**Name: Varad Uttarwar**

**Roll: 22f2001664**

**Topic: Music Streaming web application**

**Course: MAD1 Project**

Presentation video link: <https://drive.google.com/file/d/1S2YLjRzEQFIyTJV6G57-4LmrWXQPyuvH/view?usp=drive_link>

Introduction

This project revolves around the Beatx Music Streaming web application. Participation is open to users and creators through registration. Users have the ability to listen to a vast library of uploaded songs, read lyrics, and navigate songs based on timestamps. The creation of playlists and the addition of uploaded songs to these playlists are features available to users. Creators, on the other hand, can contribute to the platform by uploading songs in the form of mp3 files, along with song profile pictures and lyrics. The creator's dashboard offers basic statistical insights into their account.

Administrative accounts hold the authority to delete or flag songs. The admin dashboard provides a comprehensive view of overall user and creator statistics, including information on the best and most popular songs, as well as albums. The admin can review lyrics and take action on flagged or undesirable songs within the database. The application encompasses various routes such as register/login, admin login, home page, creator account, create album, logout, and upload songs.

The home page is designed to showcase highly-rated songs, user playlists, albums, and genre-related songs. In the creator's account, a prominent "+" sign encourages registration as a creator if not already registered, prompting them to initiate the process of uploading songs. The required information for song uploads includes the song name, date, singer, lyrics, mp3 file, and image file.

The dashboard in the creator's account provides an overview of the creator's uploaded songs. It's worth noting that this application is in active development, with ongoing efforts to incorporate additional features to enhance user-friendliness.

Features Used

Werkzeug provides essential tools for handling HTTP requests, routing, and other web-related tasks. In the context of my project, Werkzeug played a crucial role in ensuring smooth communication between the web server (Flask) and the application, handling aspects like request parsing and URL routing.

In this music streaming web application, Requests likely facilitated communication with external APIs or services, allowing the application to fetch and integrate external data seamlessly. This could include retrieving information about artists, albums, or songs from external music databases or APIs.

Matplotlib is used to display and visualize the stats for creator and admin.

Fuzzywuzzy is a library that provides fuzzy string matching capabilities. Fuzzywuzzy is used for task for improving search functionality or matching user input with music titles, artists, or album names more accurately.

Technologies Used

In the development of my music streaming web application, I employed a robust stack of technologies to ensure a seamless and engaging user experience. The backend of the application is powered by Flask, a lightweight and modular web framework for Python. Flask provides a solid foundation for handling server-side logic, managing routes, and integrating with the Jinja templating engine.

Additionally, to manage the persistence of data in my music streaming web application, I incorporated SQLite as the relational database. SQLite is a lightweight, serverless database engine that seamlessly integrates into Flask applications, providing a reliable and efficient solution for handling data storage.

Jinja templating played a pivotal role in creating dynamic and data-driven HTML templates. By embedding placeholders within the HTML markup, Jinja allowed for the seamless integration of server-side data, ensuring that the content presented to users is always current and personalized.

On the frontend, HTML and CSS were instrumental in shaping the static structure and visual design of the web pages. HTML defined the elements and layout of the user interface, while CSS was employed to style and enhance the presentation, creating an aesthetically pleasing and responsive design. The utilization of CSS was particularly crucial in ensuring a consistent and visually appealing layout across various devices and screen sizes.

To inject interactivity and dynamic features into the application, JavaScript played a key role on the client side. Leveraging JavaScript, I implemented functionalities such as asynchronous data loading, real-time updates, and user interactions. This not only enhanced the user experience but also allowed for seamless navigation and exploration within the music streaming platform.

Challenges Faced

Redundant Code with Jinja:

Using Jinja for templating can sometimes lead to redundant code, especially when dealing with complex and dynamic web applications. Frontend frameworks like React or Vue.js are designed to handle such scenarios more efficiently. They provide a component-based architecture that promotes reusability, making it easier to manage and organize your code. With the virtual DOM, these frameworks can update only the parts of the UI that have changed, reducing the need for manual DOM manipulation and minimizing redundancy.

Search Functionality with Limited JavaScript:

Implementing search functionality can indeed be challenging, especially when constrained by limited JavaScript. Dynamic searches often require client-side scripting to provide a smooth and responsive user experience. Frameworks like React or Vue.js excel in handling such dynamic interactions. They enable the creation of interactive components that can update in real-time as user’s type, providing a more dynamic and user-friendly search experience.

Implementing Upload Songs with Lyrics Display and Playback:

Handling file uploads, displaying lyrics, and implementing playback features involve a combination of frontend and backend development. JavaScript plays a crucial role in managing the user interface and handling user interactions. Frameworks like React or Vue.js can simplify the process of creating interactive components for uploading files and displaying dynamic content. Additionally, dealing with file extensions on the backend requires careful validation and processing, which can be streamlined with the help of backend frameworks or libraries.

Conclusion

Overall, it was interesting project and most of the part shown in wireframe is implemented. I found it interesting to apply the learning done in APP Dev1 course. It helped me make websites where people could use them in real life.