Cloud Music Player

Team 30

Guojie Wen
Dai Zhong
Pengyu Chen
Zhe Chen

09/15/2017

Purpose

Google Drive is a popular application to store files, and people can collect their favorite music in Google Drive. However, the built-in music player does not provide function that the users expect. The built-in music player only allows users to play one song every time. The poor function of the built-in music player will harm users' experiences in different way. Google Drive users may be tired of clicking the audio files to play. Also, they may be tired of downloading the music back to the computer and using another music player. To improve the experience of playing music, we want to create a more functional music player with a straightforward user interface which will be easier for users to view music files and create playlists of music in Google Drive.

Music should be appreciated without feeling any uncomfortable. We believe our cloud music player can provide Google Drive users a more convenient way to enjoy music.

Functional Requirements

1.Users can login with Google Drive account and manage audio files in Google Drive

As a user,

A. I would like to log in with my Google account
B. I would like to be able to view the music interface after login

C. I would like to be able to view my Google Drive directory after login
D. I would like to be able to navigate and browse all my Google Drive file directories
E. I would like to be able to only select mp3 files
F. I would like to be able to view all of the mp3 files that I selected while I am browsing for music
G. I would like to be able to one click selecting all the mp3 file I have in my Google Drive while I am browsing for music
H. I would like to logout my Google Account when I want to do it.

2 . Users can manage and create the music lists on the user interface

As a user,

A. I would like to be able to go to the music interface after I finish selecting music files from Google Drive
B. I would like to be able to view all the playlist names I created on the left side of the screen
C. I would like to be able to click the playlist name to expand and view the entire playlist
D. I would like to be able to delete songs from playlist.
E. I would like to be able to have a button in the music panel to add the currently playing song to a different playlist
F. I would like to be able to add the selected song to a different playlist on the right click panel

G. I would like to be able to select multiple songs in a playlist to add the songs into a different playlist (if time allows)

3. Users can play music in their player lists and change the progress of music

As a user,

A. I would like to be able to left click on a song, play the song, and highlight the currently playing song in the playlist.
B. I would like to pause and resume a song as long as the application is not closed.
C. I would like to be able to switch songs in the playlist by clicking “left arrow” and “right arrow” buttons beside the “play” button.
D. I would like to be able to switch listening mode of the playlist like random, single, or in order.
E. I would like to be able to change the progress of a song by dragging the progress bar.

4. Users can adjust the volume of music, see the information of the current playing music, and share the music

As a user,

A. I would like to adjust the volume of the playing song.
B. I would like to have a button to mute the sound on click.
C. As a user, I would like to be able to see the song’s name, singer name, and album name in the player interface.
D. As a user, I would like to be able to see the song image in the player interface.

E. I would like to share my music and playlist to social media like facebook, twitter, etc.

5. Users can find more options and have a better view in the music player.

As a user,

A. I would like to hide the playlists on the left when the screen size of the browser gets small.

B. I would like to be able to click a button on the top of the screen to expand and show the playlists from the left.

C. I would like to be able to right click on a song in a playlist to show more options

Non-functional Requirements

1. Client Requirements

A. Users must be able to use this app on most of the mainstream browsers, and the browsers we need to guarantee its performance and usability are Chrome, Firefox, Microsoft Edge, and IE 11

2. Server Requirements

A. The frontend will be hosted in a Heroku, and a separate developing server will also be hosted in Heroku to make sure the application works in Heroku environment

3. Appearance Requirements

A. The interface should be easy and straightforward so that no specific knowledge is required for users

4. Performance Requirements
    A. Must be able to respond immediately except for extreme situations such as too many files in Google Drive or network connection is too weak
5. Security Requirements
    A. Must only allow one user to access his or her own music files
    B. Must not upload any user token to server for privacy purposes, only allow the application to store user token to localstorage
    C. Must not allow to modify contents of Google Drive in the application. Files should not be deleted from the cloud drive when they are deleted from the playlist.

Design Outline

Our application is a web app that allows users to play their music in Google Drive with a standalone music player. We will implement the Client-Server model. The web client will access data through Google Drive and Firebase, and it will use libraries to decode mp3 files and play music.

