:

Intent startService = new Intent(this, MusicAndTimerService.class);  
startService.putExtra("ACTION", "PLAY");  
startService(startService);

להשהיית הנגינה של השיר:

Intent startService = new Intent(this, MusicAndTimerService.class);  
startService.putExtra("ACTION", "PAUSE");  
startService(startService);

1. עצירת השרות לגמרי:

*//stop button to stop music service*Intent startService = new Intent(this, MusicAndTimerService.class);  
stopService(startService);

### Content Providers

הגדרת ה-ContentProvider:

הוא רכיב המשמש להעברת מידע בין אפליקציה ContentProvider אחת לשנייה. נשתמש ב-ContentProvider במידה ונרצה שהאפליקציה שלנו תקבל מידע מאפליקציה אחרת. לדוגמא באפליקציה זו שלפתי את אנשי הקשר שבטלפון וכן הצגתי את מספרי הטלפון ושמות אנשי הקשר על מנת לשלוח לאיש קשר הנבחר הודעה הכוללת מידע על תוצאות התרגילים של המשתמש.

תכונות:

1. בקשת הרשאה לגשת לנתוני אנשי הקשר בטלפון:

private void checkPermission() {  
 *//Check condition* if (ContextCompat.*checkSelfPermission*(ShareActivity.this  
 , Manifest.permission.*READ\_CONTACTS*)  
 != PackageManager.*PERMISSION\_GRANTED*){  
 *//When permission is not granted  
 //Request permission* ActivityCompat.*requestPermissions*(ShareActivity.this  
 , new String[]{Manifest.permission.*READ\_CONTACTS*}, 100);  
 }else {  
 *//When permission is granted  
 //Create method* getContactList();  
 }  
}

1. שליפת נתונים לאובייקט (עצם) שנקרא Cursor:

private void getContactList() {  
 *//Initialize uri  
 //uniform resource identifier* Uri uri = ContactsContract.Contacts.*CONTENT\_URI*;  
  
 *//Sort by ascending (ASC)* String sort = ContactsContract.CommonDataKinds.Phone.*DISPLAY\_NAME*+" ASC";  
  
 *//Initialize cursos* Cursor cursor = getContentResolver().query(  
 uri, null, null, null, sort  
 );  
  
 *//Check condition* if(cursor.getCount() > 0){  
 *//When count is greater than 0  
 //Use while loop* while (cursor.moveToNext()){  
 *//Cursor move to next  
 //Get contact id* String id = cursor.getString(cursor.getColumnIndex(  
 ContactsContract.Contacts.*\_ID* ));  
 *//Get contact name* String name = cursor.getString(cursor.getColumnIndex(  
 ContactsContract.Contacts.*DISPLAY\_NAME* ));  
 *//Initialize phone uri* Uri uriPhone = ContactsContract.CommonDataKinds.Phone.*CONTENT\_URI*;  
 *//Initialize selection* String selection = ContactsContract.CommonDataKinds.Phone.*CONTACT\_ID* +" =?";  
 *//Initialize phone cursor* Cursor phoneCursor = getContentResolver().query(  
 uriPhone, null, selection  
 , new String[]{id}, null  
 );  
 *//Check condition* if (phoneCursor.moveToNext()) {  
 *//When phone cursor move to next* String number = phoneCursor.getString(phoneCursor.getColumnIndex(  
 ContactsContract.CommonDataKinds.Phone.*NUMBER* ));  
 *//Initialize contact model* ContactModel model = new ContactModel();  
 *//Set name* model.setName(name);  
 *//Set number* if(number.charAt(0) == '0')  
 {  
 number = "+972 "+number.substring(1);  
 }  
 model.setNumber(number);  
 *//Add model in array list* contactList.add(model);  
 *//Close number cursor* phoneCursor.close();  
 }  
 }  
 *//Close cursor* cursor.close();  
 }  
  
 *//setting filtered list to be contactList* filteredContacts = contactList;  
  
 *//Set layout manager* recyclerViewContacts.setLayoutManager(new LinearLayoutManager(this));  
  
 *//setOnClickListener* setOnClickListener();  
  
 *//Initialize adapter* adapter = new ContactListAdapter(this, contactList, listener);  
 *//Set adapter* recyclerViewContacts.setAdapter(adapter);  
}

1. הצגת הנתונים ב-RecycleView:

*//setting filtered list to be contactList*filteredContacts = contactList;  
  
*//Set layout manager*recyclerViewContacts.setLayoutManager(new LinearLayoutManager(this));  
  
*//setOnClickListener*setOnClickListener();  
  
*//Initialize adapter*adapter = new ContactListAdapter(this, contactList, listener);  
*//Set adapter*recyclerViewContacts.setAdapter(adapter);

1. לכל איבר ברשימה יש onClick לקליטת הבחירה של המשתמש באיש קשר מסוים:

private void setOnClickListener() {  
 listener = new ContactListAdapter.RecycleViewClickListener() {  
 @Override  
 public void onClick(View v, int position) {  
 *//now we are creating an intent to go to the WhatsAppSendActivity* Intent intent = new Intent(getApplicationContext(), WhatsAppSendActivity.class);  
  
 *//now we want to get the contact ID, name and number:* ContactModel contactChosen = filteredContacts.get(position);  
 String name = contactChosen.getName();  
 String number = contactChosen.getNumber();  
  
 intent.putExtra("contactName", name);  
 intent.putExtra("contactNumber", number);  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
 };  
}

### Broadcast Receivers

הגדרת ה-Broadcastreceiver:

BroadcastReceiver הינה מחלקה אשר יכולה להאזין למסרים (Intent) שמשודרים. על מנת ליצור BroadcastReceiver צריך ליצור מחלקה ולרשת מ- BroadcastReceiverוליישם את הפונקציה onReceive.

דוגמאות לשימושים: האזנה לבטרייה, לשיחה נכנסת או יוצאת, לשעון, ל-alarm manager, לכניסה של sms, לכניסה של push notification.

במעגל החיים של BroadcastReceiver ישנה רק פונקציה אחת: onReceive. לאחר הפונקציה onReceive ה- BroadcastReceiver נהרס.

#### ReminderBroadcast

BroadcastReceiver ליצירת notification מ-Intent של alarmManager בזמן שהגדיר המשתמש מראש.

תכונות:

1. יצירת ערוץ להתראות במידה וצריך – יצירת NotificationChannel(בגרסות חדשות של אנדרואיד צריך לשייך התראות לערוצים כדי שהם יופיעו):

private void createNotificationChannel(){  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*){  
 CharSequence name = "LembuitReminderChannel";  
 String description = "Channel for Lembuit Remider";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel("notifyLemubit", name, importance);  
 channel.setDescription(description);  
  
  
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
}

1. יצירת Intent עם מידע שיופיע ב-notification והפיכתו ל-pendingIntent:

*//Set notificationID & message*Intent notificationIntent = new Intent(  
 ProgramUserActivity.this, ReminderBroadcast.class);  
  
notificationIntent.putExtra("message", "Its time to train " + *activeUserName*);  
notificationIntent.putExtra("activeUserName", *activeUserName*);  
  
*//PendingIntent*PendingIntent pendingIntent = PendingIntent.*getBroadcast*(  
 ProgramUserActivity.this, 0, notificationIntent, PendingIntent.*FLAG\_UPDATE\_CURRENT*);  
*//FLAG\_CANCEL\_CURRENT-if pending already exists, the current one will be canceled.  
//FLAG\_UPDATE\_CURRENT-indicates that the pendingIntent  
//which we created now can be updated in the future*

1. יצירת AlarmManager אליו נוסיף את ה-pendingIntent ,נגדיר בו את הזמן הופעת ה-notification ויופעל ה-broadCast. ב-AlarmManager נגדיר בנוסף לזמן ההתראה את סוג ה-Alarm(לדוגמא RTC\_WAKEUP):
2. *//AlarmManager  
   //Use with getSystemService to retrieve an AlarmManager  
   //for receiving intents at a time of your choosing*AlarmManager alarmManager = (AlarmManager) getSystemService(*ALARM\_SERVICE*);  
     
   switch (v.getId()){  
    case R.id.*btnSetAlarm*:  
    *// Set Alarm - we got hour and minutes from: tvHour,tvMinute  
     
    // Create time* Calendar startTime = Calendar.*getInstance*();  
    startTime.set(Calendar.*HOUR\_OF\_DAY*, tvHour);  
    startTime.set(Calendar.*MINUTE*, tvMinute);  
    startTime.set(Calendar.*SECOND*, 1);  
    long alarmStartTime = startTime.getTimeInMillis();  
     
    *//checking that the time given is not before now* if (alarmStartTime <= System.*currentTimeMillis*()){  
     
    Toast.*makeText*(this, "This time is before now! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
     
    }else{  
    *// Set Alarm  
    //AlarmManager.RTC\_WAKEUP - wakes up the device to fire the pending intent  
    //at the specified time* alarmManager.set(AlarmManager.*RTC\_WAKEUP*, alarmStartTime, pendingIntent);  
     
    Toast.*makeText*(this, "Set Done! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
    }  
     
    break;  
     
    case R.id.*btnCancelAlarm*:  
    *// Cancel Alarm* alarmManager.cancel(pendingIntent);  
    Toast.*makeText*(this, "Canceled", Toast.*LENGTH\_SHORT*).show();  
    break;  
   }
3. קבלת ה-pendingIntent מה-AlarmManager כשהזמן ששמור ב-AlarmManager הוא הזמן כעת. בעזרת הבנאי הייעודי – NotificationCompat.Builder בונים התראה עם המידע מה-pendingIntent:
4. public class ReminderBroadcast extends BroadcastReceiver {  
    @Override  
    public void onReceive(Context context, Intent intent) {  
     
    Log.*d*("RECEIVED", "Notification received!" + " UserName" + intent.getStringExtra("activeUserName"));  
     
    *// Get id & message from intent.* String message = intent.getStringExtra("message");  
    String activeUserName = intent.getStringExtra("activeUserName");  
     
    *// Call ProgramUserActivity when notification is tapped.* Intent programUserAcIntent = new Intent(context, ProgramUserActivity.class);  
     
    *//we need to add this userName in order to take the user to his  
    //ProgramUserActivity screen* programUserAcIntent.putExtra("activeUserName", activeUserName);  
     
    *//Intent to go to user's exercise list when clicking on notification* PendingIntent contentIntent = PendingIntent.*getActivity*(  
    context, 0, programUserAcIntent, 0  
    );  
     
    NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "notifyLemubit")  
    .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
    .setContentTitle("Exercise Time!")  
    .setContentText(message)  
    .setContentIntent(contentIntent)  
    .setAutoCancel(true)  
    .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
     
    NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(context);  
     
    *//According to the official documentation  
    //starting in Android 8.0(API level 26)  
    //all notifications must be assigned to a channel  
     
    //the notificationID is a unique int for each notification that must be defined* notificationManager.notify(NotificationID.*getID*(), builder.build());  
     
    }  
     
    public static class NotificationID {  
    private final static AtomicInteger *c* = new AtomicInteger(0);  
    public static int getID() {  
    return *c*.incrementAndGet();  
    }  
    }  
     
   }

### שיקולי תכן (design)

**SQLiteOpenHelper** **:**

**SQLiteOpenHelper** היא מחלקת עוזר לניהול יצירת מסד נתונים וניהול גרסאות.

**SQLiteOpenHelper** מספק כלי עזר לפשט את המשימות של יצירת ואתחול מסד הנתונים אם הוא עדיין לא נוצר והמרת תוכן מסד הנתונים כאשר היישום שלך משודרג וסכימת מסד הנתונים משתנה.

אתה יוצר תת-מחלקה המיישמת onCreate(SQLiteDatabase) , onUpgrade(SQLiteDatabase,int,int) ואופציה אפשרית ל- onOpen(SQLiteDatabase) .מחלקה זו דואגת לפתוח את מסד הנתונים אם הוא קיים, ליצור אותו אם לא, ולשדרג אותו לפי הצורך. נעשה שימוש בעסקאות כדי לוודא שמסד הנתונים נמצא תמיד במצב הגיוני.

מחלקה זו מקלה על יישומי ContentProvider לדחות את פתיחת ושדרוג מסד הנתונים עד לשימוש ראשון, כדי למנוע חסימה של הפעלת יישומים עם שדרוגים ארוכי טווח של מסדי נתונים.

בנוסף, הרבה יותר פשוט לנהל מחלקה בה יש את כל הפעולות הבסיסיות לשליפת מידע ממוסד הנתונים. אין צורך לכתוב את אותו קוד ארוך בו לוקחים, מסננים ומציבים את המשתנים הספציפיים, ובמקום זאת פשוט קוראים לפעולה המתאימה לסיטואציה מסוימת.

מחלקה זאת ייחודית להתעסקות עם מידע בבסיס הנתונים כך שתמיד ברור ששם צריך לשלוף את המידע מבסיס הנתונים ואין בלבול בין מחלקות שונות באפליקציה.

האפשרות השנייה שלי בנושא בסיסי הנתונים ב-**SQLlite** היא להשתמש ב-**SQLiteDatabase**. למרות שבתחילת הפרויקט מימשתי בסיס נתונים זהה למה שקיים כעת ב-  **SQLiteDatabase**, עברתי לבסיס נתונים מסודר וברור יותר במחלקה היורשת מ- **SQLiteOpenHelper**. זאת מפני שרוב האנשים המתכנתים ב-**AndroidStudio** ב-**YouTube** וב-**StackOverflow** השתמשו בעוזר כדי לגשת לבסיס הנתונים ולשלוף את המידע המתאים. אני למדתי לממש הרבה עצמים ומחלקות באפליקציה מלראות איך יוטיוברים רבים ומשתמשי **StackOverflow** עושים זאת. לרוב המתכנתים נעזרו ב- **SQLiteOpenHelper** למימוש בסיס הנתונים. לכן העדפתי לממש את המחלקה הזאת ולנהל את מוסד הנתונים כך.

**MusicAndTimerService** **:**

במחלקה זו יש שירות שמפעיל מוזיקה וטיימר ברקע בעת ביצוע התרגיל במסך **ExecuteExerciseActivity**. המחלקה יורשת ממחלקת **service** שהיא מובנת בתוך ה-**android**.

**Service** הוא שירות שנועד לביצוע פעולות ארוכות טווח. טווח של זמן בו יש סיכוי שהמשתמש יעבור לאפליקציה אחרת. כשנעבור מסך ה-**Thread** (תהליכון ברקע) יופסק. כלומר כשנסגור את ה-**activity** ה-**thread** יופסק. לעומת זאת, ה-**Service** ימשיך לפעול גם כשנעבור בין מסכים. נוכל לעצור את ה-**Service** רק בסגירה מלאה של האפליקציה או בהפסקת השירות. לשירות יש תוחלת חיים משל עצמו. הוא מוגבל ביכולת שלו לבוא באינטראקציה עם רכיבים (**components** מבחינת **UI**). רוב השירותים הם **started**.

שימו ל-**StartService**: שמיעת מוזיקה ברקע.

השירות הזה מופעל בכניסה למסך ביצוע התרגילים (**ExecuteExerciseActivity**) ומוספק ביציאה ממסך זה דרך סיום ביצוע התרגיל או דרך חזרה אחורה למסך התרגילים של המשתשמש (**ProgramUserActivity**).

מפני שזהו שירות, הוא ימשיך לפעול ברקע אם המשתמש יוצא מהאפליקציה וסוגר את הטלפון (כל עוד הוא לא סגר אותה באופן מלא). באופן זה המתאמן יכול להיכנס למסך ביצוע התרגיל, להפעיל את הטיימר וללכת לבצע את התרגיל כך שהמוזיקה תמשיך לפעול והיא תיהיה הסימן לכך שהאפליקציה ממשיכה בטיימר. כאשר המשתמש מסיים את ביצוע התרגיל הוא יוכל לפתוח את מסך האפליקציה איפה שהוא עזב אותה ולסיים את הביצוע של התרגיל.

כלומר, השימוש בשירות איפשר לי ליישם אפשרות להפעיל טיימר ומוזיקה לביצוע תרגיל שימשיך גם הטלפון נירדם למצב שינה וכל עוד לא כיבו את הטלפון לגמרי ולא סגרו את האפליקציה לגמרי.

**User Admin Relationship** **:**

כחלק משימוש בעקרונות ההורשה והפולימורפיזם, מימשתי מחלה של משתמש ושל מנהל.

למשתמש השדות הבאים: שם משתמש, סיסמה, משקל, גובה, תאריך לידה, מין, תמונת פרופיל.

public class User {private String userName;  
 private String userPassword;  
 private int userWeight;  
 private int userHeight;  
 private String userBirthDate;  
 private String userGender;  
 private String profilePic;

מחלקת המנהל יורשת ממחלקת המשתמש ולה שדה חדש: **isSuperAdmin**.

public class Admin extends User{  
  
 private boolean isSuperAdmin;

מחלקה זו מאפשרת לתת למשתמשים מסוימים לגשת למידע ולאפשרויות שאני לא רוצה שכל משתמש יוכל להתעסק בהם. במצב זה נוצרים מספר דרגות למשתמש: רגיל, מנהל וסופר מנהל.

למשתמש רגיל אין סמכויות מיוחדות והוא יכול להשתמש באפליקציה באופן רגיל מבלי לדעת דבר על מיהם המתשמשים האחרים דרך האפליקציה.

משתמש מנהל יכול לגשת לרשימה מיוחדת דרך מסך ההגדרות (**SettingsActivity**) בוא הוא יכול לראות ולעבור על רשימה מלאה של כל המשתמשים עם שמם ומינם. באופן זה עדין המתשמש מוגבל מלדעת פרטים נוספים על משתמשים אחרים שהם: סיסמתם, משקלם, גובהם, מינם ודרגתם.

משתמש סופר מנהל בעל הכי הרבה סמכויות בתוך האפליקציה. הוא יכול לגשת לרשימה שאליה יכול לגשת המנהל, והוא אף יכול בבחירת אחד המשתמשים שזה לא הוא לגשת למסך עיצוב המשתמש (**EditUserActivity**) בוא הסופר מנהל יכול לעדכן את פרטי המשתמש בכאלו חדשים ואף למחוק את המשתמש הנבחר.

בחרתי להשתמש בהורשה ועקרונות הפולימורפיזם מפני שכך יכולתי לממש את הרעיון של המנהל והסופר מנהל באפליקציה. רציתי דרך פשוטה וקלה לגשת למידע על המשתמשים מבלי שכל משתמש יוכל לעשות זאת. לכן, ביצירת בסיס הנתונים נוצר משתמש alon שהוא אוטומטית סופר מנהל והוא המשתמש היחיד ביצירת בסיס הנתונים. באופן זה אני יכול לגשת ולבדוק את המידע בבסיס הנתונים, ואם ארצה אהפוך משתמשים נוספים למנהלים וסופר מנהלים. עם הפולימורפיזם לפיו אפשר להתייחס אל כמה עצמים שיורשים מאותו מחלקת על כאל עצמים מסוג מחלקת העל יכולתי לספק למשתמשים מסוימים סמכויות מיוחדות ובאותו הזמן להתייחס אל כולם כאל משתמשים. בנוסף, באופן זה כשמשתמשים בפעולה getUserLevel() של מחלקות המשתמש והמנהל, לפי הפולימורפיזם, מקבלים 0 עבור איבר שמתייחסים אליו כאל מחלקת משתמש והוא מסוג משתמש או שמקבלים 1 או 2 אם האיבר שמתייחסים אליו כאל מחלקת משתמש הוא בעצם מסוג מנהל.

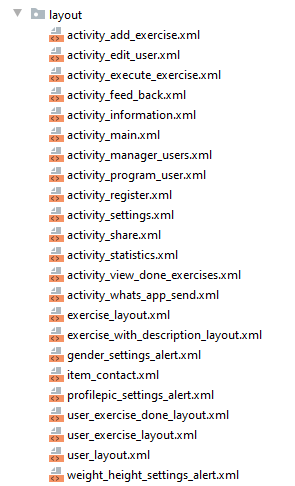
## Resources

פירוט ה- resources השונים המשמשים לפרויקט, עם הסבר לגבי כל אחד מהם:

**

#### Layouts

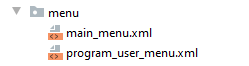
פירוט קבצי ה-XML ולאיזה Activity כל אחד מהם מתאים:



* activity\_add\_exercise: עיצוב של מסך **AddExerciseActivity**
* activity\_edit\_user: עיצוב של מסך **EditUserActivity**
* activity\_execute\_exerise: עיצוב של מסך **ExecuteExerciseActivity**
* activity\_feed\_back: עיצוב של מסך **FeedbackActivity**
* activity\_information: עיצוב של מסך **InformationActivity**
* activity\_main: עיצוב של מסך **MainScreenActivity**
* activity\_manager\_user: עיצוב של מסך **ManagerUserActivity**
* activity\_program\_user: עיצוב של מסך **ProgramUserActivity**
* activity\_register: עיצוב של מסך **RegisterActivity**
* activity\_settings: עיצוב של מסך **SettingsActivity**
* activity\_share: עיצוב של מסך **ShareActivity**
* activity\_statistics: עיצוב של מסך **StatisticsActivity**
* activity\_view\_done\_exercises: עיצוב של מסך **ViewExercisesResultsActivity**
* activity\_whats\_app\_send: עיצוב של מסך **WhatsAppSendActivity**
* exercise\_layout: עיצוב של תרגילים ב-**ListView** במסך **AddExerciseActivity**
* exercise\_with\_description\_layout: עיצוב של תרגילים ב-AlertDialogבמסך **AddExerciseActivity**
* gender\_settings\_alert: עיצוב של **AlertDialog** לשינוי מין משתמש במסך **SettingsActivity**
* item\_contact: עיצוב של אנשי קשר ב-**RecyclerView** במסך **ShareActivity**.
* profilepic\_settings\_alert: עיצוב של **AlertDialog** לשינוי תמונת משתמש במסך **SettingsActivity**
* user\_exercise\_done\_layout: עיצוב של תרגילים שבוצעו ב-**ListView** במסך **ViewExercisesResultsActivity**
* user\_exercise\_layout: עיצוב של תרגילים שלא בוצעו ב-**ListView** ב-**ProgramUserActivty**
* user\_layout: עיצוב של משתמש ברשימה במסך **ManagerUuserActivity**
* weight\_height\_settings\_alert: עיצוב של **AlertDialog** לשינוי משקל וגובה של משתמש במסך **SettingsActivity**

#### Menus

פירוט קבצי ה-XML וציון איזה תפריט כל אחד מהם מממש ובאיזה Activity נעשה בו שימוש:

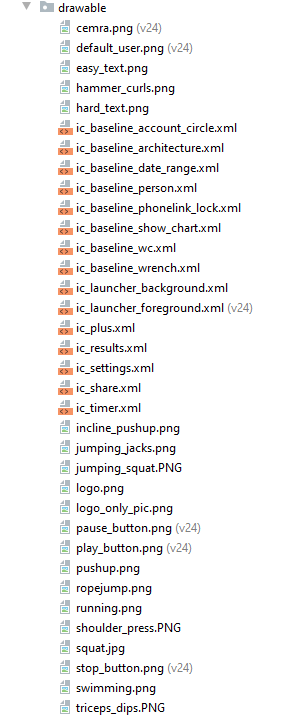


* main\_menu: מממש את התפריט של המסך הראשי **MainScreenActivity** הכוללת אפשרות לעבור למסך המידע על האפליקציה (about) ואפשרות לצאת מהאפליקציה (exit).
* Program\_user\_menu: מממש את התפריט של מסך התרגילים של המשתמש **ProgramUserActivity**. תפריט זו כוללת מספר אפשרויות:

1. אפשרות ללכת למסך הטבלה **StatisticsActivity** (graph).
2. אפשרות לעבור למסך תוצאות התרגילים שבוצעו **ViewExercisesResultsActivity** (results).
3. אפשרות לעבור למסך שיתוף המידע **ShareActivity** (share).
4. אפשרות לעבור למסך ההגדרות של המשתמש **SettingsActivity** (settings).

#### Drawables

הצגת תמונות תחת כותרות מתאימות:



סוגי תמונות:

* תמונות מ-**Vector Asset** (Asset Studio):

1) ic\_baseline\_account\_circle

2)ic\_baseline\_architecture

3)ic\_baseline\_date\_range

4)ic\_baseline\_person

5)ic\_baseline\_phonelink\_lock

6)ic\_baseline\_show\_chart

7)ic\_baseline\_wc

8)ic\_baseline\_wrench

9)ic\_launcher\_background

10)v24\ic\_launcher\_foreground

11)plus

21)results

31)settings

14)share

15)timer

* תמונות של תרגילים **Exercises**:

1) hammer\_curls

2)incline\_pushup

3)jumping\_jacks

4)jumping\_squat

5)pushup

6)ropejump

7)running

8)shoulder\_press

9)squat

10)swimming

11)triceps\_dips

* תמונות של אייקונים:

1) cemra

2)default\_user

3)easy\_text

4)hard\_text

5)logo

6)logo\_only\_pic

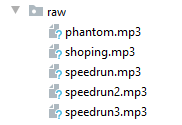
7)pause\_button

8)play\_button

9)stop\_button

#### Raw

הצגת השירים ב-resources:



כל אחד מ-5 השירים האלו נמצא בבסיס הנתונים בטבלת השירים.

בעת ביצוע התרגיל במסך ביצוע התרגילים (**ExecuteExerciseActivity**) מופעל אחד מהשירים האלה באופן רנדומלי.

## קבצים

אני משתמש באפליקציה בשני סוגי קבצים:

### SQLiteDataBase

בבסיס הנתונים קבצים SQLite flies שהם מאוכסנים בזיכרון פנימי. מחלקת DataBaseHelper שומרת נתונים שהיא מקבלת ממחלקות אחרות בקבצים כאלה בבסיס הנתונים.

קובץ DB הוא קובץ ששומר את הנתונים באפליקציה (משתמשים, תרגילים, תרגילי משתמש ושירים).

### קובץ info פנימי בזיכרון הטלפון

מדובר בזיכרון פנימי בטלפון מחוץ לאפליקציה.

מדבור בקובץ שממומש על ידי SharedPreferences בקובץ info בטלפון לשמירת תמונת משתמשים באפליקציה.

**הפעולות בהן נעשה שימוש בקובץ**

* PictureFileHelper פעולות במחלקה זו
* DataBaseHelper פעולות מרכזיות במחלקה זו

## סכמת Database

טבלאות והקשרים:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **טבלת Users** | | | | |
|  | שם השדה | טיפוס הנתונים | מפתח זר | תיאור |
| Key Icons - Download 245 Free Key icons here | userName | TEXT |  | שם המשתמש |
|  | userPassword | TEXT |  | סיסמה של משתמש |
|  | userLevel | INTEGER |  | דרגת משתמש. כאשר:  0- רגיל   1. מנהל 2. סופר מנהל |
|  | userWeight | INTEGER |  | משקל משתמש |
|  | userHeight | INTEGER |  | גובה משתמש |
|  | userBirthDate | TEXT |  | תאריך לידה של משתמש |
|  | userGender | TEXT |  | מין משתמש |
|  | profilePic | INTEGER |  | תמונת פרופיל של משתמש |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **טבלת Exercises** | | | | |
|  | שם השדה | טיפוס הנתונים | מפתח זר | תיאור |
| Key Icons - Download 245 Free Key icons here | exerciseID | INTEGER |  | ID של תרגיל |
|  | exerciseName | TEXT |  | שם תרגיל |
|  | exercisePic | INTEGER |  | תמונת תרגיל |
|  | exerciseDetail | TEXT |  | תיאור תרגיל |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **טבלת UserExercises** | | | | |
|  | שם השדה | טיפוס הנתונים | מפתח זר | תיאור |
| Key Icons - Download 245 Free Key icons here**AUTOINCREMENT** | userExerciseID | INTEGER |  | ID של תרגיל משתמש |
|  | userName | TEXT | מטבלת Users | שם משתמש |
|  | exerciseID | INTEGER | מטבלת Exercises | ID של תרגיל |
|  | date | TEXT |  | תאריך של ביצוע התרגיל |
|  | time | INTEGER |  | זמן שלקח לבצע את התרגיל |
|  | rating | INTEGER |  | דירוג דרגת הקושי של התרגיל בסקלה מ-1 עד 5. כאשר:  1-הכי קשה  5-הכי קל |
|  | repetition | INTEGER |  | מספר הביצועים שהיו בביצוע התרגיל |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **טבלת Songs** | | | | |
|  | שם השדה | טיפוס הנתונים | מפתח זר | תיאור |
| Key Icons - Download 245 Free Key icons here | songID | INTEGER |  | ID של שיר |
|  | songName | TEXT |  | שם שיר |
|  | songMP3 | INTEGER |  | ID של קובץ raw של השיר |

# מדריך משתמש

## מטרת המערכת

**המערכת עוקבת אחרי תרגילים שהמשתמש עושה בזמנו החופשי**.

באפליקציה ישנם 11 תרגילים פשוטים הכוללים הסברים מפורטים על האופן בו יש לבצע אותם.

מטרת האפליקציה היא **לסייע למתאמנים שרוצים לסדר את התוצאות התרגילים שלהם במקום אחד בו הם יכולים בקלות לנתח ולהבין את המגמה שלהם ביחס לביצועים קודמים**. זאת על ידי מעבר על רשימה מסודרת של תוצאות התרגילים שביצעו. בנוסף, המשתמש יכול לנתח טבלת עמודות הכוללת נתונים על מספר הביצועים, דירוג הקושי, החודש והשנה של התרגילים שבוצעו על ידי המשתמש.

מתאמן שרוצה **לשתף את חבריו בקלות עם תוצאות אימוניו** יכול באפליקציה זו לחשוף את חבריו ב-WhatsApp לכל תוצאות התרגילים שלו דרך האפליקציה.

אם למשתמש יש חברים שגם רוצים להצטרף לאימון הגדול, **הוא יכול לתת לחבריו להתחבר כמשתמשים שונים לאותו הטלפון**. כך הוא מעודד אותם גם לעשות אימון כושר ובמידה ויש להם טלפון **המשתמש יכול לשכנע אותם גם להוריד את האפליקציה!**

בכניסה ראשונה לאפליקציה ל**מסך הפתיחה** (**MainScreenActivity**) המשתמש יכול לעבור דרך התפריט במסך זה ל**מסך המידע** (**InformationActivity**). במסך זה ניתן לראות מהם התרגילים שיש באפליקציה עוד לפני שנרשמים. לאחר מכן, המשתמש נדרש להירשם כדי להתחיל את האימון הראשון שלו. דרך כפתור ה-Register ב**מסך הפתיחה** (**MainScreenActivity**) המשתמש עובר ל**מסך הרישום** (**RegisterActivity**). כדי להירשם יש לצלם תמונת פרופיל דרך המצלמה בטלפון ולהכניס שם משתמש, סיסמה, משקל, גובה, תאריך לידה ומין. בלחיצה על FINISH המשתמש חוזר ל**מסך הפתיחה** (**MainScreenActivity**) והוא יכול להירשם ב-LOGIN.

לאחר לחיצה על כפתור ה-LOGIN המשתמש עובר ל**מסך התרגילים של המשתמש** (**ProgramUserActivity**) בו הוא רואה את כל התרגילים שלו שהוא עוד לא ביצע. כדי להוסיף תרגילים ליבצוע עליו ללחוץ על כפתור הפלוס ולעבור ל**מסך הוספת התרגילים** (**AddExerciseActivity**) בו המשתמש בוחר תרגיל לביצוע מתוך רשימה של תרגילים. לאחר שבחר תרגיל מסוים ואישר את הוספתו, המשתמש עובר חזרה ל**מסך התרגילים של המשתמש** (**ProgramUserActivity**) שם הוא יראה את התרגיל החדש ברשימת תרגילי המשתמש. לחיצה על תמונת התרגיל או על הטקסט לידה (כל מקום על האיבר מלבד כפתור המחיקה DELETE) תעביר את המשתמש ל**מסך ביצוע התרגילים** (**ExecuteExerciseActivity**). במסך זה המשתמש מבצע את התרגיל עם הפעלת טיימר ומוזיקה תוך כדי. לאחר שסים לבצע את התרגיל ולחץ כל כפתור ה-STOP שהוא ריבוע, המשתמש עובר למסך האחרון בתהליך ביצוע התרגיל שהוא **מסך משוב על ביצוע התרגיל** (**FeedbackActivity**). במסך זה המשתמש מכניס את מספר החזרות בתרגיל ואת דרגת הקושי של ביצוע התרגיל. לאחר סיום התהליך בלחיצה על כפתור ה-FINISH המשתמש חוזר ל**מסך התרגילים של המשתמש** (**ProgramUserActivity**) באופן דומה המשתמש יכול לבצע תרגילים כל הזמן.

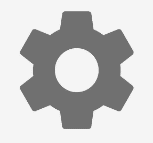
****בנוסף לתהליך זה של האפליקציה המשתמש יכול לנתח, לעקוב ולהבין את מגמת האימונים שלו דרך גרפים, רשימות ומידע שהוא שולח לחבריו ב-whatsApp.

בלחיצה על כפתור השיתוף בתפריט ב**מסך** **התרגילים של המשתמש** (**ProgramUserActivity**) המשתמש עובר ל**מסך שיתוף המידע** (**ShareActivity**) בו, באישור גישה לאנשי הקשר בטלפון שלו, יכול לבחור אחד מאנשי הקשר שיש לו ובמסך הבא שהוא **מסך שליחת הודעה לאיש קשר באווטצאפ** (**WhatsAppSendActivity**) לשלוח לאותו איש קשר הודעה מפורט עם סיכומים על כל נתוני התרגילים שבוצעו על ידי המשתמש.



****בלחיצה על כפתור המידע בתפריט ב**מסך** **התרגילים של המשתמש** (**ProgramUserActivity**) המשתמש עובר ל**מסך נתונים על התרגילים שבוצעו ברשימה** (**ViewExercisesresultsActivity**). במסך זה המשתמש רואה את ה-BMI שלו ואת תוצאות תרגיליו באופן ברור.

בלחיצה על כפתור הסטטיסטיקות בתפריט ב**מסך** **התרגילים של המשתמש** (**ProgramUserActivity**) המשתמש עובר ל**מסך** **הטבלה** (**StatisticsActivity**). במסך זה מופיעה טבלת chart בה יש נתונים על כל התרגילים שביצע המשתמש. בעזרת צבע העמודות, מיקום בציר ה-x וגובה העמודות יכול המשתמש להבין מה מגמתו בכל סוגי התרגילים. מגמות של האטה או השתפרות.

****

בלחיצה על כפתור ההגדרות בתפריט ב**מסך** **התרגילים של המשתמש** (**ProgramUserActivity**) המשתמש עובר ל**מסך** **ההגדרות** (**SettingsActivity**). במסך זה המשתמש יכול לשנות את כל הנתונים שלו בבסיס הנתונים מלבד שמו (זאת מפני ששמו הוא מפתח בבסיס הנתונים).

באפליקציה יש סמכויות מיוחדות למנהלים וסופר מנהלים.

למשתמש באפליקציה זו יש שלושה דרגות אפשריות:

1. רגיל (Normal)
2. מנהל (Admin)
3. סופר מנהל (SuperAdmin)

כך שבמסך ההגדרות (**SettingsActivity**) מופיע למשתמש שהוא מנהל או סופר מנהל אפשרות בלחיצה לעבור למסך ניהול המשתמשים (**ManagerUsersActivity**). אולם, רק סופר מנהל יכול באמת דרך מסך זה לגשת למסך עיצוב המשתמשים (**EditUserActivity**) דרכו סופר מנהל יכול לגשת לכל המשתמשים חוץ מעצמו, ולעדכן את פרטי כל המתשמשים חוץ משמם ולמחוק אותם.

## יכולות המערכת

תיאור האופציות שנותנת המערכת למשתמש (Feature List):

* רישום משתמש
* שינוי פרטי המתמשים על ידיד סופר מנהל או מעבר על רשימת משתמשים בלבד כמנהל
* בחירת תרגיל מרשימה
* שמירת תוצאות של ביצוע תרגילים לפי זמנים
* ביצוע תרגיל מסוים
* הפעלת התראה לזמן קבוע מראש
* הפעלת מוזיקה ברקע ביצוע התרגיל
* ניתוח סטטיסטי של תוצאות התרגילים ממעבר על רשימה וטבלה של תוצאות התרגילים שבוצעו
* שיתוף תוצאות ביצוע התרגילים עם אנשי קשר ב-WhatsApp

## תפעול המערכת

### תרשים זרימה בין המסכים

**מתאר את אופן הניווט בין המסכים השונים:**

**MainScreenActivity**

SUPER ADMIN ONLY

ADMIN ONLY

ManagerUsersActivity

EditUserActivity

SettingsActivity

WhatsAppSendActivity

ShareActivity

ViewExercisesResultsActivity

**ExecuteExerciseActivity**

**FeedbackActivity**

StatisticsActivity

InformationActivity

RegisterActivity

**ProgramUserActivity**

### הרשאות

פירוש ההרשאות שהאפליקציה דורשת והסבר על כל הרשאה:

uses-permission android:name="android.permission.READ\_CONTACTS"

נדרש כדי לאפשר קריאה של אנשי הקשר בטלפון.

**הצהרות:**

הצהרה של ReminderBraodcast:

<receiver  
 android:name=".ReminderBroadcast"  
 android:enabled="true" />

הצהרה של שירות MusicAndTimerService:

<service android:name=".MusicAndTimerService"   
 android:exported="false"/>

פיצר מצלמה:

<uses-feature  
 android:name="android.hardware.Camera"  
 android:required="true" />

### דרישות מיוחדות ומגבלות

יש דרישה להימצאות אפליקציה חיצונית היכולה לצלם בטלפון. אך אם אין כזאת אפליקציה, המשתמש יוכל להמשיך להתאמן כרגיל רק שהוא לא יכול לצלם לאפליקציה תמונת משתמש בעת רישום או בעת שינוי פרטי המשתמש כל עוד אין אפליקציה חיצונית כזאת.

כדי לגשת לאנשי הקשר בטלפון נדרש לאשר זאת בכניסה למסך אנשי הקשר באפליקציה (ShareActivity).

על מנת לשלוח לאיש קשר מסוים הודעה באווטצאפ נדרש שהאפליקציה WhatsApp תיהיה מותקנת בטלפון. אחרת תופיע בקשה להוריד WhatsApp בטלפון ואפשרות של שליחת הודעה דרך האפליקציה לאיש קשר ב-WhatsApp לא תיתאפשר.

#### גירסת Android מינימלית

גירסת Android המינימלית שנדרשת כדי להריץ את האפליקציה:

מינימום SDK: 19

מקסימום SDK: 30

#### מכשירים עליהם נבדקה המערכת

מכשירים בהם נבדקה המערכת: Xiaomi Mi 9T, Pixel 2 API 27, Galaxy Tab S6 Lite

# רפלקציה

העבודה עבורי הייתה מאתגרת ומהנה. במסגרת הפרויקט הייתה לי הזדמנות לתכנן ולבנות אפליקציה רצינית שמממשת את כל הרעיונות שרציתי לישם בה. מהמורה שלי ניצן קיבלתי את הכלים הבסיסיים עבור בניית מסכים, מעבר ביניהם ובניית בסיס הנתונים. לאחר מכן אני חקרתי ביוטיוב ובאתרים נוספים באינטרנט איך כדי לממש בסיס נתונים ואיך בונים מחלקות ופעולות שיעזרו לי לממש את הרעיונות שלי בפעול. להמשך אני לוקח את הידע שצברתי על ניהול מוסד נתונים גדול ומסודר דרך מחלקה נפרדת, ואת העבודה והארגון של אפליקציה עם מספר מסכים בעל תהליך אחד מרכזי ומסכים נוספים לניתוח מידע והגדרות. כדי לממש את כל הרעיונות שהיו לי בהתחלה נדרשתי לבצע מחקר מעמיק על הדרכים שקיימים לשמירת מידע, התאמת נתונים לרשימה, שימוש בשירותים להפעלת טיימר ומוזיקה, שימוש ב-ContentProvider שישיג לי מידע מאפליקציה אחרת בטלפון שמחזיקה אנשי קשר ושימוש ב-BroadcastReciever ליצירת התראה בזמן קבוע מראש. על מנת לממש את כל היכולות האלו הייתי צריך לראות אילו מחלקות ופעולות מתאימות בדיוק לצרכים שלי. המחקר והלמידה העצמית במסגרת הפרויקט הייתה מאתגרת, ובנוסף לכך גם הרבה מאוד זמן לא הייתי בבית הספר עקב התפרצות מגפת הקורונה וסגירת בתי הספר בעקבות זאת. לכן, את רוב הלמידה הייתי צריך לבצע לבד.

לאחר סיום הפרויקט אני מאמין שלמרות הקשיים שהתלוו למצב בניית אפליקציה במסגרת לימודי התוכנה שלי בבית הספר הייתה מאוד מעשירה ועזרה לי לקדם צעד אחד קדימה את ההבנה שלי בתוכנה ובניהול פרויקט רב מחלקטי.

לו הייתי מתחיל עם הידע החדש שצברתי הייתי מרחיב את התקשורת של האפליקציה עם אפליקציות אחרות ומשתמש ביותר מידע חיצוני כמו מזג אוויר ומיקום בכדה"א.

אני סבור שעבודתי הייתה יעילה ביחס למצב שהיה במהלך רוב שנת הלימודים. אם בית הספר היה פתוח לאורך כל השנה אני חושב שהייתי גומר את הפרויקט קצת יותר מוקדם מאיך שגמרתי אותו עכשיו.

לסיכום, אני מרגיש מסופק מהאפליקציה FitNote. בעיניי היא יכולה לסייע לרבים כמוני שרוצים להתאמן ולתרגל תרגילי כושר וגם רוצים לנתח את תוצאותייהם ולעקוב אחרי מגמת הביצוע שלהם, בין אם הם השתפרו ובין אם יש האטה בביצוע.

# ביבליוגראפיה

מקורות מידע שנעזרת בהם:

* ללימוד הנושא
* לגיבוש הרעיון
* לכתיבת הקוד

YouTube, StackOverflow, appschool, codeplayon, github, tutorialspoint, developers.android

# נספחים

## תיעוד – קוד הפרויקט

## עוזר בניית מוסד הנתונים (DataBaseHelper):

מחלקה עוזרת לבניית בסיס הנתונים מסוג SQLite לאפליקציה וניהול הנתונים בה.

