פרויקט גמר הנדסת תוכנה 2019

FOLLOW ME FOR SPARK

מגישים: גיא קופייב, יובל ענתבי.

מגמה: הנדסת תוכנה

מורים: איתמר פלדמן, עמליה אלמוג.

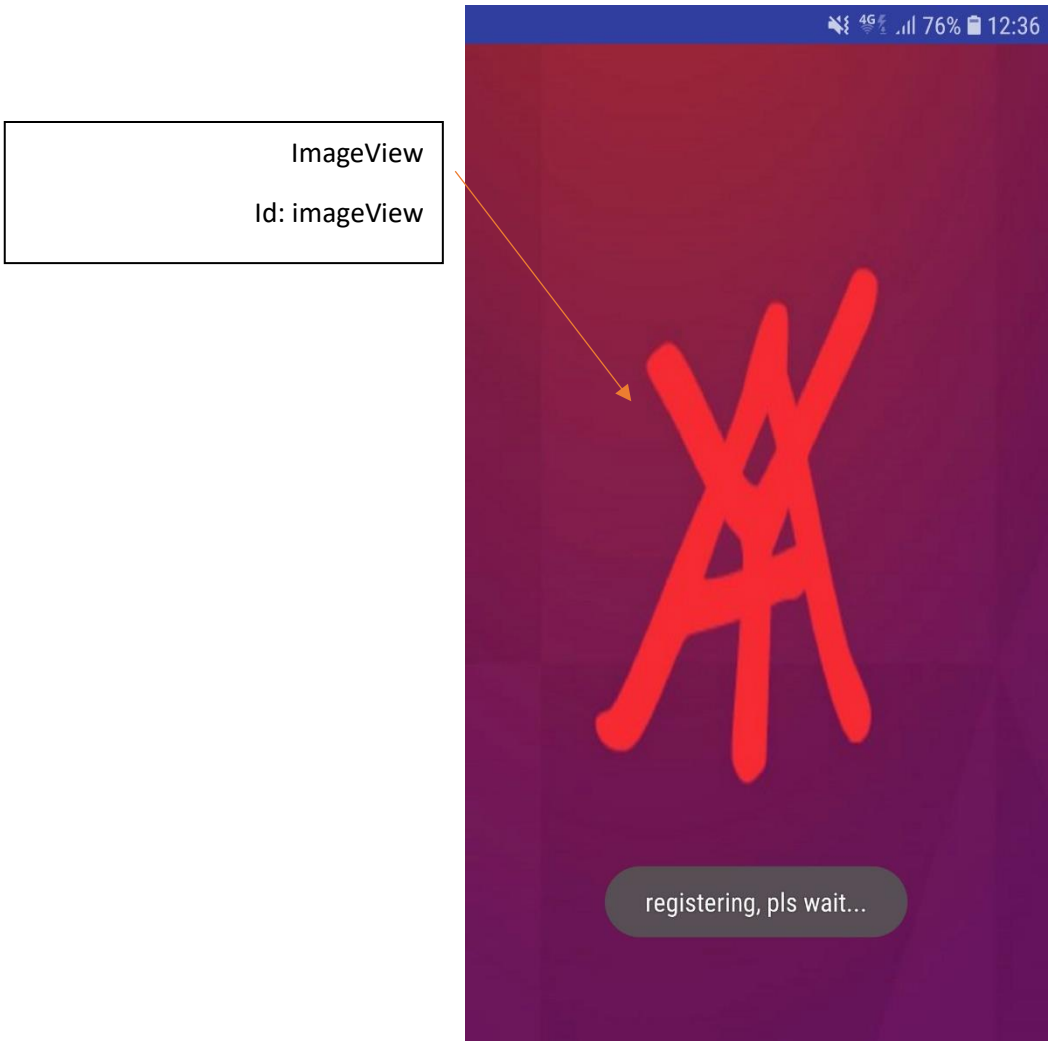
תוכן עניינים

מסך ראשון:

Splash

מסך הספלאש מטרתו היא לדאוג להתחברות ל- sdk ל-dji על מנת שהמשתמש יוכל להטיס את הרחפן מהטלפון האישי שלו.

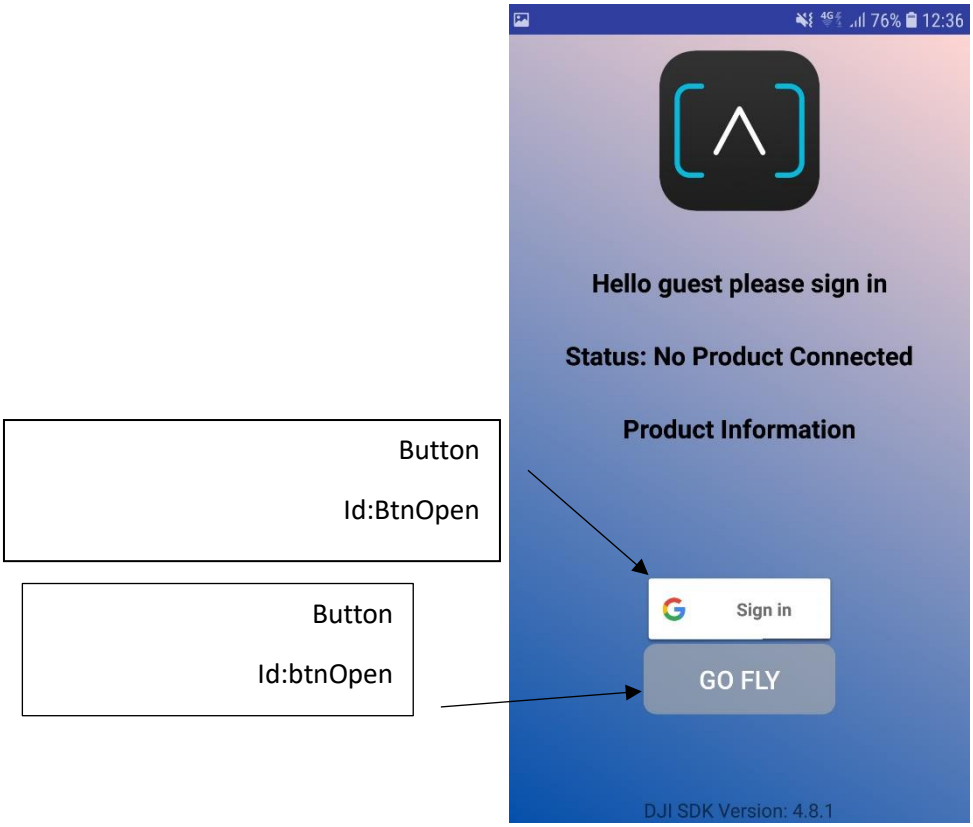
בנוסף המסך הנ"ל בודק אם המשתמש התחבר בעבר ואם כן מחבר אותו אוטומטית.



מסך שני:

Connection activity:

זהו מסך ההתחברות ומטרתו לחבר את המשתמש עם חשבון הגוגל שלו לאפליקציה, ולשמור את פרטיו האישיים. מסך זה בודק אם המשתמש מחובר לרחפן. ההתחברות הינה פרט הכרחי על מנת להטיס את הרחפן.



מסך שלישי: **מסך שמירה:** זהו מסך השמירה. בו המשתמש בוחר איזה תמונות הוא רוצה להעלות לענן(firebase).

TextView

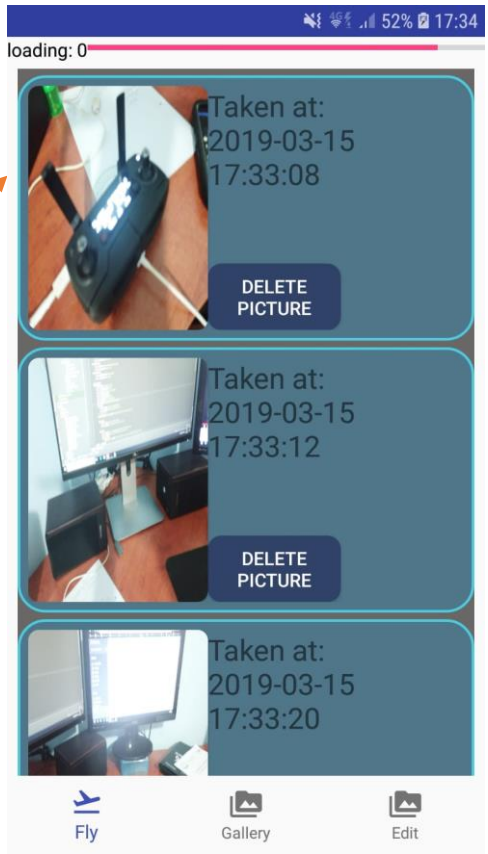
Id:tvLoad

Progresbar

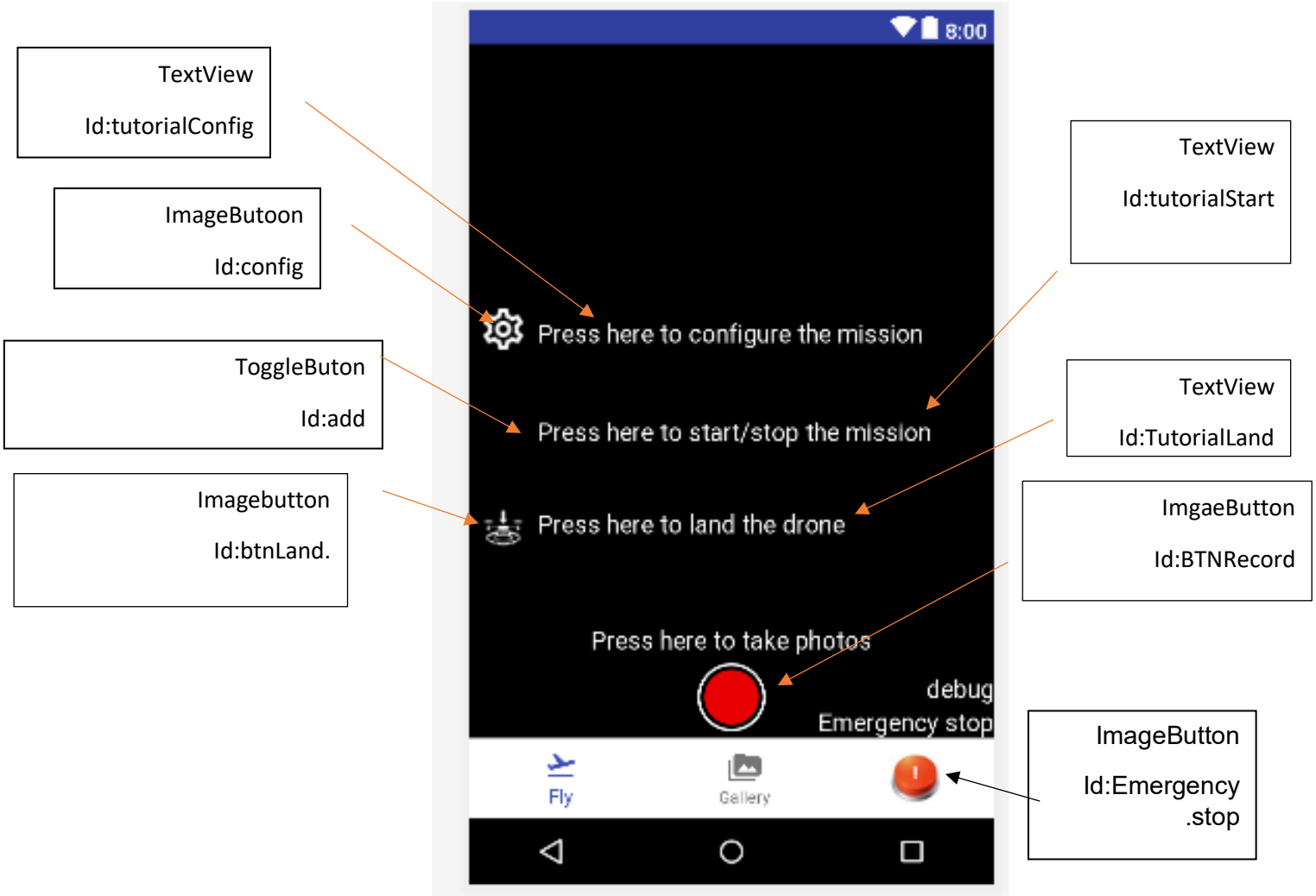
Id:progressBar

recyclerView

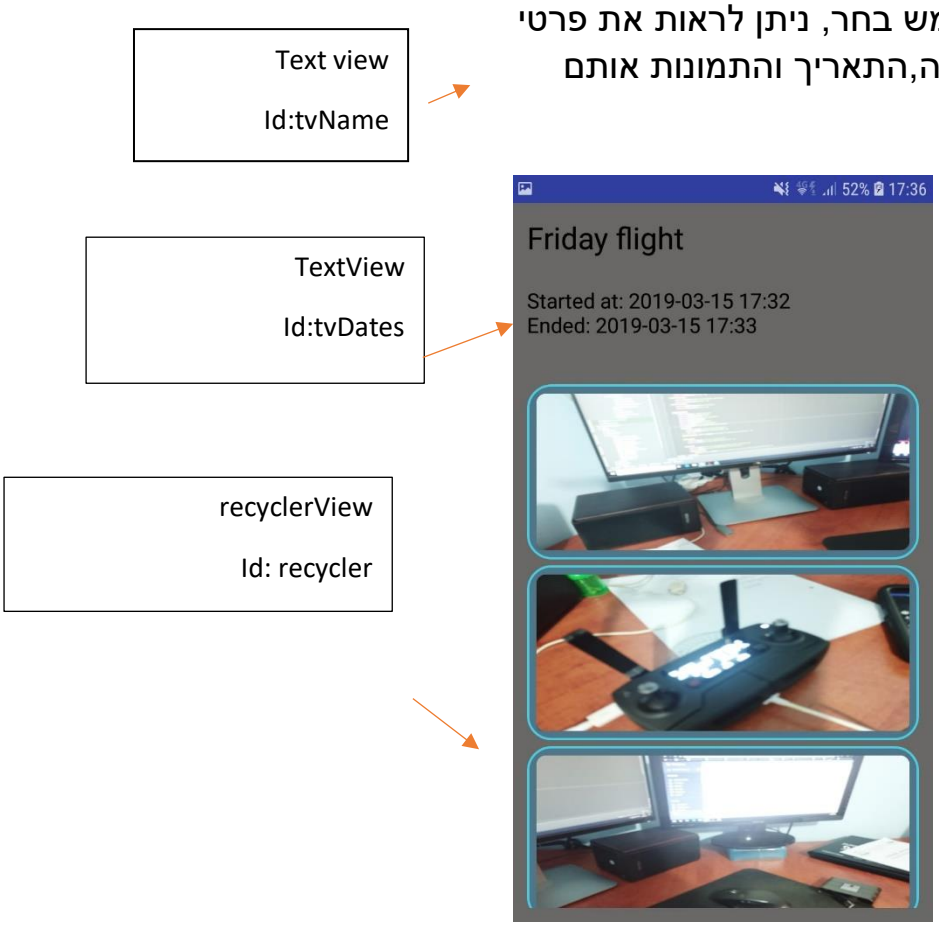
Id: recycler



מסך רביעי: זהו מסך ההטסה, על מנת להתחיל טיסה צריך פשוט ללחוץ על הכפתור והרחפן מיד מתחיל לעקוב אחרי המשתמש(טלפון), במסך זה ניתן גם לצלם תמונות, ובנוסף ניתן להזיז את המצלמה של הרחפן בעזרת הזזת הציר של הטלפון מעלה ומטה. בסיום ההטסה פשוט לוחצים על כפתור הסיום והרחפן נוחת בצורה רכה ובטוחה. לצורכי בטיחות הוספנו גם כפתור חירום(עצירת חירום) בלחיצה על הכפתור הרחפן עוצר באוויר ומפסיק לנוע.