React.js Client:

- Frontend JavaScript framework
- Handle frontend content and functions

Google Sign In:

- Google user log in API
- User log in
- Retrieve user accessToken

Google Drive API:

- Google Drive API
- Import user music and store file IDs in Firebase

Firebase:

- Backend server and database service
- Save user playlists and settings

- Provide data to web client

Howler:

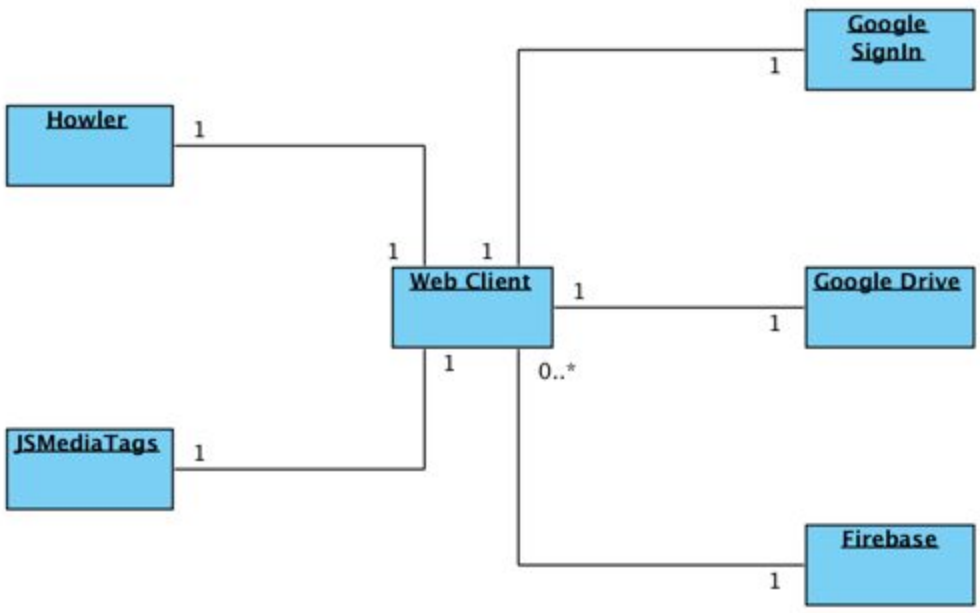
- JavaScript library
- Music control, such as playing, pausing, change playing progress, etc.

JSMediaTags:

- JavaScript library
- Retrieve music file meta tags, such as music title, author, album, and album art

System High Level Interaction

The Web Client uses Google SignIn to perform user sign in, and one client can only log in with one user at a time. Once user signs in, the client has access to user's Google Drive account, therefore it forms a one-to-one relationship. Web Client also fetch data from Firebase once user logs in, but Firebase can provide data to many clients. When user plays a song in Web Client, Howler library will take the music file and provide audio output. While user plays the song, JSMediaTags library will also take the music file and fetch its detail information, and Web Client will display those information to user.



Design Issues

During our process of designing this project, multiple issues came up immediately. We will bring some problems we encountered in our design and provide some feasible solutions to solve them.

Functional Issues

1. Is it necessary to display "mute" button on the interface?
    a. Option 1: Design the "mute" button on the right corner of interface
    b. Option 2: No display, use the keyboard instead
    c. Option 3: Clicking the "mute" button and using the keyboard are both acceptable.

    Decision: We chose to use the "mute" button and the keyboard at the same time to take care of the both conditions in which someone accidently use either one of the options. By keeping two attributes in the application, the users will have a chance to decide the player whether being muted for a while or not.

2. What kind of audio files should we provide to be able to select in the play list?
    a. Option 1: Only the filename with suffix of ".mp3" can be chosen
    b. Option 2: All the files will be able to choose from

    Decision: In case of reading the files of inconsistent format, we only allow one format of the files to be read, which are the ".mp3" files. By providing only one

kind of files to be played, we hope that the unrelated file will be ignored and only the audio list with the right format will be pop up and played.

3. When should our Cloud Music Player appear?
    a. Option 1: As soon as the user login in their Google Drive account
    b. Option 2: There will be a window pop up that will have several options for user to choose whether to go to the Google Drive directory directly or our Cloud Music Player

    Decision: We planed to design a window pop up after users login in with their Google account, to decide whether to go to Cloud Music Player or Google Driver by default. For example, Some users would like to access their video or word document instead of going to the music player directly. So it will be determined by the users for their preferences to play music immediately after logging in or boost our player with the button later.

