MyTicket

AN MAD-1 PROJECT REPORT

22f1001259 | MAD-1 Ticket App Project | April 5, 2023

DB Schema

The database schema consists of five tables: "user", "shows", "venues", "shows_in_venues", and "ticket_booked".

- "user" table has columns for user_id, email, password, cpassword, first_name, last_name, username, and isadmin.
- "shows" table has columns for show_id, show_name, ticket_price, premiere_date, end_date, rating, tags, show_description, cast, and poster_filename.
- "venues" table has columns for venue_id, venue_name, capacity, street, city, state, and venue_tags.
- "shows_in_venues" table has columns for show_id and venue_id, which are foreign keys that reference the "shows" and "venues" tables, respectively.
- "ticket_booked" table has columns for booking_id, username, showname, venuename, totalticket, and reservation_date. The "username" column is a foreign key that references the "user" table, and the "showname" and "venuename" columns are foreign keys that reference the "shows" and "venues" tables, respectively.

The entity relationships are as follows:

- One user can make many bookings (one-to-many relationship between "user" and "ticket_booked" tables).
- One show can be performed at many venues (one-to-many relationship between "shows" and "shows_in_venues" tables).
- One venue can host many shows (one-to-many relationship between "venues" and "shows_in_venues" tables).
- Many users can make bookings for one show at one venue (many-to-one relationship between "user" and "ticket_booked" tables, and between "shows_in_venues" and "ticket_booked" tables).

API's Design

The API's has been designed and stored for users and admin separately in `controllers.py` and `admin_controllers.py`. The summary of the API's is given below:

1. controllers.py
2. GET / : This endpoint serves the homepage of the application, displaying all available shows. It uses the `index()` function in the `controllers` blueprint.
3. GET /book_tickets/<show_id> : This endpoint displays the booking form for a specific show identified by `show_id`. It uses the `book_tickets()` function in the `controllers` blueprint.
4. GET/POST /booknow/<show_id> : This endpoint handles the booking form submission for a specific show identified by `show_id`. It accepts a `POST` request containing the booking details and creates a new `TicketsBooked` record in the database. If the booking is successful, it displays a success message. It uses the `booknow()` function in the `controllers` blueprint.
5. GET /user_profile : This endpoint displays the profile information of the currently logged-in user along with their booking history. It uses the `get_user_profile()` function in the `controllers` blueprint.
6. GET /api/shows : This endpoint serves the list of all shows in the application in JSON format. It uses the `ShowsAPI` resource class in the `ShowsApi` module.

2. admin_controllers.py

1. `adminhome()GET /` - Renders the admin dashboard view. Route name: `admin_controllers.adminhome`
2. `venue_mgmt()` - Renders the venue management view. Route name: `admin_controllers.venue_mgmt`
3. `create_venue()POST /` - Renders the form to create a new venue and creates a new venue on submission. Route name: `admin_controllers.create_venue`
4. `edit_venue(venue_id) POST /` - Renders the form to edit avenue and updates the venue on submission. Route name: `admin_controllers.edit_venue`
5. `delete_venue(venue_id) POST`/- Deletes a venue. Route name: `admin_controllers.delete_venue`
6. `show_mgmt()GET/` - Renders the show management view. Route name: `admin_controllers.show_mgmt`
7. `add_show()POST/` - Renders the form to create a new show and creates a new show on submission. Route name: `admin_controllers.add_show`

Architecture and Features

Description of each file in the `/app` directory of the `myTicket App` is :

- `main.py`: This is the entry point of the application.

- `__init__.py`: This file is used to define the Flask application instance and to configure its behaviour.
- `auth.py`: This file contains the routes and functions related to user authentication, such as login and registration.
- `controllers.py`: This file contains the controllers, it gives the app its all functionalities.
- `models.py`: This file contains the SQLAlchemy database models for the application, including the User, Tickets_booked and venues.
- `static/`: This directory contains static assets such as CSS and event poster files used by the application.
- `templates/`: This directory contains JINJA2 templates used to render the various pages of the application.

Other resources:

Video link :
https://drive.google.com/file/d/1tct4jt_AKVUVJCMHmCbtNtJY_OhR3_Y1/view?usp=share_link