מסך חמישי: זהו מסך שמציג את הפרטים של טיסה מסוימת אותה המשתמש בחר, ניתן לראות את פרטי הטיסה כמו: שם הטיסה,התאריך והתמונות אותם המשתמש בחר לצלם.



מבוא: כחלק מתוכנית הלימודים של חמש יחל הנדסת תוכנה נדרשנו לעשות אפליקציה. ניתנה לנו האפשרות לבחור כל רעיון, עקב כך בחרנו להשתתף בפרויקט הרחפנים . בפרויקט זה נדרשנו לעשות אפליקציה שבה הרחפן הוא החומרה העיקרית. הפרויקט GS-Demo גורם לרחפן לעקוב אחרי המשתמש תוך כדי תנוע ולצלם אותו תוך כדי. אנו מאוד אוהבים לעשות ספורט אקסטרים כגון:רכיבה על אופניים וגלישת גלים, באמצעות הרחפן אנו יכולים לעסוק בפעילות ולצלם את עצמנו מבלי לעצור את הפעילות כך שהתמונות אותם אנחנו מצלמים יוצאות מאוד אוטנטיות ובאיכות מאוד גבוהה.

נושא הפרויקט: מטרת הפרויקט היא לגרום לרחפן לעקוב אחרי המשתמש ולצלם אותו בזמן שהוא נע. הרחפן עושה זאת בעזרת חישובים מתמטיים ובעזרת GPS. בעזרת לחיצה על המסך הרחפן מצלם את המשתמש ומעלה את התמונה ל- FireBase. בנוסף ניתן לראות את הפרטים של כל טיסה כגון:תאריך,שם והתמונות. האפליקציה מיועדת לאנשים שעוסקים בספורט אקסטרים ומעוניינים לצלם את עצמם תוך כדי תנוע.

הפעולות אשר ניתנות לביצוע בעזרת האפליקציה הן:

1. צילום תמונה.
2. עצירת חירום.
3. הנחתת הרחפן.
4. העלאת התמונה לענן.

כלים:

1. רחפן DJI Spark.
2. תוכנת Android Studio.
3. סמארטפון.

נתונים: תקיית ה- AndroidManifest : אחד מקבצי המפתח

באפליקציה, המניפסט הוא מקבץ של רשומות ההרשאות אשר ניתנות

לאובייקטים שונים באפליקציה (כגון Receiver Broadcast,Service ועוד)

droidManifest.xml

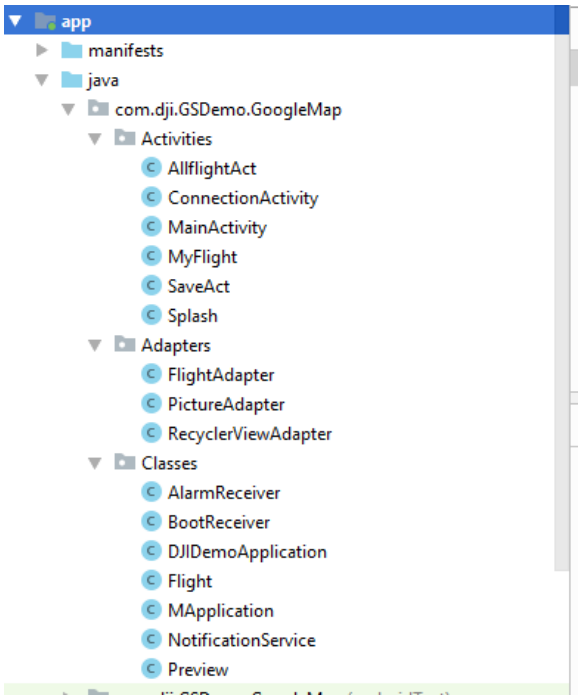
```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_CONFIGURATION" />
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
```

בנוסף לכך אחראי המניפסט על רישום כל המסכים הקיימים

האפליקציה במקום מסודר.

תקיות JAVA :

AllfighAct- מסך שמראה את כל היסטורית הטיסה שנשמרו ב FB .



ConnectionActivity- מחבר את המשתמש לאפליקציה בעזרת חשבון google. החשבון ישמש את המשתמש להעלת התמונות ל - FB.

MainActivity: זהו מסך הטיסה, דרך מסך זה המשתמש יכול לצלם תמונות מהרחפן, יכול להנחית את הרחפן ולראות דרך הטלפון את מה שהמצלמה של הרחפן רואה.

My|Fligh- מסך שמראה טיסה מסוימת שהמשתמש בוחר ובה הוא רואה את הפרטים שנשמרו וגם את התמונות אותם הוא צילם.

SaveAct: שומר את הטיסה ומעלה אותה ל-firestore. נפתחת כאשר המשתמש בוחר לסיים את הטיסה ולשמור את הטיסה.

Splash: זהו המסך הראשון של האפליקציה, בודק אם המשתמש מחובר לחשבון ה- google שלו. אם כן אז הוא לא דורש ממנו להתחבר לחשבון שלו שוב פעם.

FlightAdapter: נמצא בתוך מסך MyFlight: מתאם את המסך להסטוריית הטיסה.

PictureAdapter: מתאם את התמונות שצולמו במהלך הטיסה למסך MyFlight.

RecyclerViewAdapter: מראה את הטיסה האחרונה מתואם למסך MyFlight מראה את התאריך, תמונה. המשתמש יכול למחוק את התמונה ובכך התמונה נמחקת מהטלפון שלו.

DJIDemoAplication: זוהי מחלקה של של חברת DJI מחברת את ה-API.

Flight: מחלקת הטיסה שמתואמת למסך הטיסה ובעזרתה המשתמש שומר את כל הפרטים.

Preview:מראה את התמונה שהרחפן צילם במוסף אחראי גם להראות את הנתונים של התמונה כמו התאריך.

תקיית ה-res :

Drawable: מראה את קבצי התמונות באפליקציה.

Layout:

Allflights: מסך בו רואים את כל הטיסות שנשמרו.

Connection: מסך בו המשתמש מתחבר לחשבון הגוגל שלו.

Main: זהו מסך הטיסה, המשתמש יכול לצלם תמונות, לנחות, לעצור את הרחפן. ולראות דרך הטלפון את מה שהרחפן רואה.

My_Flight: מראה את הטיסה שנבחרה ובה יש את התמונות שהמשתמש צילם ואת הנתונים של הטיסה.

Save: מסך בו המשתמש שומר את הטיסה.

Splash: זהו המסך הראשון בודק אם המשתמש מחובר לחשבון הגוגל שלו.

dialogSetting: זהו גיאלוג שבעזרתו אפשר להגדיר את הטיסה.

EditDialog: בוחרים את השם של הטיסה שנשמרה, ובה גם מצויין התאריך והמייל.

Flight_Row: מתאם למסך allflight.

PictureRow: מתאם למסך allflight.

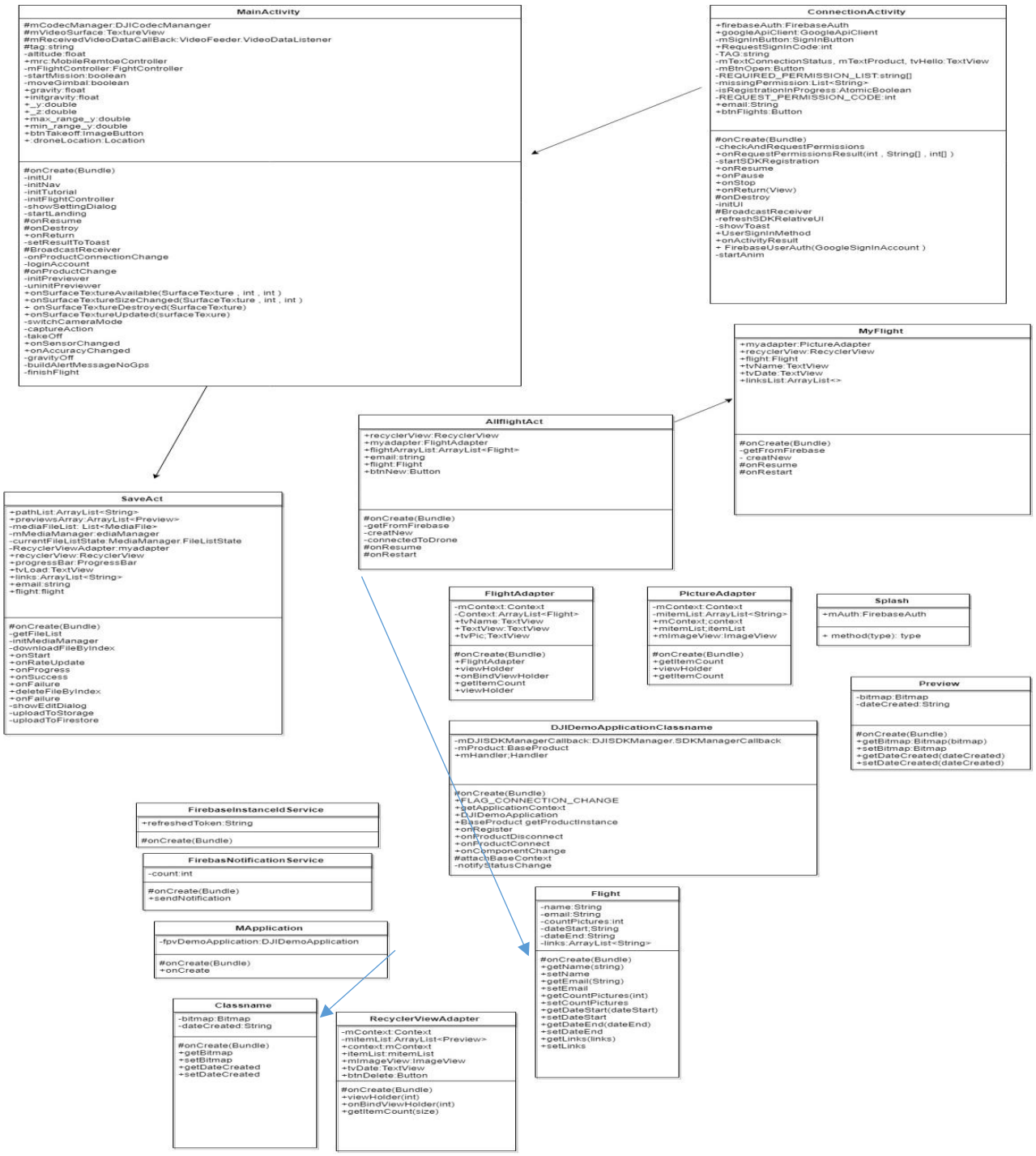
Rowitem: מתאם למסך allflight.

Menu:

Galley: התפריט לגלריה

Navigation: התפריט לניווט.