4. If the current playing song no longer exists in the Google Drive, what should we do?
    a. Option 1: Prompt the users that the song does not exist
    b. Option 2: Remove the song automatically from the playlist and go to the next song

    Decision: We choose to prompt the users that the song does not exist. Even though the option 2 allows users to continue listening to music without any interruption, we still want to inform users that the song does not exist. Then the

users can check if they accidentally delete that song. We think this can help users have better management of the audio files in Google Drive and playlist in the cloud music player.

Nonfunctional Issues

1. What front end framework should we use?
    a. Option1: React.js
    b. Option2: Angular 2
    c. Option3: PHP

    Decision: We choose the React.js as our front end Framework. Compared to other two options, PHP has more complicated syntax and logic. React.js and Angular 2 have advantages in helping us to organize and reuse code. However, since three of the teammates have no experience in developing web, and one teammate has experience in using React.js, we decide to use React.js.

2. What database software should we use ?
    a. Option1: Firebase
    b. Option2: MySQL

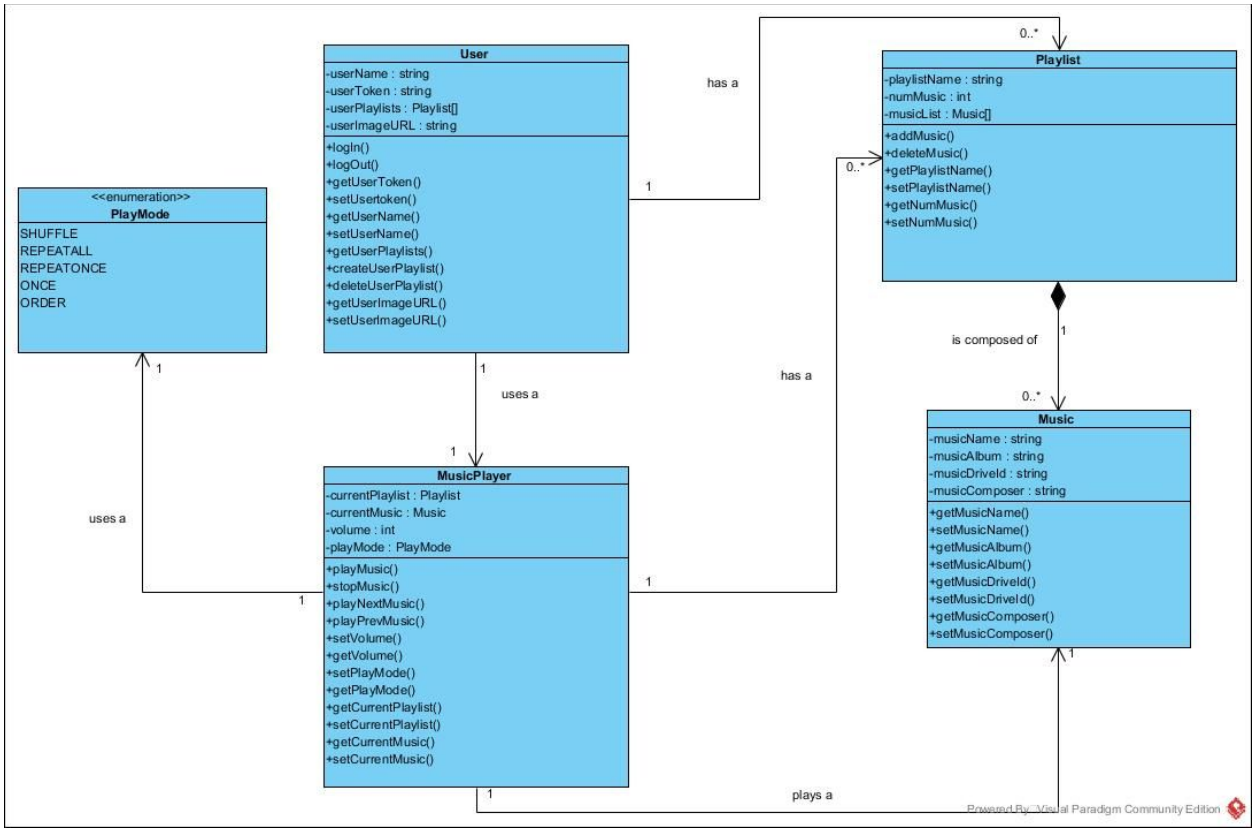
    Decision: We choose the Firebase as our database, because Firebase not only provide database storage, but also a library for us to use it like a server. Firebase is more popularly used for cloud-hosted real time communication. Thus, it will be more advantageous to use Firebase.

