# FaceMusic

https://www.youtube.com/watch?v=qLjUGwrKTVs

https://github.com/almogre02/FaceMusic

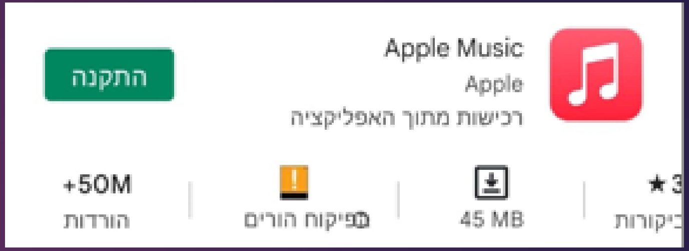
## Goals

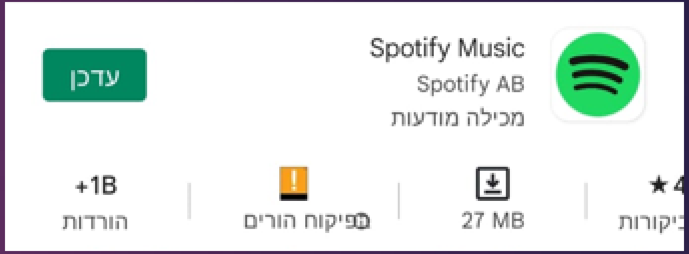
**Present:** The main purpose of the app is to be added as an add-on to music apps like Spotify, Apple Music (they have about 1 billion and 50 million downloads respectively).

**Future:**

- If the app will succeed, the goal will be to expand into the movie market. The idea: after the shoot, the user will be shown a list of movies according to the emotion he identified (for example: on Netflix, there are categories for each movie: fear, comedy and more .. those categories related to our set of emotions).
- Connection with Voice Recognition - Using voice recognition to recognize emotions, a user records a sentence and shortly afterwards the app recognizes the related emotion.

**Currently we have not found any similar app.**

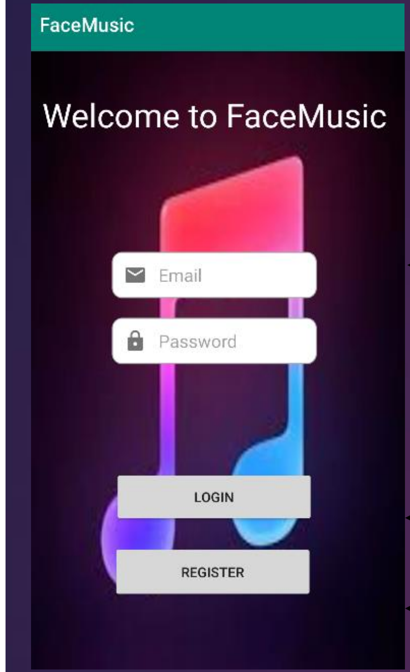

Apple Music
Apple
רכישות מתוך האפליקציה
התקנה
+50M הורדות | פיקוח הורים | 45 MB | ★ 3 ביקורות


Spotify Music
Spotify AB
מכילה מודעות
עדכן
+1B הורדות | פיקוח הורים | 27 MB | ★ 4 ביקורות

## Business goals:

- It is currently intended to be used as an add-on to well-known music apps such as: YouTube, Apple Music, Spotify, etc.
- Artists will also be able to purchase Creator users, match their songs to the right emotions, and upload them to the app, exposing their songs to a new audience.
- By adding another method other than searching for a song by name, music apps can expand their target audience

# Register and Login



Text input - Email form (****@****.***), Verify that the user has entered the appropriate text

Text input - Masking the users password ,Verify that the user has entered the appropriate text.

Login Buttons - sends request to FireBase database to login and awaits respond.

Register Button - sends to Register screen

Text input - Email form (****@****.***), Verify that the user has entered the appropriate text

Text input - Masking the users password ,Verify that the user has entered the appropriate text.

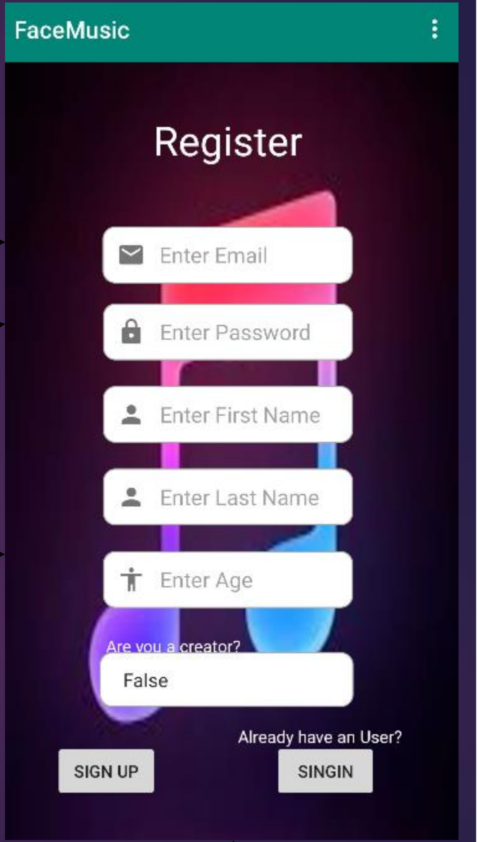
Text input - Regular text

Text input - Number text

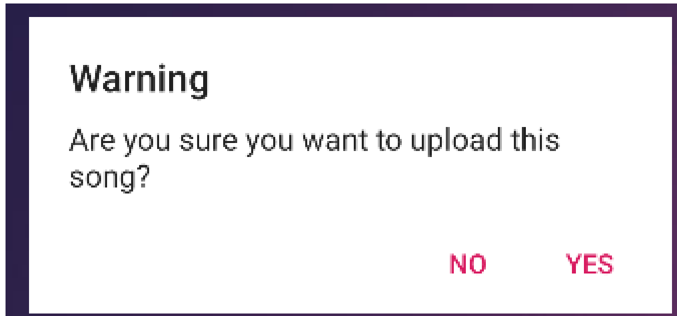
Spinner - Bollean Spinner.

SignUp Button - sends request to FireBase database to register the user, Verify that the user has entered the appropriate text.

SingIn Button - sends to Login screen.



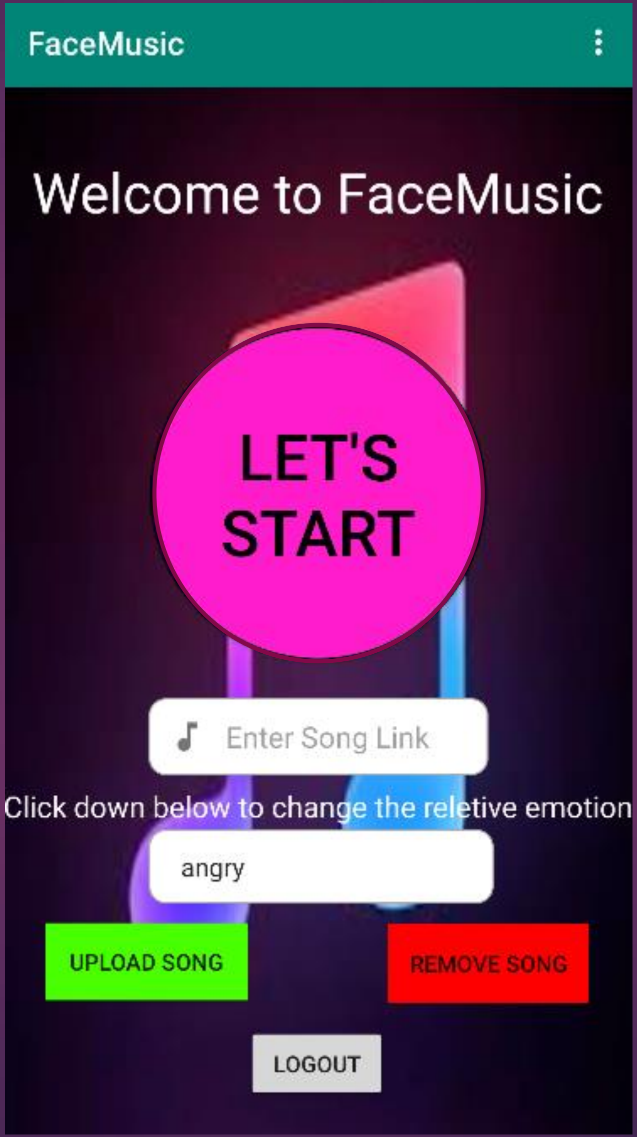
# Creator User


Warning

Are you sure you want to upload this song?

NO YES

Popup Message  after cliking Upload Song / Remove Song Buttons.



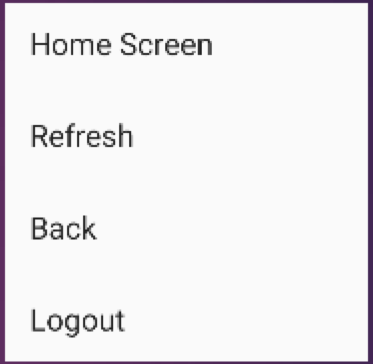
Let's Start Button - Sends the user to the emotion detection.

Text input - Link of the YouTube video you want to upload.

