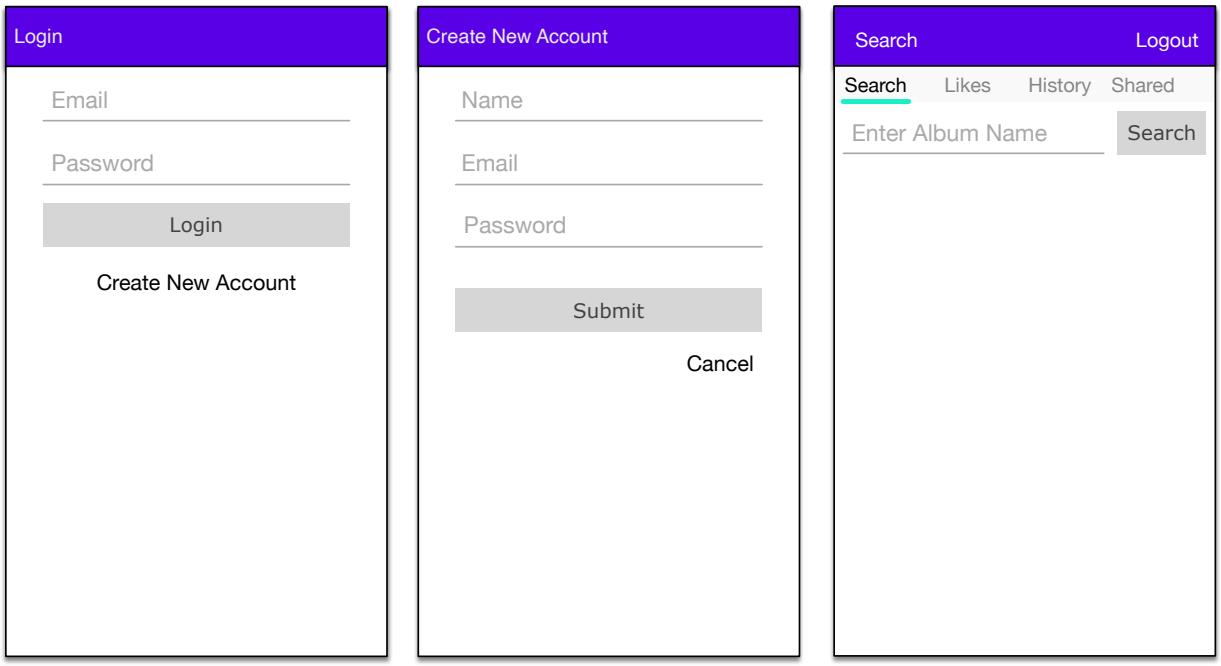
Mobile App Development
Final Exam

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.**

Final Exam (100 Points)

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.



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

Figure 1, Application Wireframe

This application should be implemented using three activities as indicated below:

- **AuthActivity:** this is the launcher activity and hosts the following fragments:
  - Login Fragment
  - SignUp Fragment
- **MainActivity:** which hosts the following:
  - ViewPager and TabLayout, that will serve the Search, Likes, History and Shared Fragments.
- **AlbumActivity:** which hosts the following:
  - Album Fragment
  - Album Sharing Fragment

Part 1: Authentication and SignUp

The AuthActivity is the launcher activity and below are the requirements:

1. If the user is already logged-in then the MainActivity should be started and the AuthActivity should be finished.
2. If the user is not logged-in then the Login Fragment should be displayed in the AuthActivity.
3. Login Fragment (Shown in Figure 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 AuthActivity to start the MainActivity and then finish the AuthActivity. If login is not successful

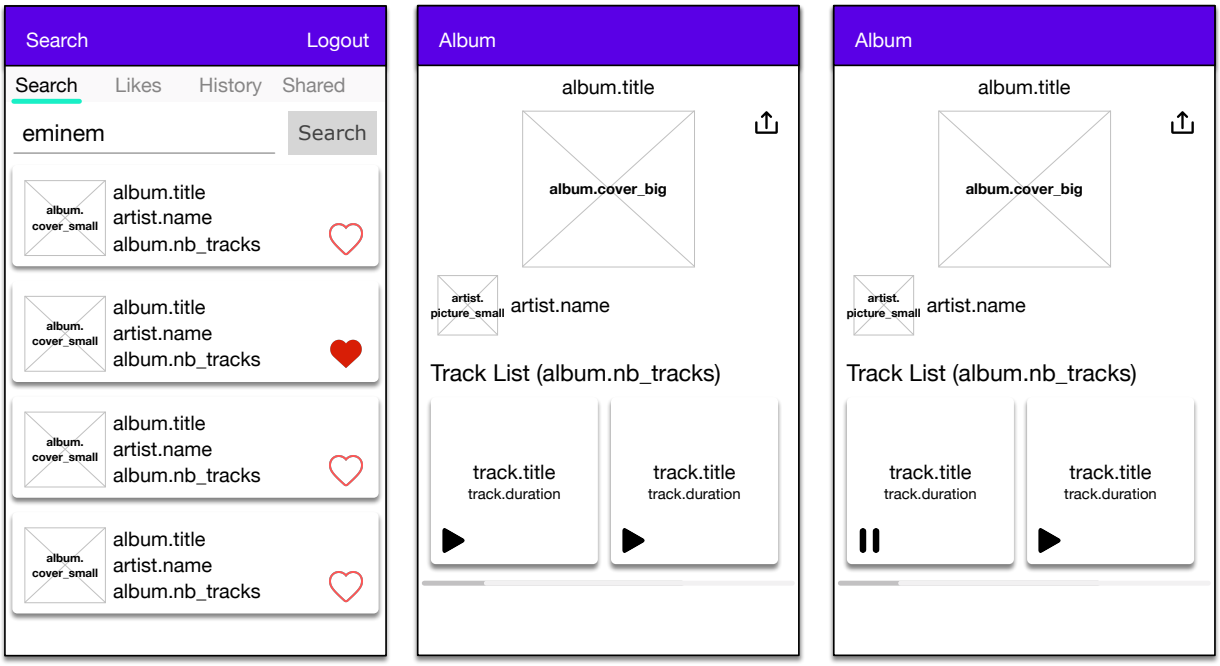
show an alert dialog highlighting the error.
   i. If there is missing input, show an alert dialog indicating missing input.
b. Clicking the “Create New Account” should replace this fragment with the SignUp Fragment.