3. What type of architecture should we use?
    a. Option1: Client-Server Architecture
    b. Option2: Model-View-Controller

Decision: We will choose the Client-Server Architecture.For our music player, we don’t need to have an extra controller to handle user’s interaction with the view and model. We basically just read the audio files from the Google Drive and retrieve playlists from Firebase, and we handle all the logical data processing in the client side.

Design Details

Class Diagram



Description of Classes and their interactions

Above is an abstract model of our data class design. These classes is based on the understanding of objects and their roles relatively in our project. Information stored in our database will be organized according to the class diagram above. Things might change afterwards but we believe this design is able to satisfy current functional requirements.

User:

- All users of our application are represented as instances of this class, including new users and returning users
- Contains username, image or avatar URL, an array of Playlists instances, and user tokens.
- A user is able to login and logout. Playlists instances array is able to be edited by a user. The rest are getters and setters for access by other classes.

Playlist:

- Playlist is the abstraction for playlists of every user. It is composed of Music instances abstractly. A music must be added in a playlist to be played by user.
- Contains information about every playlist like name of the playlist, number of music of the playlist, and an array of Music instances.
- For each playlist, functions of editing music instances array are contained. The rest are setters and getters for private attributes.

Music:

- Music is the abstraction for music of every playlist. Music files in Google Drive are obtained by drive ID contained in music class. Music must be in a Playlist instance to be played by a User who owns that Playlist
- Contains information about a song such as the name, the album, the composer, and the drive ID. Fields not specified in the original file will be marked as unknown.

- All functions are setters and getters for private attributes.

MusicPlayer:

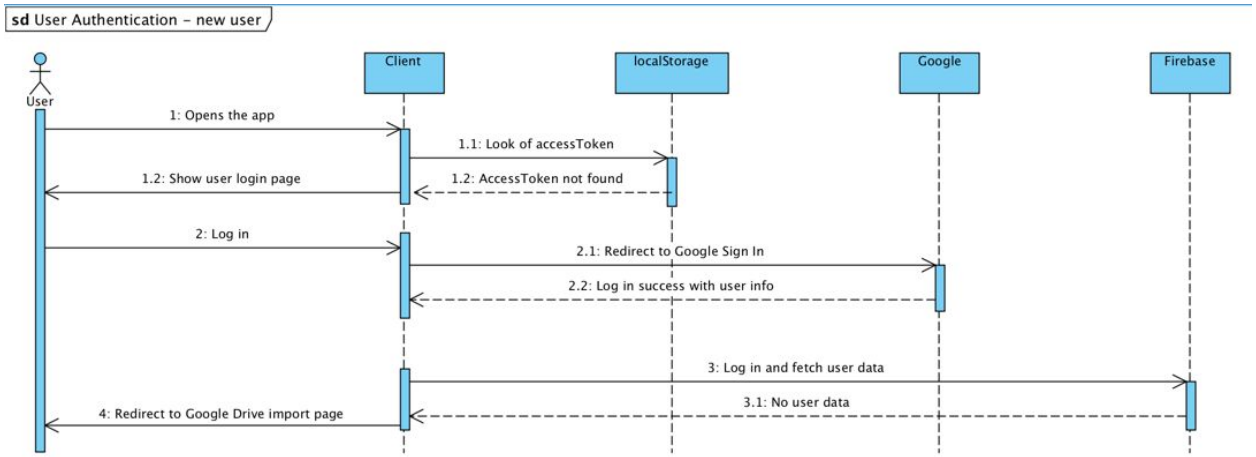
- MusicPlayer is the abstraction for interface of the music playing page. It is where the user make interactions with the application. It plays one Music instance in a Playlist instance at a time.
- Contains current Music and Playlist. Also volume and play mode can be changed as user preferences so they are contained here. PlayMode is an enumeration described in the same diagram.
- User can be able to perform operations in MusicPlayer, such as changing volume, changing play mode, playing music, stopping music, playing next music, and playing previous music. The rest are setters and getters for private fields.