Upload Song Button - sends request to FireBase real time database to add the song.

Logout - Logout the user and sends to login screen.

Menu Button - Open menu.



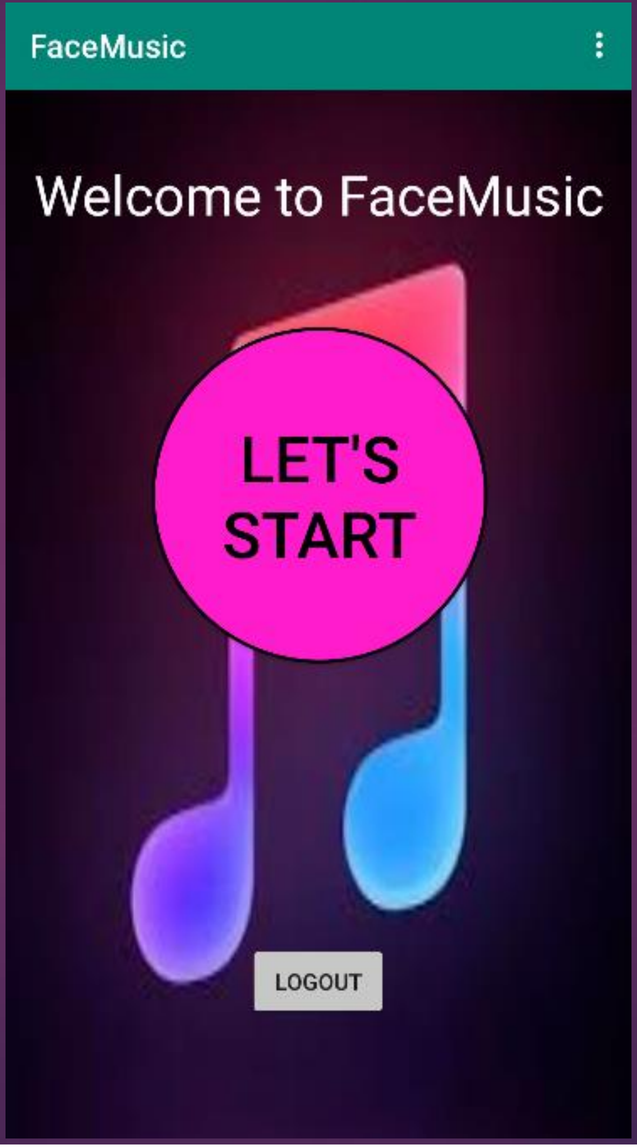
Emotion Spinner - Choose the emotion associated with the song you want to upload.

Remove Song Button -Verify that the song is is the database. sends request to FireBase real time database to delete the song.

## Client User

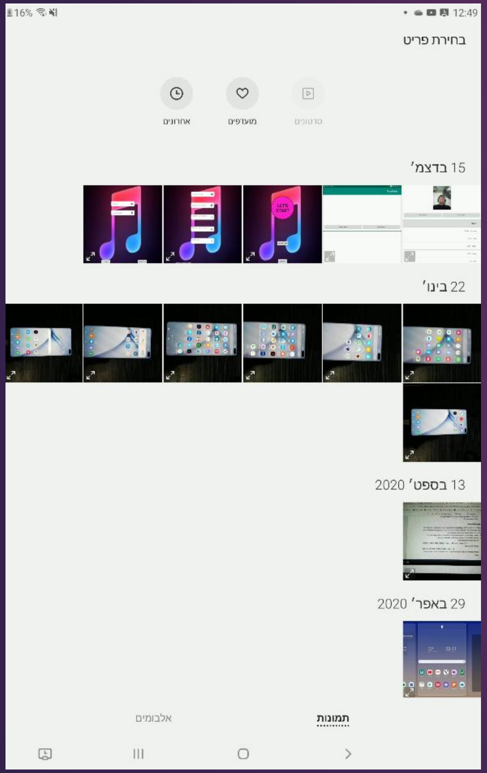
Let's Start Button - Sends the user to the emotion detection.

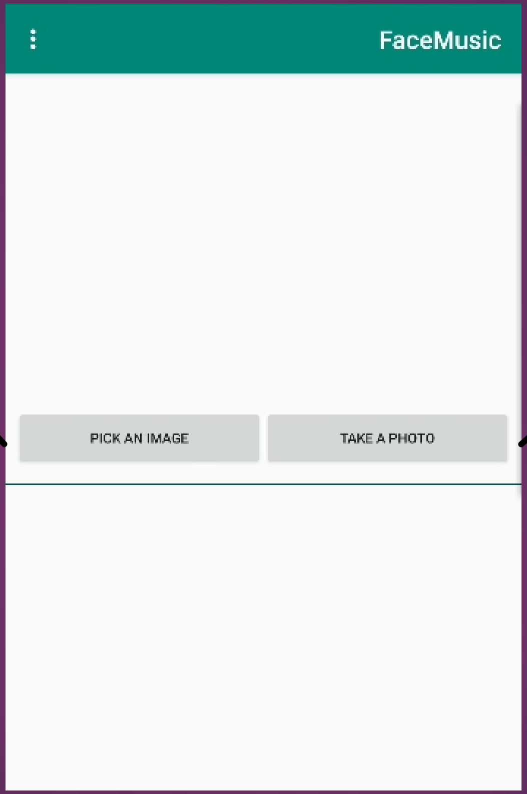
Logout - Logout the user and sends to login screen.



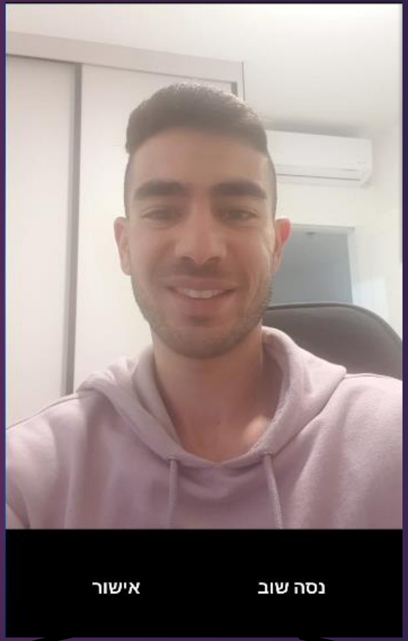
# Main Activity

PICK AN IMAGE Button - Opens the gallery on the user's phone.





TAKE A PHOTO Button - Activates the camera on the user's phone.

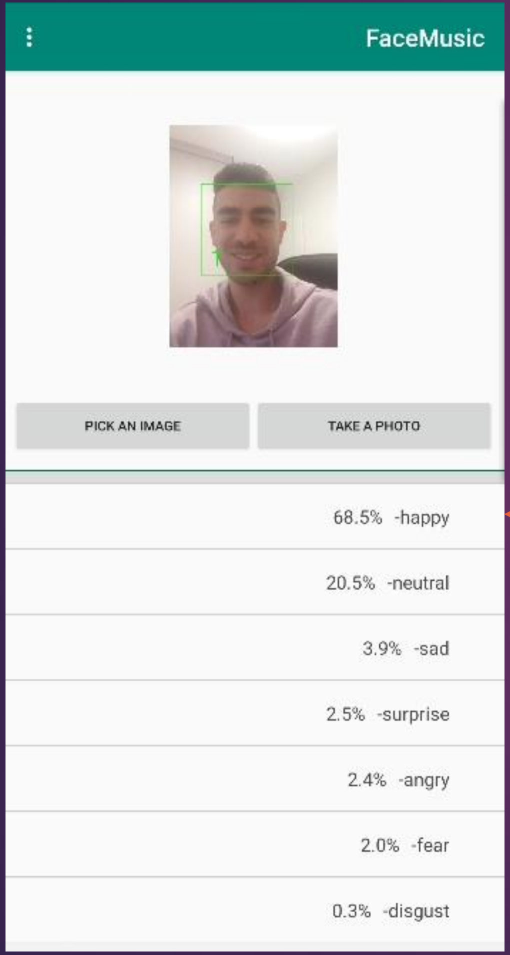


Confirmation Button - Sends the user to the image diagnostic results .

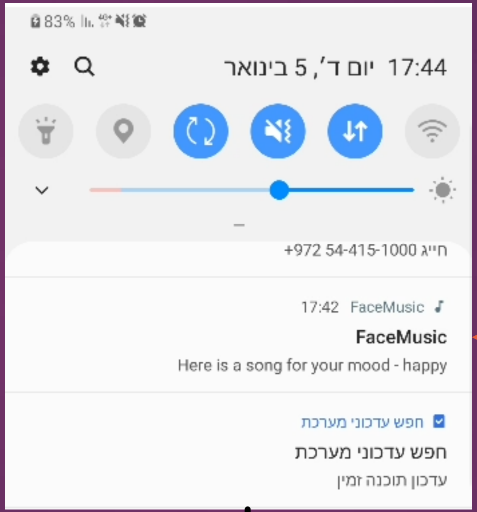
Try Again Button - Reactivates the camera on the user's phone.

