Author

Sajal Saxena
21F1003503
21f1003503@ds.study.iitm.ac.in
I am a recently passed out engineering graduate and am interested in Machine Learning.

Description

The Household Services project statement requires us to make an easy-to-use application. This app is supposed to connect Service Professionals, who excel at a household service like plumbing or carpentry, with users who may need to hire them for short-term basis to provide these services. The student is required to make an end-to-end application, with a working frontend using VueJS and backend using Flask and storing the data in an SQLite database managed using Flask-SQLAlchemy.

Technologies used

Backend: The code for backend was written in **Python**. The data was stored in an **SQLite** database which was managed using **Flask-SQLAlchemy**. Passwords stored in the database were secured by **Werkzeug Security**. **Flask-Security** was used for Role Based Access Control (RBAC) and user authentication. **Flask-Mail** was sued to send daily reminders or monthly reminders via backend jobs that were implemented using **Celery**. **Redis** was the message broker and result backend for Celery tasks. **Crontab** was used for scheduling the periodic tasks. User triggered async jobs and periodic reminders were executed on **Mailhog** and **Google Chat Webhooks**. **Flask-Cache** was used for caching.

Frontend: **VueJS** was used as the primary frontend framework to create dynamic and responsive webpages. It provided client-side navigation.

Flask RESTful was used to implement APIs to connect the frontend with the backend.

DB Schema Design

The model has 7 tables: User, Role, UserRoles, Service, Service_Request, ServiceRequestStatus and Complaints. User table included all the user data like their name, emails, phone numbers and other specific role related data like address in case of a user registered as a customer or verification status and work experience for a user registered as a service professional. Role table had roles related data like name and id of the roles. UserRoles was an association table between User and Roles. UserMixin and RoleMixin were used with User and Role models respectively.

Service table contains data related to all the services. Service Request table had data related to the request for a service made by a customer to a service professional. Service

Request Status table stores the service requests which have been sent to all eligible service professionals for a particular service and stores whether they accept or reject the service request. Complaints table stores any complaints that a customer may have regarding the work performed by the service professional for a particular service request.
ER diagram link:
https://drive.google.com/file/d/193J3si3sjfeCyJ59lNR4P8VU3Z0qoV07/view?usp=sharing

API Design

APIs were implemented using Flask-Restful. API endpoints were made for CRUD operations (get, post, put and delete) in User table (Registration of user, login, changing their verification status or flag), Service table, Service Request table, Service Request Status table and Complaints table. Endpoints were made to change status of various attributes at different points of time, for eg. Changing status of service from REQUESTED to ASSIGNED when a service professional accepts it, and to CLOSED when the customer closes it.

Architecture and Features

The app.py file lies in the root folder of the project. Along with it is a reuirements.txt file and celery_config.py file for celery configurations. There are 3 more folders within the root folder: Application, Static and Templates. All the backend files including controllers (in routes.py), api routes (resources.py), database (model.py) and other configuration and utility files for backend jobs are located inside Application folder.
Static contains another folder named components and within this lies all the VueJS files for frontend. The Vue files are the various components which are all called from script.js inside the Static folder. Templates contain html files for rendering the webpages on the browser and two templates for mailing as part of scheduled jobs via Celery.

Core feature of the application is the ability to create services and service requests for those services. Services can be created of any one of the following categories: Cleaning, Plumbing, Electrical, Appliance Repair, Salon and Carpentry.
When the customer creates a service request, that request is sent to all the service professionals who specialise in that service. The service professional can only accept a new service request if they do not have an active service request. Once a service professional accepts the request, the request becomes unavailable for all other professionals of the same service. The customer is responsible for closing the service request. While closing this request, they can also register a complaint against the professional for sub-standard work quality. The admin can flag off a professional based on this complaint.

Video

https://drive.google.com/file/d/1Bl9JUYZlrHgi4DBbD0Rsncor7Cbzg97T/view?usp=share_link