PlayMode:

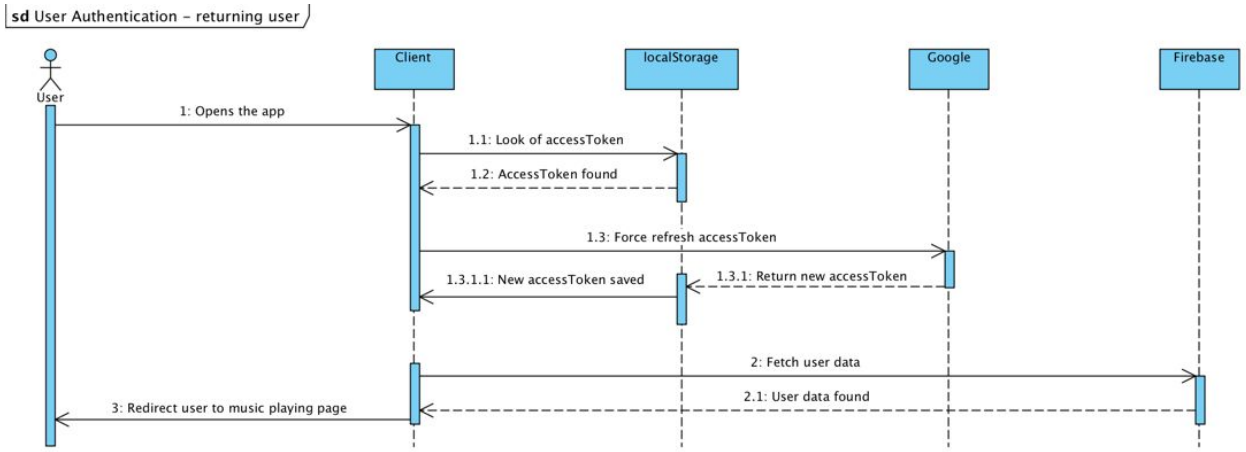
- PlayMode is the abstraction for five different play mode of music. It is used by MusicPlayer instances in playMode fields.
- Contains five different play mode of music including shuffle, repeat all, repeat once, and order.

