- **Author Information:**
    - *Name* : Patel Nujhatbanu Idrisbhai
    - *Roll No* : 21f3001798
    - *Email* : 21f3001798@ds.study.iitm.ac.in
    - I am currently pursuing a diploma in Data Science and Programming. Having completed my diploma in Data Science, I am now working on projects in MAD1 and MAD2. My primary focus is on machine learning, and I look forward to exploring it further in my academic pursuits.
- **Description:**
    - This music app enables users to create accounts using unique email IDs, upload music, view average song ratings, and manage personal albums and playlists by adding or deleting songs. Users have control over their profiles, showcasing the number of uploaded songs, and can search for music. Additionally, the admin possesses the authority to blacklist/whitelist users, restricting song uploads for those in the blacklist. The app also provides features for admin to view top songs and various charts.
- **Technologies Used:**
    - *Flask:* Minimalistic Python web framework for backend development.
    - *Flask-SQLAlchemy:* Simplifies database integration with SQLAlchemy for user and music data.
    - *Matplotlib*: Data visualization library for creating charts in the app.
    - *Flask- Login:* Manages user sessions, authentication, and personalized profiles.
    - *HTML, Jinja Templates, Bootstrap:* Standard web technologies for frontend design and user interface.
- **DB Schema Design:**
    - User Table:
        - id, email, password, first_name, last_name, white_list
        - Relationships: playlists, songs, albums
    - Song Table:
        - id, name, lyrics, duration, singer, genre, created_at, ratings, num_ratings
        - Relationships: user_id, albums, playlists
    - Rating Table:
        - id, user_id, song_id, rating
    - Playlist Table:
        - id, name, created_at
        - Relationships: user_id, songs
    - Album Table:
        - id, name, genre, artist, created_at

        - Relationships: user_id, album_songs
- **Architecture and Features:**
    - The project is organized within the zip folder named "21f3001798," which contains essential components for the music app. Inside this folder, there is an "instance" folder housing the database file, "database.db." The "website" folder encompasses key elements such as the "static" folder for CSS and music files, the "templates" folder for Jinja templates, and various Python files, including "model.py", "playlist.py", "admin.py" and "album.py." The main functionality is handled by the "main.py" file. Controllers are distributed across these Python files, each responsible for specific aspects of the application, such as user authentication, song uploading, playlist and album management, and database interactions. Templates in the "templates" folder are designed using HTML and Jinja, incorporating Bootstrap for styling. Default features include user account creation, song uploading, playlist and album creation, and user profile management. Additionally, users can rate songs, and the app calculates average ratings. Administrative features include user blacklisting/whitelisting, and admin access to top songs and charts. Overall, the project follows a modular structure, with each component contributing to the seamless functioning of the music app.
- **Video:**
    - https://drive.google.com/file/d/1vJ3UQ_A-FLyjshshN7x5jgZ4DV0pnG2W/view?usp=drive_link