comedian_joke_book

Lalo adrian morales

lablab.ai hackathon March 15, 2024

An AI-driven platform for exploring, creating and protecting original humor. Discover joke trends, author original content, and leverage NLP to ensure authenticity.

Comedian Joke Book is a groundbreaking platform that combines the power of AI with the creativity of humor enthusiasts worldwide. Submit your best original jokes and let our advanced natural language processing categorize them by theme, structure and style. Explore the resulting interactive joke visualization to discover new talent, uncover emerging trends, and find inspiration for your own comedy writing.

Most importantly, JokeMap is dedicated to protecting the intellectual property of humor creators. Our originality scoring engine compares new submissions against our vast database, flagging potential plagiarism and giving proper attribution. With a transparent, immutable record of each joke's origin and timeline, JokeMap aims to become the trusted home for comedic talent everywhere.

Whether you're an aspiring comedian, a prolific humor writer, or simply a connoisseur of clever jokes, JokeMap provides the tools and community to help you explore, create and preserve the very best in original comedy. Join us on this exciting journey as we map the frontiers of humor!

Create The Files In Terminal

```
mkdir comedian_joke_app
cd comedian_joke_app

mkdir backend
mkdir frontend

# Backend directories and files
cd backend
mkdir api
mkdir database
mkdir models
mkdir services
mkdir utils

touch api/__init__.py
touch api/joke_routes.py
touch api/auth_routes.py

touch database/__init__.py
touch database/db_config.py
touch database/joke_model.py

touch models/__init__.py
touch models/joke.py
touch models/user.py

touch services/__init__.py
touch services/audio_service.py
touch services/transcription_service.py
touch services/language_model_service.py

touch utils/__init__.py
touch utils/helpers.py

touch app.py
touch requirements.txt

cd ..

# Frontend directories and files
```

```
cd frontend
mkdir public
mkdir src

cd public
touch index.html
touch favicon.ico

cd ../src
mkdir components
mkdir pages
mkdir services
mkdir styles

touch components/JokeCard.js
touch components/JokeVisualizer.js

touch pages/Home.js
touch pages/JokeList.js
touch pages/JokeAnalytics.js

touch services/api.js

touch styles/global.css

touch App.js
touch index.js

cd ..
touch package.json
```

1. `backend/database/db_config.py`:
   - Configure the database connection settings.
   - Set up the database client (e.g., MongoDB, PostgreSQL).
   - Define functions to establish and close database connections.
2. `backend/models/joke.py`:
   - Define the data model for a joke.
   - Include fields like `joke_id`, `joke_name`, `joke_description`, `timestamp`, `percentage`, and `love_it`.
   - Implement methods for creating, reading, updating, and deleting jokes.
3. `backend/services/audio_service.py`:
   - Implement functions for handling audio file uploads and storage.
   - Define methods for processing audio files, such as converting formats or extracting metadata.
4. `backend/services/transcription_service.py`:
   - Integrate with a speech-to-text API or service (e.g., Google Cloud Speech-to-Text, AWS Transcribe).
   - Implement functions to transcribe audio files and retrieve the transcribed text.
5. `backend/services/language_model_service.py`:
   - Integrate with a language model API (e.g., OpenAI GPT, Hugging Face Transformers).
   - Implement functions to process the transcribed text and extract relevant information, such as joke name, description, and tags.
6. `backend/api/joke_routes.py`:
   - Define API routes for handling joke-related operations.
   - Implement endpoints for creating, retrieving, updating, and deleting jokes.
   - Use the joke model and services to perform the necessary operations.
7. `backend/app.py`:
   - Set up the main backend application.
   - Initialize the database connection.
   - Register the API routes.
   - Configure middleware and error handling.

After completing the backend files, you can move on to the frontend files:

8. `frontend/src/services/api.js`:
   - Implement functions to make API calls to the backend endpoints.
   - Handle request and response data formatting.
9. `frontend/src/components/JokeCard.js`:
   - Create a reusable component for displaying a single joke.
   - Include fields like joke name, description, timestamp, percentage, and love it.
10. `frontend/src/components/JokeVisualizer.js`:
    - Implement a component for visualizing jokes based on keywords or tags.
    - Use a library like D3.js to create interactive bubble visualizations.
11. `frontend/src/pages/Home.js`:
    - Create the home page component.
    - Integrate the joke list and visualizer components.

- Implement search and filtering functionality.

12. `frontend/src/pages/JokeList.js`:
    - Create a component for displaying a list of jokes.
    - Implement pagination or infinite scrolling.
    - Allow sorting and filtering based on joke properties.
13. `frontend/src/pages/JokeAnalytics.js`:
    - Create a component for displaying joke analytics and insights.
    - Visualize joke performance over time using charts or graphs.

Backend files:

14. `backend/models/user.py`:
    - Define the data model for a user.
    - Include fields like `user_id`, `username`, `email`, and `password`.
    - Implement methods for creating, reading, updating, and deleting users.
15. `backend/api/auth_routes.py`:
    - Define API routes for handling user authentication.
    - Implement endpoints for user registration, login, and logout.
    - Use the user model and authentication middleware.
16. `backend/utils/helpers.py`:
    - Implement utility functions and helper modules.
    - Include functions for data validation, error handling, and common operations.
17. `backend/requirements.txt`:
    - List the required Python dependencies for the backend.
    - Include packages like Flask, PyMongo (for MongoDB), or SQLAlchemy (for SQL databases).

Frontend files:

18. `frontend/public/index.html`:
    - Create the main HTML file for the frontend application.
    - Set up the basic structure and include necessary meta tags and links.
19. `frontend/public/favicon.ico`:
    - Add a favicon for the application.
20. `frontend/src/styles/global.css`:
    - Define global styles for the frontend application.
    - Include common styles, typography, and layout rules.
21. `frontend/src/App.js`:
    - Create the main component for the frontend application.
    - Set up routing and navigation between different pages.
    - Include common components like header and footer.
22. `frontend/src/index.js`:
    - Set up the entry point for the frontend application.
    - Render the main `App` component and mount it to the DOM.
23. `frontend/package.json`:
    - Specify the frontend dependencies and build scripts.
    - Include packages like React, React Router, Axios (for API calls), and D3.js (for visualizations).

Configuration files:

24. `.gitignore`:
    - Specify files and directories that should be ignored by Git version control.
    - Include common ignores like `node_modules`, `__pycache__`, and environment-specific files.
25. `README.md`:
    - Create a README file for the project.
    - Provide an overview of the project, installation instructions, and usage guidelines.
26. `LICENSE`:
    - Add a license file specifying the terms and conditions for using and distributing the software.