

MOBILE WEB APP DEVELOPMENT

CP5UA930

Module Leader: Dr Cain Kazimoglu

Module Tutor: Ms. Sivasankari Sivakumar

Video demonstration: <https://youtu.be/8ICRS52ZR-I>

Contents

Introduction	1
Wireframing and User Interface	2
Back End Development	5
Going above and beyond	9
Reflection	9
Conclusion and future work.....	9
References.....	10
Appendix/Code	11

Introduction

Application name: MellowSphere

Logo:



MellowSphere is a music streaming application designed to provide users with an immersive experience in listening and enjoying music online. With a library of songs spanning various genres, artists, the app aims to relate to the diverse musical preferences of its users.

Key Features of MellowSphere:

1. **User-Friendly Interface:**
MellowSphere features an intuitive and user-friendly interface, making navigation through the app a breeze. Users can easily search for their favorite albums or tracks effortlessly.
2. **Offline Listening:**
Users can download their favorite songs for offline listening, enabling them to enjoy music even in areas with limited or no internet connectivity.
3. **Favourites playlist:**
MellowSphere enhances the social aspect by displaying the number of users that have favoured tracks.

Creating an account is not needed and anyone with this application can enjoy **most** of its features. However, favouriting and downloading tracks do require the creation of one.

Users can register with their username, email and password. After creating their account, a verification mail will be sent to their email address in order to check whether the user has registered with a valid email that he/she can use. After verifying the user can login with their verified account with the username and password.

In any case if the user happens to forget the password to their account, they can recover their account by entering their email after tapping on the "Forgot password?" text in the login page. If an account with the entered email does not exist, then the app will notify the user of the invalid email, otherwise it will send a password reset link.

I have used Figma to create wireframes, online sources (AI logo maker and svgrepo.com) to obtain the logo for this app, Firebase for database and Android studio for front and back end development.

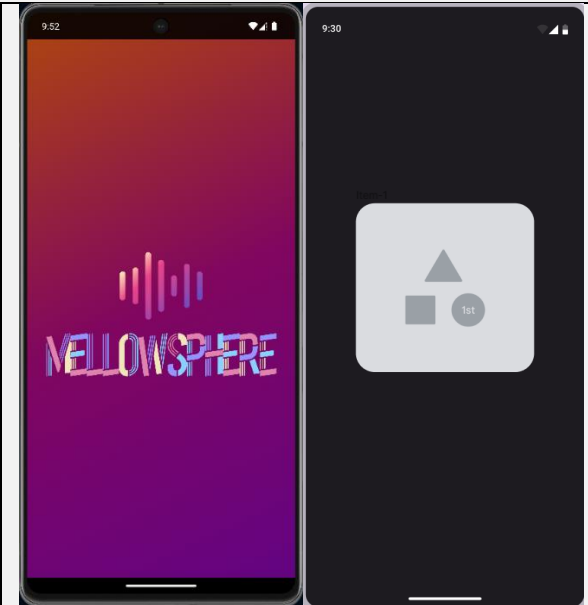
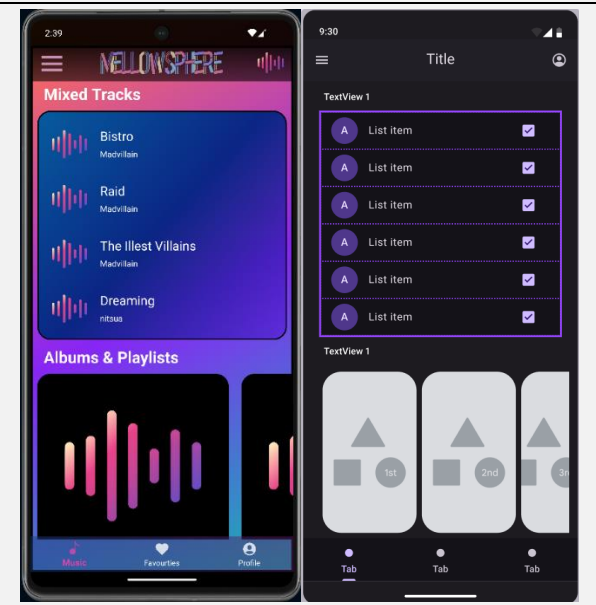
In Firebase I have used User Authentication database for managing accounts, firestore to store simple data of tracks, albums, accounts (username and email), and Cloud Storage databases to store mp3 audio files to be retrieved.

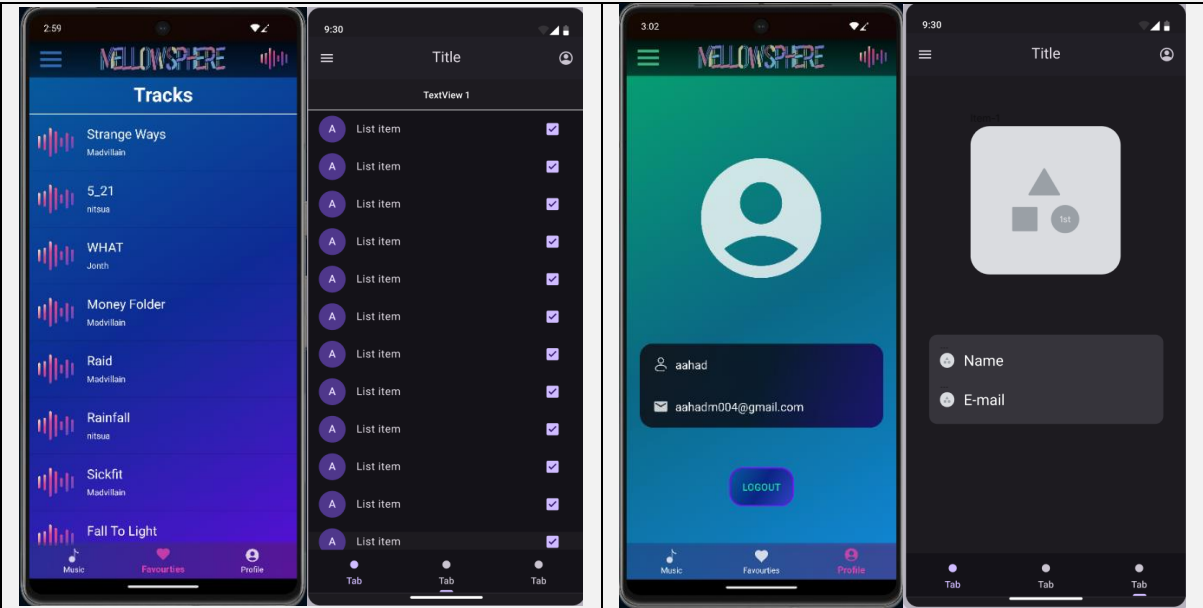
Report structure

1. **UI design and development:** This section describes the design and implementation of the UI and the reasons for them.
2. **Back End development:** Includes key points of the Java code structure.
3. **Above and beyond:** What I implemented beyond the general requirements.
4. **Reflection:** Discussion about any possible app bugs, what I have learned, etc.
5. **Conclusion:** final words, critics and future changes.
6. **References:** Resources used for the app development.
7. **Appendix/Code**

Wireframing and User Interface

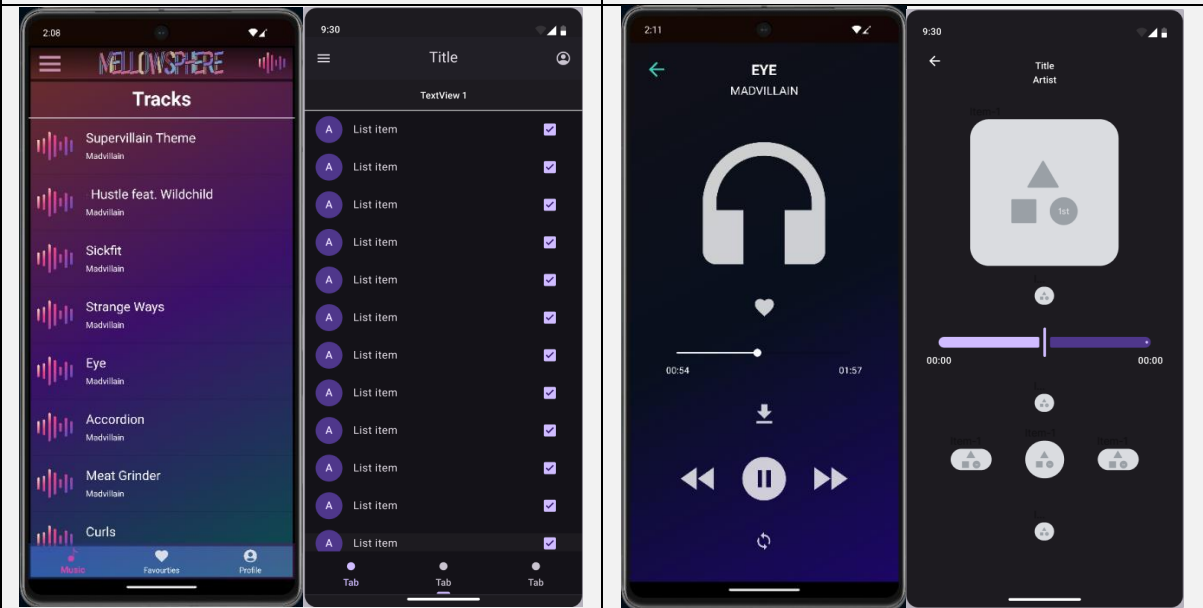
As mentioned above, I have used Figma to create the wireframes. To be more specific, I have used the material 3 design kit. (Figma, 2023)

	
<p>The splash activity appears for 2.4 milliseconds as the app starts</p>	<p>The first page that is displayed after the splash screen. It shows the playlist of all songs shuffled and all the albums the tracks come from</p>



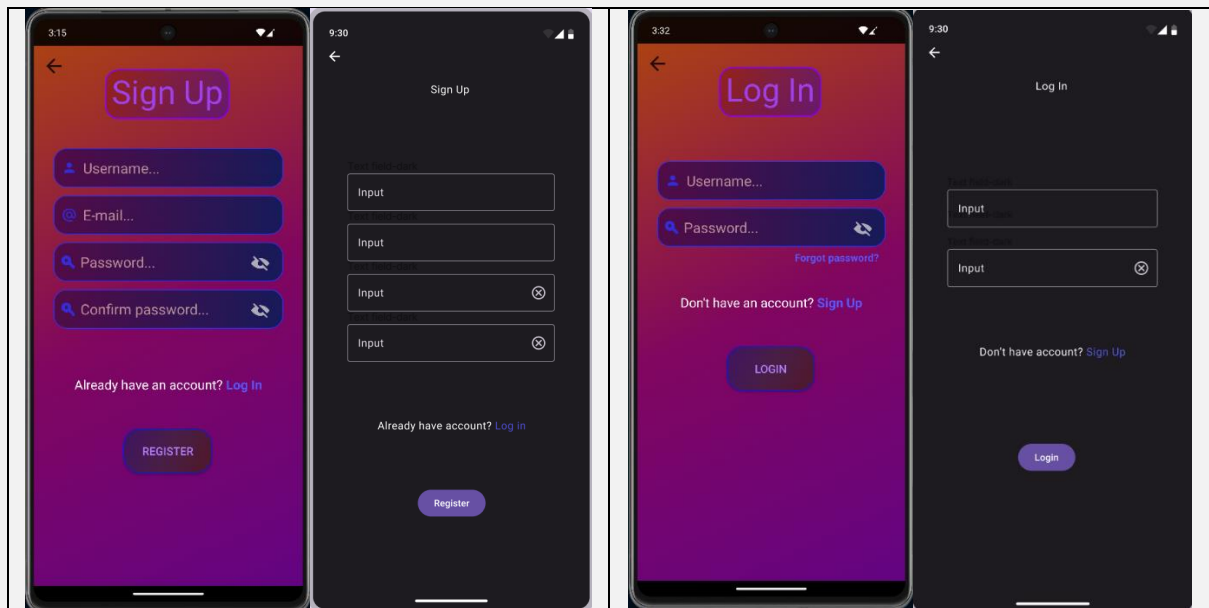
This tab displays all the users' favourite songs

This tab displays the users' name and email. It also shows an option to logout if user is logged in.