4. SignUp Fragment (Shown in Figure 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 AuthActivity to start the MainActivity and then finish the AuthActivity. 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

This activity should include a ViewPager and TabLayout as shown in Figure 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. Likes: displays the list of albums liked by the currently logged in user.
   c. History: displays the track play history for the currently logged in user.
   d. Shared: displays the albums that were shared with the currently logged in user.
2. The activity should display a “Logout” menu item as shown in Figure 1(c). Clicking the “Logout” menu button should:
   a. Logout the currently logged in user
   b. Start the AuthActivity and then finish the MainActivity.



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

Figure 2, Application Wireframe

Part 3: Search Fragment

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 Figure 2(a).
   c. Each row item should display the album image (cover_small), album title, artist name and album number of tracks (nb_tracks).
   d. For each album, the like or unlike indicator which is represented by un-filled heart icon and filled heart respectively. The user can like the album by clicking the un-filled heart icon, which should update the Firebase record to indicate that the user has liked this album, and the heart icon should be changed to filled to indicate that the user has liked the album. The list should be refreshed to reflect this update. (Hint, you can use the album id as the key for Firebase when managing the likes).
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 AlbumActivity to display the album details. Note that the intent starting the AlbumActivity should include the album information for the album that should be displayed.

Part 4: Album Fragment

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.
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 Horizontal RecyclerView as shown in Figure 2(b). Where each item should display the track title, track duration (MM:SS, minutes and then seconds) and the play button as shown in Figure 2(b).
5. Note that, each track item includes an mp3 url provided by the preview attribute in the returned JSON.
6. Clicking the “Play” button should play the mp3 url for the selected track using the mp3 URL returned in the result (preview). 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 Figure 2(c).
   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.
   d. If the user presses the “Play” button for another track while a track is playing, then the newly selected track should be played and the other track should be paused. The UI should be updated to reflect such change.
7. Note that, upon playing a track, the played track information should be stored on Firebase to record the history of track playback, which should be displayed in History Fragment.
8. Pressing the back button should show the Main Activity. Make sure to pause any playing track before finishing the activity.
9. Clicking the “Share” button beside the album cover image should replace the current fragment with the “Album Sharing” fragment, and put the current fragment on the back stack.

Part 5: Album Sharing Fragment

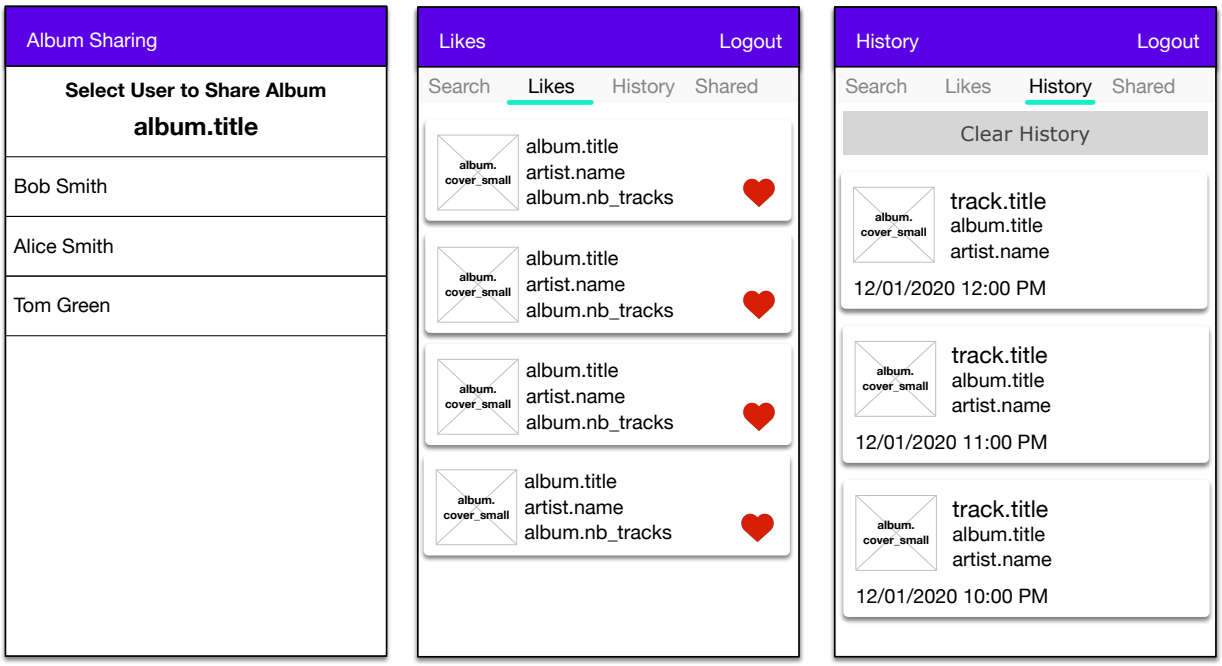
This fragment displays the list of users to enable the user to select the user to share the album with. The requirements are listed below:

1. The Album Sharing fragment is shown in Figure 3(a). The fragment shows the album title 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 album should be shared with the selected user through Firestore. Then the back stack should be popped which should show the Album Fragment.

Part 6: Likes Fragment

This fragment shows the list of liked albums for the currently logged in user. The requirements are listed below:

1. Figure 3(b) shows the Likes Fragment.
2. The list of liked albums for the currently logged-in user should be retrieved from Firestore and should be displayed as shown in Figure 3(b).
3. Each row item should display the album image, album title, artist name and album number of tracks.
4. The filled heart is displayed beside each album to represent that the user has liked this album. Upon pressing the filled heart, the selected album should be removed from the current user's likes, update Firebase and the list should be refreshed to show the updated user's likes.
5. Clicking an album row item should trigger the MainActivity to start the AlbumActivity to display the album details. Note that the intent starting the AlbumActivity should include the album information for the album that should be displayed.



(a) Album Sharing Fragment (b) Likes Fragment (c) History Fragment

Figure 3, Application Wireframe

Part 7: History Fragment

This fragment displays the track playback history history for the currently logged in user. The requirements are listed below:

1. Figure 3(c) shows the History Fragment.
2. The list of track playback history for the currently logged in user should be retrieved from Firestore and should be displayed as shown in Figure 3(c).
3. Each row item should display the album image, track title, album title, artist name and

date/time the track was played.

4. Clicking on the “Clear History” button should clear the user’s playback history and should refresh the list to show this update.

Part 8: Shared Fragment

This fragment displays the albums shared with the currently logged in user. The requirements are listed below:

1. The list of albums shared with the currently logged in user should be retrieved from Firestore and should be displayed as shown in Figure 4.
2. Each row item should display the album image, album title, artist name, album number of tracks, name of the user who shared the album, and the date/time the album was shared.
3. Clicking an album row item should trigger the MainActivity to start the AlbumActivity to display the album details. Note that the intent starting the AlbumActivity should include the album information for the album that should be displayed.

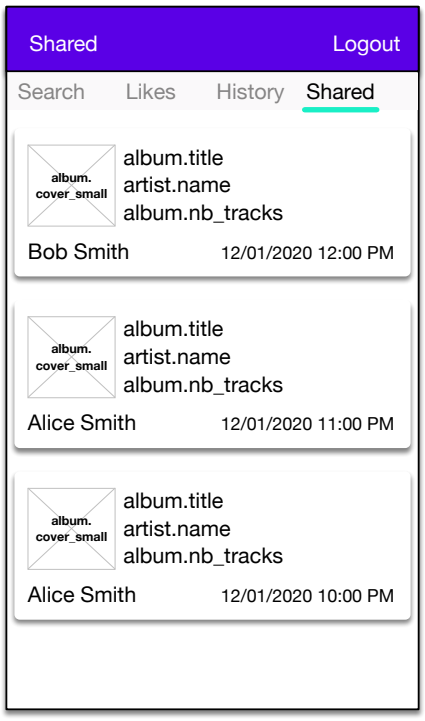


Figure 4, Shared Fragment

Features	Grade
Part 1: Login, Registration and the Activity transitions developed correctly.	5
Part 2: Main Activity - ViewPager, TabLayout, and Logout developed correctly	10
Part 3: Search Fragment - OKHttp and JSON parsing	10
Part 3: Search Fragment - Likes management	5
Part 3: Search Fragment - UI (RecyclerView and other UI)	10
Part 4: Album Fragment - OKHttp and JSON parsing	10
Part 4: Album Fragment - UI (RecyclerView and other UI)	10
Part 4: Album Fragment - Audio Playback and UI management	10
Part 5: Album Sharing Fragment - Firebase Access and UI	5
Part 6: Likes Fragment - Firebase Access and UI	10
Part 7: History Fragment - Firebase Access and UI	10
Part 8: Shared Fragment - Firebase Access and UI	5
Total	**100**