Ticket show project report

Author

Mohak Khatri
21f1000865
21f1000865@ds.study.iitm.ac.in

I am a pre-final year B.Tech. (computer science and engineering) student and currently at diploma level in the BS programme.

Description

This project is about developing and understanding a web based application using flask as the backend framework. We also have to implement RESTful APIs for CRUD functionalities.

Technologies used

Recommended technologies used: flask, jinja2, bootstrap, SQLAlchemy
Additional technologies used:

- javascript : bootstrap javascript and traditional client side javascript for basic DOM manipulation
- Icons: bootstrap icons using CDN links for aesthetics

No additional flask libraries used.

DB Schema Design

There are 6 main tables in this application's DB schema:

- Admin: admin_name(not null), admin_email(primary key), admin_password(not null)
- Venue: venue_name(not null), venue_location(not null), venue_max_no_of_shows(not null), venue_max_seats_per_show(not null), venue_num_shows_added(default=0), venue_id(primary key)
- Tag: tag_id(primary key), tag_name(unique)
- Show: show_id(primary key), show_name(not null), show_rating, show_price_per_ticket, show_num_ratings(default=0), show_total_rating(default=0)
- User: user_email(primary key), user_password(not null), user_name(not null)
- Ticket: ticket_id(primary key), user_email(foreign key, not null), ticket_total_price, ticket_num_tickets, show_id(foreign key), venue_id(foreign key)

Relationships and association tables:

- Venue and shows: many to many,association table- show_in_venue(venue_id(foreign key), show_id(foreign key), no_of_tickets_booked, show_housefull_flag)

- Show and tag: many to many, association table- tag_of_shows(tag_id(foreign key), show_id(foreign key))
- User and ticket: one to many

No relationship (in sqlalchemy orm) is there between ticket and show/venue(because what is to be done to corresponding ticket in case of show/venue delete/edit wasn't specified) in the sqlalchemy class however there's the foreign key constraint. Also, for the above mentioned problem, the controller would delete all tickets corresponding to the edited show(if venues are edited) or deleted venue/show.

API Design

The api implements CREATE and READ on venue and the api also implements READ on shows. The api checks the admin_email(passed in the url) as authentication and both the endpoints(for venue and show) receive arguments from the request body and not the url. In case of any errors, it returns 404 code and to handle the error, the validation.py file has a respective class.

Architecture and Features

Excluding the virtual environment folder, the project has an application folder(python files), a database_files folder, docs folder, static and templates folders and the main.py file.
The applications folder contains the backend python files(config.py, controllers.py, database.py, models.py, validation.py) separated by individual functionalities, as explained in a screencast by the professors.

All the default features are implemented. Most of the features are implemented by crud functionalities in the controller. The logins are not as secure as they are implemented using global variables. The admin_summary feature is implemented using bootstrap progress bars and although it is not a proper graph, it solves the purpose. Similarly some of the frontend functionalities are implemented using bootstrap.

Video

https://drive.google.com/file/d/1PGvR35yVv2qSbTX-OM1yxIBWYYqeAddf/view?usp=share_link