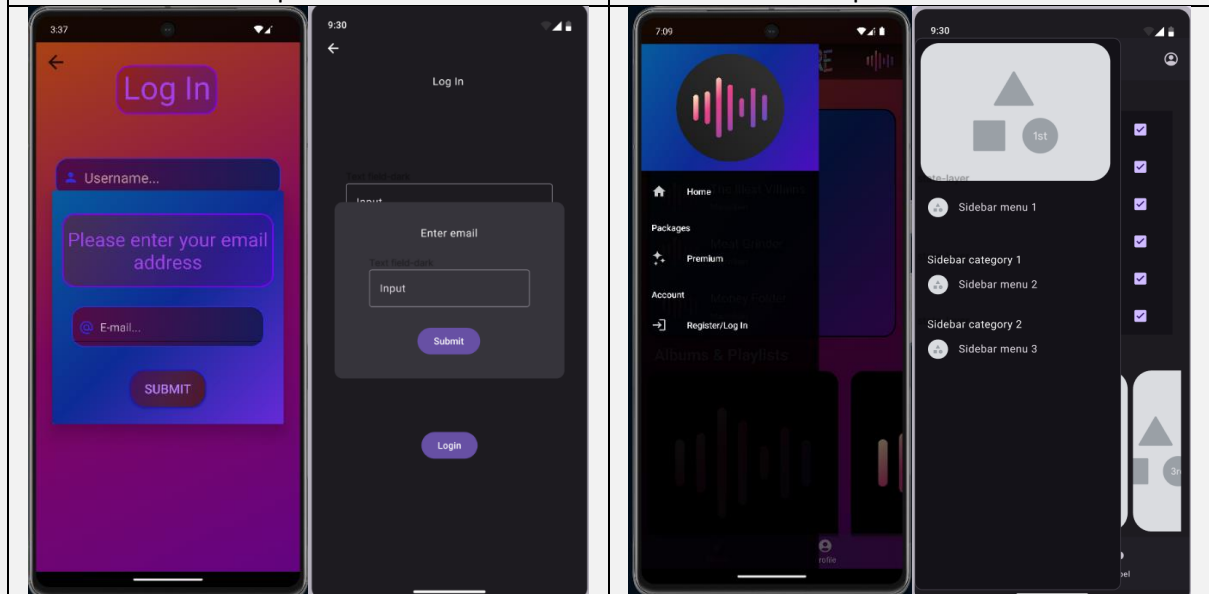
This page is shown when you tap on any one of the album. It contains all track from that specific album

This page is displayed when any of the tracks are selected. The user can control the audio, add to favourite, and download from here



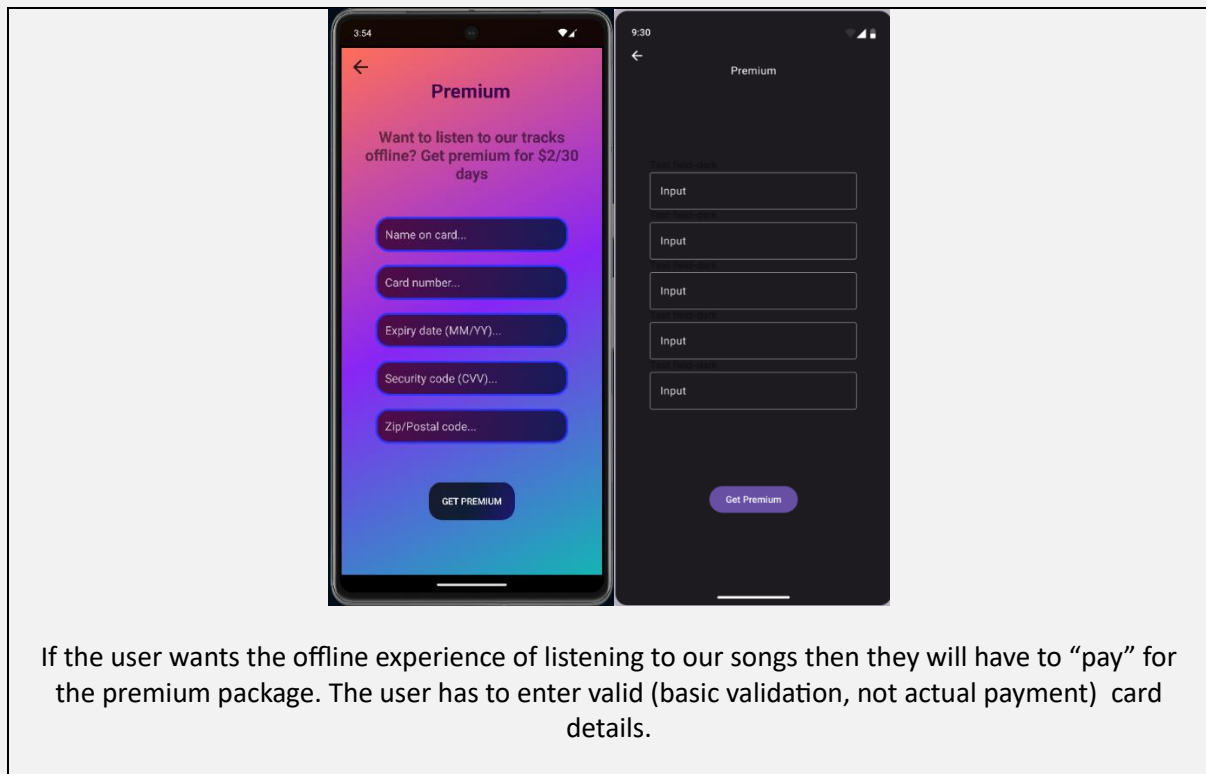
Users can create account with username, email and password

Users have to log in using username and password



The forgot password activity allows users to recover their accounts using their registered emails

The sidebar menu has three options: Home, Premium, Register/Login if the user is logged out or Logout if the user is logged in.



Back End Development

The Java code includes providing functionalities to the activities and reading/writing data from/to firebase. Validations were added for the activities dealing with user input such as login/sign up page and registering for premium page.

Login, registration, password recovery,

The main goal of the java code for any of these sections was to filter out the invalid input from the user in any of the text fields. For example, the username field has a limit of 3-26 characters that the user could enter. Another example is that in the premium activity the user could only input numbers for most of the text fields.

I have also implemented a way to use username instead of email for logging in. Once the user creates and verifies their email the username and password are stored together in the firebase firestore. Every time the user logs in the application takes the username and uses a query to search for the document that has the matching username in the “users” collection. Since the email and username are stored in pairs in each document, the email is retrieved and is then passed to the `signInWithEmailAndPassword()` function of the `FirebaseAuth` class. (SimplifyMSP, 2020)

Here is a simplified algorithm for how I implemented the email address validation, i.e., checking if the email entered is one that exists or not. If it does exist, it also checks whether the email belongs to the user creating the account.

Begin:

-> User creates account

-> Username, email and the exact time in milliseconds are stored in a firebase firestore collection called "users".

-> A new document is created and its identifier is set as the unique identifier (UID) of the user instance of FirebaseAuth which is related to the created account.

-> The application sends a verification mail to the newly registered email. The app also notifies the user that they should verify within 3 minutes

-> If the user tries logging in without verifying, the app temporarily signs them in.

-> Now, because the user is signed in, the application is able to retrieve the UID of the user instance of FirebaseAuth.

-> Using this UID, the application finds the document in the "users" collection in firestore and checks if the email is verified or not and checks how much time has passed since the account has been created.

-> If the time passed is more than 3 minutes, the account gets deleted

-> Depending on these conditions, the user is either able to successfully sign in or the account gets deleted.

-> Case 1: If the user has registered with a fake email that doesn't exist at all, the newly created account gets deleted eventually.

-> Case 2: If the user registers with a valid email but does not have access to the email, the account is still not verifiable and gets deleted eventually

End

The above discussed section of code to me felt the most challenging to come up with.

Homepage, Album tracks, Mixed tracks.

Displaying the tracks required retrieving the list of tracks in firebase firestore. The tracks would be displayed using a recycler view or a list view. The mixed track playlist and the albums list in the homepage uses recycler view while the album tracks and the favourites page uses listview.

Implementing the favourites feature for the recycler views seemed to be challenging. The favourites feature needed to display the number of users who have liked a specific track.

The above challenge was solved using ArrayLists and firestore.

```
private void retrieveFavouriteTracksFromAlbum() {
    if (user != null) {
        db.collection("alltracks").whereArrayContains("favouritedby",
            user.getId()).get()
            .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>()
            {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        List<AudioTrackModel> tempList = new ArrayList<>();

                        for (QueryDocumentSnapshot document : task.getResult())
```



```

        String albumName = document.getString("album");
        String artistName = document.getString("artist");
        String genre = document.getString("genre");
        String title = document.getString("title");

        // Convert Firestore document to
AlbumModelForRecycler
        AudioTrackModel audio = new AudioTrackModel(title,
artistName, genre, albumName);

        tempList.add(audio);

    }
    if (!(favouritesList == tempList)) {
        favouritesList = tempList;
    }
}
    }
}
    });
}
}

```

I have first constructed the logic in such a way that whenever the users tap on the favourite button, their FirebaseUser UID gets appended to the ArrayList stored in the field “favouritedby” within the document of “alltracks” collection.

Now in order to keep checking if the user has favourited the track the above function runs a query on the “alltracks” collection where the UID is searched for in the documents.

As for displaying the number of users that have liked a specific track the size of the ArrayList is returned.

Premium

Since the application is a music player/streamer, it would make more sense to have a premium subscription offer instead of checkout. The users, if they want to experience music offline, would have to subscribe for \$2 every 30 days to obtain that feature. If the user taps on the download button while they are not a premium member, the app redirects them to the premium page.

The application keeps track of whether the user is premium by retrieving the value of the “isPremium” field in the “users” collection.

Audio player

The main part of the application. This section involves two classes, AudioPlayer.java and AudioPlayerService.java

AudioPlayer.java:

What it does:

- Manages the user interface (UI) of the music player.
- Handles user interactions like play, pause, next, previous, etc.
- Displays information about the currently playing track.

How it works:

- When the app starts, it initializes the `AudioPlayerService`.
- Communicates with the `AudioPlayerService` to control playback (play, pause, seek, etc.).
- Updates the UI based on the playback status and track information.
- Manages features like favoriting tracks and downloading for offline use.

Noteworthy Methods:

- `initPlaybackService()`: Initializes the `AudioPlayerService` to handle music playback.
- `playAudio()`: Communicates with the service to start playing a specific audio file.
- `downloadAudio()`: Downloads an audio file from the internet.
- `updateSeekBar`: Periodically updates the seek bar to reflect the playback progress.
-

AudioPlayerService.java:

What it does:

- Handles the background tasks of playing audio.
- Manages the `MediaPlayer` object for actual audio playback.
- Ensures that audio playback continues even when the app is in the background.
- Creates a foreground service to prevent the system from killing it.

How it works:

- Created as a service to run in the background independently of the UI.
- Uses a `MediaPlayer` object to play audio from a given URL.
- Provides methods for the `AudioPlayer` class to control playback (play, pause, seek, etc.).
- Creates a notification to keep the service running in the foreground.

Noteworthy Methods:

- `onCreate()`: Initializes the `MediaPlayer` and sets up event listeners.
- `onStartCommand()`: Handles the start command for the service.
- `playAudio()`: Loads an audio file and starts playback.
- `pauseAudio()`, `resumeAudio()`: Pauses and resumes audio playback.
- `seek()`: Moves the playback position to a specified time.
- `onDestroy()`: Cleans up resources when the service is stopped.

Interaction:

- The `AudioPlayer` class interacts with the `AudioPlayerService` class through a service connection (`ServiceConnection`).
- The `AudioPlayer` class sends commands to the service (e.g., play, pause) using methods like `playAudio()`.
- The `AudioPlayerService` class handles the actual playback, manages the `MediaPlayer`, and notifies the `AudioPlayer` class about playback events.

In a nutshell, the `AudioPlayer` class is responsible for the user interface and user interactions, while the `AudioPlayerService` class takes care of playing audio in the background, ensuring smooth playback even when the app is not in the foreground. (Veliu, 2016)

Going above and beyond

The app doesn't have much to offer in terms of going above and beyond. I can only think of a few features off of the top of my head:

- Email address validation upon creating the app.
- Favourites feature
- Playing music offline.

Playing music offline

Since I haven't touched on this topic yet, I might as well do it here. The logic is as follows:

- > If the user selects a track to play, the application checks whether the song to be played is downloaded or not.
- > If the track was not downloaded and the user taps on the download button, the track URL retrieved from the firebase cloud storage will be used as the download link.
- > If the user has not yet granted the application the permission to access media and files, then the application will request one.
- > Else the download will begin and is possible by using DownloadManager objects. (Kumar, 2017)

Reflection

I am somewhat satisfied with the result of tedious development of this application. I wish I could've accomplished more but didn't due to me not having prior knowledge of databases and media player applications. I have, however, obtained somewhat of a knowledge/experience of making an professional applications that I always see on the play store.

If I wanted to improve upon my application, I would like to add these features:

- Build a notification where I can control the music from outside the app.
- Build a mini tab inside the app so I can control the music from anywhere inside the app.
- Make a separate downloads playlist and add a delete option
- More efficient and faster execution of the app overall

Conclusion and future work

I believe I have achieved the goal just enough for it to pass as a streaming service application. What I have learnt from the developing experience is that such projects/assessments require planning, and it is not just about programming. In the future I would like to come back to such a project and accomplish more than what I have achieved here but with a language and framework of my choice.

References

Figma, 2023. *Figma*. [Online]

Available at: <https://www.figma.com/community/file/1035203688168086460/material-3-design-kit>
[Accessed 19 January 2024].

Kumar, V., 2017. *Mobikul*. [Online]

Available at: <https://mobikul.com/downloading-audio-file-url/>
[Accessed 12 January 2024].

SimplifyMSP, 2020. *Reddit*. [Online]

Available at:
https://www.reddit.com/r/Firebase/comments/kgfhel/authenticate_user_with_username/
[Accessed 5 January 2024].

Veliu, V., 2016. *Sitepoint*. [Online]

Available at: <https://www.sitepoint.com/a-step-by-step-guide-to-building-an-android-audio-player-app/>
[Accessed 16 January 2024].