קוד המחלקה:

package com.example.fitnote13022021;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class DataBaseHelper extends SQLiteOpenHelper {  
  
 *//Initialize variables* private static final String *DATABASE\_NAME* = "fitnote\_database12345678";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 private static final String *TABLE\_USER* = "Users";  
 private static final String *TABLE\_EXERCISES* = "Exercises";  
 private static final String *TABLE\_USEREXERCISES* = "UserExercises";  
 private static final String *TABLE\_SONGS* = "Songs";  
  
 DataBaseHelper(Context context){  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 public DataBaseHelper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {  
 super(context, name, null, version);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 *//Create tables* String tableUser = "CREATE TABLE " + *TABLE\_USER* + " (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER, userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)";  
 *//userLevel 0-normal, 1-admin, 2-superAdmin* String tableExercise = "CREATE TABLE " + *TABLE\_EXERCISES* + " (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT, exercisePic INTEGER, exerciseDetail TEXT)";  
 String tableUserExercises = "CREATE TABLE " + *TABLE\_USEREXERCISES* + " (userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)";  
 String tableSongs = "CREATE TABLE " + *TABLE\_SONGS* + " (songID INTEGER PRIMARY KEY, songName TEXT, songMP3 INTEGER)";  
  
  
 db.execSQL(tableUser);  
 db.execSQL(tableExercise);  
 db.execSQL(tableUserExercises);  
 db.execSQL(tableSongs);  
  
 *// adding exercises  
 // Exercise Table;  
 // (exerciseID INT PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INT, exerciseDetail TEXT)* db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (1, 'Push Up', "+R.drawable.*pushup*+", '1. Get down on all fours, placing your hands slightly wider than your shoulders. " +  
 "\n" +  
 "2. Straighten your arms and legs. " +  
 "\n" +  
 "3. Lower your body until your chest nearly touches the floor. " +  
 "\n" +  
 "4. Pause, then push yourself back up. " +  
 "\n" +  
 "5. Repeat. ')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (2, 'Squat', "+R.drawable.*squat*+", '1. Find a foot stance that feels best for you. Pointing your toes slightly outwards helps some, but keeping them parallel is fine, too. If you’re not sure what’s best, start by putting your feet shoulder-width apart and pointed about 15 degrees outwards. " +  
 "\n" +  
 "2. Tense your abs like someone is about to punch you. " +  
 "\n" +  
 "3. Look straight ahead and stand tall!')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (3, 'Running', "+R.drawable.*running*+", 'TIPS: When you first start out, try alternating between running and walking during your session. As time goes on, make the running intervals longer until you no longer feel the need to walk. " +  
 "Give yourself a few minutes to cool down after each run by walking and a doing few stretches. Try our post-run stretch routine.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (4, 'RopeJump', "+R.drawable.*ropejump*+", 'Get a large rope that you can pass under your feet. Move the rope under your feet while you jump and repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (5, 'JumpingJacks', "+R.drawable.*jumping\_jacks*+", 'JumpingJacks: Stand upright with your legs together, arms at your sides. "+  
 "\n" +  
 "Bend your knees slightly, and jump into the air. "+  
 "\n" +  
 "As you jump, spread your legs to be about shoulder-width apart. Stretch your arms out and over your head. " +  
 "\n" +  
 "Jump back to starting position. "+  
 "\n" +  
 "Repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (6, 'TricepsDips', "+R.drawable.*triceps\_dips*+", '1. You can do it from a chair or a bench, "+  
 "you can even do it on the floor. "+  
 "\n" +  
 "2. So just getting yourself into position to start by rolling your shoulders down. "+  
 "You want to start from good posture. Sometimes we tend to slump forward and then were gonna potentially cause an extra strain on the shoulder joint. "+  
 "\n" +  
 "3. So set yourself up by rolling the shoulders back, opening up the chest. Bringing your hands directly underneath your shoulders onto the bench or onto the ground. "+  
 "And youre gonna take your legs out keeping your knees bent." +  
 "\n" +  
 "4. If you wanted to make it harder you could extend the legs. "+  
 " So start with the easier option, see how you feel first. "+  
 "\n" +  
 "5. Youre gonna bend the elbows, lowering the hips down, and then exhale to extend the elbows and lift your body up. "+  
 "Inhale down, exhale up.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (7, 'InclinePushUp', "+R.drawable.*incline\_pushup*+", '1. Place your hands on the edge of the elevated surface. " +  
 "\n" +  
 "2. Step your feet back so your legs are straight and your arms are perpendicular to your body. "+  
 "\n" +  
 "3. Inhale as you slowly lower your chest to the edge of your platform. "+  
 "\n" +  
 "4. Pause for a second. "+  
 "\n" +  
 "5. Exhale as you push back to your starting position with your arms fully extended.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (8, 'JumpingSquat', "+R.drawable.*jumping\_squat*+", 'Stand tall with your feet hip-width apart. "+  
 "\n" +  
 "Hinge at the hips to push your butt back and lower down until your thighs are parallel to the floor. "+  
 "\n" +  
 "Allow your knees to bend 45 degrees when you land, and then immediately drop back down into a squat, and jump again.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (9, 'HammerCurls', "+R.drawable.*hammer\_curls*+", 'Step 1 "+  
 "Stand up straight with a dumbbell in each hand, holding them alongside you. Your palms should face your body. Keep your feet hip-width apart and engage your core to stabilize the body. "+  
 "\n" +  
 "Step 2 "+  
 "Keep your biceps stationary and start bending at your elbows, lifting both dumbbells. "+  
 "\n" +  
 "Step 3 " +  
 "Lift until the dumbbells reach shoulder-level, but don’t actually touch your shoulders. Hold this contraction briefly, then lower back to the starting position and repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (10, 'ShoulderPress', "+R.drawable.*shoulder\_press*+", 'Stand with feet shoulder-width apart and hold the dumbbells at shoulder height with your elbows at a 90-degree angle. "+  
 "\n" +  
 "Slowly lift the dumbbells above your head without fully straightening your arms. Pause at the top. "+  
 "\n" +  
 "Slowly return to the start position.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (11, 'Swimming', "+R.drawable.*swimming*+", '1. Float with your face in the water, your body straight and horizontal. Stack your hands and keep your arms and legs long. " +  
 "Point your thumbs down. " +  
 "\n" +  
 "2. Press your hands out and back in a circle, elbows high. Lift your head slightly and inhale. " +  
 "\n" +  
 "3. Bring your hands together in front of your shoulders, thumbs pointing up. Keep your elbows close to your body. Simultaneously bend your knees, bringing your feet toward your butt and pointing your feet outward. " +  
 "\n" +  
 "4. Reach your arms forward. Kick out and back in a circle then snap your feet together. Drop your head underwater and exhale. " +  
 "\n" +  
 "5. Glide forward and repeat.')");  
  
 *// adding songs* db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (1, 'speedrun', "+R.raw.*speedrun*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (2, 'speedrun2', "+R.raw.*speedrun2*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (3, 'speedrun3', "+R.raw.*speedrun3*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (4, 'phantom', "+R.raw.*phantom*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (5, 'shoping', "+R.raw.*shoping*+")");  
  
  
  
 *//Table Users:  
 //(userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)  
 //adding user alon - me the SUPER ADMIN* db.execSQL("INSERT INTO " + *TABLE\_USER* + " VALUES('alon', 123, 2, 90, 90, 'APR 2 2003', 'male', 'alon')");  
  
  
  
 *//adding userExercises  
 //JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(1, 'alon', 1, 'MAR 7 2021', 100, 1, 100)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(2, 'alon', 2, 'JAN 8 2021', 10, 2, 10)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(3, 'alon', 3, 'FEB 9 2021', 60, 3, 60)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(4, 'alon', 4, 'APR 10 2020', 30, 4, 30)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(5, 'alon', 5, 'JUN 11 2020', 50, 5, 50)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(6, 'alon', 6, 'SEP 25 2020', 100, 1, 100)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(7, 'alon', 7, 'MAR 7 2021', 20, 2, 20)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(8, 'alon', 8, 'JAN 8 2021', 56, 3, 56)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(9, 'alon', 9, 'FEB 9 2021', 34, 4, 34)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(10, 'alon', 10, 'APR 10 2020', 70, 5, 70)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(11, 'alon', 11, 'MAR 11 2021', 120, 1, 110)");  
  
  
  
  
  
  
 }  
  
 *//Delete database* public static void deleteDatabase(Context mContext) {  
 mContext.deleteDatabase(*DATABASE\_NAME*);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 *//Drop Existing Table* db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_USER*);  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_EXERCISES*);  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_USEREXERCISES*);  
 onCreate(db);  
 }  
  
 *//method to add user* public boolean insertUser(User user) {  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", 0);  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USER*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
 }  
  
 *//method to add user* public boolean insertAdmin(Admin admin) {  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 int userLevel = 1;  
  
 if (admin.isSuperAdmin() == false){  
 userLevel = 1;  
 }  
 else {  
 userLevel = 2;  
 }  
  
  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", admin.getUserName());  
 cv.put("userPassword", admin.getUserPassword());  
 cv.put("userLevel", userLevel);  
 cv.put("userWeight", admin.getUserWeight());  
 cv.put("userHeight", admin.getUserHeight());  
 cv.put("userBirthDate", admin.getUserBirthDate());  
 cv.put("userGender", admin.getUserGender());  
 cv.put("profilePic", admin.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USER*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
 }  
  
 *//method to update User* public boolean updateUser(User user){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", user.getUserLevel());  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long result = db.update(*TABLE\_USER*, cv, "userName=?", new String []{user.getUserName()});  
  
 if(result == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to update User* public boolean updateUserAsAdmin(Admin user){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 int userLevel = 1;  
  
 if (user.isSuperAdmin())  
 userLevel = 2;  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", userLevel);  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long result = db.update(*TABLE\_USER*, cv, "userName=?", new String []{user.getUserName()});  
  
 if(result == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to delete user (by ID because its the primary key)* public void deleteUser(User user) {  
 *// find user in the database. if its found, delete it and return true.  
 // if its not found, return false.  
  
 // getWritableDatabase - we are going to delete from it* SQLiteDatabase db = this.getWritableDatabase();  
  
 db.delete(*TABLE\_USER*, "userName=?", new String[]{user.getUserName()});  
  
 }  
  
 *//method to get user's level* public int getUserLevelByUserName(String givenUserName){  
 ArrayList<User> returnList = new ArrayList<>();  
  
 int userLevel = 0;  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)  
 // get data from the database* String queryString = ("SELECT " + "\*" + " FROM " + *TABLE\_USER* + " WHERE " + "userName" + " = '" + givenUserName + "'");  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 userLevel = cursor.getInt(2);  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return userLevel;  
 }  
  
 *//method to get all users* public ArrayList<User> getUsers(){  
  
 ArrayList<User> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USER*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = cursor.getString(0);  
 String userPassword = cursor.getString(1);  
 int userLevel = cursor.getInt(2);  
 int userWeight = cursor.getInt(3);  
 int userHeight = cursor.getInt(4);  
 String userBirthDate = cursor.getString(5);  
 String userGender = cursor.getString(6);  
 String profilePic = cursor.getString(7);  
  
 User newUser = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all users* public ArrayList<User> getUsersAndAdmins(){  
  
 ArrayList<User> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USER*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = cursor.getString(0);  
 String userPassword = cursor.getString(1);  
 int userLevel = cursor.getInt(2);  
 int userWeight = cursor.getInt(3);  
 int userHeight = cursor.getInt(4);  
 String userBirthDate = cursor.getString(5);  
 String userGender = cursor.getString(6);  
 String profilePic = cursor.getString(7);  
  
 *//user is Admin at minimum* if (userLevel == 1 || userLevel == 2){  
 *//settings boolean of isSuperAdmin* boolean isSuperAdmin = false;  
 if (userLevel == 2)  
 isSuperAdmin = true;  
  
 User newUser = new Admin(userName, userPassword, isSuperAdmin, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
 }  
 *//user is normal* else {  
 User newUser = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
 }  
  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get user with specific name and password  
 //method to search for user (true - user exists | false - user doesn't exist)* public boolean searchUserByNameAndPass(String userName, String userPassword){  
 *//get all users from user table* List<User> Users = this.getUsers();  
  
 *// if there are no users* if(Users.isEmpty() || (userName.isEmpty() && userPassword.isEmpty()) )  
 return false;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName) && user.getUserPassword() .contentEquals(userPassword)){  
 return true;  
 }  
 }  
  
 return false;  
  
 }  
  
 *//method to search for user (true - user exists | false - user doesn't exist)* public boolean searchUserByName(String userName){  
 *//get all users from user table* List<User> Users = this.getUsersAndAdmins();  
  
 *// if there are no users* if(Users.isEmpty())  
 return false;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName)){  
 return true;  
 }  
 }  
  
 return false;  
  
 }  
  
 *//method to get for user (true - user exists | false - user doesn't exist)* public User getUserByName(String userName){  
 *//get all users from user table* ArrayList<User> Users = this.getUsersAndAdmins();  
  
 *// if there are no users* if(Users.isEmpty() || userName == null)  
 return null;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName)){  
 return user;  
 }  
 }  
  
 return null;  
  
 }  
  
 *//method to insert Exercise* public boolean insertExercise(Exercise exercise){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* ContentValues cv = new ContentValues();  
 cv.put("exerciseID", exercise.getExerciseID());  
 cv.put("exerciseName", exercise.getExerciseName());  
 cv.put("exercisePic", exercise.getExercisePic());  
 cv.put("exerciseDetail", exercise.getExerciseDetail());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_EXERCISES*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to get all exercises* public ArrayList<Exercise> getExercises(){  
  
 ArrayList<Exercise> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_EXERCISES*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
  
 Exercise newExercise = new Exercise(exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newExercise);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all exercises by userNameGiven* public ArrayList<UserExercisesInnerJoinEx> getExercisesByUserNameGiven(String userNameGiven){  
  
 ArrayList<UserExercisesInnerJoinEx> returnList = new ArrayList<>();  
  
 *// get data from the database  
 // First table: Exercises  
 // Second table: UserExercises  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT,  
 // userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER,  
 // rating INTEGER, repetition INTEGER)* String queryString = "SELECT Exercises.exerciseID, exerciseName, exercisePic, exerciseDetail, userExerciseID FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = '"+userNameGiven+"'";  
 *//String queryString = "SELECT Exercises.exerciseID, UserExercises.exerciseID, exerciseName, exercisePic, exerciseDetail FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = " + userNameGiven;  
 // get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *//UserExercisesInnerJoinEx (int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
 int userExerciseID = cursor.getInt(4);  
  
 UserExercisesInnerJoinEx newUserExercisesInnerJoinEx = new UserExercisesInnerJoinEx(userExerciseID, exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newUserExercisesInnerJoinEx);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all exercises by userNameGiven  
 //that have not been done by the user yet (by knowing that the date of doing the exercise is null)* public ArrayList<UserExercisesInnerJoinEx> getExercisesByUserNameGivenNotDone(String userNameGiven){  
  
 ArrayList<UserExercisesInnerJoinEx> returnList = new ArrayList<>();  
  
 *// get data from the database  
 // First table: Exercises  
 // Second table: UserExercises  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT,  
 // userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER,  
 // rating INTEGER, repetition INTEGER)* String queryString = "SELECT Exercises.exerciseID, exerciseName, exercisePic, exerciseDetail, userExerciseID FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = '"+userNameGiven+"' AND UserExercises.date IS NULL";  
 *//String queryString = "SELECT Exercises.exerciseID, UserExercises.exerciseID, exerciseName, exercisePic, exerciseDetail FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = " + userNameGiven;  
 // get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *//UserExercisesInnerJoinEx (int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
 int userExerciseID = cursor.getInt(4);  
  
 UserExercisesInnerJoinEx newUserExercisesInnerJoinEx = new UserExercisesInnerJoinEx(userExerciseID, exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newUserExercisesInnerJoinEx);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to search for exercise by ID* public ArrayList<Exercise> getExercisesByID(int exerciseID){  
 *//get all users from user table* ArrayList<Exercise> exercises = this.getExercises();  
 ArrayList<Exercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(exercises.isEmpty())  
 return null;  
  
 for (int i=0; i<exercises.size(); i++) {  
 Exercise exercise = exercises.get(i);  
 if(exercise.getExerciseID() == exerciseID){  
 returnList.add(exercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to get one exercise by its ID* public Exercise getExerciseByID(int id){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = getReadableDatabase();  
  
 *// Table: Exercises  
 //(exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* Cursor cursor = db.rawQuery("SELECT \* FROM Exercises WHERE exerciseID = " + id, null);  
  
 cursor.moveToFirst();  
  
 int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
  
 Exercise exercise = new Exercise(exerciseID, exerciseName, exercisePic, exerciseDetail);  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return exercise;  
  
 }  
  
 *//method to insert UserExercise* public boolean insertUserExercise(UserExercise userExercise){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", userExercise.getUserName());  
 cv.put("exerciseID", userExercise.getExerciseID());  
 cv.put("date", userExercise.getDate());  
 cv.put("time", userExercise.getTime());  
 cv.put("rating", userExercise.getRating());  
 cv.put("repetition", userExercise.getRepetition());  
  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USEREXERCISES*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to delete userExercise  
 //using it to delete userExercise when in dialog to delete an userExercise* public void deleteUserExercise(String userName,String exerciseName, int userExerciseID){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getWritableDatabase();  
  
 *//table: Exercises  
 //(exerciseID INTEGER PRIMARY KEY, exerciseName TEXT, exercisePic INTEGER, exerciseDetail TEXT)* Cursor c = db.rawQuery("SELECT \* FROM "+ *TABLE\_EXERCISES* +" WHERE exerciseName = '"+exerciseName+"'", null);  
  
 c.moveToFirst();  
  
 *//taking the exerciseID from the selected exercise  
 //there is supposed to be only one exercise with the name it has* int exerciseID = c.getInt(0);  
 *//Log.d("exerciseID", String.valueOf(exerciseID));  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* String queryString = "DELETE FROM " + *TABLE\_USEREXERCISES* + " WHERE userName = '"+userName+"' AND exerciseID = " + exerciseID + " AND userExerciseID = " + userExerciseID;  
  
 Log.*d*("Delete filter", String.*valueOf*(userExerciseID));  
 *//Log.d("Delete filter2", String.valueOf(exerciseID));* db.execSQL(queryString);  
  
 }  
  
 *//method to update userExercise  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* public void updateUserExercise(int userExerciseID, String date, int time, int rating, int repetition){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("date", date);  
 cv.put("time", time);  
 cv.put("rating", rating);  
 cv.put("repetition", repetition);  
  
 db.update(*TABLE\_USEREXERCISES*, cv, "userExerciseID = " + userExerciseID,null);  
  
 }  
  
 *//method to get all UserExercises* public ArrayList<UserExercise> getUserExercises(){  
  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USEREXERCISES*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* int userExerciseID = cursor.getInt(0);  
 String userName = cursor.getString(1);  
 int exerciseID = cursor.getInt(2);  
 String date = cursor.getString(3);  
 int time = cursor.getInt(4);  
 int rating = cursor.getInt(5);  
 int repetition = cursor.getInt(6);  
  
 UserExercise newUserExercise = new UserExercise(userExerciseID, userName, exerciseID, date, time, rating, repetition);  
 returnList.add(newUserExercise);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID* public ArrayList<UserExercise> getUserExercisesByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName() == userName){  
 returnList.add(userExercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID  
 //that have been done by the user (by knowing that the date of doing the exercise is not null)* public ArrayList<UserExercise> getUserExercisesDoneByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName().contentEquals(userName) && userExercise.getDate() != null){  
 returnList.add(userExercise);  
 }  
 }  
  
 if(returnList.isEmpty() == false){  
 Log.*d*("Error","List is empty");  
 }  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID  
 //that have not been done by the user yet (by knowing that the date of doing the exercise is null)* public ArrayList<UserExercise> getUserExercisesNotDoneByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName() == userName && userExercise.getDate() == null){  
 returnList.add(userExercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to get all songs* public ArrayList<Song> getSongs() {  
  
 ArrayList<Song> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_SONGS*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if (cursor.moveToFirst()) {  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do {  
 *// Song Table  
 //(songID INTEGER PRIMARY KEY, songName TEXT, songMP3 INTEGER)* int songID = cursor.getInt(0);  
 String songName = cursor.getString(1);  
 int songMP3 = cursor.getInt(2);  
  
 Song newSong = new Song(songID, songName, songMP3);  
 returnList.add(newSong);  
  
 } while (cursor.moveToNext());  
  
 } else {  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
 }  
}

## דוגמא למתאם(Adapter):

מתאם של תרגילי משתמש שלא בוצעו ברשימה במסך התרגילים של המתשמש (ProgramUserActivity).

קוד המחלקה:

package com.example.fitnote13022021;  
  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ListAdapter;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
  
public class UserExerciseNotDoneAdapter extends BaseAdapter implements ListAdapter {  
  
 private ArrayList<UserExercisesInnerJoinEx> arrayList;  
 public static Context *context*;  
 private DataBaseHelper dataBaseHelper;  
 private String activeUserName;  
  
 public UserExerciseNotDoneAdapter(ArrayList<UserExercisesInnerJoinEx> arrayList, Context context, String userName) {  
 this.arrayList = arrayList;  
 this.*context* = context;  
 dataBaseHelper = new DataBaseHelper(context);  
 this.activeUserName=userName;  
 }  
 public UserExerciseNotDoneAdapter(ArrayList<UserExercisesInnerJoinEx> arrayList, Context context) {  
 this.arrayList = arrayList;  
 this.*context* = context;  
 dataBaseHelper = new DataBaseHelper(context);  
 }  
  
 @Override  
 public int getCount() {  
 return arrayList.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return arrayList.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
  
 UserExercisesInnerJoinEx userExercisesInnerJoinEx = arrayList.get(position);  
 convertView = LayoutInflater.*from*(*context*).inflate(R.layout.*user\_exercise\_layout*, null);  
  
 TextView txtMainTitle = convertView.findViewById(R.id.*txtMainTitle*);  
 ImageView imageView = convertView.findViewById(R.id.*imageView*);  
 Button btnDelete = convertView.findViewById(R.id.*btnDelete*);  
  
 txtMainTitle.setText(userExercisesInnerJoinEx.getExerciseName());  
 imageView.setImageResource(userExercisesInnerJoinEx.getExercisePic());  
  
 btnDelete.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, txtMainTitle.getText().toString(),Toast.*LENGTH\_SHORT*).show();  
  
 *//creating the dialog to delete the userExercise* AlertDialog.Builder builder = new AlertDialog.Builder(*context*);  
  
 builder.setTitle("delete exercise");  
  
 builder.setMessage("Do you really want to delete?");  
  
 *//delete* builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dataBaseHelper.deleteUserExercise(activeUserName,userExercisesInnerJoinEx.getExerciseName(), userExercisesInnerJoinEx.getUserExerciseID());  
 arrayList.remove(position);  
 notifyDataSetChanged();  
 }  
 });  
  
 *//cancel* builder.setNegativeButton("No", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
  
 }  
 });  
  
 imageView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, userExercisesInnerJoinEx.getExerciseName(), Toast.*LENGTH\_SHORT*).show();  
  
 Intent intent1 = new Intent(*context*, ExecuteExerciseActivity.class);  
  
 intent1.putExtra("ExecuteUserExerciseID", userExercisesInnerJoinEx.getUserExerciseID());  
 intent1.putExtra("ExecuteExerciseID", userExercisesInnerJoinEx.getExerciseID());  
 intent1.putExtra("activeUserName",activeUserName);  
 *context*.startActivity(intent1);  
 ((Activity)*context*).finish();  
 }  
 });  
  
 txtMainTitle.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, userExercisesInnerJoinEx.getExerciseName(), Toast.*LENGTH\_SHORT*).show();  
  
 Intent intent1 = new Intent(*context*, ExecuteExerciseActivity.class);  
  
 intent1.putExtra("ExecuteUserExerciseID", userExercisesInnerJoinEx.getUserExerciseID());  
 intent1.putExtra("ExecuteExerciseID", userExercisesInnerJoinEx.getExerciseID());  
 intent1.putExtra("activeUserName",activeUserName);  
 *context*.startActivity(intent1);  
 ((Activity)*context*).finish();  
 }  
 });  
  
 return convertView;  
 }  
}

## מחלקת עזר לשמירה ולקיחת קבצי תמונות (PictureFileHelper):

מחלקת עזר לשמירת תמונות בקובץ info ומציאת תמונה בעזרת שם המשתמש.

קוד המחלקה:

package com.example.fitnote13022021;  
  
import android.app.Activity;  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.ArrayList;  
  
public class PictureFileHelper {  
  
 public static void writeFileToInternalStorage(Context context, Bitmap bitmap, String filename)  
 {  
 SharedPreferences sp= context.getSharedPreferences("info",0);  
 int counter=sp.getInt("counter", 0);  
 try {  
 FileOutputStream os = ((Activity)context).openFileOutput(filename+counter, Context.*MODE\_PRIVATE*);  
 *//Here compress 50%, store the compressed data in os.* bitmap.compress(Bitmap.CompressFormat.*PNG*,100,os);  
 counter++;  
 SharedPreferences.Editor editor=sp.edit();  
 editor.putInt("counter",counter);  
 editor.commit();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static Bitmap readFileFromInternalStorage(Context context,String filename)  
 {  
 Bitmap b=null;  
 try {  
 InputStream in = ((Activity)context).openFileInput(filename);  
 b= BitmapFactory.*decodeStream*(in);  
 in.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return b;  
 }  
  
 public static Bitmap getPic(Context context, String name)  
 {  
 File mydir = context.getFilesDir();  
 File lister = mydir.getAbsoluteFile();  
 Bitmap bitmap=null;  
  
 for (String list : lister.list())  
 {  
 if(list.toString().contains(name)) {  
 *//Toast.makeText(context, list, Toast.LENGTH\_LONG).show();* bitmap = *readFileFromInternalStorage*(context, list);  
  
 }  
 }  
  
 return bitmap;  
 }  
  
}

## מחלקה להשמעת התראה בעזרת BroadcastReceiver (ReminderBroadcast):

המחלקה בונה התראה ברגע שהיא מקבלת הודעה מ-AlarmManager שלו יש זמן מוגדר מראש במסך התרגילים של המתשמש (ProgramUserActivity).

קוד המחלקה:

package com.example.fitnote13022021;  
  
import android.app.PendingIntent;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.util.Log;  
  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import java.util.concurrent.atomic.AtomicInteger;  
  
public class ReminderBroadcast extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
  
 Log.*d*("RECEIVED", "Notification received!" + " UserName" + intent.getStringExtra("activeUserName"));  
  
 *// Get id & message from intent.* String message = intent.getStringExtra("message");  
 String activeUserName = intent.getStringExtra("activeUserName");  
  
 *// Call ProgramUserActivity when notification is tapped.* Intent programUserAcIntent = new Intent(context, ProgramUserActivity.class);  
  
 *//we need to add this userName in order to take the user to his  
 //ProgramUserActivity screen* programUserAcIntent.putExtra("activeUserName", activeUserName);  
  
 *//Intent to go to user's exercise list when clicking on notification* PendingIntent contentIntent = PendingIntent.*getActivity*(  
 context, 0, programUserAcIntent, 0  
 );  
  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "notifyLemubit")  
 .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
 .setContentTitle("Exercise Time!")  
 .setContentText(message)  
 .setContentIntent(contentIntent)  
 .setAutoCancel(true)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(context);  
  
 *//According to the official documentation  
 //starting in Android 8.0(API level 26)  
 //all notifications must be assigned to a channel  
  
 //the notificationID is a unique int for each notification that must be defined* notificationManager.notify(NotificationID.*getID*(), builder.build());  
  
 }  
  
 public static class NotificationID {  
 private final static AtomicInteger *c* = new AtomicInteger(0);  
 public static int getID() {  
 return *c*.incrementAndGet();  
 }  
 }  
  
}

## מחלקה להשמעת מוזיקה והפעלת טיימר בעזרת Service, MediaPlayer ו-CountDownTimer (MusicAndTimerService):

המחלקה בונה שירות כשקוראים לה בפעולה onCreate(). כאשר מפעילים פקודה לשירות נשלח intent לפעולה onStartCommand. ל-intent מצורפת מחרוזת המסבירה איזה פקודה על השירות לבצע (PLAY-לנגן ולהפעיל את הטיימר או PAUSE-להשהות את שנייהם). הפעולה onDestroy() נקראת כשעוצרים את השירות והפעולה עוצרת את הטיימר והמוזיקה ומאפסת את המונה של הזמן.

קוד המחלקה:

package com.example.fitnote13022021;  
  
import android.app.Service;  
import android.content.Intent;  
import android.media.MediaPlayer;  
import android.os.CountDownTimer;  
import android.os.IBinder;  
import android.widget.Toast;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class MusicAndTimerService extends Service {  
  
 private DataBaseHelper dataBaseHelper;  
  
 private MediaPlayer mediaPlayer;  
  
 private CountDownTimer countDownTimer;  
  
 public static int *count* = 0;  
  
 *//this method does binds the service  
 //with and activity  
 //קושרת את השירות עם הפעילות* @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 dataBaseHelper = new DataBaseHelper(this);  
 ArrayList<Song> songs = dataBaseHelper.getSongs();  
  
 int arrayLength = songs.size();  
  
 Random random = new Random();  
 *//This gives a random integer number between 0 (inclusive) and (length of array-1)* int randomNumber = random.nextInt(arrayLength);  
 Song chosenSong = songs.get(randomNumber);  
  
 Toast.*makeText*(this, "chosen song: " + chosenSong.getSongName() + "N: " + randomNumber, Toast.*LENGTH\_SHORT*).show();  
  
 *//create the mediaPlayer* mediaPlayer = MediaPlayer.*create*(this, chosenSong.getSongMP3());  
  
 mediaPlayer.setLooping(true);  
  
 if(countDownTimer != null){  
 countDownTimer.cancel();  
 countDownTimer = null;  
 }  
  
 *//setting up the timer  
 //max: 2147483347millis = 35791.389116667minutes = 596.523151944449978 hours, sec to change: 1000* countDownTimer = new CountDownTimer(Integer.*MAX\_VALUE* - 300, 1000) {  
 @Override  
 public void onTick(long millisUntilFinished) {  
 *count*++;  
 *//setting text in play to be time count from timer in MusicAndTimerService* ExecuteExerciseActivity.*txtTime*.setText(getDurationString(*count*));  
 }  
  
 @Override  
 public void onFinish() {  
 *count* = 0;  
 ExecuteExerciseActivity.*txtTime*.setText(getDurationString(0));  
 }  
 };  
  
 }  
  
 *//when we start the service the onStartCommand method will be called* @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
  
 String action = intent.getStringExtra("ACTION");  
 if(action.contentEquals("PLAY")){  
 Toast.*makeText*(this, "PLAY", Toast.*LENGTH\_SHORT*).show();  
 *//play music and timer* mediaPlayer.start();  
 countDownTimer.start();  
 }else if (action.contentEquals("PAUSE")){  
 Toast.*makeText*(this, "PAUSE", Toast.*LENGTH\_SHORT*).show();  
 *//pause music and timer* mediaPlayer.pause();  
 countDownTimer.cancel();  
 }  
  
 *// this means this service will be explicitly started and stopped  
 //פירוש הדבר ששירות זה יופעל ויפסיק במפורש* return *START\_STICKY*;  
 }  
  
  
 *//when the service is stopped onDestroy method will be called* @Override  
 public void onDestroy() {  
 super.onDestroy();  
  
 *//stop music and timer* mediaPlayer.stop();  
 countDownTimer.cancel();  
 *count* = 0;  
 ExecuteExerciseActivity.*txtTime*.setText(getDurationString(0));  
  
 }  
  
 private String getDurationString(int seconds) {  
  
 int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 seconds = seconds % 60;  
  
 return twoDigitString(hours) + " : " + twoDigitString(minutes) + " : " + twoDigitString(seconds);  
 }  
  
 private String twoDigitString(int number) {  
  
 if (number == 0) {  
 return "00";  
 }  
  
 if (number / 10 == 0) {  
 return "0" + number;  
 }  
  
 return String.*valueOf*(number);  
 }  
  
}

## מסמכים נוספים

כל מסמך שכתבתי, או מצאתי במקור מידע כל שהוא שעזר לי:

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

לפני כתיבת הקוד ועיצוב ה-XML-ים עיצבתי סקיצה של תיק עבודה לאפליקציה.

זאת בהשראה מאפליקציות קיימות בשוק.

בדף הבא מצורף תיק העבודה הראשון שלי שנכתב בתחילת שנת הלימודים הנוכחית:

FitNote תיק עבודה:

תאריך: 25.01.2021

מגיש: אלון בן דוד

מורים: ניצן

המטרה היא לכתוב אפליקציה להכנת אימון כושר יעיל ולמעקב אחרי האימון.

המשתמש מכניס לאפליקציה בעת רישום ראשון את שמו, הסיסמא שלו, תאריך לידתו, מינו ומשקלו.

לאחר מכן, נתוני המשתמש נכנסים למאגר הנתונים של התכונה והוא רשאי להיכנס.

האפליקציה משמשת ככלי לארגון תכניות תרגילים למשתמש.

יהיו תכניות הנמצאים בתוך שדה הנתונים שהתוכנה מציע למשתמש. אך הוא רשאי לבנות תכנית אישי עם שם מסוים, תיאור ואוסף של תרגילים. התרגילים באפליקציה משויכים לחלקי גוף מסוימים כך שניתן לבנות אימון כוח יעיל ומקיף על מספר איברים בגוף מבלי להבין לעומק את אנטומית גוף האדם.

המשתמש יכול לבצע את התרגילים באימון בכל סדר שירצה. סדר ביצוע התרגילים באימון דינמי כדי למנוע הגבלה על המתאמן. למשל, אם המתאמן נמצא בחדר כושר וההליכון תפוס ע"י מתאמן אחר, הוא יוכל לבחור תרגיל אחר באימון שלא דורש שימוש במתקן זה.

האפליקציה משמשת ככלי מעקב למתאמן אחרי הביצועים שלו בכל התרגילים.

המשתמש יכול להבין את מצבו ביחס לתכניות הקודמות ולהבחין במגמות של שיפור/האטה באופן ברור. זאת דרך גרפים, עמודות ופרמטרים הקשורים לתרגיל.

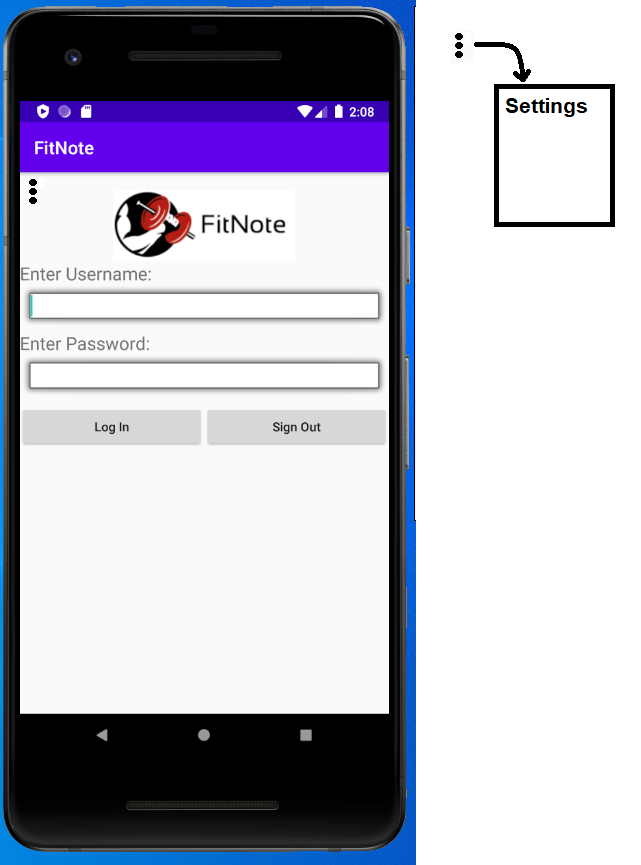
האפליקציה מציעה למשתמש לבצע תרגילי כוח בהדגש על אזורים מסוימים בגוף.

יהיו המלצות לכמות הביצועים בתרגילים אך המשתמש יכול להכניס את הכמות שעשה בפועל בזמן שלו.

מספר החזרות של כל תרגיל מצוין בהמלצת האפליקציה.

המשתמש מאשר בלחיצת כפתור על ביצוע המטלה. התוכנה עוקבת אחרי הדיווחים של המתאמן וזוכרת את ביצועיו.

לאחר שימוש מסוים באפליקציה התוכנה תספק למשתמש עידודים המבוססים על הסטטיסטיקה שלו. אם ניכר שיפור בביצוע המתאמן בתרגיל מסוים, יינתן לו עידוד ורמת התרגילים תעלה. אולם, במידה שהמתאמן מתקשה בביצוע התרגיל וביצועו חלקי, התוכנה תדע להקל עליו ובהדרגה תנסה לסייע לו להשתפר שוב.

מסכים:

מסכים ראשונים – רישום ראשון ומסך כניסה:

מסך הפתיחה:

מסך זה דורש שם משתמש וסיסמה של משתמש הקיים במאגר הנתונים כדי להיכנס לתוכנה.

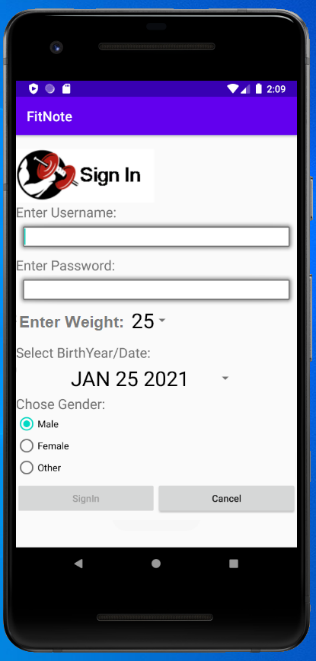
השלוש נקודות למעלה מובילות לתת מסך קטן שם יש אפשרות ללכת למסך הגדות המשתמש.

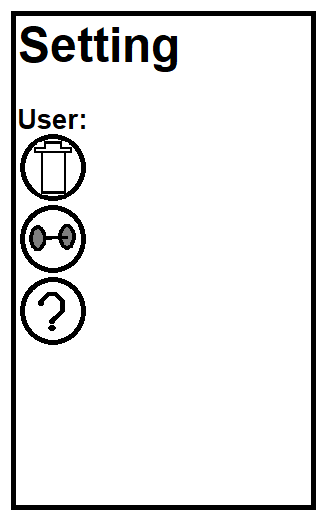
למטה יש שני כפתורים: כפתור רישום וכפתור רישום משתמש חדש.

כפתור הרישום למשתמש החדש פותח תת מסך בו הוא יידרש להכניס את כל הפרטים הבאים: שם משתמש, סיסמא, תאריך לידה, מין, משקל.

כפתור הרישום דורש שם משתמש וסיסמא כדי להיכנס לאפליקציה.

מסך הרישום:

במסך הרישום המשתמש בעת רישום ראשון מכניס את שמו, הסיסמא שלו, תאריך לידתו, מינו ומשקלו.

מסך הגדרות המשתמש:

במסך זה יש שלושה כפתורים.

כפתור הזבל מוחק את המשתמש הרשום כעת.

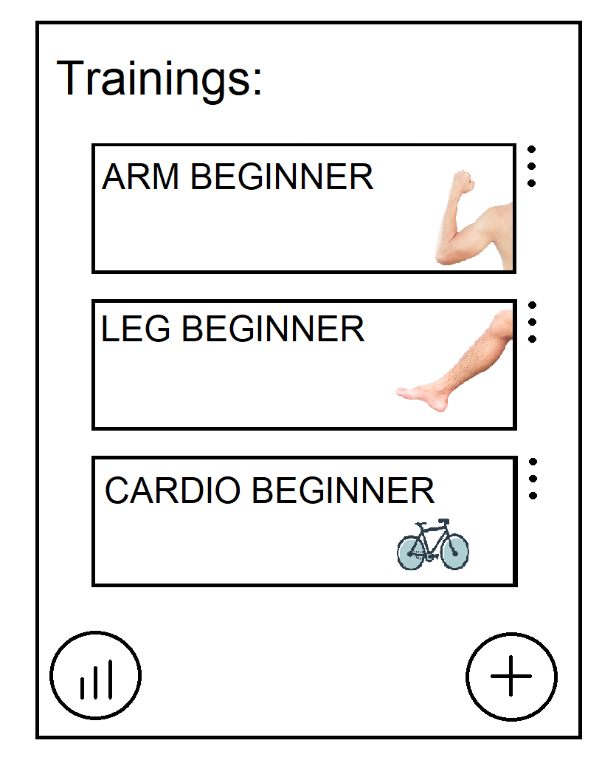
כפתור המשקולת נותן אפשרות לשינוי המשקל של המשתמש.

כפתור סימן השאלה נותן עזרה למשתמש באפליקציה. יש פרטים על האפליקציה ועל הנתונים שהיא לוקחת המשתמש במהלך תכניות התרגילים שלו.

ברגע שהמשתמש הכניס את פרטיו לאפליקציה הוא רשאי להיכנס עם שם משתמש וסיסמא לתכנית התרגילים שלו.

בהתחלה מופיעים תכניות שהתוכנה מציעה למשתמש להתנסות בהם. המשתמש יכול להכין לעצמו תכניות תרגילים המעוצבים בהדגש לשרירים אותם הוא רוצה לחזק בתרגילים.

מסך ראשי – מסך תכניות האימון:

במסך הראשי מופיעים כל התרגילים במאגר הנתונים של האפליקציה. יש תרגילים שנמצאים בתוכנה עצמה מראש כמו אלו שמופיעים בשרטוט.

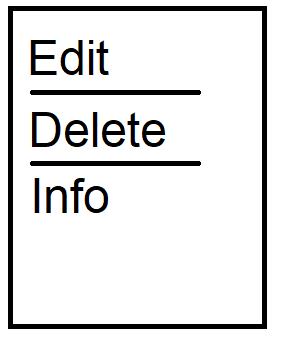
למעלה מופיעה כותרת סטטית עם השם "תכניות:".