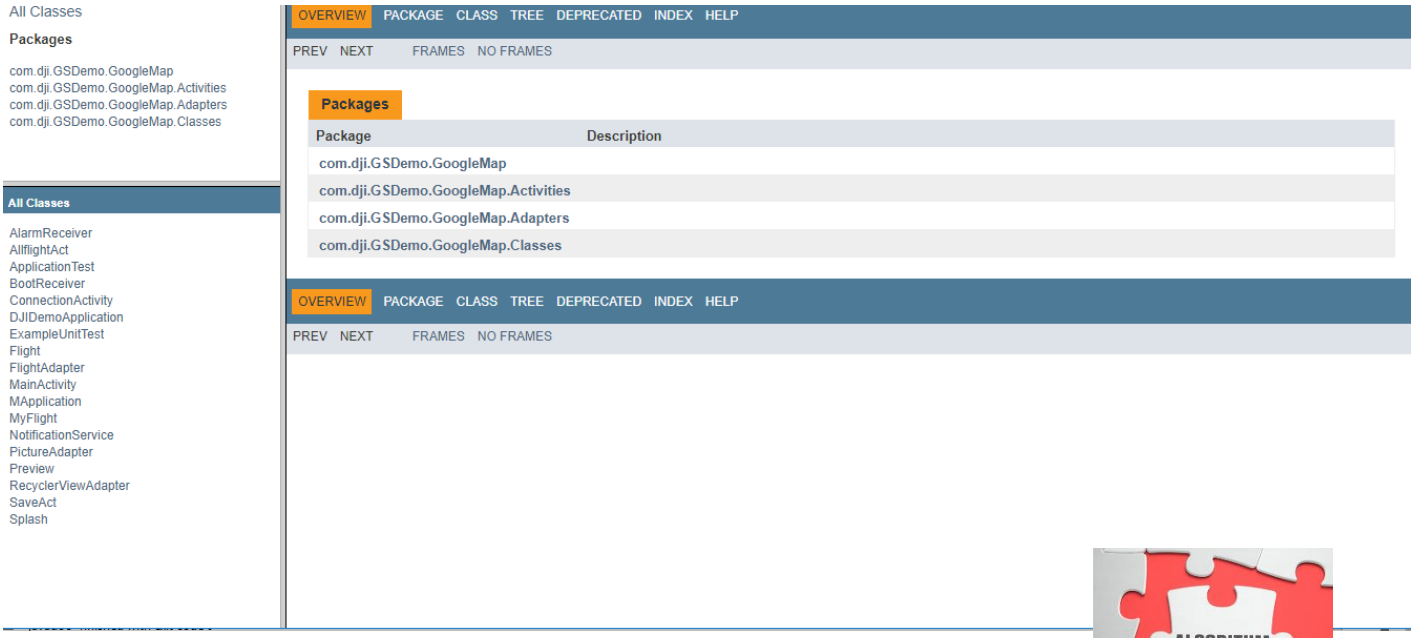
Uml


MainActivity
#mCodecManager:DJICodecMananger
#mVideoSurface:TextureView
#mReceivedVideoDataCallBack:VideoFeeder.VideoDataListener
#tag:string
-altitude:float
+mrc:MobileRemtoeController
-mFlightController:FlightController
-startMission:boolean
-moveGimbal:boolean
+gravity:float
+initgravity:float
+_y:double
+_z:double
+max_range_y:double
+min_range_y:double
+btnTakeoff:ImageButton
+ droneLocation:Location
#onCreate(Bundle)
-initUI
-initNav
-initTutorial
-initFlightController
-showSettingDialog
-startLanding
#onResume
#onDestroy
+onReturn
-setResultToToast
#BroadcastReceiver
-onProductConnectionChange
-loginAccount
#onProductChange
-initPreviewer
-uninitPreviewer
+onSurfaceTextureAvailable(SurfaceTexture , int , int)
+onSurfaceTextureSizeChanged(SurfaceTexture , int , int)
+ onSurfaceTextureDestroyed(SurfaceTexture)
+onSurfaceTextureUpdated(surfaceTexure)
-switchCameraMode
-captureAction
-takeOff
+onSensorChanged
+onAccuracyChanged
-gravityOff
-buildAlertMessageNoGps
-finishFlight
ConnectionActivity
+firebaseAuth:FirebaseAuth
+googleApiClient:GoogleApiClient
-mSignInButton:SignInButton
+RequestSignInCode:int
-TAG:string
-mTextConnectionStatus, mTextProduct, tvHello:TextView
-mBtnOpen:Button
-REQUIRED_PERMISSION_LIST:string[]
-missingPermission:List<String>
-isRegistrationInProgress:AtomicBoolean
-REQUEST_PERMISSION_CODE:int
+email:String
+btnFlights:Button
#onCreate(Bundle)
-checkAndRequestPermissions
+onRequestPermissionsResult(int , String[] , int[])
-startSDKRegistration
+onResume
+onPause
+onStop
+onReturn(View)
#onDestroy
-initUI
#BroadcastReceiver
-refreshSDKRelativeUI
-showToast
+UserSignInMethod
+onActivityResult
+ FirebaseUserAuth(GoogleSignInAccount)
-startAnim
MyFlight
+myadapter:PictureAdapter
+recyclerView:RecyclerView
+flight:Flight
+tvName:TextView
+tvDate:TextView
+linksList:ArrayList<>
#onCreate(Bundle)
-getFromFirebase
- creatNew
#onResume
#onRestart
AllflightAct
+recyclerView:RecyclerView
+myadapter:FlightAdapter
+flightArrayList:ArrayList<Flight>
+email:string
+flight:Flight
+btnNew:Button
#onCreate(Bundle)
-getFromFirebase
-creatNew
-connectedToDrone
#onResume
#onRestart
SaveAct
+pathList:ArrayList<String>
+previewsArray:ArrayList<Preview>
-mediaFileList: List<MediaFile>
-mMediaManager:ediaManager
-currentFileListState:MediaManager.FileListState
-RecyclerViewAdapter:myadapter
+recyclerView:RecyclerView
+progressBar:ProgressBar
+tvLoad:TextView
+links:ArrayList<String>
+email:string
+flight:flight
#onCreate(Bundle)
-getFileList
-initMediaManager
-downloadFileByIndex
+onStart
+onRateUpdate
+onProgress
+onSuccess
+onFailure
+deleteFileByIndex
+onFailure
-showEditDialog
-uploadToStorage
-uploadToFirestore
FlightAdapter
-mContext:Context
-Context:ArrayList<Flight>
+tvName:TextView
+TextView:TextView
+tvPic:TextView
#onCreate(Bundle)
+FlightAdapter
+viewHolder
+onBindViewHolder
+getItemCount
+viewHolder
PictureAdapter
-mContext:Context
-mitemList:ArrayList<String>
+mContext:context
+mitemList:itemList
+mImageView:ImageView
#onCreate(Bundle)
+getItemCount
+viewHolder
+getItemCount
Splash
+mAuth:FirebaseAuth
+ method(type): type
Preview
-bitmap:Bitmap
-dateCreated:String
#onCreate(Bundle)
+getBitmap:Bitmap(bitmap)
+setBitmap:Bitmap
+getDateCreated(dateCreated)
+setDateCreated(dateCreated)
DJIDemoApplicationClassname
-mDJISDKManagerCallback:DJISDKManager.SDKManagerCallback
-mProduct:BaseProduct
+mHandler:Handler
#onCreate(Bundle)
+FLAG_CONNECTION_CHANGE
+getApplicationContext
+DJIDemoApplication
+BaseProduct.getProductInstance
+onRegister
+onProductDisconnect
+onProductConnect
+onComponentChange
#attachBaseContext
-notifyStatusChange
FirebaseInstanceIdService
+refreshedToken:String
#onCreate(Bundle)
FirebasNotificationService
-count:int
#onCreate(Bundle)
+sendNotification
MApplication
-fpvDemoApplication:DJIDemoApplication
#onCreate(Bundle)
+onCreate
Flight
-name:String
-email:String
-countPictures:int
-dateStart:String
-dateEnd:String
-links:ArrayList<String>
#onCreate(Bundle)
+getName(string)
+setName
+getEmail(String)
+setEmail
+getCountPictures(int)
+setCountPictures
+getDateStart(dateStart)
+setDateStart
+getDateEnd(dateEnd)
+setDateEnd
+getLinks(links)
+setLinks
Classname
-bitmap:Bitmap
-dateCreated:String
#onCreate(Bundle)
+getBitmap
+setBitmap
+getDateCreated
+setDateCreated
RecyclerViewAdapter
-mContext:Context
-mitemList:ArrayList<Preview>
+context:mContext
+itemList:mitemList
+mImageView:ImageView
+tvDate:TextView
+btnDelete:Button
#onCreate(Bundle)
+viewHolder(int)
+onBindViewHolder(int)
+getItemCount(size)

אלגורתמיקה: האלגוריתמיקה היא זו שגורמת לרחפן לעקוב אחרי המשתמש בעזרת חישובים מתמטיים. על מנת שהרחפן כל הזמן "יסתכל" על המשתמש כלומר שהמצלמה של הרחפן תמיד תצלם את המשתמש, הרחפן מסתובב סביב צירו וכל הזמן "מתקן" את עצמו ומיישר את עצמו. כדי להפוך את רעיון זה ליעיל אנו כל הזמן בודקים את הזווית בין הרחפן(קו המצלמה) לבין הטלפון(המשתמש) ולפי כך הרחפן יודע לאיזה צד להסתובב כדי שהסיבוב יהיה מהיר ככל האפשר. אנו שולטים גם במהירות הסיבוב של הרחפן בעזרת כמות הכוח אותה אנחנו מעניקים לו. אלגוריתם זה הינו יעיל במיוחד מכיוון שהוא משתמש כמה שפחות במיקום ה-GPS כך שהכל מסובב על חוקי המתמטיקה והפיזיקה ובכך נמנעים באגים רבים. למרות שהרחפן זקוק לקליטת GPS הוא לא מבוסס אך ורק עליו וזוהי הייחודיות של הפרויקט שלנו. נוסף על כך זהו רעיון שאין כמוהו בשוק.

MainActivity.java | MApplication.java | Flight.java | PictureAdapter.java | Splash.java | SaveAct.java

```
604     @SuppressLint("MissingPermission")
605     private void setLocationManager() {
606         LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
607         if (lm != null) {
608             lm.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTime: 2000,    // 2 sec
609                     minDistance: 1.0f, new LocationListener() {
610                         @Override
611                         public void onLocationChanged(Location locationPhone) {
612
613                             if (startMission) {
614                                 if (droneLocation != null) {
615                                     double droneBearing = mFlightController.getCompass().getHeading();
616                                     double newBearing = droneLocation.bearingTo(locationPhone);
617                                     float ls = (float) (newBearing - droneBearing) / 100;
618                                     double dist = locationPhone.distanceTo(droneLocation);
619                                     float fwd = (float) dist / 5;
620                                     mrc.setLeftStickHorizontal(ls);
621                                     if ((ls < 0.1) && (dist < 30))
622                                         mrc.setRightStickVertical(fwd);
623                                     else
624                                         mrc.setRightStickVertical(0f);
625                                 }
626                             }
627                         }
628
```



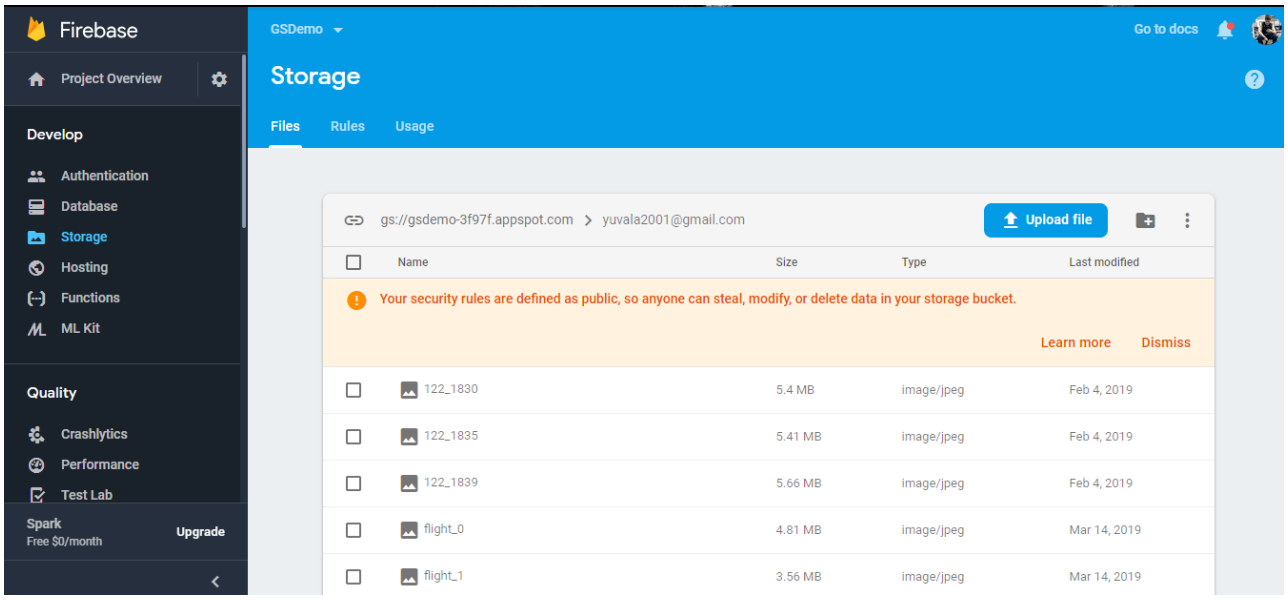
קישור ל JAVADOC:

https://drive.google.com/drive/folders/1USYI1HTge0-oxOo-eMBSTJrJT_71UJ33?usp=sharing

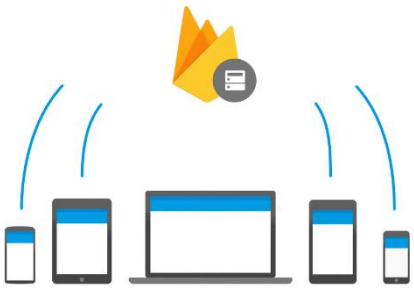

JavaDoc

באפליקציה ניתן לצלם תמונות עם הרחפן לשמור אותם בענן.

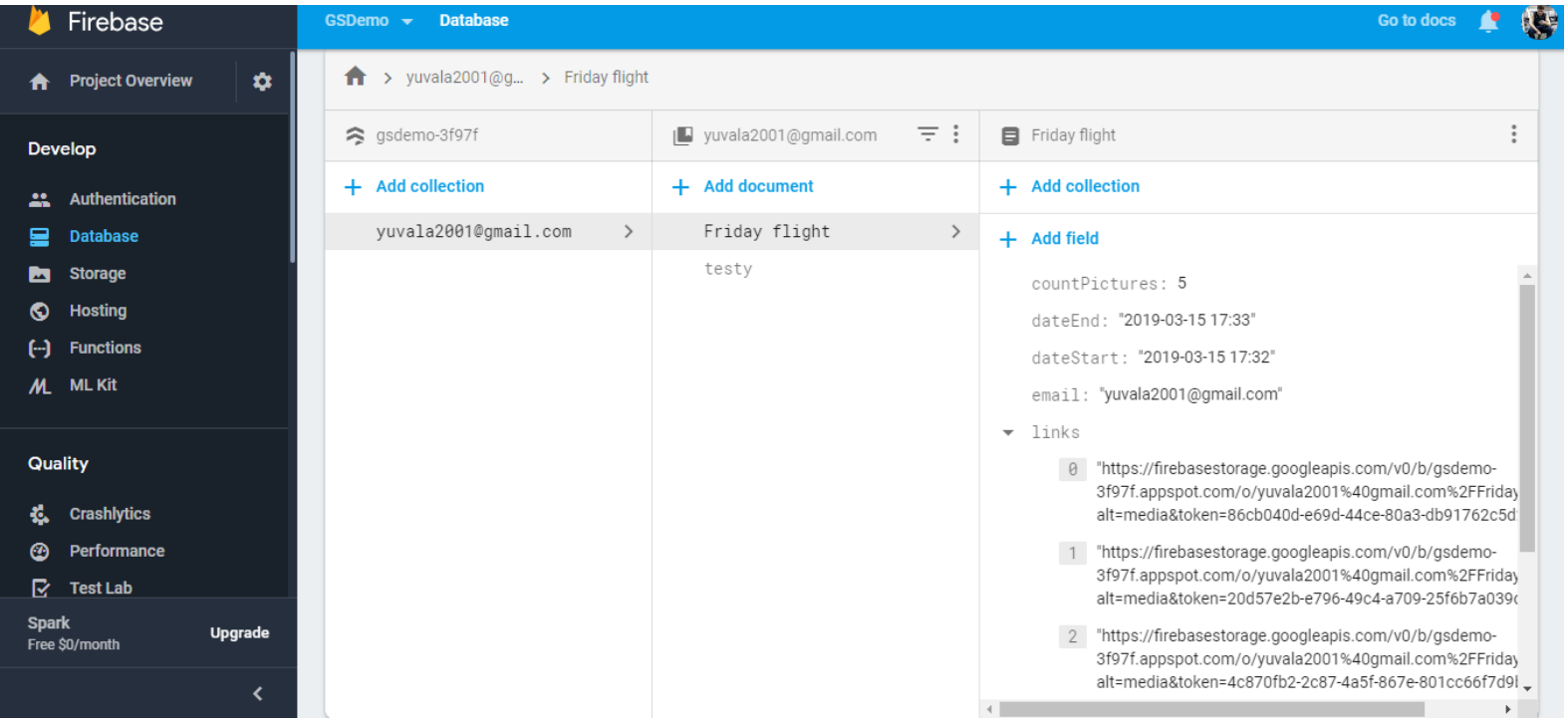
הענן זהו הפיירבייס ובתוך ה-storage ניתן לראות את התמונות



שנשמרו באמצעות לחיצה על התמונה.



באמצעות ה-database נשמרים הקישורים של התמונות. נוסף על כך גם היסטוריית הטיסות נשמרת ובכך ניתן לראות את כל הנתונים הרצויים דרך האינטרנט. הטיסות נשמרות בענן כמו התמונות.



<u>שימוש באבני יסוד:</u>

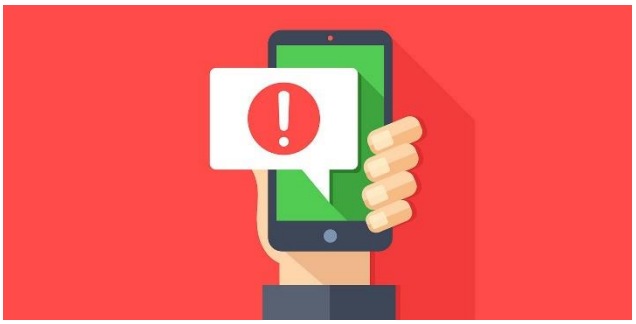
Activity: באפליקציה יש 6 activities

Intent: באפליקציה יש שימוש ב-intent לצורך העברת מידע בין המסכים השונים. לדוגמא: ניתן לראות שה-intent מעביר מהמסך של הטיסה את הנתונים של הטיסה ואת האימייל של המשתמש למסך השמירה.

```
    Intent intent = new Intent( packageContext: MainActivity.this, SaveAct.class);
    intent.putExtra( name: "email", email);
    intent.putExtra( name: "flight", flight);
    startActivity(intent);
}
```

Service: ה-service מעדכן את התאריך שבו יופיע ה-notification.
האפליקציה שולחת הודעה למשתמש שמזמינה אותו לחזור להשתמש באפליקציה.

```
src / main / java / com / dji / GSDemo / GoogleMap / Classes / NotificationService /
t.java × | MainActivity.java × | NotificationService.java ×

@Override
public IBinder onBind(Intent intent) { return null; }

@Override
public void onCreate() {
    super.onCreate();
    startForeground( id: 1, new Notification());
    startAlarm( isNotification: true,  isRepeat: true);
}

@Override
public int onStartCommand(Intent intent, int flags, int startId) { return START_NOT_STICKY; }

private void startAlarm(boolean isNotification, boolean isRepeat) {
    AlarmManager manager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
    Intent myIntent;
    PendingIntent pendingIntent;

    //THIS IS WHERE YOU SET NOTIFICATION TIME FOR CASES WHEN THE NOTIFICATION NEEDS TO BE RESCHEDULED
    Calendar calendar = Calendar.getInstance();
    calendar.set(Calendar.HOUR_OF_DAY, 10);
    calendar.set(Calendar.MINUTE, 00);

    myIntent = new Intent( packageContext: this, AlarmReceiver.class);
    pendingIntent = PendingIntent.getBroadcast( context: this,  requestCode: 0, myIntent,  flags: 0);


    if (!isRepeat)
        manager.set(AlarmManager.RTC_WAKEUP,  triggerAtMillis: SystemClock.elapsedRealtime() + 300, pendingIntent);
    else
        manager.setRepeating(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pendingIntent);
```



BroadCastreciver: ישנם שתי ברודקאסטים, הראשון מפעיל את הservice במקרה שהטלפון נדלק מחדש.

