Mobile App Development
Final Exam

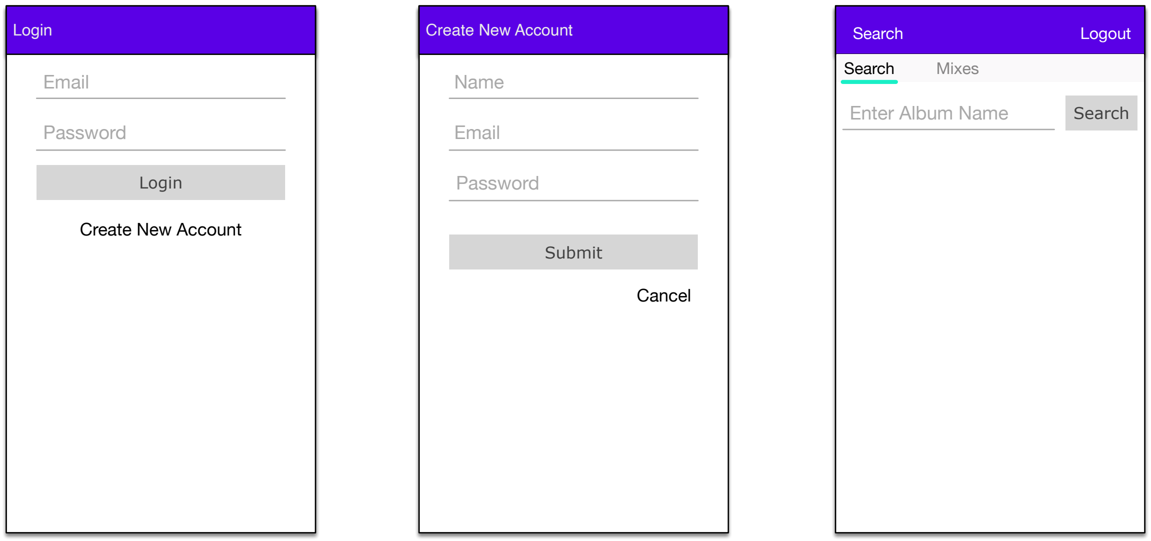
Basic Instructions:

1. This is the Final Exam, which will count for 20% of the total course grade.
2. This Final is an individual effort. Each student is responsible for her/his own Final and its submission.
3. Once you have picked up the exam, you may not discuss it in any way with anyone until the exam period is over.
4. During the exam, you are allowed to use the course videos, slides, and your code from previous home works and in class assignments. You can use the internet to search for answers. You are NOT allowed to use code provided by other students or solicit help from other online persons.
5. Answer all the exam parts, all the parts are required.
6. Please download the support files provided with the Final and use them when implementing your project.
7. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
8. Create a zip file which includes all the project folder, any required libraries, and your presentation material. Submit the exported file using the provided canvas submission link.
9. **Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
10. **Failure to follow the above instructions will result in point deductions.**
11. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

<u>Final Exam (100 Points)</u>

This app will provide a music preview application that uses Firebase Firestore to manage data storage and the Deezer API to retrieve albums and track information. This application should be implemented using activities/fragments as indicated below:

- **Auth Activity:** this is the launcher activity and hosts the following fragments:
  - Login Fragment and SignUp Fragment
- **Main Activity:** which hosts the following:
  - ViewPager and TabLayout, that will serve the Search and Mixes Fragments.
- **Album Activity:** which hosts the following:
  - Album Fragment, Add Track Fragment and AudioPlayer Fragment
- **Create New Mix Activity:** which displays the create new mix form.
- **Mix Activity:** which hosts the following:
  - Mix Details Fragment, Mix Sharing Fragment and AudioPlayer Fragment



(a) Login Fragment (b) SignUp Fragment (c) Main Activity (Search)

Figure 1, Application Wireframe

Part 1: (Auth Activity) Authentication and SignUp (5 Points)

The Auth Activity is the launcher activity and below are the requirements:

1. If the user is already logged-in then the Main Activity should be started and the Auth Activity should be finished.
2. If the user is not logged-in then the Login Fragment should be displayed in the Auth Activity. Login Fragment (Shown in Fig 1(a)):
   a. Clicking "Login" button, if all the inputs are entered correctly, you should attempt to login the user using Firebase. If the login is successful then trigger the Auth Activity to start the Main Activity and then finish the Auth Activity. If login is not successful show an alert dialog highlighting the error.
      i. If there is missing input, show an alert dialog indicating missing input.
   b. Clicking the "Create New Account" should replace this fragment with the SignUp Fragment.
3. SignUp Fragment (Shown in Fig 1(b)):
   a. Clicking the "Submit" button, if all the inputs are entered correctly, you should

attempt to signup the user using Firebase. If the signup is successful then trigger the Auth Activity to start the Main Activity and then finish the Auth Activity. If signup is not successful show an alert dialog highlighting the error.
        i. If there is missing input show an alert dialog indicating the error.
    b. Clicking "Cancel" should replace this fragment with the Login Fragment.

Part 2: Main Activity (5 Points)
This activity should include a ViewPager and TabLayout as shown in Fig 1(c), this should allow the user to pick a tab item or swipe between the tab items. The requirements are listed below:

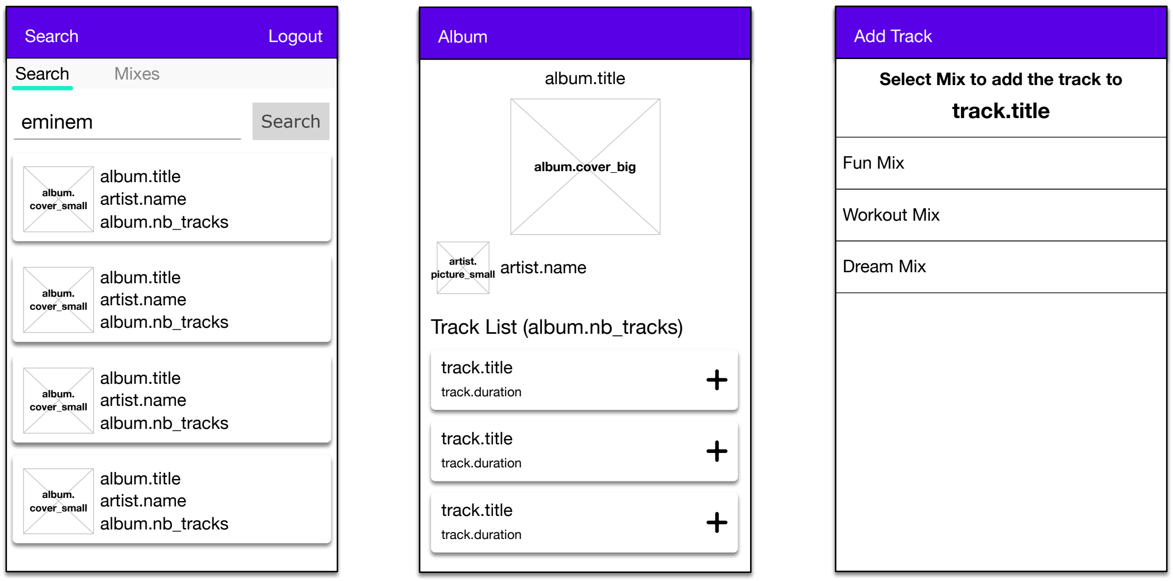
1. The TabLayout should include and serve the following options:
    a. Search: displays the Search Fragment.
    b. Mixes: displays the list of mixes created by and shared with the logged in user.
2. The activity should display a "Logout" menu item as shown in Fig 1(c). Clicking the "Logout" menu button should:
    a. Logout the currently logged in user.
    b. Then start the Auth Activity and then finish the Main Activity.

Part 3: (Main Activity) Search Fragment (15 Points)
This fragment allows the user to search albums using the Deezer api. The requirements are listed below:

1. The Search Fragment is shown in Fig 1(c). The fragment provides an EditText for the user to enter the album name to search for. You should integrate OkHTTP library into your project and use it to make the API requests.
2. Upon clicking the "Search" Button
    a. If the EditText is not empty then make a HTTP request to retrieve the album search results returned by the Deezer album search api which will be returned in JSON format. The details of the Deezer album search api is provided below.

