Capstone_Stage1

How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: “Capstone_Stage1”
3. Replace the text in green

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “Capstone Project”
3. Add this document to your repo. Make sure it’s named “Capstone_Stage1.pdf”

Table of Contents

GitHub Username: divyamary

Radio Tide

Description

Radio Tide lets you listen to a wide variety of internet radio stations from across the world for free. The app has a directory of AM/FM stations, Icecast stations and SHOUTcast stations from across the world. Listeners can search by song, genre, station name, geographic location and more. Talkshows and podcasts are also included.

Intended User

This app is for those who like music, for those who want to listen to a wide variety of radio stations from across the globe and for those who are curious to know the kind of music played in different countries.

Features

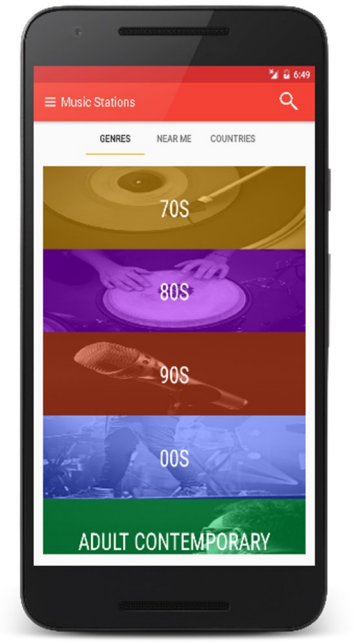
- Listen to online radio stations for free.
- Listen to talk shows/podcasts.
- Save the stations you like.
- Save the songs you like.
- Listen to the popular songs playing on the radio stations.
- Listen to local radio stations from your region.
- Search music stations by country/genre/song/artist and more.
- View the played history.

User Interface Mocks

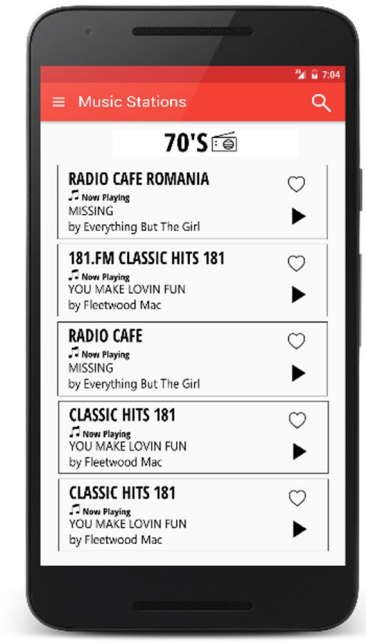
These mockups have been created in Inkscape. These mocks roughly describe the screens. The images/icons/colors/descriptions may vary in the final project.

Screen 1

The app opens with the main screen showing the list of music genres.

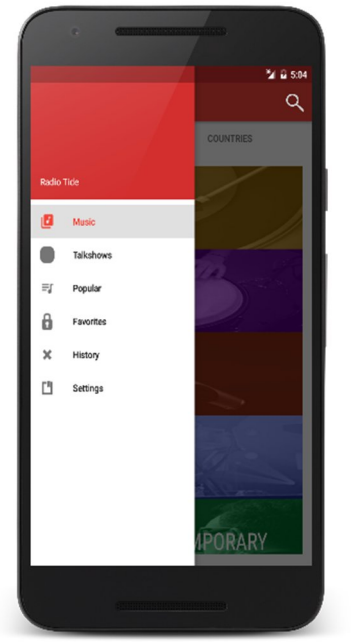


Screen 2



- Clicking on a genre from the previous screen shows you a list of stations matching that particular genre.
- Each list item has the callsign of the station and song currently playing.
- Clicking on the ‘play’ button will open a detail activity and start playing the station
- Clicking on the ‘favorite’ button will mark the radio station as a favorite and it will show up in the favorites screen.