# YouTube

## Image diagnostic results



68.5% -happy
20.5% -neutral
3.9% -sad
2.5% -surprise
2.4% -angry
2.0% -fear
0.3% -disgust



17:44 יום ד׳, 5 בינואר

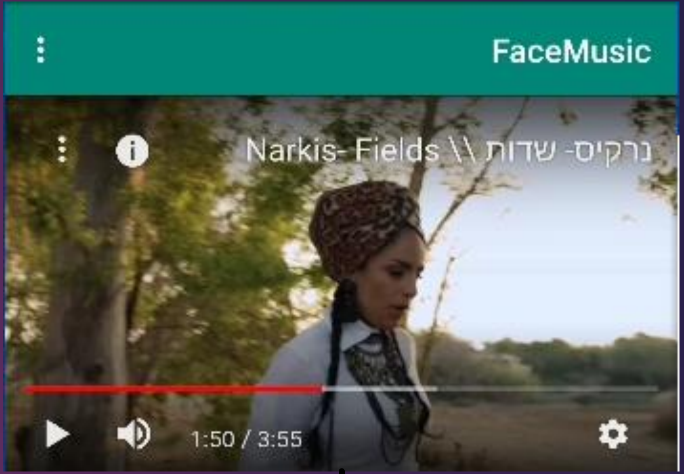
+972 54-415-1000 חו״ל

17:42 FaceMusic
**FaceMusic**
Here is a song for your mood - happy

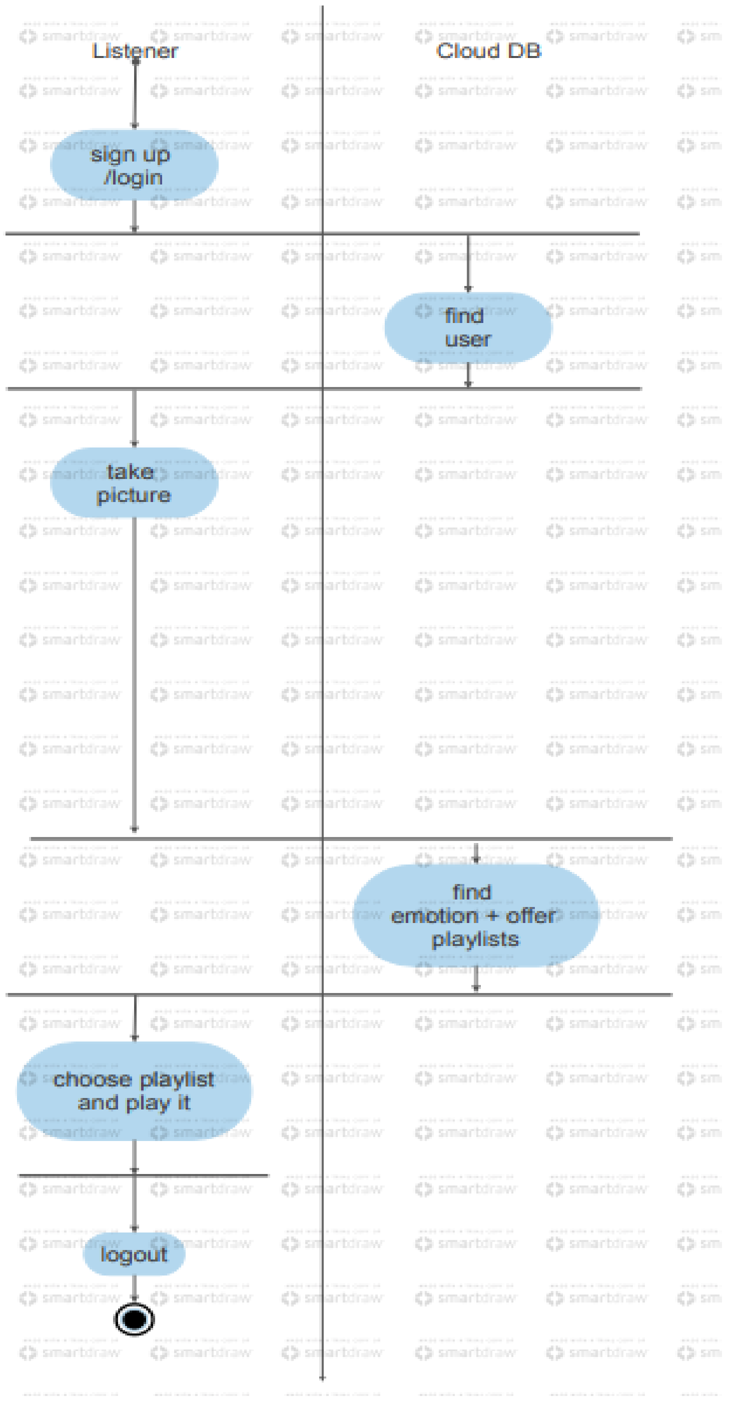
חפש עדכוני מערכת
חפש עדכוני מערכת
עדכון תוכנה זמין

A notification is sent to the user's phone with the relative emotion.

## Plays the song based on the feeling

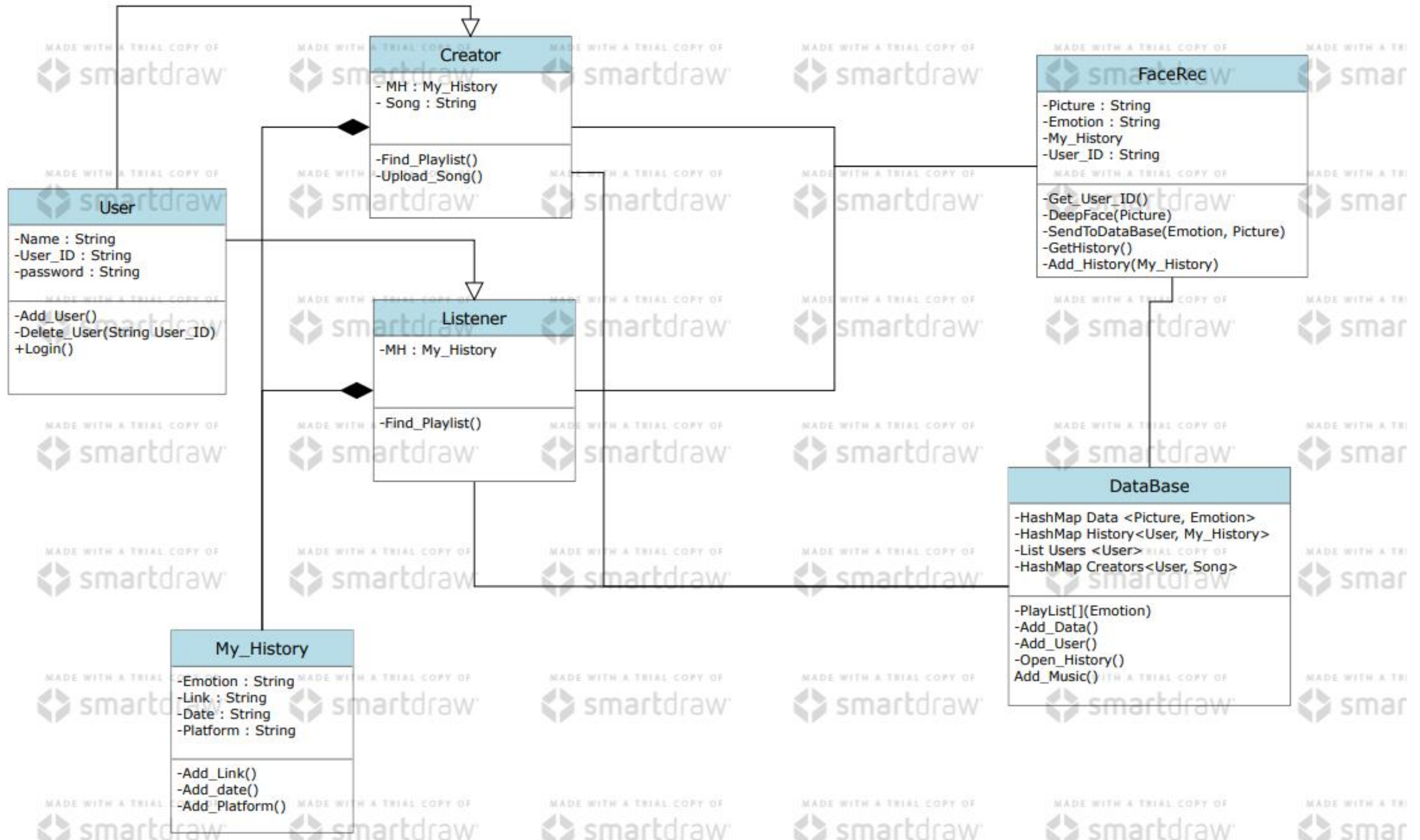


Activates a random song from the Firebase Realtime Database based on the emotion detected from YouTube.



