Music Streaming App V2

Problem definition

Modern Application Development - II

Last Updated: 23-09-03

Frameworks to be used

- Flask for API
- VueJS for UI
- VueJS Advanced with CLI (only if required, not necessary)
- Jinja2 templates if required
    - Not to be used for UI
- Bootstrap, if required (No other CSS framework is allowed)
- SQLite for database (No other database is allowed)
- Redis for caching
- Redis and Celery for batch jobs
- It should be possible to run all the demos on the student's computer, which should either be a Linux based system or should be able to simulate the same. You can use WSL for Windows OS.

Music Streaming App

- It is a multi-user app (one required admin and other users or creators)
- Used for streaming music/reading lyrics
- User can play, read lyrics, rate songs, create playlists
- Creators can add new songs/albums/lyrics
- Each Album may have
    - ID
    - Name
    - Artist etc.
- Each song will have
    - ID
    - Name
    - Lyrics
    - Genre
    - Duration
    - Date created etc.
- Every album can have a number of songs
- System will automatically show the latest albums/songs added

Terminology

- Albums, Playlists
- User, Creator, Admin
- Songs - lyrics

Similar Products in the Market:

1. Saavn App
   - Web, IOS and Android
2. Spotify
   - Web, IOS and Android

Refer to the wireframe given below;
Music Streaming App

- These are meant for exploring the idea and inspiration
- Don’t copy, get inspired

Core Functionality

- This will be graded
- Base requirements:
  - Admin’s Dashboard
  - General User signup and login (using RBAC)
  - Mandatory Admin Login (using RBAC)
  - Creator Signup and Login (using RBAC)
  - General User Profile
  - Creator Profile (users registered as creators)
  - Song Management
  - Playlist Management
  - Search functionality for songs/playlist
- Backend Jobs
  - Export Jobs
  - Reporting Jobs
  - Alert Jobs
- Backend Performance

Core - Admin and User Login

- User should be authenticated using username or email and password
- Use Flask Security or JWT based Token Based Authentication only
- Registration form for General User and Creator is required. You can use the same form for both.
- New admin cannot be created.
- Suitable model for user (model that stores all the type of users and differentiates them correctly based on their role)
- The application should have only one admin.

Core - General User functionalities

- View all the existing songs/Albums
- Read lyrics, create playlists of existing songs (one or more)
- Rate existing songs/albums
- Flag a song/album

Core - Creator functionalities

- All the functionalities of a general user
- Create a new song/album
    - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit an existing song/album
    - Change lyrics, artists name, song attributes, album name, album listing
- Remove an existing song/album
- Assign a song to a particular album

Core - Admin functionalities

- Admin dashboard to monitor app statistics like total users/creators/albums
- Statistics on how songs are performing
- Ability to remove songs/albums

Core - Search for songs/albums

- Ability to search albums based on artists, genre etc.
- Ability to filter songs based on rating

Core - Daily Reminder Jobs

- Scheduled Job - Daily reminders on Google Chat using webhook or SMS or Email
    - In the evening, every day (you can choose time of your choice)
    - Check if the user has visited the app.
    - If no, then send the alert asking them to visit

Core - Scheduled Job - Monthly Activity Report

- Scheduled Job - Monthly Activity Report
    - Come Up with a monthly progress report in HTML or PDF (email)
    - The monthly activity report is meant for a creator
    - The activity report can consist of songs & albums created, ratings received on songs/albums etc.
    - On the first day of the month
        - Start a job
        - Create a report
        - Send it as email

Recommended (graded)

- Add and play .mp3 files for songs
- Validation
  - All form inputs fields - text, numbers, dates etc. with suitable messages
  - Backend validation before storing / selecting from database

Optional

- Styling and Aesthetics
- Subscriptions or paid versions of the app
- Autoplay, shuffle songs in a playlist

Evaluation

- Report (not more than 2 pages) describing models and overall system design
  - Include as PDF inside submission folder
- All code to be submitted on portal
- A brief (2-3 minute) video explaining how you approached the problem, what you have implemented, and any extra features
  - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions
  - This includes making changes as requested and running the code for a live demo
  - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details and FAQs (possibly including suggested wireframes, but not specific implementation details) as we proceed.
- We will freeze the problem statement on or before 2nd October, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.