Architectural Design Document

MusicTaste

1 Introduction

This document presents the architecture for the software for the MusicTaste project. This project performs functions such as allowing the user to post one song per day, and the song that they post will be visible to their friends and other users of the app. People who posted the same song will be able to chat about the song that they posted that day. The app will also give users recommendations for new songs based on what they've been posting and listening to.

1.1 System Objectives

The goals of this project are to provide an easy and seamless way for music lovers to share their current obsessions, see what their friends are listening to, and find people who have similar music tastes as them. These objectives will require a main page that has a post feature where users can share one song each day. They will also require the use of the Spotify API that can gather song recommendations for users, allow users to login through Spotify, and access the Spotify library.

1.2 Hardware, Software, and Human Interfaces

The hardware interfaces consist of user devices such as smartphones (iOS) where users will be able to interact with our app. For networking, the app will mostly use wireless networking (Wi-Fi) for interactions, and the app will connect to Firebase and the Spotify API for data storage and retrieval. For software, we will use Firebase to store the user's friends, previously posted songs, and chat history. The Spotify API will be used for user login, song recommendations, and music library access. For the Backend, we will utilize a threaded server to handle multiple user connections and interactions. For Human Interfaces, the Music Taste UI will feature an interface for posting songs, viewing your added friends' daily songs, and chatting with users who posted the same song as you. The phone touchscreen and keyboard are also human interfaces as they are required to interact with and use the app.

2 CSCI Descriptions

This application will be made in three components: the **client side CSC**, the **server side CSC**, and **database CSC**.

- **Client CSC**

  The Client will be made by different pages called with functions, with a tab menu bar connecting them. The pages will all be on the same level, with specific features of the page being component functions located inside the page. Data in the pages will be shared through server and database, thus each page could be seen as independent CSU.

  The Client CSU will contain:
  - Tab Menu Bar CSU: this bar allows the user to navigate through different pages.
  - Home Page CSU, the start up page containing the following:
    - Friend Post CSU: used to show the posts made by friends. Allows the user to scroll left and right to view more.
    - Recommandation CSU: used to show the recommended songs, provided by music APIs such as Spotify.
  - Post Song Page CSU: allow users to post their favored song through selecting songs from the music library provided by APIs, along with texts and pictures.
  - Chat Page CSU: the page containing all the chats the users have with other users and groups of users.
  - Profile Page CSU: the page presenting profile information and settings with the following components:
    - Profile Information CSU: used to present all the profile information of the selected user, including username, favorite artist, song, and genre.
    - Post Calendar CSU: used to present a calendar of the given user, with the posts they made each day, visible to selected groups, among self / friends only / all.
    - Log Out CSU: allow users to log out of their account.
  - Login Page CSU: used for the users to log into their account.
  - Sign Up Page CSU: used for the users to create an account.

- **Server CSC**

The server will be run on Expo, an application that helps running cross-platform applications. There will not be any GUI on the server side, the only components for the server will be function calls fetching data from the database.

The Server CSC will contain the following:

- Fetch CSU: providing passcode and connection to Firebase, to allow client side to fetch data in the database.

- **Database CSC**

There will be no class in the database, instead, there will be SQL files storing text information, and images stored within it. The connect and fetching process will be provided by the server, and the database itself will provide API for different data.

The Database CSC will contain the following:

- Database API CSU: used for choosing the desired database the specific data wanted inside the database.

2.1 Concept of Execution

To run MusicTaste, its server must be started first. With the start of the server, execution will be run and let the application connect to Firebase. After the server starts, the clients could run the application and connect to the server to gain all the information.

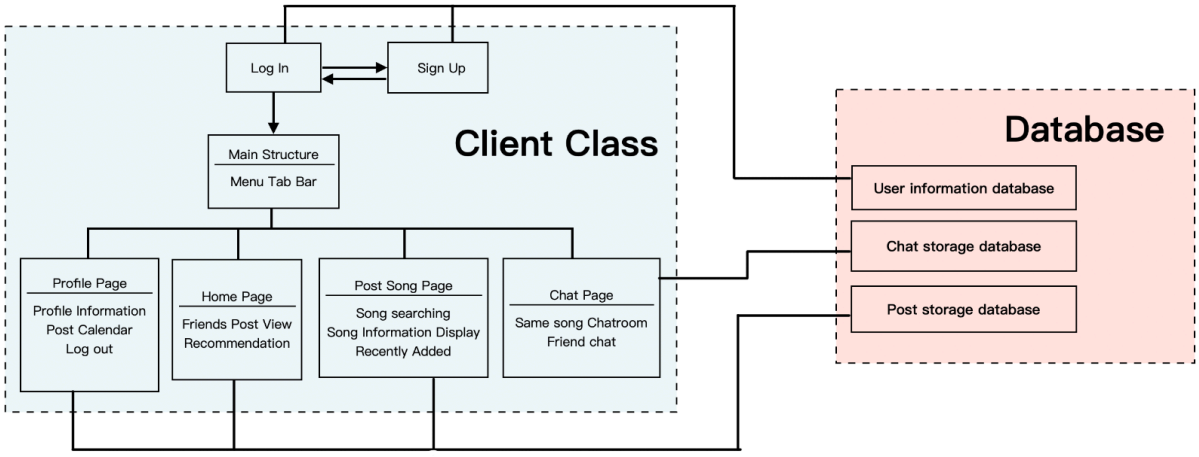
After opening the application, the user will be asked to log into their account. With verification from the database, the user could view pages and information visible to them. They would start from the Home page, and with the menu bar they could jump to other pages. With the action they took on adding / deleting friends, or adding new posts, changes will be made to the database. The database will not store the content of chats unless the users ask to.

