Author Name

Full Name: Shivam Gupta

Roll No. :21f1003942

I am a computer science student with a strong interest in web development and software engineering.

Description

This project aims to develop a web application using Flask and Vue combination to effectively manage show bookings. Database design was an important part of this app. Celery and celery-beat together help in easy execution of tasks in timely manner.

Technologies Used

- Flask: A micro web framework for building web applications in Python.

- SQLAlchemy: A SQL toolkit and Object-Relational Mapping (ORM) library for Python.

- HTML/CSS: For building the user interface and styling.

- JavaScript: For adding interactivity to the web application.

- SQLite: A lightweight, serverless database for storing task data.

- Redis: An in-memory data store used for caching and background job processing.

Purpose: For high-speed cache.

- Celery: A distributed task queue system that handles background processing of tasks asynchronously.

Purpose: For asynchronous tasks.

- Vue.js: A progressive JavaScript framework for building user interfaces.

Purpose: Vue.js is used for creating dynamic and responsive front-end components and enhancing user interactivity.

- Celery Beat: A scheduler that manages periodic tasks within the Celery framework.

Purpose: For scheduling tasks.

- Flask-JWT-Extended: An extension for Flask that provides JSON Web Token (JWT) authentication.

Purpose: Token-based Authentication.

- Purpose: These technologies were chosen for their compatibility and ease of use in developing a web-based task management system.

DB Schema Design

- **User**
  - Table Name: user
  - Columns:
    - id: Integer, Primary Key
    - role_id: Integer, Foreign Key to role.id, Not Nullable
    - username: String(50), Unique, Not Nullable
    - email: String(120), Unique, Not Nullable
    - password: String(100), Not Nullable
  - Description: Represents user information with a role association.

- **TheatreShow**
  - Table Name: theatreshow
  - Columns:
    - id: Integer, Primary Key, Autoincrement
    - theatre_id: Integer, Foreign Key to theatre.id, Not Nullable
    - show_id: Integer, Foreign Key to show.id, Not Nullable
  - Description: Represents the relationship between theatres and shows.

- **Show**

    - Table Name: show
    - Columns:
        - id: Integer, Primary Key, Autoincrement
        - name: String(150), Unique
        - duration: Time
        - rating: Float
        - tags: Text
        - price: Integer, Default 100
        - seat_available: Integer, Not Nullable
        - dateTime: DateTime
    - Description: Represents show information.

- **Order**

    - Table Name: order
    - Columns:
        - id: Integer, Primary Key, Autoincrement
        - show_id: Integer, Foreign Key to show.id, Not Nullable
        - user_id: Integer, Foreign Key to user.id, Not Nullable
        - venue_id: Integer, Foreign Key to theatre.id, Not Nullable
        - rated: Integer, Default 0, Not Nullable
        - seats: Integer, Not Nullable
        - created: DateTime, Not Nullable
    - Description: Represents user orders for shows.

- **Role**

    - Table Name: role
    - Columns:
        - id: Integer, Primary Key
        - name: String, Not Nullable, Unique
        - description: String, Unique
    - Description: Represents user roles.

- **Theatre**

    - Table Name: theatre
    - Columns:
        - id: Integer, Primary Key, Autoincrement
        - name: String, Not Nullable
        - capacity: Integer, Not Nullable
        - city: String, Not Nullable
        - place: String, Not Nullable
    - Description: Represents theatre information.

Reason

- Reasons: This design follows a relational database model to efficiently store user and task data, allowing for easy retrieval and management.

API Design

Apis are defined in two files api.py, which is a RESTful api and views.py where most of the endpoints are written in api.py file

- User Authentication: Implements user signup, signin, and signout functionalities.
- Venue and Show: Manages venues, shows, orders, and background jobs.
- Views Blueprint: Contains routes for rendering templates and handling authentication.
- User Authentication: Utilizes JWT (JSON Web Tokens) with Flask-JWT-Extended for user authentication.
- Venue Filtering: Provides endpoints for filtering venues by location, price, and tags.
- Protected Routes: Includes routes that require JWT authentication to access.

Architecture and Features

- Project Organization: The project has a templates folder which has a file index.html which is accessible at '/' endpoint defined in views follows the Model-View-Controller (MVC) architecture. All Vuejs uses cdn version, and it's code is written in .js files located in static folder vue router is defined in main.js. Database is situated in instance folder. This app is run by starting main.py file. Celery is instantiated in workers.py and it's tasks are written in tasks.py alongwith celerybeat tasks. There is single file for styling named style.css.
- Features: Implemented features include user registration and login, task creation, editing, and deletion, sending email to user on monthly and daily basis.

Video

View demo video