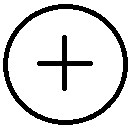
כפתור הפלוס למטה מימין לוקח את המשתמש למסך בו הוא יבנה את האימון בהתאם לשרירים שהוא רוצה לחזק.

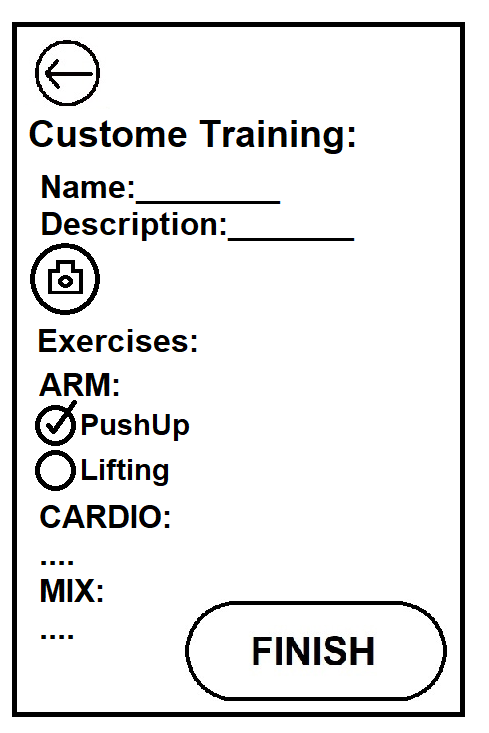
כפותר העמודות למטה משמאל לוקח את המשתמש למסך בו יש סיכום של ניתוח המידע והסטטיסטיקה על כל התרגילים שהמשתמש עשה.

לחיצה על המלבן בו מופיע האימון לוקח את המשתמש למסך בו הוא רואה רשימה של כל התרגילים באימון והוא יוכל גם לעשות את האימון ולראות נתונים על ביצועי האימון.

לחיצה על שלוש הנקודות ליד כל מלבן של אימון נותן למשתמש שלושה אופציות: שינוי האימון, מחיקה ותיאור.

מיני מסך שמופיע באת לחיצה על שלוש הנקודות:

מסך הוספת אימון:

מסך זה מופיע אחרי לחיצה על כפתור הפלוס במסך הראשי בו מופיעים כל התכניות.

למעלה מופיעה כותרת סטטית עם השם "אימון מותאם אישית:".

מתחת לכותרת מופיעים שני מקומות להכנסת טקסט שבאחד מופיע שם האימון ובשני תיאור של האימון.

לחיצה על כפתור התמונה נותן אופציה לשיוך תמונה לאימון המותאם אישית. אפשר לא להכניס תמונה ואז האפליקציה תכניס למאגר הנתונים תמונה ברירת מחדל.

מתחת לכותרת "תרגילים:" מופיעה רשימה של כל התרגילים שיש במאגר הנתונים. כל התרגילים משויכים לתת כותרות שמתארות את חלק הגוף שהם בעיקר מחזקים בעת האימון.

ליד כל תרגיל מופיע עיגול שלחיצה עליו מאשרת שהוא יכנס לאימון. אחרי הלחיצה על העיגול מופיעה ווי בתור אישור על הלחיצה.

במסך משמאל למעלה מופיע כפתור חזרה למסך תכניות התרגילים. לחיצה עליו מבטלת את תהליך הכנת האימון.

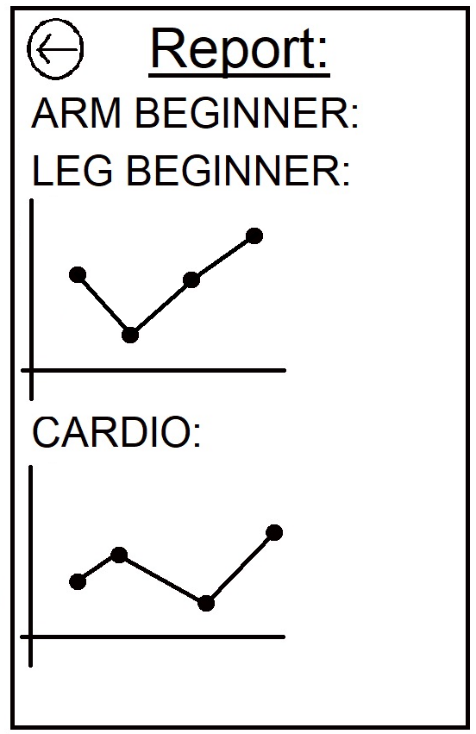
למטה מימין מופיע כפתור סיום "Finish" שלחיצה עליו מסיימת את תהליך הכנת האימון האישי ומכניסה את האימון לזיכרון התוכנה. האימון החדש מופיע במסך הראשי בו מופיעים כל התכניות, ולשם יוחזר המשתמש ישר אחרי הלחיצה על כפתור הסיום.

לחיצה על כפתור הסיום של הכנסת התוצאות תתפשר אך ורק אחרי שכל הפרטים לעיל הוכנסו. לחיצה כל כפתור הסיום תיקח את המשתמש למסך דיווח הנתונים אחרי ביצוע התרגיל.



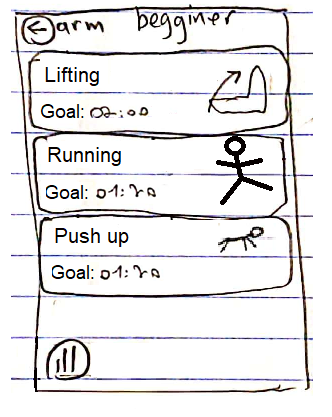
מסך סטטיסטיקות של כל התרגילים בתכניות:

מסך זה מופיע בלחיצה על כפתור העמודות במסך הראשי בו מופיעים כל התכניות.

מסך זה מציג סיכום של ניתוח כל המידע על התכניות והתרגילים שביצע המשתמש בהם.

אפשר לראות מתחת לכל כותרת את כל המידה הסטטיסטי הקשור לאימון הזה והתרגילים שבו.

במסך משמאל למעלה מופיע כפתור חזרה למסך תכניות האימון.

מסך התרגילים של האימון הנבחר במסך הראשי:

המסך הזה מופיע בלחיצה על אימון מסוים במסך תכניות האימון.

במסך משמאל למעלה מופיע כפתור חזרה למסך תכניות האימון.

שם האימון מופיע בכותרת למעלה.

מאחורי הכותרת מופיע התמונה של האימון.

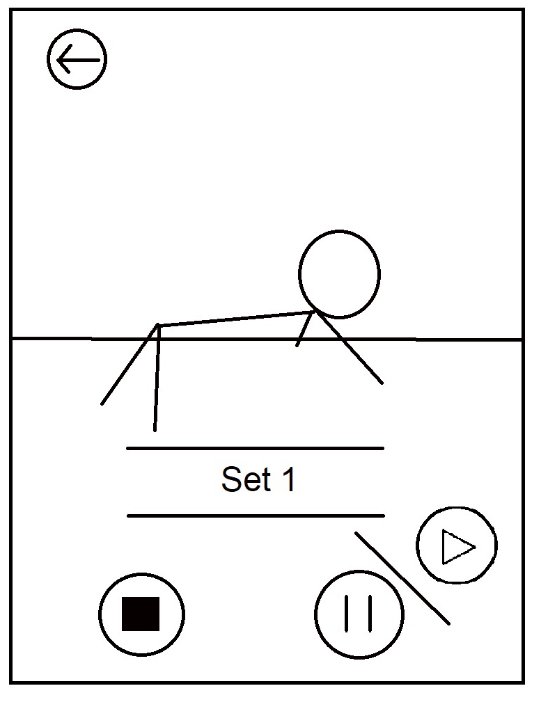
ברשימה באמצע המסך מופיעים כל התרגילים ששייכים לאותו אימון.

בכל תרגיל מופיע שמו, תמונה שמתארת אותו והזמן המומלץ לביצוע התרגיל.

סדר ביצוע התרגילים אינו משנה. המשתמש חופשי לבצע את התרגילים באיזה סדר שהוא רוצה.

לחיצה על תרגיל מסוים תיקח את המשתמש למסך בו הוא יתחיל את תהליך ביצוע התרגיל.

הכפתור למטה משמאל עם העמודות ייקח את המשתמש למסך בו מופיע סיכום של כל ניתוח המידע והסטטיסטיקה על האימון הנבחר ספציפית.



מסך ביצוע התרגיל:

מסך זה מופיע אחרי לחיצה על תרגיל במסך התרגילים של האימון.

במסך משמאל למעלה מופיע כפתור חזרה למסך הקודם. לחיצה עליו מבטלת את עשיית התרגיל ומחזירה את המשתמש חזרה למסך התרגילים של האימון.

במסך מופיע תמונה של התרגיל בעשייה. כאשר לוחצים על כפתור המשולש השוכב האנימציה פועלת והטיימר מתחיל. כשהאנימציה מתחילה סימן שאפשר להתחיל את התרגיל.

למטה מופיע כותרת עם מספר הסט שעושים.

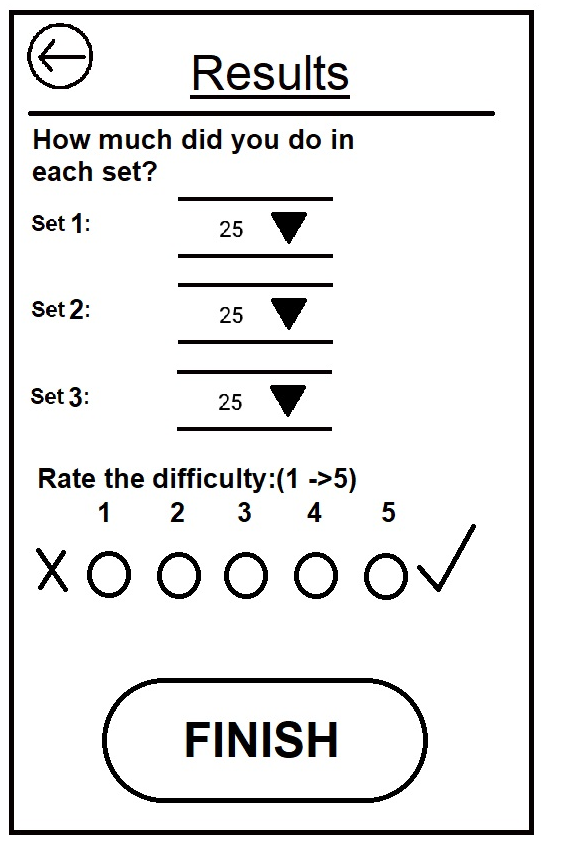
הכפתור מימין למטה הוא כפתור אחד שלפני התחלת התרגיל הוא משולש שוכב, כפתור ההפעלה, וברגע שלוחצים עליו התרגיל מתחיל – התמונה הגדולה הופכת לאנימציה והמשתמש יכול להתחיל את התרגיל. ברגע שהכפתור נלחץ הוא הופך לשני קווים מקבילים – כפתור ההשהיה. כשכפתור ההשהיה נלחץ הטיימר עוצר. ההשהיה המתבצעת ע"י לחיצה על כפתור ההשהיה עוצרת את הטיימר עד שכפתור ההפעלה נלחץ שוב.

תחילת התרגיל והפעלתו משמע הפעלת טיימר בתוך האפליקציה עצמה. זמן הביצוע הוא אחד הפרמטרים בו נעזר בעת ניתוח ביצוע התרגיל של המתאמן. הזמן שאותו ננתח הוא הזמן שהטיימר מראה בפועל. הטיימר מפסיק כשכפתור הריבוע השחור נלחץ.

לחיצה על הכפתור משמאל למטה, כפתור הריבוע השחור, עוצרת את ביצוע התרגיל. בלחיצה זו הטיימר עוצר. אחרי לחיצה על כפתור הריבוע השחור ימשיך המשתמש לעשות את שאר הסטים של התרגיל. ברגע שיסיים המשתמש את הסט האחרון הוא ימשיך למסך הכנסת תוצאות התוכנה ושביאות רצון. התוכנה מקבלת את הזמן שלקח למתאמן לבצע התרגיל. לכן במסך הבא הוא יכניס את מספר הביצועים שהוא הספיק באותו זמן ואת רמת הקושי של התרגיל.

לחיצה על הריבוע השחור מסיימת את ביצוע התרגיל. שהמשתמש לוחץ על הכפתור הזה הוא מאשר את סיום ביצוע הסט הזה של התרגיל. הטיימר באותו הזמן גם עוצר והזמן שהטיימר מראה הוא זמן ביצוע התרגיל בפועל. הזמן הזה ישמר בבסיס הנתונים וישמש לתוכנה פרמטר לניתוח מצבו של המתאמן.

ביצוע התרגיל מסתיים אחרי שהמתאמן עבר על כל הסטים שיש לאותו התרגיל.

נתון הזמן שלקח לביצוע כל סט ישמר בבסיס הנתונים. השוואה בין הזמנים בימים שונים תעזור לתוכנה לקבוע אם יש מגמה של שיפור/האטה אצל המתאמן. התוכנה בעצמה תעשה שינויים בהתאם למגמה. אם המשתמש השתפר בביצוע התרגיל וזמן הביצוע שלו קצר התוכנה תקטין את הזמן המומלץ לביצוע התרגיל. לעומת זאת, אם המשתמש מתקשה בביצוע התרגיל וזמן הביצוע שלו גדל אז התוכנה תגדיל את הזמן המומלץ לביצוע התרגיל.

מסך הכנסת תוצאות לתוכנה ושביאות רצון:

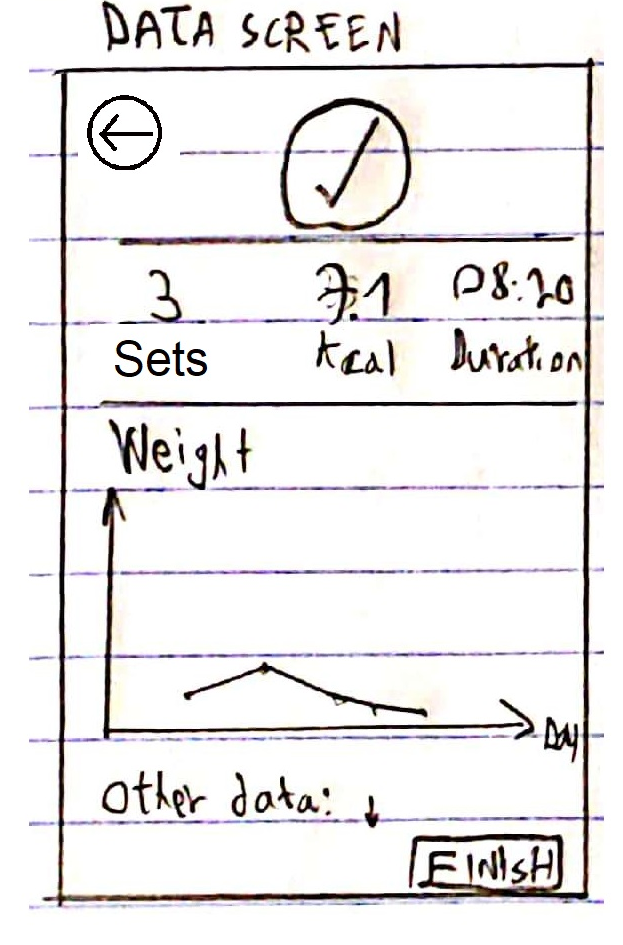
מסך זה מופיע לאחר סיום ביצוע התרגיל. המסך יבקש מהמתאמן להכניס בגלגלות את מספר הביצועים שביצע בכל סט בתרגיל. מספר הסטים תלוי בתרגיל שבוצע.

לחיצה על חץ החזרה יחזיר את המשתמש למסך התרגילים ויבטל את תהליך עשיית התרגיל.

אחרי זה יש בקשה לדרג את הקושי שהיה בביצוע התרגיל מאחת ועד חמש. כאשר אחת זה אומר שהביצוע היה קשה מאוד. בהדרגה הקושי של התרגיל יורד עד שחמש זה אומר שהביצוע היה קל למתאמן.

לחיצה על כפתור הסיום של הכנסת התוצאות תתפשר אך ורק אחרי שכל הפרטים לעיל הוכנסו. לחיצה כל כפתור הסיום תיקח את המשתמש למסך דיווח הנתונים אחרי ביצוע התרגיל.

דירוג הקושי של התרגיל, מספר הביצועים של המתאמן בכל סט והזמן שלקח בפועל למתאמן לבצע את התרגיל הם הפרמטרים שיעזרו לנתח את מצבו של המתאמן. התוכנה תוכל לדעת האם להקשות או להקל על המתאמן באותו תרגיל.



מסך דיווח הנתונים אחרי ביצוע תרגיל:

מסך זה מופיע אחרי ביצוע מלא של התרגיל והכנסת תוצאות ביצוע התרגיל.

למעלה מופיע ווי בתור אישור על סיום ביצוע התרגיל.

באמצע מופיעים: מספר הסטים שביצע המשתמש, מספר הקלוריות שירדו בעת ביצוע התרגיל והזמן הכולל שלקח לבצע את התרגיל.

מתחת לנתונים אלה מופיעים גרפים המספקים ניתוח וויזואלי על מצבו של המתאמן באותו התרגיל שביצע כרגע ביחס לפעמים קודמות שביצע את התרגיל.

לחיצה על כפתור הסיום למטה מימין או על החץ מימין למעלה מחזירה את המתאמן למסך התרגילים בו מופיעים כל התרגילים של האימון.

תפקיד בסיס הנתונים במסכים:

בחלק זה אני אסביר מה תפקידו של בסיס הנתונים בשמירת נתוני המשתמש עבור כל מסך באפליקציה.

קודם כל נתחיל מהגדרת הטבלאות שיהיו בבסיס הנתונים.

שם הטבלה : User – טבלת המשתמשים:

מטרה : שמירת פרטים אישיים של המשתמש (שדה=עמודה בטבלה).

|  |  |  |  |
| --- | --- | --- | --- |
| שם שדה בעברית | שם שדה באנגלית | סוג הנתונים | הערות |
| שם משתמש | username | TEXT | **PRIMARY KEY** |
| סיסמא | password | TEXT |  |
| משקל | weight | FLOAT |  |
| שנת לידה | birthyear | TEXT |  |
| מין | gender | TEXT |  |

שם הטבלה : BodyPart – טבלת איברי הגוף:

מטרה : שמירת כל סוגי האיברים בגוף עליהם התרגילים עובדים.

|  |  |  |  |
| --- | --- | --- | --- |
| שם שדה בעברית | שם שדה באנגלית | סוג הנתונים | הערות |
| קוד האיבר | bodyPartCode | INT | **PRIMARY KEY** |
| שם האיבר | name | TEXT |  |
| תמונה של האיבר | bodyPartPic | INT |  |

שם הטבלה : Exercise – טבלת התרגילים:

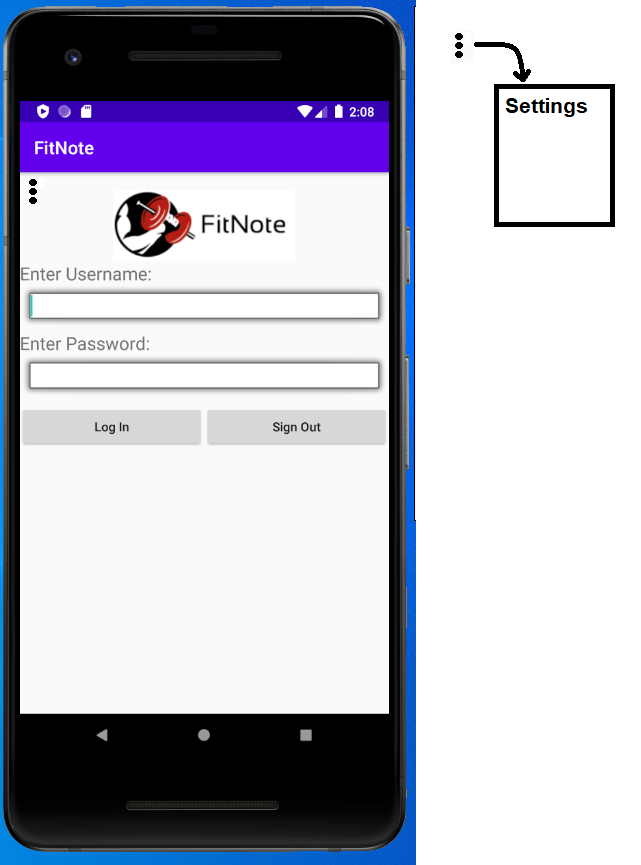
מטרה : שמירת כל סוגי התרגילים לאימון הגופני.

|  |  |  |  |
| --- | --- | --- | --- |
| שם שדה בעברית | שם שדה באנגלית | סוג הנתונים | הערות |
| קוד התרגיל | exerciseCode | INT | **PRIMARY KEY** |
| שם התרגיל | name | TEXT |  |
| סטים | sets | INT |  |
| מספר הביצועים | amount | INT |  |
| זמן עשייה של תרגיל | duration | TEXT |  |
| רמת הקושי | difficulty | TEXT |  |
| תאריך ביצוע התרגיל | exerciseDate | TEXT |  |
| קוד איבר בגוף | codeBodyPart | INT |  |
| תמונה של התרגיל | exercisePic | INT |  |
| תיאור ביצוע התרגיל | detail | TEXT |  |
| המלצה | recommendation | TEXT |  |

שם הטבלה : Program – טבלת תכנית התרגילים:

מטרה : שמירת כל התכניות. אל כל תכנית משויכים תרגילים מסוימים.

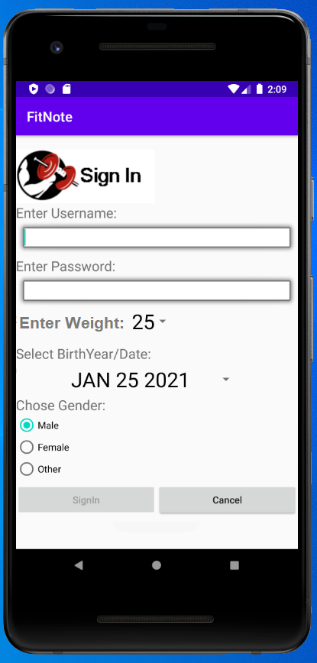
|  |  |  |  |
| --- | --- | --- | --- |
| שם שדה בעברית | שם שדה באנגלית | סוג הנתונים | הערות |
| שם המשתמש אליו האימון שייך | userName | TEXT |  |
| שם האימון | programName | TEXT |  |
| תמונה של האימון | programPic | INT |  |
| קוד תרגיל | exerciseCode | INT |  |
| תאריך ביצוע האימון | programDate | TEXT |  |
| מידע על האימון | description | TEXT |  |



מסך הפתיחה:

בכניסה ראשונה המשתמש חייב להיכנס למסך הרישום דרך הכפתור מימין למטה.

אחרי שהמשתמש הלך למסך הרישום הראשוני והמידע עליו רשום בבסיס הנתונים הוא יוכל להמשיך למסך תכניות האימון.

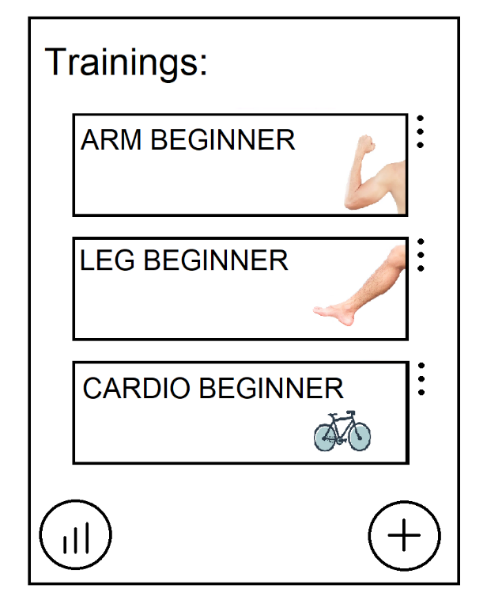
מסך הרישום:

במסך הרישום המשתמש מכניס את שם המשתמש, הסיסמא, משקלו, תאריך לידתו ומינו.

כל המידע שמוכנס עובר לרשומה של משתמש חדש שנשלחת לטבלת המשתמשים כמשתמש חדש.

עם המשתמש החדש הרשום יוכל המתאמן לחזור למסך הפתיחה ולהיכנס עם שם משתמש וסיסמא.

מסך ראשי – מסך תכניות האימון:

כל הרשומות של התכניות המשויכים לאותו המשתמש שנרשם כרגע מוצאים מבסיס הנתונים. במסך הראשי מופיעים ברשימה אחת נתונים מרשומות של טבלת התכניות שהוצאו כרגע.

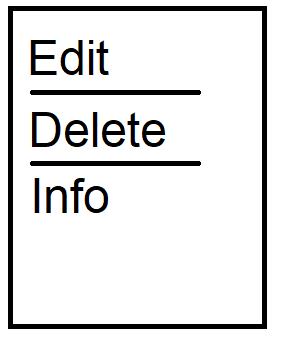
כל אימון ברשימה המוצגת מופיע עם שם האימון ותמונה של האימון מהטבלה של התכניות.

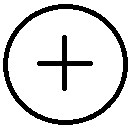
שם אימון נלקח מהשדה "שם האימון" והתמונה של האימון מובאת מהשדה "תמונה של האימון".

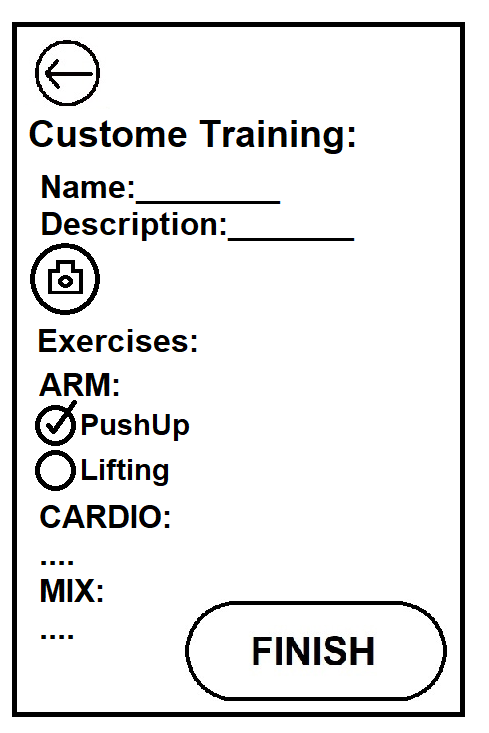
בלחיצה על שלוש הנקודות ליד כל אימון מופיע המסך הזה:

לחיצה על כפתור העריכה נותן למשתמש לערוך את האימון הנבחר. העריכה המתבצעת עורכת את האימון הנבחר בטבלת התכניות.

לחיצה כל כפתור המחיקה מוחק את האימון ואת הנתונים המשויכים לו מבסיס הנתונים.

לחיצה על כפתור המידע נותן את המידע על האימון משדה "מידע על התכנית" של תכנית זה מטבלת התכניות.



מסך הוספת האימון:

מסך זה מופיע אחרי לחיצה על כפתור הפלוס במסך הראשי בו מופיעים כל התכניות. במסך זה מתחיל תהליך של הוספת אימון מותאם אישית.

המשתמש מכניס במסך זה את הפרטים על האימון המותאם אישית שלו הכוללים: שם האימון, תיאור האימון, תמונה לאימון (לא חייב, יוכנס תמונה ברירת מחדל אם לא ישימו תמונה אחרת)ובחירת התרגילים המשויכים לאימון מטבלת התרגילים.

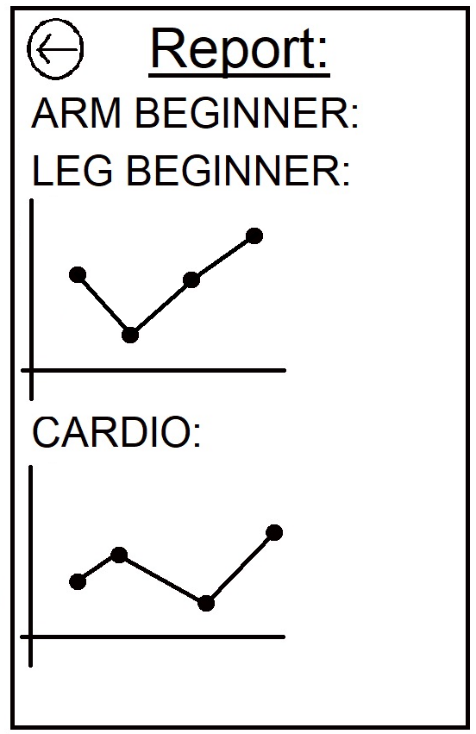
לחיצה על כפתור הסיום מכינה רשומה של אימון חדש המועברת לטבלת התכניות הכוללת את השם והתיאור והתמונה (אם שמו) של האימון המותאם אישית, וכל התרגילים שסומנו בווי יכנסו לרשומה של האימון החדש.

מסך הסטטיסטיקות של כל התכניות והתרגילים שלהם:

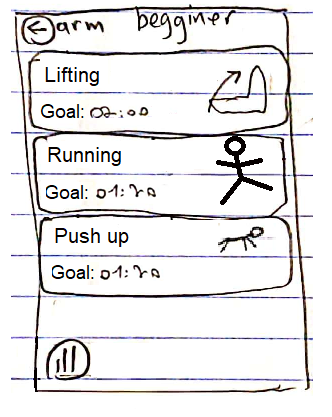
מסך זה מופיע בלחיצה על כפתור העמודות במסך הראשי בו מופיעים כל התכניות.

מסך זה פורש בכותרות את הנתונים האישיים של המשתמש עבור כל התכניות שביצע והביצועים בכל התרגילים בהם.

הנתונים נלקחים מטבלת התכניות ומהתרגילים המשויכים לכל אימון.

יש כותרת לכל אימון בטבלת התכניות השייך למשתמש. מתחת לכל כותרת יש סיכום עבור כל אחד מהתרגילים באימון (אם יש נתונים עבור התרגיל) בו יש גרפים, צירי זמן ומגמה שהתוכנה שמה לב אליה של שיפור או האטה. הסטטיסטיקה מבוססת על מספר הביצועים עבור כל סט, זמן עשייה של כל תרגיל, התאריך שבוצע בו התרגיל והמשוב שלו על רמת הקושי של התרגיל הנלקחים מהשדות "מספר הביצועים", "סטים", "תאריך ביצוע התרגיל", "זמן עשייה של תרגיל" ו-"רמת קושי" מטבלת התרגילים.

מסך התרגילים של האימון הנבחרת במסך הראשי:

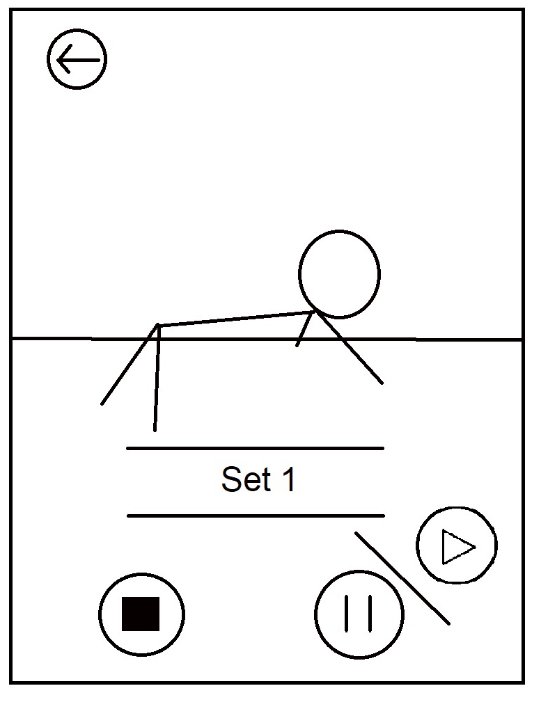
מסך זה מופיע בלחיצה על תכניות במסך הראשי בו מופיעים כל התכניות.

כל הרשומות של התרגילים המשויכים לאימון שנבחר מוצאים מבסיס הנתונים. במסך התרגילים מופיעים ברשימה אחת נתונים מרשומות של טבלת התרגילים שהוצאו כרגע.

כל תרגיל ברשימה המוצגת מופיע עם שם התרגיל, זמן מומלץ לעשיית התרגיל ותמונה של התרגיל.

שם תרגיל נלקח מהשדה "שם התרגיל", הזמן המומלץ לעשייתה התרגיל נלקח מהשדה "המלצה" והתמונה של התרגיל מובאת מהשדה "תמונה של התרגיל" מטבלת התרגילים עבור כל תרגיל ברשימה.

הכפתור למטה משמאל עם העמודות ייקח את המשתמש למסך בו מופיע סיכום של כל ניתוח המידע והסטטיסטיקה על האימון הנבחר ספציפית.



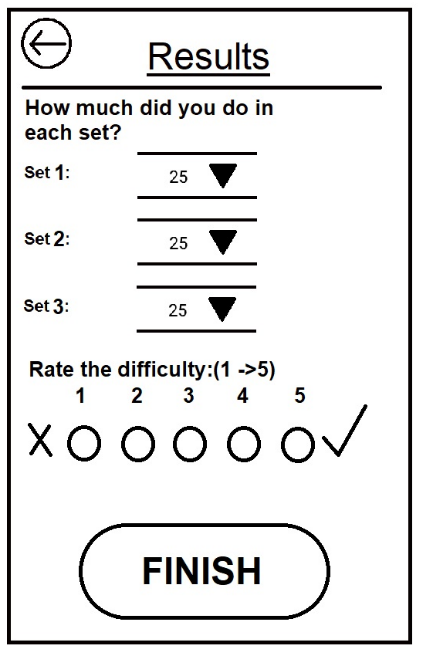
מסך ביצוע התרגיל:

במסך זה נלקחים התמונה של התרגיל, שתהיה אנימציה בהפעלת התרגיל והתרגיל ימשיך כמספר הסטים הכתובים בתרגיל הנבחר במסך התרגילים.

התמונה של התרגיל מובאת מהשדה "תמונה של התרגיל" ומספר הסטים של התרגיל נלקח מהשדה "סטים" מטבלת התרגילים.

לחיצה על הכפתור משמאל למטה, כפתור הריבוע השחור, עוצרת את ביצוע התרגיל. בלחיצה זו הטיימר עוצר. אחרי לחיצה על כפתור הריבוע השחור ימשיך המשתמש לעשות את שאר הסטים של התרגיל. ברגע שיסיים המשתמש את הסט האחרון הוא ימשיך למסך הכנסת תוצאות התוכנה ושביאות רצון.

התוכנה מקבלת את הזמן שלקח למתאמן לבצע את התרגיל. כולו ואת הזמן הזה היא מכניסה לשדה "זמן עשייה של תרגיל" בטבלת התרגילים.

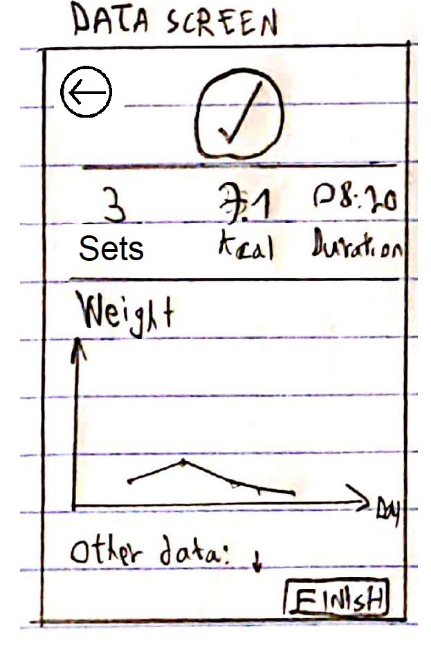


מסך הכנסת תוצאות לתוכנה ושביאות רצון:

במסך זה מכניס המתאמן עבור כל סט את מספר הביצועים שהוא עשה ואת רמת הקושי האישית שלו בביצוע התרגיל.

מספר הביצועים הכולל יכנס לשדה "מספר הביצועים" ורמת הקושי שהכניס המשתמש תיכנס לשדה "רמת הקושי" עבור תרגיל הזה בטבלת התרגילים.

מסך דיווח הנתונים אחרי ביצוע תרגיל:

מסך זה מופיע אחרי ביצוע מלא של התרגיל והכנסת תוצאות ביצוע התרגיל.

למעלה מופיע ווי בתור אישור על סיום ביצוע התרגיל.

באמצע מופיעים: מספר הסטים שביצע המשתמש, מספר הקלוריות שירדו בעת ביצוע התרגיל והזמן הכולל שלקח לבצע את התרגיל.

מספר הסטים שביצע המשתמש נלקח מהשדה "סטים" מהתרגיל הנעשה מטבלת התרגילים.

מספר הקלוריות שירדו בעת ביצוע התרגיל יחשב לפי זמן הביצוע של התרגיל מהשדה "זמן עשייה של תרגיל" מהתרגיל הנעשה מטבלת התרגילים.

הזמן הכולל שלקח לבצע את התרגיל יילקח מהשדה "זמן עשייה של תרגיל" מהתרגיל הנעשה מטבלת התרגילים.

מתחת לנתונים אלה מופיעים גרפים המספקים ניתוח וויזואלי על מצבו של המתאמן באותו התרגיל שביצע כרגע ביחס לפעמים קודמות שביצע את התרגיל.

הסטטיסטיקה מבוססת על מספר הביצועים עבור כל סט, זמן עשייה של כל תרגיל, התאריך שבוצע בו התרגיל והמשוב שלו על רמת הקושי של התרגיל הנלקחים מהשדות "מספר הביצועים", "סטים", "תאריך ביצוע התרגיל", "זמן עשייה של תרגיל" ו-"רמת קושי" מטבלת התרגילים.

לחיצה על כפתור הסיום למטה מימין או על החץ מימין למעלה מחזירה את המתאמן למסך התרגילים בו מופיעים כל התרגילים של האימון.

