HOUSEHOLD SERVICES APPLICATION

“BrightWorks”

Modern Application Development – I (September, 2024 Term)

STUDENT DETAILS

NAME : Aritri Sarkar

ROLL : 23f2004529

EMAIL-ID : 23f2004529@ds.study.iitm.ac.in

ABOUT ME :

Hi! I’m someone who loves solving problems and diving into challenges, especially when it involves numbers or logic. I am totally new to the field of web application development but thanks to the Modern Application Development course, it helped me pick up skills in Python, SQL, CSS and a little bit of JavaScript. Right now, I’m working on a fun project for a Household Services Application as part of my coursework. It’s been exciting to explore tools like Flask, Jinja2, Bootstrap, and SQLite while building something meaningful and widely applicable in the real world.

ABOUT THE PROJECT

BrightWorks is a multi-user application that acts as a platform to connect customers and service professionals for comprehensive home servicing solutions. The application involves three roles: Admin, Service Professionals, and Customers. The goal was to create a functional, user-friendly, and robust platform for service management and requests.

MY APPROACH

The project was built using Flask as the backend framework to handle the application's logic and routing, ensuring a robust foundation for the multi-user system. Jinja2 templates were used in conjunction with Bootstrap to dynamically generate HTML pages with responsive and aesthetically pleasing designs. For data storage, SQLite was chosen due to its lightweight nature and ease of integration with Flask.

The application was designed around clear, role-based functionalities as mentioned in the instruction document and the development process included the following steps:

- The project began with designing and implementing the data models using SQLite. Tables were created to store information for users, services, and service requests, ensuring proper relationships between entities like customers, service professionals, and admins.
- Once the database models were in place, routes were built using Flask to handle user requests and manage interactions between the frontend and backend. These routes were designed to support role-based functionalities.
- After setting up the backend, Jinja2 templates were developed for the user interface. Bootstrap was used for styling, ensuring a responsive and visually appealing design. The templates were dynamically generated to reflect user-specific content and functionality.

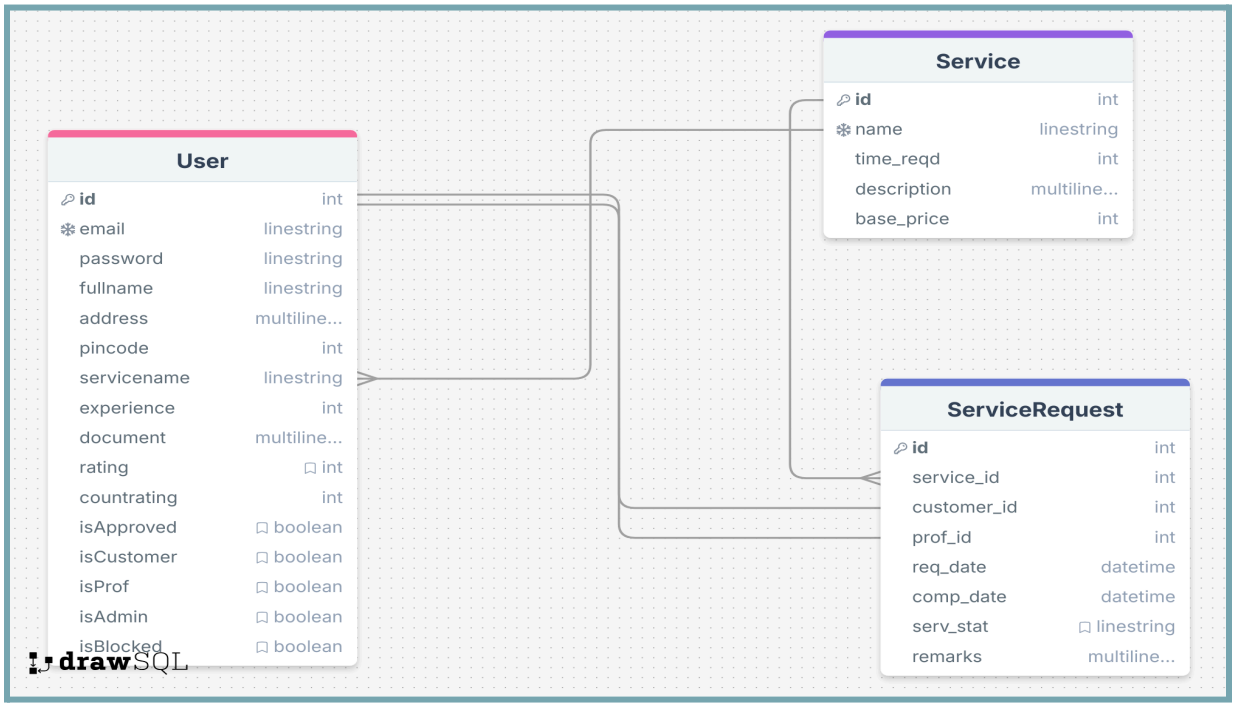
FRAMEWORKS AND LIBRARIES USED

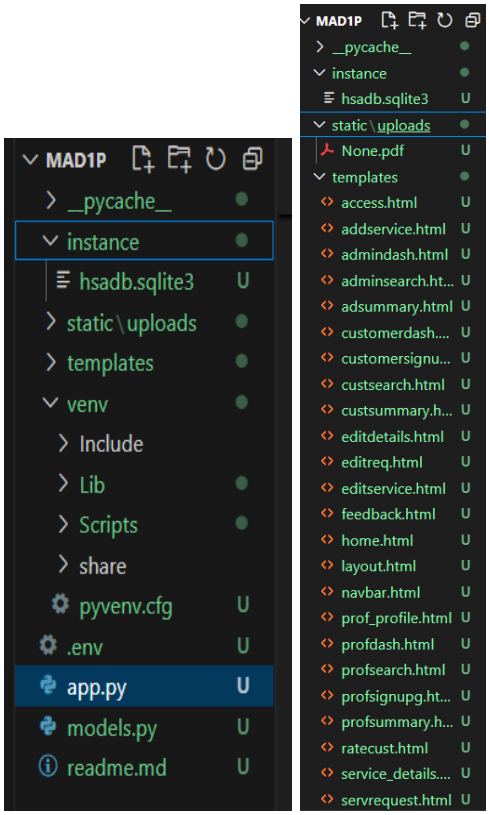
- **Flask** : Backend framework for building the web application
- **SQLalchemy** : ORM (Object-Relational Mapping) tool for database interactions.
- **SQLite** : Database management system for storing application data.
- **HTML/CSS/Bootstrap** : Frontend technologies for user interface design
- **Datetime** : Python library for handling date and time operations.
- **Jinja2** : Template engine for rendering dynamic HTML content.
- **ChartJS** : User for creating different types of charts on the admin dashboard.

LIMITATIONS

- Due to time constraints, I could not implement API resources.
- A simple login system was used without advanced authentication mechanisms like Flask-Login.

ER DIAGRAM





File arrangement inside the root folder

PROJECT VIDEO URL
:https://drive.google.com/file/d/1Q8ui0YS08JsYbsqdjlOHpgSRkj0Fai5X/view?usp=sharing