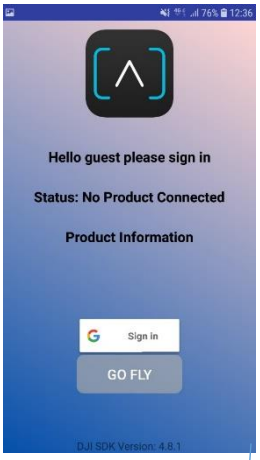
MyFlight.java | MainActivity.java | NotificationService.java | AlarmReceiver.java | BootReceiver.j

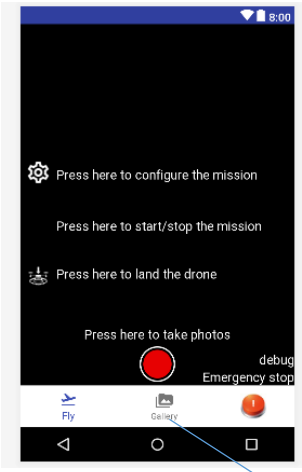
```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.support.v4.content.ContextCompat;
import android.widget.Toast;

public class BootReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {

        //code to execute when Boot Completd

        Intent i = new Intent(context, NotificationService.class);
            ContextCompat.startForegroundService(context,i);

        Toast.makeText(context, text: "Booting Completed", Toast.LENGTH_LONG).show();

    }
}
```

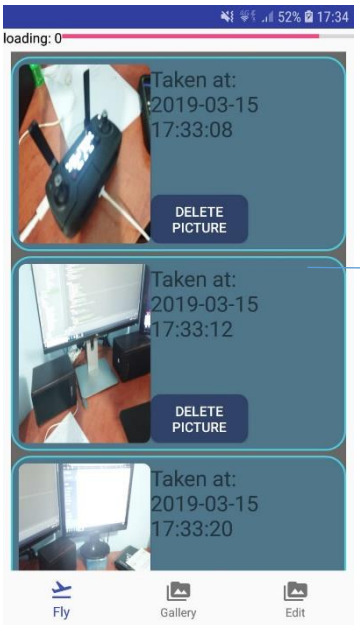
הברודקאסט השני מציג את ה- ההתרעה שמזמינה את המשתמש להשתמש באפליקציה.

```
public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        NotificationCompat.Builder builder;

        NotificationManager notificationManager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

            /* Create or update. */
            NotificationChannel channel = new NotificationChannel( id: "1",  name: "We miss you!",
                    NotificationManager.IMPORTANCE_DEFAULT);
            channel.setDescription("Please fly with us again soon");
            channel.enableLights(true);
            channel.setLightColor(Color.BLUE);
            notificationManager.createNotificationChannel(channel);
            builder = new NotificationCompat.Builder(context, channel.getId());
        } else
            builder = new NotificationCompat.Builder(context);

        Intent myIntent = new Intent(context, ConnectionActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(context,  requestCode: 0, myIntent, FLAG_ONE_SHOT);

        builder.setAutoCancel(true)
                .setDefaults(Notification.DEFAULT_ALL)
                .setWhen(System.currentTimeMillis())
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle("We miss you!")
                .setContentIntent(pendingIntent)
                .setContentText("Please fly with us again soon")
                .setDefaults(Notification.DEFAULT_LIGHTS | Notification.DEFAULT_SOUND)
                .setContentInfo("Info");
```

מדריך למשתמש:







הפעלת ממשק המשתמש:

https://drive.google.com/file/d/1my9vXD7Y1u2jd3aEXUtyCnspwjZbeBWD/view?usp=sharing

להלן קישור להורדה של האפליקציה.

לאחר התקנת האפליקציה צריך לתת את כול ההרשאות הנדרשות.

כאשר המשתמש נכנס לאפליקציה הוא קודם צריך לבדוק שיש לו חיבור אינטרנט תקין וגם קליטת GPS. לאחר מכן עליו להתחבר עם חשבון הגוגל שלו ורק לאחר מכן הוא רשאי להתחיל להטיס. המשתמש בוחר מה הוא רוצה לעשות להתחיל הטסה או לצפות בהסטוריית הטיסה שלו ובתמונות שלו. אם הוא בוחר להתחיל הטסה אז הוא עובר למסך ההטסה ושם הוא בעצם נותן את הפקודה לרחפן להמריא ולהתחיל לעקוב אחריו, בנוסף הוא גם יכול לצלם תמונות בסיום ההטסה המשתמש בוחר איזה מן התמונות הוא רוצה להעלות לענן. אם המשתמש אינו רוצה להטיס את הרחפן אלא לצפות בהסטוריית הטיסה שלו הוא מועבר למסך הטיסות ושם הוא רואה את כל ההטסות שלו מן העבר והוא רשאי לבחור איזה מן ההטסות הוא רוצה לצפות ואיזה תמונות הוא רוצה לראות.

רפלקציה:

מאוד נהננו לפתח את האפליקציה ולהשתתף בפרויקט הרחפנים. בפרויקט זה קיבלנו המון ידע שבו נוכל להשתמש גם בחיינו הבוגרים. את פרויקט הרחפנים עושים בזוג כך שמאוד נהננו לעבוד בצוות וגם העבודה בקבוצות מאוד הלהיבה אותנו. בחרנו לחבור לפרויקט הרחפנים מכיוון שתחום זה מאוד מעניין אותנו, זהו תחום חדשני שמגביר את הקצב מהר מאוד. בנוסף שנינו מאוד אוהבים את תחום האקסטרים, הרחפן משמש אותנו בתחום זה מכיוון שהוא יכול לצלם תמונות מזוויות שונות ומלמעלה ובכך לא להפריע לפעילות עצמה. לסיכום פרויקט הרחפנים הוא פרויקט מאוד מומלץ שמעניק ידע עצום.

תמונות:

אפליקציות מתחרות: הפרויקט שלנו מאוד דומה לפונקציית ה-FollowMe שיש במגוון רחב של רחפנים של חברות שונות אך הייחודיות של האפליקציה שלנו זה האלגוריתם הייחודי שמבוסס בורבו על חישובים מתמטיים לעומת שאר החברות שבהם השימוש ב-GPS הוא החלק המרכזי.

מתחרים:

Yuneec typhoon H מחברת Yuneec international

Walkera Voyager 5 מחברת Walkera

SpellPro Splash 3 Auto מחברת SwellPro

Java : AllflightAct.

```
package com.dji.GSDemo.GoogleMap.Activities;

import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

import com.dji.GSDemo.GoogleMap.Adapters.FlightAdapter;
import com.dji.GSDemo.GoogleMap.Classes.Flight;
import com.dji.GSDemo.GoogleMap.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;

/**
 * Activity to show all previous flights that were saved on firestore, user
can choose a specific flight and check the info or picture
 * that were taken during that flight
 */
public class AllflightAct extends AppCompatActivity {
```

```java
    RecyclerView recyclerView;
    FlightAdapter myadapter;
    ArrayList<Flight> flightArrayList = new ArrayList<>();
    String email = "guest";
    Flight flight;
    Button btnNew;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_allflights);
        recyclerView = findViewById(R.id.recycler);
        btnNew = findViewById(R.id.btnNew);
        if (getIntent().getStringExtra("email") != null &&
!getIntent().getStringExtra("email").equals(""))
            email = getIntent().getStringExtra("email");
        if (email == null || email.equals(""))
            email = FirebaseAuth.getInstance().getCurrentUser().getEmail();

        if (!connectedToDrone())
            btnNew.setVisibility(View.GONE);
        else
            btnNew.setVisibility(View.VISIBLE);
        flight = new Flight();


        getFromFirebase();
        btnNew.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                creatNew();
            }
        });
        recyclerView.setHasFixedSize(true);
        myadapter = new FlightAdapter(AllflightAct.this, flightArrayList);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        recyclerView.setAdapter(myadapter);

    }

    /**
     * Downloads the documents containing the previous flights, sets adapter
for the recycler to show the flights
     */
    private void getFromFirebase() {

FirebaseFirestore.getInstance().collection(email).get().addOnCompleteListene
r(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    for (QueryDocumentSnapshot document : task.getResult())
{
                        Flight flight = document.toObject(Flight.class);
                        flightArrayList.add(flight);
                        recyclerView.setAdapter(myadapter);
                    }
                } else
                    Toast.makeText(AllflightAct.this, "Error getting
documents: " + task.getException(), Toast.LENGTH_SHORT).show();
            }
        });
    }

    /**
     * Creates a new flight instance, adds the start time as the time the
func was called
     */
```

```
    private void creatNew() {
        String month, day, hour, minute;
        flight = new Flight();
        Date currentTime = Calendar.getInstance().getTime();
        int firstDigit = currentTime.getYear() % 100;
        int year = 2000 + firstDigit;
        if ((currentTime.getMonth() + 1) < 10)
            month = "0" + (currentTime.getMonth() + 1);
        else
            month = "" + (currentTime.getMonth() + 1);
        if (currentTime.getDate() < 10)
            day = "0" + currentTime.getDate();
        else
            day = "" + currentTime.getDate();
        if (currentTime.getHours() < 10)
            hour = "0" + currentTime.getHours();
        else
            hour = "" + currentTime.getHours();
        if (currentTime.getMinutes() < 10)
            minute = "0" + currentTime.getMinutes();
        else
            minute = "" + currentTime.getMinutes();
        flight.setDateStart(year + "-" + month + "-" + day + " " + hour +
":" + minute);
        flight.setCountPictures(0);
        flight.setEmail(email);
        Intent intent = new Intent(AllflightAct.this, MainActivity.class);
        intent.putExtra("flight", flight);
        startActivity(intent);
    }

    private boolean connectedToDrone() {
        ConnectivityManager connManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo mWifi =
connManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

        if (mWifi.isConnected()) {
            WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
            WifiInfo info = wifiManager.getConnectionInfo();
            String ssid = info.getSSID();
            return ssid.toLowerCase().contains("spark") ||
ssid.toLowerCase().contains("mavic");
        } else
            return false;
    }

    @Override
    protected void onResume() {
        if (!connectedToDrone())
            btnNew.setVisibility(View.GONE);
        else
            btnNew.setVisibility(View.VISIBLE);
        super.onResume();
    }

    @Override
    protected void onRestart() {
        if (!connectedToDrone())
            btnNew.setVisibility(View.GONE);
        else
            btnNew.setVisibility(View.VISIBLE);
        super.onRestart();
    }
}
```