Appendix/Code

SplashActivity.java

```
package com.example.mellowsphere;

import androidx.appcompat.app.AppCompatActivity;

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

public class SplashActivity extends AppCompatActivity {

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

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent i = new Intent(getApplicationContext(),
MainActivity.class);
                startActivity(i);
                finish();
            }
        }, 2400);
    }
}
```

MainActivity.java

```
package com.example.mellowsphere;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;

import android.content.Intent;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.os.Handler;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.FrameLayout;
import android.widget.Toast;

import androidx.appcompat.widget.Toolbar;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
```

```

import androidx.lifecycle.ViewModelProvider;
;

import com.example.mellowsphere.databinding.ActivityMainBinding;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.navigation.NavigationView;
import com.google.firebase.FirebaseApp;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.google.firebase.firestore.WriteBatch;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener {
    List<AudioTrackModel> audioList = new ArrayList<>();
    List<AudioTrackModel> favouritesList = new ArrayList<>();
    private SharedViewModel sharedViewModel;

    ActivityMainBinding binding;
    FrameLayout frameLayout;

    DrawerLayout drawerLayout;
    NavigationView navigationView;
    Toolbar toolbar;
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    boolean isUserPremium;
    String ispremium;

    boolean stopthis = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        FirebaseApp.initializeApp(this);

        if (user != null) {

db.collection("users").document(user.getUid()).get().addOnCompleteListener(
new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot>
task) {

        if (task.isSuccessful()){
            DocumentSnapshot documentSnapshot =
task.getResult();

            if (documentSnapshot.exists()) {

```

```

        String isPremium =
documentSnapshot.getString("isPremium");

        if (isPremium.equals("f")) {
            isUserPremium = false;
            ispremium = "f";
        }
        else {
            isUserPremium = true;
            ispremium = "t";
        }
    }
}

});
}

super.onCreate(savedInstanceState);
binding = ActivityMainBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());

user = FirebaseAuth.getInstance().getCurrentUser();

if (user == null) {

bottomNavFragmentReplacement(AllTabItemFragment.newInstance(isUserPremium))
;

    }

    sharedViewModel = new
ViewModelProvider(this).get(SharedViewModel.class);

    frameLayout = findViewById(R.id.frameLayout);

    drawerLayout = findViewById(R.id.drawerlayout);
    toolbar = findViewById(R.id.titlebar);
    navigationView = findViewById(R.id.sidebar_nav);

    Handler handler = new Handler();
    int delayMillis = 1000; // set your desired delay in milliseconds

    // Get the Drawable from the resource ID
    Drawable expectedDrawable = ContextCompat.getDrawable(this,
R.drawable.custom_background_9);
    final Fragment[] currentFragment = {new Fragment()};

    Runnable runnable = new Runnable() {
        @Override
        public void run() {
            if (ispremium != null && stopthis) {

bottomNavFragmentReplacement(AllTabItemFragment.newInstance(isUserPremium))
;

                stopthis = false;
            }
        }
    };
}

```

```

        Menu menu = navigationView.getMenu();
        Menu menu2 = binding.bottomNavigationView.getMenu();

        currentFragment[0] =
getSupportFragmentManager().findFragmentById(R.id.frameLayout);

        if (isUserPremium && (drawerLayout.getBackground() !=
expectedDrawable) && currentFragment[0] instanceof AllTabItemFragment) {

drawerLayout.setBackgroundResource(R.drawable.custom_background_9);
        }

        if (user == null) {
            MenuItem menuItem = menu.findItem(R.id.logout);
            menuItem.setVisible(false);

            MenuItem menuItem1 = menu2.findItem(R.id.favourites);
            menuItem1.setVisible(false);
        }
        else {
            MenuItem menuItem2 = menu.findItem(R.id.register);
            menuItem2.setVisible(false);
        }

        if (isUserPremium) {
            MenuItem menuItem = menu.findItem(R.id.packages);
            menuItem.setVisible(false);
        }

        retrievFavouriteTracksFromAlbum();

        // Execute the code again after the specified delay
        handler.postDelayed(this, delayMillis);
    }
};

// Start executing the code with the initial delay
handler.postDelayed(runnable, delayMillis);

setSupportActionBar(toolbar);
getSupportActionBar().setTitle("");

ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
    this, drawerLayout, toolbar,
    R.string.navigation_drawer_open, R.string.navigation_drawer_close);

// Set custom width and height for the icon
int iconWidth = (int)
getResources().getDimension(R.dimen.menu_icon_width);
int iconHeight = (int)
getResources().getDimension(R.dimen.menu_icon_height);

toggle.getDrawerArrowDrawable().setBarLength(iconWidth);
toggle.getDrawerArrowDrawable().setBarThickness(iconWidth / 8f);
toggle.getDrawerArrowDrawable().setGapSize(iconHeight / 8f);

```



```

toggle.getDrawerArrowDrawable().setColorFilter(getResources().getColor(R.color.menuiconColor), PorterDuff.Mode.SRC_ATOP);

drawerLayout.addDrawerListener(toggle);
toggle.syncState();

navigationView.setNavigationItemSelectedListener(this);

binding.bottomNavigationView.setOnItemSelectedListener(item -> {
    if (item.getItemId() == R.id.music) {
        switchtoFragment(drawerLayout,
R.drawable.custom_background_1, toggle, R.color.menuiconColor,
AllTabItemFragment.newInstance(isUserPremium), true);
    }
    else if (item.getItemId() == R.id.favourites) {
        switchtoFragment(drawerLayout,
R.drawable.custom_background_4, toggle, R.color.favourites_icon_color,
FavouritesFragment.newInstance(favouritesList, isUserPremium), false);
    }
    else if (item.getItemId() == R.id.profile) {
        switchtoFragment(drawerLayout,
R.drawable.custom_background_5, toggle, R.color.menuiconColorProfile, new
ProfileFragment(), false);
    }
    return true;
});
navigationView.bringToFront();
}

@Override
public void onBackPressed() {
    // Close the drawer if it's open, otherwise, perform the default
    back button behavior
    if (drawerLayout.isDrawerOpen(GravityCompat.START)) {
        drawerLayout.closeDrawer(GravityCompat.START);
    } else {
        // Get the current fragment
        Fragment currentFragment =
getSupportFragmentManager().findFragmentById(R.id.frameLayout);

        // Check if it's the AudioListViewFragment
        if (!(currentFragment instanceof AllTabItemFragment)) {

            ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawerLayout, toolbar,
R.string.navigation_drawer_open, R.string.navigation_drawer_close);
            // Switch to another fragment or perform other actions
            binding.bottomNavigationView.setSelectedItemId(R.id.music);
            switchtoFragment(drawerLayout,
R.drawable.custom_background_1, toggle, R.color.menuiconColor,
AllTabItemFragment.newInstance(isUserPremium), true);
        } else {
            // Handle the back press as usual
            super.onBackPressed();
        }
    }
}

```

```

    }

    @Override
    public boolean onNavigationItemSelected(@NonNull MenuItem item) {
        drawerLayout.closeDrawer(GravityCompat.START);
        int itemID = item.getItemId();

        if (itemID == R.id.logout) {
            if (FirebaseAuth.getInstance().getCurrentUser() != null) {
                FirebaseAuth.getInstance().signOut();
                Intent intent = getBaseContext().getPackageManager()
                    .getLaunchIntentForPackage(getBaseContext().getPackageName());
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        }
        else if (itemID == R.id.register) {
            Intent i = new Intent(getApplicationContext(), Register.class);
            startActivity(i);
        }
        else if (itemID == R.id.premium) {
            if (user == null) {
                Toast.makeText(this, "Please create an account or log in",
                    Toast.LENGTH_SHORT).show();
                Intent i = new Intent(getApplicationContext(),
                    Register.class);
                startActivity(i);
            }
            else {
                Intent i = new Intent(this, PremiumRegister.class);
                startActivity(i);
            }
        }
        return true;
    }

    private void switchtoFragment(DrawerLayout drawerLayoutx, int
        backgroundID, ActionBarDrawerToggle toggle, int colorID, Fragment fragment,
        boolean home) {
        if (!home) {
            bottomNavFragmentReplacement(fragment);
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    drawerLayoutx.setBackgroundResource(backgroundID);

                    toggle.getDrawerArrowDrawable().setColorFilter(getResources().getColor(colorID), PorterDuff.Mode.SRC_ATOP);
                }
            }, 00);
        }
        else {
            drawerLayoutx.setBackgroundResource(backgroundID);

            toggle.getDrawerArrowDrawable().setColorFilter(getResources().getColor(colorID), PorterDuff.Mode.SRC_ATOP);
            new Handler().postDelayed(new Runnable() {

```

```

        @Override
        public void run() {
            bottomNavFragmentReplacement(fragment);
        }
    }, 00);
}

private void bottomNavFragmentReplacement(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();

    // Replace the current fragment with the new one
    fragmentTransaction.replace(R.id.frameLayout, fragment);

    // Commit the transaction
    fragmentTransaction.commit();
}

private void retrieveFavouriteTracksFromAlbum() {

    if (user != null) {
        db.collection("alltracks"). whereArrayContains("favouritedby",
user.getId()).get()
            .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                @Override
                public void onComplete(@NonNull Task<QuerySnapshot> task) {
                    if (task.isSuccessful()) {
                        List<AudioTrackModel> tempList = new ArrayList<>();

                        for (QueryDocumentSnapshot document :
task.getResult()) {
                            String albumName = document.getString("album");
                            String artistName =
document.getString("artist");
                            String genre = document.getString("genre");
                            String title = document.getString("title");

                            // Convert Firestore document to
AlbumModelForRecycler
                            AudioTrackModel audio = new
AudioTrackModel(title, artistName, genre, albumName);

                            tempList.add(audio);
                        }
                        if (!(favouritesList == tempList)) {
                            favouritesList = tempList;
                        }
                    }
                }
            });
    }

    public void traverseThroughCollection(String collection) {

```

```

db.collection(collection.toLowerCase()).get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {
            WriteBatch batch = db.batch();

            for (QueryDocumentSnapshot document : task.getResult())
{
            // For each document, update the "favouritedby"
field
            String documentId = document.getId();
            DocumentReference documentRef =
db.collection(collection.toLowerCase()).document(documentId);

            // Assuming you want to set "favouritedby" to some
specific value
            // You can modify this based on your requirements
Map<String, Object> updates = new HashMap<>();
            updates.put("favouritedby", null);

            batch.update(documentRef, updates);
        }

        // Commit the batched write
        batch.commit();
    }
});
}
}

```

AllTabItemFragment.java

```

package com.example.mellowsphere;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;

import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
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.Collections;
import java.util.List;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link AllTabItemFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class AllTabItemFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public AllTabItemFragment() {
        // Required empty public constructor
    }

    boolean isPremium;

    public static AllTabItemFragment newInstance(boolean isPremium) {
        AllTabItemFragment fragment = new AllTabItemFragment();
        Bundle args = new Bundle();
        args.putBoolean("isPremium", isPremium);
        fragment.setArguments(args);
        return fragment;
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment AllTabItemFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static AllTabItemFragment newInstance(String param1, String
param2) {
        AllTabItemFragment fragment = new AllTabItemFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {

```

```

        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
            isPremium = getArguments().getBoolean("isPremium", false);
        }
    }

    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    private CustomRecyclerView albumRecyclerView;
    private CustomRecyclerView allTrackRecyclerView;
    private SharedViewModel sharedViewModel;
    AlbumRecyclerViewAdapter albumAdapter;
    MixRecyclerViewAdapter mixRecyclerViewAdapter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.fragment_all_tab_item,
            container, false);

        allTrackRecyclerView = v.findViewById(R.id.mix);

        albumRecyclerView = v.findViewById(R.id.recyclerView);

        sharedViewModel = new
        ViewModelProvider(requireActivity()).get(SharedViewModel.class);
        sharedViewModel.getViewPager2().observe(getViewLifecycleOwner(),
        viewPager2 -> {
            if (viewPager2 != null) {
                albumRecyclerView.setViewPager2(viewPager2);

                // Disable ViewPager2 scrolling when RecyclerView is being
                touched
                albumRecyclerView.setOnTouchListener(new
                View.OnTouchListener() {
                    @Override
                    public boolean onTouch(View v, MotionEvent event) {
                        switch (event.getAction()) {
                            case MotionEvent.ACTION_DOWN:
                                viewPager2.setUserInputEnabled(false);
                                break;
                            case MotionEvent.ACTION_UP:
                            case MotionEvent.ACTION_CANCEL:
                                viewPager2.setUserInputEnabled(true);
                                break;
                        }
                        return false;
                    }
                });
            }
        });

        albumRecyclerView.setLayoutManager(new
        LinearLayoutManager(getActivity(), LinearLayoutManager.HORIZONTAL, false));
        allTrackRecyclerView.setLayoutManager(new
        LinearLayoutManager(getActivity(), LinearLayoutManager.VERTICAL, false));

```

```

        fetchDataFromFirestore();

        retrieveAllTracksFromAlbum("alltracks");

        return v;
    }

    private void retrieveAllTracksFromAlbum(String album) {
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection(album.toLowerCase()).get().addOnCompleteListener(new
        OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    List<AudioTrackModel> audioList = new ArrayList<>();

                    for (QueryDocumentSnapshot document : task.getResult())
                    {

                        String albumName = document.getString("album");
                        String artistName = document.getString("artist");
                        String genre = document.getString("genre");
                        String title = document.getString("title");

                        // Convert Firestore document to
                        AlbumModelForRecycler
                        AudioTrackModel audio = new AudioTrackModel(title,
                        artistName, genre, albumName);
                        audioList.add(audio);
                    }

                    Collections.shuffle(audioList);

                    mixRecyclerViewAdapter = new
                    MixRecyclerViewAdapter(audioList, getContext(), isPremium);

                    allTrackRecyclerView.setAdapter(mixRecyclerViewAdapter);

                    // Notify the adapter that the data has changed
                    mixRecyclerViewAdapter.notifyDataSetChanged();
                } else {
                    Toast.makeText(getContext(), "Error in fetching tracks
                    from " + album, Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    private void fetchDataFromFirestore() {
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection("albums")
            .get()

```

```

        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>()
        {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    List<AlbumModelForRecycler> albumList = new
ArrayList<>();

                    for (QueryDocumentSnapshot document :
task.getResult()) {
                        // Retrieve only the "album" and "artist"
fields
                        String albumName =
document.getString("albumName");
                        String artistName =
document.getString("artistName");
                        int cover = R.drawable.audiowavesups;

                        // Convert Firestore document to
AlbumModelForRecycler
                        AlbumModelForRecycler album = new
AlbumModelForRecycler(cover, albumName, artistName);
                        albumList.add(album);
                    }

                    Collections.shuffle(albumList);

                    // Create an instance of the
AlbumRecyclerViewAdapter and pass the albumList
                    albumAdapter = new
AlbumRecyclerViewAdapter(albumList, getContext(), isPremium);

                    // Set the adapter to the recyclerView
albumRecyclerView.setAdapter(albumAdapter);

                    // Notify the adapter that the data has changed
albumAdapter.notifyDataSetChanged();
                } else {
                    // Handle failure
                }
            }
        });
    }
}