Screen 3

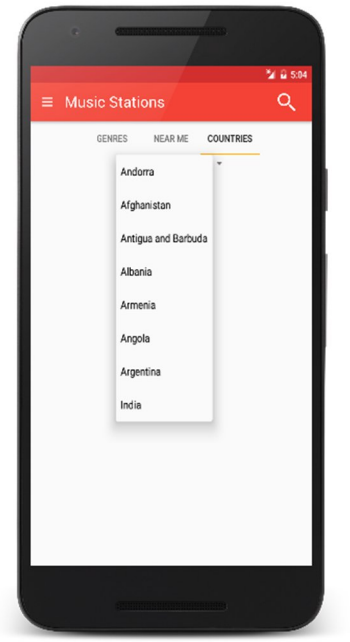


- This screen shows the expanded navigation drawer.
- It displays these items in the drawer:
  1. Music
  2. Talkshows
  3. Popular(Songs)
  4. Favorites
  5. History
  6. Settings

Screen 4

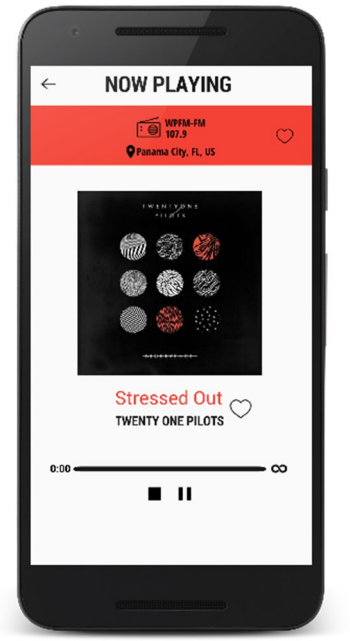
The ‘Near Me’ screen shows a list of radio stations around you.

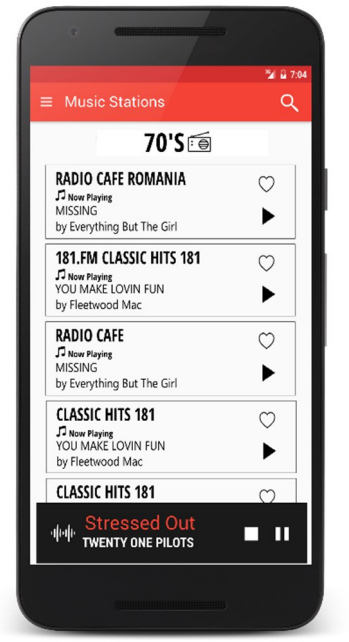
Screen 5



- The ‘Countries’ screen has a dropdown selector with a list of countries.
- Selecting a country will show radio stations that belong to the selected country.
- The populated view after selection is similar to Screen 2

Screen 6





A **B**

- Screen ‘**A**’ displays details of the song or show currently playing.

- A user will land on this screen from any screen which has a play button incorporated.

- The screen contains details of:

1. Station name/location
2. A button to favorite the station
3. Song/show title
4. Artist (if it's a song)
5. Album art(if it's a song)
6. Talkshow image(if it's a talkshow)
7. A button to favorite the song
8. Media controls

- Screen '**B**' shows the currently playing song along with media controls in a fragment that will be located at the bottom of the screen.

Screen 7

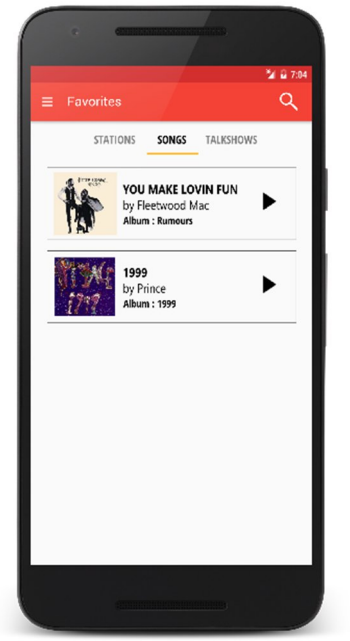
- This screen displays the most popular songs

- The list displays the most played songs over the last several days of internet radio, and that can be listened to right now.

