Report File

Video Link: https://drive.google.com/file/d/1gzne0N37TszTx8kkuQj2q09MEkWIx3W0/view

- There are 2 folders in my main project folder named frontend and backend.
- The backend folder contains all code related to the backend of the website and the frontend folder contains all code relevant to the frontend of the website.
- In the backend folder there are 2 folders: web and instance, and main.py, requirements.txt file.
- The backend has all main files used to run the server properly.
    - **Init.py** : It contains all configuration related files.
    - **Api.py** : It contains all api information.
    - **Auth.py** : It has Login, Signup, Logout and JWT implementation.
    - **Package.py** : It contain database, cache, and mail instances
    - **Mail.py** : It has a general mail implementation for sending mails.
    - **Models.py** : It has all models and tables information made in the database.
    - **Views.py** : It has all other classes implementation that is Dashboard, Song, Playlist etc.
    - **Worker.py** : It has all info regarding async jobs and celery tasks.
- The database file will be in the instance folder which will be inside the backend folder and our init file automatically creates our database file with an admin user if it does not exist.
- There is a templates folder inside the Web folder of the backend which has a report.html template which is used when we have to send a monthly report of music streaming rating to the creator.
- Now my models has mainly 4 tables:
    1. **User table:** Represents a user with unique email, password, and name. Has albums, playlists, and ratings relationships. Tracks last login and has a default role of 'user'.
    2. **Album Model:** Represents an album with a unique ID and name. Belongs to a user and has songs associated with it.
    3. **Song Model:** Represents a song with details like name, rating, lyrics, and duration. Has ratings and playlist relationships.
    4. **Rating Model:** Represents a user's rating for a specific song. Associates with both the user and the song.
    5. **Playlist Model:** Represents a user's playlist with a unique ID and name. Belongs to a user and has playlist songs associated.
    6. **PlaylistSong Model:** Represents the association between playlists and songs.

- Now my frontend folder has a src folder. It has most of the components required to start the vue server website.
- The src folder has following structure :
    - Lyrics : This folder contains uploaded songs of a creator.
    - Assets : This folder contains image files for my website.
    - Router : It has an index.js file for my router setup.

    - Store : It has a file to configure my store.
    - Views : It has all pages of my website.
    - **App.vue** : Used to connect all these things with the main.js file and start the server, can be considered as a main page in which all the other pages change one by one.
- The structure of my Views folder is given below:
    1. **AdminLogin.vue** : Used to login as admin.
    2. **Album_add.vue** : Used to add albums by the creator.
    3. **Dashboard.vue** : the place where user, creator and admin home page present.
    4. **Edit_album.vue** : Used to edit the album.
    5. **HomeView.vue** : This is the home page and landing page.
    6. **Login.vue** : Used to login user and creator.
    7. **Playlists.vue** : Used to create and show a playlist.
    8. **PlaylistSong.vue** : Used to add the song and show the song in a playlist.
    9. **Signup.vue** : Used to create new users and creators.
    10. **Song_action.vue** : Used to edit and delete songs.
    11. **Song_add.vue** : Used to add songs.
    12. **ViewLyrics.vue** : Used to view songs and rate songs.
- There is a requirements.txt file which specifies the libraries and header files used by this web application to run properly.
- There is a search box functionality for the user to search songs based on the name of the albums, genre, song name and the ratings of the songs.