Deezer Album Search API	
URL	https://api.deezer.com/search/album?q=eminem
Method	GET
Params	q: which should contain the search keyword that was entered in the EditText
Result	JSON result, which contains a list of album objects that match the provided search criteria.

    b. The HTTP request should be done in the background or use the OkHTTP asynchronous feature. The returned JSON album search results should be parsed and displayed in a RecyclerView as shown in Fig 2(a).
    c. Each row item should display the album image (cover_small), album title, artist name and album number of tracks (nb_tracks).
3. Upon reentering a new search keyword and pressing "Search" the old search results should be cleared and the new results should be displayed.
4. Clicking an album row item should trigger the MainActivity to start the Album Activity to display the album details. Note that the intent starting the Album Activity should include the album information for the album that should be displayed.



(a) Search Fragment (b) Album Fragment (c) Add Track Fragment

Figure 2, Application Wireframe

Part 4: (Album Activity) Album Fragment (15 Points)

This fragment displays the album details, and the album tracks. The requirements are listed below:

1. This fragment should be displayed when the Album Activity starts. The activity should pass on the received album information to the Album Fragment. You should consider creating an Album class to store album information and allow the passing of data between the different screens.
2. The Album Fragment is shown in Fig 2(b). This fragment should display the album details such as the album title, album image (cover_big), artist name, artist image (picture_small), album number of tracks (nb_tracks) and the list of album tracks.
3. The list of album tracks should be retrieved using the Deezer album tracks API, which will return a list of album tracks in JSON format. The details of the Deezer album tracks api is provided below.

Deezer Album Tracks API	
URL	https://api.deezer.com/album/{album_id}/tracks For example for album_id 1401302 the url is https://api.deezer.com/album/1401302/tracks
Method	GET
Params	No parameters, need to provide the album_id as part of the URL as shown above.
Result	JSON result, which contains a list of album track objects.

4. The HTTP request should be done in the background or use the OkHTTP asynchronous feature. The returned JSON album tracks result should be parsed and displayed in a RecyclerView as shown in Fig 2(b). Where each item should display the track title, track duration (MM:SS, minutes and then seconds) and the “Plus” button as shown in Fig 2(b).

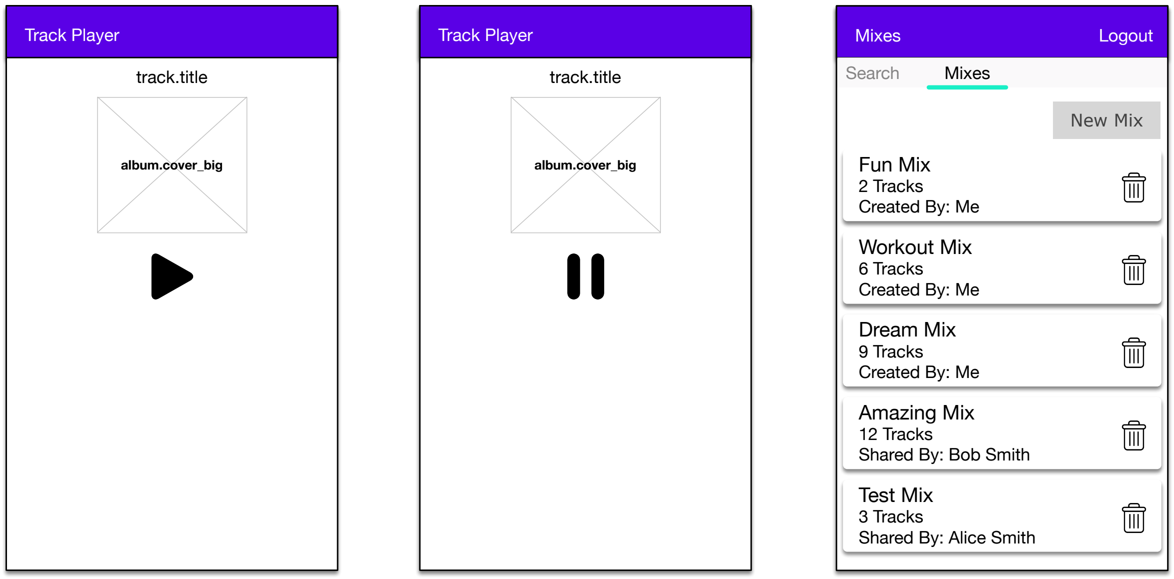
5. ***Note that, each track item includes an mp3 url provided by the preview attribute in the returned JSON. This mp3 url is required by the AudioPlayer Fragment to play the audio track.***
6. Clicking the “Plus” button should replace the current fragment with the “Add Track” fragment, and put the current fragment on the back stack. It should also send the track information to the “Add Track” fragment.
7. Clicking the track row item should replace the current fragment with the “AudioPlayer” fragment, and put the current fragment on the back stack. It should also send the track information to the “AudioPlayer” fragment.
8. Pressing the back button from the “Album” fragment should go back to Main Activity.