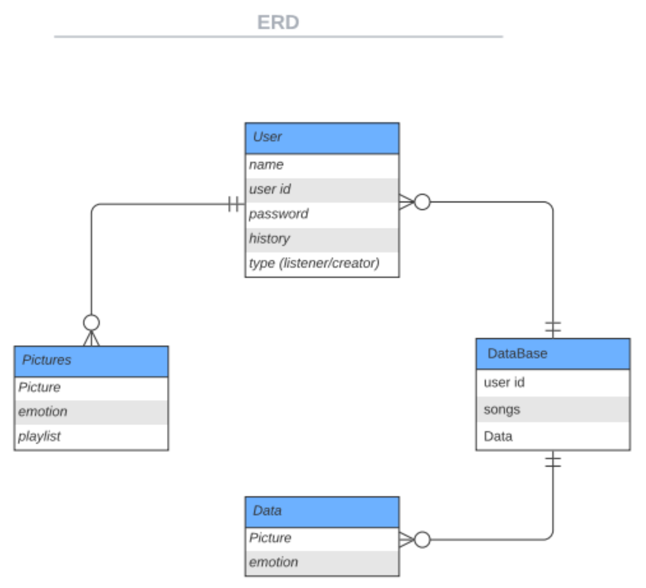
## Activity Diagram

# Class Diagram

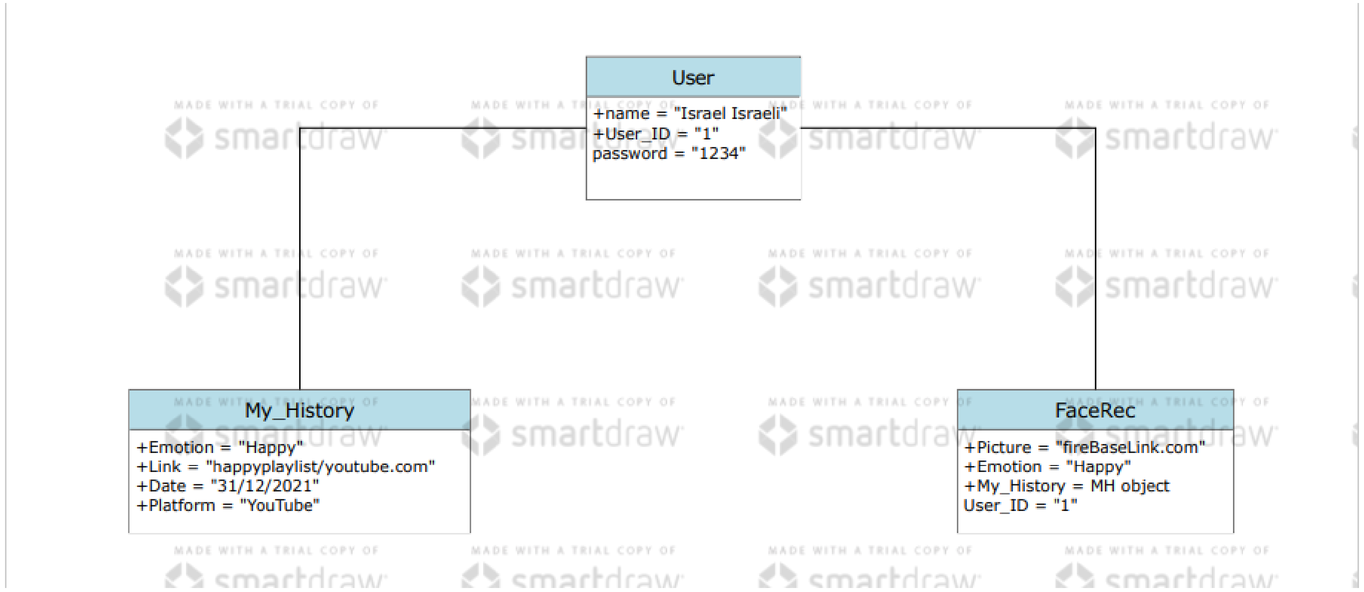


MADE WITH A TRIAL COPY OF smartdraw

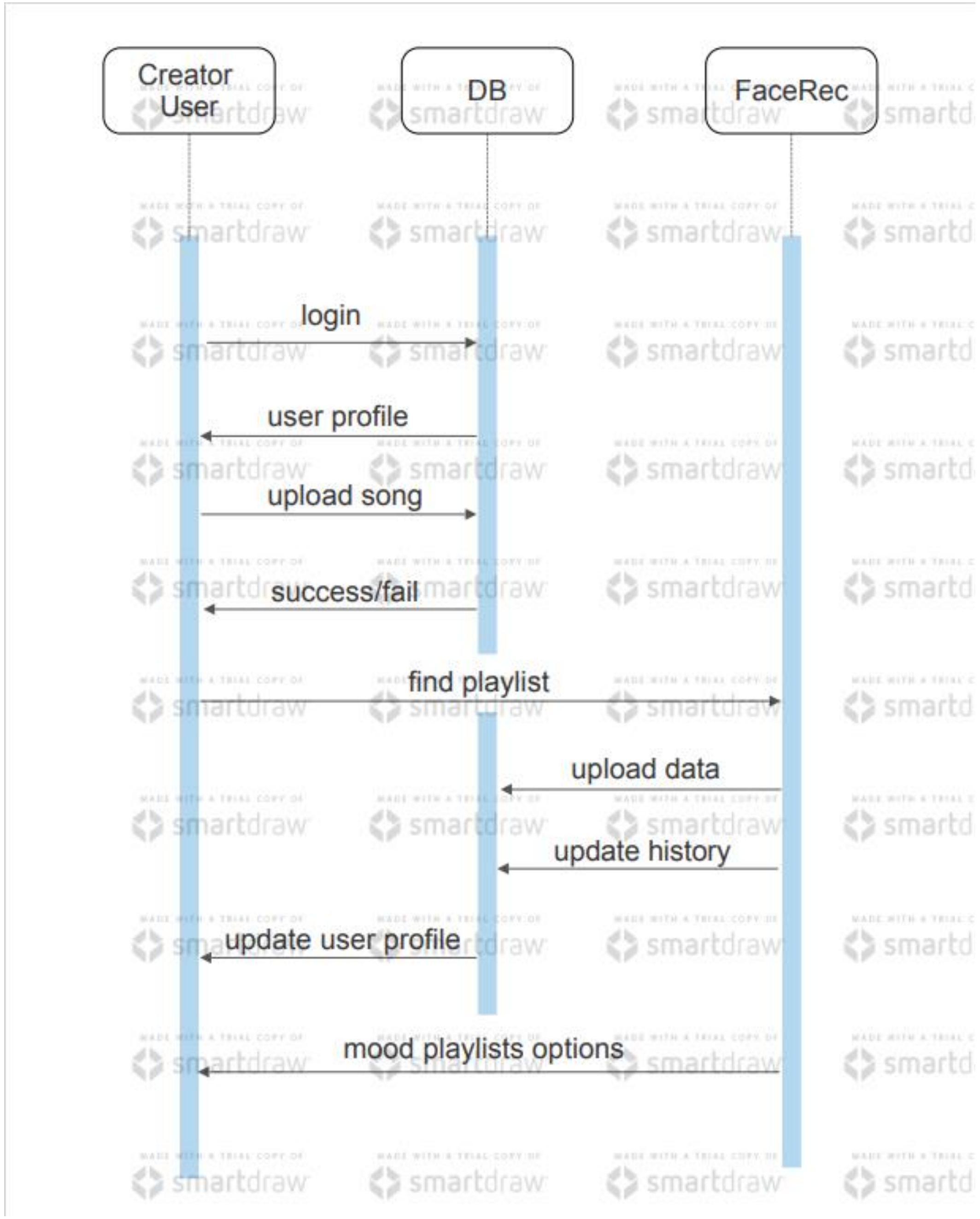
# ERD


ERD
User
name
user id
password
history
type (listener/creator)
Pictures
Picture
emotion
playlist
DataBase
user id
songs
Data
Data
Picture
emotion