## כל המחלקות והקבצים בפרויקט

## AndroidManifest

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.fitnote13022021">  
  
 <uses-permission android:name="android.permission.READ\_CONTACTS" />  
  
 <uses-feature  
 android:name="android.hardware.Camera"  
 android:required="true" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@drawable/logo\_only\_pic"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/Theme.FitNote13022021">  
 <activity android:name=".EditUserActivity"></activity>  
 <activity android:name=".ManagerUsersActivity" />  
 <activity  
 android:name=".InformationActivity"  
 android:label="@string/title\_activity\_information"  
 android:theme="@style/Theme.FitNote13022021.NoActionBar" />  
 <activity  
 android:name=".ViewExercisesResultsActivity"  
 android:label="Results" />  
 <activity  
 android:name=".SettingsActivity"  
 android:label="Settings" />  
 <activity  
 android:name=".WhatsAppSendActivity"  
 android:label="Whats App Send" />  
 <activity  
 android:name=".ShareActivity"  
 android:label="Share With Friends" />  
 <activity  
 android:name=".StatisticsActivity"  
 android:label="Statistics" />  
 <activity  
 android:name=".FeedbackActivity"  
 android:label="feedback" />  
 <activity  
 android:name=".ExecuteExerciseActivity"  
 android:label="execute-training" />  
  
 <service android:name=".MusicAndTimerService"  
 android:exported="false"/>  
  
 <activity  
 android:name=".AddExerciseActivity"  
 android:label="AddExercise" />  
 <activity  
 android:name=".ProgramUserActivity"  
 android:label="User-Program" />  
 <activity  
 android:name=".RegisterActivity"  
 android:label="Register" />  
 <activity  
 android:name=".MainScreenActivity"  
 android:label="FitNote">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity> *<!-- Register the Reminder Receiver -->* <receiver  
 android:name=".ReminderBroadcast"  
 android:enabled="true" />  
 </application>  
  
</manifest>

## Java Folder Classes

## AddExerciseActivity

package com.example.fitnote13022021;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.SearchView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class AddExerciseActivity extends AppCompatActivity implements View.OnClickListener, AdapterView.OnItemClickListener {  
  
 DataBaseHelper dataBaseHelper;  
  
 ListView listViewExercises;  
  
 ArrayList<Exercise> exercises;  
  
 ArrayList<Exercise> filteredExercises;  
  
 ExerciseAdaptor exerciseAdaptor;  
  
 Button btCancel;  
  
 SearchView searchViewExerciseList;  
  
 String activeUserName;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_add\_exercise*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 listViewExercises = findViewById(R.id.*listViewExercises*);  
 btCancel = findViewById(R.id.*btCancel*);  
 searchViewExerciseList = findViewById(R.id.*searchViewExerciseList*);  
  
 exercises = dataBaseHelper.getExercises();  
  
 *//setting filteredExercises to be exercises* filteredExercises = exercises;  
  
 exerciseAdaptor = new ExerciseAdaptor(exercises, this);  
  
 listViewExercises.setAdapter(exerciseAdaptor);  
  
 *//initialize the searchWidget in order to filter exercises* initSearchWidgets();  
  
 *//settings onClick Listeners* listViewExercises.setOnItemClickListener(this);  
  
 btCancel.setOnClickListener(this);  
  
 *//getting userName from intent* Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 }  
  
 *//method to go back to ProgramUserActivity* @Override  
 public void onClick(View v) {  
  
 if (v == btCancel){  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
 }  
  
 *//method to add this specific exercise* @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 *//gets the clicked exercise* Exercise chosenExercise = filteredExercises.get(position);  
  
 int exerciseID = chosenExercise.getExerciseID();  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to add this exercise?");  
  
 *//Setting the builder's layout inflater* View alertView = getLayoutInflater().inflate(R.layout.*exercise\_with\_description\_layout*, null);  
  
 *//getting views from layout* ImageView exerciseImage = alertView.findViewById(R.id.*imageViewOfExerciseInDescription*);  
 TextView txtExerciseNameInDescription = alertView.findViewById(R.id.*txtExerciseNameInDescription*);  
 TextView txtExerciseDescription = alertView.findViewById(R.id.*txtExerciseDescription*);  
  
 *//getting the whole chosen exercise information* Exercise exerciseChosen = null;  
  
 for (int i=0; i<exercises.size(); i++) {  
 Exercise exercise = exercises.get(i);  
 if(exercise.getExerciseID() == exerciseID){  
 exerciseChosen = exercise;  
 }  
 }  
  
 *//Setting the imageView to show exercise image* exerciseImage.setImageResource(exerciseChosen.getExercisePic());  
  
 *//Setting textViews to show exercise details* txtExerciseNameInDescription.setText(exerciseChosen.getExerciseName());  
 txtExerciseDescription.setText(exerciseChosen.getExerciseDetail());  
  
  
 *//Setting the view to be in builder of alert dialog* builder.setView(alertView);  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 UserExercise newUserExercise = new UserExercise(activeUserName, exerciseID);  
  
 *//method to insert UserExercise  
 //public boolean insertUserExercise(UserExercise userExercise)  
 //so we wiil turn this chosen exercise into a new userExercise  
 //the builder - (String userName, int exerciseID)* dataBaseHelper.insertUserExercise(newUserExercise);  
  
 Intent intent = new Intent(AddExerciseActivity.this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
  
 }  
  
 *//initialize the searchWidget in order to filter exercises* public void initSearchWidgets(){  
  
 SearchView searchView = (SearchView)searchViewExerciseList;  
  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 *//this method is called every time a user  
 //puts in any character into the search view  
 //literally any chang :D* @Override  
 public boolean onQueryTextChange(String newText) {  
  
 filteredExercises = new ArrayList<Exercise>();  
  
 *//same as a regular for lop* for (Exercise exercise: exercises)  
 {  
 *//if an exercise has one of the letter in the written text* if(exercise.getExerciseName().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredExercises.add(exercise);  
 }  
 }  
  
 ExerciseAdaptor exerciseAdaptor = new ExerciseAdaptor(filteredExercises, AddExerciseActivity.this);  
  
 listViewExercises.setAdapter(exerciseAdaptor);  
  
 return false;  
 };  
 });  
  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(@NonNull MenuItem item) {  
 return super.onOptionsItemSelected(item);  
  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
}

## Admin

package com.example.fitnote13022021;  
  
public class Admin extends User{  
  
 private boolean isSuperAdmin;  
  
 *//constructors  
 //constructor for all info* public Admin(String userName, String userPassword,boolean isSuperAdmin , int userWeight, int userHeight, String userBirthDate, String userGender, String profilePic) {  
  
 super(userName, userPassword, userWeight, userHeight, userBirthDate, userGender, profilePic);  
  
 this.isSuperAdmin = isSuperAdmin;  
  
 }  
  
 *//constructor for user* public Admin(User user, boolean isSuperAdmin ) {  
  
 super(user.getUserName(), user.getUserPassword(), user.getUserWeight(), user.getUserHeight(), user.getUserBirthDate(), user.getUserGender(), user.getProfilePic());  
  
 this.isSuperAdmin = isSuperAdmin;  
  
 }  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return super.toString()+  
 "\n" +  
 "Admin{" +  
 "isSuperAdmin=" + isSuperAdmin +  
 '}';  
 }  
  
 *//Getters and Setters* public boolean isSuperAdmin() {  
 return isSuperAdmin;  
 }  
  
 public void setSuperAdmin(boolean superAdmin) {  
 isSuperAdmin = superAdmin;  
 }  
  
 public int getUserLevel(){  
 if (isSuperAdmin())  
 return 2;  
 else  
 return 1;  
 }  
}

## ContactListAdapter

package com.example.fitnote13022021;  
  
import android.app.Activity;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import java.util.ArrayList;  
  
public class ContactListAdapter extends RecyclerView.Adapter<ContactListAdapter.ViewHolder> {  
  
 *//Initialize variable* Activity activity;  
 ArrayList<ContactModel> arrayList;  
 RecycleViewClickListener listener;  
  
 *//Create constructor* public ContactListAdapter(Activity activity, ArrayList<ContactModel> arrayList, RecycleViewClickListener listener){  
 this.activity = activity;  
 this.arrayList = arrayList;  
 this.listener = listener;  
 notifyDataSetChanged();  
 }  
  
 @NonNull  
 @Override  
 public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {  
 *//Initialize variable* View view = LayoutInflater.*from*(parent.getContext())  
 .inflate(R.layout.*item\_contact*,parent,false);  
 *//Return view* return new ViewHolder(view);  
 }  
  
 @Override  
 public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
 *//Initialize contact model* ContactModel model = arrayList.get(position);  
  
 *//Set name* holder.tv\_nameContact.setText(model.getName());  
  
 *//Set number* holder.tv\_numberContact.setText(model.getNumber());  
  
 }  
  
 @Override  
 public int getItemCount() {  
 *//Return array list size* return arrayList.size();  
 }  
  
 public interface RecycleViewClickListener {  
 void onClick(View v, int position);  
 }  
  
 public class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
  
 *//Initialize variable* TextView tv\_nameContact, tv\_numberContact;  
  
 public ViewHolder(@NonNull View itemView) {  
 super(itemView);  
 *//Assign variable* tv\_nameContact = itemView.findViewById(R.id.*tv\_nameContact*);  
 tv\_numberContact = itemView.findViewById(R.id.*tv\_numberContact*);  
 itemView.setOnClickListener(this);  
  
 }  
  
 *// A method that is triggered when you click an item in the list* @Override  
 public void onClick(View v) {  
 *// we need to the pass the view and the position in the array* listener.onClick(v, getAdapterPosition());  
 }  
 }  
  
}

## ContactModel

package com.example.fitnote13022021;  
  
public class ContactModel {  
  
 *//Initialize variables* String name, number;  
 *//Generate getter and setter* public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getNumber() {  
 return number;  
 }  
  
 public void setNumber(String number) {  
 this.number = number;  
 }  
  
}

## DataBaseHelper

package com.example.fitnote13022021;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.database.Cursor;  
import android.database.sqlite.SQLiteDatabase;  
import android.database.sqlite.SQLiteOpenHelper;  
import android.util.Log;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class DataBaseHelper extends SQLiteOpenHelper {  
  
 *//Initialize variables* private static final String *DATABASE\_NAME* = "fitnote\_database12345678";  
 private static final int *DATABASE\_VERSION* = 1;  
  
 private static final String *TABLE\_USER* = "Users";  
 private static final String *TABLE\_EXERCISES* = "Exercises";  
 private static final String *TABLE\_USEREXERCISES* = "UserExercises";  
 private static final String *TABLE\_SONGS* = "Songs";  
  
 DataBaseHelper(Context context){  
 super(context, *DATABASE\_NAME*, null, *DATABASE\_VERSION*);  
 }  
  
 public DataBaseHelper(@Nullable Context context, @Nullable String name, @Nullable SQLiteDatabase.CursorFactory factory, int version) {  
 super(context, name, null, version);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 *//Create tables* String tableUser = "CREATE TABLE " + *TABLE\_USER* + " (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER, userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)";  
 *//userLevel 0-normal, 1-admin, 2-superAdmin* String tableExercise = "CREATE TABLE " + *TABLE\_EXERCISES* + " (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT, exercisePic INTEGER, exerciseDetail TEXT)";  
 String tableUserExercises = "CREATE TABLE " + *TABLE\_USEREXERCISES* + " (userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)";  
 String tableSongs = "CREATE TABLE " + *TABLE\_SONGS* + " (songID INTEGER PRIMARY KEY, songName TEXT, songMP3 INTEGER)";  
  
  
 db.execSQL(tableUser);  
 db.execSQL(tableExercise);  
 db.execSQL(tableUserExercises);  
 db.execSQL(tableSongs);  
  
 *// adding exercises  
 // Exercise Table;  
 // (exerciseID INT PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INT, exerciseDetail TEXT)* db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (1, 'Push Up', "+R.drawable.*pushup*+", '1. Get down on all fours, placing your hands slightly wider than your shoulders. " +  
 "\n" +  
 "2. Straighten your arms and legs. " +  
 "\n" +  
 "3. Lower your body until your chest nearly touches the floor. " +  
 "\n" +  
 "4. Pause, then push yourself back up. " +  
 "\n" +  
 "5. Repeat. ')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (2, 'Squat', "+R.drawable.*squat*+", '1. Find a foot stance that feels best for you. Pointing your toes slightly outwards helps some, but keeping them parallel is fine, too. If you’re not sure what’s best, start by putting your feet shoulder-width apart and pointed about 15 degrees outwards. " +  
 "\n" +  
 "2. Tense your abs like someone is about to punch you. " +  
 "\n" +  
 "3. Look straight ahead and stand tall!')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (3, 'Running', "+R.drawable.*running*+", 'TIPS: When you first start out, try alternating between running and walking during your session. As time goes on, make the running intervals longer until you no longer feel the need to walk. " +  
 "Give yourself a few minutes to cool down after each run by walking and a doing few stretches. Try our post-run stretch routine.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (4, 'RopeJump', "+R.drawable.*ropejump*+", 'Get a large rope that you can pass under your feet. Move the rope under your feet while you jump and repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (5, 'JumpingJacks', "+R.drawable.*jumping\_jacks*+", 'JumpingJacks: Stand upright with your legs together, arms at your sides. "+  
 "\n" +  
 "Bend your knees slightly, and jump into the air. "+  
 "\n" +  
 "As you jump, spread your legs to be about shoulder-width apart. Stretch your arms out and over your head. " +  
 "\n" +  
 "Jump back to starting position. "+  
 "\n" +  
 "Repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (6, 'TricepsDips', "+R.drawable.*triceps\_dips*+", '1. You can do it from a chair or a bench, "+  
 "you can even do it on the floor. "+  
 "\n" +  
 "2. So just getting yourself into position to start by rolling your shoulders down. "+  
 "You want to start from good posture. Sometimes we tend to slump forward and then were gonna potentially cause an extra strain on the shoulder joint. "+  
 "\n" +  
 "3. So set yourself up by rolling the shoulders back, opening up the chest. Bringing your hands directly underneath your shoulders onto the bench or onto the ground. "+  
 "And youre gonna take your legs out keeping your knees bent." +  
 "\n" +  
 "4. If you wanted to make it harder you could extend the legs. "+  
 " So start with the easier option, see how you feel first. "+  
 "\n" +  
 "5. Youre gonna bend the elbows, lowering the hips down, and then exhale to extend the elbows and lift your body up. "+  
 "Inhale down, exhale up.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (7, 'InclinePushUp', "+R.drawable.*incline\_pushup*+", '1. Place your hands on the edge of the elevated surface. " +  
 "\n" +  
 "2. Step your feet back so your legs are straight and your arms are perpendicular to your body. "+  
 "\n" +  
 "3. Inhale as you slowly lower your chest to the edge of your platform. "+  
 "\n" +  
 "4. Pause for a second. "+  
 "\n" +  
 "5. Exhale as you push back to your starting position with your arms fully extended.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (8, 'JumpingSquat', "+R.drawable.*jumping\_squat*+", 'Stand tall with your feet hip-width apart. "+  
 "\n" +  
 "Hinge at the hips to push your butt back and lower down until your thighs are parallel to the floor. "+  
 "\n" +  
 "Allow your knees to bend 45 degrees when you land, and then immediately drop back down into a squat, and jump again.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (9, 'HammerCurls', "+R.drawable.*hammer\_curls*+", 'Step 1 "+  
 "Stand up straight with a dumbbell in each hand, holding them alongside you. Your palms should face your body. Keep your feet hip-width apart and engage your core to stabilize the body. "+  
 "\n" +  
 "Step 2 "+  
 "Keep your biceps stationary and start bending at your elbows, lifting both dumbbells. "+  
 "\n" +  
 "Step 3 " +  
 "Lift until the dumbbells reach shoulder-level, but don’t actually touch your shoulders. Hold this contraction briefly, then lower back to the starting position and repeat.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (10, 'ShoulderPress', "+R.drawable.*shoulder\_press*+", 'Stand with feet shoulder-width apart and hold the dumbbells at shoulder height with your elbows at a 90-degree angle. "+  
 "\n" +  
 "Slowly lift the dumbbells above your head without fully straightening your arms. Pause at the top. "+  
 "\n" +  
 "Slowly return to the start position.')");  
  
 db.execSQL("INSERT INTO " + *TABLE\_EXERCISES* + " VALUES (11, 'Swimming', "+R.drawable.*swimming*+", '1. Float with your face in the water, your body straight and horizontal. Stack your hands and keep your arms and legs long. " +  
 "Point your thumbs down. " +  
 "\n" +  
 "2. Press your hands out and back in a circle, elbows high. Lift your head slightly and inhale. " +  
 "\n" +  
 "3. Bring your hands together in front of your shoulders, thumbs pointing up. Keep your elbows close to your body. Simultaneously bend your knees, bringing your feet toward your butt and pointing your feet outward. " +  
 "\n" +  
 "4. Reach your arms forward. Kick out and back in a circle then snap your feet together. Drop your head underwater and exhale. " +  
 "\n" +  
 "5. Glide forward and repeat.')");  
  
 *// adding songs* db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (1, 'speedrun', "+R.raw.*speedrun*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (2, 'speedrun2', "+R.raw.*speedrun2*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (3, 'speedrun3', "+R.raw.*speedrun3*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (4, 'phantom', "+R.raw.*phantom*+")");  
 db.execSQL("INSERT INTO " + *TABLE\_SONGS* + " VALUES (5, 'shoping', "+R.raw.*shoping*+")");  
  
  
  
 *//Table Users:  
 //(userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)  
 //adding user alon - me the SUPER ADMIN* db.execSQL("INSERT INTO " + *TABLE\_USER* + " VALUES('alon', 123, 2, 90, 90, 'APR 2 2003', 'male', 'alon')");  
  
  
  
 *//adding userExercises  
 //JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(1, 'alon', 1, 'MAR 7 2021', 100, 1, 100)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(2, 'alon', 2, 'JAN 8 2021', 10, 2, 10)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(3, 'alon', 3, 'FEB 9 2021', 60, 3, 60)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(4, 'alon', 4, 'APR 10 2020', 30, 4, 30)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(5, 'alon', 5, 'JUN 11 2020', 50, 5, 50)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(6, 'alon', 6, 'SEP 25 2020', 100, 1, 100)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(7, 'alon', 7, 'MAR 7 2021', 20, 2, 20)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(8, 'alon', 8, 'JAN 8 2021', 56, 3, 56)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(9, 'alon', 9, 'FEB 9 2021', 34, 4, 34)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(10, 'alon', 10, 'APR 10 2020', 70, 5, 70)");  
 db.execSQL("INSERT INTO " + *TABLE\_USEREXERCISES* + " VALUES(11, 'alon', 11, 'MAR 11 2021', 120, 1, 110)");  
  
  
  
  
  
  
 }  
  
 *//Delete database* public static void deleteDatabase(Context mContext) {  
 mContext.deleteDatabase(*DATABASE\_NAME*);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 *//Drop Existing Table* db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_USER*);  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_EXERCISES*);  
 db.execSQL("DROP TABLE IF EXISTS " + *TABLE\_USEREXERCISES*);  
 onCreate(db);  
 }  
  
 *//method to add user* public boolean insertUser(User user) {  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", 0);  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USER*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
 }  
  
 *//method to add user* public boolean insertAdmin(Admin admin) {  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 int userLevel = 1;  
  
 if (admin.isSuperAdmin() == false){  
 userLevel = 1;  
 }  
 else {  
 userLevel = 2;  
 }  
  
  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", admin.getUserName());  
 cv.put("userPassword", admin.getUserPassword());  
 cv.put("userLevel", userLevel);  
 cv.put("userWeight", admin.getUserWeight());  
 cv.put("userHeight", admin.getUserHeight());  
 cv.put("userBirthDate", admin.getUserBirthDate());  
 cv.put("userGender", admin.getUserGender());  
 cv.put("profilePic", admin.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USER*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
 }  
  
 *//method to update User* public boolean updateUser(User user){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", user.getUserLevel());  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long result = db.update(*TABLE\_USER*, cv, "userName=?", new String []{user.getUserName()});  
  
 if(result == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to update User* public boolean updateUserAsAdmin(Admin user){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 int userLevel = 1;  
  
 if (user.isSuperAdmin())  
 userLevel = 2;  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", user.getUserName());  
 cv.put("userPassword", user.getUserPassword());  
 cv.put("userLevel", userLevel);  
 cv.put("userWeight", user.getUserWeight());  
 cv.put("userHeight", user.getUserHeight());  
 cv.put("userBirthDate", user.getUserBirthDate());  
 cv.put("userGender", user.getUserGender());  
 cv.put("profilePic", user.getProfilePic());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long result = db.update(*TABLE\_USER*, cv, "userName=?", new String []{user.getUserName()});  
  
 if(result == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to delete user (by ID because its the primary key)* public void deleteUser(User user) {  
 *// find user in the database. if its found, delete it and return true.  
 // if its not found, return false.  
  
 // getWritableDatabase - we are going to delete from it* SQLiteDatabase db = this.getWritableDatabase();  
  
 db.delete(*TABLE\_USER*, "userName=?", new String[]{user.getUserName()});  
  
 }  
  
 *//method to get user's level* public int getUserLevelByUserName(String givenUserName){  
 ArrayList<User> returnList = new ArrayList<>();  
  
 int userLevel = 0;  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)  
 // get data from the database* String queryString = ("SELECT " + "\*" + " FROM " + *TABLE\_USER* + " WHERE " + "userName" + " = '" + givenUserName + "'");  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 userLevel = cursor.getInt(2);  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return userLevel;  
 }  
  
 *//method to get all users* public ArrayList<User> getUsers(){  
  
 ArrayList<User> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USER*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = cursor.getString(0);  
 String userPassword = cursor.getString(1);  
 int userLevel = cursor.getInt(2);  
 int userWeight = cursor.getInt(3);  
 int userHeight = cursor.getInt(4);  
 String userBirthDate = cursor.getString(5);  
 String userGender = cursor.getString(6);  
 String profilePic = cursor.getString(7);  
  
 User newUser = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all users* public ArrayList<User> getUsersAndAdmins(){  
  
 ArrayList<User> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USER*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = cursor.getString(0);  
 String userPassword = cursor.getString(1);  
 int userLevel = cursor.getInt(2);  
 int userWeight = cursor.getInt(3);  
 int userHeight = cursor.getInt(4);  
 String userBirthDate = cursor.getString(5);  
 String userGender = cursor.getString(6);  
 String profilePic = cursor.getString(7);  
  
 *//user is Admin at minimum* if (userLevel == 1 || userLevel == 2){  
 *//settings boolean of isSuperAdmin* boolean isSuperAdmin = false;  
 if (userLevel == 2)  
 isSuperAdmin = true;  
  
 User newUser = new Admin(userName, userPassword, isSuperAdmin, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
 }  
 *//user is normal* else {  
 User newUser = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender, profilePic);  
 returnList.add(newUser);  
 }  
  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get user with specific name and password  
 //method to search for user (true - user exists | false - user doesn't exist)* public boolean searchUserByNameAndPass(String userName, String userPassword){  
 *//get all users from user table* List<User> Users = this.getUsers();  
  
 *// if there are no users* if(Users.isEmpty() || (userName.isEmpty() && userPassword.isEmpty()) )  
 return false;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName) && user.getUserPassword() .contentEquals(userPassword)){  
 return true;  
 }  
 }  
  
 return false;  
  
 }  
  
 *//method to search for user (true - user exists | false - user doesn't exist)* public boolean searchUserByName(String userName){  
 *//get all users from user table* List<User> Users = this.getUsersAndAdmins();  
  
 *// if there are no users* if(Users.isEmpty())  
 return false;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName)){  
 return true;  
 }  
 }  
  
 return false;  
  
 }  
  
 *//method to get for user (true - user exists | false - user doesn't exist)* public User getUserByName(String userName){  
 *//get all users from user table* ArrayList<User> Users = this.getUsersAndAdmins();  
  
 *// if there are no users* if(Users.isEmpty() || userName == null)  
 return null;  
  
 for (int i=0; i<Users.size(); i++) {  
 User user = Users.get(i);  
 if(user.getUserName().contentEquals(userName)){  
 return user;  
 }  
 }  
  
 return null;  
  
 }  
  
 *//method to insert Exercise* public boolean insertExercise(Exercise exercise){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* ContentValues cv = new ContentValues();  
 cv.put("exerciseID", exercise.getExerciseID());  
 cv.put("exerciseName", exercise.getExerciseName());  
 cv.put("exercisePic", exercise.getExercisePic());  
 cv.put("exerciseDetail", exercise.getExerciseDetail());  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_EXERCISES*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to get all exercises* public ArrayList<Exercise> getExercises(){  
  
 ArrayList<Exercise> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_EXERCISES*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
  
 Exercise newExercise = new Exercise(exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newExercise);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all exercises by userNameGiven* public ArrayList<UserExercisesInnerJoinEx> getExercisesByUserNameGiven(String userNameGiven){  
  
 ArrayList<UserExercisesInnerJoinEx> returnList = new ArrayList<>();  
  
 *// get data from the database  
 // First table: Exercises  
 // Second table: UserExercises  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT,  
 // userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER,  
 // rating INTEGER, repetition INTEGER)* String queryString = "SELECT Exercises.exerciseID, exerciseName, exercisePic, exerciseDetail, userExerciseID FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = '"+userNameGiven+"'";  
 *//String queryString = "SELECT Exercises.exerciseID, UserExercises.exerciseID, exerciseName, exercisePic, exerciseDetail FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = " + userNameGiven;  
 // get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *//UserExercisesInnerJoinEx (int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
 int userExerciseID = cursor.getInt(4);  
  
 UserExercisesInnerJoinEx newUserExercisesInnerJoinEx = new UserExercisesInnerJoinEx(userExerciseID, exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newUserExercisesInnerJoinEx);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to get all exercises by userNameGiven  
 //that have not been done by the user yet (by knowing that the date of doing the exercise is null)* public ArrayList<UserExercisesInnerJoinEx> getExercisesByUserNameGivenNotDone(String userNameGiven){  
  
 ArrayList<UserExercisesInnerJoinEx> returnList = new ArrayList<>();  
  
 *// get data from the database  
 // First table: Exercises  
 // Second table: UserExercises  
 // Exercise Table;  
 // (exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT,  
 // userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER,  
 // rating INTEGER, repetition INTEGER)* String queryString = "SELECT Exercises.exerciseID, exerciseName, exercisePic, exerciseDetail, userExerciseID FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = '"+userNameGiven+"' AND UserExercises.date IS NULL";  
 *//String queryString = "SELECT Exercises.exerciseID, UserExercises.exerciseID, exerciseName, exercisePic, exerciseDetail FROM Exercises INNER JOIN UserExercises ON Exercises.exerciseID = UserExercises.exerciseID WHERE UserExercises.userName = " + userNameGiven;  
 // get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *//UserExercisesInnerJoinEx (int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail)* int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
 int userExerciseID = cursor.getInt(4);  
  
 UserExercisesInnerJoinEx newUserExercisesInnerJoinEx = new UserExercisesInnerJoinEx(userExerciseID, exerciseID, exerciseName, exercisePic, exerciseDetail);  
 returnList.add(newUserExercisesInnerJoinEx);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to search for exercise by ID* public ArrayList<Exercise> getExercisesByID(int exerciseID){  
 *//get all users from user table* ArrayList<Exercise> exercises = this.getExercises();  
 ArrayList<Exercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(exercises.isEmpty())  
 return null;  
  
 for (int i=0; i<exercises.size(); i++) {  
 Exercise exercise = exercises.get(i);  
 if(exercise.getExerciseID() == exerciseID){  
 returnList.add(exercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to get one exercise by its ID* public Exercise getExerciseByID(int id){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = getReadableDatabase();  
  
 *// Table: Exercises  
 //(exerciseID INTEGER PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INTEGER, exerciseDetail TEXT)* Cursor cursor = db.rawQuery("SELECT \* FROM Exercises WHERE exerciseID = " + id, null);  
  
 cursor.moveToFirst();  
  
 int exerciseID = cursor.getInt(0);  
 String exerciseName = cursor.getString(1);  
 int exercisePic = cursor.getInt(2);  
 String exerciseDetail = cursor.getString(3);  
  
 Exercise exercise = new Exercise(exerciseID, exerciseName, exercisePic, exerciseDetail);  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return exercise;  
  
 }  
  
 *//method to insert UserExercise* public boolean insertUserExercise(UserExercise userExercise){  
  
 *//getWritableDatabase - method from the default properties of SQLiteOpenHelper  
 //getWritableDatabase for insert actions...getReadableDatabase for select (read) actions.* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// ContentValues - a special class that works like a associative array (PHP)  
 // can take pairs of values and associate with them (like the bundle in intent)  
 // The ID column is auto increment  
 // UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("userName", userExercise.getUserName());  
 cv.put("exerciseID", userExercise.getExerciseID());  
 cv.put("date", userExercise.getDate());  
 cv.put("time", userExercise.getTime());  
 cv.put("rating", userExercise.getRating());  
 cv.put("repetition", userExercise.getRepetition());  
  
  
 *// insert - success variable... 1)positive -> it was inserted...2) -1 -> it was a fail* long insert = db.insert(*TABLE\_USEREXERCISES*, null, cv);  
  
 if(insert == -1){  
 return false;  
 }else{  
 return true;  
 }  
  
 }  
  
 *//method to delete userExercise  
 //using it to delete userExercise when in dialog to delete an userExercise* public void deleteUserExercise(String userName,String exerciseName, int userExerciseID){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getWritableDatabase();  
  
 *//table: Exercises  
 //(exerciseID INTEGER PRIMARY KEY, exerciseName TEXT, exercisePic INTEGER, exerciseDetail TEXT)* Cursor c = db.rawQuery("SELECT \* FROM "+ *TABLE\_EXERCISES* +" WHERE exerciseName = '"+exerciseName+"'", null);  
  
 c.moveToFirst();  
  
 *//taking the exerciseID from the selected exercise  
 //there is supposed to be only one exercise with the name it has* int exerciseID = c.getInt(0);  
 *//Log.d("exerciseID", String.valueOf(exerciseID));  
 //table UserExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* String queryString = "DELETE FROM " + *TABLE\_USEREXERCISES* + " WHERE userName = '"+userName+"' AND exerciseID = " + exerciseID + " AND userExerciseID = " + userExerciseID;  
  
 Log.*d*("Delete filter", String.*valueOf*(userExerciseID));  
 *//Log.d("Delete filter2", String.valueOf(exerciseID));* db.execSQL(queryString);  
  
 }  
  
 *//method to update userExercise  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* public void updateUserExercise(int userExerciseID, String date, int time, int rating, int repetition){  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getWritableDatabase();  
  
 *// UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* ContentValues cv = new ContentValues();  
 cv.put("date", date);  
 cv.put("time", time);  
 cv.put("rating", rating);  
 cv.put("repetition", repetition);  
  
 db.update(*TABLE\_USEREXERCISES*, cv, "userExerciseID = " + userExerciseID,null);  
  
 }  
  
 *//method to get all UserExercises* public ArrayList<UserExercise> getUserExercises(){  
  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_USEREXERCISES*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if(cursor.moveToFirst()){  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do{  
 *// UserExercises Table  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time INTEGER, rating INTEGER, repetition INTEGER)* int userExerciseID = cursor.getInt(0);  
 String userName = cursor.getString(1);  
 int exerciseID = cursor.getInt(2);  
 String date = cursor.getString(3);  
 int time = cursor.getInt(4);  
 int rating = cursor.getInt(5);  
 int repetition = cursor.getInt(6);  
  
 UserExercise newUserExercise = new UserExercise(userExerciseID, userName, exerciseID, date, time, rating, repetition);  
 returnList.add(newUserExercise);  
  
 } while (cursor.moveToNext());  
  
 }else{  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID* public ArrayList<UserExercise> getUserExercisesByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName() == userName){  
 returnList.add(userExercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID  
 //that have been done by the user (by knowing that the date of doing the exercise is not null)* public ArrayList<UserExercise> getUserExercisesDoneByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName().contentEquals(userName) && userExercise.getDate() != null){  
 returnList.add(userExercise);  
 }  
 }  
  
 if(returnList.isEmpty() == false){  
 Log.*d*("Error","List is empty");  
 }  
  
 return returnList;  
  
 }  
  
 *//method to search for userExercise by User ID  
 //that have not been done by the user yet (by knowing that the date of doing the exercise is null)* public ArrayList<UserExercise> getUserExercisesNotDoneByUserName(String userName){  
 *//get all users from user table* ArrayList<UserExercise> userExercises = this.getUserExercises();  
 ArrayList<UserExercise> returnList = new ArrayList<>();  
  
 *// if there are no users* if(userExercises.isEmpty())  
 return null;  
  
 for (int i=0; i<userExercises.size(); i++) {  
  
 UserExercise userExercise = userExercises.get(i);  
  
 if(userExercise.getUserName() == userName && userExercise.getDate() == null){  
 returnList.add(userExercise);  
 }  
 }  
  
 return returnList;  
  
 }  
  
 *//method to get all songs* public ArrayList<Song> getSongs() {  
  
 ArrayList<Song> returnList = new ArrayList<>();  
  
 *// get data from the database* String queryString = "SELECT \* FROM " + *TABLE\_SONGS*;  
  
 *// get a reference to the active database  
 // getWritableDatabase - insert, update or delete records  
 // getReadableDatabase - SELECT items from the database* SQLiteDatabase db = this.getReadableDatabase();  
  
 *// Cursor is the result set from a SQL statement* Cursor cursor = db.rawQuery(queryString, null);  
  
 *// moveToFirst returns a true if there were items selected* if (cursor.moveToFirst()) {  
  
 *// loop through the cursor (result set) and create new customer objects. Put them into the return list* do {  
 *// Song Table  
 //(songID INTEGER PRIMARY KEY, songName TEXT, songMP3 INTEGER)* int songID = cursor.getInt(0);  
 String songName = cursor.getString(1);  
 int songMP3 = cursor.getInt(2);  
  
 Song newSong = new Song(songID, songName, songMP3);  
 returnList.add(newSong);  
  
 } while (cursor.moveToNext());  
  
 } else {  
 *// failure. do not add anything to the list.* }  
  
 *// always clean up after yourself  
 // lets close the connection to the database so others can use it  
 // close the cursor when done.* cursor.close();  
  
 return returnList;  
 }  
}

## EditUserActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.content.Intent;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.View;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.Calendar;  
  
public class EditUserActivity extends AppCompatActivity implements TextWatcher, View.OnClickListener {  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 User userChosen;  
  
 String activeUserName;  
  
 int activeUserLevel;  
  
 EditText etPasswordEditUser;  
  
 TextView txtTitleEditUser, txtSeekBarWeightEditUser, txtSeekBarHeightEditUser;  
  
 SeekBar seekBarWeightEditUser, seekBarHeightEditUser;  
  
 DatePickerDialog datePickerDialog;  
 Button btDatePickerButtonEditUser, btUpdateEditUser, btCancelEditUser, btDeleteEditUser;  
  
 RadioGroup radioGroupGenderEditUser, radioGroupUserLevelEditUser;  
 RadioButton radioButtonGender, radioButtonUserLevel;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_edit\_user*);  
  
 *//setting database* dataBaseHelper = new DataBaseHelper(this);  
  
 *//Assign btDatePickerButtonEditUser* btDatePickerButtonEditUser = findViewById(R.id.*btDatePickerButtonEditUser*);  
 *//initialize date picker* initDatePicker();  
 btDatePickerButtonEditUser.setText(getTodaysDate());  
  
 Intent intent = getIntent();  
  
 String userNameChosen = intent.getStringExtra("userNameChosen");  
 activeUserName = intent.getStringExtra("activeUserName");  
 activeUserLevel = intent.getIntExtra("activeUserLevel", 0);  
  
 *//getting chosen user from database* userChosen = dataBaseHelper.getUserByName(userNameChosen);  
  
 *//Assign variables* etPasswordEditUser = findViewById(R.id.*etPasswordEditUser*);  
  
 txtTitleEditUser = findViewById(R.id.*txtTitleEditUser*);  
 txtSeekBarWeightEditUser = findViewById(R.id.*txtSeekBarWeightEditUser*);  
 txtSeekBarHeightEditUser = findViewById(R.id.*txtSeekBarHeightEditUser*);  
  
 seekBarWeightEditUser = findViewById(R.id.*seekBarWeightEditUser*);  
 seekBarHeightEditUser = findViewById(R.id.*seekBarHeightEditUser*);  
  
 btUpdateEditUser = findViewById(R.id.*btUpdateEditUser*);  
 btCancelEditUser = findViewById(R.id.*btCancelEditUser*);  
 btDeleteEditUser = findViewById(R.id.*btDeleteEditUser*);  
  
 radioGroupGenderEditUser = findViewById(R.id.*radioGroupGenderEditUser*);  
 radioGroupUserLevelEditUser = findViewById(R.id.*radioGroupUserLevelEditUser*);  
  
 *//setting title to show userNameChosen:* txtTitleEditUser.setText("Edit " + userChosen.getUserName() + " :");  
  
 *//setting seekBar and txt to show progress from user chosen* seekBarWeightEditUser.setProgress(userChosen.getUserWeight());  
 txtSeekBarWeightEditUser.setText(""+userChosen.getUserWeight());  
 seekBarHeightEditUser.setProgress(userChosen.getUserHeight());  
 txtSeekBarHeightEditUser.setText(""+userChosen.getUserHeight());  
  
 seekBarWeightEditUser.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarWeightEditUser.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 seekBarHeightEditUser.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarHeightEditUser.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 *//setting the edit text to show password* etPasswordEditUser.setText(userChosen.getUserPassword());  
  
 *//setting textChange listener* etPasswordEditUser.addTextChangedListener(this);  
  
 *//setting click listener* btUpdateEditUser.setOnClickListener(this);  
 btCancelEditUser.setOnClickListener(this);  
 btDeleteEditUser.setOnClickListener(this);  
  
 }  
  
 @Override  
 public void onClick(View v) {  
 if(v == btUpdateEditUser)  
 {  
 *//gets the id of the checked radio button in radio group* int radioIdGender = radioGroupGenderEditUser.getCheckedRadioButtonId();  
  
 *//sets radioButton variable to whatever is checked* radioButtonGender = findViewById(radioIdGender);  
  
 *//sets String gender to be checked gender* String gender = radioButtonGender.getText().toString();  
  
 *//gets the id of the checked radio button in radio group* int radioIdUserLevel = radioGroupUserLevelEditUser.getCheckedRadioButtonId();  
  
 *//sets radioButton variable to whatever is checked* radioButtonUserLevel = findViewById(radioIdUserLevel);  
  
 *//sets String gender to be checked gender* String userLevelTXT = radioButtonUserLevel.getText().toString();  
  
 int chosenUserLevel = 0;  
  
 if (userLevelTXT.equals("Normal")){  
 chosenUserLevel = 0;  
 }else if (userLevelTXT.equals("Admin")){  
 chosenUserLevel = 1;  
 }else if (userLevelTXT.equals("Super Admin")){  
 chosenUserLevel = 2;  
 }  
  
 */// User (username TEXT PRIMARY KEY, pass TEXT, birthYear INT, gender TEXT)  
  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userWeight INTEGER,  
 // userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = userChosen.getUserName();  
 String userPassword = etPasswordEditUser.getText().toString();  
 Integer userWeight = Integer.*parseInt*(txtSeekBarWeightEditUser.getText().toString());  
 Integer userHeight= Integer.*parseInt*(txtSeekBarHeightEditUser.getText().toString());  
 String userBirthDate = btDatePickerButtonEditUser.getText().toString();  
 String userGender = gender;  
 String profilePic = userName;  
  
 User user = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender,profilePic);  
  
 *//putting user info in User table* if(chosenUserLevel == 0){  
 User user1 = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender,profilePic);  
 *//update chosen user info* dataBaseHelper.updateUser(user1);  
 }  
  
  
 *//if user is not noraml* if (chosenUserLevel == 1 || chosenUserLevel == 2){  
 Admin user2;  
  
 if (chosenUserLevel == 1)  
 user2 = new Admin(userName, userPassword, false,userWeight, userHeight, userBirthDate, userGender,profilePic);  
 else  
 user2 = new Admin(userName, userPassword, true,userWeight, userHeight, userBirthDate, userGender,profilePic);  
  
 *//update chosen user info* dataBaseHelper.updateUserAsAdmin(user2);  
  
 }  
  
  
  
 *//going to ManagerUsersActivity* Intent intent = new Intent(this, ManagerUsersActivity.class);  
 intent.putExtra("activeAdminName", activeUserName);  
 intent.putExtra("activeAdminLevel",activeUserLevel);  
 startActivity(intent);  
 finish();  
  
 }  
  
 if(v == btCancelEditUser)  
 {  
 *//going to ManagerUsersActivity* Intent intent = new Intent(this, ManagerUsersActivity.class);  
 intent.putExtra("activeAdminName", activeUserName);  
 intent.putExtra("activeAdminLevel",activeUserLevel);  
 startActivity(intent);  
 finish();  
 }  
  
 if (v == btDeleteEditUser){  
  
 dataBaseHelper.deleteUser(userChosen);  
  
 *//going to ManagerUsersActivity* Intent intent = new Intent(this, ManagerUsersActivity.class);  
 intent.putExtra("activeAdminName", activeUserName);  
 intent.putExtra("activeAdminLevel",activeUserLevel);  
 startActivity(intent);  
 finish();  
 }  
  
 }  
  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 *//gets editTexts inputs, trim() -> removing spaces between strings* String passwordInput = etPasswordEditUser.getText().toString().trim();  
  
 boolean allWritten = !passwordInput.isEmpty();  
  
 *// true - only if all buttons have text* btUpdateEditUser.setEnabled(allWritten);  
 }  
  
 @Override  
 public void afterTextChanged(Editable s) {  
  
 }  
  
 *//initialize date picker* private void initDatePicker() {  
 *//setting date listener* DatePickerDialog.OnDateSetListener dateSetListener = new DatePickerDialog.OnDateSetListener()  
 {  
 @Override  
 public void onDateSet(DatePicker datePicker, int year, int month, int day)  
 {  
 *//adding 1 to month because its default is 0* month = month + 1;  
 String date = makeDateString(day, month, year);  
 btDatePickerButtonEditUser.setText(date);  
  
 }  
 };  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
  
 int style = AlertDialog.*THEME\_HOLO\_LIGHT*;  
  
 *// (context to datePickerDialog, style, date listener, year, month, day)* datePickerDialog = new DatePickerDialog(this, style, dateSetListener, year, month, day);  
 *//sets max date to be today* datePickerDialog.getDatePicker().setMaxDate(System.*currentTimeMillis*());  
  
 }  
  
 *//creats date as a string* private String makeDateString(int day, int month, int year) {  
 return getMonthFormat(month) + " " + day + " " + year;  
 }  
  
 *//makes the number month a real month (1 - JAN)* private String getMonthFormat(int month) {  
 if(month == 1)  
 return "JAN";  
 if(month == 2)  
 return "FEB";  
 if(month == 3)  
 return "MAR";  
 if(month == 4)  
 return "APR";  
 if(month == 5)  
 return "MAY";  
 if(month == 6)  
 return "JUN";  
 if(month == 7)  
 return "JUL";  
 if(month == 8)  
 return "AUG";  
 if(month == 9)  
 return "SEP";  
 if(month == 10)  
 return "OCT";  
 if(month == 11)  
 return "NOV";  
 if(month == 12)  
 return "DEC";  
  
 *//default should never happen* return "JAN";  
 }  
  
 public void openDatePicker(View view)  
 {  
 datePickerDialog.show();  
 }  
  
 *//gets today's date* private String getTodaysDate() {  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 *//adding 1 to month because its defualt is 0* month = month + 1;  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
 return makeDateString(day, month, year);  
 }  
  
 *//a method to go back to SettingsActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 *//going to ManagerUsersActivity* Intent intent = new Intent(this, ManagerUsersActivity.class);  
 intent.putExtra("activeAdminName", activeUserName);  
 intent.putExtra("activeAdminLevel",activeUserLevel);  
 startActivity(intent);  
 finish();  
  
 }  
  
}

## ExecuteExerciseActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.os.CountDownTimer;  
import android.view.View;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.Calendar;  
  
public class ExecuteExerciseActivity extends AppCompatActivity implements View.OnClickListener {  
  
 DataBaseHelper dataBaseHelper;  
  
 ImageView btnFinish, btnPlay;  
  
 public static TextView *txtTime*;  
  
 ImageView imgExercise;  
  
 public static boolean *playTheTimer* = true;  
  
 int userExerciseID;  
  
 String activeUserName;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_execute\_exercise*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding XML parts* btnFinish = findViewById(R.id.*btnFinish*);  
 btnPlay = findViewById(R.id.*btnPlay*);  
 *txtTime* = findViewById(R.id.*txtTime*);  
 imgExercise = findViewById(R.id.*imgExercise*);  
  
 *//refresh boolean for new screen  
 playTheTimer* = true;  
  
 Intent intent = getIntent();  
  
 userExerciseID = intent.getIntExtra("ExecuteUserExerciseID", 0);  
 int exerciseID = intent.getIntExtra("ExecuteExerciseID", 0);  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 Exercise exercise = dataBaseHelper.getExerciseByID(exerciseID);  
  
 imgExercise.setImageResource(exercise.getExercisePic());  
  
 btnPlay.setOnClickListener(this);  
 btnFinish.setOnClickListener(this);  
  
 }  
  
 *//on button click* @Override  
 public void onClick(View v) {  
  
 if (v == btnPlay) {  
  
 *//Playing the clock count and starting music* if (*playTheTimer* == true) {  
  
 Intent startService = new Intent(this, MusicAndTimerService.class);  
 startService.putExtra("ACTION", "PLAY");  
 startService(startService);  
  
 *//setting the image to be pause button* btnPlay.setImageResource(R.drawable.*pause\_button*);  
  
 }  
  
 *//Pausing the clock count and the music* if (*playTheTimer* == false) {  
  
 Intent startService = new Intent(this, MusicAndTimerService.class);  
 startService.putExtra("ACTION", "PAUSE");  
 startService(startService);  
  
 *//setting the image to be play button* btnPlay.setImageResource(R.drawable.*play\_button*);  
  
 }  
  
 *//Switching modes (Play/Pause) to switch between them* if (*playTheTimer* == true) {  
 *playTheTimer* = false;  
 } else {  
 *playTheTimer* = true;  
 }  
  
 }  
  
 if (v == btnFinish){  
  
 *//Pausing the clock count* if(!*playTheTimer*){  
 btnPlay.setImageResource(R.drawable.*play\_button*);  
 }  
  
 int countFromService = MusicAndTimerService.*count*;  
  