Sequence Diagram

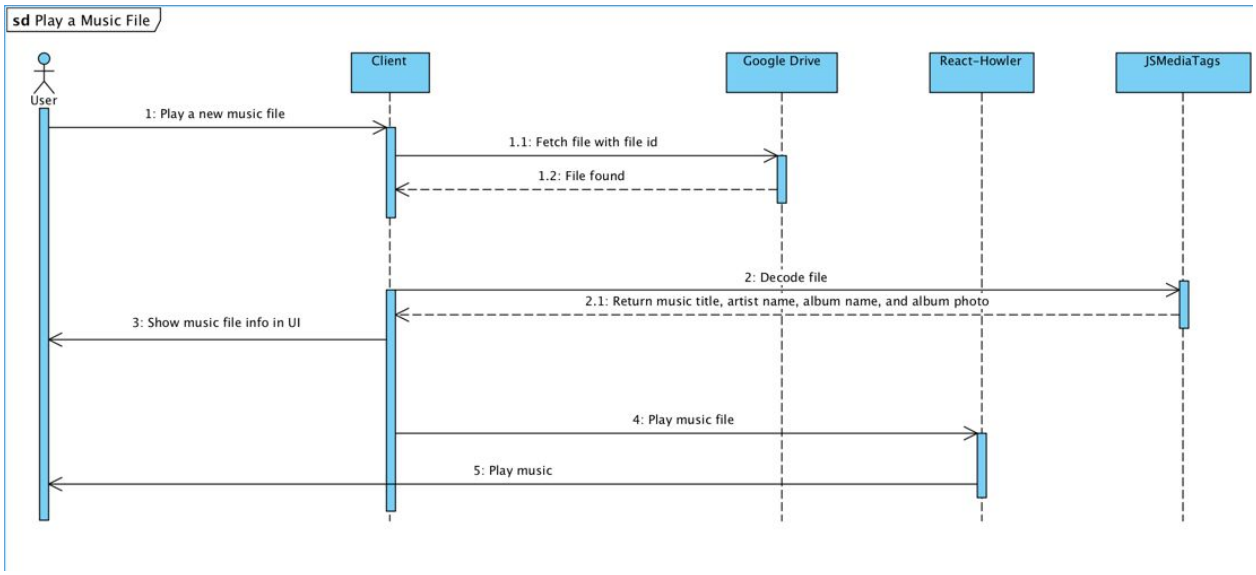
User Authentication for new user



User Authentication for returning user

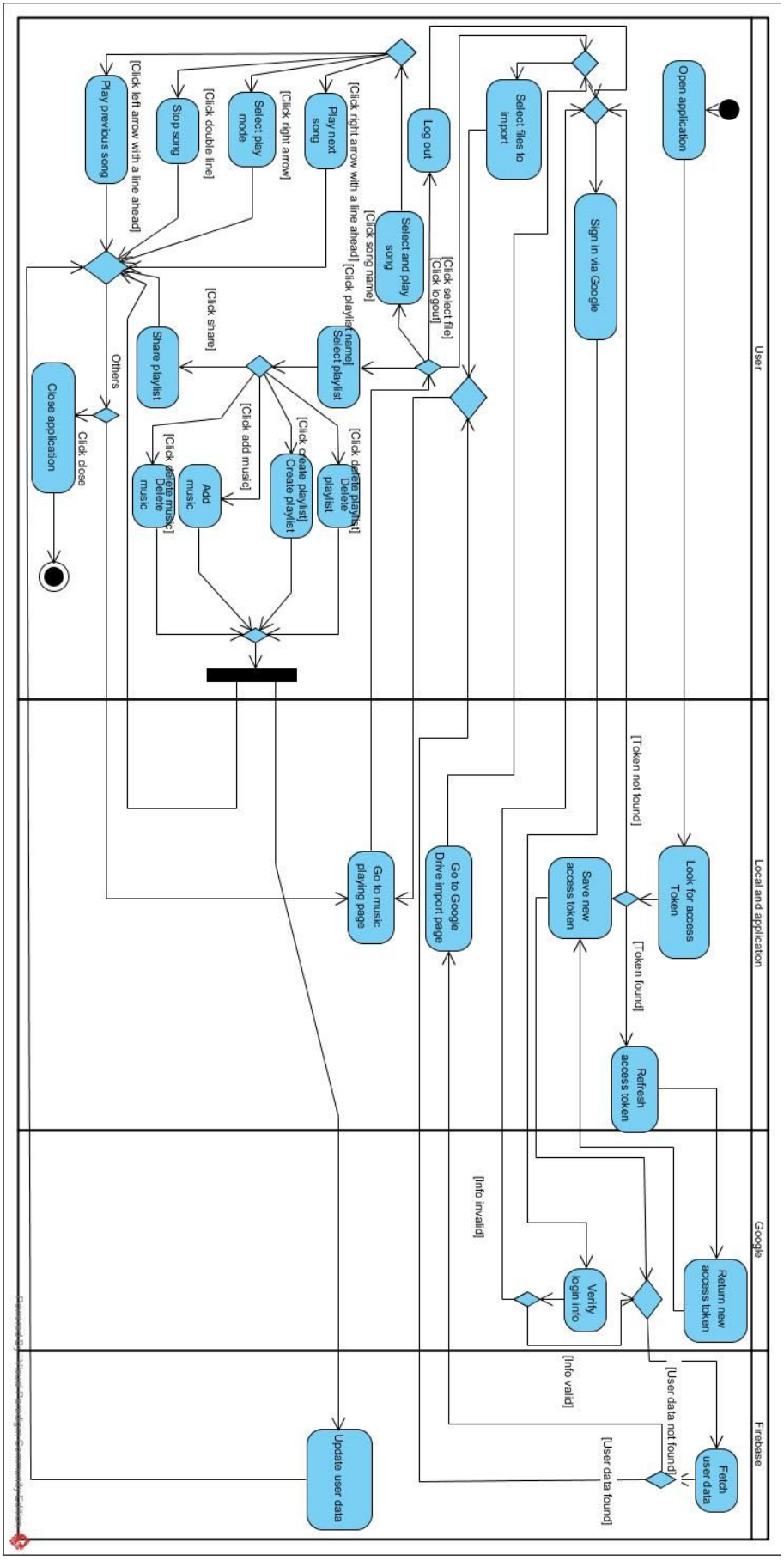


Play Music


sd Play a Music File
User
Client
Google Drive
React-Howler
JSMediaTags
1: Play a new music file
1.1: Fetch file with file id
1.2: File found
2: Decode file
2.1: Return music title, artist name, album name, and album photo
3: Show music file info in UI
4: Play music file
5: Play music

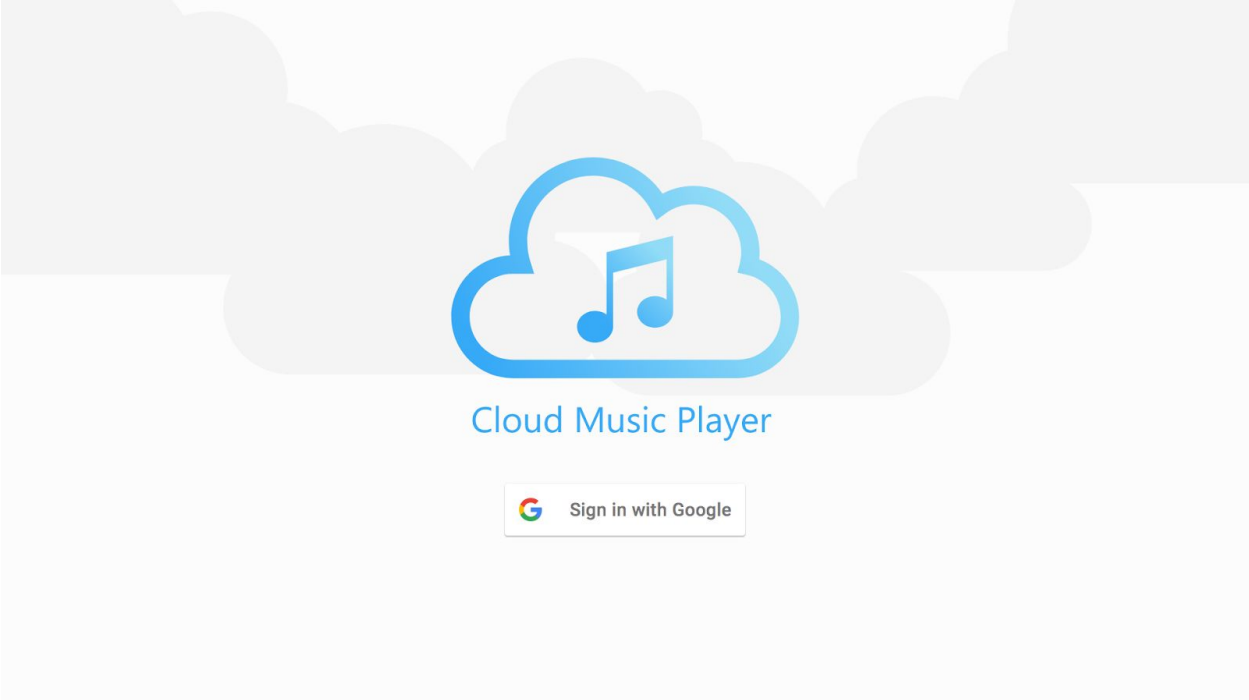
Activity Diagram

This is a detailed activity diagram for the actions of users, local and applications, Google, and database. When the user opens the application, the application checks if access token exists. This involves actions of Google and Firebase. The returning user with access token is directed to music playing page with user data retrieved successfully in Firebase. The new user without access token is directed to the login page, and he or she can sign in via google here. This involves verification in Google. If login valid, user data like playlists will be retrieved from Firebase. However, nothing will be found, so the new user will be directed to Google Drive import page. After finishing selecting music files, the new user will be directed to the music playing page as well. In the music playing page, user can perform various operations like playing music, create playlist, etc. Operations like creating playlist, deleting playlist, adding music to playlist, and deleting music from playlist are result in updating user data in Firebase. Finally, user can close application and the activities reach the end. Note that user can close application in nearly every step but to be make diagram look cleaner these control flows are not specified.

