🎵 **Music Recommendation System** 🎵

**----------------------------------------------------**

**Team Members:**

* **Mohab Mohamed – 22010273**
* **Ahmed Farag – 22010028**
* **Ziad Shalaby – 22010343**
* **Ahmed Mohamed – 22010030**
* **Ahmed Abdelghani - 22010024**

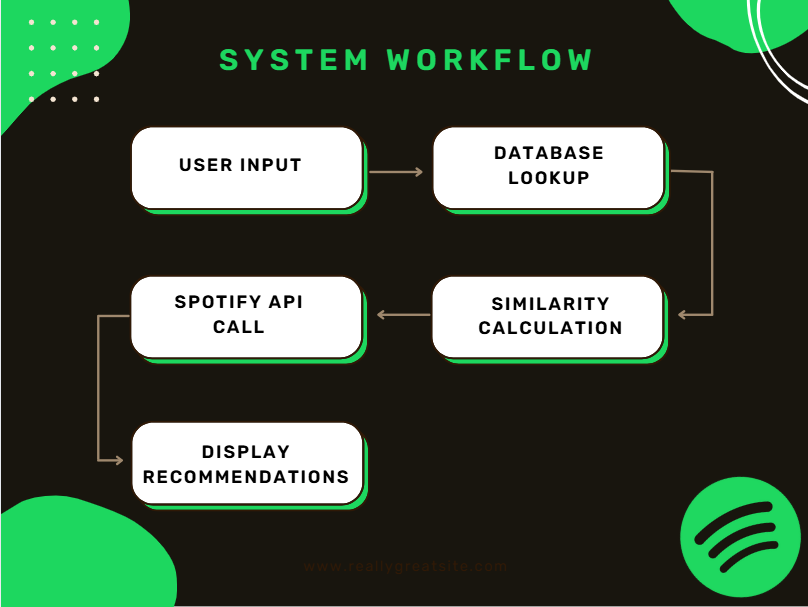
**Overview**

**-------------------**

Music is an integral part of people's lives, and personalized recommendations enhance the listening experience by introducing users to songs they are likely to enjoy. This project focuses on building a Music Recommendation System by integrating data scraping, machine learning, and web app deployment.

The system leverages data collected from Spotify and an external music dataset to provide users with dynamic recommendations. By implementing machine learning techniques, the project identifies patterns in music preferences and suggests the top five related songs based on user input. The recommendations are enhanced by integrating the Spotify API to fetch song details and album posters, creating an engaging user interface.

The recommendation system is deployed as an interactive Streamlit web application, enabling users to search for songs by name and receive recommendations with visual elements. This project demonstrates expertise in data collection, database management, machine learning, and web app development.

****

**Key Features of the Project**

**--------------------------------------**

1. **Data Collection and Storage**:
   * Scraped data from Spotify's platform to gather initial information.
   * Utilized SQLite to store and manage scraped data for efficient querying and processing.
2. **Data Parsing and Preparation**:
   * Developed Python scripts to parse the scraped data into structured Data Frames for analysis.
   * Searched for and incorporated an extensive dataset of music tracks from Kaggle to enrich the database with diverse songs and attributes.
3. **Recommendation System**:
   * Built a **music recommendation system** using machine learning techniques to provide personalized suggestions.
   * The system suggests a list of the **top 5 related songs** based on user queries.
4. **Integration with Spotify API**:
   * Leveraged the Spotify API to fetch song posters and enhance the visual appeal of the recommendations.
   * Implemented a search feature where users can input a **song name** and retrieve details about the song along with recommendations.
5. **Web App Deployment**:
   * Deployed the recommendation system as an interactive **web application** using Streamlit.
   * Enabled a user-friendly interface to input song names and receive recommendations dynamically.
6. **Scalability and Extensibility**:
   * Designed the system to be scalable, allowing easy integration of additional datasets or features.
   * Provides flexibility for future enhancements, such as adding user-based or artist-based recommendations.
7. **Visualization and Insights**:
   * Incorporated visual elements, such as posters and intuitive layouts, to improve user experience.
   * Streamlined recommendations to display meaningful results tailored to the input.
   * Incorporated a dashboard to visualize key insights, such as genre trends, popular songs, and user preferences, providing an analytical overview of the data.

**Implementation Details**

**---------------------------------------------------**

**Data Collection and Preparation**

* Data scraping scripts were implemented to gather metadata from Spotify.
* Data cleaning and structuring were carried out using Pandas, ensuring high-quality inputs for the recommendation system.

**Recommendation Algorithm**

* A similarity-based model computes distances between songs, ranking them based on relevance to the user's input.
* The songs.pkl file stores the song database, while similarity.pkl maintains precomputed similarity scores for efficiency.

**Spotify API Integration**

* **Authentication**: Used Spotify API's Client Credentials Flow to authenticate and fetch tokens dynamically.
* **Endpoints**: Integrated with Spotify’s track search API to retrieve metadata and album art.

**Streamlit Web App**

* **Interface**: Dropdown for song selection and a button to trigger recommendations.
* **Visualization**: Display of song recommendations as clickable album posters, linking to Spotify previews.

**Challenges and Solutions**

1. **Data Quality**: Ensured consistent formatting through preprocessing steps.
2. **API Rate Limits**: Implemented caching mechanisms to reduce redundant API calls.
3. **UI Optimization**: Used Streamlit's layout features for an engaging user experience.

**Results and Benefits**

* **Accurate Recommendations**: A robust similarity-based system delivers precise suggestions.
* **User Engagement**: The interactive web interface and visual elements encourage repeat use.
* **Scalable Design**: Allows for future enhancements with minimal redevelopment.

**Conclusion**

The Songs Recommendation System demonstrates the integration of machine learning, API services, and web technologies to deliver a practical and user-friendly application. Its scalable design ensures adaptability for future features and larger datasets.

**Future Work**

* Adding user-specific recommendations using collaborative filtering.
* Enhancing UI/UX with additional customization options.
* Incorporating more detailed insights, such as genre or artist trends.

**Project Visuals**

**------------------------**

**Web App Interface**

A black screen with white text

Description automatically generated

**Recommendation Output**

**A screenshot of a computer

Description automatically generated**