Project Report
Modern Application Development I
Household Services Application

Name - Rituparno Dhar
Roll no. - 21f1000601

Description:

Household Service App is a platform connecting customers, service professionals, and administrators to manage household services seamlessly. Customers can explore and request services like AC servicing or plumbing, providing details such as preferred schedules and remarks. Each service has attributes like ID, name, price, and time required.

The application offers tailored logins for each user type. Admins access a dashboard to manage users, verify professional profiles, create or update services, and address issues like fraud by blocking the users and professionals. Customers can search for services by their type or name and manage their requests, including updates and closures. Service professionals can view, accept, or reject requests.

How did I approach the problem statement?

I used **Flask** as the web framework to handle routing, requests, and database interactions. For the database, I employed **SQLAlchemy ORM** to define models for key entities like **Customer**, **Professional**, **Service**, and **ServiceRequest**, enabling smooth communication with the database.

To render dynamic content, I utilized **Jinja2** templating, which allowed me to inject data from Flask routes (such as customer information and service requests) directly into the HTML templates. I also styled the application using **CSS**, linking static files served by Flask to create an attractive user interface.

For user input, I implemented **HTML forms** that collect data, such as customer details or service requests, which are then processed by Flask routes and stored in the database.

In essence, Flask managed the backend, **SQLAlchemy** handled database operations, **Jinja2** render dynamic content, and **CSS** gave the application its visual style, resulting in a seamless full-stack solution.

Frameworks and Libraries used:

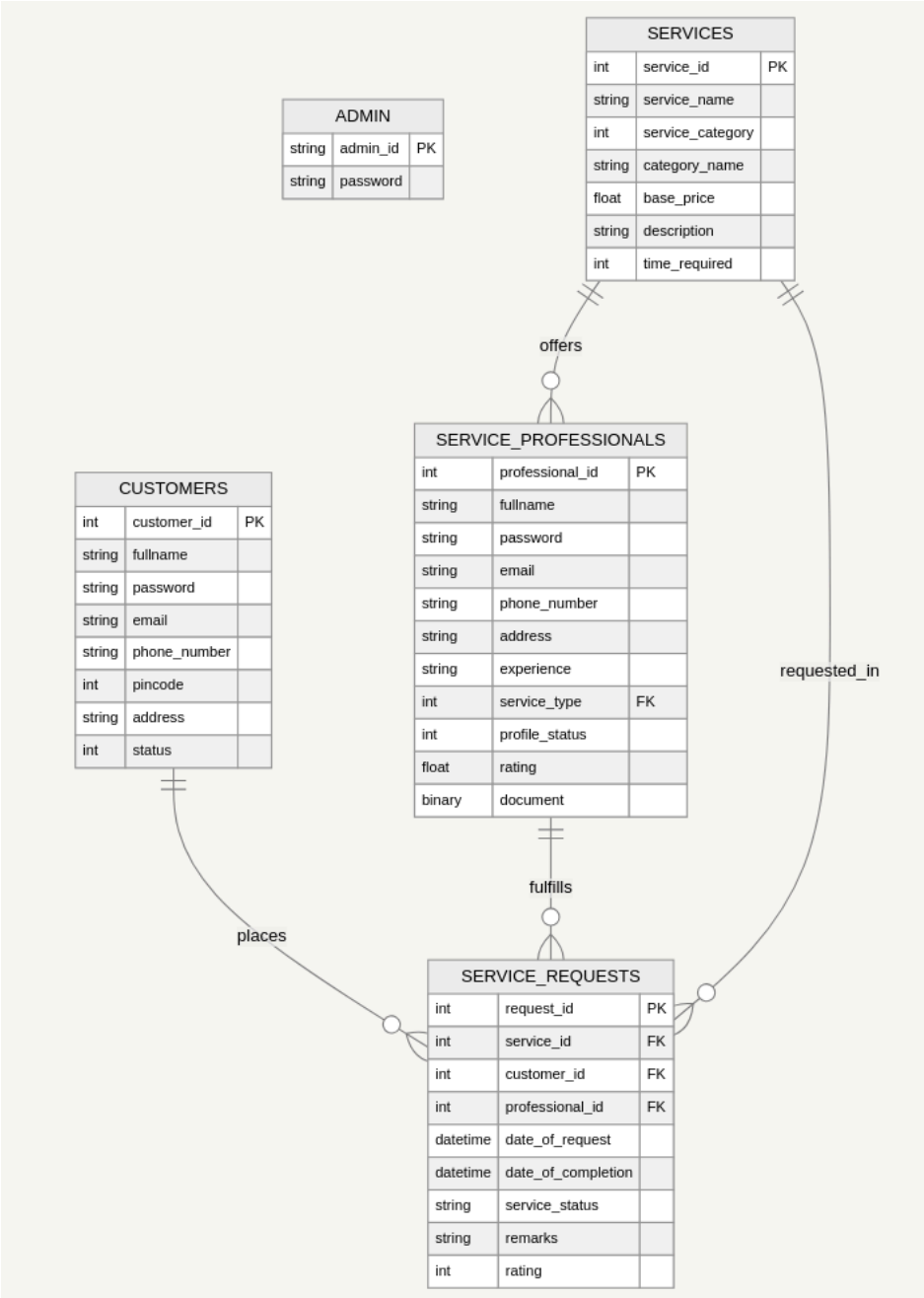
The code provided uses the following technologies:

1. **Flask**: A lightweight Python web framework used to build web applications. It defines the app, sets configurations, handles routes, and renders HTML templates.
2. **SQLAlchemy**: An Object Relational Mapper (ORM) for Python that interacts with relational databases (like SQLite) using Python objects instead of raw SQL queries.
3. **SQLite**: A lightweight, file-based relational database management system used to store data locally in a .sqlite3 file.

4. **HTML Templates (Jinja2)**: Flask uses the Jinja2 templating engine to render dynamic HTML content via the render_template() function.
5. **CSS**: Used to style the web application, defining layout, colors, fonts, and design for a visually appealing user interface, linked through static files in Flask.
6. **Flash Messages**: The flash() function stores messages to be displayed to the user on the next page request, typically for notifications or alerts.

The app is essentially a Flask-based web application for a "Household Service" system, with database integration, user sessions, and templating.

Database Schem:



Presentation Video Link -

2024-12-03 23-17-23.mkv