Modern Application Development 2-Project

Household Services Application - V2

Student Details

Name: Irina Agastin
Student ID: 23f2004169
Email: 23f2004169@ds.study.iitm.ac.in

Project Details

Title: Household Services Application - V2
Objective: To develop a multi-user platform that provides comprehensive home servicing and solutions, facilitating easy interactions between customers, service professionals, and admins.

Problem Statement:

The problem this application seeks to solve is to improve the management and delivery of household services by enabling easy service request creation, professional selection, and request monitoring. The app also focuses on automating repetitive tasks like sending reminders, generating reports, and facilitating actions by professionals to ensure efficient service delivery. Additionally, it will allow admins to manage users and services efficiently, ensuring only verified and trusted professionals are available for services.

Approach:

1. **Admin Roles:**
   Implementing a role-based access control (RBAC) system allows the Admin to manage all approving service professionals, manage services, and monitor user activity. Providing the ability for admins to create, update, and delete services, while also allowing professionals to accept or reject requests based on their availability and expertise.

2. **Customer Interaction:**
   Enabling customers to easily search for services, create and manage requests, and post reviews once a service is completed. They can also track the status of their requests.

3. **Professional Actions:**
   Allowing service professionals to view, accept, and close service requests while also providing them with an option to review the request after completion.

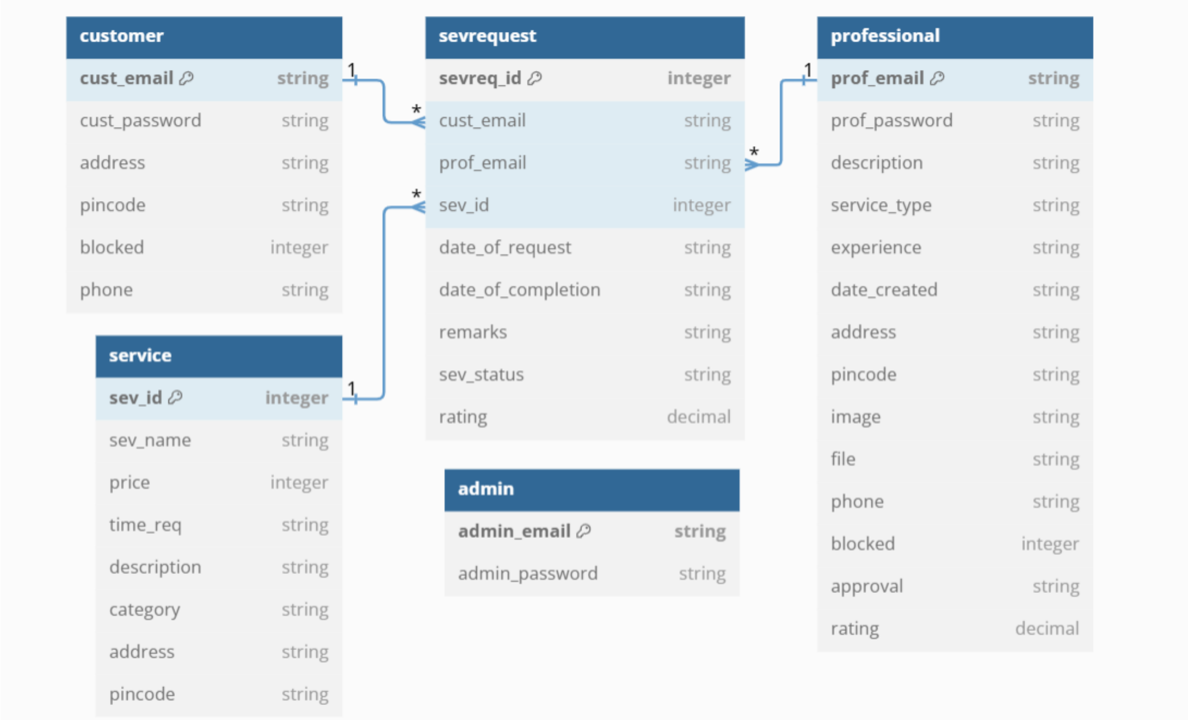
4. **Backend Jobs:**
   Implementing scheduled jobs for daily reminders to service professionals and generating monthly customer activity reports. Asynchronous jobs for exporting a professional's service request data in CSV format for the Admin are also included.

5. **Performance and Caching:**
   Using Redis for caching to improve the performance of frequently accessed data and reduce API load. Implementing cache expiry mechanisms to ensure data freshness.

Frameworks and Libraries Used

- **Flask:** For the application backend and routing.
- **VueJS:** For user interface and dynamic content rendering. (Vue CLI via npm and integrated Vite)
- **SQLite:** Managing the database, including tables for users, services, and service requests.
- **Redis:** Caching for faster access and improved performance**.**
- **Celery:** Background job processing and asynchronous tasks
- **Celery Beats:** Task scheduling
- **Bootstrap and css:** Creating responsive front-end layouts and styling the application.
- **Jinja2:** For rendering HTML templates (single page application)
- **ChartJS:** For generating graphical reports and statistics.
- **JWT (JSON Web Tokens):** Token-based and role-based authentication
- **Service-Sent Events (SSE):** Real-time alerts and event updates to users
- **Mailhog:** Testing mail functionality

ER Diagram



customer	
cust_email	**string**
cust_password	string
address	string
pincode	string
blocked	integer
phone	string

service	
sev_id	**integer**
sev_name	string
price	integer
time_req	string
description	string
category	string
address	string
pincode	string

sevrequest	
sevreq_id	**integer**
cust_email	string
prof_email	string
sev_id	integer
date_of_request	string
date_of_completion	string
remarks	string
sev_status	string
rating	decimal

admin	
admin_email	**string**
admin_password	string

professional	
prof_email	**string**
prof_password	string
description	string
service_type	string
experience	string
date_created	string
address	string
pincode	string
image	string
file	string
phone	string
blocked	integer
approval	string
rating	decimal

Presentation

The following link provides access to a brief video presentation explaining the project's implementation and features

https://drive.google.com/file/d/1Ug5dNAdhFcqXdLoGtnnd6B9MaVFBaeOw/view?usp=drive_link

Conclusion:

This project provides an efficient platform for managing home service requests and connecting customers, professionals, and admins. By leveraging Flask, VueJS, SQLite, Celery (for task management), and Redis (for caching) the application ensures a seamless user experience and robust data management.