MAD 2 Project Report – Household Services

Student details

Name - Shreya Saxena
Roll number - 22f3001013
Email - 22f3001013@ds.study.iitm.ac.in

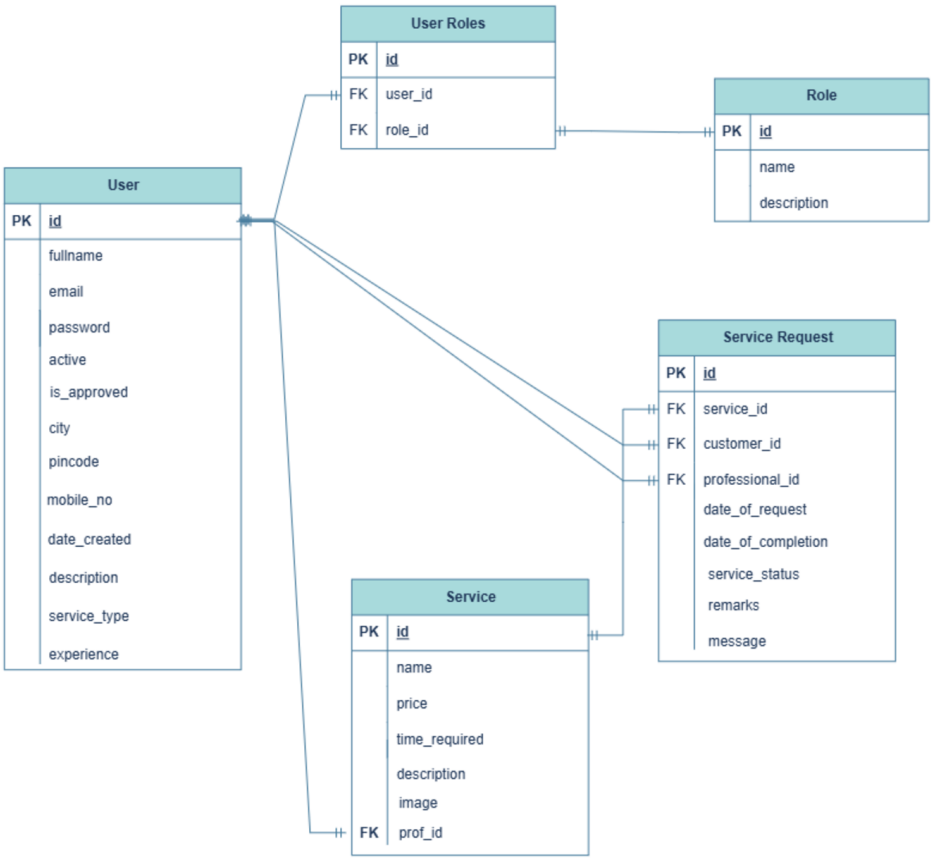
Description

It is a multi-user app (requires one admin and other service professionals/ customers) which acts as platform for providing comprehensive home servicing and solutions. A web application and an API is created for multiuser interaction which allows the service professionals to accept and reject a particular service request and the customers can create a service request or book a service. Admin of the application can perform CRUD operations on Service . Each user has their own personalised dashboards.

Frameworks and libraries used

- **Backend –** Flask-Restful (for APIs), Flask-Security-too (for security) , Flask-Mail (for sendin mails), Flask-SQLAlchemy & SQLite (for Database storage) , Redis (Caching) , Celery (Backend Jobs and Scheduling)
- **Frontend –** Vuejs (for the frontend), Vue3 [CLI], Jinja2 (for styling html templates).

Database Schema



• **User table**: id is PRIMARY KEY, email, password, fullname and many other attributes
• **Role table**: id is PRIMARY KEY, name and description of the role. In the application there are three roles : “Admin”, “Customer” and “Service Professional”
• **User Roles table** : id is PRIMARY , user_id FOREIGN KEY to user table and role_id FOREIGN KEY to role table. This is an association table between User and Role table.
• **Service table**: id is PRIMARY KEY with name, price, time_required, description as attributes and the prof_id is FOREIGN KEY to User table
• **Service Request table**: : id is PRIMARY KEY with service_id, customer_id, professional_id as FOREIGN KEYS to service table and user table respectively.

Approach Used

In this application I have used RBAC (Role based access control), I have 3 roles in my application namely Admin, Customer and Service Professional. Also I have used token based authentication for the security purposes of the application.

Features and Functionalities

● UserAuthentication and Role-Based Access: ○ Built using Flask-Security and SQLAlchemy, users are directed to role-specific dashboards upon login. ○ Different dashboards for each role include functionalities tailored to their respective needs.
● Service Request System: ○ Customers can book services, and Service Professionals can accept or reject requests. ○ Admins can manage available services and user accounts.
● DynamicSignup Fields: ○ The signup page includes role-specific fields, dynamically displayed based on the user’s selected role.

API Design of the application

There are 3 resources created for API using the Resource class from Flask-RESTful. They can handle GET/POST/PUT/DELETE or CRUD requests and their response is in JSON format.

- AuthApi – Resource for CRUD on User Model
- ServiceAPI – Resource for CRUD on Service Model
- ServiceRequestAPI – Resource for CRUD on Service Request Model
- CeleryRoutesAPI – Resource for the celery backend jobs

Architecture of the application

The root folder consists of the project report and the folder names code. Inside the code there are three folders backend which has all the backend code, frontend folder which contains the vue files and lastly the instance folder which has database and the app.py file through which we run our application.

Presentation video

https://drive.google.com/file/d/14j0vTHSv7-xOJJ3BeXB2WNxiNjJKv23x/view?usp=drivesdk