# Object Diagram

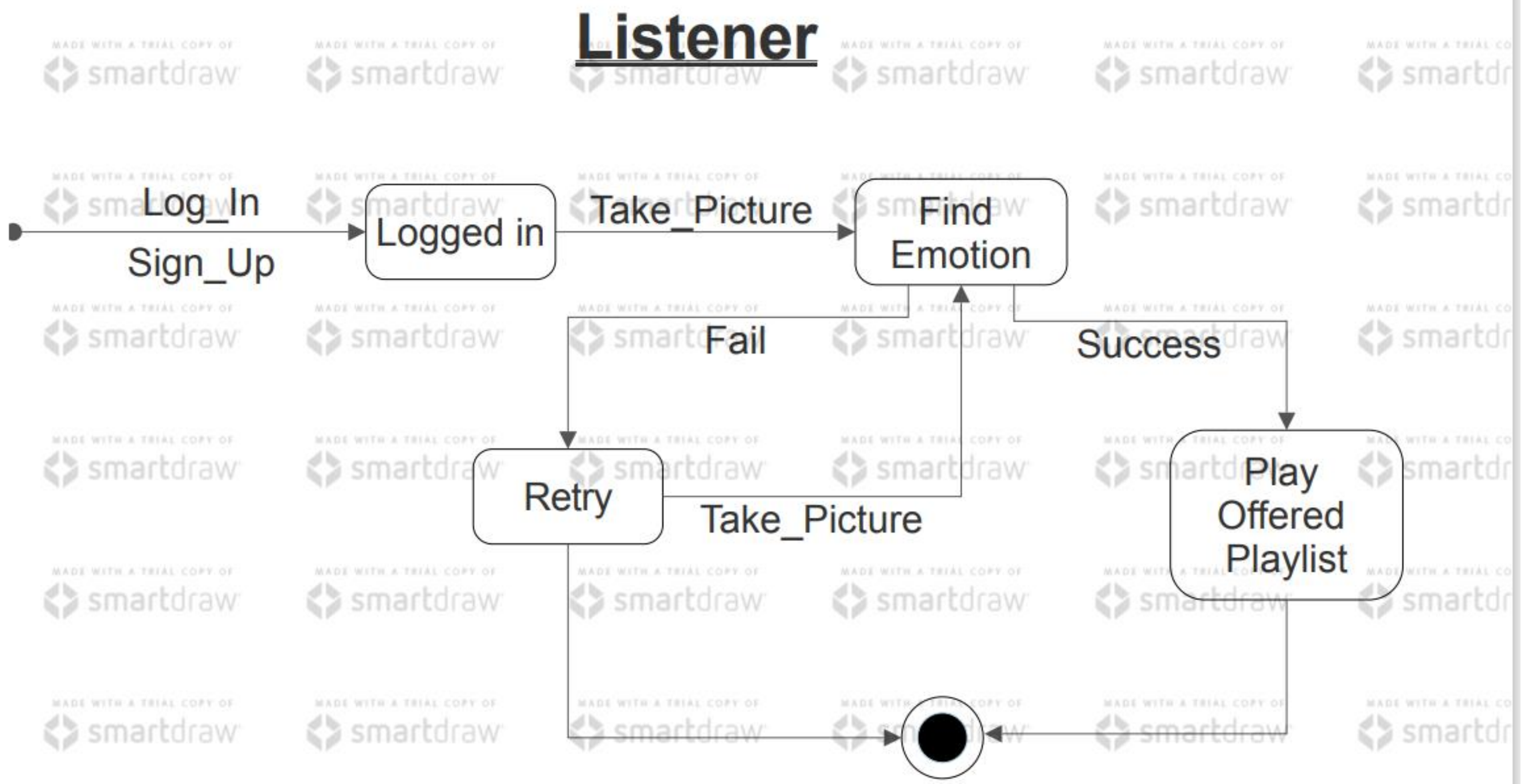


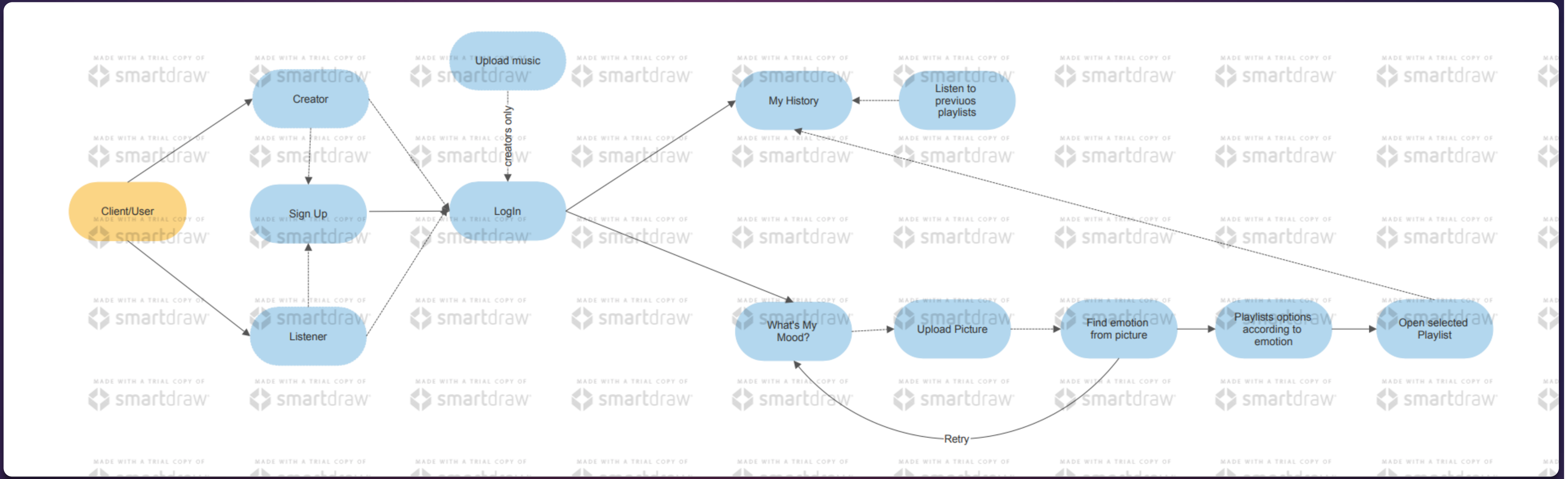
MADE WITH A TRIAL COPY OF smartdraw



## Sequence Diagram

# State Machine Diagram





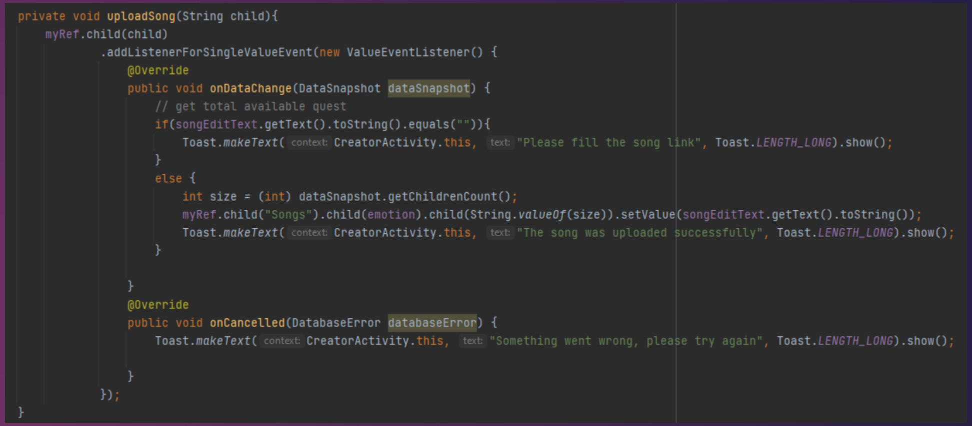
MADE WITH A TRIAL COPY OF smartdraw

## Use Case

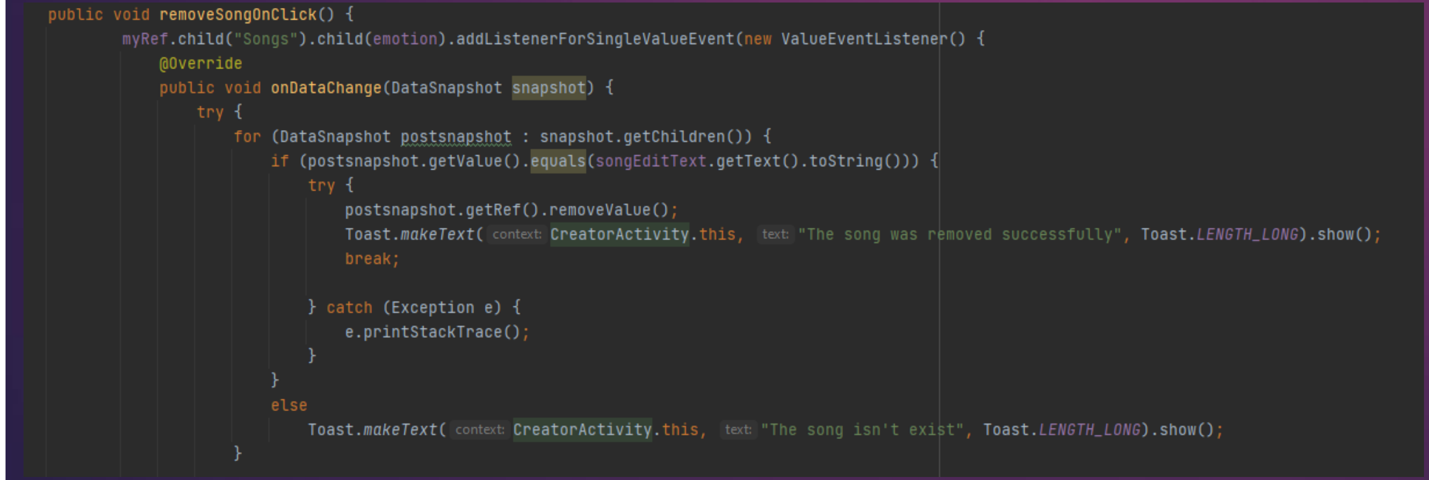
# Objects and Methods

## Creator Class

**Upload Song Method -** Gets the relative index indicating the location in which to upload the song. Use this function to upload the song to the Firebase Realtime database. Uploaded songs are associated with the relative emotion the creator entered.



```
private void uploadSong(String child){
    myRef.child(child)
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    // get total available quest
                    if(songEditText.getText().toString().equals("")){
                        Toast.makeText( context: CreatorActivity.this,  text: "Please fill the song link", Toast.LENGTH_LONG).show();
                    }
                    else {
                        int size = (int) dataSnapshot.getChildrenCount();
                        myRef.child("Songs").child(emotion).child(String.valueOf(size)).setValue(songEditText.getText().toString());
                        Toast.makeText( context: CreatorActivity.this,  text: "The song was uploaded successfully", Toast.LENGTH_LONG).show();
                    }

                }
                @Override
                public void onCancelled(DatabaseError databaseError) {
                    Toast.makeText( context: CreatorActivity.this,  text: "Something went wrong, please try again", Toast.LENGTH_LONG).show();

                }
            });
}
```