```

Favourites.java

```

package com.example.mellowsphere;

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

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;

```



```

import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.FrameLayout;
import android.widget.ListView;
import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;

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

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link FavouritesFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class FavouritesFragment extends Fragment {

    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;

    public FavouritesFragment() {
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment FavouritesFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static FavouritesFragment newInstance(String param1, String
param2) {
        FavouritesFragment fragment = new FavouritesFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    boolean isPremium;
    private static List<AudioTrackModel> favouritesList = new
ArrayList<>();

```

```

    public static FavouritesFragment newInstance(List<AudioTrackModel>
favouritesList, boolean isPremium) {
        FavouritesFragment fragment = new FavouritesFragment();
        Bundle args = new Bundle();
        args.putBoolean("isPremium", isPremium);
        args.putParcelableArrayList("favouritesList", new
ArrayList<>(favouritesList));
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
            isPremium = getArguments().getBoolean("isPremium", false);
            favouritesList =
getArguments().getParcelableArrayList("favouritesList");
        }
    }

    ListView listView;
    AudioTrackListViewAdapter audioTrackListViewAdapter;
    FrameLayout emptyList;

    FirebaseFirestore db = FirebaseFirestore.getInstance();
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View v = inflater.inflate(R.layout.fragment_favourites, container,
false);

        listView = v.findViewById(R.id.listView);

        emptyList = v.findViewById(R.id.emptylist);

        if (favouritesList.isEmpty()) {
            listView.setVisibility(View.GONE);
            emptyList.setVisibility(View.VISIBLE);
        }

        audioTrackListViewAdapter = new
AudioTrackListViewAdapter(getContext(), favouritesList);

        listView.setAdapter(audioTrackListViewAdapter);

        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {

```

```

        AudioTrackModel selectedTrack =
audioTrackListViewAdapter.getAudioList().get(position);

        if (selectedTrack.getTitle() == null) {
            Toast.makeText(getContext(), "Error",
Toast.LENGTH_SHORT).show();
        }
        else {
            Intent i = new Intent(getContext(), AudioPlayer.class);
            i.putExtra("TrackName", selectedTrack.getTitle());
            i.putExtra("ArtistName", selectedTrack.getArtist());
            i.putExtra("AlbumName", selectedTrack.getAlbum());
            i.putExtra("TrackList", new
ArrayList<>(favouritesList));
            i.putExtra("IsPremium", isPremium);
            startActivity(i);
        }
    }
});

return v;
}
}

```

ProfileFragment.java

```

package com.example.mellowsphere;

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

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

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.Map;

/**
 * A simple {@link Fragment} subclass.
 * Use the {@link ProfileFragment#newInstance} factory method to
 * create an instance of this fragment.
 */
public class ProfileFragment extends Fragment {

```

```

// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private static final String ARG_PARAM1 = "param1";
private static final String ARG_PARAM2 = "param2";

// TODO: Rename and change types of parameters
private String mParam1;
private String mParam2;

public ProfileFragment() {
    // Required empty public constructor
}

/**
 * Use this factory method to create a new instance of
 * this fragment using the provided parameters.
 *
 * @param param1 Parameter 1.
 * @param param2 Parameter 2.
 * @return A new instance of fragment ProfileFragment.
 */
// TODO: Rename and change types and number of parameters
public static ProfileFragment newInstance(String param1, String param2)
{
    ProfileFragment fragment = new ProfileFragment();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
    }
}

TextView name, email, notloggedin;
Button logout;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View v = inflater.inflate(R.layout.fragment_profile, container,
false);

    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

    name = v.findViewById(R.id.name);
    email = v.findViewById(R.id.email);
    notloggedin = v.findViewById(R.id.notloggedin);
    logout = v.findViewById(R.id.logout);

```

```

        if (user != null) {
            name.setVisibility(View.VISIBLE);
            email.setVisibility(View.VISIBLE);
            logout.setVisibility(View.VISIBLE);

            notloggedin.setVisibility(View.GONE);

            //Initialize firestore
            FirebaseFirestore db = FirebaseFirestore.getInstance();

            db.collection("users").document(user.getId()).get().addOnCompleteListener(
new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot>
task) {
                    if (task.isSuccessful()) {
                        DocumentSnapshot document = task.getResult();

                        if (document.exists()) {
                            // DocumentSnapshot contains the data
                            Map<String, Object> userData =
document.getData();

                            // Access specific fields
                            if (userData != null) {
                                String username = (String)
userData.get("username");
                                String e_mail = (String)
userData.get("email");

                                name.setText(username);
                                email.setText(e_mail);
                            }
                        }
                    }
                }
            });

            logout.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    FirebaseAuth.getInstance().signOut();
                    Intent intent = new
Intent(getActivity().getPackageManager()

.getLaunchIntentForPackage(getActivity().getPackageName()));
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(intent);
                }
            });
        }
        else {
            name.setVisibility(View.GONE);
            email.setVisibility(View.GONE);
            logout.setVisibility(View.GONE);

```

```

        notloggedin.setVisibility(View.VISIBLE);
    }

    return v;
}
}

```

PremiumRegister.java

```

package com.example.mellowsphere;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class PremiumRegister extends AppCompatActivity {
    private long lastBackPressedTime = 0;
    int emptySpacePosIncrement = 0;
    EditText nameOnCard, cardNumber, expiryDate, securityCode, zipCode;
    Button getPremium, back;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_premium_register);

        nameOnCard = findViewById(R.id.nameoncard);
        cardNumber = findViewById(R.id.cardnumber);
        expiryDate = findViewById(R.id.expiry);
        securityCode = findViewById(R.id.security);
        zipCode = findViewById(R.id.zipcode);

        getPremium = findViewById(R.id.getpremium);
    }
}

```

```

back = findViewById(R.id.exitpremium);

// Define the regex pattern for MM/YY format
String regex = "^(0[1-9]|1[0-2])/(\d{2})$";

// Create a Pattern object
Pattern pattern = Pattern.compile(regex);

cardNumber.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int
before, int count) {

    }

    @Override
    public void afterTextChanged(Editable s) {
        try {
            String cardnumber =
cardNumber.getText().toString().replaceAll("\\s", "");

            Long.parseLong(cardnumber);
        }
        catch (NumberFormatException e) {
            if (!cardNumber.getText().toString().equals("")) {
                cardNumber.setError("Invalid card number");
            }
        }

        // Remove the existing spaces
        String cardnumber = s.toString().replace(" ", "");

        // Add spaces every fourth digit
        StringBuilder formattedCardNumber = new StringBuilder();
        for (int i = 0; i < cardnumber.length(); i++) {
            formattedCardNumber.append(cardnumber.charAt(i));
            if ((i + 1) % 4 == 0 && i + 1 < cardnumber.length()) {
                formattedCardNumber.append(" ");
            }
        }

        // Set the formatted text back to the EditText
        cardNumber.removeTextChangedListener(this); // Avoid
infinite loop
        cardNumber.setText(formattedCardNumber.toString());
        cardNumber.setSelection(formattedCardNumber.length());
        cardNumber.addTextChangedListener(this);
    }
});

getPremium.setOnClickListener(new View.OnClickListener() {

```

```

@Override
public void onClick(View v) {
    if (!(nameOnCard.getText().toString().length() >= 3 &&
nameOnCard.getText().toString().length() <= 26)){
        nameOnCard.setError("Name should be within 3-26
characters");
        return;
    }

    if (!(cardNumber.getText().toString().length() == 19)){
        cardNumber.setError("Card number must be 16 digit");
        return;
    }

    // Create a Matcher object
    Matcher matcher =
pattern.matcher(expiryDate.getText().toString());

    // Check if the input matches the pattern
    if (!matcher.matches()) {
        expiryDate.setError("Date should be in the format
MM/YY");
        return;
    }

    if (!(securityCode.getText().toString().length() == 3)) {
        securityCode.setError("Security code must be 3
digits");
        return;
    }

    if (zipCode.getText().toString().length() != 5) {
        zipCode.setError("Zip code must be 5 digits");
        return;
    }

    FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    // Specify the field and its new value in a Map
    Map<String, Object> updates = new HashMap<>();
    updates.put("isPremium", "t");

    if (user != null) {
        db.collection("users").document(user.getUid()).update(updates).addOnSuccess
Listener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void unused) {
                Toast.makeText(PremiumRegister.this, "Enjoy our
premium services!", Toast.LENGTH_SHORT).show();

                Intent intent =
getBaseContext().getPackageManager()

.getLaunchIntentForPackage(getBaseContext().getPackageName());

```



```

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
        });
    }

    });

    back.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();
        }
    });

    // Set touch listener on the root view
    ViewGroup rootView = findViewById(android.R.id.content);
    rootView.setOnTouchListener(new View.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            hideKeyboardAndCursor();
            return false;
        }
    });

}

private void hideKeyboardAndCursor() {
    InputMethodManager imm = (InputMethodManager)
getSystemService(INPUT_METHOD_SERVICE);
    if (getCurrentFocus() != null) {
        imm.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
0);
        getCurrentFocus().clearFocus();
    }
}

@Override
public void onBackPressed() {
    long currentTime = System.currentTimeMillis();

    if (currentTime - lastBackPressedTime < 2000) {
        super.onBackPressed();
    } else {
        Toast.makeText(this, "Press back again to exit",
Toast.LENGTH_SHORT).show();
        lastBackPressedTime = currentTime;
    }
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        onBackPressed();
        return true;
    }
}

```

```

    }
    return super.onKeyDown(keyCode, event);
}
}

```

Register.java

```

package com.example.mellowsphere;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.core.content.ContextCompat;

import android.content.Intent;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.text.method.PasswordTransformationMethod;
import android.text.method.SingleLineTransformationMethod;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

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;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseAuthException;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;

import java.util.HashMap;
import java.util.Map;

public class Register extends AppCompatActivity {
    private long lastBackPressed = 0;
    private FirebaseAuth mAuth;

    //for switching between normal and error display of edittext
    private boolean switchEditTextdisp;

```

```

        //for setting the constraint of linear layout containing option to
switch to sign up or login
        ConstraintLayout mainLayout;
        LinearLayout linearLayout;
        ConstraintLayout.LayoutParams layoutParams;
        TextView title;

        //components of sign up display
        EditText usernameET;
        EditText emailET;
        RelativeLayout relativeLayout;
        EditText passwordET;
        ImageView togglePasswordVisibility;
        RelativeLayout relativeLayout2;
        EditText confPassET;
        ImageView togglePasswordVisibility2;

        //components of sign up display
        EditText usernameLogin;
        RelativeLayout relativeLayout3;
        EditText passwordLogin;
        ImageView togglePasswordVisibility3;
        TextView forgotPass;
private boolean isPasswordVisible = false;

        //components for switching display
        TextView switchTV;
        TextView switchDisplayTV;
        Button submit;
private boolean switchDisplay = false;
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_register);

            mAuth = FirebaseAuth.getInstance();

            mainLayout = findViewById(R.id.constrlayout);
            linearLayout = findViewById(R.id.linearlayout);
            layoutParams = (ConstraintLayout.LayoutParams)
linearLayout.getLayoutParams();

            title = findViewById(R.id.signuplogin);

            //signup
            usernameET = findViewById(R.id.usernameET);
            emailET = findViewById(R.id.emailET);
            relativeLayout = findViewById(R.id.relativelayout);
            passwordET = findViewById(R.id.passwordET);
            togglePasswordVisibility =
findViewById(R.id.togglePasswordVisibility);
            relativeLayout2 = findViewById(R.id.relativelayout2);
            confPassET = findViewById(R.id.confpassET);
            togglePasswordVisibility2 =
findViewById(R.id.togglePasswordVisibility2);

            //login

```

```

        usernameLogin = findViewById(R.id.usernamelogin);
        relativeLayout3 = findViewById(R.id.relativelayout3);
        passwordLogin = findViewById(R.id.passwordlogin);
        togglePasswordVisibility3 =
findViewById(R.id.togglePasswordVisibility3);
        forgotPass = findViewById(R.id.forgotpass);

        //switch display
        switchTV = findViewById(R.id.switchTV);
        switchDisplayTV = findViewById(R.id.switchdisplayTV);
        submit = findViewById(R.id.submit);

        // Set touch listener on the root view
        ViewGroup rootView = findViewById(android.R.id.content);
        rootView.setOnTouchListener(new View.OnTouchListener() {
            @Override
            public boolean onTouch(View v, MotionEvent event) {
                hideKeyboardAndCursor();
                return false;
            }
        });

        //onclick function for submitting
        submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (switchDisplay) {
                    login();
                }
                else {
                    register();
                }
            }
        });

        handlePasswordVisibility(togglePasswordVisibility, passwordET);
        handlePasswordVisibility(togglePasswordVisibility2, confPassET);
        handlePasswordVisibility(togglePasswordVisibility3, passwordLogin);

        forgotPass.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showForgotPasswordDialog();
            }
        });
    }

    private void showForgotPasswordDialog() {
        ForgotPassword forgotPasswordDialog = new ForgotPassword();
        forgotPasswordDialog.show(getSupportFragmentManager(),
"ForgotPasswordDialog");
    }

    public void switchDisp(View view) {
        switchDisplay = !switchDisplay;

        //hide keyboard if displayed