Part 5: (Album Activity) Add Track Fragment (5 Points)
This fragment displays the list of mixes created by the currently logged in user. *A mix is a group of tracks that are grouped by the user.* This fragment enables the user to pick a mix to add the selected track to it. The requirements are listed below:
1. The Add Track fragment is shown in Fig 2(c). The fragment shows the track title, and the list of mix names for the currently logged in user retrieved from Firestore.
2. When a mix name is clicked then the track should be added to the selected mix by adding the selected track to the current user’s selected mix collection through Firestore. Then back stack should be popped to should show the Album Fragment.

Part 6: AudioPlayer Fragment (10 Points)
This fragment displays the track information and enable the playback of the selected audio track. The requirements are listed below:
1. Fig 3(a) shows this Fragment, which shows the track title and the album image.
2. Clicking the “Play” button should play the mp3 url for the selected track using the mp3 URL returned in the result (preview attribute). For help in loading and playing mp3 url using android check https://developer.android.com/guide/topics/media/mediaplayer
   a. Once the mp3 starts playing the Play button should be updated to show the “Pause” button as shown in Fig 3(b).
   b. If the user presses the “Pause” button then the audio should be stopped and the “Pause” button should be changed to “Play” button.
   c. Once the mp3 file completes playing then the “Pause” button should be changed back to “Play” button.
3. Pressing the back button should pop the back stack, and make sure to pause the playing track before popping the back stack.

Part 7: (Main Activity) Mixes Fragment (15 Points)
This fragment displays the list of mixes both created by and shared with the currently logged in user. The requirements are listed below:
1. Fig 3(c) shows the list of mixes. Each row item should display the name of the mix, the number of tracks currently added to this mix, and “Created By: Me” if the currently logged in users created this mix, or “Shared By:” followed by the name of the users who shared this mix with the currently logged in user.
2. This mix list should be retrieved from Firestore and should include all the mixes that were created by and shared with the currently logged in user.

   a. Clicking on the “Trash” icon, should delete the selected mix from Firestore and should reload the displayed RecyclerView to show the updated list of mixes.
   b. Clicking on a row item should trigger the Main Activity to start the Mix Activity to display the mix details. Note that the intent staring the Mix Activity should include the mix information for the mix that should be displayed.
3. Clicking the “New Mix” button should trigger the Main Activity to start “Create New Mix” Activity.



(a) Track Player (Not Playing) (b) Track Player (Playing) (c) Mixes Fragment

Figure 3, Application Wireframe

Part 8: Create New Mix Activity (10 Points)

This activity enables the currently logged in user to create a new mix. The requirements are listed below:

1. Fig 4(a) shows the form for the user to enter the new mix name to be created.
   a. Clicking “Submit” button, if name input is not empty, the new mix should be created on Firestore. Upon successful completion, should finish the current activity, which should show the previously displayed “Mixes” fragment.
      - Upon returning to the “Mixes” Fragment, the list should be refreshed to show the updated list of mixes.
      - If there is missing input or any errors, show an alert dialog indicating the error or missing input.
   b. Clicking the “Cancel” should finish the current activity, which should show the previously displayed “Mixes” fragment.

Part 9: (Mix Activity) Mix Details Fragment (15 Points)