Java: ConnectionActivity.

```
package com.dji.GSDemo.GoogleMap.Activities;

import android.Manifest;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.drawable.AnimationDrawable;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.os.SystemClock;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.view.Menu;
import android.view.View;
import android.widget.Button;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.dji.GSDemo.GoogleMap.Classes.AlarmReceiver;
import com.dji.GSDemo.GoogleMap.Classes.DJIDemoApplication;
import com.dji.GSDemo.GoogleMap.R;
import com.google.android.gms.auth.api.Auth;
import com.google.android.gms.auth.api.signin.GoogleSignInAccount;
import com.google.android.gms.auth.api.signin.GoogleSignInOptions;
import com.google.android.gms.auth.api.signin.GoogleSignInResult;
import com.google.android.gms.common.ConnectionResult;
```

```java
import com.google.android.gms.common.SignInButton;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.GoogleAuthProvider;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.concurrent.atomic.AtomicBoolean;

import dji.common.error.DJIError;
import dji.common.error.DJISDKError;
import dji.common.useraccount.UserAccountState;
import dji.common.util.CommonCallbacks;
import dji.log.DJILog;
import dji.sdk.base.BaseComponent;
import dji.sdk.base.BaseProduct;
import dji.sdk.products.Aircraft;
import dji.sdk.sdkmanager.DJISDKManager;
import dji.sdk.useraccount.UserAccountManager;

/**
 * Used to connect the user to the app using a google account, the email
will be saved for future uses of the app and uploading of
 * flights to firestore, user can go to Allflight activity to check previous
flights or make a new flight while using the drone
 */
public class ConnectionActivity extends AppCompatActivity {
    public FirebaseAuth firebaseAuth;
    // Google API Client object.
    public GoogleApiClient googleApiClient;
    private SignInButton mSignInButton;
    // Request sing in code. Could be anything as you required.
    public static final int RequestSignInCode = 7;

    private static final String TAG = ConnectionActivity.class.getName();
    private TextView mTextConnectionStatus, mTextProduct, tvHello;
    private Button mBtnOpen;
    private static final String[] REQUIRED_PERMISSION_LIST = new String[]{
            Manifest.permission.VIBRATE,
            Manifest.permission.INTERNET,
            Manifest.permission.ACCESS_WIFI_STATE,
            Manifest.permission.WAKE_LOCK,
            Manifest.permission.ACCESS_COARSE_LOCATION,
            Manifest.permission.ACCESS_NETWORK_STATE,
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.CHANGE_WIFI_STATE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE,
            Manifest.permission.BLUETOOTH,
            Manifest.permission.BLUETOOTH_ADMIN,
            Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.READ_PHONE_STATE,
    };
    private List<String> missingPermission = new ArrayList<>();
    private AtomicBoolean isRegistrationInProgress = new
AtomicBoolean(false);
    private static final int REQUEST_PERMISSION_CODE = 12345;

    public String email = "Guest";

    Button btnFlights;

    @Override
```

```java
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        checkAndRequestPermissions();
        setContentView(R.layout.activity_connection);
        startAnim();
        startAlarm(true, true);
        btnFlights = findViewById(R.id.btnFlights);
        btnFlights.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ConnectionActivity.this,
AllflightAct.class);
                startAlarm(true, true);
                intent.putExtra("email", email);
                startActivity(intent);
            }
        });

        // Getting Firebase Auth Instance into firebaseAuth object.
        firebaseAuth = FirebaseAuth.getInstance();
        // Creating and Configuring Google Sign In object.
        GoogleSignInOptions googleSignInOptions = new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN).requestIdTo
ken("186766913591-
dddchgci6jn5k69taugr9psa2qph5l1r.apps.googleusercontent.com")
                .requestEmail().build();
        // Creating and Configuring Google Api Client.
        googleApiClient = new
GoogleApiClient.Builder(ConnectionActivity.this)
                .enableAutoManage(ConnectionActivity.this, new
GoogleApiClient.OnConnectionFailedListener() {
                    @Override
                    public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {
                    }
                } /* OnConnectionFailedListener
*/).addApi(Auth.GOOGLE_SIGN_IN_API, googleSignInOptions).build();
        initUI();

        if (firebaseAuth != null) {
            FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
            String userDisplayName;

            if (user != null) {
                email = user.getEmail();
                userDisplayName = user.getDisplayName();
                tvHello.setText("Hello " + userDisplayName + " please
connect to drone");
                //    mSignInButton.setVisibility(View.INVISIBLE);
                mBtnOpen.setVisibility(View.VISIBLE);
                btnFlights.setVisibility(View.VISIBLE);
            } else {
                //  mBtnOpen.setVisibility(View.INVISIBLE);
                mSignInButton.setVisibility(View.VISIBLE);
                btnFlights.setVisibility(View.INVISIBLE);
            }

        } else {
            //    mBtnOpen.setVisibility(View.INVISIBLE);
            mSignInButton.setVisibility(View.VISIBLE);
            btnFlights.setVisibility(View.INVISIBLE);
        }
        // Register the broadcast receiver for receiving the device
connection's changes.
        IntentFilter filter = new IntentFilter();
        filter.addAction(DJIDemoApplication.FLAG_CONNECTION_CHANGE);
        registerReceiver(mReceiver, filter);
    }
```

```java
    /**
     * Checks if there is any missing permissions, and
     * requests runtime permission if needed.
     */
    private void checkAndRequestPermissions() {
        // Check for permissions
        for (String eachPermission : REQUIRED_PERMISSION_LIST) {
            if (ContextCompat.checkSelfPermission(this, eachPermission) !=
PackageManager.PERMISSION_GRANTED) {
                missingPermission.add(eachPermission);
            }
        }
        // Request for missing permissions
        if (!missingPermission.isEmpty() && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.M) {
            ActivityCompat.requestPermissions(this,
                    missingPermission.toArray(new
String[missingPermission.size()]),
                    REQUEST_PERMISSION_CODE);
        }

    }

    /**
     * Result of runtime permission request
     */
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        // Check for granted permission and remove from missing list
        if (requestCode == REQUEST_PERMISSION_CODE) {
            for (int i = grantResults.length - 1; i >= 0; i--) {
                if (grantResults[i] == PackageManager.PERMISSION_GRANTED) {
                    missingPermission.remove(permissions[i]);
                }
            }
        }
        // If there is enough permission, we will start the registration
        if (missingPermission.isEmpty()) {
            startSDKRegistration();
        } else {
            showToast("Missing permissions!!!");
        }
    }

    private void startSDKRegistration() {
        if (isRegistrationInProgress.compareAndSet(false, true)) {
            AsyncTask.execute(new Runnable() {
                @Override
                public void run() {
                    showToast("registering, pls wait...");

DJISDKManager.getInstance().registerApp(getApplicationContext(), new
DJISDKManager.SDKManagerCallback() {
                        @Override
                        public void onRegister(DJIError djiError) {
                            if (djiError ==
DJISDKError.REGISTRATION_SUCCESS) {
                                DJILog.e("App registration",
DJISDKError.REGISTRATION_SUCCESS.getDescription());

DJISDKManager.getInstance().startConnectionToProduct();
                                showToast("Register Success");
                            } else
                                showToast("Register sdk fails, check network
```

```
is available");
                            Log.v(TAG, djiError.getDescription());
                        }

                        @Override
                        public void onProductDisconnect() {
                            Log.d(TAG, "onProductDisconnect");
                            showToast("Product Disconnected");
                        }

                        @Override
                        public void onProductConnect(BaseProduct
baseProduct) {
                            Log.d(TAG, String.format("onProductConnect
newProduct:%s", baseProduct));
                            showToast("Product Connected");
                        }

                        @Override
                        public void
onComponentChange(BaseProduct.ComponentKey componentKey, BaseComponent
oldComponent, BaseComponent newComponent) {
                            if (newComponent != null) {
                                newComponent.setComponentListener(new
BaseComponent.ComponentListener() {

                                    @Override
                                    public void onConnectivityChange(boolean
isConnected) {
                                        Log.d(TAG,
"onComponentConnectivityChanged: " + isConnected);
                                    }
                                });
                            }
                            Log.d(TAG, String.format("onComponentChange
key:%s, oldComponent:%s, newComponent:%s", componentKey, oldComponent,
newComponent));
                        }
                    });
                }
            });
        }
    }

    @Override
    public void onResume() {
        Log.e(TAG, "onResume");
        super.onResume();
    }

    @Override
    public void onPause() {
        Log.e(TAG, "onPause");
        super.onPause();
    }

    @Override
    public void onStop() {
        Log.e(TAG, "onStop");
        super.onStop();
    }

    public void onReturn(View view) {
        Log.e(TAG, "onReturn");
        this.finish();
    }

    @Override
```

```
    protected void onDestroy() {
        Log.e(TAG, "onDestroy");
        unregisterReceiver(mReceiver);
        super.onDestroy();
    }

    private void initUI() {
        mSignInButton = (SignInButton) findViewById(R.id.sign_in_button);
        mSignInButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                UserSignInMethod();
            }
        });

        tvHello = findViewById(R.id.tvHello);

        mTextConnectionStatus = (TextView)
findViewById(R.id.text_connection_status);
        mTextProduct = (TextView) findViewById(R.id.text_product_info);
        mBtnOpen = (Button) findViewById(R.id.btn_open);
        mBtnOpen.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(ConnectionActivity.this,
AllflightAct.class);
                intent.putExtra("email", email);
                startActivity(intent);
            }
        });
        mBtnOpen.setEnabled(false);

        TextView mVersionTv = (TextView) findViewById(R.id.textView2);
        mVersionTv.setText(getResources().getString(R.string.sdk_version,
DJISDKManager.getInstance().getSDKVersion()));
    }

    protected BroadcastReceiver mReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            refreshSDKRelativeUI();
        }
    };

    private void refreshSDKRelativeUI() {
        BaseProduct mProduct = DJIDemoApplication.getProductInstance();

        if (null != mProduct && mProduct.isConnected()) {
            mBtnOpen.setEnabled(true);

            String str = mProduct instanceof Aircraft ? "DJIAircraft" :
"DJIHandHeld";
            mTextConnectionStatus.setText("Status: " + str + " connected");

            if (null != mProduct.getModel())
                mTextProduct.setText("" +
mProduct.getModel().getDisplayName());
            else
                mTextProduct.setText(R.string.product_information);

        } else {
            Log.v(TAG, "refreshSDK: False");
            mBtnOpen.setEnabled(false);

            mTextProduct.setText(R.string.product_information);
            mTextConnectionStatus.setText(R.string.connection_loose);
        }
```

```
    }

    private void showToast(final String toastMsg) {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(getApplicationContext(), toastMsg,
Toast.LENGTH_LONG).show();

            }
        });
    }

    /**
     * Creates the dialog for users
     */
    public void UserSignInMethod() {
        // Passing Google Api Client into Intent.
        Intent AuthIntent =
Auth.GoogleSignInApi.getSignInIntent(googleApiClient);
        startActivityForResult(AuthIntent, RequestSignInCode);
    }

    /**
     * Result of the chosen user from phone
     *
     * @param requestCode - code in case you start activity after signing in
     * @param resultCode  - not in use
     * @param data        - google's user
     */
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == RequestSignInCode) {
            GoogleSignInResult googleSignInResult =
Auth.GoogleSignInApi.getSignInResultFromIntent(data);
            if (googleSignInResult.isSuccess()) {
                GoogleSignInAccount googleSignInAccount =
googleSignInResult.getSignInAccount();
                FirebaseUserAuth(googleSignInAccount);
            }

        } else {
            Intent start = new Intent(ConnectionActivity.this,
Splash.class);
            startActivity(start);
            finish();
        }
    }

    /**
     * Uses google auth for signing in with firebase, if successful will get
email and show the name of the user
     *
     * @param googleSignInAccount - Chosen account
     */
    public void FirebaseUserAuth(GoogleSignInAccount googleSignInAccount) {
        AuthCredential authCredential =
GoogleAuthProvider.getCredential(googleSignInAccount.getIdToken(), null);
        firebaseAuth.signInWithCredential(authCredential)
                .addOnCompleteListener(ConnectionActivity.this, new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
AuthResultTask) {
                        if (AuthResultTask.isSuccessful()) {
                            // Getting Current Login user details.
```

```
                            FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();
                            if (user != null) {
                                email = user.getEmail();
                                tvHello.setText("Hello " +
user.getDisplayName() + " please connect to drone");
                            }

                            //  mSignInButton.setVisibility(View.INVISIBLE);
                            btnFlights.setVisibility(View.VISIBLE);
                            mBtnOpen.setVisibility(View.VISIBLE);
                            Toast.makeText(ConnectionActivity.this, "Please
turn on the drone", Toast.LENGTH_SHORT).show();
                        } else {
                            Toast.makeText(ConnectionActivity.this,
"Something Went Wrong", Toast.LENGTH_LONG).show();
                        }
                    }
                });
    }

    /**
     * Starts the color animation in the background
     */
    private void startAnim() {
        RelativeLayout mFirstish = (RelativeLayout)
findViewById(R.id.firstish);
        AnimationDrawable animationDrawable = (AnimationDrawable)
mFirstish.getBackground();
        animationDrawable.setEnterFadeDuration(2000);
        animationDrawable.setExitFadeDuration(4000);
        animationDrawable.start();
    }

    private void startAlarm(boolean isNotification, boolean isRepeat) {
        AlarmManager manager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        Intent myIntent;
        PendingIntent pendingIntent;

        // SET TIME HERE
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, 10);
        calendar.set(Calendar.MINUTE, 00);


        myIntent = new Intent(ConnectionActivity.this, AlarmReceiver.class);
        pendingIntent = PendingIntent.getBroadcast(this, 0, myIntent, 0);


        if (!isRepeat)
            manager.set(AlarmManager.RTC_WAKEUP,
SystemClock.elapsedRealtime() + 300, pendingIntent);
        else
            manager.setRepeating(AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pendingIntent);
    }
}
```

