Brief Description for Show Booking App

System Design

The purpose of the app is to allow users to book tickets for various shows. The app is divided into two parts: a user's side, from which they can book tickets for shows, and an admin side, where shows can be created and managed.

Architecture

The front-end uses HTML and CSS, while the back-end runs through a Flask script hosted on the local system. The application uses an SQL database, which is also run locally.

The front-end communicates with the Flask script through REST APIs.

When the user hits any application endpoint, the Flask code mapped to that route is run. If interaction with the database is required, it is done through the SQLAlchemy API. The required HTML template is then returned, which is displayed in the user's web browser. At this stage, any dynamic information required is passed through arguments in the Flask app and rendered on the front-end through Jinja2.

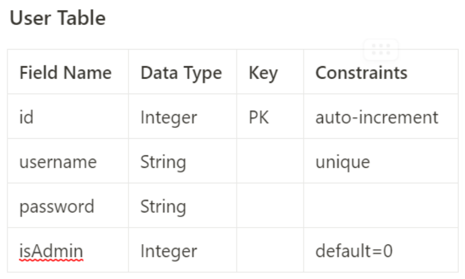
The app also uses the Bootstrap API to quickly and beautifully style the HTML and CSS.

Security

The application uses Flask-Login library to provide user authentication to both the user and admin. If the user hits any of the app's endpoints without logging in, those pages are restricted and not displayed to the user.

Models

The Flask app has four main database models: User, Venue, Show, and Booking

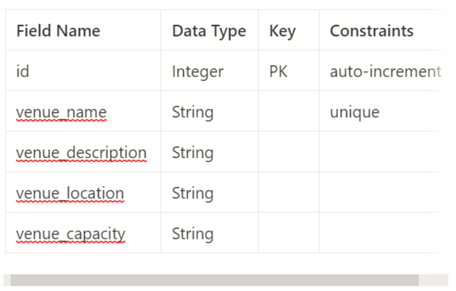


User Table

Field Name	Data Type	Key	Constraints
id	Integer	PK	auto-increment
username	String		unique
password	String		
isAdmin	Integer		default=0

The user table is primarily used to store users' authentication information. It stores both the users' and admins' information. The password is stored as an encrypted string for security reasons. Without logging in, the user is restricted from viewing and editing other pages of the app.

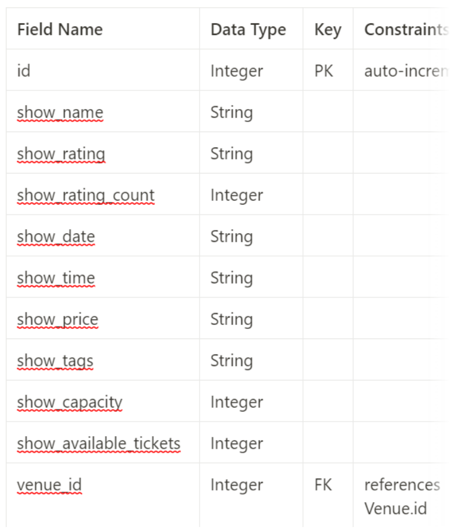
Venue Table



Field Name	Data Type	Key	Constraints
id	Integer	PK	auto-increment
venue_name	String		unique
venue_description	String		
venue_location	String		
venue_capacity	String		

The venue table is used by multiple parts of the app: to allow the admin to create and edit venues, to show independent venue pages, and to display related shows to the user. Each booking created by the user also has a reference to the venue id and show id, as described further in the document.

Show Table

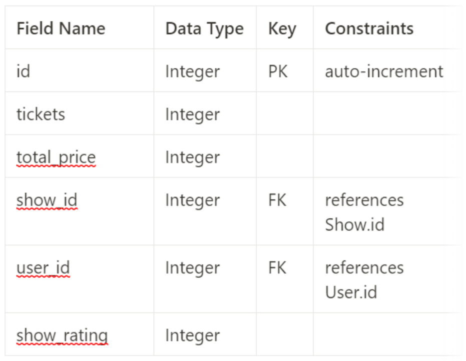


Field Name	Data Type	Key	Constraints
id	Integer	PK	auto-incre
show_name	String		
show_rating	String		
show_rating_count	Integer		
show_date	String		
show_time	String		
show_price	String		
show_tags	String		
show_capacity	Integer		
show_available_tickets	Integer		
venue_id	Integer	FK	references Venue.id

The `show` table stores information about each show. When a user makes a booking, the `show_available_tickets` field is updated. Information from this table is also used to generate graphs and statistics for the admin, which are displayed on the summary page accessible to the admin. The tags that the admin provides for the show are searchable from the search page, along with the show name. The venue_id field is a foreign key that makes it easier to fetch the venue information associated with the `show_id`.

The `show_rating` field is updated every time a user provides a rating for the show after creating a booking.

Booking Table



Field Name	Data Type	Key	Constraints
id	Integer	PK	auto-increment
tickets	Integer		
total_price	Integer		
show_id	Integer	FK	references Show.id
user_id	Integer	FK	references User.id
show_rating	Integer		

An entry in the booking table is created when a user books a show from either their home page or the venue page. The show ID and user ID are foreign keys that are used to fetch the associated show, and the user ID is used to keep track of which user has booked the show. This way, only the shows that the user has booked are shown on the booking history page.