Project Report

Personal Details

Name: Agrim Srivastava

Roll No.: 23f3002782

Email: 23f3002782@ds.study.iitm.ac.in

Project Details

Question Statement

A multi-user web app connecting customers with service professionals for seamless home servicing, featuring admin management, service booking, and reviews.

Approach

The project was developed using a structured, two-phase approach. It began with backend development, where a RESTful API was created using Flask to handle authentication, user roles, and service management. Once the backend was fully functional, the focus shifted to the frontend, built with Vue.js, to provide a user-friendly interface for customers, service professionals, and the admin. This modular approach ensured a clear separation of concerns and streamlined integration.

Frameworks and Libraries

The frameworks and libraries used are:

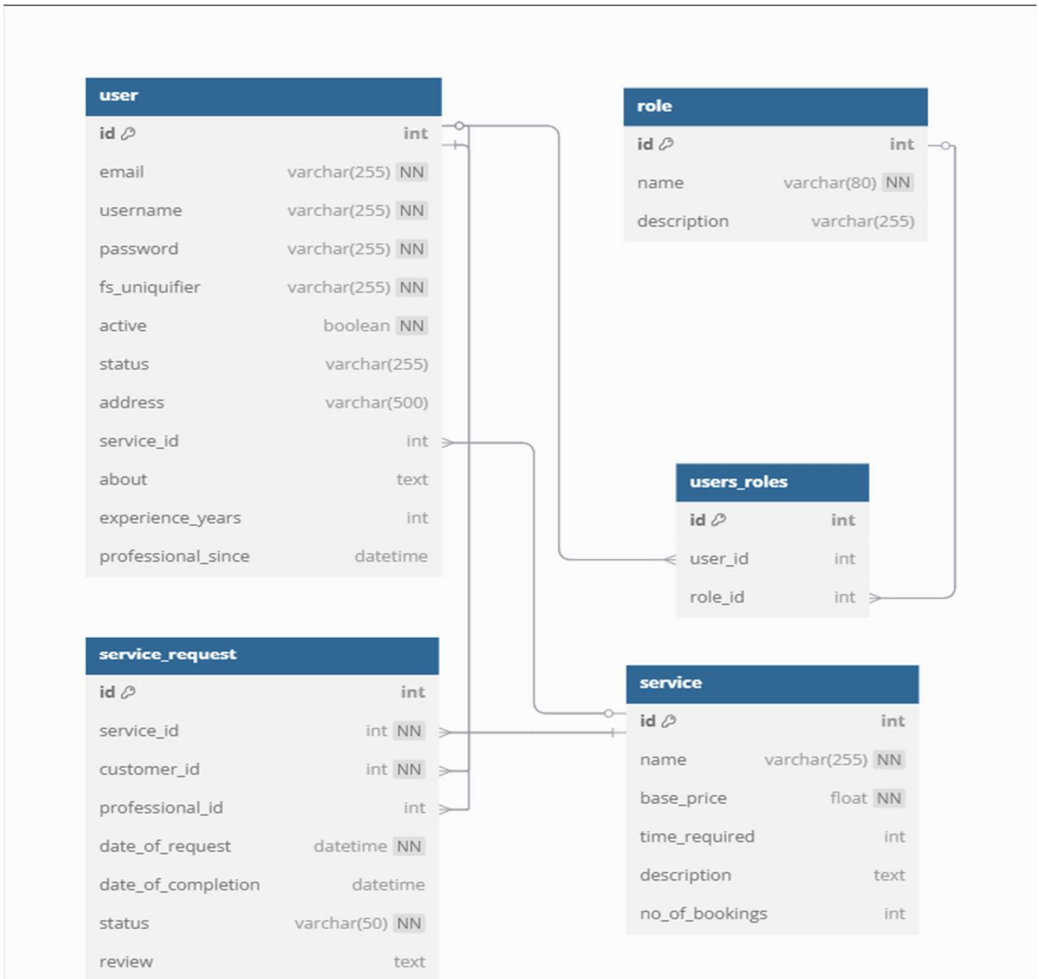
- **Flask**: A lightweight web framework serving as the backbone of the application.
- **Flask-SQLAlchemy**: Provides an ORM layer for seamless database interactions.
- **Flask-Security-Too**: Handles authentication and security for user management.
- **Flask-RESTful**: Facilitates the creation of RESTful APIs for backend communication.
- **Celery**: Manages asynchronous tasks such as background job processing.
- **Redis**: Used as a message broker and caching solution in conjunction with Celery.
- **Vue.js**: A progressive frontend framework used to build a responsive and dynamic user interface.

API Design

The API endpoints cover key resources including:

- **Users**: For registration, authentication, and profile management.
- **Roles**: To assign and manage user permissions.
- **Services**: For listing and managing available home service offerings.
- **Service Requests**: To create, update, and track service bookings.

Database Design



Architecture and Features

The project is organized into two main folders: **frontend** and **backend**. The **backend** folder contains key files such as **resources.py**, which holds the REST API definitions, and **models.py**, which defines the database structure with SQLAlchemy models. The **frontend** folder is built using Vue.js and includes a **src** folder where various components, views, and assets are organized to create a responsive and dynamic user interface.

I have implemented a range of essential features to ensure the project is both secure and efficient. In the backend, I utilized Flask-Security-Too to establish robust authentication, authorization, and role-based access control (RBAC). I also set up email notifications and managed asynchronous backend jobs with Celery and Redis, and integrated Google Spaces webhooks for automated updates and interactions. On the frontend, built with Vue.js, I adopted **Pinia** for state management to maintain a seamless data flow, and employed Bootstrap to achieve a responsive design.

Video Presentation

https://drive.google.com/file/d/1H7mor_vuyxSj5d1pzKMg_TpTZMWSnBWl/view?usp=sharing