 *//stop button to stop music service* Intent startService = new Intent(this, MusicAndTimerService.class);  
 stopService(startService);  
  
 *//going to SignIn activity* Intent intent = new Intent(this, FeedbackActivity.class);  
  
 String date = getTodaysDate();  
  
 *//adding extra in order to update the userExercise later* intent.putExtra("userExerciseID", userExerciseID);  
 intent.putExtra("countOfExercise", countFromService);  
 intent.putExtra("date", date);  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 *//finishing the activity to go to onCreate when coming back* finish();  
 }  
  
 }  
  
 *//a method to go back to ProgramUserActivity* @Override  
 public void onBackPressed() {  
 super.onBackPressed();  
  
 *//stop button to stop music service* Intent startService = new Intent(this, MusicAndTimerService.class);  
 stopService(startService);  
  
 *//setting an intent to go back to the ProgramUserActivity after finishing an exercise* Intent intent = new Intent(ExecuteExerciseActivity.this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
 @Override  
 protected void onDestroy() {  
 super.onDestroy();  
  
 *//stop button to stop music service* Intent startService = new Intent(this, MusicAndTimerService.class);  
 stopService(startService);  
  
 finish();  
 }  
  
 *//gets today's date* private String getTodaysDate() {  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 *//adding 1 to month because its defualt is 0* month = month + 1;  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
 return makeDateString(day, month, year);  
 }  
  
 *//creats date as a string* private String makeDateString(int day, int month, int year) {  
 return getMonthFormat(month) + " " + day + " " + year;  
 }  
  
 *//makes the number month a real month (1 - JAN)* private String getMonthFormat(int month) {  
 if(month == 1)  
 return "JAN";  
 if(month == 2)  
 return "FEB";  
 if(month == 3)  
 return "MAR";  
 if(month == 4)  
 return "APR";  
 if(month == 5)  
 return "MAY";  
 if(month == 6)  
 return "JUN";  
 if(month == 7)  
 return "JUL";  
 if(month == 8)  
 return "AUG";  
 if(month == 9)  
 return "SEP";  
 if(month == 10)  
 return "OCT";  
 if(month == 11)  
 return "NOV";  
 if(month == 12)  
 return "DEC";  
  
 *//default should never happen* return "JAN";  
 }  
  
  
}

## Exercise

package com.example.fitnote13022021;  
  
public class Exercise {  
  
 *// Exercise Table;  
 // (exerciseID INT PRIMARY KEY, exerciseName TEXT,  
 // exercisePic INT, exerciseDetail TEXT)* private int exerciseID;  
 private String exerciseName;  
 private int exercisePic;  
 private String exerciseDetail;  
  
 *//constructors* public Exercise(int exerciseID, String exerciseName, int exercisePic, String exerciseDetail) {  
 this.exerciseID = exerciseID;  
 this.exerciseName = exerciseName;  
 this.exercisePic = exercisePic;  
 this.exerciseDetail = exerciseDetail;  
 }  
  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return "Exercise{" +  
 "exerciseID=" + exerciseID +  
 ", exerciseName='" + exerciseName + '\'' +  
 ", exercisePic=" + exercisePic +  
 ", exerciseDetail='" + exerciseDetail + '\'' +  
 '}';  
 }  
  
  
 *//Getters and Setters* public int getExerciseID() {  
 return exerciseID;  
 }  
  
 public void setExerciseID(int exerciseID) {  
 this.exerciseID = exerciseID;  
 }  
  
 public String getExerciseName() {  
 return exerciseName;  
 }  
  
 public void setExerciseName(String exerciseName) {  
 this.exerciseName = exerciseName;  
 }  
  
 public int getExercisePic() {  
 return exercisePic;  
 }  
  
 public void setExercisePic(int exercisePic) {  
 this.exercisePic = exercisePic;  
 }  
  
 public String getExerciseDetail() {  
 return exerciseDetail;  
 }  
  
 public void setExerciseDetail(String exerciseDetail) {  
 this.exerciseDetail = exerciseDetail;  
 }  
  
}

## ExerciseAdaptor

package com.example.fitnote13022021;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
  
public class ExerciseAdaptor extends BaseAdapter {  
  
 private ArrayList<Exercise> arrayList;  
 private Context context;  
  
 public ExerciseAdaptor(ArrayList<Exercise> arrayList, Context context) {  
 this.arrayList = arrayList;  
 this.context = context;  
 }  
  
 @Override  
 public int getCount() {  
 return arrayList.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return arrayList.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
  
 Exercise exercise = arrayList.get(position);  
  
 convertView = LayoutInflater.*from*(context).inflate(R.layout.*exercise\_layout*, null);  
  
 TextView txtMainTitle = convertView.findViewById(R.id.*txtMainTitle*);  
 ImageView imageView = convertView.findViewById(R.id.*imageView*);  
  
 txtMainTitle.setText(exercise.getExerciseName());  
 imageView.setImageResource(exercise.getExercisePic());  
  
 return convertView;  
 }  
  
}

## FeedbackActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.NumberPicker;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.TextView;  
  
public class FeedbackActivity extends AppCompatActivity {  
  
 DataBaseHelper dataBaseHelper;  
  
 int userExerciseID;  
 int timeCount = 0;  
 int repetition = 0;  
 String date;  
  
 TextView txtTestInput, txtFirstSentence, txtSecondSentence;  
  
 NumberPicker numberPicker;  
  
 Button btnFinish;  
  
 RadioGroup radioGroup;  
 RadioButton radioButton;  
  
 String activeUserName;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_feed\_back*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding XML parts* txtTestInput = findViewById(R.id.*txtTestInput*);  
 txtFirstSentence = findViewById(R.id.*txtFirstSentence*);  
 txtSecondSentence = findViewById(R.id.*txtSecondSentence*);  
 numberPicker = findViewById(R.id.*numberPicker*);  
 btnFinish = findViewById(R.id.*btnFinish*);  
 radioGroup = findViewById(R.id.*radioGroup*);  
  
 *//setting txtInput* txtTestInput.setText("Repetition: " + repetition);  
  
 *//getting the userExerciseId and seconds count from ExecuteExerciseActivity* Intent intent = getIntent();  
  
 userExerciseID = intent.getIntExtra("userExerciseID", 0);  
 timeCount = intent.getIntExtra("countOfExercise", 0);  
 date = intent.getStringExtra("date");  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 numberPicker.setMinValue(0);  
 numberPicker.setMaxValue(150);  
  
 *//setting a listener to repetition choose* numberPicker.setOnValueChangedListener(new NumberPicker.OnValueChangeListener() {  
 @Override  
 public void onValueChange(NumberPicker picker, int oldVal, int newVal) {  
 repetition = newVal;  
 txtTestInput.setText("Repetition: " + repetition);  
 }  
 });  
  
 *//setting a listener to click on finishing reporting feedback* btnFinish.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 *//gets the id of the checked radio button in radio group* int radioId = radioGroup.getCheckedRadioButtonId();  
  
 *//sets radioButton variable to whatever is checked* radioButton = findViewById(radioId);  
  
 *//sets String gender to be checked gender* int rating = Integer.*parseInt*(radioButton.getText().toString());  
  
 *////method to update userExercise  
 //changeUserExercise(int userExerciseID, String date, int time, int rating, int repetition)* dataBaseHelper.updateUserExercise(userExerciseID, date, timeCount, rating, repetition);  
  
 *//setting an intent to go back to the ProgramUserActivity after finishing an exercise* Intent intent = new Intent(FeedbackActivity.this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
 });  
  
  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
  
}

## InformationActivity

package com.example.fitnote13022021;  
  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
  
import com.google.android.material.floatingactionbutton.FloatingActionButton;  
import com.google.android.material.snackbar.Snackbar;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.widget.Toolbar;  
  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
  
public class InformationActivity extends AppCompatActivity implements AdapterView.OnItemClickListener{  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 ListView listViewExercisesInfo;  
  
 ArrayList<Exercise> exercises;  
  
 ExerciseAdaptor exerciseAdaptor;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_information*);  
  
 *//setting database* dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding XML parts  
 //Assign variables* listViewExercisesInfo = findViewById(R.id.*listViewExercisesInfo*);  
  
 exercises = dataBaseHelper.getExercises();  
  
 exerciseAdaptor = new ExerciseAdaptor(exercises, this);  
  
 listViewExercisesInfo.setAdapter(exerciseAdaptor);  
  
 *//settings onClick Listener* listViewExercisesInfo.setOnItemClickListener(this);  
  
 }  
  
 *//method to show a specific exercise* @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 *//gets the clicked exercise* Exercise chosenExercise = exercises.get(position);  
  
 int exerciseID = chosenExercise.getExerciseID();  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage(chosenExercise.getExerciseName() + ":");  
  
 *//Setting the builder's layout inflater* View alertView = getLayoutInflater().inflate(R.layout.*exercise\_with\_description\_layout*, null);  
  
 *//getting views from layout* ImageView exerciseImage = alertView.findViewById(R.id.*imageViewOfExerciseInDescription*);  
 TextView txtExerciseNameInDescription = alertView.findViewById(R.id.*txtExerciseNameInDescription*);  
 TextView txtExerciseDescription = alertView.findViewById(R.id.*txtExerciseDescription*);  
  
 *//getting the whole chosen exercise information* Exercise exerciseChosen = null;  
  
 for (int i=0; i<exercises.size(); i++) {  
 Exercise exercise = exercises.get(i);  
 if(exercise.getExerciseID() == exerciseID){  
 exerciseChosen = exercise;  
 }  
 }  
  
 *//Setting the imageView to show exercise image* exerciseImage.setImageResource(exerciseChosen.getExercisePic());  
  
 *//Setting textViews to show exercise details* txtExerciseNameInDescription.setText(exerciseChosen.getExerciseName());  
 txtExerciseDescription.setText(exerciseChosen.getExerciseDetail());  
  
  
 *//Setting the view to be in builder of alert dialog* builder.setView(alertView);  
  
 builder.setPositiveButton("Got it!", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 dialog.cancel();  
  
 }  
 });  
 *//builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 // public void onClick(DialogInterface dialog, int id) {  
 // dialog.cancel();  
 // }  
 // });  
  
 //Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
  
 }  
  
 *//a method to go back to MainScreenActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, MainScreenActivity.class);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

## MainScreenActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.Toast;  
  
public class MainScreenActivity extends AppCompatActivity implements View.OnClickListener {  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 EditText etUsername, etPassword;  
  
 Button btRegister, btLogIn;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 *//setting database* dataBaseHelper = new DataBaseHelper(this);  
  
 *//toolbar = findViewById(R.id.toolbar);  
 //setSupportActionBar(toolbar);  
 //https://www.youtube.com/watch?v=DMkzIOLppf4  
 //3:50  
 //\*\* https://www.youtube.com/watch?v=IrJ8Hzuz2LU  
  
 //finding XML parts  
 //Assign variables* etUsername = findViewById(R.id.*etUserName*);  
 etPassword = findViewById(R.id.*etPassword*);  
 btLogIn = findViewById(R.id.*btLogIn*);  
 btRegister = findViewById(R.id.*btRegister*);  
  
 *//setting click listener* btLogIn.setOnClickListener(this);  
 btRegister.setOnClickListener(this);  
  
 *//adding exercises (one time)  
 //dataBaseHelper.deleteDatabase(this);  
 //dataBaseHelper.addExercises();* }  
  
 @Override  
 public void onClick(View v) {  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT,  
 // userGender TEXT)* String userName = etUsername.getText().toString();  
 String userPassword = etPassword.getText().toString();  
  
 if (v == btRegister) {  
  
 *//clear text on editTexts* etUsername.setText("");  
 etPassword.setText("");  
  
 *//going to SignIn activity* Intent intent = new Intent(this, RegisterActivity.class);  
 startActivity(intent);  
 finish();  
 }  
  
 if (v == btLogIn) {  
  
 if (dataBaseHelper.searchUserByNameAndPass(userName, userPassword)) {  
 Toast.*makeText*(this, "Welcome: " + userName, Toast.*LENGTH\_SHORT*).show();  
  
 *//clear text on editTexts* etUsername.setText("");  
 etPassword.setText("");  
  
 *//going to programs activity* Intent intent = new Intent(this, ProgramUserActivity.class);  
 intent.putExtra("activeUserName", userName);  
 startActivity(intent);  
 finish();  
 } else {  
 Toast.*makeText*(this, "No user with that info", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
  
 *//a method that sets an AlertDialog if the user wants to leave* @Override  
 public void onBackPressed() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to leave? ");  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 MainScreenActivity.super.onBackPressed();  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//on create menu method (inflating) the layout in menu)* @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
  
 getMenuInflater().inflate(R.menu.*main\_menu*, menu);  
  
 return true;  
 }  
  
 *//on item click of menu* @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
  
  
 if (id == R.id.*aboutItem*){  
 *//going to InformationActivity activity* Intent intent = new Intent(this, InformationActivity.class);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
 *//shows dialog if user wants to exit* if (id == R.id.*exitItem*){  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to leave? ");  
 builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 MainScreenActivity.super.onBackPressed();  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 return true;}}

## ManagerUsersActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.ListView;  
import android.widget.SearchView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class ManagerUsersActivity extends AppCompatActivity {  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 String activeUserName;  
  
 User activeUser;  
  
 TextView txtWelcomeManageUsers, txtExplainManageUser;  
  
 ListView listViewManagerUsers;  
  
 SearchView searchViewManageUsersList;  
  
 ArrayList<User> users;  
  
 ArrayList<User> filteredUsers;  
  
 UsersAdapter usersAdaptor;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_manager\_users*);  
  
 *//setting database* dataBaseHelper = new DataBaseHelper(this);  
  
 Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeAdminName");  
  
 *//getting user from database* activeUser = dataBaseHelper.getUserByName(activeUserName);  
  
 *//Assign variables* txtWelcomeManageUsers = findViewById(R.id.*txtWelcomeManageUsers*);  
 txtExplainManageUser = findViewById(R.id.*txtExplainManageUser*);  
  
 listViewManagerUsers = findViewById(R.id.*listViewManagerUsers*);  
  
 searchViewManageUsersList = findViewById(R.id.*searchViewManageUsersList*);  
  
 users = dataBaseHelper.getUsersAndAdmins();  
  
 filteredUsers = users;  
  
 usersAdaptor = new UsersAdapter(users, this, activeUserName, activeUser.getUserLevel());  
  
 listViewManagerUsers.setAdapter(usersAdaptor);  
  
 *//initialize the searchWidget in order to filter exercises* initSearchWidgets();  
  
 String userLevel = "Admin";  
  
 if (activeUser.getUserLevel() == 2){  
 userLevel = "Super-Admin";  
 }  
  
 Toast.*makeText*(this, activeUserName + " Level: " + activeUser.getUserLevel(), Toast.*LENGTH\_LONG*).show();  
  
 *//settings text to say hello userName* txtWelcomeManageUsers.setText("Welcome " + activeUserName + " a known " + userLevel);  
  
 *//settings click listener to list* listViewManagerUsers.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 String userNameChosen = users.get(position).getUserName();  
  
 *//avoid user changing his own info and prevent non SUPER ADMIN people to do it* if(!userNameChosen.equals(activeUserName) && activeUser.getUserLevel() == 2){  
 *//going to EditUserActivity* Intent intent = new Intent(ManagerUsersActivity.this, EditUserActivity.class);  
 intent.putExtra("activeUserName", activeUserName);  
 intent.putExtra("activeUserLevel",activeUser.getUserLevel());  
 intent.putExtra("userNameChosen", users.get(position).getUserName());  
 startActivity(intent);  
 }else {  
 if(userNameChosen.equals(activeUserName))  
 Toast.*makeText*(ManagerUsersActivity.this, "YOU CANT EDIT YOURSELF!!!!!!", Toast.*LENGTH\_SHORT*).show();  
else if (activeUser.getUserLevel() != 2)  
 Toast.*makeText*(ManagerUsersActivity.this, "Your'e level doesn't allow you to edit users", Toast.*LENGTH\_SHORT*).show();

}  
  
 }  
 });  
  
 }  
  
 *//initialize the searchWidget in order to filter exercises* public void initSearchWidgets(){  
  
 SearchView searchView = (SearchView)searchViewManageUsersList;  
  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 *//this method is called every time a user  
 //puts in any character into the search view  
 //literally any chang :D* @Override  
 public boolean onQueryTextChange(String newText) {  
  
 filteredUsers = new ArrayList<User>();  
  
 *//same as a regular for lop* for (User user: users)  
 {  
  
 int userLevel = user.getUserLevel();  
 String txtLevel = "Normal";  
 if (userLevel == 1){  
 txtLevel = "Admin";  
 }else if(userLevel == 2){  
 txtLevel = "Super-Admin";  
 }  
  
 *//if the user's name has one of the letters in the written text* if(user.getUserName().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's password has one of the letters in the written text* else if(user.getUserPassword().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's weight has one of the letters in the written text* else if(Integer.*toString*(user.getUserWeight()).contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's height has one of the letters in the written text* else if(Integer.*toString*(user.getUserHeight()).contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's birth date has one of the letters in the written text* else if(user.getUserBirthDate().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's gender has one of the letters in the written text* else if(user.getUserGender().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 *//if the user's level has one of the letters in the written text* else if(txtLevel.toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUsers.add(user);  
 }  
  
 }  
  
 UsersAdapter usersAdaptorFiltered = new UsersAdapter(filteredUsers, ManagerUsersActivity.this, activeUserName, activeUser.getUserLevel());  
  
 listViewManagerUsers.setAdapter(usersAdaptorFiltered);  
  
 return false;  
 };  
 });  
  
 }  
  
 *//a method to go back to SettingsActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, SettingsActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

## MusicAndTimerService

package com.example.fitnote13022021;  
  
import android.app.Service;  
import android.content.Intent;  
import android.media.MediaPlayer;  
import android.os.CountDownTimer;  
import android.os.IBinder;  
import android.widget.Toast;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class MusicAndTimerService extends Service {  
  
 private DataBaseHelper dataBaseHelper;  
  
 private MediaPlayer mediaPlayer;  
  
 private CountDownTimer countDownTimer;  
  
 public static int *count* = 0;  
  
 *//this method does binds the service  
 //with and activity  
 //קושרת את השירות עם הפעילות* @Nullable  
 @Override  
 public IBinder onBind(Intent intent) {  
 return null;  
 }  
  
 @Override  
 public void onCreate() {  
 super.onCreate();  
 dataBaseHelper = new DataBaseHelper(this);  
 ArrayList<Song> songs = dataBaseHelper.getSongs();  
  
 int arrayLength = songs.size();  
  
 Random random = new Random();  
 *//This gives a random integer number between 0 (inclusive) and (length of array-1)* int randomNumber = random.nextInt(arrayLength);  
 Song chosenSong = songs.get(randomNumber);  
  
 Toast.*makeText*(this, "chosen song: " + chosenSong.getSongName() + "N: " + randomNumber, Toast.*LENGTH\_SHORT*).show();  
  
 *//create the mediaPlayer* mediaPlayer = MediaPlayer.*create*(this, chosenSong.getSongMP3());  
  
 mediaPlayer.setLooping(true);  
  
 if(countDownTimer != null){  
 countDownTimer.cancel();  
 countDownTimer = null;  
 }  
  
 *//setting up the timer  
 //max: 2147483347millis = 35791.389116667minutes = 596.523151944449978 hours, sec to change: 1000* countDownTimer = new CountDownTimer(Integer.*MAX\_VALUE* - 300, 1000) {  
 @Override  
 public void onTick(long millisUntilFinished) {  
 *count*++;  
 *//setting text in play to be time count from timer in MusicAndTimerService* ExecuteExerciseActivity.*txtTime*.setText(getDurationString(*count*));  
 }  
  
 @Override  
 public void onFinish() {  
 *count* = 0;  
 ExecuteExerciseActivity.*txtTime*.setText(getDurationString(0));  
 }  
 };  
  
 }  
  
 *//when we start the service the onStartCommand method will be called* @Override  
 public int onStartCommand(Intent intent, int flags, int startId) {  
  
 String action = intent.getStringExtra("ACTION");  
 if(action.contentEquals("PLAY")){  
 Toast.*makeText*(this, "PLAY", Toast.*LENGTH\_SHORT*).show();  
 *//play music and timer* mediaPlayer.start();  
 countDownTimer.start();  
 }else if (action.contentEquals("PAUSE")){  
 Toast.*makeText*(this, "PAUSE", Toast.*LENGTH\_SHORT*).show();  
 *//pause music and timer* mediaPlayer.pause();  
 countDownTimer.cancel();  
 }  
  
 *// this means this service will be explicitly started and stopped  
 //פירוש הדבר ששירות זה יופעל ויפסיק במפורש* return *START\_STICKY*;  
 }  
  
  
 *//when the service is stopped onDestroy method will be called* @Override  
 public void onDestroy() {  
 super.onDestroy();  
  
 *//stop music and timer* mediaPlayer.stop();  
 countDownTimer.cancel();  
 *count* = 0;  
 ExecuteExerciseActivity.*txtTime*.setText(getDurationString(0));  
  
 }  
  
 private String getDurationString(int seconds) {  
  
 int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 seconds = seconds % 60;  
  
 return twoDigitString(hours) + " : " + twoDigitString(minutes) + " : " + twoDigitString(seconds);  
 }  
  
 private String twoDigitString(int number) {  
  
 if (number == 0) {  
 return "00";  
 }  
  
 if (number / 10 == 0) {  
 return "0" + number;  
 }  
  
 return String.*valueOf*(number);  
 }  
  
}

## PictureFileHelper

package com.example.fitnote13022021;  
  
import android.app.Activity;  
import android.content.Context;  
import android.content.SharedPreferences;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.util.ArrayList;  
  
public class PictureFileHelper {  
  
 public static void writeFileToInternalStorage(Context context, Bitmap bitmap, String filename)  
 {  
 SharedPreferences sp= context.getSharedPreferences("info",0);  
 int counter=sp.getInt("counter", 0);  
 try {  
 FileOutputStream os = ((Activity)context).openFileOutput(filename+counter, Context.*MODE\_PRIVATE*);  
 *//Here compress 50%, store the compressed data in os.* bitmap.compress(Bitmap.CompressFormat.*PNG*,100,os);  
 counter++;  
 SharedPreferences.Editor editor=sp.edit();  
 editor.putInt("counter",counter);  
 editor.commit();  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static Bitmap readFileFromInternalStorage(Context context,String filename)  
 {  
 Bitmap b=null;  
 try {  
 InputStream in = ((Activity)context).openFileInput(filename);  
 b= BitmapFactory.*decodeStream*(in);  
 in.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return b;  
 }  
  
 public static Bitmap getPic(Context context, String name)  
 {  
 File mydir = context.getFilesDir();  
 File lister = mydir.getAbsoluteFile();  
 Bitmap bitmap=null;  
  
 for (String list : lister.list())  
 {  
 if(list.toString().contains(name)) {  
 *//Toast.makeText(context, list, Toast.LENGTH\_LONG).show();* bitmap = *readFileFromInternalStorage*(context, list);  
  
 }  
 }  
  
 return bitmap;  
 }  
  
}

## ProgramUserActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlarmManager;  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.app.TimePickerDialog;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.os.Build;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.AdapterView;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.SearchView;  
import android.widget.TextView;  
import android.widget.TimePicker;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Locale;  
  
public class ProgramUserActivity extends AppCompatActivity implements View.OnClickListener {  
  
 *//Initialize variables* public static String *activeUserName* = null;  
  
 DataBaseHelper dataBaseHelper;  
  
 Button btnSetAlarm, btnCancelAlarm;  
  
 ImageView imgProgramUserPic;  
  
 TextView txtMainTitle, txtEmptyListMessage, tv\_timer;  
  
 ListView listViewUserExercises;  
  
 ArrayList<UserExercisesInnerJoinEx> activeUsersExercises;  
  
 UserExerciseNotDoneAdapter userExerciseAdaptor;  
  
 ImageView btnAddExercise;  
  
 SearchView searchViewUserExerciseList;  
  
 private int tvHour=0, tvMinute=0;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_program\_user*);  
  
 *//creating the notification channel (if needed)* createNotificationChannel();  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding XML parts / Assign variables* imgProgramUserPic = findViewById(R.id.*imgProgramUserPic*);  
 listViewUserExercises = findViewById(R.id.*listViewUserExercises*);  
 txtMainTitle = findViewById(R.id.*txtMainTitle*);  
 txtEmptyListMessage = findViewById(R.id.*txtEmptyListMessage*);  
 tv\_timer = findViewById(R.id.*tv\_timer*);  
 btnAddExercise = findViewById(R.id.*btnAddExercise*);  
 searchViewUserExerciseList = findViewById(R.id.*searchViewUserExerciseList*);  
 btnSetAlarm = findViewById(R.id.*btnSetAlarm*);  
 btnCancelAlarm = findViewById(R.id.*btnCancelAlarm*);  
  
  
 Intent intent = getIntent();  
  
 *//getting the active userName from getting the user who entered with log in with his name given from the intent* String userName = intent.getStringExtra("activeUserName");  
  
 *activeUserName* = userName;  
  
 *//changing textView to show userName's name as well* String welcomeText = txtMainTitle.getText().toString();  
 txtMainTitle.setText(welcomeText + " " + *activeUserName*);  
  
 *//changing the userPicture to show the user's picture* Bitmap userPic = null;  
  
 if(userName != null)  
 userPic = PictureFileHelper.*getPic*(this, userName);  
  
 if(userPic != null && imgProgramUserPic != null){  
 imgProgramUserPic.setImageBitmap(userPic);  
 }  
  
  
 activeUsersExercises = dataBaseHelper.getExercisesByUserNameGivenNotDone(*activeUserName*);  
  
 *//setting up a new adaptor* userExerciseAdaptor = new UserExerciseNotDoneAdapter(activeUsersExercises, this, *activeUserName*);  
  
 listViewUserExercises.setAdapter(userExerciseAdaptor);  
  
 *//setting the txtEmptyListMessage visible if the listView is empty* if(activeUsersExercises.isEmpty()){  
 txtEmptyListMessage.setVisibility(View.*VISIBLE*);  
 }  
  
 *//goes to ExecuteExerciseActivity* listViewUserExercises.setOnItemClickListener(new AdapterView.OnItemClickListener() {  
 @Override  
 public void onItemClick(AdapterView<?> parent, View view, int position, long id) {  
  
 *//Toast.makeText(ProgramUserActivity.this, "1", Toast.LENGTH\_SHORT).show();* }  
 });  
  
 *//initialize the searchWidget in order to filter exercises* initSearchWidgets();  
  
 btnAddExercise.setOnClickListener(this);  
  
 tv\_timer.setOnClickListener(this);  
  
 btnSetAlarm.setOnClickListener(this);  
  
 btnCancelAlarm.setOnClickListener(this);  
  
 }  
  
 @Override  
 public void onClick(View v) {  
  
 *//if the user wants to add an exercise* if(v == btnAddExercise){  
  
 *//going to SignIn activity* Intent intent = new Intent(this, AddExerciseActivity.class);  
  
 *//adding extra* intent.putExtra("activeUserName", *activeUserName*);  
  
 startActivity(intent);  
  
 *//finishing the activity to go to onCreate when coming back* finish();  
  
 }  
  
 *//if the user want to set himself a timer* if(v == tv\_timer){  
 *//Initialize time picked dialog* TimePickerDialog timePickerDialog = new TimePickerDialog(  
 ProgramUserActivity.this,  
 new TimePickerDialog.OnTimeSetListener() {  
 @Override  
 public void onTimeSet(TimePicker view, int hourOfDay, int minute) {  
 *//Initialize hour and minute* tvHour = hourOfDay;  
 tvMinute = minute;  
 *//Initialize calendar* Calendar calendar = Calendar.*getInstance*();  
 *//Set hour and minute* calendar.set(0,0,0,tvHour,tvMinute);  
 *//Set selected time on text view* tv\_timer.setText(android.text.format.DateFormat.*format*("h:mm a", calendar));  
 }  
 },12,0,true  
 );  
 *//Displayed previous selected time* timePickerDialog.updateTime(tvHour,tvMinute);  
 *//Show dialog* timePickerDialog.show();  
 }  
  
 if(v == btnSetAlarm || v == btnCancelAlarm){  
  
 *//Set notificationID & message* Intent notificationIntent = new Intent(  
 ProgramUserActivity.this, ReminderBroadcast.class);  
  
 notificationIntent.putExtra("message", "Its time to train " + *activeUserName*);  
 notificationIntent.putExtra("activeUserName", *activeUserName*);  
  
 *//PendingIntent* PendingIntent pendingIntent = PendingIntent.*getBroadcast*(  
 ProgramUserActivity.this, 0, notificationIntent, 0  
 );  
 *//FLAG\_CANCEL\_CURRENT-if pending already exists, the current one will be canceled.  
 //FLAG\_UPDATE\_CURRENT-indicates that the pendingIntent  
 //which we created now can be updated in the future  
  
  
 //AlarmManager  
 //Use with getSystemService to retrieve an AlarmManager  
 //for receiving intents at a time of your choosing* AlarmManager alarmManager = (AlarmManager) getSystemService(*ALARM\_SERVICE*);  
  
 switch (v.getId()){  
 case R.id.*btnSetAlarm*:  
 *// Set Alarm - we got hour and minutes from: tvHour,tvMinute  
  
 // Create time* Calendar startTime = Calendar.*getInstance*();  
 startTime.set(Calendar.*HOUR\_OF\_DAY*, tvHour);  
 startTime.set(Calendar.*MINUTE*, tvMinute);  
 startTime.set(Calendar.*SECOND*, 1);  
 long alarmStartTime = startTime.getTimeInMillis();  
  
 *//checking that the time given is not before now* if (alarmStartTime <= System.*currentTimeMillis*()){  
  
 Toast.*makeText*(this, "This time is before now! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
  
 }else{  
 *// Set Alarm  
 //AlarmManager.RTC\_WAKEUP - wakes up the device to fire the pending intent  
 //at the specified time* alarmManager.set(AlarmManager.*RTC\_WAKEUP*, alarmStartTime, pendingIntent);  
  
 Toast.*makeText*(this, "Set Done! : " + startTime.getTime(), Toast.*LENGTH\_LONG*).show();  
 }  
  
 break;  
  
 case R.id.*btnCancelAlarm*:  
 *// Cancel Alarm* alarmManager.cancel(pendingIntent);  
 Toast.*makeText*(this, "Canceled", Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
  
 }  
  
 }  
  
 private void createNotificationChannel(){  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*O*){  
 CharSequence name = "LembuitReminderChannel";  
 String description = "Channel for Lembuit Remider";  
 int importance = NotificationManager.*IMPORTANCE\_DEFAULT*;  
 NotificationChannel channel = new NotificationChannel("notifyLemubit", name, importance);  
 channel.setDescription(description);  
  
  
 NotificationManager notificationManager = getSystemService(NotificationManager.class);  
 notificationManager.createNotificationChannel(channel);  
 }  
 }  
  
 *//initialize the searchWidget in order to filter exercises* public void initSearchWidgets(){  
  
 SearchView searchView = (SearchView)searchViewUserExerciseList;  
  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 *//this method is called every time a user  
 //puts in any character into the search view  
 //literally any change :D* @Override  
 public boolean onQueryTextChange(String newText) {  
  
 ArrayList<UserExercisesInnerJoinEx> filteredExercises = new ArrayList<UserExercisesInnerJoinEx>();  
  
 *//same as a regular for lop* for (UserExercisesInnerJoinEx userExercisesInnerJoinEx: activeUsersExercises)  
 {  
 *//if an exercise as one of the letter in the written text* if(userExercisesInnerJoinEx.getExerciseName().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredExercises.add(userExercisesInnerJoinEx);  
 }  
 }  
  
 UserExerciseNotDoneAdapter userExerciseAdaptor = new UserExerciseNotDoneAdapter(filteredExercises, ProgramUserActivity.this);  
  
 listViewUserExercises.setAdapter(userExerciseAdaptor);  
  
 return false;  
 };  
 });  
  
 }  
  
 *//on create menu method (inflating) the layout in menu)* @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
  
 getMenuInflater().inflate(R.menu.*program\_user\_menu*, menu);  
  
 return true;  
 }  
  
 *//on item click of menu* @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 int id = item.getItemId();  
  
 *//if the user wants to see graphs of his performances* if (id == R.id.*graphsItem*){  
  
 *//going to StatisticsActivity activity* Intent intent = new Intent(this, StatisticsActivity.class);  
  
 *//adding extra* intent.putExtra("activeUserName", *activeUserName*);  
  
 startActivity(intent);  
  
 *//finishing the activity to go to onCreate when coming back* finish();  
 }  
  
 *//if the user wants to change his settings* if (id == R.id.*settingsItem*){  
 *//going to SettingsActivity activity* Intent intent = new Intent(this, SettingsActivity.class);  
  
 *//adding extra* intent.putExtra("activeUserName", *activeUserName*);  
  
 startActivity(intent);  
  
 *//finishing the activity to go to onCreate when coming back* finish();  
 }  
  
 *//if the user wants to share his data with others* if(id == R.id.*shareItem*){  
  
 *//going to ShareActivity* Intent intent = new Intent(this, ShareActivity.class);  
 intent.putExtra("activeUserName", *activeUserName*);  
 startActivity(intent);  
 finish();  
  
 }  
  
 *//if the user wants to see his results on a list* if (id == R.id.*exercisesResultsItem*){  
 *//going to ViewExercisesResultsActivity* Intent intent = new Intent(this, ViewExercisesResultsActivity.class);  
 intent.putExtra("activeUserName", *activeUserName*);  
 startActivity(intent);  
 finish();  
 }  
  
  
 return true;  
 }  
  
 *//a method to go back to SettingsActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 *//going to MainScreenActivity* Intent intent = new Intent(this, MainScreenActivity.class);  
 startActivity(intent);  
 finish();  
  
 }  
  
}

## RegisterActivity

package com.example.fitnote13022021;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
  
import android.Manifest;  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.content.ContextWrapper;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.drawable.BitmapDrawable;  
import android.graphics.drawable.Drawable;  
import android.media.MediaPlayer;  
import android.os.Bundle;  
import android.os.Environment;  
import android.provider.MediaStore;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.View;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.util.Calendar;  
  
public class RegisterActivity extends AppCompatActivity implements TextWatcher, View.OnClickListener {  
  
 DataBaseHelper dataBaseHelper;  
  
 ImageView imgProfilePic, imgCemraButton;  
  
 Bitmap bitmapToSave;  
  
 private static final int *REQUEST\_IMAGE\_CAPTURE* = 100;  
  
 EditText etUsername, etPassword;  
  
 TextView txtSeekBarWeight, txtSeekBarHeight;  
 SeekBar seekBarWeight, seekBarHeight;  
  
 DatePickerDialog datePickerDialog;  
 Button dateButton;  
  
 Button btFinishReg, btCancel;  
  
 RadioGroup radioGroup;  
 RadioButton radioButton;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_register*);  
  
 *//initialize DataBaseHelper* dataBaseHelper = new DataBaseHelper(this);  
  
 *//initialize date picker* initDatePicker();  
 dateButton = findViewById(R.id.*btDatePickerButton*);  
 dateButton.setText(getTodaysDate());  
  
 *//finding XML parts* imgProfilePic = findViewById(R.id.*imgProfilePic*);  
 imgCemraButton = findViewById(R.id.*imgCemraButton*);  
 etUsername = findViewById(R.id.*etUserName*);  
 etPassword = findViewById(R.id.*etPassword*);  
  
 txtSeekBarWeight = findViewById(R.id.*txtSeekBarWeight*);  
 txtSeekBarHeight = findViewById(R.id.*txtSeekBarHeight*);  
  
 seekBarWeight = findViewById(R.id.*seekBarWeight*);  
 seekBarHeight = findViewById(R.id.*seekBarHeight*);  
  
 radioGroup = findViewById(R.id.*radioGroup*);  
  
 btFinishReg = findViewById(R.id.*btFinishReg*);  
 btCancel = findViewById(R.id.*btCancel*);  
  
 *//setting a tag to the imgProfilePic to be the default image:* imgProfilePic.setTag(R.drawable.*default\_user*);  
  
 seekBarWeight.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarWeight.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 seekBarHeight.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarHeight.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 *//setting textChange listener* etUsername.addTextChangedListener(this);  
 etPassword.addTextChangedListener(this);  
  
 *//setting click listener* btFinishReg.setOnClickListener(this);  
 btCancel.setOnClickListener(this);  
 imgCemraButton.setOnClickListener(this);  
  
 }  
  
  
 @Override  
 public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence s, int start, int before, int count) {  
 *//gets editTexts inputs, trim() -> removing spaces between strings* String usernameInput = etUsername.getText().toString().trim();  
 String passwordInput = etPassword.getText().toString().trim();  
  
 boolean allWritten = !usernameInput.isEmpty() && !passwordInput.isEmpty();  
  
 *// true - only if all buttons have text* btFinishReg.setEnabled(allWritten);  
 }  
  
 @Override  
 public void afterTextChanged(Editable s) {  
  
 }  
  
 @Override  
 public void onClick(View v) {  
 if(v == btFinishReg)  
 {  
 *//gets the id of the checked radio button in radio group* int radioId = radioGroup.getCheckedRadioButtonId();  
  
 *//sets radioButton variable to whatever is checked* radioButton = findViewById(radioId);  
  
 *//sets String gender to be checked gender* String gender = radioButton.getText().toString();  
  
 */// User (username TEXT PRIMARY KEY, pass TEXT, birthYear INT, gender TEXT)  
  
 // User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userWeight INTEGER,  
 // userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* String userName = etUsername.getText().toString();  
 String userPassword = etPassword.getText().toString();  
 Integer userWeight = Integer.*parseInt*(txtSeekBarWeight.getText().toString());  
 Integer userHeight= Integer.*parseInt*(txtSeekBarHeight.getText().toString());  
 String userBirthDate = dateButton.getText().toString();  
 String userGender = gender;  
 String profilePic = userName;  
  
 *//putting user info in User table* User user = new User(userName, userPassword, userWeight, userHeight, userBirthDate, userGender,profilePic);  
  
 *//method to search for user with the same userName (true - user exists | false - user doesn't exist)* if(dataBaseHelper.searchUserByName(user.getUserName()) == false) {  
 dataBaseHelper.insertUser(user);  
  
 saveToInternalStorage(userName);  
  
  
 Toast.*makeText*(this, user + " is inserted", Toast.*LENGTH\_SHORT*).show();  
  
 *//going to MainScreenActivity* Intent intent = new Intent(this, MainScreenActivity.class);  
 startActivity(intent);  
 finish();  
  
 } else{  
 Toast.*makeText*(this, "user with the name " + user.getUserName() + " already exists", Toast.*LENGTH\_SHORT*).show();  
 }  
  
  
 }  
  
 if(v == btCancel)  
 {  
 Toast.*makeText*(this, "Canceling...", Toast.*LENGTH\_SHORT*).show();  
  
 *//going to MainScreenActivity* Intent intent = new Intent(this, MainScreenActivity.class);  
 startActivity(intent);  
 finish();  
 }  
  
 if(v == imgCemraButton)  
 {  
 takePicture(imgCemraButton);  
 }  
  
 }  
  
 *//method to start another application using and intent object* public void takePicture(View view){  
  
 *//the intents intention is to capture an image  
 //we have to specify the action for the intent* Intent imageTakeIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
  
 *//now we are able to check if there is any application  
 //capable of handling this intent  
 //otherwise your'e application will crush* if(imageTakeIntent.resolveActivity(getPackageManager()) != null)  
 {  
 *//we have to user startActivityForResult method  
 //second parameter is the requestCode* startActivityForResult(imageTakeIntent, *REQUEST\_IMAGE\_CAPTURE*);  
 }  
  
  
  
 }  
  
 *//in order to receive the result from the other application we  
 //have to override a method called onActivityResult* @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
  
 *//we have to check if the requestCode is the same  
 //AND if the requestCode is ok* super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *REQUEST\_IMAGE\_CAPTURE* && resultCode == *RESULT\_OK*) {  
 *//here we have the data available on this intent  
 //now we can retrieve the data* Bundle extras = data.getExtras();  
 Bitmap imageBitmap = (Bitmap) extras.get("data");  
  
 *//now we can display the image on the image view* imgProfilePic.setImageBitmap(imageBitmap);  
  
 *//getting the image to save* bitmapToSave = imageBitmap;  
 }  
  
  
 }  
 *//save image to phone* private void saveToInternalStorage(String userName){  
 *//bitmapToSave* if(bitmapToSave != null && userName != null){  
 PictureFileHelper.*writeFileToInternalStorage*(this, bitmapToSave, userName);  
 }else {  
 *//default pic* Bitmap bitmap\_default\_user = BitmapFactory.*decodeResource*(getResources(),R.drawable.*default\_user*);  
 PictureFileHelper.*writeFileToInternalStorage*(this, bitmap\_default\_user, userName);  
 }  
  
  
 }  
  
  
 *//gets today's date* private String getTodaysDate() {  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 *//adding 1 to month because its defualt is 0* month = month + 1;  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
 return makeDateString(day, month, year);  
 }  
  
 *//initialize date picker* private void initDatePicker() {  
 *//setting date listener* DatePickerDialog.OnDateSetListener dateSetListener = new DatePickerDialog.OnDateSetListener()  
 {  
 @Override  
 public void onDateSet(DatePicker datePicker, int year, int month, int day)  
 {  
 *//adding 1 to month because its default is 0* month = month + 1;  
 String date = makeDateString(day, month, year);  
 dateButton.setText(date);  
  
 }  
 };  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
  
 int style = AlertDialog.*THEME\_HOLO\_LIGHT*;  
  
 *// (context to datePickerDialog, style, date listener, year, month, day)* datePickerDialog = new DatePickerDialog(this, style, dateSetListener, year, month, day);  
 *//sets max date to be today* datePickerDialog.getDatePicker().setMaxDate(System.*currentTimeMillis*());  
  
 }  
  
 *//creats date as a string* private String makeDateString(int day, int month, int year) {  
 return getMonthFormat(month) + " " + day + " " + year;  
 }  
  