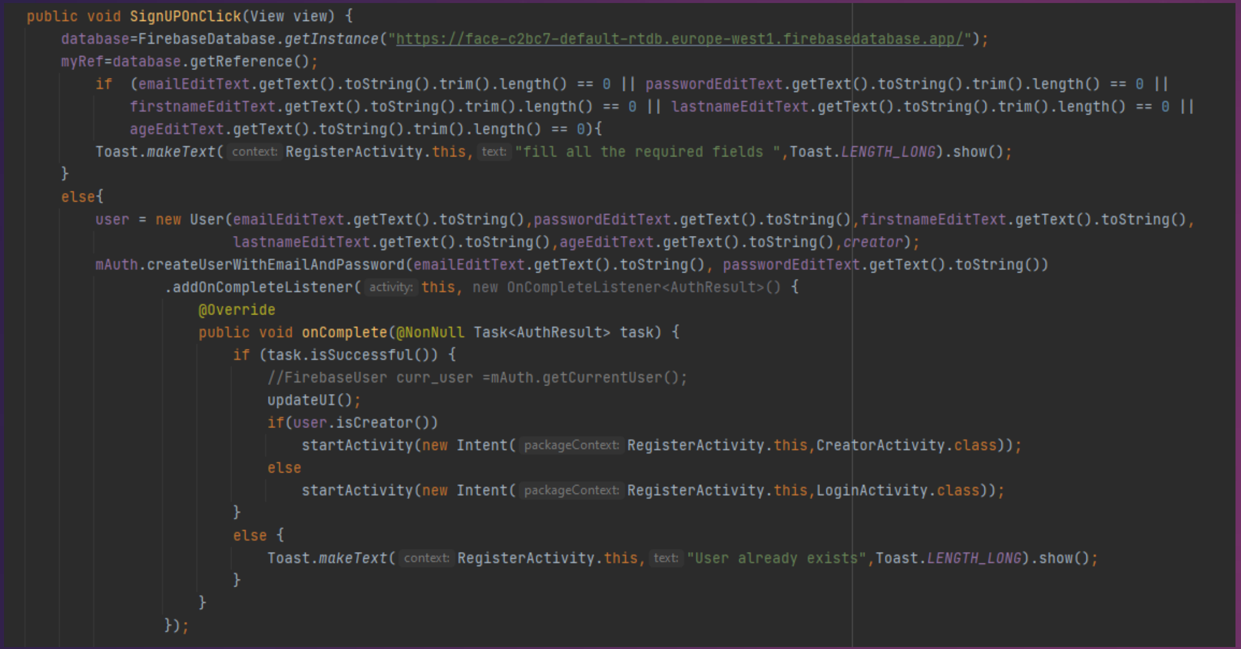


```
public void removeSongOnClick() {
        myRef.child("Songs").child(emotion).addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot snapshot) {
                try {
                    for (DataSnapshot postsnapshot : snapshot.getChildren()) {
                        if (postsnapshot.getValue().equals(songEditText.getText().toString())) {
                            try {
                                postsnapshot.getRef().removeValue();
                                Toast.makeText( context: CreatorActivity.this,  text: "The song was removed successfully", Toast.LENGTH_LONG).show();
                                break;

                            } catch (Exception e) {
                                e.printStackTrace();
                            }
                        }
                        else
                            Toast.makeText( context: CreatorActivity.this,  text: "The song isn't exist", Toast.LENGTH_LONG).show();
                    }
```

**Remove Song Method -** Moves over all the songs and checks if the relative song is present, if it is, this function removes that song from the Firebase Realtime database, otherwise it sends a message saying that the song does not exist

# Objects and Methods

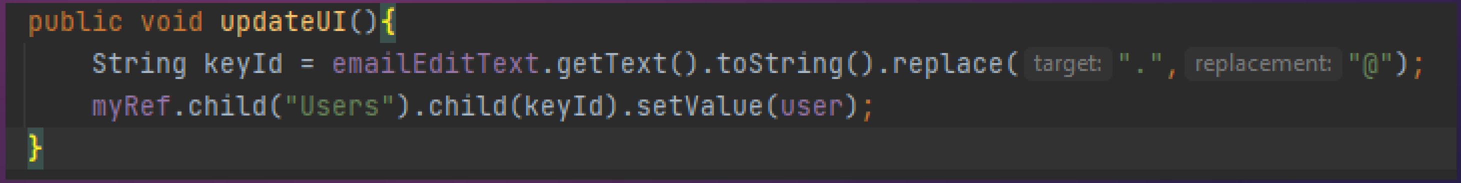
## Register Class



```java
public void SignUPOnClick(View view) {
    database=FirebaseDatabase.getInstance("https://face-c2bc7-default-rtdb.europe-west1.firebasedatabase.app/");
    myRef=database.getReference();
        if  (emailEditText.getText().toString().trim().length() == 0 || passwordEditText.getText().toString().trim().length() == 0 ||
            firstnameEditText.getText().toString().trim().length() == 0 || lastnameEditText.getText().toString().trim().length() == 0 ||
            ageEditText.getText().toString().trim().length() == 0){
        Toast.makeText( context: RegisterActivity.this, text: "fill all the required fields ",Toast.LENGTH_LONG).show();
    }
    else{
        user = new User(emailEditText.getText().toString(),passwordEditText.getText().toString(),firstnameEditText.getText().toString(),
                        lastnameEditText.getText().toString(),ageEditText.getText().toString(),creator);
        mAuth.createUserWithEmailAndPassword(emailEditText.getText().toString(), passwordEditText.getText().toString())
                .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            //FirebaseUser curr_user =mAuth.getCurrentUser();
                            updateUI();
                            if(user.isCreator())
                                startActivity(new Intent( packageContext: RegisterActivity.this,CreatorActivity.class));
                            else
                                startActivity(new Intent( packageContext: RegisterActivity.this,LoginActivity.class));
                        }
                        else {
                            Toast.makeText( context: RegisterActivity.this, text: "User already exists",Toast.LENGTH_LONG).show();
                        }
                    }
                });
```

**SingUpOnClick -** In this function, the user is entered into the Firebase Realtime database. The user's personal details are obtained from the details that the user enters within the application. If you don't fill out all the fields, you will be allerted

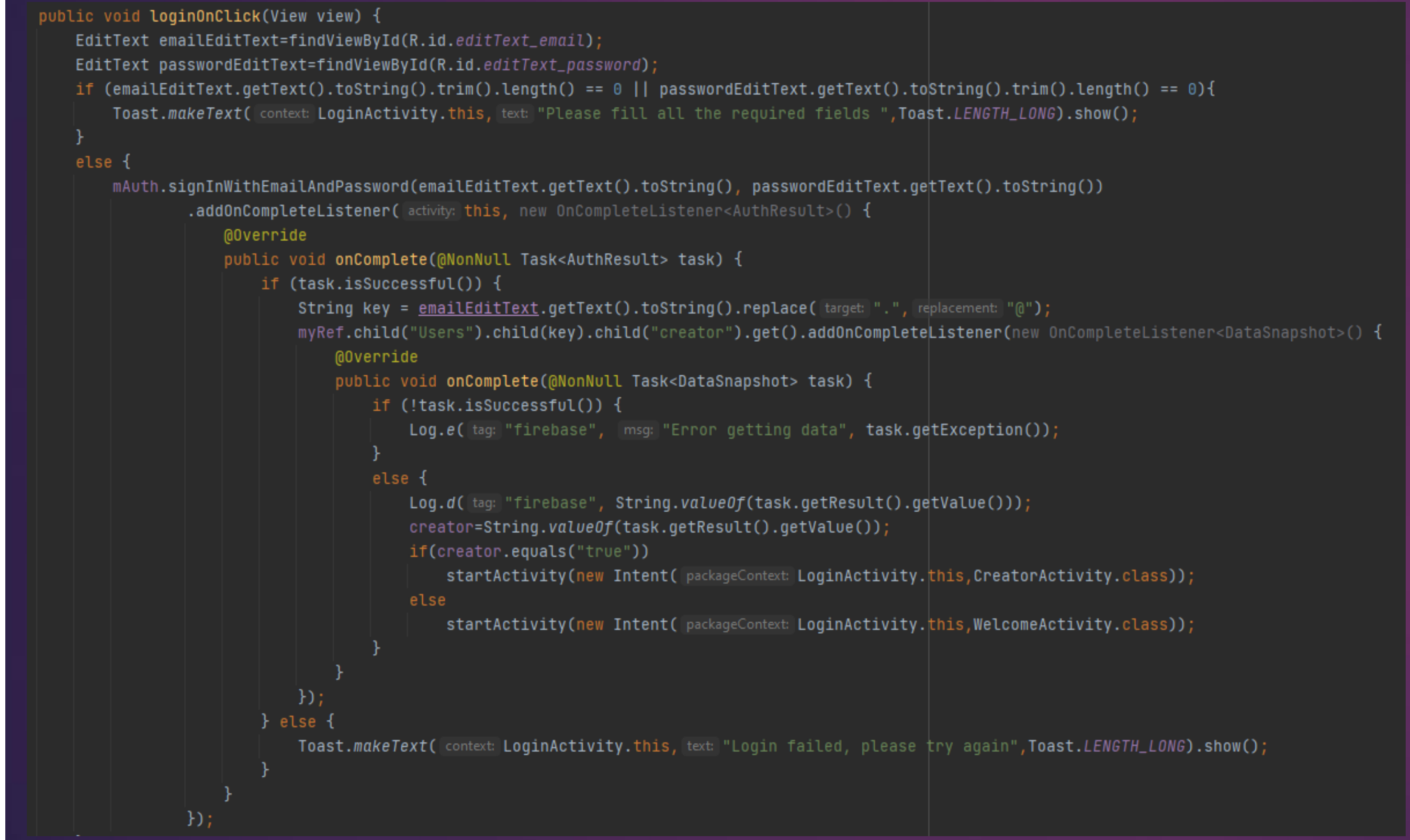
**UpdateUI -** this method push the user details to the firebase real time database



```java
public void updateUI(){
    String keyId = emailEditText.getText().toString().replace( target: ".", replacement: "@");
    myRef.child("Users").child(keyId).setValue(user);
}
```

