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

GitHub Username: https://github.com/Charmmonkey

Echo

Description

Listen to music, chat, and create the perfect playlist with your friends! Echo opens up the traditional way of listening to music into a communal activity. Through Spotify, you can listen millions of soundtracks and explore millions more with the help of friends.

Intended User

Anyone who likes listening to music with friends

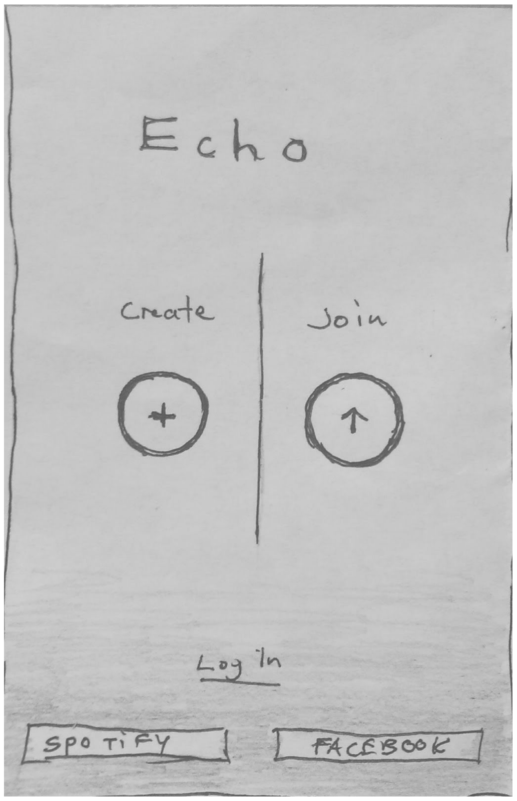
Features

List the main features of your app. For example:

- Craft the perfect music playlist together
- Play Spotify music in the foreground or background
- Message/chat with friends at the same time

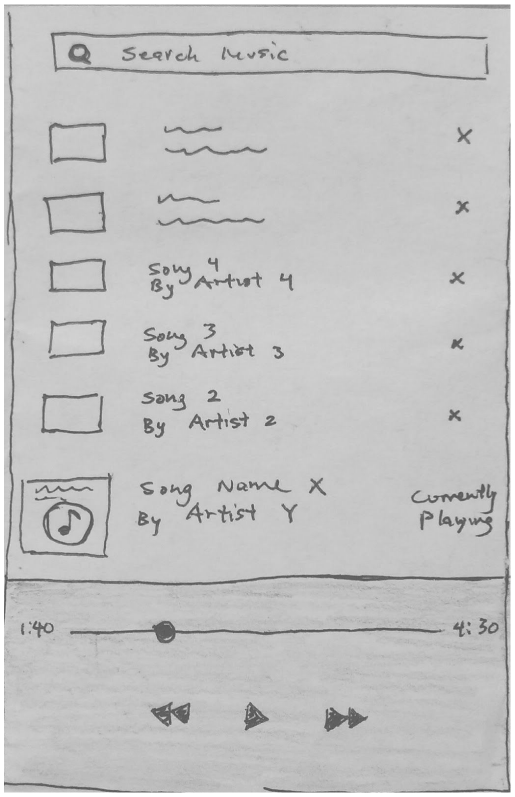
User Interface Mocks

Screen 1



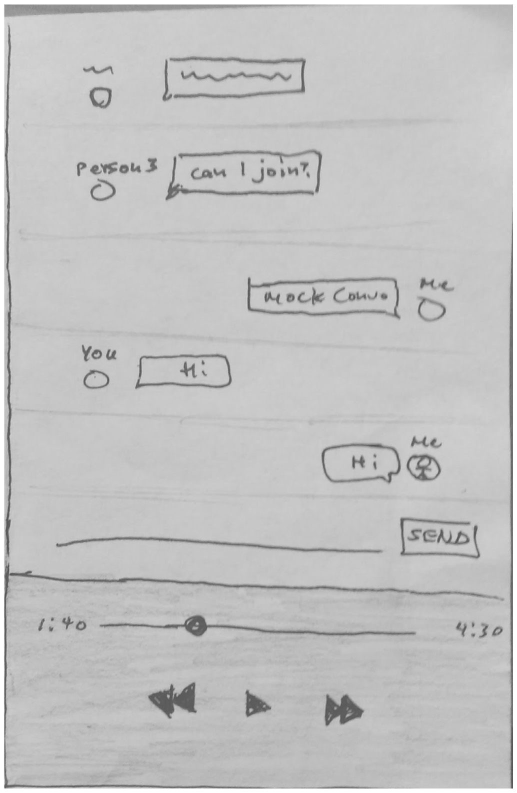
Login page with options to Create or Join a room. These will be initially grayed out and unclickable until user logs in to their Spotify account through either Spotify or Facebook.

