Household Services App(V2)

*Servzy***

Project Report by Sudipta (22f3000840) Modern Application Development-II, January 2025 Term

Student Details

Name: Sudipta Das

Roll No- 22f3000840   Email: 22f3000840@ds.study.iitm.ac.in

About me - - Coding is where I find my creative flow. I love the chance to work on complex problems and find elegant solutions. Well, curiosity lands me learning a new programming language or framework. From designing intuitive websites to exploring the depths of machine learning, I am passionate about pushing the boundaries of technology. But when I am not glued to the screen, you will probably find me on the Cricket pitch or hitting the gym. Project Description It's a platform which serves as a dynamic bridge between service professionals and customers, facilitating seamless collaborations for mutual benefit. Customers can effortlessly find the right service professionals to get their work done for their products or services. Service Professionals, in turn, increase their reach and expand their business. This synergy not only maximizes exposure and engagement for customers but also empowers service professionals to grow and thrive in their respective services.

Project Description-- It's a platform which serves as a dynamic bridge between service professionals and customers, facilitating seamless collaborations for mutual benefit. Customers can effortlessly find the right service professionals to get their work done for their products or services. Service Professionals, in turn, increase their reach and expand their business. This synergy not only maximizes exposure and engagement for customers but also empowers service professionals to grow and thrive in their respective services.

How I approached the problem statement-- I began by breaking down the problem into core components: user management, service functionality, admin controls, and background jobs, creating the architecture of the app on a paper. After the architecture I moved on to the database schema design. Prioritized building a robust database schema, carefully designing relationships between service professionals, customers, services, service requests to ensure scalability and maintainability. On 20th January, I initialized the project on WSL. Implemented authentication and authorization using JWT tokens, establishing a secure foundation for role-based access control (RBAC) between admin and user functionalities. Spent 4 hours on a silly mistake in the implementation of JWT, finally GitHub comments thread helped. Then developed the backend API layer using Flask-RESTful, focusing on creating clean, modular endpoints that follow RESTful principles and include proper error handling and validation. After I had coded a decent amount of backend and I moved to the frontend part. I created a responsive frontend using Vue.js, emphasizing user experience with intuitive navigation, real-time feedback, and a consistent design language across both student and admin interfaces. First designed the interface in Figma and then started coding it out. My Vue.js skills weren't strong initially, so followed a course too. Then finally in the end, I addressed the elephant in the room, "Backend Jobs". Integrated Celery with Redis for handling asynchronous tasks, implementing scheduled jobs for daily reminders and monthly reports, ensuring efficient background processing. Spent a good amount of time here and on the export file as CSV functionality. Finally made some minor UI changes, code refactoring and I was done with the project in the span of 20 days and approximately 150 hours. Ultimately, my proactive approach paid off, and I completed the project well ahead of the second deadline. This experience underscored the importance of perseverance and adaptability in overcoming challenges and achieving project goals.

Technologies Used

Flask: A lightweight backend framework for building web applications with Python.

SQL Alchemy: ORM (Object-Relational Mapping) tool for database interactions.

SQLite: Database management system for storing application data.

Vue.js: A progressive JavaScript framework for building user interfaces and enhancing interactivity.

Flask JWT Extended: An extension that provides tools for managing user sessions and authentication using JSON Web Tokens (JWT).

Flask Restful: A Python library that simplifies the creation of RESTful APIs for web applications.

Celery: An asynchronous task queue that enables the execution of background jobs and scheduled tasks.