Java:MainActivity.

```
package com.dji.GSDemo.GoogleMap.Activities;

import android.Manifest;
import android.annotation.SuppressLint;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.graphics.SurfaceTexture;
import android.graphics.drawable.Drawable;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Build;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.util.Log;
import android.view.MenuItem;
import android.view.TextureView;
import android.view.TextureView.SurfaceTextureListener;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
```

```java
import android.widget.ImageButton;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

import com.dji.GSDemo.GoogleMap.Classes.DJIDemoApplication;
import com.dji.GSDemo.GoogleMap.Classes.Flight;
import com.dji.GSDemo.GoogleMap.R;

import java.util.Calendar;
import java.util.Date;

import dji.common.camera.SettingsDefinitions;
import dji.common.error.DJIError;
import dji.common.flightcontroller.ControlMode;
import dji.common.flightcontroller.FlightControllerState;
import dji.common.gimbal.Rotation;
import dji.common.gimbal.RotationMode;
import dji.common.product.Model;
import dji.common.util.CommonCallbacks;
import dji.sdk.base.BaseProduct;
import dji.sdk.camera.Camera;
import dji.sdk.camera.VideoFeeder;
import dji.sdk.codec.DJICodecManager;
import dji.sdk.flightcontroller.FlightController;
import dji.sdk.mobilerc.MobileRemoteController;
import dji.sdk.products.Aircraft;
import dji.ux.widget.WiFiSignalWidget;
import dji.ux.widget.dashboard.AltitudeWidget;
import dji.ux.widget.dashboard.CompassWidget;
import dji.ux.widget.dashboard.HorizontalVelocityWidget;
import dji.ux.widget.dashboard.VerticalVelocityWidget;

import static
com.dji.GSDemo.GoogleMap.Classes.DJIDemoApplication.getProductInstance;


/**
 * Activity of flight, will automatically be opened when choosing a new
flight in Allflight Activity
 */
public class MainActivity extends FragmentActivity implements
SurfaceTextureListener, SensorEventListener {
    Flight flight;
    private String email = "Guest";

    // Codec for video live view
    protected DJICodecManager mCodecManager = null;
    protected TextureView mVideoSurface = null;
    protected VideoFeeder.VideoDataListener mReceivedVideoDataCallBack =
null;
    //--
    protected static final String TAG = "GSDemoActivity";

    private float altitude = 2.0f;

    MobileRemoteController mrc;
    private FlightController mFlightController;

    private boolean startMission = false, moveGimbal = false;

    /**
     * the gravity array and the initgravity array will store the real-time
value of the gravity sensor in all 3 axis
     * and the initgravity will store the calibrated gravity sensor position
to be used ad the "stand still" position for the drone
     */
```

```
    float gravity[] = {0, 0, 0};
    float initgravity[] = {0, 0, 0};
    double _y, _z;
    /**
     * the end points of the range of the controller, so the maximum value
for the drone controller will be set to the calibrated
     * "stand-still" point plus the max_range for the relevant axis.
     */
    double max_range_y = 25, max_range_z = 25;
    double min_range_y = 5, min_range_z = 5;

    ImageButton btnTakeoff;
    Location droneLocation = new Location("");

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // When the compile and target version is higher than 22, request
these permissions
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            ActivityCompat.requestPermissions(this,
                    new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
Manifest.permission.VIBRATE,
                            Manifest.permission.INTERNET,
Manifest.permission.ACCESS_WIFI_STATE,
                            Manifest.permission.WAKE_LOCK,
Manifest.permission.ACCESS_COARSE_LOCATION,
                            Manifest.permission.ACCESS_NETWORK_STATE,
Manifest.permission.ACCESS_FINE_LOCATION,
                            Manifest.permission.CHANGE_WIFI_STATE,
Manifest.permission.MOUNT_UNMOUNT_FILESYSTEMS,
                            Manifest.permission.READ_EXTERNAL_STORAGE,
Manifest.permission.SYSTEM_ALERT_WINDOW,
                            Manifest.permission.READ_PHONE_STATE,
                    }
                    , 1);
        }

        setContentView(R.layout.activity_main);
        if (getIntent().getStringExtra("email") != null &&
!getIntent().getStringExtra("email").equals(""))
            email = getIntent().getStringExtra("email");
        flight = (Flight) getIntent().getExtras().getSerializable("flight");

        initUI();
        switchCameraMode();
        IntentFilter filter = new IntentFilter();
        filter.addAction(DJIDemoApplication.FLAG_CONNECTION_CHANGE);
        registerReceiver(mReceiver, filter);
        //    IntentFilter intentFilter= new IntentFilter();
        //     intentFilter.addAction("ACTION_PROVIDER_CHANGED");
        // registerReceiver(gpsLocationReceiver, new
IntentFilter(Intent.ACTION_PROVIDER_CHANGED));

        // The callback for receiving the raw H264 video data for camera
live view
        mReceivedVideoDataCallBack = new VideoFeeder.VideoDataListener() {

            @Override
            public void onReceive(byte[] videoBuffer, int size) {
                if (mCodecManager != null)
                    mCodecManager.sendDataToDecoder(videoBuffer, size);
            }
        };
        SensorManager mSensorManager = (SensorManager)
getSystemService(SENSOR_SERVICE);
        Sensor senRotation =
mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
```

```
        mSensorManager.registerListener(this, senRotation,
SensorManager.SENSOR_DELAY_GAME);

        // init gravity vector
        gravity[0] = 0;
        gravity[1] = 0;
        gravity[2] = 0;
        calibrate_gravity();

        setLocationManager();
    }

    /**
     * initialize the video previewer and sets click listeners
     */
    private void initUI() {
        initTutorial();
        initNav();
        ImageButton btnLand = (ImageButton) findViewById(R.id.btnLand);
        btnLand.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                startLanding();
            }
        });
        btnLand.bringToFront();

        btnTakeoff = (ImageButton) findViewById(R.id.btnTakeoff);
        btnTakeoff.bringToFront();
        btnTakeoff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (buildAlertMessageNoGps()) {
                    if (!startMission) {
                        if (buildAlertMessageNoGps()) {

btnTakeoff.setImageResource(R.drawable.outline_pan_tool_white_18dp);
                            startMission = true;
                            takeOff();
                        }
                    } else {
                        btnTakeoff.setImageResource(R.drawable.takeoff);
                        mrc.setRightStickHorizontal(0);
                        mrc.setLeftStickHorizontal(0);
                        mrc.setLeftStickVertical(0);
                        mrc.setRightStickVertical(0);
                        startMission = false;
                    }
                }
            }
        });

        ImageButton mRecordBtn = (ImageButton) findViewById(R.id.btnRecord);
        mRecordBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                captureAction();
            }
        });
        mRecordBtn.bringToFront();

        // init mVideoSurface
        mVideoSurface = (TextureView)
findViewById(R.id.video_previewer_surface);
        if (null != mVideoSurface) {
            mVideoSurface.setSurfaceTextureListener(this);
            mVideoSurface.setOnClickListener(new View.OnClickListener() {
                @Override
```

```
                public void onClick(View v) {
                    calibrate_gravity();
                    if (!moveGimbal) {
                        moveGimbal = true;
                        Toast.makeText(MainActivity.this, "Move gimbal ON",
Toast.LENGTH_SHORT).show();
                    } else {
                        gravityOff();
                        moveGimbal = false;
                        mrc.setLeftStickHorizontal(0f);
                        Toast.makeText(MainActivity.this, "Move gimbal OFF",
Toast.LENGTH_SHORT).show();
                    }
                }
            });
        }
        //--

        RelativeLayout widgets = findViewById(R.id.relativeDashboard);
        widgets.bringToFront();
    }

    /**
     * connects the bottom menu, set click listener for it
     */
    private void initNav() {
        BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
        navigation.getMenu().findItem(R.id.navigation_fly).setChecked(true);
        navigation.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem
menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.navigation_fly:
                        Toast.makeText(MainActivity.this, "Already in
activity", Toast.LENGTH_SHORT).show();
                        return true;

                    case R.id.navigation_gallery:
                        // if (!email.equals(null) &&
!email.equals("Guest"))
                        finishFlight();
                     /*   else
                            setResultToToast("Cannot edit and upload flight
for guests");*/
                        return true;

                    case R.id.navigation_config:
                        showSettingDialog();
                        return true;

                }
                return false;
            }
        });
    }

    /**
     * set click listener for the got it button, makes all the text views
gone and shows and first person view on click
     */
    private void initTutorial() {
        final TextView tvLand = findViewById(R.id.tutorialLand);
        final TextView tvStart = findViewById(R.id.tutorialStart);
        final TextView tvRecord = findViewById(R.id.tutorialPhoto);
        final Button btnGotit = findViewById(R.id.btnGot);
```

```
        btnGotit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                tvLand.setVisibility(View.GONE);
                tvStart.setVisibility(View.GONE);
                tvRecord.setVisibility(View.GONE);
                btnGotit.setVisibility(View.GONE);
                mVideoSurface.setVisibility(View.VISIBLE);
            }
        });
    }

    private void initFlightController() {
        BaseProduct product = getProductInstance();
        mFlightController = ((Aircraft) product).getFlightController();

        Aircraft aircraft = (Aircraft) getProductInstance();
        mrc = aircraft.getMobileRemoteController();

        mFlightController.setControlMode(ControlMode.SMART, new
CommonCallbacks.CompletionCallback() {
            @Override
            public void onResult(DJIError djiError) {

            }
        });
        mFlightController.setMaxFlightHeight(20, new
CommonCallbacks.CompletionCallback() {
            @Override
            public void onResult(DJIError djiError) {

            }
        });
        mFlightController.setMaxFlightRadiusLimitationEnabled(false, new
CommonCallbacks.CompletionCallback() {
            @Override
            public void onResult(DJIError djiError) {

            }
        });
        mFlightController.setStateCallback(new
FlightControllerState.Callback() {
            @Override
            public void onUpdate(@NonNull FlightControllerState
flightControllerState) {
                //Check GPS level
                double droneLocationLat =
flightControllerState.getAircraftLocation().getLatitude();
                double droneLocationLng =
flightControllerState.getAircraftLocation().getLongitude();
                droneLocation.setLatitude(droneLocationLat);
                droneLocation.setLongitude(droneLocationLng);
                if (startMission) {
                    float height =
flightControllerState.getUltrasonicHeightInMeters();
                    if (height > altitude)
                        mrc.setLeftStickVertical(-0.3f);
                    if (height < altitude)
                        mrc.setLeftStickVertical(0.3f);
                    if (height == altitude)
                        mrc.setLeftStickVertical(0);
                }
            }
        });
        mFlightController.setHomeLocationUsingAircraftCurrentLocation(new
CommonCallbacks.CompletionCallback() {
            @Override
            public void onResult(DJIError djiError) {
```

```
            }
        });

       /* AltitudeWidget altitudeWidget = findViewById(R.id.altitudeWidget);
        altitudeWidget.bringToFront();*/

        CompassWidget compassWidget = findViewById(R.id.compassWidget);
        compassWidget.bringToFront();

        AltitudeWidget altitudeWidget = findViewById(R.id.altitudeWidget);
        altitudeWidget.bringToFront();

        VerticalVelocityWidget verticalVelocityWidget =
findViewById(R.id.verticalVelocityWidget);
        verticalVelocityWidget.bringToFront();

        HorizontalVelocityWidget horizontalVelocityWidget =
findViewById(R.id.horizontalVelocityWidget);
        horizontalVelocityWidget.bringToFront();

        WiFiSignalWidget wiFiSignalWidget = findViewById(R.id.wifiWidget);
        wiFiSignalWidget.bringToFront();
    }

    private void showSettingDialog() {
        final Dialog dialog = new Dialog(MainActivity.this);
        dialog.setContentView(R.layout.dialog_setting);

dialog.getWindow().setBackgroundDrawableResource(android.R.color.transparent
);
        dialog.setCancelable(true);
        final EditText etAltidude = dialog.findViewById(R.id.altitude);

        Button btnCancel = dialog.findViewById(R.id.btnCancel);
        btnCancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dialog.dismiss();
            }
        });

        Button btnFinish = dialog.findViewById(R.id.btnFinish);
        btnFinish.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String heightWanted = etAltidude.getText().toString();
                altitude = Float.parseFloat(heightWanted);
                dialog.dismiss();
            }
        });
        dialog.show();
    }

    /**
     * Lands the drone only if no error occurred when waypoint mission
stopped
     */
    private void startLanding() {
        if (mFlightController.isConnected() && mFlightController != null) {
            startMission = false;
            mrc.setRightStickHorizontal(0);
            mrc.setLeftStickHorizontal(0);
            mrc.setLeftStickVertical(0);
            mrc.setRightStickVertical(0);
            mFlightController.startLanding(new
CommonCallbacks.CompletionCallback() {
                @Override
```

```
                public void onResult(DJIError djiError) {
                    if (djiError != null) {
                        Log.d("startLanding: ", "ERROR: " +
djiError.getDescription());
                        finishFlight();
                    }
                }
            });
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        initFlightController();
        initPreviewer();
        onProductChange();
        if (mVideoSurface == null) {
            Log.e(TAG, "mVideoSurface is null");
        }
    }

    @Override
    protected void onDestroy() {
        unregisterReceiver(mReceiver);
        uninitPreviewer();
        super.onDestroy();
    }

    /**
     * @Description : RETURN Button RESPONSE FUNCTION
     */
    public void onReturn(View view) {
        Log.d(TAG, "onReturn");
        this.finish();
    }

    /**
     * shows any toast any time from anywhere.
     *
     * @param string
     */
    private void setResultToToast(final String string) {
        MainActivity.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(MainActivity.this, string,
Toast.LENGTH_SHORT).show();
            }
        });
    }

    protected BroadcastReceiver mReceiver = new BroadcastReceiver() {

        @Override
        public void onReceive(Context context, Intent intent) {
            onProductConnectionChange();
        }
    };

    private void onProductConnectionChange() {
        initFlightController();
        loginAccount();
    }

    private void loginAccount() {
    }
```

```
    protected void onProductChange() {
        initPreviewer();
        loginAccount();
    }

    /**
     * connects the previewer and video listener
     */
    private void initPreviewer() {
        BaseProduct product = getProductInstance();
        if (product == null || !product.isConnected()) {
            setResultToToast(getString(R.string.disconnected));
        } else {
            if (null != mVideoSurface) {
                mVideoSurface.setSurfaceTextureListener(this);
                mVideoSurface.setOnLongClickListener(new
View.OnLongClickListener() {
                    @Override
                    public boolean onLongClick(View v) {
                        return false;
                    }
                });
            }
            if (!product.getModel().equals(Model.UNKNOWN_AIRCRAFT)) {

VideoFeeder.getInstance().getPrimaryVideoFeed().addVideoDataListener(mReceiv
edVideoDataCallBack);
            }
        }
    }

    /**
     * disconnects the previewer and video listener
     */
    private void uninitPreviewer() {
        Camera camera = DJIDemoApplication.getCameraInstance();
        if (camera != null) {
            // Reset the callback

VideoFeeder.getInstance().getPrimaryVideoFeed().addVideoDataListener(null);
        }
    }

    @Override
    public void onSurfaceTextureAvailable(SurfaceTexture surface, int width,
int height) {
        if (mCodecManager == null) {
            mCodecManager = new DJICodecManager(this, surface, width,
height);
        }
    }

    /**
     * @param surface
     * @param width
     * @param height
     */
    @Override
    public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int
width, int height) {
    }

    /**
     * @param surface
     * @return - not in use
     */
    @Override
    public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {
```

```
        if (mCodecManager != null) {
            mCodecManager.cleanSurface();
            mCodecManager = null;
        }

        return false;
    }

    @Override
    public void onSurfaceTextureUpdated(SurfaceTexture surface) {
    }

    /**
     * sets camera mode to be ready to record video
     */
    private void switchCameraMode() {
        Camera camera = DJIDemoApplication.getCameraInstance();
        if (camera != null) {
            // For video: SettingsDefinitions.CameraMode.RECORD_VIDEO
            camera.setMode(SettingsDefinitions.CameraMode.SHOOT_PHOTO, new
CommonCallbacks.CompletionCallback() {
                @Override
                public void onResult(DJIError error) {

                    if (error == null) {
                        setResultToToast("Capture is ready");
                        //setResultToToast("Record is ready");
                    } else {
                        setResultToToast(error.getDescription());
                    }
                }
            });
        }
    }

    /**
     * Method for taking photo
     */
    private void captureAction() {
        final Camera camera = DJIDemoApplication.getCameraInstance();
        if (camera != null) {
            SettingsDefinitions.ShootPhotoMode photoMode =
SettingsDefinitions.ShootPhotoMode.SINGLE; // Set the camera capture mode as
Single mode
            camera.setShootPhotoMode(photoMode, new
CommonCallbacks.CompletionCallback() {
                @Override
                public void onResult(DJIError djiError) {
                    if (null == djiError) {
                        camera.startShootPhoto(new
CommonCallbacks.CompletionCallback() {
                            @Override
                            public void onResult(DJIError djiError) {
                                if (djiError == null) {

flight.setCountPictures(flight.getCountPictures() + 1);
                                    setResultToToast("take photo: success");
                                } else

setResultToToast(djiError.getDescription());

                            }
                        });
                    }
                }
            });
        }
    }
```

```
    private void takeOff() {
        if (mFlightController.isConnected()) {
            mFlightController.startTakeoff(new
CommonCallbacks.CompletionCallback() {
                @Override
                public void onResult(DJIError djiError) {
                    if (djiError != null) {
                        Log.d("startTakeOff: ", "ERROR: " +
djiError.getDescription());
                    }
                }
            });
        }
    }

    /**
     * Sets a location manager and request location update every second, if
location is greater than 1.5m will btnTakeoff it to the list of the mission
and tries to start it if list has more than 2 locations
     */
    @SuppressLint("MissingPermission")
    private void setLocationManager() {
        LocationManager lm = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        if (lm != null) {
            lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 2000,
// 2 sec
                    1.0f, new LocationListener() {
                        @Override
                        public void onLocationChanged(Location
locationPhone) {

                            if (startMission) {
                                if (droneLocation != null) {
                                    double droneBearing =
mFlightController.getCompass().getHeading();
                                    double newBearing =
droneLocation.bearingTo(locationPhone);
                                    float ls = (float) (newBearing -
droneBearing) / 100;
                                    double dist =
locationPhone.distanceTo(droneLocation);
                                    float fwd = (float) dist / 5;
                                    mrc.setLeftStickHorizontal(ls);
                                    if ((ls < 0.1) && (dist < 30))
                                        mrc.setRightStickVertical(fwd);
                                    else
                                        mrc.setRightStickVertical(0f);
                                }
                            }
                        }

                        @Override
                        public void onStatusChanged(String s, int i, Bundle
bundle) {
                        }

                        @Override
                        public void onProviderEnabled(String s) {
                        }

                        @Override
                        public void onProviderDisabled(String s) {
                        }
                    });

            lm.addGpsStatusListener(new GpsStatus.Listener() {
```

```
                @Override
                public void onGpsStatusChanged(int event) {
                    if (event == 2)
                        buildAlertMessageNoGps();
                }
            });
        } else
            setResultToToast("Cant get location try again");
    }

    /**
     * the gravity sensor callback, it is called on every change in the
angle of the device and this function
     * calculates the direction of the gimbal in which the user wants the
drone to see
     * (as long as the drone doesn't have to rotate).
     *
     * @param sensorEvent - not in use.
     */
    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        Aircraft aircraft = (Aircraft) getProductInstance();
        mrc = aircraft.getMobileRemoteController();

        gravity = sensorEvent.values;
        _y = initgravity[1] -
(Math.toDegrees(Math.acos(sensorEvent.values[1] / 9.81)));
        _z = initgravity[2] -
(Math.toDegrees(Math.acos(sensorEvent.values[2] / 9.81)));

        if (_y > max_range_y)
            _y = max_range_y;
        else if (_y < -1 * max_range_y)
            _y = -1 * max_range_y;

        if (_z > max_range_z)
            _z = max_range_z;
        else if (_z < -1 * max_range_z)
            _z = -1 * max_range_z;

        if (_y > 0 && _y < min_range_y)
            _y = 0;
        else if (_y < 0 && _y > -1 * min_range_y)
            _y = 0;

        if (_z > 0 && _z < min_range_z)
            _z = 0;
        else if (_z < 0 && _z > -1 * min_range_z)
            _z = 0;

        if (moveGimbal) {//camera
            _z = _z * -1;
            _y = _y * -1;
            aircraft.getGimbal().rotate(new
Rotation.Builder().mode(RotationMode.SPEED).pitch((float) (_z / max_range_z)
* 80)

.roll(Float.MAX_VALUE).yaw(Float.MAX_VALUE).time(0.0).build(), new
CommonCallbacks.CompletionCallback() {
                @Override
                public void onResult(DJIError djiError) {

                }
            });
            // mrc.setLeftStickHorizontal((float) (_y / max_range_y));
        }
    }
```

```java
    /**
     * not in use.
     *
     * @param sensor   - not in use.
     * @param accuracy - not in use.
     */
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    /**
     * this function calibrates the gravity sensor and sets the new position
as the "stand-still" position.
     */
    public void calibrate_gravity() {
        CountDownTimer ctd = new CountDownTimer(500, 500) {
            @Override
            public void onTick(long l) {
            }

            @Override
            public void onFinish() {
                initgravity[0] = (float)
(Math.toDegrees(Math.acos(gravity[0] / 9.81)));
                initgravity[1] = (float)
(Math.toDegrees(Math.acos(gravity[1] / 9.81)));
                initgravity[2] = (float)
(Math.toDegrees(Math.acos(gravity[2] / 9.81)));
                Toast.makeText(MainActivity.this, "Calibrated!",
Toast.LENGTH_SHORT).show();
            }
        };
        ctd.start();
    }

    /**
     * Turns off the mobile remote controller
     */
    private void gravityOff() {
        mrc.setLeftStickVertical(0);
        mrc.setRightStickHorizontal(0);
        mrc.setRightStickVertical(0);
    }

    /**
     * Checks if user have GPS on, if not will require him to turn on (must
be on for the mission)
     */
    private boolean buildAlertMessageNoGps() {
        final LocationManager manager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        if (manager != null &&
!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
            final AlertDialog.Builder builder = new
AlertDialog.Builder(this);
            builder.setMessage("Your GPS seems to be disabled. Please turn
it on")
                    .setCancelable(false)
                    .setPositiveButton("Turn On", new
DialogInterface.OnClickListener() {
                        public void onClick(@SuppressWarnings("unused")
final DialogInterface dialog, @SuppressWarnings("unused") final int id) {
                            startActivity(new
Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
                        }
                    });
            final AlertDialog alert = builder.create();
            alert.show();
```

```
            return false;
        }
        return true;
    }

    private void finishFlight() {
        startMission = false;
        String month, day, hour, minute;
        Date currentTime = Calendar.getInstance().getTime();
        int firstDigit = currentTime.getYear() % 100;
        int year = 2000 + firstDigit;
        if ((currentTime.getMonth() + 1) < 10)
            month = "0" + (currentTime.getMonth() + 1);
        else
            month = "" + (currentTime.getMonth() + 1);
        if (currentTime.getDate() < 10)
            day = "0" + currentTime.getDate();
        else
            day = "" + currentTime.getDate();
        if (currentTime.getHours() < 10)
            hour = "0" + currentTime.getHours();
        else
            hour = "" + currentTime.getHours();
        if (currentTime.getMinutes() < 10)
            minute = "0" + currentTime.getMinutes();
        else
            minute = "" + currentTime.getMinutes();
        flight.setDateEnd(year + "-" + month + "-" + day + " " + hour + ":"
+ minute);
        Intent intent = new Intent(MainActivity.this, SaveAct.class);
        intent.putExtra("email", email);
        intent.putExtra("flight", flight);
        startActivity(intent);
    }

}
```