```

```

        InputMethodManager imm = (InputMethodManager)
getSystemService (INPUT_METHOD_SERVICE);
        if (getCurrentFocus() != null) {
            imm.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
0);
            getCurrentFocus().clearFocus();
        }
        if (switchDisplay) {
            loginDisplay();
        }
        else{
            signUpDisplay();
        }
    }

    public void loginDisplay() {
        layoutParams.topToBottom = relativeLayout3.getId();
        linearLayout.setLayoutParams(layoutParams);
        mainLayout.requestLayout();

        //showing login components
        usernameLogin.setVisibility(View.VISIBLE);
        relativeLayout3.setVisibility(View.VISIBLE);
        forgotPass.setVisibility(View.VISIBLE);

        usernameLogin.setBackgroundResource(R.drawable.custom_edittext_1);
        passwordLogin.setBackgroundResource(R.drawable.custom_edittext_1);

        usernameLogin.setText("");
        passwordLogin.setText("");

        Drawable vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_username_icon);

        usernameLogin.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset,
null, null, null);

        vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_password_icon);

        passwordLogin.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset,
null, null, null);

        //hiding signup components
        usernameET.setVisibility(View.GONE);
        emailET.setVisibility(View.GONE);
        relativeLayout.setVisibility(View.GONE);
        relativeLayout2.setVisibility(View.GONE);

        title.setText("Log In");
        switchTV.setText("Don't have an account? ");
        switchDisplayTV.setText("Sign Up");

        submit.setText("Login");
    }

    public void signUpDisplay() {
        layoutParams.topToBottom = relativeLayout2.getId();

```

```

        linearLayout.setLayoutParams(layoutParams);
        mainLayout.requestLayout();

        //hiding login components
        usernameLogin.setVisibility(View.GONE);
        relativeLayout3.setVisibility(View.GONE);
        forgotPass.setVisibility(View.GONE);

        //showing signup components
        usernameET.setVisibility(View.VISIBLE);
        emailET.setVisibility(View.VISIBLE);
        relativeLayout.setVisibility(View.VISIBLE);
        relativeLayout2.setVisibility(View.VISIBLE);

        usernameET.setBackgroundResource(R.drawable.custom_edittext_1);
        emailET.setBackgroundResource(R.drawable.custom_edittext_1);
        passwordET.setBackgroundResource(R.drawable.custom_edittext_1);
        confPassET.setBackgroundResource(R.drawable.custom_edittext_1);

        usernameET.setText("");
        emailET.setText("");
        passwordET.setText("");
        confPassET.setText("");

        Drawable vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_username_icon);

        usernameET.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset,
null, null, null);

        vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_email_icon);

        emailET.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset, null,
null, null);

        vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_password_icon);

        passwordET.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset,
null, null, null);

        vectorAsset = ContextCompat.getDrawable(this,
R.drawable.custom_password_icon);

        confPassET.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset,
null, null, null);

        passwordET.setTransformationMethod(new
        PasswordTransformationMethod());

        togglePasswordVisibility.setImageResource(R.drawable.password_invisible);

        confPassET.setTransformationMethod(new
        PasswordTransformationMethod());

        togglePasswordVisibility2.setImageResource(R.drawable.password_invisible);

```

```

        passwordLogin.setTransformationMethod(new
        PasswordTransformationMethod());

togglePasswordVisibility3.setImageResource(R.drawable.password_invisible);

        title.setText("Sign Up");
        switchTV.setText("Already have an account? ");
        switchDisplayTV.setText("Log In");

        submit.setText("Register");
    }

    public void back(View view) {
        finish();
    }

    private void login(){
        String username, password;

        username = usernameLogin.getText().toString();
        password = passwordLogin.getText().toString();

        checkIfEmpty(username, usernameLogin,
        R.drawable.custom_username_icon_error, R.drawable.custom_username_icon);
        checkIfEmpty(password, passwordLogin,
        R.drawable.custom_password_icon_error, R.drawable.custom_password_icon);

        if (username.isEmpty() || password.isEmpty()) {
            Toast.makeText(getApplicationContext(), "Please enter required
            credentials", Toast.LENGTH_SHORT).show();
            return;
        }

        verifyUsername(username, password);
    }

    private void verifyUsername(String username, String password) {

        // Store additional user data in Firestore
        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection("users")
            .whereEqualTo("username", username)
            .get().addOnCompleteListener(new
        OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task)
        {
            if (task.isSuccessful()) {
                if (task.getResult().isEmpty()){
                    Toast.makeText(Register.this, "Authentication
                    failed.",
                                Toast.LENGTH_SHORT).show();
                }
                else{

```

```

        for (QueryDocumentSnapshot userDoc :
task.getResult()) {
        // Retrieve the email from the user
        document

        String email = userDoc.getString("email");
        String uid = userDoc.getId();

        signIn(email, password, db);
    }
}
else {
    Toast.makeText(Register.this, "Authentication
failed.",
        Toast.LENGTH_SHORT).show();
}
}
);
}
private void signIn(String email, String password, FirebaseFirestore
db) {
    mAuth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    if (task.getResult() == null) {
                        Toast.makeText(Register.this, "Authentication
failed.",
                            Toast.LENGTH_SHORT).show();
                    }
                    else {
                        FirebaseUser user = mAuth.getCurrentUser();

                        assert user != null;

                        db.collection("expiration").document(user.getUid()).get().addOnCompleteList
ener(new OnCompleteListener<DocumentSnapshot>() {
                            @Override
                            public void onComplete(@NonNull
Task<DocumentSnapshot> task) {
                                DocumentSnapshot result =
task.getResult();

                                if (result == null) {
                                    Toast.makeText(Register.this, ".",
                                        Toast.LENGTH_SHORT).show();
                                }
                                else {
                                    long expirationTimestamp =

                                    if (result.getLong("current") ==

                                    long currentTime =
System.currentTimeMillis();

```



```

                                if (currentTime <
expirationTimestamp && user.isEmailVerified()) {

db.collection("expiration").document(user.getUid()).update("current",
currentTime);

                                                                    // Sign in success, update
UI with the signed-in user's information

Toast.makeText(Register.this, "Login successful",

Toast.LENGTH_SHORT).show();

//                                                                    Intent homepage = new
Intent(getApplicationContext(), MainActivity.class);
//
homepage.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
//                                                                    startActivity(homepage);
//                                                                    finish();

                                                                    Intent intent =
getBaseContext().getPackageManager()

.getLaunchIntentForPackage(getBaseContext().getPackageName());

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                                                                    startActivity(intent);
}
                                else if (currentTime <
expirationTimestamp && !user.isEmailVerified()) {
                                                                    int minutes = (int)
(expirationTimestamp-currentTime)/1000/60;
                                                                    int seconds = (int)
(expirationTimestamp-currentTime)/1000 - minutes*60;

Toast.makeText(Register.this, "Please verify email within "+minutes+"m
"+seconds+"s",

Toast.LENGTH_SHORT).show();

                                                                    mAuth.signOut();
}
                                else {
                                                                    // Get a Firestore instance
                                                                    FirebaseFirestore
firestore = FirebaseFirestore.getInstance();

                                                                    String users = "users";
                                                                    String expiration =
"expiration";
                                                                    String documentId =
user.getUid();

                                                                    // Create a reference to
the document

firestore.collection(users).document(documentId).delete();

```

```

// Create a reference to
the document

firestore.collection(expiration).document(documentId).delete();

        mAuth.signOut();
        user.delete();
    }
}
else {
    long currentTime =
result.getLong("current");

        if (currentTime <
expirationTimestamp && user.isEmailVerified()) {
            // Sign in success, update
            UI with the signed-in user's information

            Toast.makeText(Register.this, "Login successful",
            Toast.LENGTH_SHORT).show();

            //
            Intent homepage = new
            Intent(getApplicationContext(), MainActivity.class);
            //
            homepage.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            //
            startActivity(homepage);
            //
            finish();

            Intent intent =
getBaseContext().getPackageManager()

.getLaunchIntentForPackage(getBaseContext().getPackageName());

intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
        }
        else if (currentTime <
expirationTimestamp && !user.isEmailVerified()) {
            int minutes = (int)
(expirationTimestamp-currentTime)/1000/60;
            int seconds = (int)
(expirationTimestamp-currentTime)/1000 - minutes*60;

            Toast.makeText(Register.this, "Please verify email within "+minutes+"m
            "+seconds+"s",
            Toast.LENGTH_SHORT).show();

            mAuth.signOut();
        }
        else {
            // Get a Firestore instance
            FirebaseFirestore

            firestore = FirebaseFirestore.getInstance();

            String users = "users";

```

```

        String expiration =
        String documentId =

        // Create a reference to
        the document
        firestore.collection(users).document(documentId).delete();

        // Create a reference to
        the document
        firestore.collection(expiration).document(documentId).delete();

        mAuth.signOut();
        user.delete();
    }
}
});
}
else {
    Toast.makeText(Register.this, "Authentication
failed.",
                    Toast.LENGTH_SHORT).show();
}
}
);
}

private void register(){
    String username, email, password;

    username = usernameET.getText().toString();
    email = emailET.getText().toString();
    password = passwordET.getText().toString();

    checkIfEmpty(username, usernameET,
R.drawable.custom_username_icon_error, R.drawable.custom_username_icon);
    checkIfEmpty(email, emailET, R.drawable.custom_email_icon_error,
R.drawable.custom_email_icon);
    checkIfEmpty(password, passwordET,
R.drawable.custom_password_icon_error, R.drawable.custom_password_icon);
    checkIfEmpty(confPassET.getText().toString(), confPassET,
R.drawable.custom_password_icon_error, R.drawable.custom_password_icon);

    if (username.isEmpty() || email.isEmpty() || password.isEmpty()) {
        Toast.makeText(getApplicationContext(), "Please enter required
credentials", Toast.LENGTH_SHORT).show();
        return;
    }
}

```

```

    }
    if (!(username.length() >= 3 && username.length() <= 30)) {
        Toast.makeText(getApplicationContext(), "Username must be
minimum 3 and maximum 30 characters", Toast.LENGTH_SHORT).show();
        return;
    }
    if (!(password.length() >= 8 && password.length() <= 16)){
        Toast.makeText(getApplicationContext(), "Password must be
minimum 8 characters or maximum 16 characters", Toast.LENGTH_SHORT).show();
        return;
    }
    if (!password.equals(confPassET.getText().toString())){
        Toast.makeText(getApplicationContext(), "Passwords don't
match", Toast.LENGTH_SHORT).show();
        return;
    }
    FirebaseFirestore db = FirebaseFirestore.getInstance();

    db.collection("users").whereEqualTo("username",
username).get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                QuerySnapshot querySnapshot = task.getResult();
                if (querySnapshot != null && !querySnapshot.isEmpty())
{
                    Toast.makeText(getApplicationContext(), "Username
already exists",
                        Toast.LENGTH_SHORT).show();
                }
                else {
                    createUser(username, email, password);
                }
            }
        }
    });
}

private void createUser(String username, String email, String
password){
    mAuth.createUserWithEmailAndPassword(email, password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // Sign in success, update UI with the signed-in
user's information
                FirebaseUser user = mAuth.getCurrentUser();

                // Store additional user data in Firestore
                FirebaseFirestore db =
FirebaseFirestore.getInstance();

                sendVerificationEmailAndCreateUser(user, db, email,
username);
            }
            else {

```

```

        // If sign in fails, display a message to the user.
        Toast.makeText(Register.this, "Authentication
failed.",
                                Toast.LENGTH_SHORT).show();
    }
}

);
}

private void sendVerificationEmailAndCreateUser(FirebaseUser user,
FirebaseFirestore db, String email, String username) {
    // Send verification email
    user.sendEmailVerification()
        .addOnCompleteListener(new OnCompleteListener() {
            @Override
            public void onComplete(@NonNull Task emailTask) {
                if (emailTask.isSuccessful()) {
                    // Email sent successfully
                    Toast.makeText(getApplicationContext(),
"Verification mail sent to "+email, Toast.LENGTH_SHORT).show();
                    Toast.makeText(getApplicationContext(), "Verify
email within 3 minutes to successfully create account",
Toast.LENGTH_SHORT).show();
                    // Store timestamp in Firebase Database or Cloud
Firestore
                    long expirationTimestamp =
System.currentTimeMillis() + (3 * 60 * 1000);
                    Map<String, Object> expirationTime = new
HashMap<>();
                    expirationTime.put("expiry", expirationTimestamp);
                    expirationTime.put("current", null);

                    db.collection("expiration").document(user.getUid()).set(expirationTime);

                    Map<String, Object> userData = new HashMap<>();
                    userData.put("email", email);
                    userData.put("username", username);
                    userData.put("isPremium", "f");

                    assert user != null;

                    db.collection("users").document(user.getUid()).set(userData)
                        .addOnSuccessListener(new
OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                // Registration and data storage
successful

                                switchDisp(findViewById(R.id.switchdisplayTV));
                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override

```

```

        public void onFailure(@NonNull Exception e)
    {
        Toast.makeText(getApplicationContext(),
        "Firestore data storing failed.",
        Toast.LENGTH_SHORT).show();
    }
    };
}
else {
    // Email not sent, handle the error
    Exception exception = emailTask.getException();
    if (exception instanceof FirebaseAuthException) {
        FirebaseAuthException authException =
        (FirebaseAuthException) exception;
        if
        (authException.getErrorCode().equals("ERROR_INVALID_EMAIL")) {
            // Invalid email address
            Toast.makeText(Register.this, "Invalid
            email address.",
            Toast.LENGTH_SHORT).show();
            user.delete();
        }
        else {
            // Handle other FirebaseAuthExceptions
            Toast.makeText(Register.this, "Email
            verification failed: " + exception.getMessage(),
            Toast.LENGTH_SHORT).show();
            user.delete();
        }
    }
}
});
}

private void hideKeyboardAndCursor() {
    InputMethodManager imm = (InputMethodManager)
    getSystemService(INPUT_METHOD_SERVICE);
    if (getCurrentFocus() != null) {
        imm.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),
    0);
        getCurrentFocus().clearFocus();
    }
}

@Override
public void onBackPressed() {
    long currentTime = System.currentTimeMillis();

    if (currentTime - lastBackPressedTime < 2000) {
        super.onBackPressed();
    } else {
        Toast.makeText(this, "Press back again to exit",
        Toast.LENGTH_SHORT).show();
        lastBackPressedTime = currentTime;
    }
}