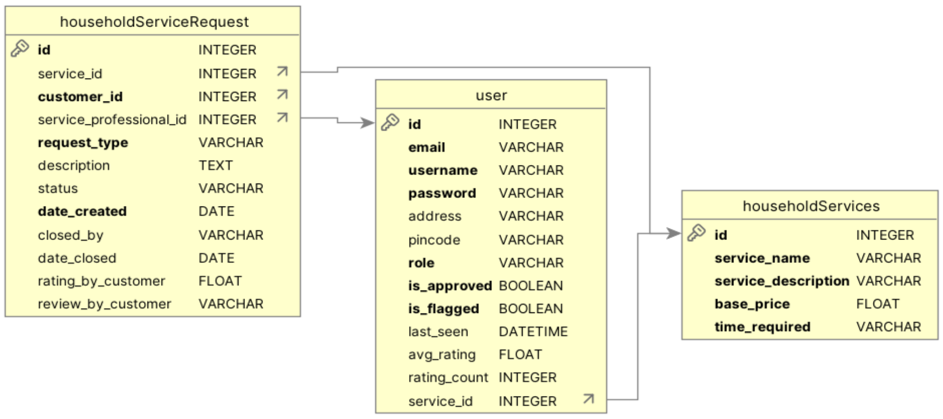
Redis: An in-memory data structure store used as a caching database to optimize application performance.

Werkzeug: Utility for securely managing passwords and authentication

Mailhog: MailHog is an Open Source email testing tool with a fake SMTP server underneath.

DB Schema Design

The database schema encompasses tables for user, householdServices, householdServiceRequest. These entities are interconnected to track various user activities within the system. Each table contains essential fields such as user details, household services data and all requests data. Relationships are established to manage interactions like service requests, service description, and customer ratings and reviews, ensuring robust data management across the platform. This structure supports comprehensive tracking and management of activities related to campaigns, influencers, and user engagements within the application



API Endpoints

AuthApi---'/api/login','/api/sp','/api/sp/<int:sp_id>'

SignupApiSp----'/api/signupsp'

SignupApiCu----'/api/signupcu'

ServiceApi----'/api/services','/api/services/<int:service_id>'

AdminSearchAPI---- "/api/admin_dashboard/search”

BlockUserAPI---- '/api/admin_dashboard/block_user/<int:user_id>'

ExportDataAPI----'/api/admin_dashboard/export_data/<int:sp_id>'

ServiceRequestAPI---- '/api/requests'

CustomerDashboardAPI---- '/api/customer_dashboard'

ServiceDetailsAPI---- '/api/service_details/<int:service_id>

FetchServiceRequestAPI----
'/api/customer_dashboard/service_request/<int:service_request_id>'

CreateServiceRequestAPI----
'/api/customer_dashboard/create_service_request/<int:service_id>'

EditServiceRequestAPI----
'/api/customer_dashboard/edit_service_request/<int:service_request_id>'

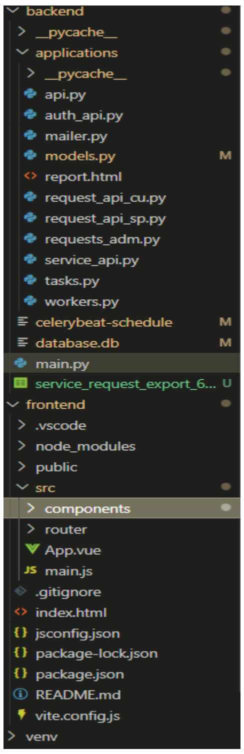
CancelServiceRequestAPI----
'/api/customer_dashboard/cancel_service_request/<int:service_request_id>'

CloseServiceRequestAPI---- '/api/customer_dashboard/close_request/<int:service_request_id>'

CustomerSearchAPI---- '/api/customer_dashboard/search'

ServiceHistoryAPI---- '/api/customer_dashboard/service_history'

RequestApiSp----'/api/requests_sp','/api/requests_sp/<int:service_request_id>'



```
backend
  __pycache__
  applications
    __pycache__
    api.py
    auth_api.py
    mailer.py
    models.py                      M
    report.html
    request_api_cu.py
    request_api_sp.py
    requests_adm.py
    service_api.py
    tasks.py
    workers.py
  celerybeat-schedule              M
  database.db                      M
  main.py
  service_request_export_6...      U
frontend
  .vscode
  node_modules
  public
  src
    components
    router
    App.vue
    main.js
  .gitignore
  index.html
  jsconfig.json
  package-lock.json
  package.json
  README.md
  vite.config.js
venv
```

Video Presentation link—

https://drive.google.com/file/d/1fM-Ex9ENMFZn3neUxFSYsA22-oHE_jVC/view?usp=sharing