2.2 Interface Design

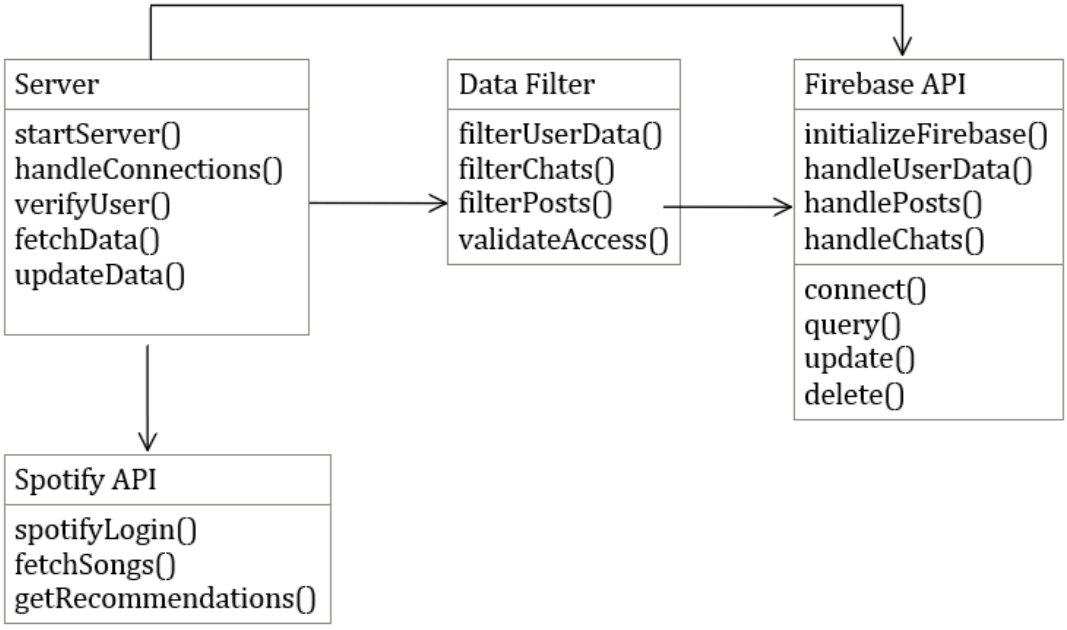
The following provides diagrams representing the interface designs for this project.

2.2.1 Interface Identification and Diagrams

2.2.1.1 Client Class and Function Modules



2.2.1.2 Server and Database API Modules



2.2.2 Project Interactions

The interaction between the CSU happens when either client side or database changes. A package will be sent with the changes for the other to update its data, either to change the database or the client's local data.

The application will update its renderer every few ms, and pass multiple tests during each refresh. Page jumping follows this process, including the use of tab menu bar, and the switch between sign up, log in and log out. Same with the rendering of the

items on the pages, that since real-time chatting and posting functions is implemented inside the application, with the detection of the change in data, if the user is on the chat page, the page will refresh with the new chat showing; if the user is not on the page, a notification as a red dot will appear on the chat icon in the tab bar. Overall, the command of changing is not made by the server, but rather by the detection of local data made by the client side.

The data transferred between the client side and the database are limited with user's accessibility for both security and efficiency purposes. A filter will be made with the account information provided by the client side, and all the requests it passes to the server will have the filter information with it. With the filter, the server could grab only the relative chat, friend posts to pass back. For the same security issue, the client side also has a lock on the data, that allows the user to see only the information they have access to, so that when account changes on the client side, the old chat data stored locally would not be accessed. The same check will happen twice, but it is a decision of security over speed.

3 Preliminary User Manual

To get started, download the musicTaste app on your mobile device. Securely log in to the app with a valid Spotify or Google account, or register your email to get started.

Once logged in, you'll see the home page which consists of your friends' daily tracks, song recommendations, and a ratings system that you can use to give your favorite songs a rating. The menu bar at the bottom of the page includes tabs for **profile**, **home**, **add song**, and **chat**.

Tapping the **profile** icon takes you to your profile page, where your username will be displayed along with a profile picture that you can add from your camera roll. There is also a place for you to add your favorite artist, album, and song of all time. Finally, at the bottom of the profile page, you can click "see previously posted songs" and a calendar will popup displaying all of the songs that you posted previously.

If you click the **add song** icon, a popup will appear that will allow you to choose a song from Spotify's music library to post for the day. You can add an optional picture and caption to accompany your post. Click "post", and now your post will show up on your friends' feeds.

By selecting the **chat** tab, you will see a list of users who have posted the same song as you. You can easily add them as friends or start a chat with a specific user to discuss the song.

To get back to the main page, tap the **home** icon.

On the technical side, the app’s server is threaded to handle multiple connections, and uses a system of "callbacks" to handle functions for all buttons and indicators.