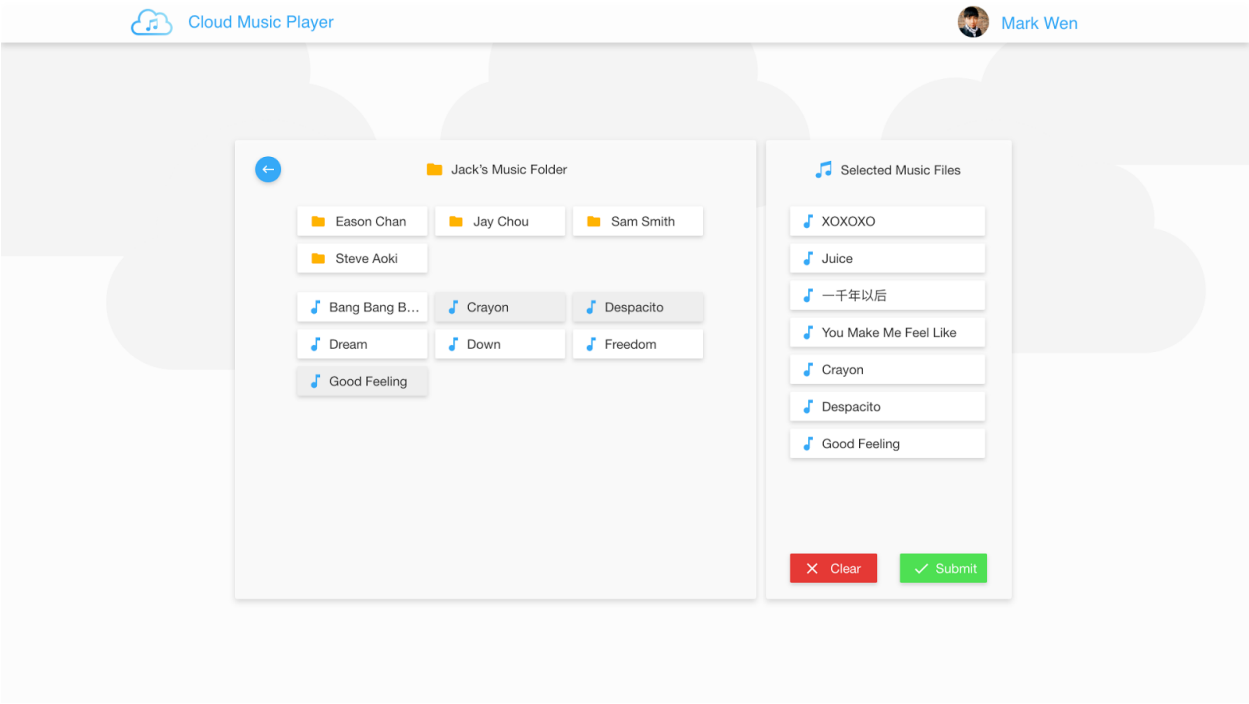

User
Local and application
Google
Firebase
Open application
Select files to import
Sign in via Google
Log out
Select and play song
Play previous song
Stop song
Select play mode
Play next song
Share playlist
Select playlist
Add music
Delete music
Create playlist
Delete playlist
Close application
Click close
Others
[Click left arrow with a line ahead]
[Click double line]
[Click right arrow]
[Click right arrow with a line ahead]
[Click song name]
[Click playlist name]
[Click share]
[Click add music]
[Click delete music]
[Click create playlist]
[Click delete playlist]
[Click select file]
[Click logout]
Look for access Token
Save new access token
Refresh access token
Go to Google Drive import page
Go to music playing page
[Token not found]
[Token found]
[Info invalid]
[Info valid]
Return new access token
Verify login info
Fetch user data
Update user data
[User data not found]
[User data found]

UI Mockups

We designed the UI such that users can have a simple and straightforward experience using our application. User first enters the login screen if user has not previously logged in with their Google account. Otherwise, user will go directly into the music playing page.



After login, user enters to the music import page, where user navigates and select music files from his/her account to import into our application.



Finally, user enters the music playing page, where they can play music, as well as create and share playlists.