```

```

    }

    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK) {
            onBackPressed();
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }

    private void checkIfEmpty(String str, EditText editText, int
errorDrawableId, int normalDrawableID){
        if (str.isEmpty()) {
            switchEditTextdisp = true;
            switchEditTextDisp(editText, errorDrawableId,
normalDrawableID);
        }
        else{
            switchEditTextdisp = false;
            switchEditTextDisp(editText, errorDrawableId,
normalDrawableID);
        }
    }

    private void switchEditTextDisp(EditText edittext, int errorDrawableId,
int normalDrawableID) {
        if (switchEditTextdisp){
            edittext.setBackgroundResource(R.drawable.custom_edittext_2);
            Drawable vectorAsset = ContextCompat.getDrawable(this,
errorDrawableId);

            edittext.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset, null,
null, null);
        }
        else{
            edittext.setBackgroundResource(R.drawable.custom_edittext_1);
            Drawable vectorAsset = ContextCompat.getDrawable(this,
normalDrawableID);

            edittext.setCompoundDrawablesRelativeWithIntrinsicBounds(vectorAsset, null,
null, null);
        }
    }

    private void handlePasswordVisibility(ImageView imageView, EditText
editText) {
        imageView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if
(editText.getTransformationMethod().getClass().getSimpleName()
.equals("PasswordTransformationMethod")) {
                    editText.setTransformationMethod(new
SingleLineTransformationMethod());
                }
            }
        });
        imageView.setImageResource(R.drawable.password_visible);
    }

```



```

    public AudioListViewFragment(List<AudioTrackModel> audioList, boolean
isPremium) {
        this.audioList = audioList;
        this.isPremium = isPremium;
        // Required empty public constructor
    }

    /**
     * Use this factory method to create a new instance of
     * this fragment using the provided parameters.
     *
     * @param param1 Parameter 1.
     * @param param2 Parameter 2.
     * @return A new instance of fragment AudioListViewFragment.
     */
    // TODO: Rename and change types and number of parameters
    public static AudioListViewFragment newInstance(String param1, String
param2) {
        AudioListViewFragment fragment = new AudioListViewFragment();
        Bundle args = new Bundle();
        args.putString(ARG_PARAM1, param1);
        args.putString(ARG_PARAM2, param2);
        fragment.setArguments(args);
        return fragment;
    }

    public static AudioListViewFragment newInstance(List<AudioTrackModel>
audioList, boolean isPremium) {
        AudioListViewFragment fragment = new AudioListViewFragment();
        Bundle args = new Bundle();
        args.putBoolean("isPremium", isPremium);
        args.putParcelableArrayList("audioList", new
ArrayList<>(audioList));
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            mParam1 = getArguments().getString(ARG_PARAM1);
            mParam2 = getArguments().getString(ARG_PARAM2);
            isPremium = getArguments().getBoolean("isPremium", false);
            audioList = getArguments().getParcelableArrayList("audioList");
        }
    }

    ListView listView;
    AudioTrackListViewAdapter audioTrackListViewAdapter;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        // Inflate the layout for this fragment

```

```

        View v = inflater.inflate(R.layout.fragment_audio_list_view,
container, false);

        listView = v.findViewById(R.id.listView);

        audioTrackListViewAdapter = new
AudioTrackListViewAdapter(getContext(), audioList);

        listView.setAdapter(audioTrackListViewAdapter);

        listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
                AudioTrackModel selectedTrack =
audioTrackListViewAdapter.getAudioList().get(position);

                if (selectedTrack.getTitle() == null) {
                    Toast.makeText(getContext(), "Error",
Toast.LENGTH_SHORT).show();
                }
                else {
                    Intent i = new Intent(getContext(), AudioPlayer.class);
                    i.putExtra("TrackName", selectedTrack.getTitle());
                    i.putExtra("ArtistName", selectedTrack.getArtist());
                    i.putExtra("AlbumName", selectedTrack.getAlbum());
                    i.putExtra("TrackList", new ArrayList<>(audioList));
                    i.putExtra("IsPremium", isPremium);
                    startActivity(i);
                }
            }
        });

        return v;
    }
}

```

AlbumRecyclerViewAdapter.java

```

package com.example.mellowsphere;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
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.List;

public class AlbumRecyclerViewAdapter extends
RecyclerView.Adapter<AlbumRecyclerViewAdapter.ViewHolder> {

    static Context context;
    private List<AlbumModelForRecycler> albumList;
    static boolean isPremium;
    //private List<AudioTrackModel> audioList;

    public AlbumRecyclerViewAdapter(List<AlbumModelForRecycler> albumList,
Context context, boolean isPremium) {
        this.albumList = albumList;
        this.context = context;
        this.isPremium = isPremium;
    }

    @NonNull
    @Override
    public AlbumRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        // Inflate the item layout
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.album_recyclervie
w, parent, false);

        // Create a ViewHolder and pass the CardView as the itemView
        return new ViewHolder((CardView) view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
{
        // Get the data for the current position
        AlbumModelForRecycler album = albumList.get(position);
        // CardView cv = holder.cv;
        // ImageView albumImage = cv.findViewById(R.id.albumImage);
        // TextView albumName= cv.findViewById(R.id.albumName);
        // TextView artistName = cv.findViewById(R.id.artistName);

        // Set data to views in the ViewHolder
        holder.albumImage.setImageResource(album.getCover());
        holder.albumName.setText(album.getAlbumName());
        holder.artistName.setText(album.getArtistName());

        FirebaseFirestore db = FirebaseFirestore.getInstance();

        db.collection(album.getAlbumName().toLowerCase()).get().
addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
    @Override
    public void onComplete(@NonNull Task<QuerySnapshot> task) {
        if (task.isSuccessful()) {

```

```

        List<AudioTrackModel> audioTrackModelList = new
ArrayList<>();

        for (QueryDocumentSnapshot document : task.getResult())
        {
            String albumName = document.getString("album");
            String artistName = document.getString("artist");
            String genre = document.getString("genre");
            String title = document.getString("title");

            // Convert Firestore document to
AlbumModelForRecycler
            AudioTrackModel audio = new AudioTrackModel(title,
artistName, genre, albumName);
            audioTrackModelList.add(audio);

            holder.audioList = audioTrackModelList;
        }
    }
});
}

@Override
public int getItemCount() {
    // Return the size of your dataset
    return albumList.size();
}

public static class ViewHolder extends RecyclerView.ViewHolder {
    ImageView albumImage;
    TextView albumName;
    TextView artistName;
    CardView cv;
    List<AudioTrackModel> audioList;

    public ViewHolder(CardView cardView) {
        super(cardView);
        albumImage = cardView.findViewById(R.id.albumImage);
        albumName = cardView.findViewById(R.id.albumName);
        artistName = cardView.findViewById(R.id.artistName);
        audioList = new ArrayList<>();

        cardView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (audioList != null){
                    ((MainActivity)context).getSupportFragmentManager().beginTransaction().repl
ace(R.id.frameLayout, AudioListViewFragment.newInstance(audioList,
isPremium)).commit();
                }
            }
        });
    }
}
}
}

```

MixRecyclerViewAdapter.java

```
package com.example.mellowsphere;

import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.constraintlayout.widget.ConstraintLayout;
import androidx.recyclerview.widget.RecyclerView;

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

public class MixRecyclerViewAdapter extends
RecyclerView.Adapter<MixRecyclerViewAdapter.ViewHolder> {

    static Context context;
    private List<AudioTrackModel> audioList;
    static String album1;
    boolean isPremium;
    //private List<AudioTrackModel> audioList;

    public MixRecyclerViewAdapter(List<AudioTrackModel> audioList, Context
context, boolean isPremium) {
        this.audioList = audioList;
        this.context = context;
        this.isPremium = isPremium;
    }

    @NonNull
    @Override
    public MixRecyclerViewAdapter.ViewHolder onCreateViewHolder(@NonNull
ViewGroup parent, int viewType) {
        // Inflate the item layout
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.audio_listview_la
yout, parent, false);

        // Create a ViewHolder and pass the CardView as the itemView
        return new ViewHolder((ConstraintLayout) view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
{
        // Get the data for the current position
        AudioTrackModel audio = audioList.get(position);

        // Set data to views in the ViewHolder
        holder.title.setText(audio.getTitle());
        holder.artistName.setText(audio.getArtist());
    }
}
```

```

album1 = audio.getAlbum();

android.os.Handler handler = new Handler();
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        holder.title.setSelected(true);
    }
}, 2000);

// Access the root layout (ConstraintLayout)
ConstraintLayout rootLayout = (ConstraintLayout) holder.itemView;

rootLayout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (audioList != null) {
            Intent i = new Intent(context, AudioPlayer.class);
            i.putExtra("TrackName", audio.getTitle());
            i.putExtra("ArtistName", audio.getArtist());
            i.putExtra("AlbumName", album1);
            i.putExtra("TrackList", new ArrayList<>(audioList));
            i.putExtra("IsPremium", isPremium);
            context.startActivity(i);
        }
    }
});
}

@Override
public int getItemCount() {
    // Return the size of your dataset
    return audioList.size();
}

public static class ViewHolder extends RecyclerView.ViewHolder {
//    ImageView albumImage;
    TextView title;
    TextView artistName;
    List<AudioTrackModel> audioList;

    public ViewHolder(ConstraintLayout cl) {
        super(cl);
        title = cl.findViewById(R.id.trackname);
        artistName = cl.findViewById(R.id.artistName);
        audioList = null;
    }
}
}

```

ForgotPassword.java

```

package com.example.mellowsphere;

import android.app.Dialog;
import android.content.Context;

```

```

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

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

import com.google.android.material.textfield.TextInputLayout;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.firestore.CollectionReference;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.Query;

public class ForgotPassword extends DialogFragment {
    EditText email;
    TextInputLayout emailInputLayout;
    Button submitEmail;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
container, @Nullable Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.activity_forgot_password,
container, false);

        emailInputLayout = view.findViewById(R.id.TILEmail);
        email = emailInputLayout.getEditText();
        submitEmail = view.findViewById(R.id.submit_forgot_pass);

        submitEmail.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Retrieve the email address from the TextInputEditText
                String emailAddress = email.getText().toString().trim();

                // Validate the email address if needed
                if (isValidEmail(emailAddress)) {
                    // TODO: Send "forgot password" email
                    sendForgotPasswordEmail(emailAddress);
                } else {
                    Toast.makeText(getContext(), "Invalid email address",
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // Set touch listener on the root view

```

```

        if (view != null) {
            view.setOnTouchListener(new View.OnTouchListener() {
                @Override
                public boolean onTouch(View v, MotionEvent event) {
                    hideKeyboardAndCursor();
                    return false;
                }
            });
        }

        // You can do any additional setup or UI initialization here
        return view;
    }

    @NonNull
    @Override
    public Dialog onCreateDialog(@Nullable Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        Dialog dialog = super.onCreateDialog(savedInstanceState);
        // Set other dialog properties if needed
        return dialog;
    }

    private boolean isValidEmail(String email) {
        // You can implement your email validation logic here
        return
        android.util.Patterns.EMAIL_ADDRESS.matcher(email).matches();
    }

    private void sendForgotPasswordEmail(String emailAddress) {
        // Create a reference to the Firestore "users" collection
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        CollectionReference usersCollection = db.collection("users");

        // Query to check if the provided email address exists in the
        "users" collection
        Query query = usersCollection.whereEqualTo("email", emailAddress);

        query.get().addOnCompleteListener(task -> {
            if (task.isSuccessful()) {
                // Check if any documents match the query
                if (!task.getResult().isEmpty()) {
                    // Email address found in Firestore "users" collection
                    // Now, send the password reset email using
                    FirebaseAuth
                    FirebaseAuth.getInstance().sendPasswordResetEmail(emailAddress)
                        .addOnCompleteListener(passwordResetTask -> {
                            if (passwordResetTask.isSuccessful()) {
                                // Email sent successfully
                                Toast.makeText(getContext(), "Password
                                reset email sent", Toast.LENGTH_SHORT).show();
                            } else {
                                // If there is an issue with sending
                                the reset email
                                Toast.makeText(getContext(), "Failed to
                                send reset email", Toast.LENGTH_SHORT).show();

```



```

        }
    });
    } else {
        // Email address not found in Firestore "users"
        collection
        Toast.makeText(getContext(), "Email address not found",
        Toast.LENGTH_SHORT).show();
    }
    } else {
        // Error occurred while querying the "users" collection
        Toast.makeText(getContext(), "Error checking email
        address", Toast.LENGTH_SHORT).show();
    }
    });
}

private void hideKeyboardAndCursor() {
    if (getView() == null || getActivity() == null) {
        return; // Fragment not attached to an activity or view is null
    }

    InputMethodManager imm = (InputMethodManager)
    getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);
    View currentFocus = getView().findFocus();
    if (currentFocus != null) {
        imm.hideSoftInputFromWindow(currentFocus.getWindowToken(), 0);
        currentFocus.clearFocus();
    }
}
}

```

CustomRecyclerView.java

```

package com.example.mellowsphere;

import android.content.Context;
import android.util.AttributeSet;
import android.view.MotionEvent;

import androidx.recyclerview.widget.RecyclerView;
import androidx.viewpager2.widget.ViewPager2;

public class CustomRecyclerView extends RecyclerView {
    private ViewPager2 viewPager2;

    public CustomRecyclerView(Context context) {
        super(context);
    }

    public CustomRecyclerView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public CustomRecyclerView(Context context, AttributeSet attrs, int
    defStyle) {

