Ticket Show

Problem definition

Modern Application Development - I

Last Updated: 22-12-16

Frameworks to be used

- Flask for application code
- Jinja2 templates + Bootstrap for HTML generation and styling
- SQLite for data storage
- All demos should be possible on a standalone platform like replit.com and should not require setting up new servers for database and frontend management

Ticket Show

- It is a multi-user app (one required admin and other users)
- Used for booking show tickets
- User can book many tickets for many movies
- Admin can create venues and shows
- Each venue will have
  - ID
  - Name
  - Place
  - Capacity etc.
- Each show will have
  - ID
  - Name
  - Rating
  - Tags
  - TicketPrice etc.
- Every venue can run a number of shows
- System will automatically show the latest added shows

Terminology

- Ticket Booking Platform
- Venue - List of shows, Capacity etc.
- Show - Name, Rating, Price etc.
- Dynamic Pricing (optional) - Show prices can go up/down depending upon the popularity

Wireframe: Ticketshow

Similar Products in the Market:

1. BookMyShow
    - Web, IOS and Android
2. TicketNew
    - Web, IOS and Android

- These are meant for exploring the idea and inspiration
- Don’t copy, get inspired

Core Functionality

- This will be graded
- Base requirements:
  - Admin login and User login
  - Venue Management
  - Show Management
  - Booking show tickets
  - Search for shows/venues

Core - Admin and User Login

- Form for username and password for user
- Separate form for admin login
- You can either use a proper login framework, or just use a simple HTML form with username and password - we are not concerned with how secure the login or the app is
- Suitable model for user

Core - Venue Management (Only for Admin)

- Create a new venue
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit a venue
  - Change title/caption or image
- Remove a venue
  - With a confirmation from the admin

Core - Show management (Only for Admin)

- Create a new show
  - Storage should handle multiple languages - usually UTF-8 encoding is sufficient for this
- Edit a show
  - Change title/caption or image
- Remove a show
  - With a confirmation from the admin
- Allocate venues while creating shows
- Different Pricing for each venue (optional)

Core - Search for Shows/Venues

- Ability to search theaters based on location preference
- Ability to search movies based on tags, rating etc.
- Basic home view for a venue

Core - Book Show Tickets

- Show the shows available for a given timeframe to the users
- Ability to book multiple tickets for a show at a given venue
- Ability to stop taking bookings in case of a houseful.

Recommended (graded)

- APIs for interaction with venues and shows
  - CRUD on venues
  - CRUD on shows
  - Additional APIs for getting the venues/shows to display
- Validation
  - All form inputs fields - text, numbers, dates etc. with suitable messages
  - Backend validation before storing / selecting from database

Optional

- Styling and Aesthetics
- Proper login system
- Export venue/show engagement (number of tickets booked, venue performance)
- Predict popularity of a show/venue based on the previous trends

Evaluation

- Report (not more than 2 pages) describing models and overall system design
  - Include as PDF inside submission folder
- All code to be submitted on portal
- A brief (2-3 minute) video explaining how you approached the problem, what you have implemented, and any extra features
  - This will be viewed during or before the viva, so should be a clear explanation of your work
- Viva: after the video explanation, you are required to give a demo of your work, and answer any questions
  - This includes making changes as requested and running the code for a live demo
  - Other questions that may be unrelated to the project itself but are relevant for the course

Instructions

- This is a live document and will be updated with more details and FAQs (possibly including suggested wireframes, but not specific implementation details) as we proceed.
- We will freeze the problem statement on or before 22nd December, beyond which any modifications to the statement will be communicated via proper announcements.
- The project has to be submitted as a single zip file.