# Objects and Methods

## Login Class

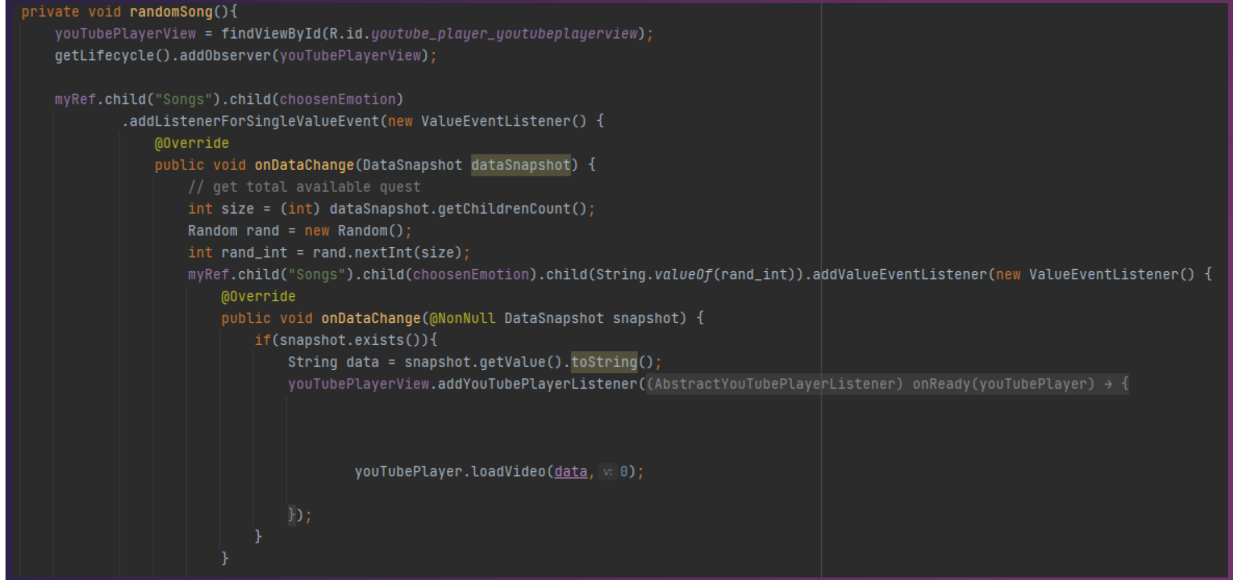


```
public void loginOnClick(View view) {
    EditText emailEditText=findViewById(R.id.editText_email);
    EditText passwordEditText=findViewById(R.id.editText_password);
    if (emailEditText.getText().toString().trim().length() == 0 || passwordEditText.getText().toString().trim().length() == 0){
        Toast.makeText( context: LoginActivity.this, text: "Please fill all the required fields ",Toast.LENGTH_LONG).show();
    }
    else {
        mAuth.signInWithEmailAndPassword(emailEditText.getText().toString(), passwordEditText.getText().toString())
                .addOnCompleteListener( activity: this, new OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult> task) {
                        if (task.isSuccessful()) {
                            String key = emailEditText.getText().toString().replace( target: ".", replacement: "@");
                            myRef.child("Users").child(key).child("creator").get().addOnCompleteListener(new OnCompleteListener<DataSnapshot>() {
                                @Override
                                public void onComplete(@NonNull Task<DataSnapshot> task) {
                                    if (!task.isSuccessful()) {
                                        Log.e( tag: "firebase", msg: "Error getting data", task.getException());
                                    }
                                    else {
                                        Log.d( tag: "firebase", String.valueOf(task.getResult().getValue()));
                                        creator=String.valueOf(task.getResult().getValue());
                                        if(creator.equals("true"))
                                            startActivity(new Intent( packageContext: LoginActivity.this,CreatorActivity.class));
                                        else
                                            startActivity(new Intent( packageContext: LoginActivity.this,WelcomeActivity.class));
                                    }
                                }
                            });
                        } else {
                            Toast.makeText( context: LoginActivity.this, text: "Login failed, please try again",Toast.LENGTH_LONG).show();
                        }
                    }
                });
```

1.LoginOnClick - Verification of the user in the Firebase Realtime database, by checking the email and password with which he registered for the application. If you don't fill out all the fields, you will be allerted

# Objects and Methods

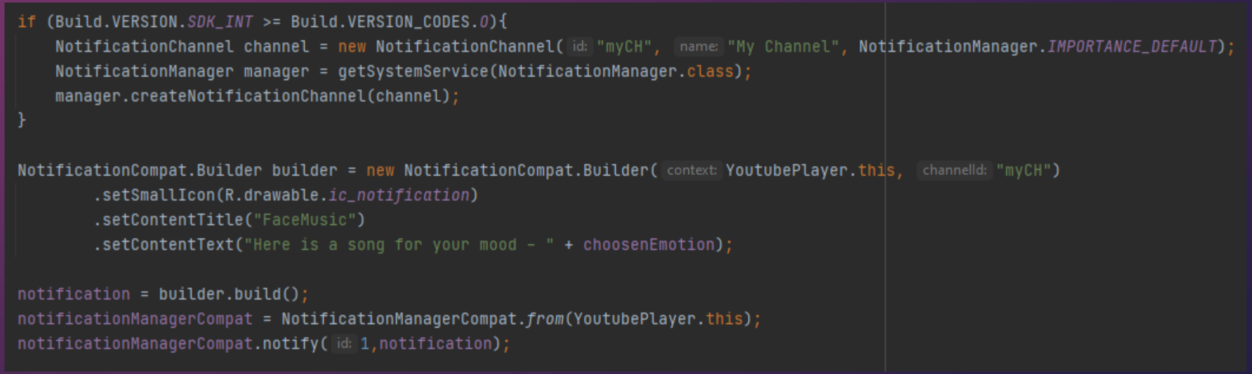
## YouTube Player Class



```
private void randomSong(){
    youTubePlayerView = findViewById(R.id.youtube_player_youtubeplayerview);
    getLifecycle().addObserver(youTubePlayerView);

    myRef.child("Songs").child(choosenEmotion)
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    // get total available quest
                    int size = (int) dataSnapshot.getChildrenCount();
                    Random rand = new Random();
                    int rand_int = rand.nextInt(size);
                    myRef.child("Songs").child(choosenEmotion).child(String.valueOf(rand_int)).addValueEventListener(new ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot snapshot) {
                            if(snapshot.exists()){
                                String data = snapshot.getValue().toString();
                                youTubePlayerView.addYouTubePlayerListener((AbstractYouTubePlayerListener) onReady(youTubePlayer) → {



                                        youTubePlayer.loadVideo(data, v: 0);

                                });
                            }
                        }
```

RandomSong - this function gets access to the Firebase Realtime database and then runs a song on YouTube randomly from a list of songs that are categorized for the same emotion by the creators.

