Software Design Detailed Document

Nonstop

1 Introduction

This document presents the detailed design 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.

1.0 Detailed Design Descriptions

The following sections contain the descriptions of the details of the design of MusicTaste CSCI. The design is systematically described in terms of the CSCs for the project, and the CSUs for each CSC.

2 Detailed Design Descriptions

2.1 CSC - Tab Menu Bar

This bar allows the user to navigate through different pages.

2.2 CSC - Home Page

The start up page containing the following:

2.2.1 Friend Post CSU

Used to show the posts made by friends. Allows the user to scroll left and right to view more.

2.2.2 Recommandation CSU

Used to show the recommended songs, provided by music APIs such as Spotify.

2.2.3 Song Posting CSU

Allow users to post their favored song through selecting songs from the music library provided by APIs, along with texts and pictures.

2.3 CSC - Chat Page

The Chat Page contains the following:

2.3.1 Navigate Bar CSU

A scroll bar that shows all the chat rooms the user's in, with buttons leading the user to the chatrooms.

2.3.2 Live Chat CSU

A scrollable window showing all the history chat that's updated lively.

2.3.3 Input CSU

Allows the user to send chat and save it to the database.

2.3.4 Rank CSU

A rank system showing the rank for the history chat based on highest vote / most replies.

2.4 CSC - Profile Page

The page presenting profile information and settings with the following components:

2.4.1 Profile Information CSU

Used to present all the profile information of the selected user, including username, favorite artist, song, and genre.

2.4.2 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.

2.4.3 Logout CSU

Allow users to log out of their account.

2.4.4 Taste and Favor CSU

Allow the users to add and show their favorite albums and songs on their profile page.

2.5 CSC - Login Page

The login page will have the following components:

2.5.1 Email / password login CSU

Provides the function to login with email & password through firebase authentication.

2.5.2 Sign Up CSU

Function to register an account on firebase authentication to log in with account and password.

2.5.3 Spotify login CSU

Provides the function to login with a spotify account.

2.5.4 Database Creation CSU

Databases will be created when a user logs into the application for the first time. Information will be fetched and stored.

3 Database Design Descriptions

3.1 User Database CSC

While fetching documents from firestore, the whole file would be fetched, thus user data are stored in separated files.

3.1.1 User General Data CSU

User's general data will be located in root/profiles/<userID>, content will include: first name, last name, register date, username, birthday, following list, follower list, chat room ID list.

3.1.2 User Post Data CSU

User's posts will be stored in root/profiles/<userID>/posts, and the document will have fields: map[], where each map will contain title, content, imageURI, songRef, likes and replies.

3.1.3 User Favorite Album Data CSU

User's favorite albums will be stored in root/profiles/<userID>/albums/<albumID>, with the information artists, id, imgURL, and album's name.

3.1.4 User Favorite Song Data CSU

User's favorite songs will be stored in root/profiles/<userID>/songs/<songID>, and include fields artist, id, imgURL and song's name.

3.2 Chat Room Database CSC

Each chatroom will be stored in the format root/chats/<chatroomID>, where if it is not a group chat, the chat room ID would be the two user's ID concat in alphabetical order. The database document will contain the following fields: profile Image URL, name, members with UID as key and nickname as value, and an array of chat contents in map. Each map will have the following fields: dialog, user, timestamp, reply (in array of string), upvotes and downvotes.

4 Detailed Interface Designs

Overall, the interactions inside client side are done by react's router function to switch from one page to the next. Going deeper, all the exchange in data is done with firebase. The application will connect to firebase when it starts up with a file that contains provided keys to access the right database, then with firebase's authentication api an unique ID will be given to the user. Different pages in the application access different parts of the database with the ID as key, and the data will be sent back with protocols.

More specifically, the login page allows users to login through email or Spotify. It uses expo-secure-store to store Spotify tokens and user information. The user is then directed to the main page. The main page fetches the Spotify user profile dynamically. The SpotifyAuth component passes the token from the login page via the onTokenFetched callback. The tokens are stored securely and used across components. The main page then calls the firestore database to retrieve information about the users' friends' posts. The profile and chat pages also use the access token and fetch user data from Firebase.