Ticket Show V2 Web App Report

Author:

- **Name:** Sachin Kumar
- **Roll No:** 21f2000143
- **Student Email ID:** 21f2000143@ds.study.iitm.ac.in
- **LinkedIn Profile:** https://www.linkedin.com/in/sachin-kumar-9375211a0
- **Github Profile:** https://github.com/SentiSachin
- **About:** Hey there! this is Sachin, a young web technologies enthusiast. I am also an undergrad student at IIT Madras BS Degree (Programming and Data Science) || Completed my bachelor's in Computer Science and Engineering. Interested in Development, AI, Data Science and Programming.

Apart from these, I am young energetic guy who never settles without discovering anything new each day. I like traveling, listening music, visiting new places.

Description

The Ticket Show Web App is a web-based application designed to display ticket information to users also **creation**, **deletion**, and **deletion** of venues, show by the User. The purpose of this report is to provide an overview of the app, including its features, functionality, and potential areas for improvement.

Frameworks used in the project

- ***Flask***:- for backend of the application
- ***Vue2***:- for UI in the frontend of the application
- ***Flask-Restful***:- for API creation and CRUD operations in Venues, Show, Users tables in the backend of the application
- ***Flask-Security-too***:- for Security features implementation using session with RBAC logic.
- ***Celery***:- For backend asynchronous tasks.
- ***Flask-SSE***:- For server side event alert and publishing messages.
- ***Redis***:- For message broker.
- ***GmailAPI***:- for sending email to users.
- ***Google chat Webhooks***:- for sending messages in google chat.
- ***Bootstrap5***:- for styling and responsiveness.

Tools and Technologies

These are tools and technologies to develop Ticket Show Web App. These include:

- ***Bootstrap***:- for styling and aesthetics of the application
- ***Git***:- Used local git repo for version control tool.

- ***requests***:- Requests is a popular Python library used for making HTTP requests to APIs, websites, and other web services.
- ***os***:- for some operation related with files directory in the application
- ***session***:- Flask extension supports Server-side session to our application.
- ***redirect***:-used to redirect a user to another endpoint using a specified URL and assign a specified status code.
- ***request***:- used to handle HTTP requests and responses.
- ***csv***:- python csv module for generating csv formatted report for venues.
- ***render_template***:- used to render html templates based on the Jinja2 engine that is found in the application's templates folder for html formatted emails. information provided by the user.
- ***Flask-sqlalchemy***:- used to create database schema and tables using SQLAlchemy with Flask by providing defaults and helpers.

Database Schema

1. ***Relations:*** There are five tables in the database namely Venue, Show, User, Role, Image_Collection, TopThree, Ticket and two association tables one of show and venue that is Venue_Shows and one for user and tickets that is RolesUsers.

Architecture and Features:

The project code is organised based on its utility in different files. I have named my project Simple-vue-flask-integration. Inside this folder there are four folder including application, db_directory, static, templates and files main.py, .gitignore, requirements.txt and some other files for help in some other tasks. Images in static folder, templates in template forlder.

APIs(classes) that is being used in this application

- api.add_resource(venueApi, '/api/get/venue', '/api/get/venue/int:venue_id')
- api.add_resource(specificvenueApi,'/api/location/string:venue_location')
- api.add_resource(specificshowApi,'/api/get/one/show/int:show_id')
- api.add_resource(searchApi,'/api/search/string:search_words')
- api.add_resource(showApi, '/api/get/show', '/api/get/show/int:show_id')
- api.add_resource(userApi, '/api/get/user', '/api/get/user/int:id')
- api.add_resource(topthreeapi, '/api/get/top3/movies')
- api.add_resource(topgenresapi, '/api/get/top/genres')

Routers used in this application

- @app.route('/user/user_booking/string:user_id', methods=['POST','GET'])
- @app.route('/user/book/int:show_id/int:venue_id', methods=['POST','GET'])
- @app.route('/user/theatre/int:vid')
- @app.route('/user/info')

A short demo video link is here Video link .