Notification - after the emotion detection the user will be notify the chosen emotion.



```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O){
    NotificationChannel channel = new NotificationChannel( id: "myCH",  name: "My Channel", NotificationManager.IMPORTANCE_DEFAULT);
    NotificationManager manager = getSystemService(NotificationManager.class);
    manager.createNotificationChannel(channel);
}

NotificationCompat.Builder builder = new NotificationCompat.Builder( context: YoutubePlayer.this,  channelId: "myCH")
        .setSmallIcon(R.drawable.ic_notification)
        .setContentTitle("FaceMusic")
        .setContentText("Here is a song for your mood - " + choosenEmotion);

notification = builder.build();
notificationManagerCompat = NotificationManagerCompat.from(YoutubePlayer.this);
notificationManagerCompat.notify( id: 1,notification);
```

## Objects and Methods

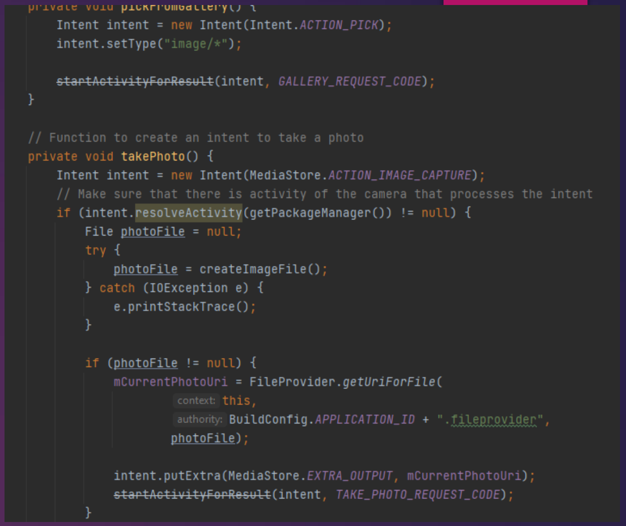
### FaceRec Class

Open source code. The dataset is trained with a Convolutional Neural Network model.
It contains 46,614 images.
The model provide output that consist of probabilities for each class:

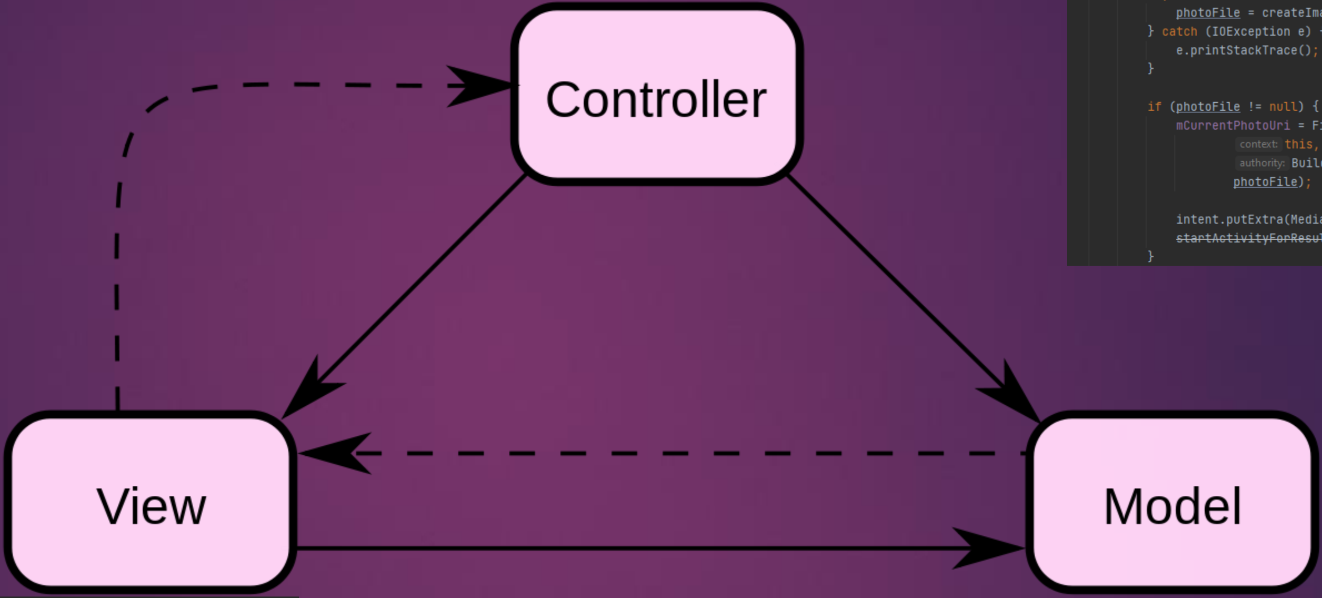
- angry
- disgust
- fear
- happy
- neutral
- sad
- surprise

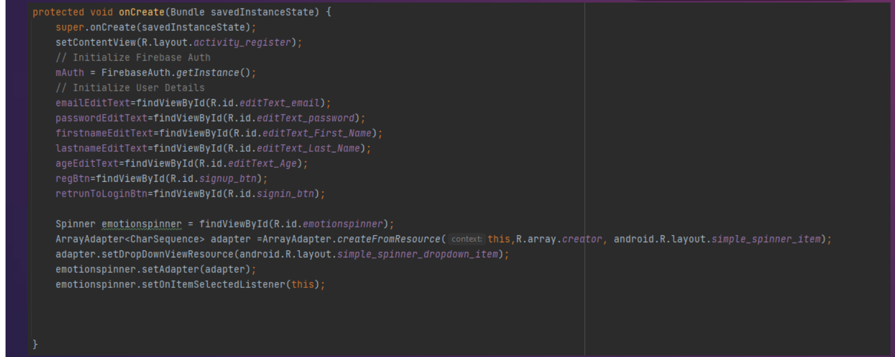
# Models

## MVC

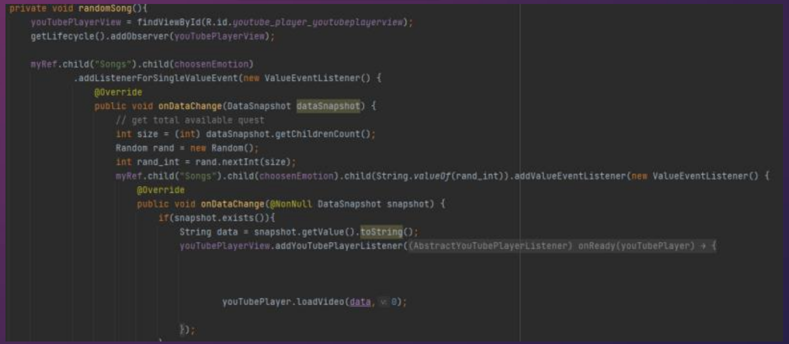


```java
private void pickFromGallery() {
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType("image/*");

    ~~startActivityForResult~~(intent, GALLERY_REQUEST_CODE);
}

// Function to create an intent to take a photo
private void takePhoto() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    // Make sure that there is activity of the camera that processes the intent
    if (intent.resolveActivity(getPackageManager()) != null) {
        File photoFile = null;
        try {
            photoFile = createImageFile();
        } catch (IOException e) {
            e.printStackTrace();
        }

        if (photoFile != null) {
            mCurrentPhotoUri = FileProvider.getUriForFile(
                    context: this,
                    authority: BuildConfig.APPLICATION_ID + ".fileprovider",
                    photoFile);

            intent.putExtra(MediaStore.EXTRA_OUTPUT, mCurrentPhotoUri);
            ~~startActivityForResult~~(intent, TAKE_PHOTO_REQUEST_CODE);
        }
```





```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_register);
    // Initialize Firebase Auth
    mAuth = FirebaseAuth.getInstance();
    // Initialize User Details
    emailEditText=findViewById(R.id.editText_email);
    passwordEditText=findViewById(R.id.editText_password);
    firstnameEditText=findViewById(R.id.editText_First_Name);
    lastnameEditText=findViewById(R.id.editText_Last_Name);
    ageEditText=findViewById(R.id.editText_Age);
    regBtn=findViewById(R.id.signup_btn);
    retrunToLoginBtn=findViewById(R.id.signin_btn);

    Spinner emotionspinner = findViewById(R.id.emotionspinner);
    ArrayAdapter<CharSequence> adapter =ArrayAdapter.createFromResource( context: this,R.array.creator, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    emotionspinner.setAdapter(adapter);
    emotionspinner.setOnItemSelectedListener(this);



}
```



```java
private void randomSong(){
    youTubePlayerView = findViewById(R.id.youtube_player_youtubeplayerview);
    getLifecycle().addObserver(youTubePlayerView);

    myRef.child("Songs").child(choosenEmotion)
            .addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    // get total available quest
                    int size = (int) dataSnapshot.getChildrenCount();
                    Random rand = new Random();
                    int rand_int = rand.nextInt(size);
                    myRef.child("Songs").child(choosenEmotion).child(String.valueOf(rand_int)).addValueEventListener(new ValueEventListener() {
                        @Override
                        public void onDataChange(@NonNull DataSnapshot snapshot) {
                            if(snapshot.exists()){
                                String data = snapshot.getValue().toString();
                                youTubePlayerView.addYouTubePlayerListener((AbstractYouTubePlayerListener) onReady(youTubePlayer) → {



                                        youTubePlayer.loadVideo(data, v: 0);

                                });
```

## Summery

Our project is developed using Android-Studio, using YouTube player and Face Recognition open source code.
Our Project allows:

- Users to enjoy music based on the emotion they display.
- Creator users to upload / remove songs from the database

Each emotion has a different number of songs, when the face recognition detect the right emotion a song will be played randomly from the list of songs in FireBase real time database.

There are 2 types of users:

- Client user
- Creator user

## Gaps

- Users have direct access to their own personal history - the last detected emotions and songs are saved and displayed on a dedicated screen.
- When the app recognizes a face and selects a song, it allows the user to rate the selection and recognition on a scale of 1 to 5 for the purpose of improving the app.
- For the creator user, payment will be required in order to publish his songs (by classifying them according to emotions). In addition, creator user will be required to upload appropriate and proper content. (The creator will have to sign an agreement and go through a number of verification steps before receiving the appropriate permissions) .
- Contact selected applications such as: Spotify, Apple Music for reviews and business contact.