Screen 8

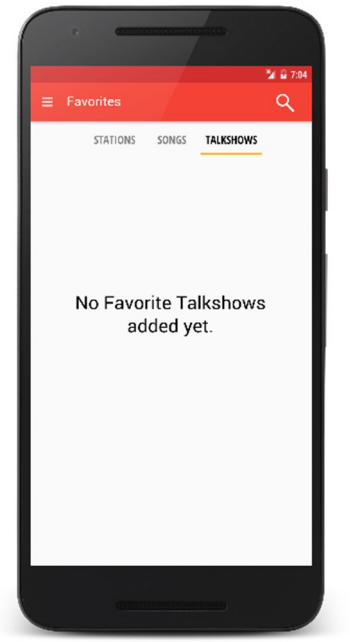
- This is the Favorite Stations screen.
- Any radio station marked as a favorite will show up here.
- A message will be displayed here in case no stations have been favorited.

Screen 9



- This is the Favorite Songs screen.
- Any song marked as a favorite will show up here.
- Pressing play will play the selected song only if it is playing on any of the radio stations.
- A message will be displayed here in case no songs have been favorited.

Screen 10

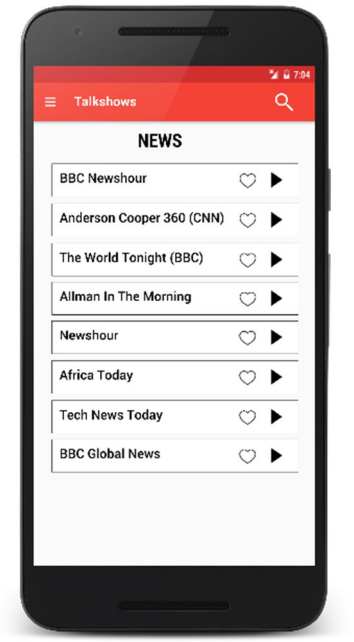


- This is the Favorite Talkshows screen.
- Any talkshow marked as a favorite will show up here.
- A message will be displayed here in case no talkshows have been favorited.

Screen 11

- This is the Main Talkshows screen.
- The screen shows the common categories of talkshows.