```

```

        super(context, attrs, defStyle);
    }

    public void setViewPager2(ViewPager2 viewPager2) {
        this.viewPager2 = viewPager2;
    }

    @Override
    public boolean onInterceptTouchEvent(MotionEvent e) {
        // Disable ViewPager2 scrolling when RecyclerView is being touched
        if (viewPager2 != null) {
            switch (e.getAction()) {
                case MotionEvent.ACTION_DOWN:
                    viewPager2.setUserInputEnabled(false);
                    break;
                case MotionEvent.ACTION_UP:
                case MotionEvent.ACTION_CANCEL:
                    viewPager2.setUserInputEnabled(true);
                    break;
            }
        }
        return super.onInterceptTouchEvent(e);
    }

    @Override
    public boolean performClick() {
        // Dummy implementation to satisfy lint check
        return super.performClick();
    }
}

```

AudioTrackListViewAdapter.java

```

package com.example.mellowsphere;

import android.content.Context;
import android.os.Handler;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import java.util.List;

public class AudioTrackListViewAdapter extends BaseAdapter {
    List<AudioTrackModel> audioList;
    Context context;

    public AudioTrackListViewAdapter(Context context, List<AudioTrackModel>
audioList) {
        this.audioList = audioList;
        this.context = context;
    }

    public List<AudioTrackModel> getAudioList() {

```

```

        return audioList;
    }

    public void setAudioList(List<AudioTrackModel> audioList) {
        this.audioList = audioList;
    }

    public Context getContext() {
        return context;
    }

    public void setContext(Context context) {
        this.context = context;
    }

    @Override
    public int getCount() {
        return audioList.size();
    }

    @Override
    public Object getItem(int position) {
        return position;
    }

    @Override
    public long getItemId(int position) {
        return position;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = LayoutInflater.from(context);
        convertView = inflater.inflate(R.layout.audio_listview_layout,
parent, false);
        TextView trackname = convertView.findViewById(R.id.trackname);
        TextView artistname = convertView.findViewById(R.id.artistName);
        ImageView download = convertView.findViewById(R.id.download);

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                trackname.setSelected(true);
            }
        }, 2000);

        trackname.setText(audioList.get(position).getTitle());
        artistname.setText(audioList.get(position).getArtist());

        return convertView;
    }
}

```

AudioPlayer.java

```
package com.example.mellowsphere;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.app.DownloadManager;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.pm.PackageManager;
import android.content.res.ColorStateList;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.os.IBinder;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QueryDocumentSnapshot;
import com.google.firebase.firestore.QuerySnapshot;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

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

public class AudioPlayer extends AppCompatActivity {
    Handler seekBarHandler;
    boolean islooped = false;
    Button playOrPause, btnForward, btnBack, loopBtn;
    String url, track, artist, album, localFilePath;
    List<AudioTrackModel> audioTrackModelList = new ArrayList<>();
    ImageView download;
    TextView trackname, artistname, numberOfFavs, trackDuration,
    currentTime;
    StorageReference storageReference;
```

```

SeekBar seekBar;
Button favourite;
boolean isFavourited = false;
private AudioPlayerService mediaPlayerService;
boolean isUserPremium = false;
FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
FirebaseFirestore db = FirebaseFirestore.getInstance();
private ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service)
{
    AudioPlayerService.LocalBinder binder =
(AudioPlayerService.LocalBinder) service;
    mediaPlayerService = binder.getService();
    // Now you have a reference to your service

    init();
}

    @Override
    public void onServiceDisconnected(ComponentName name) {
        mediaPlayerService = null;
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    initPlaybackService();

    setContentView(R.layout.activity_audio_player);

}

public void init(){
    seekBarHandler = new Handler();

    playOrPause = findViewById(R.id.playorpause);
    playOrPause.setBackgroundResource(R.drawable.ic_pause);

    btnBack = findViewById(R.id.btnback);
    btnForward = findViewById(R.id.btnforward);

    favourite = findViewById(R.id.favouriteaudioplayer);

    loopBtn = findViewById(R.id.loopbtn);

    seekBar = findViewById(R.id.seekBarTime);
    seekBar.setProgress(0);

    download = findViewById(R.id.download);

    trackname = findViewById(R.id.trackname);
    artistname = findViewById(R.id.artistname);
    trackDuration = findViewById(R.id.tvDuration);

```

```

currentTime = findViewById(R.id.currDuration);
numberOfFavs = findViewById(R.id.numberoffavs);

Bundle b = getIntent().getExtras();
if (b != null) {
    track = b.getString("TrackName");
    artist = b.getString("ArtistName");
    album = b.getString("AlbumName");
    url = b.getString("AudioUrl");
    isUserPremium = b.getBoolean("IsPremium");
    audioTrackModelList = b.getParcelableArrayList("TrackList");

    trackname.setText(track);
    artistname.setText(artist);
} else {
    finish();
}

// Using a handler to delay the start of the scrolling
Handler handler = new Handler();
handler.postDelayed(() -> {
    trackname.setSelected(true);
    artistname.setSelected(true);

    if (isAudioFileDownloaded(track+".mp3")) {
        Toast.makeText(this, "Playing offline",
Toast.LENGTH_SHORT).show();

        playAudio(localFilePath);
        download.setVisibility(View.GONE);

        if
(!NetworkUtils.isNetworkAvailable(getApplicationContext())) {
            playOrPause.setOnClickListener(v -> {
                if (mediaPlayerService.isPlaying()) {
                    mediaPlayerService.pauseAudio();

playOrPause.setBackgroundResource(R.drawable.ic_play);
                }
                else {
                    playAudio(localFilePath);

playOrPause.setBackgroundResource(R.drawable.ic_pause);
                }
            });
        }
    }
}, 2000);

seekBarHandler.postDelayed(updateSeekBar, 100);

storageReference =
FirebaseStorage.getInstance().getReference().child(album.toLowerCase()+ "/" +
track+".mp3");

```

```

storageReference.getDownloadUrl().addOnSuccessListener((OnSuccessListener<U
ri>) uri -> {
    String link = uri.toString();

    // Download the audio file using DownloadManager
    if (!isAudioFileDownloaded(track+".mp3")){
        Toast.makeText(AudioPlayer.this, "Playing online",
Toast.LENGTH_SHORT).show();
        playAudio(link);
        download.setVisibility(View.VISIBLE);
    }
    else {
        link = localFilePath;
    }

    download.setOnClickListener(v -> {
        if (user == null) {
            Intent i = new Intent(getApplicationContext(),
Register.class);
            startActivity(i);
            Toast.makeText(AudioPlayer.this, "Login required",
Toast.LENGTH_SHORT).show();
        }
        else {
            if (isUserPremium) {
                downloadAudio(uri.toString(), track+".mp3");
                download.setVisibility(View.GONE);
            }
            else {
                displayPremiumOption();
            }
        }
    });

    String finalLink = link;
    playOrPause.setOnClickListener(v -> {
        if (mediaPlayerService.isPlaying()) {
            mediaPlayerService.pauseAudio();
            playOrPause.setBackgroundResource(R.drawable.ic_play);
        }
        else {
            playAudio(finalLink);
            playOrPause.setBackgroundResource(R.drawable.ic_pause);
        }
    });

});

btnBack.setOnClickListener(v -> {
    for (int i = 0; i < audioTrackModelList.size(); i++) {
        if (audioTrackModelList.get(i).getTitle().equals(track)){
            try {
                Intent i2 = new Intent(getApplicationContext(),
AudioPlayer.class);

```

```

        i2.putExtra("TrackName", audioTrackModelList.get(i-1).getTitle());
        i2.putExtra("ArtistName", audioTrackModelList.get(i-1).getArtist());
        i2.putExtra("AlbumName", audioTrackModelList.get(i-1).getAlbum());
        i2.putExtra("TrackList", new ArrayList<>(audioTrackModelList));
        i2.putExtra("IsPremium", isUserPremium);

        finish();

        startActivity(i2);
    }
    catch (IndexOutOfBoundsException e) {
        Intent i2 = new Intent(getApplicationContext(),
AudioPlayer.class);

        i2.putExtra("TrackName", audioTrackModelList.get(audioTrackModelList.size()-1).getTitle());
        i2.putExtra("ArtistName", audioTrackModelList.get(audioTrackModelList.size()-1).getArtist());
        i2.putExtra("AlbumName", audioTrackModelList.get(audioTrackModelList.size()-1).getAlbum());
        i2.putExtra("TrackList", new ArrayList<>(audioTrackModelList));
        i2.putExtra("IsPremium", isUserPremium);

        finish();

        startActivity(i2);
    }
}

});

btnForward.setOnClickListener(v -> {
    for (int i = 0; i < audioTrackModelList.size(); i++) {
        if (audioTrackModelList.get(i).getTitle().equals(track)) {
            try {
                Intent i2 = new Intent(getApplicationContext(),
AudioPlayer.class);

                i2.putExtra("TrackName", audioTrackModelList.get(i+1).getTitle());
                i2.putExtra("ArtistName", audioTrackModelList.get(i+1).getArtist());
                i2.putExtra("AlbumName", audioTrackModelList.get(i+1).getAlbum());
                i2.putExtra("TrackList", new ArrayList<>(audioTrackModelList));
                i2.putExtra("IsPremium", isUserPremium);

                startActivity(i2);

                finish();
            }

```



```

        catch (IndexOutOfBoundsException e) {
            Intent i2 = new Intent(getApplicationContext(),
AudioPlayer.class);

                i2.putExtra("TrackName",
audioTrackModelList.get(0).getTitle());
                i2.putExtra("ArtistName",
audioTrackModelList.get(0).getArtist());
                i2.putExtra("AlbumName",
audioTrackModelList.get(0).getAlbum());
                i2.putExtra("TrackList", new
ArrayList<>(audioTrackModelList));
                i2.putExtra("IsPremium", isUserPremium);

                startActivity(i2);

                finish();
            }
        }
    });

    loopBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            islooped = !islooped;
            mediaPlayerService.setLoop(islooped);
            if (islooped) {
                setBackgroundTint(loopBtn, R.color.loop);
            }
            else {
                setBackgroundTint(loopBtn, R.color.white);
            }
        }
    });

    if (user != null) {
        favourite.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                isFavourited = !isFavourited;
                if (isFavourited) {
                    setBackgroundTint(favourite, R.color.favourite);

                    // Query the document for the track
                    db.collection("alltracks").whereEqualTo("title",
track).get()

                                .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                                    @Override
                                    public void onComplete(@NonNull
Task<QuerySnapshot> task) {
                                        if (task.isSuccessful()) {
                                            for (QueryDocumentSnapshot document
: task.getResult()) {
                                                // Retrieve the existing
favouritedby field

```

```

        ArrayList<String> favouritedBy
= (ArrayList<String>) document.get("favouritedby");

        // Check if it's null, and
        initialize it if necessary
        if (favouritedBy == null) {
            favouritedBy = new
ArrayList<>();
        }

        // Check if the user ID is not
        already in the list before adding it
        if
        (!favouritedBy.contains(user.getUid())) {
            favouritedBy.add(user.getUid());

            // Update the document with
            the modified favouritedby field
            ArrayList<String>
            finalFavouritedBy = favouritedBy;

            db.collection("alltracks").document(document.getId())
                .update("favouritedby",
                favouritedBy)

            .addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void
                onSuccess(Void aVoid) {
                    // Update
                    String Str;

                    if
                    Str =
                    finalFavouritedBy.size() > 1) {
                        finalFavouritedBy.size()+" users love this track";
                    }
                    else {
                        Str = "You
                    are the only one who loves this track";
                    }

                    numberOfFavs.setText(Str);

                    Toast.makeText(getApplicationContext(), "Track favourited.",
                    Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
}
});
}

```

```

        else {
            setBackgroundTint(favourite, R.color.white);

            // Query the document for the track
            db.collection("alltracks").whereEqualTo("title",
track).get()

                .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull
Task<QuerySnapshot> task) {
                        if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document
: task.getResult()) {
                                // Retrieve the existing
                                ArrayList<String> favouritedBy
= (ArrayList<String>) document.get("favouritedby");

                                // Check if it's not null and
                                if (favouritedBy != null) {
                                    favouritedBy.remove(user.getUid());

                                    // Update the document with
                                    the modified favouritedby field
                                    db.collection("alltracks").document(document.getId())
                                        .update("favouritedby",
favouritedBy)

                                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                                            @Override
                                            public void
onSuccess(Void aVoid) {
                                                // Update
                                                successful
                                                numberOfFavs.setText("");

                                                Toast.makeText(getApplicationContext(), "Track unfavourited",
                                                Toast.LENGTH_SHORT).show();
                                            }
                                        });
                                    }
                                }
                            }
                        }
                    }
                });
        }
    }
}

```

```

    public void checkIfFavourited() {
        if (user != null && (favourite.getVisibility() == View.GONE)) {
            db.collection("alltracks").whereEqualTo("title", track).get()
                .addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<QuerySnapshot>
task) {
                        if (task.isSuccessful()) {
                            for (QueryDocumentSnapshot document :
task.getResult()) {
                                // Retrieve the existing favouritedby field
                                ArrayList<String> favouritedBy =
                                (ArrayList<String>) document.get("favouritedby");

                                favourite.setVisibility(View.VISIBLE);
                                if (favouritedBy != null) {
                                    if
(favouritedBy.contains(user.getUid())) {
                                        isFavourited = true;
                                        setBackgroundTint(favourite,
R.color.favourite);

                                        // Update successful
                                        // Update successful
                                        String Str;

                                        if (favouritedBy.size() > 1) {
                                            Str = favouritedBy.size()+"
users love this track";

                                            }
                                            else {
                                                Str = "You are the only one who
loves this track";

                                            }

                                            numberOfFavs.setText(Str);

                                        }
                                        else {
                                            isFavourited = false;
                                            setBackgroundTint(favourite,
R.color.white);

                                            numberOfFavs.setText("");
                                        }
                                    }
                                }
                            }
                        }
                    }
                });
        }
    }

    public void setBackgroundTint(Button btn, int colorid) {
        // Create a new color state list for the desired tint color
        int[][] states = new int[][] {