Screen 2a



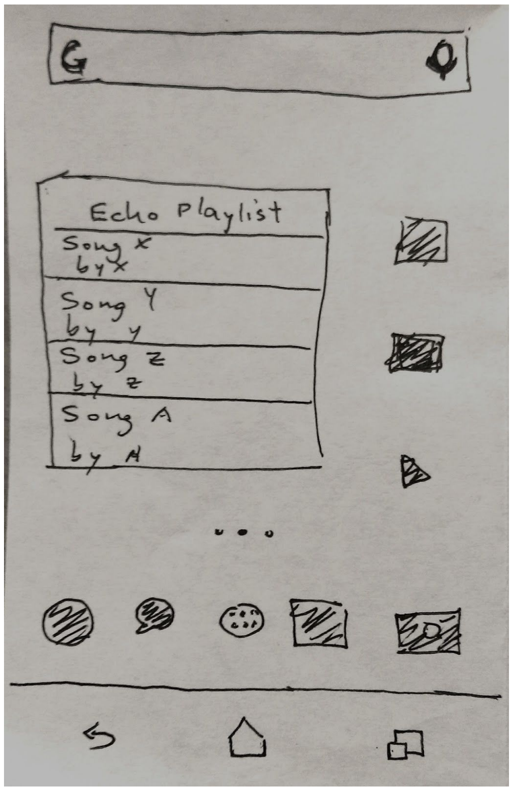
Inside the room, an Activity will contain a music player and a ViewPager. ViewPager Fragment 1 is where users can search and add music to the playlist. Music player will be hosted in a Service instead of inside the activity, but the controls will be in the Activity

Screen 2b



ViewPager Fragment 2 is where users can chat as well as control his or her own music sourced from the playlist

Homescreen (Widget)



Widget that displays the current playlist as a list.

Key Considerations

How will your app handle data persistence?

- Firebase Realtime Database for messages and music (url, name, artist names) as music playlist will be updated in real time as people add/delete
- SharedPreferences for account information such as spotify token.
- AsyncTask to fetch user's Spotify account information when entering a room.

Describe any edge or corner cases in the UX.

- Hitting back button twice will exit and destroy the music room (with warning) and return the screen back to the room creation activity.
- If user quit the app will in a room

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

- Picasso for image processing

- Butterknife for view binding
- Kaaes' spotify web api wrapper

Describe how you will implement Google Play Services or other external services.

- Spotify authentication to login to Spotify account and get authentication token
- Spotify player to play Spotify premium music
- Firebase Real Database to store information of each room instance (chat messages, music, users)
- Firebase Cloud Messaging with Cloud Functions for sending notification on FRD event triggers

Next Steps: Required Tasks

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

Task 1: Project Setup

Configure and add project dependencies
- Kaaes' Spotify web wrapper
- Spotify .aar files
- Firebase real time database
- Firebase Cloud Messaging with Cloud Functions
- Recyclerview and other support library as necessary
- Butterknife by Jake wharton

Set up Spotify premium account and firebase console account.
- Generate client id, keys as necessary

Task 2: Implement UI for Each Activity and Fragment

Two activities
1) Spotify log in page that leads to a lobby or room creation/joining page
2) Inside the room
    - Activity consisting of music player and ViewPager with 2 pages (1 for music selection and 1 for chat)

Task 3: Implement Firebase Realtime Database interface

Create a FRD event bus class with different interfaces for callbacks and access methods.

- Intended for use by all three following activity/fragments

Task 4: Implement music search and playlist fragment

Music Fragment: Add search bar for Spotify music

- Using Kaess' implementation
- Selected music will be added to FRD and displayed in room's playlist (recyclerview)

Task 5: Implement chat capability

Chat Fragment

- Edit text box with send button that sends message to FRD
- RecyclerView populated by messages fetched from FRD
- Callback that adds new messages to chat view every time someone presses send (ChildEventListener)

Task 6: Implement music player Activity

Contains Spotify music player that fetches from FRD the playlist

- Create Service that houses the Spotify music player
- Create interfaces to interact between Activity play/pause/next button to music player control
- Create layout and connection to FRD

Music player communicates with a SeekBar and periodically update according to music playback

- Create runnable and handler to update on main UI

Task 7: Add room creation activity

Login screen with room creation/joining option initially unclickable until login

- Each room created will have a password and unique room ID

Home page after login that allows two actions

- Create a room where other people can join

- Join a room by entering the correct room # and password

Task 8: Implement Firebase Cloud Messaging and Cloud Functions

FirebaseInstanceIdService

- Acquire this unique ID for each device to send notification to

FirebaseMessagingService

- Write some JavaScript code to send notification downstream to relevant devices when new Users join the room.
- Handles messages while app is in foreground. When in background, default notification is created.

Task 9: Widget

Create widget with collections of songs from playlist. Updates every time the playlist updates.

Task 10: Polish UI

Add vector animation, transitions, make sure everything look good and material.

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
  - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"