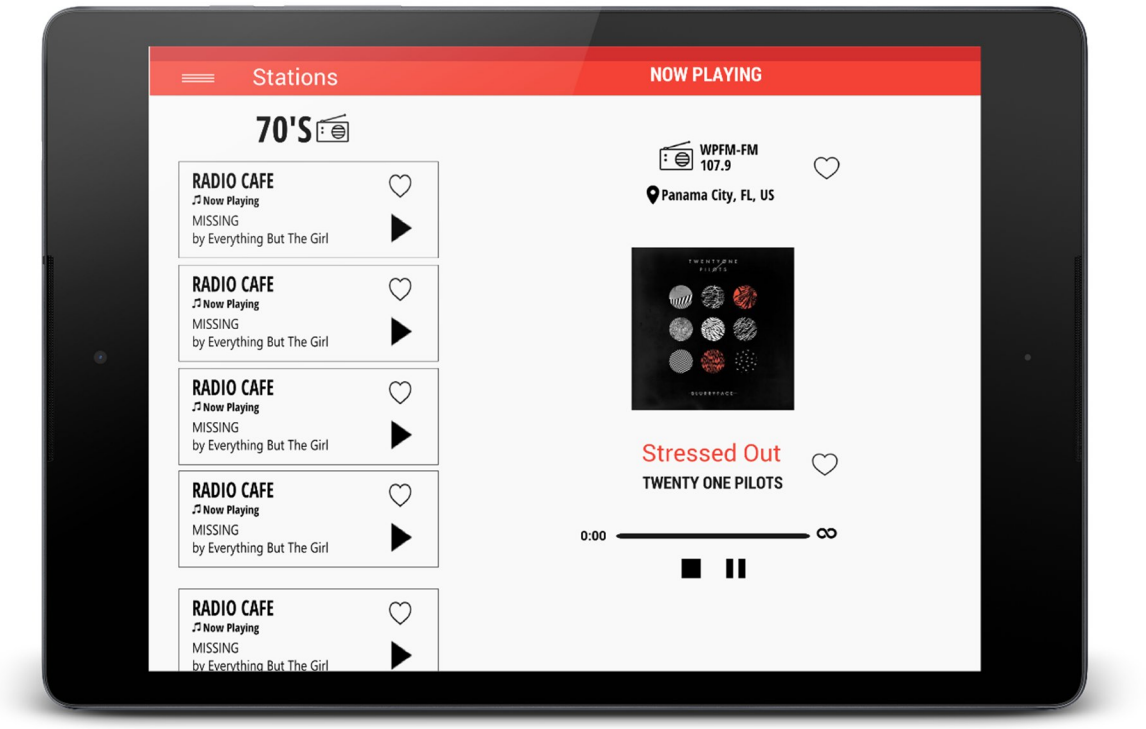
Screen 12



- This screen shows a list of shows from a selected talkshow category.
- On selection, it will show the ‘Now Playing’ screen.

Screen 13

- This is the ‘History’ screen. It displays the stations listened to in the last 7 days.



Tablet Mock which shows a Master-Detail layout.

Key Considerations

How will your app handle data persistence?

I will make my own Content Provider. When a user marks a station/show as a favorite, the details will be pushed into the SQLite database. Shared Preferences will be used to store data related to Settings.

Describe any corner cases in the UX.

- If a searched artist/song/genre/phrase is not found, a message is displayed to the user.
- If internet connectivity is lost, a message is displayed to the user.
- If a song has been favorited and a user tries to play it when none of the radio stations are playing it, user sees a message.
- If album art/station image does not exist, placeholder images will be shown.
- If headphones/earphones are unplugged, the music stops playing.
- If user hits the back button from the ‘Now Playing’ screen, the current screen will have a playback fragment displayed at the bottom. Clicking on that screen will take the user to the details screen.

- If a user exits the app, music will continue playing and it can be controlled from a notification shown in the notification bar.

Describe any libraries you'll be using and share your reasoning for including them.

- Picasso to handle the loading and caching of images.
- Retrofit to make network calls
- Butterknife to bind views
- Admob to incorporate ads and thereby monetize the app
- Google Analytics to monitor usage, track screen hits and understand the userbase.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

Creating the project

- Create a new project in Android Studio.
- Fill out the application name, domain and package name.
- Check the box for Phone and Tablet under form factors the app is going to run on.
- Select the minimum SDK as API 13(Honeycomb)
- Choose the Blank Activity template and click 'Next'
- Give a name for the activity, the layout and menu names will be populated accordingly.
- Click Finish and the project is setup.

Configure the build.gradle file

- Add the v7 appcompat, cardview, palette, recyclerview libraries
- Add the design support library
- Add the retrofit libraries for making network calls
- Add the Picasso library for downloading images
- Add the event bus library
- Add Play Services admob library
- Add Play Services analytics library

Get the DAR FM Partner Token

- Go to http://dar.fm/partners.php
- Sign up for a partner token.

Setup Admob

Follow this guide: https://firebase.google.com/docs/admob/android/quick-start

Setup Analytics

Follow this guide: https://developers.google.com/analytics/devguides/collection/android/v4/

Task 2: Implement UI for Each Activity and Fragment

1. Build UI for Main Activity which contains
   - A Toolbar
   - Navigation drawer
   - A view pager which holds fragments.

2. For the first drawer item: ‘Music’, there would be three fragments contained in a View Pager
   - A Fragment that displays all the genres. Genres are displayed in a Recycler View. Fragment that displays all the nearby radio stations. Radio Stations are displayed in a Recycler View.
   - A Fragment that displays all radio stations from around the user’s current location. Radio Stations are displayed in a Recycler View.
   - A Fragment that displays a dropdown selector containing countries. Selecting a country will show radio stations from that country. Radio Stations are displayed in a Recycler View.