 *//makes the number month a real month (1 - JAN)* private String getMonthFormat(int month) {  
 if(month == 1)  
 return "JAN";  
 if(month == 2)  
 return "FEB";  
 if(month == 3)  
 return "MAR";  
 if(month == 4)  
 return "APR";  
 if(month == 5)  
 return "MAY";  
 if(month == 6)  
 return "JUN";  
 if(month == 7)  
 return "JUL";  
 if(month == 8)  
 return "AUG";  
 if(month == 9)  
 return "SEP";  
 if(month == 10)  
 return "OCT";  
 if(month == 11)  
 return "NOV";  
 if(month == 12)  
 return "DEC";  
  
 *//default should never happen* return "JAN";  
 }  
  
 public void openDatePicker(View view)  
 {  
 datePickerDialog.show();  
 }  
  
 @Override  
 public void onBackPressed() {  
 *//going to MainScreenActivity* Intent intent = new Intent(this, MainScreenActivity.class);  
 startActivity(intent);  
 finish();  
 }  
}

## ReminderBroadcast

package com.example.fitnote13022021;  
  
import android.app.PendingIntent;  
import android.content.BroadcastReceiver;  
import android.content.Context;  
import android.content.Intent;  
import android.util.Log;  
  
import androidx.core.app.NotificationCompat;  
import androidx.core.app.NotificationManagerCompat;  
  
import java.util.concurrent.atomic.AtomicInteger;  
  
public class ReminderBroadcast extends BroadcastReceiver {  
 @Override  
 public void onReceive(Context context, Intent intent) {  
  
 Log.*d*("RECEIVED", "Notification received!" + " UserName" + intent.getStringExtra("activeUserName"));  
  
 *// Get id & message from intent.* String message = intent.getStringExtra("message");  
 String activeUserName = intent.getStringExtra("activeUserName");  
  
 *// Call ProgramUserActivity when notification is tapped.* Intent programUserAcIntent = new Intent(context, ProgramUserActivity.class);  
  
 *//we need to add this userName in order to take the user to his  
 //ProgramUserActivity screen* programUserAcIntent.putExtra("activeUserName", activeUserName);  
  
 *//Intent to go to user's exercise list when clicking on notification* PendingIntent contentIntent = PendingIntent.*getActivity*(  
 context, 0, programUserAcIntent, 0  
 );  
  
 NotificationCompat.Builder builder = new NotificationCompat.Builder(context, "notifyLemubit")  
 .setSmallIcon(android.R.drawable.*ic\_dialog\_info*)  
 .setContentTitle("Exercise Time!")  
 .setContentText(message)  
 .setContentIntent(contentIntent)  
 .setAutoCancel(true)  
 .setPriority(NotificationCompat.*PRIORITY\_DEFAULT*);  
  
 NotificationManagerCompat notificationManager = NotificationManagerCompat.*from*(context);  
  
 *//According to the official documentation  
 //starting in Android 8.0(API level 26)  
 //all notifications must be assigned to a channel  
  
 //the notificationID is a unique int for each notification that must be defined* notificationManager.notify(NotificationID.*getID*(), builder.build());  
  
 }  
  
 public static class NotificationID {  
 private final static AtomicInteger *c* = new AtomicInteger(0);  
 public static int getID() {  
 return *c*.incrementAndGet();  
 }  
 }  
  
}

## SettingsActivity

package com.example.fitnote13022021;  
  
import androidx.annotation.Nullable;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.text.Layout;  
import android.view.View;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.Calendar;  
  
public class SettingsActivity extends AppCompatActivity implements View.OnClickListener{  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 String activeUserName;  
  
 int activeUserLevel;  
  
 User activeUser;  
  
 ImageView imgSettingUserPic;  
  
 TextView txtSettingsUserNameShow, txtSettingsUserGenderShow  
 , txtSettingsPassword, txtSettingsWeightHeight  
 , txtSettingsGenderEdit, txtSettingsPictureEdit  
 ,txtAdminOPTitleSettings;  
  
 View layoutSettingsPassword, layoutSettingsWeightAndHeight  
 ,layoutSettingBirthDate, layoutSettingsGender, layoutSettingPic  
 ,layoutSettingManagerUsers;  
  
 DatePickerDialog datePickerDialog;  
 Button dateButton;  
  
 private static final int *REQUEST\_IMAGE\_CAPTURE* = 100;  
 Bitmap bitmapToSave;  
 ImageView imgSettingsEditProfilePic;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_settings*);  
  
 *//setting database* dataBaseHelper = new DataBaseHelper(this);  
  
 Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 activeUser = dataBaseHelper.getUserByName(activeUserName);  
  
 *//Assign variables* imgSettingUserPic = findViewById(R.id.*imgSettingUserPic*);  
  
 txtSettingsUserNameShow = findViewById(R.id.*txtSettingsUserNameShow*);  
 txtSettingsUserGenderShow = findViewById(R.id.*txtSettingsUserGenderShow*);  
 txtSettingsPassword = findViewById(R.id.*txtSettingsPassword*);  
 txtSettingsWeightHeight = findViewById(R.id.*txtSettingsWeightHeight*);  
 txtSettingsGenderEdit = findViewById(R.id.*txtSettingsGenderEdit*);  
 txtSettingsPictureEdit = findViewById(R.id.*txtSettingsPictureEdit*);  
 txtAdminOPTitleSettings = findViewById(R.id.*txtAdminOPTitleSettings*);  
  
 layoutSettingsPassword = findViewById(R.id.*layoutSettingsPassword*);  
 layoutSettingsWeightAndHeight = findViewById(R.id.*layoutSettingsWeightAndHeight*);  
 layoutSettingBirthDate = findViewById(R.id.*layoutSettingBirthDate*);  
 layoutSettingsGender = findViewById(R.id.*layoutSettingsGender*);  
 layoutSettingPic = findViewById(R.id.*layoutSettingPic*);  
 layoutSettingManagerUsers = findViewById(R.id.*layoutSettingManagerUsers*);  
  
 *//Setting the userPic to be the activeUser's pic  
 //changing the userPicture to show the user's picture* Bitmap userPic = PictureFileHelper.*getPic*(this, activeUserName);  
 if(userPic != null && imgSettingUserPic != null){  
 imgSettingUserPic.setImageBitmap(userPic);  
 }  
  
 *//changing layoutSettingManagerUsers to show if user is admin at least (1 or 2 in userLevel)* activeUserLevel = dataBaseHelper.getUserLevelByUserName(activeUserName);  
  
 *//if user is admin at least, show layoutSettingManagerUsers:* if (activeUserLevel == 1 || activeUserLevel == 2){  
 layoutSettingManagerUsers.setVisibility(View.*VISIBLE*);  
  
 *//txt: option: manage users* String textFromAdminTitle = txtAdminOPTitleSettings.getText().toString();  
  
 if(activeUserLevel == 1){  
 txtAdminOPTitleSettings.setText("Admin " + textFromAdminTitle);  
 }else {  
 txtAdminOPTitleSettings.setText("Super-Admin " + textFromAdminTitle);  
 }  
  
 }  
  
 *//Setting text to be user's info from database* txtSettingsUserNameShow.setText(activeUser.getUserName());  
 txtSettingsUserGenderShow.setText(activeUser.getUserGender());  
 txtSettingsPassword.setText(activeUser.getUserPassword());  
 txtSettingsWeightHeight.setText("Weight: " + activeUser.getUserWeight() + " Height: " + activeUser.getUserHeight());  
 txtSettingsGenderEdit.setText(activeUser.getUserGender());  
  
 layoutSettingsPassword.setOnClickListener(this);  
 layoutSettingsWeightAndHeight.setOnClickListener(this);  
 layoutSettingBirthDate.setOnClickListener(this);  
 layoutSettingsGender.setOnClickListener(this);  
 layoutSettingPic.setOnClickListener(this);  
 layoutSettingManagerUsers.setOnClickListener(this);  
  
 }  
  
 @Override  
 public void onClick(View v) {  
  
 *//If user wants to change his password* if (v == layoutSettingsPassword){  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to change the password? ");  
  
 *//Setting the builder's editTexts* final EditText passwordInput = new EditText(SettingsActivity.this);  
 passwordInput.setText(activeUser.getUserPassword());  
  
 *//Setting the builder's editTexts in the builder* builder.setView(passwordInput);  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 *//Setting the updatedUser* User updatedUser = activeUser;  
  
 *//Setting the password to be the new one* updatedUser.setUserPassword(passwordInput.getText().toString());  
  
 dataBaseHelper.updateUser(updatedUser);  
  
 activeUser = updatedUser;  
  
 txtSettingsPassword.setText(activeUser.getUserPassword());  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//If user wants to change his weight and/or height* if (v == layoutSettingsWeightAndHeight){  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to change the weight and/or height? ");  
  
 *//Setting the builder's layout inflater* View alertView = getLayoutInflater().inflate(R.layout.*weight\_height\_settings\_alert*, null);  
  
 *//Setting the view to be in builder of alert dialog* builder.setView(alertView);  
  
 *//Getting the xml parts from alert view* SeekBar seekBarSettingsWeight, seekBarSettingsHeight;  
 TextView txtSeekBarSettingsWeight, txtSeekBarSettingsHeight;  
  
 seekBarSettingsWeight = alertView.findViewById(R.id.*seekBarSettingsWeight*);  
 seekBarSettingsHeight = alertView.findViewById(R.id.*seekBarSettingsHeight*);  
 txtSeekBarSettingsWeight = alertView.findViewById(R.id.*txtSeekBarSettingsWeight*);  
 txtSeekBarSettingsHeight = alertView.findViewById(R.id.*txtSeekBarSettingsHeight*);  
  
 *//Setting for both seekers change listeners* seekBarSettingsWeight.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarSettingsWeight.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
 seekBarSettingsHeight.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  
 @Override  
 public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {  
 txtSeekBarSettingsHeight.setText(""+progress);  
 }  
  
 @Override  
 public void onStartTrackingTouch(SeekBar seekBar) {  
  
 }  
  
 @Override  
 public void onStopTrackingTouch(SeekBar seekBar) {  
  
 }  
 });  
  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 *//Setting the updatedUser* User updatedUser = activeUser;  
  
 *//Setting the weight and/or height to be the new one* updatedUser.setUserWeight(Integer.*parseInt*(txtSeekBarSettingsWeight.getText().toString()));  
 updatedUser.setUserHeight(Integer.*parseInt*(txtSeekBarSettingsHeight.getText().toString()));  
  
 dataBaseHelper.updateUser(updatedUser);  
  
 activeUser = updatedUser;  
  
 txtSettingsWeightHeight.setText("Weight: " + activeUser.getUserWeight() + " Height: " + activeUser.getUserHeight());  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//If user wants to change his birth date* if (v == layoutSettingBirthDate){  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to change your birth date? ");  
  
 *//Setting the builder's datePickerDialog  
 //initialize date picker* dateButton = new Button(SettingsActivity.this);  
  
 dateButton.setWidth(250);  
 dateButton.setHeight(200);  
 dateButton.setText("JAN 01 2020");  
 dateButton.setTextColor(Color.*BLACK*);  
 dateButton.setTextSize(30);  
 *//dateButton.setScrollBarStyle(Integer.parseInt("?android:spinnerStyle"));* dateButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 datePickerDialog.show();  
 }  
 });  
  
 initDatePicker();  
 dateButton.setText(getTodaysDate());  
  
 *//Setting the builder's editTexts in the builder* builder.setView(dateButton);  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 *//Setting the updatedUser* User updatedUser = activeUser;  
  
 *//Setting the birth date to be the new one* updatedUser.setUserBirthDate(dateButton.getText().toString());  
  
 dataBaseHelper.updateUser(updatedUser);  
  
 activeUser = updatedUser;  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//If user wants to change his gender* if (v == layoutSettingsGender){  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to change your gender? ");  
  
 *//Setting the builder's layout inflater* View alertView = getLayoutInflater().inflate(R.layout.*gender\_settings\_alert*, null);  
  
 *//Setting the view to be in builder of alert dialog* builder.setView(alertView);  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 *// Setting the updatedUser* User updatedUser = activeUser;  
  
 *// Setting the gender to be the new one* RadioButton radioButton;  
  
 final RadioGroup radioGroupSettingsGender = (RadioGroup)alertView.findViewById(R.id.*radioGroupSettingsGender*);  
  
 *// get selected radioButton from radioGroup // radioGroupSettingsGender.getCheckedRadioButtonId()* int selectedId = radioGroupSettingsGender.getCheckedRadioButtonId();  
  
 *// find the radioButton by returned id* radioButton = (RadioButton) alertView.findViewById(selectedId);  
  
 *// radioButton text* String radiovalue = String.*valueOf*(radioButton.getText());  
  
 updatedUser.setUserGender(radiovalue);  
  
 dataBaseHelper.updateUser(updatedUser);  
  
 activeUser = updatedUser;  
  
 txtSettingsGenderEdit.setText(activeUser.getUserGender());  
  
 txtSettingsUserGenderShow.setText(activeUser.getUserGender());  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//If user wants to change his profile pic* if (v == layoutSettingPic) {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
  
 builder.setMessage("Do you want to change your profile pic? ");  
  
 *//Setting the builder's layout inflater* View alertView = getLayoutInflater().inflate(R.layout.*profilepic\_settings\_alert*, null);  
  
 *//Setting the view to be in builder of alert dialog* builder.setView(alertView);  
  
 *//Setting the user picture to be the new one* imgSettingsEditProfilePic = alertView.findViewById(R.id.*imgSettingsEditProfilePic*);  
 final ImageView imgSettingsEditCemraButton = alertView.findViewById(R.id.*imgSettingsEditCemraButton*);  
  
 imgSettingsEditCemraButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 takePicture(imgSettingsEditCemraButton);  
 }  
 });  
  
 builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
  
 *// Setting the updatedUser* User updatedUser = activeUser;  
  
 updatedUser.setProfilePic(activeUser.getUserName());  
  
 dataBaseHelper.updateUser(updatedUser);  
  
 activeUser = updatedUser;  
  
 *// Setting the imageView of user pic  
 // from the settings layout to show new pic* imgSettingUserPic.setImageBitmap(bitmapToSave);  
  
 *//save image to internal storage* saveToInternalStorage(activeUser.getUserName());  
  
 }  
 });  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
 }  
  
 *//if the user(admin/superAdmin) wants to go to manager users* if (v == layoutSettingManagerUsers){  
  
 *//going to ManagerUsersActivity* Intent intent = new Intent(this, ManagerUsersActivity.class);  
 intent.putExtra("activeAdminName", activeUser.getUserName());  
 intent.putExtra("activeAdminLevel",activeUserLevel);  
 startActivity(intent);  
 }  
  
 }  
  
 *//gets today's date* private String getTodaysDate() {  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 *//adding 1 to month because its defualt is 0* month = month + 1;  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
 return makeDateString(day, month, year);  
 }  
  
 *//initialize date picker* private void initDatePicker() {  
 *//setting date listener* DatePickerDialog.OnDateSetListener dateSetListener = new DatePickerDialog.OnDateSetListener()  
 {  
 @Override  
 public void onDateSet(DatePicker datePicker, int year, int month, int day)  
 {  
 *//adding 1 to month because its default is 0* month = month + 1;  
 String date = makeDateString(day, month, year);  
 dateButton.setText(date);  
  
 }  
 };  
  
 Calendar cal = Calendar.*getInstance*();  
 *//setting default date to be current date* int year = cal.get(Calendar.*YEAR*);  
 int month = cal.get(Calendar.*MONTH*);  
 int day = cal.get(Calendar.*DAY\_OF\_MONTH*);  
  
 int style = AlertDialog.*THEME\_HOLO\_LIGHT*;  
  
 *// (context to datePickerDialog, style, date listener, year, month, day)* datePickerDialog = new DatePickerDialog(this, style, dateSetListener, year, month, day);  
 *//sets max date to be today* datePickerDialog.getDatePicker().setMaxDate(System.*currentTimeMillis*());  
  
 }  
  
 *//creats date as a string* private String makeDateString(int day, int month, int year) {  
 return getMonthFormat(month) + " " + day + " " + year;  
 }  
  
 *//makes the number month a real month (1 - JAN)* private String getMonthFormat(int month) {  
 if(month == 1)  
 return "JAN";  
 if(month == 2)  
 return "FEB";  
 if(month == 3)  
 return "MAR";  
 if(month == 4)  
 return "APR";  
 if(month == 5)  
 return "MAY";  
 if(month == 6)  
 return "JUN";  
 if(month == 7)  
 return "JUL";  
 if(month == 8)  
 return "AUG";  
 if(month == 9)  
 return "SEP";  
 if(month == 10)  
 return "OCT";  
 if(month == 11)  
 return "NOV";  
 if(month == 12)  
 return "DEC";  
  
 *//default should never happen* return "JAN";  
 }  
  
 *//method to start another application using and intent object* public void takePicture(View view){  
  
 *//the intents intention is to capture an image  
 //we have to specify the action for the indent* Intent imageTakeIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
  
 *//now we are able to check if there is any application  
 //capable of handling this intent  
 //otherwise your'e application will crush* if(imageTakeIntent.resolveActivity(getPackageManager()) != null)  
 {  
 *//we have to user startActivityForResult method  
 //second parameter is the requestCode* startActivityForResult(imageTakeIntent, *REQUEST\_IMAGE\_CAPTURE*);  
 }  
  
  
  
 }  
  
 *//in order to receive the result from the other application we  
 //have to override a method called* @Override  
 protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  
  
 *//we have to check if the requestCode is the same  
 //AND if the requestCode is ok* super.onActivityResult(requestCode, resultCode, data);  
 if (requestCode == *REQUEST\_IMAGE\_CAPTURE* && resultCode == *RESULT\_OK*) {  
 *//here we have the data available on this intent  
 //now we can retrieve the data* Bundle extras = data.getExtras();  
 Bitmap imageBitmap = (Bitmap) extras.get("data");  
  
 *//now we can display the image on the image view  
 //imgSettingsEditProfilePic is the profile pic on the alert  
 //and on the imageView from the settings layout* imgSettingsEditProfilePic.setImageBitmap(imageBitmap);  
  
 *//getting the image to save* bitmapToSave = imageBitmap;  
 }  
  
  
 }  
 *//save image to phone* private void saveToInternalStorage(String userName){  
 *//bitmapToSave* if(bitmapToSave != null){  
 PictureFileHelper.*writeFileToInternalStorage*(this, bitmapToSave, userName);  
 }else {  
 *//default pic* Bitmap bitmap\_default\_user = BitmapFactory.*decodeResource*(getResources(),R.drawable.*default\_user*);  
 PictureFileHelper.*writeFileToInternalStorage*(this, bitmap\_default\_user, userName);  
 }  
  
  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

## ShareActivity

package com.example.fitnote13022021;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
  
import android.Manifest;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.database.Cursor;  
import android.net.Uri;  
import android.os.Bundle;  
import android.provider.ContactsContract;  
import android.view.View;  
import android.widget.SearchView;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
  
public class ShareActivity extends AppCompatActivity {  
  
 *//Initialize variable* String activeUserName;  
  
 TextView txtShareWithContactScreen;  
  
 RecyclerView recyclerViewContacts;  
  
 ArrayList<ContactModel> contactList = new ArrayList<ContactModel>();  
  
 ArrayList<ContactModel> filteredContacts;  
  
 ContactListAdapter adapter;  
  
 ContactListAdapter.RecycleViewClickListener listener;  
  
 SearchView searchViewContactList;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_share*);  
  
 Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 *//Assign variables* txtShareWithContactScreen = findViewById(R.id.*txtShareWithContactScreen*);  
 recyclerViewContacts = findViewById(R.id.*recyclerViewContacts*);  
 searchViewContactList = findViewById(R.id.*searchViewContactList*);  
  
 *//edit TextView* String text = txtShareWithContactScreen.getText().toString();  
 txtShareWithContactScreen.setText(text + " " + activeUserName);  
  
 *//Check permission* checkPermission();  
  
 *//initialize the searchWidget in order to filter exercises* initSearchWidgets();  
  
  
 }  
  
 private void checkPermission() {  
 *//Check condition* if (ContextCompat.*checkSelfPermission*(ShareActivity.this  
 , Manifest.permission.*READ\_CONTACTS*)  
 != PackageManager.*PERMISSION\_GRANTED*){  
 *//When permission is not granted  
 //Request permission* ActivityCompat.*requestPermissions*(ShareActivity.this  
 , new String[]{Manifest.permission.*READ\_CONTACTS*}, 100);  
 }else {  
 *//When permission is granted  
 //Create method* getContactList();  
 }  
 }  
  
 private void getContactList() {  
 *//Initialize uri  
 //uniform resource identifier* Uri uri = ContactsContract.Contacts.*CONTENT\_URI*;  
  
 *//Sort by ascending (ASC)* String sort = ContactsContract.CommonDataKinds.Phone.*DISPLAY\_NAME*+" ASC";  
  
 *//Initialize cursos* Cursor cursor = getContentResolver().query(  
 uri, null, null, null, sort  
 );  
  
 *//Check condition* if(cursor.getCount() > 0){  
 *//When count is greater than 0  
 //Use while loop* while (cursor.moveToNext()){  
 *//Cursor move to next  
 //Get contact id* String id = cursor.getString(cursor.getColumnIndex(  
 ContactsContract.Contacts.*\_ID* ));  
 *//Get contact name* String name = cursor.getString(cursor.getColumnIndex(  
 ContactsContract.Contacts.*DISPLAY\_NAME* ));  
 *//Initialize phone uri* Uri uriPhone = ContactsContract.CommonDataKinds.Phone.*CONTENT\_URI*;  
 *//Initialize selection* String selection = ContactsContract.CommonDataKinds.Phone.*CONTACT\_ID* +" =?";  
 *//Initialize phone cursor* Cursor phoneCursor = getContentResolver().query(  
 uriPhone, null, selection  
 , new String[]{id}, null  
 );  
 *//Check condition* if (phoneCursor.moveToNext()) {  
 *//When phone cursor move to next* String number = phoneCursor.getString(phoneCursor.getColumnIndex(  
 ContactsContract.CommonDataKinds.Phone.*NUMBER* ));  
 *//Initialize contact model* ContactModel model = new ContactModel();  
 *//Set name* model.setName(name);  
 *//Set number* if(number.charAt(0) == '0')  
 {  
 number = "+972 "+number.substring(1);  
 }  
 model.setNumber(number);  
 *//Add model in array list* contactList.add(model);  
 *//Close number cursor* phoneCursor.close();  
 }  
 }  
 *//Close cursor* cursor.close();  
 }  
  
 *//setting filtered list to be contactList* filteredContacts = contactList;  
  
 *//Set layout manager* recyclerViewContacts.setLayoutManager(new LinearLayoutManager(this));  
  
 *//setOnClickListener* setOnClickListener();  
  
 *//Initialize adapter* adapter = new ContactListAdapter(this, contactList, listener);  
 *//Set adapter* recyclerViewContacts.setAdapter(adapter);  
 }  
  
 private void setOnClickListener() {  
 listener = new ContactListAdapter.RecycleViewClickListener() {  
 @Override  
 public void onClick(View v, int position) {  
 *//now we are creating an intent to go to the WhatsAppSendActivity* Intent intent = new Intent(getApplicationContext(), WhatsAppSendActivity.class);  
  
 *//now we want to get the contact ID, name and number:* ContactModel contactChosen = filteredContacts.get(position);  
 String name = contactChosen.getName();  
 String number = contactChosen.getNumber();  
  
 intent.putExtra("contactName", name);  
 intent.putExtra("contactNumber", number);  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
 };  
 }  
  
 @Override  
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  
 super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
 *//Check condition* if (requestCode == 100 & grantResults.length > 0 &&grantResults[0]  
 == PackageManager.*PERMISSION\_GRANTED*){  
 *//When permission is granted  
 //Call method* getContactList();  
 }else {  
 *//When permission is denied  
 //Display toast* Toast.*makeText*(this, "Permission Denied.", Toast.*LENGTH\_SHORT*).show();  
 *//Call check permission method* checkPermission();  
 }  
 }  
  
 *//initialize the searchWidget in order to filter contacts* public void initSearchWidgets(){  
  
 SearchView searchView = (SearchView)searchViewContactList;  
  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 *//this method is called every time a user  
 //puts in any character into the search view  
 //literally any chang :D* @Override  
 public boolean onQueryTextChange(String newText) {  
  
 filteredContacts = new ArrayList<ContactModel>();  
  
 *//same as a regular for lop* for (ContactModel contactModel: contactList)  
 {  
 *//if a contact has one of the letter in the written text  
 //OR a contact has one of the numbers in the written number* if(contactModel.getName().toLowerCase().contains(newText.toLowerCase())  
 || contactModel.getNumber().contains(newText)){  
 filteredContacts.add(contactModel);  
 }  
 }  
  
 ContactListAdapter contactListAdapter = new ContactListAdapter(ShareActivity.this, filteredContacts, listener);  
  
 recyclerViewContacts.setAdapter(contactListAdapter);  
  
 return false;  
 };  
 });  
  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

## Song

package com.example.fitnote13022021;  
  
public class Song {  
  
 *//Song (songID INTEGER PRIMARY KEY, songName TEXT, songMP3 INTEGER)* private int songID;  
 private String songName;  
 private int songMP3;  
  
 *//constructors* public Song(int songID, String songName, int songMP3) {  
 this.songID = songID;  
 this.songName = songName;  
 this.songMP3 = songMP3;  
 }  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return "Song{" +  
 "songID=" + songID +  
 ", songName='" + songName + '\'' +  
 ", songMP3=" + songMP3 +  
 '}';  
 }  
  
 *//Getters and Setters* public int getSongID() {  
 return songID;  
 }  
  
 public void setSongID(int songID) {  
 this.songID = songID;  
 }  
  
 public String getSongName() {  
 return songName;  
 }  
  
 public void setSongName(String songName) {  
 this.songName = songName;  
 }  
  
 public int getSongMP3() {  
 return songMP3;  
 }  
  
 public void setSongMP3(int songMP3) {  
 this.songMP3 = songMP3;  
 }  
}

## StatisticsActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.content.ContextCompat;  
  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.ArrayAdapter;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.Spinner;  
import android.widget.Toast;  
  
import com.github.mikephil.charting.charts.BarChart;  
import com.github.mikephil.charting.components.AxisBase;  
import com.github.mikephil.charting.components.Description;  
import com.github.mikephil.charting.components.XAxis;  
import com.github.mikephil.charting.components.YAxis;  
import com.github.mikephil.charting.data.BarData;  
import com.github.mikephil.charting.data.BarDataSet;  
import com.github.mikephil.charting.data.BarEntry;  
import com.github.mikephil.charting.data.BubbleEntry;  
import com.github.mikephil.charting.data.CandleEntry;  
import com.github.mikephil.charting.data.Entry;  
import com.github.mikephil.charting.data.PieEntry;  
import com.github.mikephil.charting.data.RadarEntry;  
import com.github.mikephil.charting.formatter.DefaultAxisValueFormatter;  
import com.github.mikephil.charting.formatter.IAxisValueFormatter;  
import com.github.mikephil.charting.formatter.IndexAxisValueFormatter;  
import com.github.mikephil.charting.formatter.ValueFormatter;  
import com.github.mikephil.charting.utils.ColorTemplate;  
  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.Locale;  
import java.util.TimeZone;  
  
public class StatisticsActivity extends AppCompatActivity {  
  
 DataBaseHelper dataBaseHelper;  
  
 ImageView imgUserPicture;  
  
 *//Objects to choose exercise in graph* Spinner exercise\_spinner;  
  
 ArrayList<Exercise> exercises;  
  
 ArrayList<String> exerciseNames;  
  
 Spinner year\_spinner;  
  
 ArrayList<String> years;  
  
 *//Button to change filter of exercise in graph* Button btnSubmit;  
  
 *//Objects to create the BarChart(graph)* BarChart barChart;  
  
 BarData barData;  
  
 BarDataSet barDataSet;  
  
 ArrayList<BarEntry> barEntries;  
  
 ArrayList<UserExercise> userExercisesDone;  
  
 ArrayList<Integer> colors = new ArrayList<Integer>();  
  
 *//Values to save* String activeUserName;  
  
 *//Filter ID of exercise for graph* int exerciseIDToCheck = 1;  
 String yearToCheck = "2021";  
  
 *//Object for date format* SimpleDateFormat dateFormat = new SimpleDateFormat("YYYY/MM/dd", Locale.*ENGLISH*);  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_statistics*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding xml parts  
 //finding the BarChart (the graph)  
 //finding spinner* imgUserPicture = findViewById(R.id.*imgUserPicture*);  
 barChart = findViewById(R.id.*barChart*);  
 exercise\_spinner = findViewById(R.id.*exercise\_spinner*);  
 year\_spinner = findViewById(R.id.*year\_spinner*);  
 btnSubmit = findViewById(R.id.*btnSubmit*);  
  
  
 *//setting the exercise\_spinner list* exerciseNames = new ArrayList<String>();  
 *//getting the exercises* exercises = dataBaseHelper.getExercises();  
  
 *//populating the exercise spinner* populateExerciseSpinner();  
  
 *//setting the year\_spinner list* int yearNow = 2021;  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 DateTimeFormatter dtf = DateTimeFormatter.*ofPattern*("yyyy/MM/dd HH:mm:ss");  
 LocalDateTime now = LocalDateTime.*now*();  
 yearNow = now.getYear();  
 }  
  
 *//getting the years* years = *yearRange*("2020",String.*valueOf*(yearNow));  
  
 *//populating the year spinner* populateYearSpinner();  
  
 *//getting userName from intent* Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 *//changing the userPicture to show the user's picture* Bitmap userPic = PictureFileHelper.*getPic*(this, activeUserName);  
 if(userPic != null && imgUserPicture != null){  
 imgUserPicture.setImageBitmap(userPic);  
 }  
  
 *//getting userExercises by the active user name* userExercisesDone = dataBaseHelper.getUserExercisesDoneByUserName(activeUserName);  
  
 updateDataInBarChart();  
  
 *//setting onClickListener to btnSubmit* btnSubmit.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 *//gets the exercise name that was chosen* String exerciseNameChosen = exercise\_spinner.getSelectedItem().toString();  
  
 int size = exercises.size();  
  
 *//searching for exercise which has that name  
 //and setting the chosen exercise's ID to be  
 //the one who is filtering the barChart* for (int i=0; i<size; i++){  
 Exercise exercise = exercises.get(i);  
 if(exercise.getExerciseName().contentEquals(exerciseNameChosen)){  
 exerciseIDToCheck = exercise.getExerciseID();  
 }  
 }  
  
 *//gets the chosen year* String chosenYear = year\_spinner.getSelectedItem().toString();  
  
 *//setting the chosen year to be  
 //the one who is filtering the barChart* yearToCheck = chosenYear;  
  
 updateDataInBarChart();  
  
 }  
 });  
  
 }  
  
 private void updateDataInBarChart(){  
  
 *//calling a method to clear the barChart in order to update it properly* barChart.clear();  
  
 *//retrieve data for our barChart* getData();  
  
 *//now we will start our barChart* String labelDataSet = "Number above column is the repetition." + "\n";  
  
 labelDataSet += " Year: " + yearToCheck;  
  
 barDataSet = new BarDataSet(barEntries, labelDataSet);  
  
 barData = new BarData(barDataSet);  
  
 *//now we have to set our data in our bar chart* barChart.setData(barData);  
  
 barChart.getDescription().setText("Exercises data");  
  
 *//customizing barChart  
 //setting the colors to be the colors representing  
 //the user's ratings* barDataSet.setColors(colors);  
 barDataSet.setValueTextColor(Color.*BLACK*);  
 barDataSet.setValueTextSize(18f);  
  
 *//formatting XAxis to show date* XAxis xAxis = barChart.getXAxis();  
  
 *//JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC* String[] moths = new String[] {"JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"};  
  
 xAxis.setValueFormatter(new IndexAxisValueFormatter(moths));  
  
 xAxis.setCenterAxisLabels(true);  
 xAxis.setPosition(XAxis.XAxisPosition.*BOTTOM*);  
 xAxis.setGranularity(1);  
 xAxis.setGranularityEnabled(true);  
  
 *//setting YAxis* YAxis yAxisLeft = barChart.getAxisLeft();  
 YAxis yAxisRight = barChart.getAxisRight();  
  
 yAxisLeft.setAxisMaximum(150);  
 yAxisLeft.setAxisMinimum(0);  
  
 yAxisRight.setAxisMaximum(150);  
 yAxisRight.setAxisMinimum(0);  
  
  
 barChart.setDragEnabled(true);  
 barChart.setVisibleXRangeMaximum(3);  
  
 float barSpace = 0.08f;  
 float groupSpace = 0.44f;  
  
 barChart.getXAxis().setAxisMinimum(0);  
 barChart.getXAxis().setAxisMaximum(12);  
 *//barChart.getAxisLeft().setAxisMinimum(0);  
  
  
 //barChart.groupBars(0, groupSpace, barSpace);* barChart.invalidate();  
  
 }  
  
 *//adds content to exercise spinner* private void populateExerciseSpinner() {  
  
 int size = exercises.size();  
  
 for (int i=0; i<size; i++){  
 Exercise exercise = exercises.get(i);  
 exerciseNames.add(exercise.getExerciseName());  
 }  
  
 ArrayAdapter<String> exerciseAdapter = new ArrayAdapter<>(this, android.R.layout.*simple\_spinner\_item*, exerciseNames);  
  
 exerciseAdapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
  
 exercise\_spinner.setAdapter(exerciseAdapter);  
  
 }  
  
 *//adds content to year spinner* private void populateYearSpinner(){  
  
 ArrayAdapter<String> exerciseAdapter = new ArrayAdapter<>(this, android.R.layout.*simple\_spinner\_item*, years);  
  
 exerciseAdapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);  
  
 year\_spinner.setAdapter(exerciseAdapter);  
 }  
  
 *//a method to get data to barChard (to barEntries)* private void getData(){  
  
 barEntries = new ArrayList<BarEntry>();  
  
 if(userExercisesDone != null)  
 {  
 int length = userExercisesDone.size();  
  
 for (int i=0; i<length; i++) {  
  
 UserExercise userExercise = userExercisesDone.get(i);  
  
 *//checks the filter to what exercise should the graph show* if(userExercise.getExerciseID() == exerciseIDToCheck){  
  
 int rating = userExercise.getRating();  
 int repetition = userExercise.getRepetition();  
 String date = userExercise.getDate();  
 *//EXAMPLE DATE: "FEB 28 2021" 11 (year - 7-10, day - 5-6, month 0-2)* String[] splited = date.split("\\s+");  
  
 int year = Integer.*parseInt*(splited[2]);  
 int day = Integer.*parseInt*(splited[1]);  
 int month = getMonthNumber(date);  
  
 *//String dateForGraph = day + " " + month + " " + year;  
  
 //checks the filter to what year should the graph show* if(year == Integer.*parseInt*(yearToCheck)) {  
 long dateInMillis = *getTimeInMillis*(day, month, year);  
  
 Date dateOfExercise = new Date(year, month, day);  
  
 BarEntry entry = new BarEntry(month, repetition);  
  
 *//adding a color that represents the difficulty of  
 //doing the exercise  
 //the color is added to a list of colors later adapted  
 //to the barChart* colorRating(rating);  
  
 barEntries.add(entry);  
  
 }  
  
 }  
  
 }  
 }  
  
  
 }  
  
 private void colorRating(int rating){  
  
 switch(rating) {  
 case 1:  
 *//Hard* colors.add(getResources().getColor(R.color.*Hard*));  
 break;  
 case 2:  
 *//Hard-Medium* colors.add(getResources().getColor(R.color.*HardMedium*));  
 break;  
 case 3:  
 *//* colors.add(getResources().getColor(R.color.*Medium*));  
 break;  
 case 4:  
 *//Easy-Medium* colors.add(getResources().getColor(R.color.*EasyMedium*));  
 break;  
 case 5:  
 *//Easy* colors.add(getResources().getColor(R.color.*Easy*));  
 break;  
 default:  
 colors.add(getResources().getColor(R.color.*Easy*));  
 break;  
 }  
  
 }  
  
 *//This Method will convert given specific day,month and year into milliseconds.  
 // It will be very help when using Timpicker or Datepicker.* public static long getTimeInMillis(int day, int month, int year) {  
 Calendar calendar = Calendar.*getInstance*();  
 calendar.set(year, month, day);  
 return calendar.getTimeInMillis();  
 }  
  
 *//makes the real month a number month (JAN -> 1)* private int getMonthNumber(String date) {  
 if(date.contains("JAN"))  
 return 1;  
 if(date.contains("FEB"))  
 return 2;  
 if(date.contains("MAR"))  
 return 3;  
 if(date.contains("APR"))  
 return 4;  
 if(date.contains("MAY"))  
 return 5;  
 if(date.contains("JUN"))  
 return 6;  
 if(date.contains("JUL"))  
 return 7;  
 if(date.contains("AUG"))  
 return 8;  
 if(date.contains("SEP"))  
 return 9;  
 if(date.contains("OCT"))  
 return 10;  
 if(date.contains("NOV"))  
 return 11;  
 if(date.contains("DEC"))  
 return 12;  
  
 *//default should never happen* return 1;  
 }  
  
 public static ArrayList<String> yearRange(String startYear, String endYear) {  
 int cur = Integer.*parseInt*(startYear);  
 int stop = Integer.*parseInt*(endYear);  
 ArrayList<String> list = new ArrayList<String>();  
 while (cur <= stop) {  
 list.add(String.*valueOf*(cur++));  
 }  
 return list;  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

## User

package com.example.fitnote13022021;  
  
public class User {  
  
 *// User Table:  
 // (userName TEXT PRIMARY KEY, userPassword TEXT, userLevel INTEGER,  
 // userWeight INTEGER, userHeight INTEGER, userBirthDate TEXT, userGender TEXT, profilePic INTEGER)* private String userName;  
 private String userPassword;  
 private int userWeight;  
 private int userHeight;  
 private String userBirthDate;  
 private String userGender;  
 private String profilePic;  
  
 *//constructors  
 //constructor for all info* public User(String userName, String userPassword, int userWeight, int userHeight, String userBirthDate, String userGender, String profilePic) {  
 this.userName = userName;  
 this.userPassword = userPassword;  
 this.userWeight = userWeight;  
 this.userHeight = userHeight;  
 this.userBirthDate = userBirthDate;  
 this.userGender = userGender;  
 this.profilePic = profilePic;  
 }  
  
 *//constructor for user* public User(User user) {  
  
 this.userName = user.getUserName();  
 this.userPassword =user.getUserPassword();  
 this.userWeight =user.getUserWeight();  
 this.userHeight =user.getUserHeight();  
 this.userBirthDate =user.getUserBirthDate();  
 this.userGender =user.getUserGender();  
 this.profilePic =user.getProfilePic();  
  
 }  
  
 *//constructors  
 //constructor for no info* public User() {  
 }  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return "User{" +  
 "userName='" + userName + '\'' +  
 ", userPassword='" + userPassword + '\'' +  
 ", userWeight=" + userWeight +  
 ", userHeight=" + userHeight +  
 ", userBirthDate='" + userBirthDate + '\'' +  
 ", userGender='" + userGender + '\'' +  
 ", profilePic=" + profilePic +  
 '}';  
 }  
  
 *//Getters and Setters* public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public String getUserPassword() {  
 return userPassword;  
 }  
  
 public void setUserPassword(String userPassword) {  
 this.userPassword = userPassword;  
 }  
  
 public int getUserWeight() {  
 return userWeight;  
 }  
  
 public void setUserWeight(int userWeight) {  
 this.userWeight = userWeight;  
 }  
  
 public int getUserHeight() {  
 return userHeight;  
 }  
  
 public void setUserHeight(int userHeight) {  
 this.userHeight = userHeight;  
 }  
  
 public String getUserBirthDate() {  
 return userBirthDate;  
 }  
  
 public void setUserBirthDate(String userBirthDate) {  
 this.userBirthDate = userBirthDate;  
 }  
  
 public String getUserGender() {  
 return userGender;  
 }  
  
 public void setUserGender(String userGender) {  
 this.userGender = userGender;  
 }  
  
 public String getProfilePic() {  
 return profilePic;  
 }  
  
 public void setProfilePic(String profilePic) {  
 this.profilePic = profilePic;  
 }  
  
 public int getUserLevel(){  
 return 0;  
 }  
  
}

## UserExercise

package com.example.fitnote13022021;  
  
public class UserExercise{  
  
 *//table userExercises  
 //(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT,  
 // userName TEXT, exerciseID INTEGER, date TEXT, time INTEGER,  
 // rating INTEGER, repetition INTEGER)* private int userExerciseID;  
 private String userName;  
 private int exerciseID;  
 private String date;  
 private int time;  
 private int rating;  
 private int repetition;  
  
 *//constructors  
 //constructor for new UserExercises* public UserExercise(String userName, int exerciseID) {  
 this.userName = userName;  
 this.exerciseID = exerciseID;  
  
 this.date = null;  
 this.time = 0;  
 this.rating = 0;  
 this.repetition = 0;  
 }  
  
 *//constructor to take exsisting newUserExercises* public UserExercise(int userExerciseID, String userName, int exerciseID, String date, int time, int rating, int repetition) {  
 this.userExerciseID = userExerciseID;  
 this.userName = userName;  
 this.exerciseID = exerciseID;  
 this.date = date;  
 this.time = time;  
 this.rating = rating;  
 this.repetition = repetition;  
 }  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return "UserExercise{" +  
 "userExerciseID=" + userExerciseID +  
 ", userName='" + userName + '\'' +  
 ", exerciseID=" + exerciseID +  
 ", date='" + date + '\'' +  
 ", time=" + time +  
 ", rating=" + rating +  
 ", repetition=" + repetition +  
 '}';  
 }  
  
 *//Getters and Setters* public int getUserExerciseID() {  
 return userExerciseID;  
 }  
  
 public void setUserExerciseID(int userExerciseID) {  
 this.userExerciseID = userExerciseID;  
 }  
  
 public String getUserName() {  
 return userName;  
 }  
  
 public void setUserName(String userName) {  
 this.userName = userName;  
 }  
  
 public int getExerciseID() {  
 return exerciseID;  
 }  
  
 public void setExerciseID(int exerciseID) {  
 this.exerciseID = exerciseID;  
 }  
  
 public String getDate() {  
 return date;  
 }  
  
 public void setDate(String date) {  
 this.date = date;  
 }  
  
 public int getTime() {  
 return time;  
 }  
  
 public void setTime(int time) {  
 this.time = time;  
 }  
  
 public int getRating() {  
 return rating;  
 }  
  
 public void setRating(int rating) {  
 this.rating = rating;  
 }  
  
 public int getRepetition() {  
 return repetition;  
 }  
  
 public void setRepetition(int repetition) {  
 this.repetition = repetition;  
 }  
  
