UGAHacks X Project Log

Project Information

Project Title: RockVerse

Team Members: Molly Sohaney

Tier Level: 2 (Intermediate)

Project Description: RockVerse, a web application where users can generate rock lyrics based on their desired style, mood, and keywords. This project uses session-based authentication and AI-powered lyrics generation with the Cohere API

Friday

Goals:

- [ ] Brainstorm project ideas
- [ ] Start development

Progress:
Brainstormed ideas:

- Rock Lyrics Generator – Generates rock-style lyrics based on user input.
- Rock Band Name Generator – Suggests creative band names inspired by rock themes.
- Mood-Based Playlist Generator – Recommends rock playlists based on user mood.

Challenges:
1. Choosing the best programming language/framework for implementation
    - Potential solution: Considering Python (Flask), React, and JavaScript.

Learning:
React
Python Flask
Cross-Origin Resource Sharing (CORS)

AI Usage (if any):

Tool used: ChatGPT
Purpose: Assistance with React

How it contributed to learning: Provided foundational guidance, helping me focus on key React concepts as I begin learning the framework.

Saturday

Saturday is the longest day of the Hackathon! The bulk of your project will get done within today, so set your goals wisely!

Goals:

- [ ] Complete Frontend
- [ ] Complete Backend
- [ ] Expand to Login

Progress:

- Switched from OpenAI to Cohere for API calls
- Frontend for dashboard completed
- Backend for dashboard completed

Challenges:

1. The API calls to OpenAI weren't being executed as expected. When I noticed that no responses were coming through for the API requests, I attempted to manually test the API using curl.

- **Solution**: During this manual test, I received an error indicating that my API call limit had been exceeded, even though it was my first attempt. After investigating further, I discovered that my OpenAI usage had hit an unexpected restriction. As a result, I decided to switch to a different AI service, Cohere, for generating lyrics, which resolved the issue and allowed the project to continue smoothly.

Learning:

Cohere
React.js
CSS
Flask Session

AI Usage (if any):

Tool used: ChatGPT, Cohere
Purpose: ChatGPT used for assistance with Flask Session. Cohere used for API calls
How it contributed to learning: Helped with implementation of Flask Session in the application.

Sunday

Submissions are due at 8AM today!! Fit in your final touches for the project and make sure to check the submission checklist below to ensure you're ready for judging!

Goals:

- [ ] Create database for users
- [ ] Frontend and Backend for Login Page
- [ ] Frontend and Backend for Signup Page
- [ ] Testing

Progress:
Used PostgreSQL to create database for user information
Added login and signup page
Finished frontend and backend for all web pages
Switched from JWT to Flask Session for user authentication
Conducted testing
Finished Readme
Completed Project Log
Recorded Demo Video

Challenges:
1. Initially, I implemented JWT (JSON Web Tokens) for user authentication. The goal was to use JWT to securely authenticate users during login and ensure that only authorized users could access protected routes, like the dashboard. However, I ran into multiple issues while working with JWT. Despite successfully generating tokens during login, I faced difficulties with token validation during subsequent requests, leading to authentication failures. The system was not properly recognizing valid tokens, causing issues with protected routes. This resulted in users being unable to access their personalized dashboards after logging in.

- **Solution**: After troubleshooting the JWT implementation and encountering challenges with token validation and expiration handling, I decided to switch to Flask-Session for managing user authentication. With Flask-Session, the session information is stored server-side, and the session ID is saved as a cookie on the client. This simplified the process as it removed the need for handling JWT tokens manually. By switching to session-based authentication, I was able to reliably authenticate users and store their session data across requests, ensuring that only logged-in users could access protected pages like the dashboard. This approach also allowed me to avoid some of the common pitfalls of JWT handling, such as token expiration and renewal.

Learning:
PostgreSQL
Flask Session
Flask Bcyrpt

AI Usage (if any):

Tool used: ChatGPT
Purpose: Convert MySQL queries to PostgreSQL
How it contributed to learning: The syntax for PostgreSQL and MySQL is so different but I wanted to make sure I am using the correct query syntax

---

Submission Checklist

Make sure to submit on the UGAHacks __ Devpost at 8AM on Sunday!

- [x] Project Github Repo
- [x] Readme file (summary of project log)
- [x] Completed Project Log\ as PDF
- [ ] Live Project Site (optional)