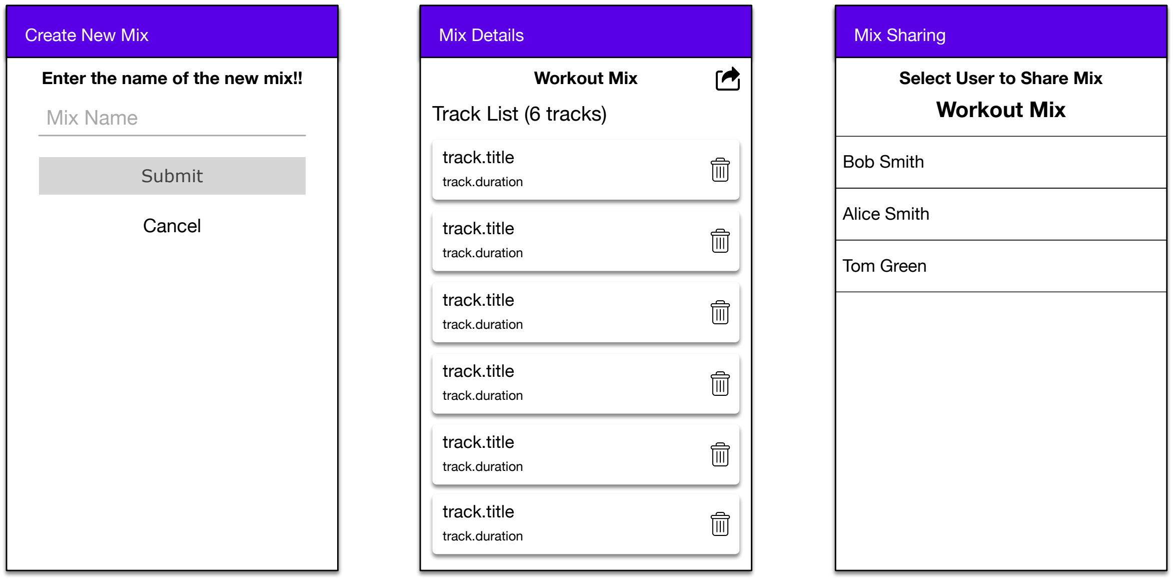
This fragment enables the currently logged in user to view the mix details and the list of tracks added to this mix. The requirements are listed below:

1. This fragment should be displayed when the Mix Activity starts. The activity should pass on the received mix information to the Mix Details Fragment. Create a Mix class

to store mix information and allow the passing of data between the different screens.

2. Fig 4(b) shows the mix details and the list of tracks associated with the selected mix. Each row item should display the track title and track duration. This track list should be retrieved from Firestore and show show the tracks that were added to this mix.
   a. Clicking on the “Trash” icon, should delete the selected track from this mix on Firestore and reload the displayed RecyclerView to show the updated track list.
   b. Clicking the track row item should replace the current fragment with the “AudioPlayer” fragment, and put the current fragment on the back stack. It should also send the track information to the “AudioPlayer” fragment.
3. Clicking the “Share” button at the top right corner of the screen, should replace the current fragment with the “Mix Sharing” fragment, and put the current fragment on the back stack. It should also send the mix information to the “Mix Sharing” fragment.
4. Pressing the back button should finish the current activity which should show the previously displayed “Mixes” Fragment, which should reload the list in the “Mixes” fragment to show the updated list of mixes.

Part 10: (Mix Activity) Mix Sharing Fragment (5 Points)
This fragment displays the name of the selected mix, and the list of users to enable the selection of the user to share the mix with. The requirements are listed below:
1. The Album Sharing fragment is shown in Fig 4(c). The fragment shows the mix name which will be shared, and the list of users on Firebase. You should consider creating a collection of users on Firestore to facilitate the listing of users.
2. When the user row item is clicked then the mix should be shared with the selected user through Firestore. Then the back stack should be popped which should show the “Mix Details” Fragment.



(a) Create New Mix Activity (b) Mix Details (c) Mix Sharing

Figure 4, Application Wireframe

Grading Key

Features	Grade
Part 1: Login, Registration and the Activity transitions developed correctly.	5
Part 2: Main Activity - ViewPager, TabLayout, and Logout developed correctly	5
Part 3: Search Fragment - OKHttp and JSON parsing	10
Part 3: Search Fragment - UI (RecyclerView and other UI)	5
Part 4: Album Fragment - OKHttp and JSON parsing	10
Part 4: Album Fragment - UI (RecyclerView and other UI)	5
Part 5: Add Track Fragment - Firestore and UI	5
Part 6: Audio Player Fragment - Audio Playback and UI Management	10
Part 7: Mixes Fragment - Firestore and UI Management	10
Part 7: Mixes Fragment - Delete Mix	5
Part 8: Create New Mix Activity - UI and Firestore	10
Part 9: Mix Details Fragment - Firestore and UI Management	10
Part 9: Mix Details Fragment - Delete Track	5
Part 10: Mix Share Fragment	5
Total	**100**