```

```

        new int[] { android.R.attr.state_enabled }, // enabled
        // Add more states if needed, e.g., disabled, pressed, etc.
    };

    int[] colors = new int[] {
        // Color for the enabled state
        ContextCompat.getColor(getApplicationContext(), colorId),
        // Add more colors corresponding to the states above
    };

    ColorStateList newTintList = new ColorStateList(states, colors);

    btn.setBackgroundTintList(newTintList);
}

public static String formatDuration(int durationInMillis) {
    int seconds = (durationInMillis / 1000) % 60;
    int minutes = ((durationInMillis / (1000 * 60)) % 60);

    return String.format(Locale.getDefault(), "%02d:%02d", minutes,
seconds);
}

private void initPlaybackService() {
    Intent serviceIntent = new Intent(this, AudioPlayerService.class);
    startService(serviceIntent);
    bindService(serviceIntent, serviceConnection,
Context.BIND_AUTO_CREATE);
}

private void displayPremiumOption() {
    Toast.makeText(AudioPlayer.this, "Get premium for offline
streaming", Toast.LENGTH_SHORT).show();
    Intent i = new Intent(this, PremiumRegister.class);
    startActivity(i);
}

private void playAudio(String link) {
    mediaPlayerService.playAudio(link);
}

// Method to check if the file is already downloaded
private boolean isAudioFileDownloaded(String fileName) {
    if (!permission()) {
        requestPermission();
        return false;
    }
    else {
        File file = new
File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DO
WNLOADS), fileName);
        if (file.exists()) {
            localFilePath = file.getAbsolutePath();
            return true;
        }
    }
    return false;
}

```

```

    }

    private boolean permission() {
        String permission;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU)
            permission = Manifest.permission.READ_MEDIA_AUDIO;
        else
            permission = Manifest.permission.READ_EXTERNAL_STORAGE;
        int result = ContextCompat.checkSelfPermission(AudioPlayer.this,
permission );
        return result == PackageManager.PERMISSION_GRANTED;
    }

    private void requestPermission() {
        String permission;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU)
            permission = Manifest.permission.READ_MEDIA_AUDIO;
        else
            permission = Manifest.permission.READ_EXTERNAL_STORAGE;
        ActivityCompat.requestPermissions(AudioPlayer.this, new
String[]{permission}, 123);
    }

    private void downloadAudio(String url, String fileName) {
        DownloadManager downloadManager = (DownloadManager)
getSystemService(DOWNLOAD_SERVICE);

        Uri uri = Uri.parse(url);

        DownloadManager.Request request = new DownloadManager.Request(uri);

        // Set the destination directory and filename for the downloaded
file

request.setDestinationInExternalPublicDir(Environment.DIRECTORY_DOWNLOADS,
fileName);

        // Optional: Set other parameters for the download request
        // For example, you can set the notification visibility

request.setNotificationVisibility(DownloadManager.Request.VISIBILITY_VISIBL
E_NOTIFY_COMPLETED);

        // Enqueue the download request
        downloadManager.enqueue(request);

        Toast.makeText(this, "Downloading "+fileName,
Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onBackPressed() {

        super.onBackPressed();
    }

    private final Runnable updateSeekBar = new Runnable() {

```

```

        @Override
        public void run() {
            if (mediaPlayerService.getMediaPlayer() != null &&
mediaPlayerService.isPlaying()) {
                seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
                    @Override
                    public void onProgressChanged(SearchBar seekBar, int
progress, boolean fromUser) {
                        if (fromUser) {
                            mediaPlayerService.seek(progress);
                        }
                    }

                    @Override
                    public void onStartTrackingTouch(SearchBar seekBar) {
                    }

                    @Override
                    public void onStopTrackingTouch(SearchBar seekBar) {
                    }
                });
            }

seekBar.setMax(mediaPlayerService.getMediaPlayer().getDuration());

            int currentPosition =
mediaPlayerService.getMediaPlayer().getCurrentPosition();
            seekBar.setProgress(currentPosition);

            int durationInMillis =
mediaPlayerService.getMediaPlayer().getDuration();
            String formattedDuration =
formatDuration(durationInMillis);

            trackDuration.setText(formattedDuration);

            String formatCurrPos = formatDuration(currentPosition);
            currentTime.setText(formatCurrPos);
        }

        checkIfFavourited();

        // Repeat the update after a short delay
        seekBarHandler.postDelayed(this, 100);
    }
};

public void back(View view) {
    finish();
}
}

```

AudioPlayerService.java

```

package com.example.mellowsphere;

```

```

import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.Binder;
import android.os.Build;
import android.os.IBinder;
import android.widget.Toast;

import androidx.core.app.NotificationCompat;

import java.io.IOException;

public class AudioPlayerService extends Service implements
    MediaPlayer.OnPreparedListener,
    MediaPlayer.OnErrorListener, MediaPlayer.OnInfoListener{
    IBinder binder = new LocalBinder();
    private static final int NOTIFICATION_ID = 1; // Unique ID for the
notification
    private static final String CHANNEL_ID = "AudioPlayerChannel"; //
Notification channel ID
    private MediaPlayer mediaPlayer = new MediaPlayer();
    private String url = "";
    private int pausePosition;

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        return binder;
    }

    public class LocalBinder extends Binder {
        public AudioPlayerService getService() {
            return AudioPlayerService.this;
        }
    }

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

        mediaPlayer = new MediaPlayer();

        mediaPlayer.setOnPreparedListener(this);
        mediaPlayer.setOnErrorListener(this);
        mediaPlayer.setOnInfoListener(this);

        createNotification();
    }

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

```



```

    public void playAudio(String url) {
        playAudio(url, false);
    }

    public void playAudio(String url, boolean loop) {
        if (this.url.equals(url) && !mediaPlayer.isPlaying()) {
            resumeAudio();
            return;
        } else {
            this.url = url;
            mediaPlayer.reset();
        }

        if (!mediaPlayer.isPlaying()) {
            try {
                mediaPlayer.setDataSource(url);
                mediaPlayer.setLooping(loop);
                mediaPlayer.prepareAsync();
            }
            catch (IOException e) {
                Toast.makeText(this, "Error"+e.getMessage(),
                    Toast.LENGTH_SHORT).show();
                e.printStackTrace();
            }
        }
    }

    public MediaPlayer getMediaPlayer() {
        return mediaPlayer;
    }

    public void seek(int pos) {
        mediaPlayer.seekTo(pos);
    }

    public void setLoop(boolean loop) {
        mediaPlayer.setLooping(loop);
    }

    public boolean isPlaying() {
        return mediaPlayer != null && mediaPlayer.isPlaying();
    }

    public void pauseAudio() {
        if (mediaPlayer != null && mediaPlayer.isPlaying()) {
            mediaPlayer.pause();
            pausePosition = mediaPlayer.getCurrentPosition();
        }
    }

    public void resumeAudio() {
        if (mediaPlayer != null && !mediaPlayer.isPlaying()) {
            mediaPlayer.seekTo(pausePosition);
            mediaPlayer.start();
        }
    }

    @Override

```

```

    public void onDestroy() {
        super.onDestroy();

        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
        }
        mediaPlayer.release();
    }

    @Override
    public boolean onError(MediaPlayer mp, int what, int extra) {
        switch (what){
            case
MediaPlayer.MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK:
                Toast.makeText(this,
"MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK",
Toast.LENGTH_SHORT).show();

                case MediaPlayer.MEDIA_ERROR_SERVER_DIED:
                    Toast.makeText(this, "MEDIA_ERROR_SERVER_DIED",
Toast.LENGTH_SHORT).show();

                case MediaPlayer.MEDIA_ERROR_UNKNOWN:
                    Toast.makeText(this, "MEDIA_ERROR_UNKNOWN",
Toast.LENGTH_SHORT).show();
                }

            return false;
        }

    @Override
    public boolean onInfo(MediaPlayer mp, int what, int extra) {
        return false;
    }

    @Override
    public void onPrepared(MediaPlayer mp) {
        if (!mp.isPlaying()) {
            mp.start();
        }
    }

    private void createNotification() {
        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this)
            .setContentTitle("Audio Player")
            .setContentText("Playing in the background")
            .setSmallIcon(R.drawable.baseline_person_24);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(
                CHANNEL_ID,
                "Audio Player Service",
                NotificationManager.IMPORTANCE_DEFAULT
            );

```

```

        NotificationManager manager =
getSystemService(NotificationManager.class);
        manager.createNotificationChannel(channel);
        builder.setChannelId(CHANNEL_ID);
    }

    Notification notification = builder.build();

    // Set the service as a foreground service
    startForeground(NOTIFICATION_ID, notification);
}
}

```

AlbumModelForRecycler.java

```

package com.example.mellowsphere;

public class AlbumModelForRecycler {
    private int imageResource; // Change the type to String
    private String albumName;
    private String artistName;

    public AlbumModelForRecycler(int imageResource, String albumName,
String artistName) {
        this.imageResource = imageResource;
        this.albumName = albumName;
        this.artistName = artistName;
    }

    public int getCover() {
        return imageResource;
    }

    public void setCoverUrl(int imageResource) {
        this.imageResource = imageResource;
    }

    public String getAlbumName() {
        return albumName;
    }

    public void setAlbumName(String albumName) {
        this.albumName = albumName;
    }

    public String getArtistName() {
        return artistName;
    }

    public void setArtistName(String artistName) {
        this.artistName = artistName;
    }
}

```

AudioTrackModel.java

```
package com.example.mellowsphere;

import android.os.Parcel;
import android.os.Parcelable;

import androidx.annotation.NonNull;

public class AudioTrackModel implements Parcelable {
    private String title;
    private String artist;
    private String genre;
    private String album;

    public AudioTrackModel(String title, String artist, String genre,
String album) {
        this.title = title;
        this.artist = artist;
        this.genre = genre;
        this.album = album;
    }

    protected AudioTrackModel(Parcel in) {
        title = in.readString();
        artist = in.readString();
        genre = in.readString();
        album = in.readString();
    }

    public static final Creator<AudioTrackModel> CREATOR = new
Creator<AudioTrackModel>() {
        @Override
        public AudioTrackModel createFromParcel(Parcel in) {
            return new AudioTrackModel(in);
        }

        @Override
        public AudioTrackModel[] newArray(int size) {
            return new AudioTrackModel[size];
        }
    };

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getArtist() {
        return artist;
    }

    public void setArtist(String artist) {
        this.artist = artist;
    }
}
```

```

    }

    public String getGenre() {
        return genre;
    }

    public void setGenre(String genre) {
        this.genre = genre;
    }

    public String getAlbum() {
        return album;
    }

    public void setAlbum(String album) {
        this.album = album;
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(@NonNull Parcel dest, int flags) {
        dest.writeString(title);
        dest.writeString(artist);
        dest.writeString(genre);
        dest.writeString(album);
    }
}

```

NetworkUtils.java

```

package com.example.mellowsphere;

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

public class NetworkUtils {

    public static boolean isNetworkAvailable(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager)
context.getSystemService(Context.CONNECTIVITY_SERVICE);
        if (connectivityManager != null) {
            NetworkInfo activeNetworkInfo =
connectivityManager.getActiveNetworkInfo();
            return activeNetworkInfo != null &&
activeNetworkInfo.isConnected();
        }
        return false;
    }
}

```

SharedViewModel.java

```
package com.example.mellowsphere;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;
import androidx.viewpager2.widget.ViewPager2;

public class SharedViewModel extends ViewModel {
    private MutableLiveData<ViewPager2> viewPager2 = new
MutableLiveData<>();
    private MutableLiveData<AlbumModelForRecycler> selectedAlbum = new
MutableLiveData<>();

    public void onAlbumItemClick(AlbumModelForRecycler album) {
        selectedAlbum.setValue(album);
    }

    public void setViewPager2(ViewPager2 viewPager2) {
        this.viewPager2.setValue(viewPager2);
    }

    public LiveData<ViewPager2> getViewPager2() {
        return viewPager2;
    }

    public LiveData<AlbumModelForRecycler> getSelectedAlbum() {
        return selectedAlbum;
    }
}
```

ViewPagerAdapter.java

```
package com.example.mellowsphere;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.lifecycle.Lifecycle;
import androidx.viewpager2.adapter.FragmentStateAdapter;

public class ViewPagerAdapter extends FragmentStateAdapter {

    public ViewPagerAdapter(@NonNull FragmentManager fragmentManager,
@NonNull Lifecycle lifecycle) {
        super(fragmentManager, lifecycle);
    }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        switch (position) {
            case 1:
                return new PlaylistTabItemFragment();
            default:

```

```

        return new AllTabItemFragment();
    }

    @Override
    public int getItemCount() {
        // Return the total number of fragments
        return 2; // Adjust based on the actual number of fragments
    }
}

```

MellowSphere.java

```

package com.example.mellowsphere;

import android.app.Application;

import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreSettings;

public class MellowSphere extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        // Configure Firestore settings with persistence enabled
        FirebaseFirestore firestore = FirebaseFirestore.getInstance();
        FirebaseFirestoreSettings settings = new
        FirebaseFirestoreSettings.Builder()
            .setPersistenceEnabled(true)
            .build();
        firestore.setFirestoreSettings(settings);
    }
}

```