Java:MyFlight:

```java
package com.dji.GSDemo.GoogleMap.Activities;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.widget.TextView;
import android.widget.Toast;

import com.dji.GSDemo.GoogleMap.Adapters.PictureAdapter;
import com.dji.GSDemo.GoogleMap.Classes.Flight;
import com.dji.GSDemo.GoogleMap.R;

import java.util.ArrayList;

/**
 * Shows info and pictures of a specific flight, the flight is chosen from
list of previous flights that were saved on firestore
 */
public class MyFlight extends AppCompatActivity {
    PictureAdapter myadapter;
    RecyclerView recyclerView;
    Flight flight;
    TextView tvName, tvDate;
    ArrayList<String> linksList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my_flight);
        tvDate = findViewById(R.id.tvDates);
        tvName = findViewById(R.id.tvName);
        recyclerView = findViewById(R.id.recycler);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        flight = (Flight)
```

```
getIntent().getExtras().getSerializable("flight_chosen");
        tvName.setText("" + flight.getName());
        tvDate.setText("Started at: " + flight.getDateStart() +
System.getProperty("line.separator") + "Ended: " + flight.getDateEnd());

        if (flight.getLinks().size() > 0) {
            linksList.addAll(flight.getLinks());
            myadapter = new PictureAdapter(MyFlight.this, linksList);
            recyclerView.setAdapter(myadapter);
        } else
            Toast.makeText(this, "No pictures where taken",
Toast.LENGTH_SHORT).show();
    }
}
```

```
package com.dji.GSDemo.GoogleMap.Activities;

import android.app.Dialog;
import android.content.Context;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.net.Uri;
import android.net.wifi.WifiInfo;
import android.net.wifi.WifiManager;
import android.os.Bundle;
import android.os.Environment;
import android.support.annotation.NonNull;
import android.support.design.widget.BottomNavigationView;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.dji.GSDemo.GoogleMap.Adapters.RecyclerViewAdapter;
import com.dji.GSDemo.GoogleMap.Classes.DJIDemoApplication;
import com.dji.GSDemo.GoogleMap.Classes.Flight;
import com.dji.GSDemo.GoogleMap.Classes.Preview;
import com.dji.GSDemo.GoogleMap.R;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
```

```java
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

import dji.common.camera.SettingsDefinitions;
import dji.common.error.DJICameraError;
import dji.common.error.DJIError;
import dji.common.util.CommonCallbacks;
import dji.log.DJILog;
import dji.sdk.media.DownloadListener;
import dji.sdk.media.MediaFile;
import dji.sdk.media.MediaManager;

/**
 * Used to save the flight and upload to firestore, will be opened once the
user pressed on saving the flight in bottom menu of MainActivity
 */
public class SaveAct extends AppCompatActivity {
    ArrayList<String> pathList = new ArrayList<>();
    ArrayList<Preview> previewsArray = new ArrayList<>();
    private static List<MediaFile> mediaFileList = new ArrayList<>();
    private static MediaManager mMediaManager;
    private MediaManager.FileListState currentFileListState =
MediaManager.FileListState.UNKNOWN;

    static RecyclerViewAdapter myadapter;
    RecyclerView recyclerView;

    ProgressBar progressBar;
    TextView tvLoad;

    ArrayList<String> links = new ArrayList<>();
    String email = "guest";
    Flight flight;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_save);
        if (getIntent().getStringExtra("email") != null &
!getIntent().getStringExtra("email").equals(""))
            email = getIntent().getStringExtra("email");
        progressBar = findViewById(R.id.progressBar);
        tvLoad = findViewById(R.id.tvLoad);
        flight = (Flight) getIntent().getExtras().getSerializable("flight");
        recyclerView = findViewById(R.id.recycler);
        initNav();

        initMediaManager();

        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

    }

    /**
     * connects the bottom menu, set click listener for it
     */
    private void initNav() {
        BottomNavigationView navigation = (BottomNavigationView)
findViewById(R.id.navigation);
        navigation.getMenu().findItem(R.id.navigation_fly).setChecked(true);
```

```
        navigation.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem
menuItem) {
                switch (menuItem.getItemId()) {
                    case R.id.navigation_fly:
                        Intent intent = new Intent(SaveAct.this,
MainActivity.class);
                        intent.putExtra("flight", flight);
                        startActivity(intent);
                        finish();
                        return true;

                    case R.id.navigation_gallery:
                        Toast.makeText(SaveAct.this, "Already in activity",
Toast.LENGTH_SHORT).show();
                        return true;

                    case R.id.navigation_edit:
                        showEditDialog();
                        return true;
                }
                return false;
            }
        });
    }

    /**
     * shows any toast any time from anywhere.
     *
     * @param string
     */
    private void setResultToToast(final String string) {
        SaveAct.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(SaveAct.this, string,
Toast.LENGTH_SHORT).show();
            }
        });
    }

    /**
     * Gets list of the files from the SD card on the drone
     */
    private void getFileList() {
        mMediaManager =
DJIDemoApplication.getCameraInstance().getMediaManager();
        if (mMediaManager != null) {
            if ((currentFileListState == MediaManager.FileListState.SYNCING)
|| (currentFileListState == MediaManager.FileListState.DELETING))
                DJILog.e("SaveAct: ", "Media Manager is busy.");
            else {

mMediaManager.refreshFileListOfStorageLocation(SettingsDefinitions.StorageLo
cation.SDCARD, new CommonCallbacks.CompletionCallback() {
                    @Override
                    public void onResult(DJIError djiError) {
                        if (null == djiError) {
                            //Reset data
                            if (currentFileListState !=
MediaManager.FileListState.INCOMPLETE)
                                mediaFileList.clear();

                            mediaFileList =
mMediaManager.getSDCardFileListSnapshot();
                            for (int i = 0; i < mediaFileList.size(); i++) {
```

```
                                progressBar.setProgress((100 * i) /
mediaFileList.size());
                                downloadFileByIndex(i);
                            }
                        } else
                            setResultToToast("Get Media File List Failed:" +
djiError.getDescription());
                    }
                });

            }
        }
    }

    /**
     * Initializing the media manager, then calls the get file list func to
get photos and videos
     */
    private void initMediaManager() {
        if (DJIDemoApplication.getProductInstance() == null) {
            mediaFileList.clear();
            return;
        } else {
            if (null != DJIDemoApplication.getCameraInstance() &&
DJIDemoApplication.getCameraInstance().isMediaDownloadModeSupported()) {
                mMediaManager =
DJIDemoApplication.getCameraInstance().getMediaManager();
                if (null != mMediaManager) {
                    mMediaManager.addUpdateFileListStateListener(new
MediaManager.FileListStateListener() {
                        @Override
                        public void
onFileListStateChange(MediaManager.FileListState fileListState) {
                            currentFileListState = fileListState;
                        }
                    });

DJIDemoApplication.getCameraInstance().setMode(SettingsDefinitions.CameraMod
e.MEDIA_DOWNLOAD, new CommonCallbacks.CompletionCallback() {
                        @Override
                        public void onResult(DJIError error) {
                            if (error == null)
                                getFileList();
                            else {
                                setResultToToast("Set cameraMode failed");
                                Toast.makeText(SaveAct.this, "" +
error.getDescription(), Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
                }
            } else if (null != DJIDemoApplication.getCameraInstance() &&
!DJIDemoApplication.getCameraInstance().isMediaDownloadModeSupported())
                setResultToToast("Media Download Mode not Supported");
        }
        return;
    }

    /**
     * Download the photo/video from drone to phone
     *
     * @param i - position of photo/ video in the list
     */
    private void downloadFileByIndex(final int i) {
        File destDir = new
File(Environment.getExternalStorageDirectory().getPath() + "/GSdemo/");
        if ((mediaFileList.get(i).getMediaType() ==
MediaFile.MediaType.PANORAMA)
```

```
                || (mediaFileList.get(i).getMediaType() ==
MediaFile.MediaType.SHALLOW_FOCUS)) {
            return;
        }
        mediaFileList.get(i).fetchFileData(destDir, null, new
DownloadListener<String>() {
            @Override
            public void onStart() {
            }

            @Override
            public void onRateUpdate(long total, long current, long persize)
{
            }

            @Override
            public void onProgress(long l, long l1) {
            }

            @Override
            public void onSuccess(final String filePath) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        MediaFile selectedMedia = mediaFileList.get(i);
                        Toast.makeText(SaveAct.this, "selected created: " +
selectedMedia.getDateCreated() + " flight started: " + flight.getDateStart()
+ " ended: " + flight.getDateEnd(), Toast.LENGTH_SHORT).show();
                        if (selectedMedia.getDateCreated().substring(0,
4).equals(flight.getDateStart().substring(0, 4)) &&
selectedMedia.getDateCreated().substring(5,
7).equals(flight.getDateStart().substring(5, 7))
                                &&
selectedMedia.getDateCreated().substring(8,
10).equals(flight.getDateStart().substring(8, 10))
                                &&
Integer.parseInt(flight.getDateStart().substring(11, 13)) <=
Integer.parseInt(selectedMedia.getDateCreated().substring(11, 13)) &&
Integer.parseInt(flight.getDateEnd().substring(11, 13)) >=
Integer.parseInt(selectedMedia.getDateCreated().substring(11, 13))
                                &&
Integer.parseInt(flight.getDateStart().substring(14, 16)) <=
Integer.parseInt(selectedMedia.getDateCreated().substring(14, 16)) &&
Integer.parseInt(flight.getDateEnd().substring(14, 16)) <=
Integer.parseInt(selectedMedia.getDateCreated().substring(14, 16))) {

                            Bitmap myBitmap =
BitmapFactory.decodeFile(filePath + "/" + selectedMedia.getFileName());
                            previewsArray.add(new Preview(myBitmap,
selectedMedia.getDateCreated()));
                            myadapter = new
RecyclerViewAdapter(SaveAct.this, previewsArray);
                            recyclerView.setAdapter(myadapter);
                            pathList.add(filePath + "/" +
selectedMedia.getFileName());
                            if (i == mediaFileList.size())
                                tvLoad.setText("Finished");
                        }
                    }
                });
            }

            @Override
            public void onFailure(DJIError djiError) {
                setResultToToast("Download File Failed" +
djiError.getDescription());
            }
        });
```

```
    }

    /**
     * @param index   - Index of the photo in the list
     * @param context - SaveAct
     */
    public static void deleteFileByIndex(final int index, final Context
context) {
        ArrayList<MediaFile> fileToDelete = new ArrayList<>();
        if (mediaFileList.size() > index) {
            fileToDelete.add(mediaFileList.get(index));
            mMediaManager.deleteFiles(fileToDelete, new
CommonCallbacks.CompletionCallbackWithTwoParam<List<MediaFile>,
DJICameraError>() {
                @Override
                public void onSuccess(List<MediaFile> x, DJICameraError y) {
                    Toast.makeText(context, "Delete file from drone
success", Toast.LENGTH_SHORT).show();
                    MediaFile file = mediaFileList.remove(index);
                    //Update recyclerView
                    myadapter.notifyItemRemoved(index);
                }

                @Override
                public void onFailure(DJIError error) {
                    Toast.makeText(context, "Delete file failed",
Toast.LENGTH_SHORT).show();
                }
            });
        }
    }

    /**
     * Shows dialog for saving the flight in firebase
     */
    private void showEditDialog() {
        final Dialog d = new Dialog(this);
        d.setContentView(R.layout.edit_dialog);

d.getWindow().setBackgroundDrawableResource(android.R.color.transparent);
        d.setCancelable(false);
        final EditText etName = d.findViewById(R.id.etName);
        EditText etDates = d.findViewById(R.id.etDates);
        EditText etEmail = d.findViewById(R.id.etEmail);
        EditText etPicture = d.findViewById(R.id.etPicture);
        etDates.setText("Started: " + flight.getDateStart() + " Ended: " +
flight.getDateEnd());
        etEmail.setText("" + flight.getEmail());
        etPicture.setText(pathList.size() + " pictures taken");
        Button btnSave = d.findViewById(R.id.btnSave);
        btnSave.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (etName.getText().toString().equals(null) ||
etName.getText().toString().equals(""))
                    Toast.makeText(SaveAct.this, "Please set a name",
Toast.LENGTH_SHORT).show();
                else {
                    if (connectedToDrone())
                        setResultToToast("Please disconnect from drone");
                    else {
                        setResultToToast("Saving flight....");
                        flight.setName(etName.getText().toString());
                        d.dismiss();
                        uploadToStorage();
                    }
```

```
                }
            }
        });
        d.show();
    }

    private boolean connectedToDrone() {
        ConnectivityManager connManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo mWifi =
connManager.getNetworkInfo(ConnectivityManager.TYPE_WIFI);

        if (mWifi.isConnected()) {
            WifiManager wifiManager = (WifiManager)
getApplicationContext().getSystemService(Context.WIFI_SERVICE);
            WifiInfo info = wifiManager.getConnectionInfo();
            String ssid = info.getSSID();
            return ssid.contains("spark".toLowerCase()) ||
ssid.contains("mavic".toLowerCase());
        } else
            return false;
    }

    /**
     * Uploads all photos taken during the last flight to firebase storage,
saves the download links in links ArrayList
     */
    private void uploadToStorage() {
        StorageReference mStorageRef =
FirebaseStorage.getInstance().getReference(flight.getEmail());

        for (int i = 0; i < pathList.size(); i++) {
            String filepath = pathList.get(i);
            Uri file = Uri.fromFile(new File(filepath));

            final StorageReference fileReference =
mStorageRef.child(flight.getName() + "_" + i);
            fileReference.putFile(file).addOnCompleteListener(new
OnCompleteListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onComplete(@NonNull
Task<UploadTask.TaskSnapshot> task) {
                    if (task.isSuccessful()) {

fileReference.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
                            @Override
                            public void onSuccess(Uri downloadUrl) {
                                String url = downloadUrl.toString();
                                links.add(url);
                                if (links.size() == previewsArray.size()) {
                                    flight.setCountPictures(links.size());
                                    flight.setLinks(links);
                                    uploadToFirestore();
                                }
                            }
                        });
                    }
                }
            })
                    .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception exception)
{
                            setResultToToast("error: " +
exception.getCause().getLocalizedMessage());
                        }
                    });
```

```
        }
    }

    /**
     * Saves flight's info in firestore (name, user's email, how many
pictures taken, time of start and end, links of the taken pictures
     */
    private void uploadToFirestore() {
        DocumentReference docRef =
FirebaseFirestore.getInstance().collection(flight.getEmail()).document(fligh
t.getName());
        docRef.set(flight)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText(SaveAct.this, "saved",
Toast.LENGTH_LONG).show();
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(SaveAct.this, "error",
Toast.LENGTH_LONG).show();

                    }
                });
    }
}
```

