## Author

Yukti Sethi
21f2001272
21f2001272@ds.study.iitm.ac.in
I belong to New Delhi and along with this degree, I am pursuing BTech in Computer Science Engineering from Indraprastha University. I aspire to become a data scientist.

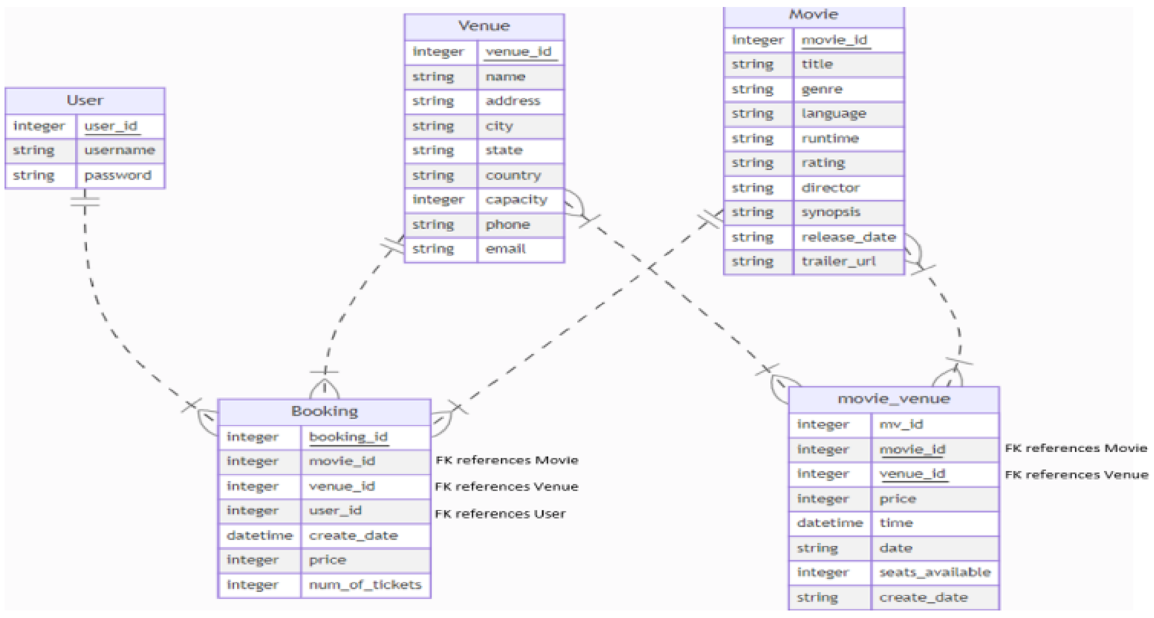
## Description

The Ticket Show Booking Application is an online single page application platform that allows the users to book movie tickets in venues. An admin is responsible for adding, deleting, and updating the movies and venues along with export, import and scheduling jobs.

## Technologies used

Python - A lightweight programming language for backend development. Flask - micro web framework to build the web application. SQLAlchemy - object relational mapper in order to interact with databases. SQLite - A lightweight relational database that stores data in the local server. Flask RestFul extension - for restful APIs to transfer JSON data. flask_cors - To make cross-origin requests to APIs. Matplotlib - Provides an easy way to generate charts and plots. Json – for sending and receiving data. HTML, Javascript, Bootstrap for CSS- used for building the user interface. Vuejs2 CLI- For client-side javascript development framework. Flask_jwt_extended – for implementing token-based authentication of users and admin. flask_sse – to implement server sent events for real-time communication. Celery - distributed task queue management system. Redis – key-value database used as cache, message broker and backend. Flask_caching – to cache the output of functions for performance.

## DB Schema Design



All the fields are not null. There is a many-to-many relationship between Movie and Venue However, with respect to the frontend, it is a one-to-many relationship (an admin can only add a movie inside a particular venue). A user can book multiple tickets of a movie in a venue and thus, there is a one-to -many relationship between User and Booking. Booking has a many-to-one relationship with Venue, Movie and User.

## API Design

I have created CRUD APIs for Venue and Movie and have used RESTful APIs throughout for every end point in my application which return JSON responses. I used swagger to document my APIs and the same interface to send cross-origin requests to my application.

## Architecture and Features

The project folder contains 3 main folders - BACKEND, FRONTEND and docs. The backend →application folder - controllers -  apis.py for RESTFUL apis, controllers.py- html responses, webhooks.py- to implement webhooks. data folder -  that database.py, models.py and the cached functions. jobs folder- tasks.py -celery tasks and workers.py -celery workers. utils - validation.py - to validate requests. templates folder,config.json, go.mod, requirements.txt, restapp.py-main flask app, ticketshow.db - database. Frontend contains : build, config -containing javascript files for configuring. src -> assets, components -.vue file components, router : index.js, App.vue - root vue component and main.js. The static folder contains image icons and manifest.json for PWA. .babelrc, .editorconfig, .gitignore, .postcssrc.js, package-lock.json, package.json, index.html - the root html file for frontend, readme.md provided by vue pwa setup. docs folder: yaml file, project report, Readme.md.
Features implemented : All the core and recommended requirements have been satisfied.

- ☑ Sign up and Login form - common for both user and admin, model for users.
- ☑ Flask Jwt authentication and Role based access control, add an admin on new db creation
- ☑ Venue management (CRUD on venues) (admin)
- ☑ Movie management (CRUD on movies) (admin)
- ☑ Different pricing for each venue, allocate venues while creating shows.
- ☑ Search (user): movies based on title | venues based on location
- ☑ Basic home view for venue , show the movies available for a given timeframe.
- ☑ Ability to book multiple tickets and stop when houseful.
- ☑ Daily reminder email job for users who did not book in the last 24 hours.
- ☑ Monthly report sending job with an option to either get it in html or pdf.
- ☑ Export csv job for admin to download a csv containing data about venues.
- ☑ API performance and caching along with handling cache expiry.
- ☑ Well designed PDF reports, Single responsive UI, add to desktop feature.
- ☑ Predict popularity of a venue based on previous trends.
- ☑ Validation - Frontend on html forms as well as backend in APIs

## Video

https://drive.google.com/file/d/1LaXYT815kl0yK3YmfH5E7Ydne0_30teV/view?usp=sharing