   For the above fragments that make use of a recycler view:
   - Create a model class that would act as the data source.
   - Create a layout xml and add the v7-RecyclerView.
   - Create a custom row layout that would be used for each row within the list.
   - Create a RecyclerView.Adapter which would populate the data into the Reycler View.
   - Bind the adapter to the Recycler View.

3. For the second drawer item: ‘Talkshows’ there would be a fragment showing talkshows sorted by categories such as ‘Politics’, ‘News’, ‘Sports’ etc.
   - Clicking on one of these categories would show the talkshows of that category in a Recycler View.
   - Clicking an item will show the ‘Now Playing Screen’.

4. For the third drawer item: ‘Popular Songs, there would be a fragment which would show all the popular songs currently playing. These items are displayed in a Recycler View Clicking an item will show the ‘Now Playing Screen’.

5. For the fourth drawer item: ‘Favorites, there would be three fragments contained in a viewpager.
   - A Fragment that displays all the favorited stations. These are displayed in a Recycler View and are fetched from the database via a content provider.
   - A Fragment that displays all the favorited songs. These are displayed in a Recycler View and are fetched from the database via a content provider.
   - A Fragment that displays all the favorited talkshows. These are displayed in a Recycler View and are fetched from the database via a content provider.

   - Clicking on any of these items will show the ‘Now Playing Screen’.

6. For the fifth drawer item ‘History’, there would be a fragment that would display the radio stations listened to in the past 7 days. Whenever a radio station is played, its details are logged and pushed to the database. These details are displayed in this fragment. A content provider would be used to retrieve the items. The items are displayed in a Recycler View

7. For the ‘Now Playing Activity’, create a layout with having TextViews which will show the radio station name/location, song artist/title, an ImageView which will populate the album art, Buttons to favorite the station/song. The layout will include a fragment that will have the playback controls. This fragment is bound to the Music Service.

Task 3: Create the Music Service

1. Create a class which extends ‘android.app.Service’
2. Add this service in the Manifest
3. The Service class should implement MediaPlayer.OnCompletionListener, MediaPlayer.OnPreparedListener and MediaPlayer.OnErrorListener
4. The Music Service will initialize the MediaPlayer.
    - It uses a wake lock to prevent the music from stopping when the service is running.
    - It also makes use of a WifiLock in case the user is listening to music over Wifi.
    - The Audio Stream Type is set to AudioManager.STREAM_MUSIC.
    - The data source is set to the streaming url which would be passed via the intent.
5. This Service will be called from the ‘Now Playing Screen’ via an intent.
6. Depending on the intent, we either process the play or pause requests.
7. The Music Service would manage state.
8. The Music Service would create a notification and call startForeground in order to turn it into a ‘Foreground Service’.

Task 4: Implement the Content Provider

1. Create a class that extends ContentProvider. Implement the methods and Initialize the URI Matcher.
2. Create the Contract class. Provide definitions for the content authority, URIs, table names and column names.
3. The tables that would be created be Stations, Songs, Talkshows and StationHistory.

Task 5: Add Search Functionality

1. Add the SearchView widget to the main menu xml.
2. Inflate this xml in the onCreateOptionsMenu of the activity
3. Create a searchable.xml file in the res/xml folder. It must contain the android:label and androint:hint attributes.
4. Add the following things to the Manifest:
    - A<meta-data> element that points to the xml file create above. Declare this in all activities where we want search to appear

- Add the ACTION_SEARCH intent

5. In the searchable activity, check for the intent in the onCreate method and use the string result to form a network call that will search for the user entered phrase.

Task 6: Consume APIs

1. Capture the JSON outputs and generate the POJOs.
2. Create the Retrofit instance and specify the base URL(http://api.dar.fm). Specify a Gson converter to be used with it.
3. Create an interface and define the endpoints within it.
4. Perform the network call by instantiating a Call object and specify the HTTP Query to be executed.

Submission Instructions

1. **After you've completed all the sections, download this document as a PDF [File → Download as PDF]**
2. **Create a new GitHub repo for the capstone. Name it "Capstone Project"**
3. **Add this document to your repo. Make sure it's named "Capstone_Stage1.pdf"**