\

```
package com.dji.GSDemo.GoogleMap.Activities;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

import com.dji.GSDemo.GoogleMap.R;
import com.google.firebase.auth.FirebaseAuth;

/**
 * First activity, Checks if user has signed to the app using a google
account, will move to ConnectionActivity after 2 seconds
 */
public class Splash extends AppCompatActivity {

    public FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);
        mAuth = FirebaseAuth.getInstance();

        ImageView photo = (ImageView) findViewById(R.id.imageView);
        Animation myanim = AnimationUtils.loadAnimation(this,
R.anim.mytransition);
        photo.startAnimation(myanim);
        final Intent k = new Intent(getBaseContext(),
ConnectionActivity.class);
        Thread timer = new Thread() {
            public void run() {
                try {
                    sleep(2000);
                } catch (InterruptedException e) {
```

```
                    e.printStackTrace();
                } finally {
                    if (mAuth.getCurrentUser() != null) {
                        k.putExtra("Signed", true);
                        startActivity(k);
                        finish();
                    } else {
                        k.putExtra("Signed", false);
                        startActivity(k);
                        finish();
                    }
                }
            }
        };
        timer.start();
    }

}
```

```
package com.dji.GSDemo.GoogleMap.Adapters;

import android.content.Context;
import android.content.Intent;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.dji.GSDemo.GoogleMap.Activities.MyFlight;
import com.dji.GSDemo.GoogleMap.Classes.Flight;
import com.dji.GSDemo.GoogleMap.R;

import java.util.ArrayList;

/**
 * Adapter for recyclerView on Allflight Activity, will show the name of the
flight, email of the user that made it and amount of
 * pictures taken during the flight
 */
public class FlightAdapter extends
RecyclerView.Adapter<FlightAdapter.viewHolder> {
    private Context mContext;
    private ArrayList<Flight> mitemList;


    public FlightAdapter(Context context, ArrayList<Flight> itemList) {
        mContext = context;
        mitemList = itemList;

    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(mContext).inflate(R.layout.flight_row,
parent, false);
```

```java
        return new viewHolder(v);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, final int position) {
        final Flight currentItem = mitemList.get(position);

        holder.tvName.setText(currentItem.getName());
        holder.tvEmail.setText(currentItem.getEmail());
        holder.tvPic.setText("Pics: " + currentItem.getCountPictures());

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent forItem = new Intent(mContext, MyFlight.class);
                forItem.putExtra("flight_chosen", mitemList.get(position));
                mContext.startActivity(forItem);
            }
        });
    }

    @Override
    public int getItemCount() {
        return mitemList.size();
    }

    public class viewHolder extends RecyclerView.ViewHolder {

        public TextView tvName, tvEmail, tvPic;


        public viewHolder(final View itemView) {
            super(itemView);

            tvName = (TextView) itemView.findViewById(R.id.tvName);
            tvEmail = (TextView) itemView.findViewById(R.id.tvEmail);
            tvPic = (TextView) itemView.findViewById(R.id.tvPic);
        }
    }

}
```

```java
package com.dji.GSDemo.GoogleMap.Adapters;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;

import com.dji.GSDemo.GoogleMap.R;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;

/**
 * Adapter to show the pictures that were taken during a flight in the
MyFlight activity, only shows a picture without on click method
 */
public class PictureAdapter extends
RecyclerView.Adapter<PictureAdapter.viewHolder> {
    private Context mContext;
    private ArrayList<String> mitemList;


    public PictureAdapter(Context context, ArrayList<String> itemList) {
        mContext = context;
        mitemList = itemList;
    }

    @Override
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(mContext).inflate(R.layout.picture_row,
parent, false);
        return new viewHolder(v);
    }

    @Override
```

```
    public void onBindViewHolder(viewHolder holder, final int position) {


Picasso.with(mContext).load(mitemList.get(position)).fit().centerInside().in
to(holder.mImageView);

    }

    @Override
    public int getItemCount() {
        return mitemList.size();
    }

    public class viewHolder extends RecyclerView.ViewHolder {
        public ImageView mImageView;


        public viewHolder(final View itemView) {
            super(itemView);
            mImageView = (ImageView) itemView.findViewById(R.id.thumbnail);
        }
    }


}
```

```
package com.dji.GSDemo.GoogleMap.Adapters;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import com.dji.GSDemo.GoogleMap.Activities.SaveAct;
import com.dji.GSDemo.GoogleMap.Classes.Preview;
import com.dji.GSDemo.GoogleMap.R;

import java.util.ArrayList;

/**
 * Adapter to show pictures taken during last flight while saving in
SaveAct, contains bitmap of the picture and the time the picture was
 * taken at. User has an option to delete the picture from the phone and
drone (will not upload the picture to firestore)
 */
public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewAdapter.viewHolder> {
    private Context mContext;
    private ArrayList<Preview> mitemList;


    public RecyclerViewAdapter(Context context, ArrayList<Preview> itemList)
{
        mContext = context;
        mitemList = itemList;
    }

    @Override
```

```
    public viewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View v = LayoutInflater.from(mContext).inflate(R.layout.row_item,
parent, false);
        return new viewHolder(v);
    }

    @Override
    public void onBindViewHolder(viewHolder holder, final int position) {
        final Preview currentItem = mitemList.get(position);

        holder.mImageView.setImageBitmap(currentItem.getBitmap());

        holder.tvDate.setText("Taken at: " + currentItem.getDateCreated());

        holder.btnDelete.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SaveAct.deleteFileByIndex(position, mContext);
            }
        });
    }

    @Override
    public int getItemCount() {
        return mitemList.size();
    }

    public class viewHolder extends RecyclerView.ViewHolder {
        public ImageView mImageView;
        public TextView tvDate;
        public Button btnDelete;

        public viewHolder(final View itemView) {
            super(itemView);
            mImageView = (ImageView)
itemView.findViewById(R.id.img_thumbnail);
            tvDate = (TextView) itemView.findViewById(R.id.tvDate);
            btnDelete = (Button) itemView.findViewById(R.id.btnDelete);

        }
    }


}
```

```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.support.v4.app.NotificationCompat;

import com.dji.GSDemo.GoogleMap.Activities.ConnectionActivity;
import com.dji.GSDemo.GoogleMap.R;

import static android.app.PendingIntent.FLAG_ONE_SHOT;

/**
 * Created by azem on 10/29/17.
 */

public class AlarmReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        NotificationCompat.Builder builder;

        NotificationManager notificationManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {

            /* Create or update. */
            NotificationChannel channel = new NotificationChannel("1", "We
miss you!",
                    NotificationManager.IMPORTANCE_DEFAULT);
            channel.setDescription("Please fly with us again soon");
            channel.enableLights(true);
```

```
            channel.setLightColor(Color.BLUE);
            notificationManager.createNotificationChannel(channel);
            builder = new NotificationCompat.Builder(context,
channel.getId());
        } else
            builder = new NotificationCompat.Builder(context);

        Intent myIntent = new Intent(context, ConnectionActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
myIntent, FLAG_ONE_SHOT);

        builder.setAutoCancel(true)
                .setDefaults(Notification.DEFAULT_ALL)
                .setWhen(System.currentTimeMillis())
                .setSmallIcon(R.mipmap.ic_launcher)
                .setContentTitle("We miss you!")
                .setContentIntent(pendingIntent)
                .setContentText("Please fly with us again soon")
                .setDefaults(Notification.DEFAULT_LIGHTS |
Notification.DEFAULT_SOUND)
                .setContentInfo("Info");

        notificationManager.notify(1, builder.build());
    }
}
```

```
package com.dji.GSDemo.GoogleMap.Classes;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.support.v4.content.ContextCompat;
import android.widget.Toast;

public class BootReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {

        //code to execute when Boot Completd

        Intent i = new Intent(context, NotificationService.class);
            ContextCompat.startForegroundService(context,i);

        Toast.makeText(context, "Booting Completed",
Toast.LENGTH_LONG).show();

    }
}
```

```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.app.Application;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import android.os.Handler;
import android.os.Looper;
import android.support.multidex.MultiDex;
import android.support.v4.content.ContextCompat;
import android.util.Log;
import android.widget.Toast;

import dji.sdk.base.BaseComponent;
import dji.sdk.base.BaseProduct;
import dji.sdk.camera.Camera;
import dji.sdk.products.Aircraft;
import dji.sdk.products.HandHeld;
import dji.sdk.sdkmanager.DJISDKManager;
import dji.common.error.DJIError;
import dji.common.error.DJISDKError;

/**
 * Class used for the dji api
 */
public class DJIDemoApplication extends Application {

    private static final String TAG = DJIDemoApplication.class.getName();

    public static final String FLAG_CONNECTION_CHANGE =
"dji_sdk_connection_change";

    private DJISDKManager.SDKManagerCallback mDJISDKManagerCallback;
    private static BaseProduct mProduct;
    public Handler mHandler;

    private Application instance;
```

```java
    public void setContext(Application application) {
        instance = application;
    }

    @Override
    public Context getApplicationContext() {
        return instance;
    }

    public DJIDemoApplication() {

    }

    public static synchronized BaseProduct getProductInstance() {
        if (null == mProduct) {
            mProduct = DJISDKManager.getInstance().getProduct();
        }
        return mProduct;
    }

    @Override
    public void onCreate() {
        super.onCreate();

        mHandler = new Handler(Looper.getMainLooper());

        /**
         * When starting SDK services, an instance of interface
DJISDKManager.DJISDKManagerCallback will be used to listen to
         * the SDK Registration result and the product changing.
         */
        mDJISDKManagerCallback = new DJISDKManager.SDKManagerCallback() {

            //Listens to the SDK registration result
            @Override
            public void onRegister(DJIError error) {

                if (error == DJISDKError.REGISTRATION_SUCCESS) {

                    Handler handler = new Handler(Looper.getMainLooper());
                    handler.post(new Runnable() {
                        @Override
                        public void run() {
                            Toast.makeText(getApplicationContext(),
"Register Success", Toast.LENGTH_LONG).show();
                        }
                    });

                    DJISDKManager.getInstance().startConnectionToProduct();

                } else {

                    Handler handler = new Handler(Looper.getMainLooper());
                    handler.post(new Runnable() {

                        @Override
                        public void run() {
                            Toast.makeText(getApplicationContext(),
"Register sdk fails, check network is available", Toast.LENGTH_LONG).show();
                        }
                    });

                }
                Log.e("TAG", error.toString());
            }

            @Override
```

```
            public void onProductDisconnect() {
                Log.d("TAG", "onProductDisconnect");
                notifyStatusChange();
            }

            @Override
            public void onProductConnect(BaseProduct baseProduct) {
                Log.d("TAG", String.format("onProductConnect newProduct:%s",
baseProduct));
                notifyStatusChange();

            }

            @Override
            public void onComponentChange(BaseProduct.ComponentKey
componentKey, BaseComponent oldComponent,
                                          BaseComponent newComponent) {
                if (newComponent != null) {
                    newComponent.setComponentListener(new
BaseComponent.ComponentListener() {

                        @Override
                        public void onConnectivityChange(boolean
isConnected) {
                            Log.d("TAG", "onComponentConnectivityChanged: "
+ isConnected);

                            notifyStatusChange();
                        }
                    });
                }

                Log.d("TAG",
                        String.format("onComponentChange key:%s,
oldComponent:%s, newComponent:%s",
                                      componentKey,
                                      oldComponent,
                                      newComponent));

            }
        };
        //Check the permissions before registering the application for
android system 6.0 above.
        int permissionCheck =
ContextCompat.checkSelfPermission(getApplicationContext(),
android.Manifest.permission.WRITE_EXTERNAL_STORAGE);
        int permissionCheck2 =
ContextCompat.checkSelfPermission(getApplicationContext(),
android.Manifest.permission.READ_PHONE_STATE);
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M ||
(permissionCheck == 0 && permissionCheck2 == 0)) {
            //This is used to start SDK services and initiate SDK.
            DJISDKManager.getInstance().registerApp(getApplicationContext(),
mDJISDKManagerCallback);
            Toast.makeText(getApplicationContext(), "registering, pls
wait...", Toast.LENGTH_LONG).show();

        } else {
            Toast.makeText(getApplicationContext(), "Please check if the
permission is granted.", Toast.LENGTH_LONG).show();
        }

    }

    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        MultiDex.install(this);
    }
```

```java
    private void notifyStatusChange() {
        mHandler.removeCallbacks(updateRunnable);
        mHandler.postDelayed(updateRunnable, 500);
    }

    private Runnable updateRunnable = new Runnable() {

        @Override
        public void run() {
            Intent intent = new Intent(FLAG_CONNECTION_CHANGE);
            getApplicationContext().sendBroadcast(intent);
        }
    };

    public static synchronized Camera getCameraInstance() {

        if (getProductInstance() == null) return null;

        Camera camera = null;

        if (getProductInstance() instanceof Aircraft) {
            camera = ((Aircraft) getProductInstance()).getCamera();

        } else if (getProductInstance() instanceof HandHeld) {
            camera = ((HandHeld) getProductInstance()).getCamera();
        }

        return camera;
    }
}


package com.dji.GSDemo.GoogleMap.Classes;


import java.io.Serializable;
import java.util.ArrayList;

/**
 * Object of the flight, each time the user makes a new flight their email
will automatically be in the flight object as well as the date the flight
started (the date of pressing new flight)
 */
public class Flight implements Serializable {
    private String name;
    private String email;
    private int countPictures;
    private String dateStart;
    private String dateEnd;
    private ArrayList<String> links;

    public Flight() {
    }


    public Flight(String name, String email, int countPictures, String
dateStart, String dateEnd, ArrayList<String> links) {
        this.name = name;
        this.email = email;
        this.countPictures = countPictures;
        this.dateStart = dateStart;
        this.dateEnd = dateEnd;
        this.links = links;
    }

    public String getName() {
        return name;
    }
```

```java
    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public int getCountPictures() {
        return countPictures;
    }

    public void setCountPictures(int countPictures) {
        this.countPictures = countPictures;
    }

    public String getDateStart() {
        return dateStart;
    }

    public void setDateStart(String dateStart) {
        this.dateStart = dateStart;
    }

    public String getDateEnd() {
        return dateEnd;
    }

    public void setDateEnd(String dateEnd) {
        this.dateEnd = dateEnd;
    }

    public ArrayList<String> getLinks() {
        return links;
    }

    public void setLinks(ArrayList<String> links) {
        this.links = links;
    }


}
```

```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.app.Application;
import android.content.Context;

import com.secneo.sdk.Helper;

/**
 * Used for the dji api
 */
public class MApplication extends Application {

    private DJIDemoApplication fpvDemoApplication;

    @Override
    protected void attachBaseContext(Context paramContext) {
        super.attachBaseContext(paramContext);
        Helper.install(MApplication.this);
        if (fpvDemoApplication == null) {
            fpvDemoApplication = new DJIDemoApplication();
            fpvDemoApplication.setContext(this);
        }
    }

    @Override
    public void onCreate() {
        super.onCreate();
        fpvDemoApplication.onCreate();
    }

}
```

```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.app.AlarmManager;
import android.app.Notification;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import android.os.SystemClock;
import android.support.annotation.NonNull;
import android.support.v4.app.JobIntentService;

import java.util.Calendar;

public class NotificationService extends Service {

    public NotificationService() {
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        startForeground(1, new Notification());
        startAlarm(true, true);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        return START_NOT_STICKY;
    }
```

```java
    private void startAlarm(boolean isNotification, boolean isRepeat) {
        AlarmManager manager = (AlarmManager)
getSystemService(Context.ALARM_SERVICE);
        Intent myIntent;
        PendingIntent pendingIntent;

        //THIS IS WHERE YOU SET NOTIFICATION TIME FOR CASES WHEN THE
NOTIFICATION NEEDS TO BE RESCHEDULED
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.HOUR_OF_DAY, 10);
        calendar.set(Calendar.MINUTE, 00);

        myIntent = new Intent(this, AlarmReceiver.class);
        pendingIntent = PendingIntent.getBroadcast(this, 0, myIntent, 0);


        if (!isRepeat)
            manager.set(AlarmManager.RTC_WAKEUP,
SystemClock.elapsedRealtime() + 300, pendingIntent);
        else
            manager.setRepeating(AlarmManager.RTC_WAKEUP,
calendar.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pendingIntent);
    }
}
```

```java
package com.dji.GSDemo.GoogleMap.Classes;

import android.graphics.Bitmap;

/**
 * Used to show the pictures taken by user using the drone and will be
showed in the saving activity
 * bitman for showing the picture from the phone using the path, date of
when the picture was taken (getting it using the dji api)
 */
public class Preview {
    private Bitmap bitmap;
    private String dateCreated;

    public Preview(Bitmap bitmap, String dateCreated) {
        this.bitmap = bitmap;
        this.dateCreated = dateCreated;
    }

    public Bitmap getBitmap() {
        return bitmap;
    }

    public void setBitmap(Bitmap bitmap) {
        this.bitmap = bitmap;
    }

    public String getDateCreated() {
        return dateCreated;
    }

    public void setDateCreated(String dateCreated) {
        this.dateCreated = dateCreated;
    }
}
```