}

## UserExerciseDoneAdapter

package com.example.fitnote13022021;  
  
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import java.util.ArrayList;  
  
public class UserExerciseDoneAdapter extends BaseAdapter {  
  
 private ArrayList<UserExercise> arrayList;  
 public static Context *context*;  
 private DataBaseHelper dataBaseHelper;  
  
 public UserExerciseDoneAdapter(ArrayList<UserExercise> arrayList, Context context){  
 this.arrayList = arrayList;  
 this.*context* = context;  
 dataBaseHelper = new DataBaseHelper(context);  
 }  
  
 @Override  
 public int getCount() {  
 return arrayList.size();  
 }  
  
 @Override  
 public Object getItem(int position) { return arrayList.get(position); }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
  
 UserExercise userExercise = arrayList.get(position);  
 convertView = LayoutInflater.*from*(*context*).inflate(R.layout.*user\_exercise\_done\_layout*, null);  
  
 ImageView imageViewOfUserExerciseDone = convertView.findViewById(R.id.*imageViewOfUserExerciseDone*);  
 TextView txtExerciseOfUserExerciseDone = convertView.findViewById(R.id.*txtExerciseOfUserExerciseDone*);  
 TextView txtDateOfUserExerciseDone = convertView.findViewById(R.id.*txtDateOfUserExerciseDone*);  
 TextView txtTimeOfUserExerciseDone = convertView.findViewById(R.id.*txtTimeOfUserExerciseDone*);  
 TextView txtRatingOfUserExerciseDone = convertView.findViewById(R.id.*txtRatingOfUserExerciseDone*);  
 TextView txtRepetitionOfUserExerciseDone = convertView.findViewById(R.id.*txtRepetitionOfUserExerciseDone*);  
  
 *//setting textviews to show userExercise info* String textViewText = txtDateOfUserExerciseDone.getText().toString();  
  
 txtDateOfUserExerciseDone.setText(textViewText + userExercise.getDate());  
  
 textViewText = txtTimeOfUserExerciseDone.getText().toString();  
  
 txtTimeOfUserExerciseDone.setText(textViewText + userExercise.getTime());  
  
 textViewText = txtRatingOfUserExerciseDone.getText().toString();  
  
 txtRatingOfUserExerciseDone.setText(textViewText + userExercise.getRating());  
  
 textViewText = txtRepetitionOfUserExerciseDone.getText().toString();  
  
 txtRepetitionOfUserExerciseDone.setText(textViewText + userExercise.getRepetition());  
  
 *//getting the exercise by exerciseID* Exercise exerciseFromUserExercise = dataBaseHelper.getExerciseByID(userExercise.getExerciseID());  
  
 *//setting the image to show exercise image* imageViewOfUserExerciseDone.setImageResource(exerciseFromUserExercise.getExercisePic());  
  
 *//setting the lasy textView to show the exercise's name* textViewText = txtExerciseOfUserExerciseDone.getText().toString();  
  
 txtExerciseOfUserExerciseDone.setText(textViewText + exerciseFromUserExercise.getExerciseName());  
  
  
 return convertView;  
 }  
}

## UserExerciseNotDonteAdapter

package com.example.fitnote13022021;  
  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.Button;  
import android.widget.ImageView;  
import android.widget.ListAdapter;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
  
public class UserExerciseNotDoneAdapter extends BaseAdapter implements ListAdapter {  
  
 private ArrayList<UserExercisesInnerJoinEx> arrayList;  
 public static Context *context*;  
 private DataBaseHelper dataBaseHelper;  
 private String activeUserName;  
  
 public UserExerciseNotDoneAdapter(ArrayList<UserExercisesInnerJoinEx> arrayList, Context context, String userName) {  
 this.arrayList = arrayList;  
 this.*context* = context;  
 dataBaseHelper = new DataBaseHelper(context);  
 this.activeUserName=userName;  
 }  
 public UserExerciseNotDoneAdapter(ArrayList<UserExercisesInnerJoinEx> arrayList, Context context) {  
 this.arrayList = arrayList;  
 this.*context* = context;  
 dataBaseHelper = new DataBaseHelper(context);  
 }  
  
 @Override  
 public int getCount() {  
 return arrayList.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return arrayList.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
  
 UserExercisesInnerJoinEx userExercisesInnerJoinEx = arrayList.get(position);  
 convertView = LayoutInflater.*from*(*context*).inflate(R.layout.*user\_exercise\_layout*, null);  
  
 TextView txtMainTitle = convertView.findViewById(R.id.*txtMainTitle*);  
 ImageView imageView = convertView.findViewById(R.id.*imageView*);  
 Button btnDelete = convertView.findViewById(R.id.*btnDelete*);  
  
 txtMainTitle.setText(userExercisesInnerJoinEx.getExerciseName());  
 imageView.setImageResource(userExercisesInnerJoinEx.getExercisePic());  
  
 btnDelete.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, txtMainTitle.getText().toString(),Toast.*LENGTH\_SHORT*).show();  
  
 *//creating the dialog to delete the userExercise* AlertDialog.Builder builder = new AlertDialog.Builder(*context*);  
  
 builder.setTitle("delete exercise");  
  
 builder.setMessage("Do you really want to delete?");  
  
 *//delete* builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dataBaseHelper.deleteUserExercise(activeUserName,userExercisesInnerJoinEx.getExerciseName(), userExercisesInnerJoinEx.getUserExerciseID());  
 arrayList.remove(position);  
 notifyDataSetChanged();  
 }  
 });  
  
 *//cancel* builder.setNegativeButton("No", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 dialog.cancel();  
 }  
 });  
  
 *//Creating and showing the dialog* AlertDialog dialog = builder.create();  
  
 dialog.show();  
  
 }  
 });  
  
 imageView.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, userExercisesInnerJoinEx.getExerciseName(), Toast.*LENGTH\_SHORT*).show();  
  
 Intent intent1 = new Intent(*context*, ExecuteExerciseActivity.class);  
  
 intent1.putExtra("ExecuteUserExerciseID", userExercisesInnerJoinEx.getUserExerciseID());  
 intent1.putExtra("ExecuteExerciseID", userExercisesInnerJoinEx.getExerciseID());  
 intent1.putExtra("activeUserName",activeUserName);  
 *context*.startActivity(intent1);  
 ((Activity)*context*).finish();  
 }  
 });  
  
 txtMainTitle.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 Toast.*makeText*(*context*, userExercisesInnerJoinEx.getExerciseName(), Toast.*LENGTH\_SHORT*).show();  
  
 Intent intent1 = new Intent(*context*, ExecuteExerciseActivity.class);  
  
 intent1.putExtra("ExecuteUserExerciseID", userExercisesInnerJoinEx.getUserExerciseID());  
 intent1.putExtra("ExecuteExerciseID", userExercisesInnerJoinEx.getExerciseID());  
 intent1.putExtra("activeUserName",activeUserName);  
 *context*.startActivity(intent1);  
 ((Activity)*context*).finish();  
 }  
 });  
  
 return convertView;  
 }  
}

## UserExercisesInnerJoinEx

package com.example.fitnote13022021;  
  
public class UserExercisesInnerJoinEx {  
  
 *//UserExercisesInnerJoinEx (int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail)* private int userExerciseID;  
 private int exerciseID;  
 private String exerciseName;  
 private int exercisePic;  
 private String exerciseDetail;  
  
 *//constructors* public UserExercisesInnerJoinEx(int userExerciseID, int exerciseID, String exerciseName, int exercisePic, String exerciseDetail) {  
 this.userExerciseID = userExerciseID;  
 this.exerciseID = exerciseID;  
 this.exerciseName = exerciseName;  
 this.exercisePic = exercisePic;  
 this.exerciseDetail = exerciseDetail;  
 }  
  
  
 *// toString is necessary for printing the contents of a class object* @Override  
 public String toString() {  
 return "UserExercisesInerJoinEx{" +  
 "userExerciseID=" + userExerciseID +  
 ", exerciseID=" + exerciseID +  
 ", exerciseName='" + exerciseName + '\'' +  
 ", exercisePic=" + exercisePic +  
 ", exerciseDetail='" + exerciseDetail + '\'' +  
 '}';  
 }  
  
  
 *//Getters and Setters* public int getUserExerciseID() {  
 return userExerciseID;  
 }  
  
 public void setUserExerciseID(int userExerciseID) {  
 this.userExerciseID = userExerciseID;  
 }  
  
 public int getExerciseID() {  
 return exerciseID;  
 }  
  
 public void setExerciseID(int exerciseID) {  
 this.exerciseID = exerciseID;  
 }  
  
 public String getExerciseName() {  
 return exerciseName;  
 }  
  
 public void setExerciseName(String exerciseName) {  
 this.exerciseName = exerciseName;  
 }  
  
 public int getExercisePic() {  
 return exercisePic;  
 }  
  
 public void setExercisePic(int exercisePic) {  
 this.exercisePic = exercisePic;  
 }  
  
 public String getExerciseDetail() {  
 return exerciseDetail;  
 }  
  
 public void setExerciseDetail(String exerciseDetail) {  
 this.exerciseDetail = exerciseDetail;  
 }  
  
}

## UsersAdapter

package com.example.fitnote13022021;  
  
import android.app.AlertDialog;  
import android.app.DatePickerDialog;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Color;  
import android.os.Bundle;  
import android.provider.MediaStore;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.BaseAdapter;  
import android.widget.Button;  
import android.widget.DatePicker;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.RadioButton;  
import android.widget.RadioGroup;  
import android.widget.SeekBar;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import androidx.annotation.Nullable;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
  
public class UsersAdapter extends BaseAdapter {  
  
 private ArrayList<User> arrayList;  
 private User user;  
  
 *//get activeUserName to not delete and change him* String activeUserName;  
 int activeUserLevel;  
  
 public static Context *context*;  
  
 public UsersAdapter(ArrayList<User> arrayList, Context context, String activeUserName, int activeUserLevel){  
 this.arrayList = arrayList;  
 this.*context* = context;  
 this.activeUserName = activeUserName;  
 this.activeUserLevel = activeUserLevel;  
 }  
  
 @Override  
 public int getCount() {  
 return arrayList.size();  
 }  
  
 @Override  
 public Object getItem(int position) {  
 return arrayList.get(position);  
 }  
  
 @Override  
 public long getItemId(int position) {  
 return 0;  
 }  
  
 public class MyHolder {  
 ImageView imgUserPicUserLayout;  
 TextView txtUserNameShowUserLayout, txtUserGenderShowUserLayout;  
  
 public MyHolder(View v) {  
 imgUserPicUserLayout = (ImageView) v.findViewById(R.id.*imgUserPicUserLayout*);  
  
 txtUserNameShowUserLayout = (TextView) v.findViewById(R.id.*txtUserNameShowUserLayout*);  
 txtUserGenderShowUserLayout = (TextView) v.findViewById(R.id.*txtUserGenderShowUserLayout*);  
 }  
  
 }  
  
 *// This is a very important method:  
 // You can write your code in this function  
 // You can set your vies methods:* @Override  
 public View getView(int position, View convertView, ViewGroup parent) {  
  
 View view = convertView;  
  
 MyHolder holder;  
  
 *// using polymorphism on users to get Admin info as well* user = arrayList.get(position);  
  
 *//Check if an existing view is being reused, otherwise inflate the view* if (view == null) {  
 view = LayoutInflater.*from*(*context*).inflate(R.layout.*user\_layout*, null);  
 holder = new MyHolder(view);  
 view.setTag(holder);  
 }else {  
 holder = (MyHolder) view.getTag();  
 }  
  
 *//Setting the userPic to be the user's pic  
 //changing the userPicture to show the user's picture* Bitmap userPic = PictureFileHelper.*getPic*(*context*, user.getUserName());  
 if(userPic != null && holder.imgUserPicUserLayout != null){  
 holder.imgUserPicUserLayout.setImageBitmap(userPic);  
 }  
  
 *//setting textviews to show userExercise info* holder.txtUserNameShowUserLayout.setText(user.getUserName());  
 holder.txtUserGenderShowUserLayout.setText(user.getUserGender());  
  
 return view;  
 }  
  
}

## ViewExercisesResultsActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.os.Bundle;  
import android.widget.ImageView;  
import android.widget.ListView;  
import android.widget.SearchView;  
import android.widget.TextView;  
  
import java.text.DecimalFormat;  
import java.util.ArrayList;  
import java.util.Locale;  
  
public class ViewExercisesResultsActivity extends AppCompatActivity {  
  
 *//Initialize variables* public static String *activeUserName* = null;  
  
 DataBaseHelper dataBaseHelper;  
  
 ImageView imgUserPictureExercisesResultsScreen;  
  
 TextView txtBMIDoneExercises;  
  
 ListView listViewUserExercisesDone;  
  
 ArrayList<UserExercise> userExercisesDone;  
  
 ArrayList<UserExercise> filteredUserExercisesDone;  
  
 SearchView searchViewUserExerciseDoneList;  
  
 UserExerciseDoneAdapter userExerciseDoneAdapter;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_view\_done\_exercises*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//finding XML parts / Assign variables* imgUserPictureExercisesResultsScreen = findViewById(R.id.*imgUserPictureExercisesResultsScreen*);  
 txtBMIDoneExercises = findViewById(R.id.*txtBMIDoneExercises*);  
 searchViewUserExerciseDoneList = findViewById(R.id.*searchViewUserExerciseDoneList*);  
 listViewUserExercisesDone = findViewById(R.id.*listViewUserExercisesDone*);  
  
 *//setting filteredExercises to be exercises* filteredUserExercisesDone = userExercisesDone;  
  
 Intent intent = getIntent();  
  
 *//getting the active userName from getting the user who entered with log in with his name given from the intent* String userName = intent.getStringExtra("activeUserName");  
  
 *activeUserName* = userName;  
  
 *//changing the userPicture to show the user's picture* Bitmap userPic = PictureFileHelper.*getPic*(this, *activeUserName*);  
 if(userPic != null && imgUserPictureExercisesResultsScreen != null){  
 imgUserPictureExercisesResultsScreen.setImageBitmap(userPic);  
 }  
  
 *//getting the exercises done by the user* userExercisesDone = dataBaseHelper.getUserExercisesDoneByUserName(*activeUserName*);  
  
 userExerciseDoneAdapter = new UserExerciseDoneAdapter(userExercisesDone, ViewExercisesResultsActivity.this);  
  
 listViewUserExercisesDone.setAdapter(userExerciseDoneAdapter);  
  
 *//initialize the searchWidget in order to filter exercises* initSearchWidgets();  
  
 *//changing textView to show user's BMI  
 //Getting the user's weight and height* User activeUser = dataBaseHelper.getUserByName(*activeUserName*);  
 double userWeight = activeUser.getUserWeight();  
 double userHeight = activeUser.getUserHeight() \* 0.01;*//turning cm to m  
 //BMI = kg/m2* double userBMI = userWeight / (userHeight\*userHeight);  
  
 String BMIMessageToUser = *activeUserName* + " BMI is: " + new DecimalFormat("##.##").format(userBMI);  
  
 txtBMIDoneExercises.setText(BMIMessageToUser);  
 *//A BMI of 25.0 or more is overweight, while the healthy range is 18.5 to 24.9.* }  
  
 private void initSearchWidgets() {  
  
 SearchView searchView = (SearchView)searchViewUserExerciseDoneList;  
  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 *//this method is called every time a user  
 //puts in any character into the search view  
 //literally any chang :D* @Override  
 public boolean onQueryTextChange(String newText) {  
  
 filteredUserExercisesDone = new ArrayList<UserExercise>();  
  
 *//same as a regular for lop* for (UserExercise userExercise: userExercisesDone)  
 {  
  
 *//getting the exerciseName by exerciseID* String exerciseName = dataBaseHelper.getExerciseByID(userExercise.getExerciseID()).getExerciseName();  
  
 *//if a userExercise has one of the letter in the date* if(userExercise.getDate().toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUserExercisesDone.add(userExercise);  
 }  
  
 *//if a userExercise has one of the letter in the time* else if(String.*valueOf*(userExercise.getTime()).contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUserExercisesDone.add(userExercise);  
 }  
  
 *//if a userExercise has one of the letter in the rating* else if(String.*valueOf*(userExercise.getRating()).contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUserExercisesDone.add(userExercise);  
 }  
  
 *//if a userExercise has one of the letter in the repetition* else if(String.*valueOf*(userExercise.getRepetition()).contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUserExercisesDone.add(userExercise);  
 }  
  
 *//if a userExercise has one of the letter in the name* else if(exerciseName.toLowerCase().contains(newText.toLowerCase(Locale.*ROOT*))){  
 filteredUserExercisesDone.add(userExercise);  
 }  
  
 }  
  
 UserExerciseDoneAdapter exerciseAdaptor = new UserExerciseDoneAdapter(filteredUserExercisesDone, ViewExercisesResultsActivity.this);  
  
 listViewUserExercisesDone.setAdapter(exerciseAdaptor);  
  
 return false;  
 };  
 });  
  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ProgramUserActivity.class);  
  
 intent.putExtra("activeUserName", *activeUserName*);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

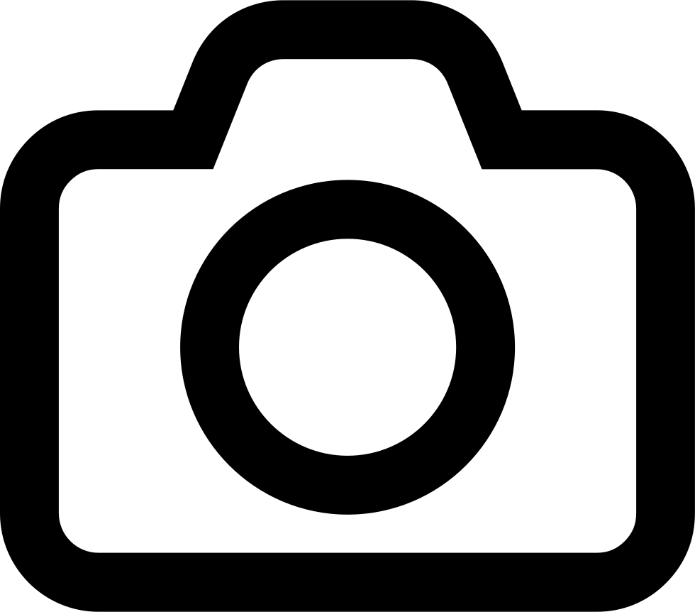
## WhatsAppSendActivity

package com.example.fitnote13022021;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlarmManager;  
import android.content.Intent;  
import android.content.pm.PackageManager;  
import android.net.Uri;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
import android.widget.Toast;  
  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
  
public class WhatsAppSendActivity extends AppCompatActivity {  
  
 *//Initialize variables* DataBaseHelper dataBaseHelper;  
  
 TextView tvName, tvPhone;  
  
 EditText etWhatsAppMessage;  
  
 Button btSendWhatsAppMessage;  
  
 String userExercisesSummaryText = "Here is my data on all the exercises I did in the last week successfully:";  
  
 String activeUserName;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_whats\_app\_send*);  
  
 dataBaseHelper = new DataBaseHelper(this);  
  
 *//Assign variables* tvName = findViewById(R.id.*tvName*);  
 tvPhone = findViewById(R.id.*tvPhone*);  
 etWhatsAppMessage = findViewById(R.id.*etWhatsAppMessage*);  
 btSendWhatsAppMessage = findViewById(R.id.*btSendWhatsAppMessage*);  
  
 Intent intent = getIntent();  
  
 activeUserName = intent.getStringExtra("activeUserName");  
  
 *//getting information from database to send on WhatsApp* ArrayList<UserExercise> userExercisesDoneByUser = dataBaseHelper.getUserExercisesDoneByUserName(activeUserName);  
  
 *//Takes only the UserExercises done in the last week* ArrayList<UserExercise> userExercisesDoneByUserLastWeek = new ArrayList<UserExercise>();  
  
 int arrayListUserExercisesSize= userExercisesDoneByUser.size();  
  
 Calendar dayOfExercise = Calendar.*getInstance*();  
  
  
 *//go over userExercisesDoneByUser to get exercises done in last week* for (int i=0; i<arrayListUserExercisesSize; i++) {  
  
 UserExercise userExercise = userExercisesDoneByUser.get(i);  
  
 String date = userExercise.getDate();  
  
 *//EXAMPLE DATE: "FEB 28 2021" 11 (year - 7-10, day - 5-6, month 0-2)* String[] splited = date.split("\\s+");  
  
 int year = Integer.*parseInt*(splited[2]);  
 int day = Integer.*parseInt*(splited[1]);  
 int month = getMonthNumber(date);  
  
 dayOfExercise.set(Calendar.*YEAR*, year);  
 *//month starts at 0 in Calendar* dayOfExercise.set(Calendar.*MONTH*, month-1);  
 dayOfExercise.set(Calendar.*DAY\_OF\_MONTH*, day);  
 dayOfExercise.set(Calendar.*HOUR\_OF\_DAY*, 0);  
 dayOfExercise.set(Calendar.*MINUTE*, 0);  
 dayOfExercise.set(Calendar.*SECOND*, 0);  
 dayOfExercise.set(Calendar.*MILLISECOND*, 0);  
  
 Calendar currentDate = Calendar.*getInstance*();  
  
 int dayDifference = 0;  
 *//get the day difference between today and the date of the exercise  
 //if the two dates are not null* if (dayOfExercise != null && currentDate != null){  
 dayDifference = (int)( (currentDate.getTime().getTime() - dayOfExercise.getTime().getTime()) / (1000 \* 60 \* 60 \* 24));  
 }  
  
 if (dayDifference >=0 && dayDifference <= 7){  
 userExercisesDoneByUserLastWeek.add(userExercise);  
 }  
  
 }  
  
 *//We need the exercises in order to send the name of the exercise with the data* ArrayList<Exercise> exercises = dataBaseHelper.getExercises();  
  
 int arrayListExercisesSize = exercises.size();  
  
 *//go over userExercisesDoneByUserLastWeek to take data* int sizeExercisesLastWeek = userExercisesDoneByUserLastWeek.size();  
 for (int i=0; i<sizeExercisesLastWeek; i++) {  
  
 *//(userExerciseID INTEGER PRIMARY KEY AUTOINCREMENT, userName TEXT,  
 // exerciseID INTEGER, date TEXT, time(SEC) INTEGER, rating INTEGER, repetition INTEGER)* UserExercise userExercise = userExercisesDoneByUserLastWeek.get(i);  
  
 int exerciseID = userExercise.getExerciseID();  
  
 String exerciseName ="";  
  
 *//getting the exercise name* for (int j=0; j<arrayListExercisesSize; j++){  
 Exercise exercise = exercises.get(j);  
 if(exerciseID == exercise.getExerciseID()){  
 exerciseName = exercise.getExerciseName();  
 }  
 }  
  
 String date = userExercise.getDate();  
 int time = userExercise.getTime();  
 int rating = userExercise.getRating();  
 int repetition = userExercise.getRepetition();  
  
 userExercisesSummaryText += "\n" + "<---->" + "\n";  
  
 userExercisesSummaryText += " - The exercise: "  
 + exerciseName + "\n";  
  
 userExercisesSummaryText += " - The date in which the exercise was done:"  
 + date + "\n";  
  
 userExercisesSummaryText += " - The time it took to perform the exercise: "  
 + getDurationString(time) + "\n";  
  
 userExercisesSummaryText += " - The rating " + activeUserName + " gave to the exercise: "  
 + getRating(rating) + "\n";  
  
 userExercisesSummaryText += " - " + activeUserName + " did the exercise "  
 + repetition + "times" + "\n";  
  
  
 }  
  
 *//get contactName* String contactName = intent.getStringExtra("contactName");  
 *//get contactNumber* String contactNumber = intent.getStringExtra("contactNumber");  
  
 String tvNameTxt = tvName.getText().toString();  
 String tvPhoneTxt = tvPhone.getText().toString();  
  
 tvName.setText(tvNameTxt + contactName);  
 tvPhone.setText(tvPhoneTxt + contactNumber);  
  
 btSendWhatsAppMessage.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *//get text to send from etWhatsAppMessage* String textMessage = etWhatsAppMessage.getText().toString() + "\n";  
 textMessage += userExercisesSummaryText;  
 sendMessage(contactNumber, textMessage);  
 }  
 });  
  
 }  
  
 private void sendMessage(String phoneNumber, String text){  
 *//boolean to check if WhatsApp exists* boolean installed = isAppInstalled("com.whatsapp");  
  
 if (installed) {  
 Intent intent = new Intent(Intent.*ACTION\_VIEW*);  
 *//WhatsApp message requires phone number & text(the message itself)* intent.setData(Uri.*parse*("http://api.whatsapp.com/send?phone="+phoneNumber+"&text="+text));  
 startActivity(intent);  
 }else{  
 Toast.*makeText*(this, "WhatsApp is not installed!", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
  
 *// A method to check if WhatsApp exists on the users phone* private boolean isAppInstalled(String s) {  
  
 PackageManager packageManager = getPackageManager();  
 boolean is\_installed;  
  
 try {  
 packageManager.getPackageInfo(s, PackageManager.*GET\_ACTIVITIES*);  
 is\_installed = true;  
 } catch (PackageManager.NameNotFoundException e) {  
 e.printStackTrace();  
 is\_installed = false;  
 }  
  
 return is\_installed;  
  
 }  
  
 private String getDurationString(int seconds) {  
  
 int hours = seconds / 3600;  
 int minutes = (seconds % 3600) / 60;  
 seconds = seconds % 60;  
  
 return twoDigitString(hours) + " : " + twoDigitString(minutes) + " : " + twoDigitString(seconds);  
 }  
  
 private String twoDigitString(int number) {  
  
 if (number == 0) {  
 return "00";  
 }  
  
 if (number / 10 == 0) {  
 return "0" + number;  
 }  
  
 return String.*valueOf*(number);  
 }  
  
 private String getRating(int rating){  
  
 String ratingString = "Easy";  
  
 switch (rating){  
 case 1:  
 *//Hard* ratingString = "Hard";  
 break;  
  
 case 2:  
 *//Hard-Medium* ratingString = "Hard-Medium";  
 break;  
  
 case 3:  
 *//Medium* ratingString = "Medium";  
 break;  
  
 case 4:  
 *//Easy-Medium* ratingString = "Easy-Medium";  
 break;  
  
 case 5:  
 *//Easy* ratingString = "Easy";  
 break;  
  
 }  
  
 return ratingString;  
  
  
 }  
  
 *//makes the real month a number month (JAN -> 1)* private int getMonthNumber(String date) {  
 if(date.contains("JAN"))  
 return 1;  
 if(date.contains("FEB"))  
 return 2;  
 if(date.contains("MAR"))  
 return 3;  
 if(date.contains("APR"))  
 return 4;  
 if(date.contains("MAY"))  
 return 5;  
 if(date.contains("JUN"))  
 return 6;  
 if(date.contains("JUL"))  
 return 7;  
 if(date.contains("AUG"))  
 return 8;  
 if(date.contains("SEP"))  
 return 9;  
 if(date.contains("OCT"))  
 return 10;  
 if(date.contains("NOV"))  
 return 11;  
 if(date.contains("DEC"))  
 return 12;  
  
 *//default should never happen* return 1;  
 }  
  
 *//a method to go back to ProgramUserActivity when clicking back arrow in bottom* @Override  
 public void onBackPressed() {  
  
 Intent intent = new Intent(this, ShareActivity.class);  
  
 intent.putExtra("activeUserName", activeUserName);  
  
 startActivity(intent);  
  
 finish();  
  
 }  
  
}

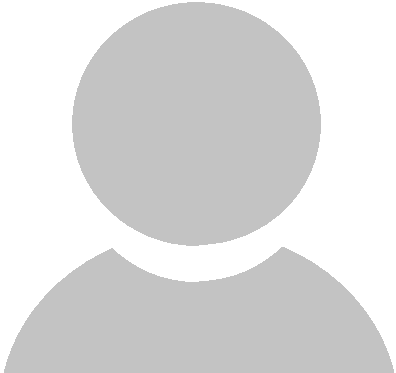
## res Folder

## drawable

cemra.png



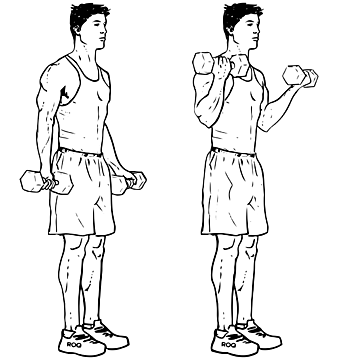
default\_user.png



easy\_text.png

H:\Program_files_H\Projects_android\FitNote13022021\app\src\main\res\drawable\easy_text.png

hammer\_curls.png



hard\_text.png

H:\Program_files_H\Projects_android\FitNote13022021\app\src\main\res\drawable\hard_text.png

ic\_baseline\_account\_circle.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M12,2C6.48,2 2,6.48 2,12s4.48,10 10,10 10,-4.48 10,-10S17.52,2 12,2zM12,5c1.66,0 3,1.34 3,3s-1.34,3 -3,3 -3,-1.34 -3,-3 1.34,-3 3,-3zM12,19.2c-2.5,0 -4.71,-1.28 -6,-3.22 0.03,-1.99 4,-3.08 6,-3.08 1.99,0 5.97,1.09 6,3.08 -1.29,1.94 -3.5,3.22 -6,3.22z"/>  
</vector>

ic\_baseline\_architecture.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M6.36,18.78L6.61,21l1.62,-1.54l2.77,-7.6c-0.68,-0.17 -1.28,-0.51 -1.77,-0.98L6.36,18.78z"/>  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M14.77,10.88c-0.49,0.47 -1.1,0.81 -1.77,0.98l2.77,7.6L17.39,21l0.26,-2.22L14.77,10.88z"/>  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M15,8c0,-1.3 -0.84,-2.4 -2,-2.82V3h-2v2.18C9.84,5.6 9,6.7 9,8c0,1.66 1.34,3 3,3S15,9.66 15,8zM12,9c-0.55,0 -1,-0.45 -1,-1c0,-0.55 0.45,-1 1,-1s1,0.45 1,1C13,8.55 12.55,9 12,9z"/>  
</vector>

ic\_baseline\_date\_range.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M9,11L7,11v2h2v-2zM13,11h-2v2h2v-2zM17,11h-2v2h2v-2zM19,4h-1L18,2h-2v2L8,4L8,2L6,2v2L5,4c-1.11,0 -1.99,0.9 -1.99,2L3,20c0,1.1 0.89,2 2,2h14c1.1,0 2,-0.9 2,-2L21,6c0,-1.1 -0.9,-2 -2,-2zM19,20L5,20L5,9h14v11z"/>  
</vector>

ic\_baseline\_preson.xml

<vector android:height="24dp" android:tint="#FFB72B"  
 android:viewportHeight="24" android:viewportWidth="24"  
 android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">  
 <path android:fillColor="@android:color/white" android:pathData="M12,12c2.21,0 4,-1.79 4,-4s-1.79,-4 -4,-4 -4,1.79 -4,4 1.79,4 4,4zM12,14c-2.67,0 -8,1.34 -8,4v2h16v-2c0,-2.66 -5.33,-4 -8,-4z"/>  
</vector>

ic\_baseline\_phonelink\_lock.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M19,1L9,1c-1.1,0 -2,0.9 -2,2v3h2L9,4h10v16L9,20v-2L7,18v3c0,1.1 0.9,2 2,2h10c1.1,0 2,-0.9 2,-2L21,3c0,-1.1 -0.9,-2 -2,-2zM10.8,11L10.8,9.5C10.8,8.1 9.4,7 8,7S5.2,8.1 5.2,9.5L5.2,11c-0.6,0 -1.2,0.6 -1.2,1.2v3.5c0,0.7 0.6,1.3 1.2,1.3h5.5c0.7,0 1.3,-0.6 1.3,-1.2v-3.5c0,-0.7 -0.6,-1.3 -1.2,-1.3zM9.5,11h-3L6.5,9.5c0,-0.8 0.7,-1.3 1.5,-1.3s1.5,0.5 1.5,1.3L9.5,11z"/>  
</vector>

ic\_baseline\_show\_chart.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M3.5,18.49l6,-6.01 4,4L22,6.92l-1.41,-1.41 -7.09,7.97 -4,-4L2,16.99z"/>  
</vector>

ic\_baseline\_wc.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M5.5,22v-7.5L4,14.5L4,9c0,-1.1 0.9,-2 2,-2h3c1.1,0 2,0.9 2,2v5.5L9.5,14.5L9.5,22h-4zM18,22v-6h3l-2.54,-7.63C18.18,7.55 17.42,7 16.56,7h-0.12c-0.86,0 -1.63,0.55 -1.9,1.37L12,16h3v6h3zM7.5,6c1.11,0 2,-0.89 2,-2s-0.89,-2 -2,-2 -2,0.89 -2,2 0.89,2 2,2zM16.5,6c1.11,0 2,-0.89 2,-2s-0.89,-2 -2,-2 -2,0.89 -2,2 0.89,2 2,2z"/>  
</vector>

ic\_baseline\_wrench.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M22.7,19l-9.1,-9.1c0.9,-2.3 0.4,-5 -1.5,-6.9 -2,-2 -5,-2.4 -7.4,-1.3L9,6 6,9 1.6,4.7C0.4,7.1 0.9,10.1 2.9,12.1c1.9,1.9 4.6,2.4 6.9,1.5l9.1,9.1c0.4,0.4 1,0.4 1.4,0l2.3,-2.3c0.5,-0.4 0.5,-1.1 0.1,-1.4z"/>  
</vector>

ic\_launcher\_background.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="108dp"  
 android:height="108dp"  
 android:viewportWidth="108"  
 android:viewportHeight="108">  
 <path  
 android:fillColor="#3DDC84"  
 android:pathData="M0,0h108v108h-108z" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M9,0L9,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,0L19,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M29,0L29,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M39,0L39,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M49,0L49,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M59,0L59,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M69,0L69,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M79,0L79,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M89,0L89,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M99,0L99,108"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,9L108,9"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,19L108,19"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,29L108,29"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,39L108,39"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,49L108,49"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,59L108,59"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,69L108,69"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,79L108,79"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,89L108,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M0,99L108,99"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,29L89,29"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,39L89,39"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,49L89,49"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,59L89,59"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,69L89,69"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M19,79L89,79"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M29,19L29,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M39,19L39,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M49,19L49,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M59,19L59,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M69,19L69,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
 <path  
 android:fillColor="#00000000"  
 android:pathData="M79,19L79,89"  
 android:strokeWidth="0.8"  
 android:strokeColor="#33FFFFFF" />  
</vector>

ic\_launcher\_foreground.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:aapt="http://schemas.android.com/aapt"  
 android:width="108dp"  
 android:height="108dp"  
 android:viewportWidth="108"  
 android:viewportHeight="108">  
 <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">  
 <aapt:attr name="android:fillColor">  
 <gradient  
 android:endX="85.84757"  
 android:endY="92.4963"  
 android:startX="42.9492"  
 android:startY="49.59793"  
 android:type="linear">  
 <item  
 android:color="#44000000"  
 android:offset="0.0" />  
 <item  
 android:color="#00000000"  
 android:offset="1.0" />  
 </gradient>  
 </aapt:attr>  
 </path>  
 <path  
 android:fillColor="#FFFFFF"  
 android:fillType="nonZero"  
 android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028 31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"  
 android:strokeWidth="1"  
 android:strokeColor="#00000000" />  
</vector>

ic\_plus.xml

<vector android:height="24dp" android:tint="#FF9800"  
 android:viewportHeight="24" android:viewportWidth="24"  
 android:width="24dp" xmlns:android="http://schemas.android.com/apk/res/android">  
 <path android:fillColor="@android:color/white" android:pathData="M13,7h-2v4L7,11v2h4v4h2v-4h4v-2h-4L13,7zM12,2C6.48,2 2,6.48 2,12s4.48,10 10,10 10,-4.48 10,-10S17.52,2 12,2zM12,20c-4.41,0 -8,-3.59 -8,-8s3.59,-8 8,-8 8,3.59 8,8 -3.59,8 -8,8z"/>  
</vector>

ic\_results.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal"  
 android:autoMirrored="true">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M19,3h-4.18C14.4,1.84 13.3,1 12,1c-1.3,0 -2.4,0.84 -2.82,2L5,3c-1.1,0 -2,0.9 -2,2v14c0,1.1 0.9,2 2,2h14c1.1,0 2,-0.9 2,-2L21,5c0,-1.1 -0.9,-2 -2,-2zM12,3c0.55,0 1,0.45 1,1s-0.45,1 -1,1 -1,-0.45 -1,-1 0.45,-1 1,-1zM14,17L7,17v-2h7v2zM17,13L7,13v-2h10v2zM17,9L7,9L7,7h10v2z"/>  
</vector>

ic\_settings.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M19.14,12.94c0.04,-0.3 0.06,-0.61 0.06,-0.94c0,-0.32 -0.02,-0.64 -0.07,-0.94l2.03,-1.58c0.18,-0.14 0.23,-0.41 0.12,-0.61l-1.92,-3.32c-0.12,-0.22 -0.37,-0.29 -0.59,-0.22l-2.39,0.96c-0.5,-0.38 -1.03,-0.7 -1.62,-0.94L14.4,2.81c-0.04,-0.24 -0.24,-0.41 -0.48,-0.41h-3.84c-0.24,0 -0.43,0.17 -0.47,0.41L9.25,5.35C8.66,5.59 8.12,5.92 7.63,6.29L5.24,5.33c-0.22,-0.08 -0.47,0 -0.59,0.22L2.74,8.87C2.62,9.08 2.66,9.34 2.86,9.48l2.03,1.58C4.84,11.36 4.8,11.69 4.8,12s0.02,0.64 0.07,0.94l-2.03,1.58c-0.18,0.14 -0.23,0.41 -0.12,0.61l1.92,3.32c0.12,0.22 0.37,0.29 0.59,0.22l2.39,-0.96c0.5,0.38 1.03,0.7 1.62,0.94l0.36,2.54c0.05,0.24 0.24,0.41 0.48,0.41h3.84c0.24,0 0.44,-0.17 0.47,-0.41l0.36,-2.54c0.59,-0.24 1.13,-0.56 1.62,-0.94l2.39,0.96c0.22,0.08 0.47,0 0.59,-0.22l1.92,-3.32c0.12,-0.22 0.07,-0.47 -0.12,-0.61L19.14,12.94zM12,15.6c-1.98,0 -3.6,-1.62 -3.6,-3.6s1.62,-3.6 3.6,-3.6s3.6,1.62 3.6,3.6S13.98,15.6 12,15.6z"/>  
</vector>

ic\_share.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M18,16.08c-0.76,0 -1.44,0.3 -1.96,0.77L8.91,12.7c0.05,-0.23 0.09,-0.46 0.09,-0.7s-0.04,-0.47 -0.09,-0.7l7.05,-4.11c0.54,0.5 1.25,0.81 2.04,0.81 1.66,0 3,-1.34 3,-3s-1.34,-3 -3,-3 -3,1.34 -3,3c0,0.24 0.04,0.47 0.09,0.7L8.04,9.81C7.5,9.31 6.79,9 6,9c-1.66,0 -3,1.34 -3,3s1.34,3 3,3c0.79,0 1.5,-0.31 2.04,-0.81l7.12,4.16c-0.05,0.21 -0.08,0.43 -0.08,0.65 0,1.61 1.31,2.92 2.92,2.92 1.61,0 2.92,-1.31 2.92,-2.92s-1.31,-2.92 -2.92,-2.92z"/>  
</vector>

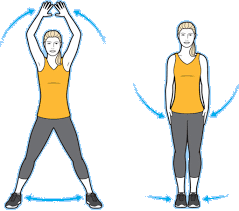
ic\_timer.xml

<vector xmlns:android="http://schemas.android.com/apk/res/android"  
 android:width="24dp"  
 android:height="24dp"  
 android:viewportWidth="24"  
 android:viewportHeight="24"  
 android:tint="?attr/colorControlNormal">  
 <path  
 android:fillColor="@android:color/white"  
 android:pathData="M22,5.72l-4.6,-3.86 -1.29,1.53 4.6,3.86L22,5.72zM7.88,3.39L6.6,1.86 2,5.71l1.29,1.53 4.59,-3.85zM12.5,8L11,8v6l4.75,2.85 0.75,-1.23 -4,-2.37L12.5,8zM12,4c-4.97,0 -9,4.03 -9,9s4.02,9 9,9c4.97,0 9,-4.03 9,-9s-4.03,-9 -9,-9zM12,20c-3.87,0 -7,-3.13 -7,-7s3.13,-7 7,-7 7,3.13 7,7 -3.13,7 -7,7z"/>  
</vector>

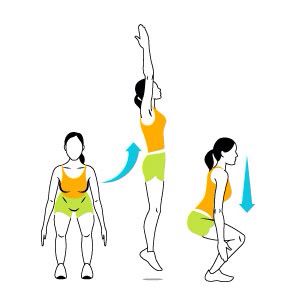
incline\_pushup.png



jumping\_jacks.png



jumping\_squat.png



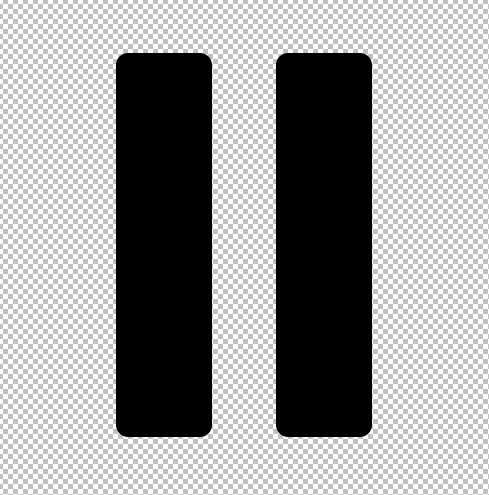
logo.png



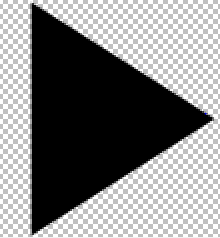
logo\_only\_pic.png



pause\_button.png



play\_button.png



pushup.png



ropejump.png



running.png



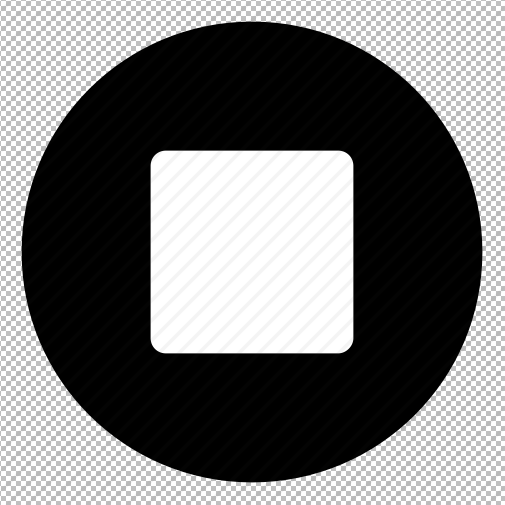
shoulder\_press.png



squat.png



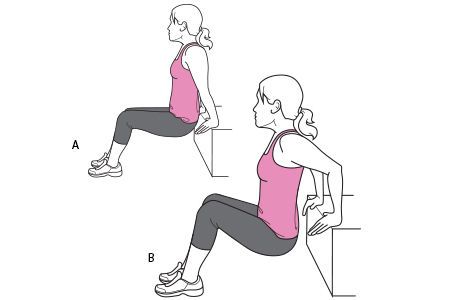
stop\_button.png



swimming.png



triceps\_dips.png



## layout

activity\_add\_exercises.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=".AddExerciseActivity"  
 android:background="@color/white"  
 android:orientation="vertical">  
  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="20sp"  
 android:src="@drawable/logo"></ImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtMainTitle"  
 android:text="Choose your Exercise:"  
 android:textSize="30dp"  
 android:textColor="@color/black"  
 ></TextView>  
  
 <SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchViewExerciseList"  
 android:iconifiedByDefault="false"  
 android:queryHint="Exercise Name"  
 ></SearchView>  
  
 <ListView  
 android:id="@+id/listViewExercises"  
 android:layout\_width="match\_parent"  
 android:layout\_height="350dp"></ListView>  
  
 <Button  
 android:id="@+id/btCancel"  
 android:layout\_width="150dp"  
 android:layout\_height="50dp"  
 android:layout\_gravity="end"  
 android:layout\_marginTop="5dp"  
 android:layout\_marginEnd="20dp"  
 android:text="Cancel"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF9800"/>  
  
</LinearLayout>

activity\_edit\_user.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=".EditUserActivity"  
 android:orientation="vertical">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:text="Edit User:"  
 android:textSize="35sp"  
 android:textColor="@color/colorPrimary"  
 android:id="@+id/txtTitleEditUser"  
 ></TextView>  
  
 <EditText  
 android:id="@+id/etPasswordEditUser"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:inputType="textPassword"></EditText>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Weight (kilograms):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarWeightEditUser"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="40" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarWeightEditUser"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="200"  
 android:min="40"  
 android:layout\_marginTop="15dp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
  
 android:layout\_weight="1"  
 android:text="Height (centimeters):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarHeightEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="50" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarHeightEditUser"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="250"  
 android:min="50"  
 android:layout\_marginTop="15dp"/>  
  
 <TextView  
 android:id="@+id/tv\_date"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Select BirthDate:"  
 android:textSize="20sp"></TextView>  
  
 <Button  
 android:id="@+id/btDatePickerButtonEditUser"  
 style="?android:spinnerStyle"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:onClick="openDatePicker"  
 android:text="JAN 01 2020"  
 android:textColor="#000000"  
 android:textSize="30sp"></Button>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Chose Gender:"  
 android:textSize="14sp"></TextView>  
  
 <RadioGroup  
 android:id="@+id/radioGroupGenderEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <RadioButton  
 android:id="@+id/radio\_maleEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Male" />  
  
 <RadioButton  
 android:id="@+id/radio\_femaleEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Female" />  
  
 <RadioButton  
 android:id="@+id/radio\_otherEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Other" />  
  
  
 >  
 </RadioGroup>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Chose UserLevel:"  
 android:textSize="14sp"></TextView>  
  
 <RadioGroup  
 android:id="@+id/radioGroupUserLevelEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <RadioButton  
 android:id="@+id/radio\_NormalEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Normal" />  
  
 <RadioButton  
 android:id="@+id/radio\_AdminEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Admin" />  
  
 <RadioButton  
 android:id="@+id/radio\_Super\_AdminEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Super Admin" />  
  
  
 >  
 </RadioGroup>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <Button  
 android:id="@+id/btUpdateEditUser"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:text="Update"  
 android:textAllCaps="false"  
 android:textColor="@color/black"  
 android:backgroundTint="#4CAF50"/>  
  
 <Button  
 android:id="@+id/btCancelEditUser"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:text="Cancel"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF7700"/>  
  
 >  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/btDeleteEditUser"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_gravity="center"  
 android:text="Delete"  
 android:textAllCaps="false"  
 android:textColor="#FFFFFF"  
 android:backgroundTint="#F44336"/>  
  
</LinearLayout>

activity\_execute\_exercise.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=".ExecuteExerciseActivity"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="300dp"  
 android:layout\_height="300dp"  
 android:src="@drawable/logo"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="50dp"  
 android:id="@+id/imgExercise"  
 ></ImageView>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="00 : 00 : 00"  
 android:gravity="center"  
 android:textSize="30sp"  
 android:textStyle="bold"  
 android:id="@+id/txtTime"  
 ></TextView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="70dp"  
 android:layout\_marginTop="100dp"  
 android:weightSum="2"  
 >  
  
 <ImageView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginHorizontal="30dp"  
 android:layout\_weight="1"  
 android:backgroundTint="@color/white"  
 android:textColor="@color/black"  
 android:id="@+id/btnFinish"  
 android:src="@drawable/stop\_button"  
 ></ImageView>  
  
 <ImageView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginHorizontal="30dp"  
 android:layout\_weight="1"  
 android:backgroundTint="@color/white"  
 android:textColor="@color/black"  
 android:id="@+id/btnPlay"  
 android:src="@drawable/play\_button"  
 ></ImageView>  
  
  
  
 </LinearLayout>  
  
  
  
</LinearLayout>

activity\_feed\_back.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=".FeedbackActivity"  
 android:orientation="vertical">  
  
 <TextView  
 android:id="@+id/txtFirstSentence"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="How much did you do?"  
 android:layout\_marginTop="50dp"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtTestInput"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 ></TextView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="35dp"  
 android:weightSum="4"  
 >  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="40sp"  
 android:text="Please enter here:"  
 android:textSize="15sp"  
 android:layout\_weight="1.5"  
 ></TextView>  
  
  
 <NumberPicker  
 android:id="@+id/numberPicker"  
 android:layout\_width="0dp"  
 android:layout\_height="50dp"  
 android:layout\_weight="1"  
 ></NumberPicker>  
  
 </LinearLayout>  
  
  
 <TextView  
 android:id="@+id/txtSecondSentence"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Rate the difficulty:"  
 android:layout\_marginTop="35dp"  
 android:textSize="18sp"  
 android:textStyle="bold"  
 ></TextView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="35dp"  
 android:weightSum="8"  
 >  
  
 <ImageView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:src="@drawable/hard\_text"></ImageView>  
  
 <RadioGroup  
 android:id="@+id/radioGroup"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="6"  
 android:orientation="horizontal">  
  
 <RadioButton  
 android:id="@+id/radio\_1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="1" />  
  
 <RadioButton  
 android:id="@+id/radio\_2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="2" />  
  
 <RadioButton  
 android:id="@+id/radio\_3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="3" />  
  
 <RadioButton  
 android:id="@+id/radio\_4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="4" />  
  
 <RadioButton  
 android:id="@+id/radio\_5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="5" />  
  
  
 >  
 </RadioGroup>  
  
 <ImageView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:src="@drawable/easy\_text"></ImageView>  
  
 </LinearLayout>  
  
 <Button  
 android:id="@+id/btnFinish"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="140dp"  
 android:layout\_gravity="center"  
 android:textSize="30sp"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF9800"></Button>  
  
  
  
</LinearLayout>

activity\_information.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=".InformationActivity"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="35dp"  
 android:src="@drawable/logo"  
 ></ImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="10dp"  
 android:gravity="center"  
 android:textSize="15sp"  
 android:textColor="#6F777E"  
 android:text="Register, pick an exercise and start training and sharing your success!!!"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="20dp"  
 android:gravity="center"  
 android:textSize="15sp"  
 android:textColor="#6F777E"  
 android:text="Start by seeing guides on how to do some exercises:"  
 ></TextView>  
  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="400dp"  
 android:id="@+id/listViewExercisesInfo"  
 ></ListView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 >  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="from"  
 android:textColor="#aabbcc"  
 />  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="FITNOTE"  
 android:textSize="16sp"  
 android:textColor="@color/colorPrimary"  
 />  
  
 </LinearLayout>  
  
</LinearLayout>

activity\_main.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=".MainScreenActivity"  
 android:background="@color/white"  
 android:orientation="vertical"  
 >  
  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="20dp"  
 android:src="@drawable/logo"  
 ></ImageView>  
  
 <EditText  
 android:id="@+id/etUserName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Username"  
 android:layout\_marginTop="35dp"  
 ></EditText>  
  
 <EditText  
 android:id="@+id/etPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:inputType="textPassword"  
 android:hint="Enter Password"  
 android:layout\_marginTop="35dp"  
 >  
  
 </EditText>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2"  
 android:layout\_marginTop="35dp">  
  
 <Button  
 android:id="@+id/btLogIn"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:text="Log In"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF9800"/>  
  
 <Button  
 android:id="@+id/btRegister"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:text="Register"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF9800"/>  
  
  
 </LinearLayout>  
  
</LinearLayout>

activity\_manager\_users.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=".ManagerUsersActivity"  
 android:orientation="vertical">  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="35dp"  
 android:src="@drawable/logo"  
 ></ImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="10dp"  
 android:gravity="center"  
 android:textSize="15sp"  
 android:textColor="@color/colorPrimary"  
 android:text="Welcome UNDETECTED USER"  
 android:id="@+id/txtWelcomeManageUsers"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:layout\_marginTop="20dp"  
 android:gravity="center"  
 android:textSize="15sp"  
 android:textColor="@color/colorPrimary"  
 android:text="Here you can view users information"  
 android:id="@+id/txtExplainManageUser"  
 ></TextView>  
  
 <SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchViewManageUsersList"  
 android:iconifiedByDefault="false"  
 android:queryHint="Enter Here"  
 ></SearchView>  
  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="400dp"  
 android:id="@+id/listViewManagerUsers"  
 ></ListView>  
  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 >  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="from"  
 android:textColor="#aabbcc"  
 />  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="FITNOTE"  
 android:textSize="16sp"  
 android:textColor="@color/colorPrimary"  
 />  
  
 </LinearLayout>  
  
</LinearLayout>

activity\_program\_user.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/white"  
 android:orientation="vertical"  
 tools:context=".ProgramUserActivity">  
  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="3"  
 android:orientation="horizontal"  
 android:layout\_marginTop="35dp"  
 >  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtMainTitle"  
 android:text="Welcome"  
 android:textSize="30dp"  
 android:textColor="@color/black"  
 android:layout\_weight="1"  
 ></TextView>  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:id="@+id/imgProgramUserPic"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_weight="2"  
 android:src="@drawable/default\_user"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="@color/black"  
 ></de.hdodenhof.circleimageview.CircleImageView>  
  
 </LinearLayout>  
  
 <SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchViewUserExerciseList"  
 android:iconifiedByDefault="false"  
 android:queryHint="Exercise Name"  
 ></SearchView>  
  
 <LinearLayout  
 android:id="@+id/layoutOfListView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="310dp"  
 android:orientation="vertical"  
 android:weightSum="3"  
 >  
  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/listViewUserExercises"  
 android:layout\_weight="2"  
 ></ListView>  
  
 <TextView  
 android:id="@+id/txtEmptyListMessage"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Its seems you don't have any exercises to do. you can add an exercise with the plus button :D"  
 android:textSize="20sp"  
 android:layout\_weight="1"  
 android:visibility="gone"  
 ></TextView>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 >  
  
 </LinearLayout>  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:id="@+id/btnAddExercise"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_gravity="end"  
 android:layout\_marginLeft="150dp"  
 android:src="@drawable/ic\_plus"  
 android:text="+"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="#00FFFFFF"  
 ></de.hdodenhof.circleimageview.CircleImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Select Your Notification:"  
 android:textSize="16sp"  
 android:textColor="@color/colorPrimary"  
 android:textStyle="bold"  
 android:layout\_marginTop="20dp"  
 android:layout\_gravity="bottom"  
 ></TextView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:weightSum="6"  
 android:layout\_marginTop="10dp"  
 >  
  
 <TextView  
 android:layout\_width="80dp"  
 android:layout\_height="70dp"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginRight="25dp"  
 android:id="@+id/tv\_timer"  
 android:textSize="15sp"  
 android:text="0:00"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:drawableTop="@drawable/ic\_timer"  
 android:background="@android:drawable/editbox\_background"  
 ></TextView>  
  
 <Button  
 android:id="@+id/btnSetAlarm"  
 android:textSize="10sp"  
 android:layout\_width="70dp"  
 android:layout\_height="50dp"  
 android:layout\_gravity="end"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginRight="10dp"  
 android:layout\_weight="1"  
 android:backgroundTint="#00BCD4"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:text="SET"></Button>  
 <Button  
 android:id="@+id/btnCancelAlarm"  
 android:textSize="10sp"  
 android:layout\_width="85dp"  
 android:layout\_height="50dp"  
 android:layout\_gravity="end"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginRight="35dp"  
 android:layout\_weight="1"  
 android:backgroundTint="#00BCD4"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:text="DELETE"></Button>  
  
  
 </LinearLayout>  
  
  
  
</LinearLayout>

activity\_register.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/white"  
 android:orientation="vertical"  
 tools:context=".RegisterActivity">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Profile pic:"  
 android:layout\_marginTop="10sp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10sp">  
  
 <ImageView  
 android:id="@+id/imgProfilePic"  
 android:layout\_width="65dp"  
 android:layout\_height="65dp"  
 android:src="@drawable/default\_user"  
 ></ImageView>  
  
 <ImageView  
 android:id="@+id/imgCemraButton"  
 android:layout\_width="35dp"  
 android:layout\_height="35dp"  
 android:layout\_gravity="bottom"  
 android:layout\_marginLeft="30dp"  
 android:src="@drawable/cemra"></ImageView>  
  
  
 >  
 </LinearLayout>  
  
  
 <EditText  
 android:id="@+id/etUserName"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Username"  
 android:inputType="textPersonName"></EditText>  
  
 <EditText  
 android:id="@+id/etPassword"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Password"  
 android:inputType="textPassword"></EditText>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Weight (kilograms):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarWeight"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="40" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarWeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="200"  
 android:min="40"  
 android:layout\_marginTop="15dp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
  
 android:layout\_weight="1"  
 android:text="Height (centimeters):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarHeight"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="50" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarHeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="250"  
 android:min="50"  
 android:layout\_marginTop="15dp"/>  
  
 <TextView  
 android:id="@+id/tv\_date"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Select BirthDate:"  
 android:textSize="20sp"></TextView>  
  
 <Button  
 android:id="@+id/btDatePickerButton"  
 style="?android:spinnerStyle"  
 android:layout\_width="250dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center"  
 android:onClick="openDatePicker"  
 android:text="JAN 01 2020"  
 android:textColor="#000000"  
 android:textSize="30sp"></Button>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Chose Gender:"  
 android:textSize="10sp"></TextView>  
  
 <RadioGroup  
 android:id="@+id/radioGroup"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content">  
  
 <RadioButton  
 android:id="@+id/radio\_male"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Male" />  
  
 <RadioButton  
 android:id="@+id/radio\_female"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Female" />  
  
 <RadioButton  
 android:id="@+id/radio\_other"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Other" />  
  
  
 >  
 </RadioGroup>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <Button  
 android:id="@+id/btFinishReg"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:enabled="false"  
 android:text="Finish"  
 android:textAllCaps="false"  
 android:textColor="@color/black"  
 android:backgroundTint="#4CAF50"/>  
  
 <Button  
 android:id="@+id/btCancel"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_weight="1"  
 android:text="Cancel"  
 android:textAllCaps="false"  
 android:backgroundTint="#FF7700"/>  
  
 >  
 </LinearLayout>  
  
</LinearLayout>

activity\_settings.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"  
 tools:context=".SettingsActivity">  
  
  
 <androidx.appcompat.widget.Toolbar  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:title="Change your'e info here:"  
 app:titleTextColor="@color/white"  
 android:background="@color/colorPrimary">  
 </androidx.appcompat.widget.Toolbar>  
  
  
 <ScrollView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp">  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:layout\_width="88dp"  
 android:layout\_height="88dp"  
 android:id="@+id/imgSettingUserPic"  
 android:src="@drawable/default\_user"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="@color/black"  
 />  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsUserNameShow"  
 android:textSize="20sp"  
 android:text="userName"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsUserGenderShow"  
 android:textSize="15sp"  
 android:text="userGender"  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1dp"  
 android:background="#FFFFFF"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingsPassword"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_phonelink\_lock" />  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:textSize="16sp"  
 android:text="Password"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsPassword"  
 android:textSize="15sp"  
 android:text=""  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingsWeightAndHeight"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_architecture" />  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:textSize="16sp"  
 android:text="Weight and Height"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsWeightHeight"  
 android:textSize="15sp"  
 android:text=""  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingBirthDate"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_date\_range" />  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:textSize="16sp"  
 android:text="Birth Date"  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingsGender"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_wc" />  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:textSize="16sp"  
 android:text="Gender"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsGenderEdit"  
 android:textSize="15sp"  
 android:text=""  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingPic"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_account\_circle" />  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="10dp"  
 android:textSize="16sp"  
 android:text="User Picture"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtSettingsPictureEdit"  
 android:textSize="15sp"  
 android:text=""  
 />  
  
 </LinearLayout>  
  
 </LinearLayout>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 android:id="@+id/layoutSettingManagerUsers"  
 android:visibility="gone"  
 >  
  
 <ImageView  
 android:layout\_width="30dp"  
 android:layout\_height="30dp"  
 android:layout\_marginStart="10dp"  
 android:src="@drawable/ic\_baseline\_wrench" />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="25dp"  
 android:textSize="16sp"  
 android:text="option: manage users"  
 android:id="@+id/txtAdminOPTitleSettings"  
 />  
  
 </LinearLayout>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="from"  
 android:textColor="#aabbcc"  
 />  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_horizontal"  
 android:text="FITNOTE"  
 android:textSize="16sp"  
 android:textColor="@color/colorPrimary"  
 />  
  
  
 </LinearLayout>  
  
 </ScrollView>  
  
  
  
</LinearLayout>

activity\_share.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=".ShareActivity"  
 android:orientation="vertical"  
 android:text="shareWithContact">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtShareWithContactScreen"  
 android:layout\_marginTop="35dp"  
 android:text="Choose contact to send message to"  
 android:gravity="center"  
 android:layout\_gravity="center"  
 android:textSize="20sp"  
 android:textColor="#FF8C00"  
 android:textStyle="bold"  
 ></TextView>  
  
 <SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchViewContactList"  
 android:iconifiedByDefault="false"  
 android:queryHint="Contact Name"  
 ></SearchView>  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/recyclerViewContacts"  
 tools:listitem="@layout/item\_contact"  
 android:layout\_marginBottom="35dp"  
 ></androidx.recyclerview.widget.RecyclerView>  
  
  
</LinearLayout>

activity\_statistics.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=".StatisticsActivity"  
 android:orientation="vertical"  
 android:layout\_margin="10dp"  
 android:backgroundTint="@color/colorPrimary">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2"  
 android:layout\_marginBottom="10dp"  
 android:orientation="horizontal">  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:id="@+id/imgUserPicture"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_weight="1"  
 android:src="@drawable/default\_user"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="@color/black"  
 android:layout\_gravity="center"  
 ></de.hdodenhof.circleimageview.CircleImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="GRAPHS"  
 android:layout\_gravity="center"  
 android:textSize="30sp"  
 android:textColor="@color/colorPrimary"  
 android:layout\_marginTop="20dp"  
 android:layout\_weight="1"  
 ></TextView>  
  
 </LinearLayout>  
  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2"  
 android:layout\_marginBottom="10dp"  
 android:orientation="horizontal">  
  
 <Spinner  
 android:id="@+id/exercise\_spinner"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 ></Spinner>  
  
 <Spinner  
 android:id="@+id/year\_spinner"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 ></Spinner>  
  
 </LinearLayout>  
  
  
 <Button  
 android:id="@+id/btnSubmit"  
 android:layout\_width="75dp"  
 android:layout\_height="45dp"  
 android:layout\_gravity="center"  
 android:backgroundTint="#FF9800"  
 android:text="Submit"  
 android:textAllCaps="false"  
 android:textColor="@color/white"  
 android:textSize="10sp"></Button>  
  
 <com.github.mikephil.charting.charts.BarChart  
 android:layout\_width="match\_parent"  
 android:layout\_height="320dp"  
 android:id="@+id/barChart"  
 ></com.github.mikephil.charting.charts.BarChart>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="----Bellow is the meaning of the colors - user's rating----"  
 android:textColor="@color/colorPrimary"  
 android:textStyle="bold"  
 android:textSize="10sp"  
 android:layout\_gravity="center"  
 android:gravity="center"  
 ></TextView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="5"  
 android:layout\_marginBottom="10dp"  
 android:orientation="horizontal">  
  
 <TextView  
 android:layout\_width="50dp"  
 android:layout\_height="60dp"  
 android:text="Hard"  
 android:textSize="8sp"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:background="@android:drawable/editbox\_background"  
 android:backgroundTint="@color/Hard"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="50dp"  
 android:layout\_height="60dp"  
 android:text="Hard-Medium"  
 android:textSize="8sp"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:background="@android:drawable/editbox\_background"  
 android:backgroundTint="@color/HardMedium"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="50dp"  
 android:layout\_height="60dp"  
 android:text="Medium"  
 android:textSize="8sp"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:background="@android:drawable/editbox\_background"  
 android:backgroundTint="@color/Medium"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="50dp"  
 android:layout\_height="60dp"  
 android:text="Easy-Medium"  
 android:textSize="8sp"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:background="@android:drawable/editbox\_background"  
 android:backgroundTint="@color/EasyMedium"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="50dp"  
 android:layout\_height="60dp"  
 android:text="Easy"  
 android:textSize="8sp"  
 android:layout\_weight="1"  
 android:textStyle="bold"  
 android:gravity="center"  
 android:background="@android:drawable/editbox\_background"  
 android:backgroundTint="@color/Easy"  
 ></TextView>  
  
 </LinearLayout>  
  
</LinearLayout>

activity\_view\_done\_exercises.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=".ViewExercisesResultsActivity"  
 android:orientation="vertical">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="35dp"  
 android:weightSum="2"  
 android:layout\_marginBottom="10dp"  
 android:orientation="horizontal">  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:id="@+id/imgUserPictureExercisesResultsScreen"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_weight="1"  
 android:src="@drawable/default\_user"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="@color/black"  
 android:layout\_gravity="center"  
 ></de.hdodenhof.circleimageview.CircleImageView>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Results"  
 android:layout\_gravity="center"  
 android:textSize="30sp"  
 android:textColor="@color/colorPrimary"  
 android:layout\_marginTop="20dp"  
 android:layout\_weight="1"  
 ></TextView>  
  
 </LinearLayout>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtBMIDoneExercises"  
 android:text="BMI of "  
 android:textSize="20sp"  
 android:textColor="@color/colorPrimary"  
 ></TextView>  
  
 <SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchViewUserExerciseDoneList"  
 android:iconifiedByDefault="false"  
 android:queryHint="Enter Here"  
 ></SearchView>  
  
 <LinearLayout  
 android:id="@+id/layoutOfListView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="300dp"  
 android:orientation="vertical"  
 android:weightSum="3"  
 >  
  
 <ListView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/listViewUserExercisesDone"  
 android:layout\_weight="2"  
 ></ListView>  
  
 <TextView  
 android:id="@+id/txtEmptyListMessage"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Its seems you don't have any exercises done. When you finish you're exercise you will be able to view the results here."  
 android:textSize="20sp"  
 android:layout\_weight="1"  
 android:visibility="gone"  
 ></TextView>  
  
 </LinearLayout>  
  
</LinearLayout>

activity\_whats\_app\_send.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=".WhatsAppSendActivity"  
 android:orientation="vertical">  
  
 <TextView  
 android:text="Name: "  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"  
 android:id="@+id/tvName"  
 android:textColor="@color/colorPrimary"  
 android:textSize="30sp"/>  
  
 <TextView  
 android:text="Phone number: "  
 android:layout\_height="wrap\_content"  
 android:layout\_width="match\_parent"  
 android:id="@+id/tvPhone"  
 android:textColor="@color/colorPrimary"  
 android:textSize="30sp"/>  
  
 <EditText  
 android:id="@+id/etWhatsAppMessage"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Enter Your message here"  
 android:layout\_marginTop="35dp"  
 android:inputType="textMultiLine"  
 android:scrollbars="vertical"  
 android:minLines="3"  
 android:maxLines="10"  
 />  
  
 <Button  
 android:id="@+id/btSendWhatsAppMessage"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="5dp"  
 android:layout\_gravity="center"  
 android:text="SEND"  
 android:textAllCaps="false"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:backgroundTint="#4CAF50"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="35dp"  
 android:layout\_gravity="center"  
 android:text="Your message will be combined with your exercises results from this week"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:gravity="center"  
 ></TextView>  
  
  
  
</LinearLayout>

exercise\_layout.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="75dp"  
 android:layout\_height="75dp"  
 android:src="@drawable/pushup"  
 ></ImageView>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 >  
  
 <TextView  
 android:id="@+id/txtMainTitle"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Main Title"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:layout\_margin="5dp"  
 android:textSize="18sp"  
 ></TextView>  
  
 </LinearLayout>  
  
</LinearLayout>

exercise\_with\_description\_layout.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical"  
 android:padding="16dp">  
  
 <ImageView  
 android:id="@+id/imageViewOfExerciseInDescription"  
 android:layout\_width="75dp"  
 android:layout\_height="75dp"  
 android:src="@drawable/pushup"  
 android:layout\_gravity="center"  
 ></ImageView>  
  
 <TextView  
 android:id="@+id/txtExerciseNameInDescription"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Exercise name"  
 android:textColor="@color/colorPrimary"  
 android:textStyle="bold"  
 android:layout\_margin="5dp"  
 android:textSize="25sp"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtExerciseDescription"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Exercise description"  
 android:textColor="@color/black"  
 android:layout\_margin="5dp"  
 android:textSize="15sp"  
 ></TextView>  
  
  
  
</LinearLayout>

gender\_settings\_alert.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" 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="Chose Gender:"  
 android:textColor="@color/colorPrimary"  
 android:textSize="20sp"></TextView>  
  
 <RadioGroup  
 android:id="@+id/radioGroupSettingsGender"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content">  
  
 <RadioButton  
 android:id="@+id/radio\_maleSettingsGender"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:checked="true"  
 android:text="Male" />  
  
 <RadioButton  
 android:id="@+id/radio\_femaleSettingsGender"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Female" />  
  
 <RadioButton  
 android:id="@+id/radio\_otherSettingsGender"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Other" />  
  
  
 >  
 </RadioGroup>  
  
</LinearLayout>

item\_contact.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal">  
  
 <ImageView  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:src="@drawable/ic\_baseline\_person"  
 android:id="@+id/iv\_imageContact"  
 ></ImageView>  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginEnd="8dp"  
 android:layout\_marginStart="8dp">  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_nameContact"  
 android:textSize="18sp"  
 android:textColor="@color/colorPrimary"  
 android:paddingTop="8dp"  
 android:paddingBottom="8dp"  
 android:text="name"  
 ></TextView>  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_numberContact"  
 android:textSize="14sp"  
 android:textColor="@color/colorPrimary"  
 android:paddingTop="8dp"  
 android:paddingBottom="8dp"  
 android:text="number"  
 ></TextView>  
  
 </LinearLayout>  
  
</LinearLayout>

profilepic\_settings\_alert.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" 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="Profile pic:"  
 android:layout\_marginTop="10sp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="10sp">  
  
 <ImageView  
 android:id="@+id/imgSettingsEditProfilePic"  
 android:layout\_width="65dp"  
 android:layout\_height="65dp"  
 android:src="@drawable/default\_user"  
 ></ImageView>  
  
 <ImageView  
 android:id="@+id/imgSettingsEditCemraButton"  
 android:layout\_width="35dp"  
 android:layout\_height="35dp"  
 android:layout\_gravity="bottom"  
 android:layout\_marginLeft="30dp"  
 android:src="@drawable/cemra"></ImageView>  
  
  
 >  
 </LinearLayout>  
  
</LinearLayout>

user\_exercise\_done\_layout.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <ImageView  
 android:id="@+id/imageViewOfUserExerciseDone"  
 android:layout\_width="75dp"  
 android:layout\_height="75dp"  
 android:src="@drawable/pushup"  
 android:layout\_gravity="center"  
 ></ImageView>  
  
 <TextView  
 android:id="@+id/txtExerciseOfUserExerciseDone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Exercise name: "  
 android:textColor="@color/colorPrimary"  
 android:textStyle="bold"  
 android:layout\_margin="5dp"  
 android:textSize="25sp"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtDateOfUserExerciseDone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Date: "  
 android:textColor="@color/black"  
 android:layout\_margin="5dp"  
 android:textSize="15sp"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtTimeOfUserExerciseDone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Time: "  
 android:textColor="@color/black"  
 android:layout\_margin="5dp"  
 android:textSize="15sp"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtRatingOfUserExerciseDone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Rating: "  
 android:textColor="@color/black"  
 android:layout\_margin="5dp"  
 android:textSize="15sp"  
 ></TextView>  
  
 <TextView  
 android:id="@+id/txtRepetitionOfUserExerciseDone"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="Repetition: "  
 android:textColor="@color/black"  
 android:layout\_margin="5dp"  
 android:textSize="15sp"  
 ></TextView>  
  
</LinearLayout>

user\_exercise\_layout.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="horizontal"  
 android:padding="16dp"  
 android:weightSum="3">  
  
 <ImageView  
 android:id="@+id/imageView"  
 android:layout\_width="0dp"  
 android:layout\_height="75dp"  
 android:src="@drawable/pushup"  
 android:layout\_weight="1"  
 ></ImageView>  
  
 <TextView  
 android:id="@+id/txtMainTitle"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1.5"  
 android:text="Main Title"  
 android:textColor="@color/black"  
 android:textStyle="bold"  
 android:layout\_margin="5dp"  
 android:textSize="18sp"  
 ></TextView>  
  
 <Button  
 android:id="@+id/btnDelete"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="0.5"  
 android:backgroundTint="#FFFFFF"  
 android:text="DELETE"  
 android:textColor="@color/black"  
 android:layout\_marginHorizontal="10dp"  
 ></Button>  
  
</LinearLayout>

user\_layout.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="300dp"  
 android:layout\_height="match\_parent"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:orientation="horizontal"  
 android:layout\_marginTop="10dp"  
 >  
  
  
 <de.hdodenhof.circleimageview.CircleImageView  
 android:layout\_width="88dp"  
 android:layout\_height="88dp"  
 android:id="@+id/imgUserPicUserLayout"  
 android:src="@drawable/default\_user"  
 app:civ\_border\_width="3dp"  
 app:civ\_border\_color="@color/black"  
 />  
  
 <LinearLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 android:layout\_marginStart="15dp">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtUserNameShowUserLayout"  
 android:textSize="20sp"  
 android:text="userName"  
 />  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/txtUserGenderShowUserLayout"  
 android:textSize="15sp"  
 android:text="userGender"  
 />  
  
 </LinearLayout>  
  
  
</LinearLayout>

weight\_height\_settings\_alert.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<LinearLayout  
 xmlns:android="http://schemas.android.com/apk/res/android" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="Weight (kilograms):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarSettingsWeight"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="40" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarSettingsWeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="200"  
 android:min="40"  
 android:layout\_marginTop="15dp"/>  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:weightSum="2">  
  
 <TextView  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
  
 android:layout\_weight="1"  
 android:text="Height (centimeters):" />  
  
 <TextView  
 android:id="@+id/txtSeekBarSettingsHeight"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:text="50" />  
  
 >  
 </LinearLayout>  
  
 <SeekBar  
 android:id="@+id/seekBarSettingsHeight"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:max="250"  
 android:min="50"  
 android:layout\_marginTop="15dp"/>  
  
</LinearLayout>

## menu

main\_menu.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
  
 <item  
 android:title="about"  
 android:id="@+id/aboutItem"  
 ></item>  
  
 <item  
 android:title="exit"  
 android:id="@+id/exitItem"  
 ></item>  
  
  
</menu>

program\_user\_menu.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<menu xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <item  
 android:title="graphs"  
 android:id="@+id/graphsItem"  
 android:icon="@drawable/ic\_baseline\_show\_chart"  
 app:showAsAction="always"  
 android:iconTint="@color/black"  
 ></item>  
  
 <item  
 android:title="results"  
 android:id="@+id/exercisesResultsItem"  
 android:icon="@drawable/ic\_results"  
 app:showAsAction="always"  
 android:iconTint="@color/black"  
 ></item>  
  
 <item  
 android:title="share"  
 android:id="@+id/shareItem"  
 android:icon="@drawable/ic\_share"  
 app:showAsAction="always"  
 android:iconTint="@color/black"  
 ></item>  
  
 <item  
 android:title="settings"  
 android:id="@+id/settingsItem"  
 android:icon="@drawable/ic\_settings"  
 android:iconTint="@color/black"  
 app:showAsAction="always"  
 ></item>  
  
</menu>

## mipmap

## ic\_launcer file

ic\_launcer.png



ic\_launcer.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">  
 <background android:drawable="@drawable/ic\_launcher\_background" />  
 <foreground android:drawable="@drawable/ic\_launcher\_foreground" />  
</adaptive-icon>

## ic\_launcer\_round file

ic\_launcer\_round.png



ic\_launcer\_round.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">  
 <background android:drawable="@drawable/ic\_launcher\_background" />  
 <foreground android:drawable="@drawable/ic\_launcher\_foreground" />  
</adaptive-icon>

## raw

phantom.mp3

shoping.mp3

speedrun.mp3

speedrun2.mp3

speedrun3.mp3

## values

colors.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<resources>  
 <color name="purple\_200">#FFBB86FC</color>  
 <color name="purple\_500">#FF6200EE</color>  
 <color name="purple\_700">#FF3700B3</color>  
 <color name="teal\_200">#FF03DAC5</color>  
 <color name="teal\_700">#FF018786</color>  
 <color name="black">#FF000000</color>  
 <color name="white">#FFFFFFFF</color>  
 <color name="colorPrimary">#009688</color>  
 <color name="Hard">#F44336</color>  
 <color name="HardMedium">#FF9800</color>  
 <color name="Medium">#FFEB3B</color>  
 <color name="EasyMedium">#CDDC39</color>  
 <color name="Easy">#4CAF50</color>  
</resources>

dimens.xml

<resources>  
 <dimen name="fab\_margin">16dp</dimen>  
</resources>

strings.xml

<resources>  
 <string name="app\_name">FitNote</string>  
 <string name="title\_activity\_settings">Settings</string>  
  
 *<!-- Preference Titles -->* <string name="messages\_header">Messages</string>  
 <string name="sync\_header">Sync</string>  
  
 *<!-- Messages Preferences -->* <string name="signature\_title">Your signature</string>  
 <string name="reply\_title">Default reply action</string>  
  
 *<!-- Sync Preferences -->* <string name="sync\_title">Sync email periodically</string>  
 <string name="attachment\_title">Download incoming attachments</string>  
 <string name="attachment\_summary\_on">Automatically download attachments for incoming emails  
 </string>  
 <string name="attachment\_summary\_off">Only download attachments when manually requested</string>  
 <string name="title\_activity\_information">InformationActivity</string>  
  
  
  
</resources>

themes.xml

<resources xmlns:tools="http://schemas.android.com/tools">  
 *<!-- Base application theme. -->* <style name="Theme.FitNote13022021" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 *<!-- Primary brand color. -->* <item name="colorPrimary">@color/colorPrimary</item>  
 <item name="colorPrimaryVariant">@color/colorPrimary</item>  
 <item name="colorOnPrimary">@color/white</item>  
 *<!-- Secondary brand color. -->* <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_700</item>  
 <item name="colorOnSecondary">@color/black</item>  
 *<!-- Status bar color. -->* <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>  
 *<!-- Customize your theme here. -->* </style>  
  
 <style name="Theme.FitNote13022021.NoActionBar">  
 <item name="windowActionBar">false</item>  
 <item name="windowNoTitle">true</item>  
 </style>  
  
 <style name="Theme.FitNote13022021.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />  
  
 <style name="Theme.FitNote13022021.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />  
</resources>

night\themes.xml

<resources xmlns:tools="http://schemas.android.com/tools">  
 *<!-- Base application theme. -->* <style name="Theme.FitNote13022021" parent="Theme.MaterialComponents.DayNight.DarkActionBar">  
 *<!-- Primary brand color. -->* <item name="colorPrimary">@color/purple\_200</item>  
 <item name="colorPrimaryVariant">@color/purple\_700</item>  
 <item name="colorOnPrimary">@color/black</item>  
 *<!-- Secondary brand color. -->* <item name="colorSecondary">@color/teal\_200</item>  
 <item name="colorSecondaryVariant">@color/teal\_200</item>  
 <item name="colorOnSecondary">@color/black</item>  
 *<!-- Status bar color. -->* <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>  
 *<!-- Customize your theme here. -->* </style>  
</resources>

## Grable Scripts

project build

*// Top-level build file where you can add configuration options common to all sub-projects/modules.*buildscript **{** repositories **{** google()  
 jcenter()  
 **}** dependencies **{** classpath 'com.android.tools.build:gradle:4.1.3'  
  
 *// NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files* **}  
}**allprojects **{** repositories **{** google()  
 jcenter()  
 maven **{** url 'https://jitpack.io' **}  
 }  
}**task clean(type: Delete) **{** delete rootProject.buildDir  
**}**

module build

plugins **{** id 'com.android.application'  
**}**android **{** compileSdkVersion 30  
 buildToolsVersion "30.0.2"  
  
 defaultConfig **{** applicationId "com.example.fitnote13022021"  
 minSdkVersion 19  
 targetSdkVersion 30  
 versionCode 1  
 versionName "1.0"  
  
 testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** release **{** minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 **}  
 }** compileOptions **{** sourceCompatibility JavaVersion.*VERSION\_1\_8* targetCompatibility JavaVersion.*VERSION\_1\_8* **}  
}**dependencies **{** *// Dependency for circular ImageView* implementation 'de.hdodenhof:circleimageview:3.1.0'  
  
 implementation 'androidx.appcompat:appcompat:1.2.0'  
 implementation 'com.google.android.material:material:1.3.0'  
 implementation 'com.google.android.material:material:1.2.1'  
 implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
 implementation 'com.github.PhilJay:MPAndroidChart-Realm:v3.0.3@aar'  
 implementation 'com.github.PhilJay:MPAndroidChart:v3.0.3'  
 implementation 'androidx.preference:preference:1.1.1'  
 implementation 'androidx.navigation:navigation-fragment:2.3.4'  
 implementation 'androidx.navigation:navigation-ui:2.3.4'  
 testImplementation 'junit:junit:4.+'  
 androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
 implementation 'com.github.PhilJay:MPAndroidChart:v3.1.0'  
  
 androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
  
  
**}**

grable-wrapper.properties

*#Sat Feb 13 09:48:45 IST 2021*distributionBase=GRADLE\_USER\_HOME  
distributionPath=wrapper/dists  
zipStoreBase=GRADLE\_USER\_HOME  
zipStorePath=wrapper/dists  
distributionUrl=https\://services.gradle.org/distributions/gradle-6.5-bin.zip

progaurd-rules.pro

*# Add project specific ProGuard rules here.  
# You can control the set of applied configuration files using the  
# proguardFiles setting in build.gradle.  
#  
# For more details, see  
# http://developer.android.com/guide/developing/tools/proguard.html  
  
# If your project uses WebView with JS, uncomment the following  
# and specify the fully qualified class name to the JavaScript interface  
# class:  
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {  
# public \*;  
#}  
  
# Uncomment this to preserve the line number information for  
# debugging stack traces.  
#-keepattributes SourceFile,LineNumberTable  
  
# If you keep the line number information, uncomment this to  
# hide the original source file name.  
#-renamesourcefileattribute SourceFile*

gradle.properties

*# Project-wide Gradle settings.  
# IDE (e.g. Android Studio) users:  
# Gradle settings configured through the IDE \*will override\*  
# any settings specified in this file.  
# For more details on how to configure your build environment visit  
# http://www.gradle.org/docs/current/userguide/build\_environment.html  
# Specifies the JVM arguments used for the daemon process.  
# The setting is particularly useful for tweaking memory settings.*org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8  
*# When configured, Gradle will run in incubating parallel mode.  
# This option should only be used with decoupled projects. More details, visit  
# http://www.gradle.org/docs/current/userguide/multi\_project\_builds.html#sec:decoupled\_projects  
# org.gradle.parallel=true  
# AndroidX package structure to make it clearer which packages are bundled with the  
# Android operating system, and which are packaged with your app"s APK  
# https://developer.android.com/topic/libraries/support-library/androidx-rn*android.useAndroidX=true  
*# Automatically convert third-party libraries to use AndroidX*android.enableJetifier=true

Settings.gradle (FitNote13.02.2021)

include ':app'  
rootProject.name = "FitNote13.02.2021"

local.properties

*## This file must \*NOT\* be checked into Version Control Systems,  
# as it contains information specific to your local configuration.  
#  
# Location of the SDK. This is only used by Gradle.  
# For customization when using a Version Control System, please read the  
# header note.  
#Thu Apr 22 15:29:58 IDT 2021*sdk.dir=C\:\\Users\\Home\\AppData\\Local\\Android\\Sdk