Author
Name: Balivada Sri Charitha
Roll number: 21f2000491
Email: 21f2000491@ds.study.iitm.ac.in
About: I am a recent graduate with a B.Tech degree in Electronics and communications Engineering, currently dedicated to pursuing a BS degree in Data Science and Applications at IITM. Fueled by youthful enthusiasm and curiosity. I am committed to expanding my knowledge and skills in these domains.

Description
The application is a Household Services platform designed to connect customers with service professionals for various home services. The platform has three user roles: Admin, Service Professional, and Customer. The Admin has full control, including managing services, approving or blocking service professionals, and monitoring users' activities. Service Professionals can register, manage their services, accept or reject requests, and complete tasks. Customers can search for available services, create service requests, and provide feedback on completed services. Additionally, the platform integrates notifications for service reminders, monthly reports for the admin, and a CSV export feature for service requests. The application uses technologies such as Flask for the backend, VueJS for the frontend, SQLite for data storage, Redis for caching, and Celery for handling background tasks.

Technologies used

1. **Python** (Used for creating the backend)
    a. Flask (Used to create the web application)
    b. Flask-SQLAlchemy (Used for interacting with the SQLite database) ○ Celery (Used for scheduling tasks)
    c. Flask-Security (Used for authentication and RBAC)
    .and related dependencies for the above packages to function
2. **Redis** (Used for caching and storing async result)
3. **SQLite** (Used for storing data)
4. **VueJS** (Used for the front end)
5. **CSS3** (Used for custom styling)
6. **Bootstrap** (Used for basic styling)

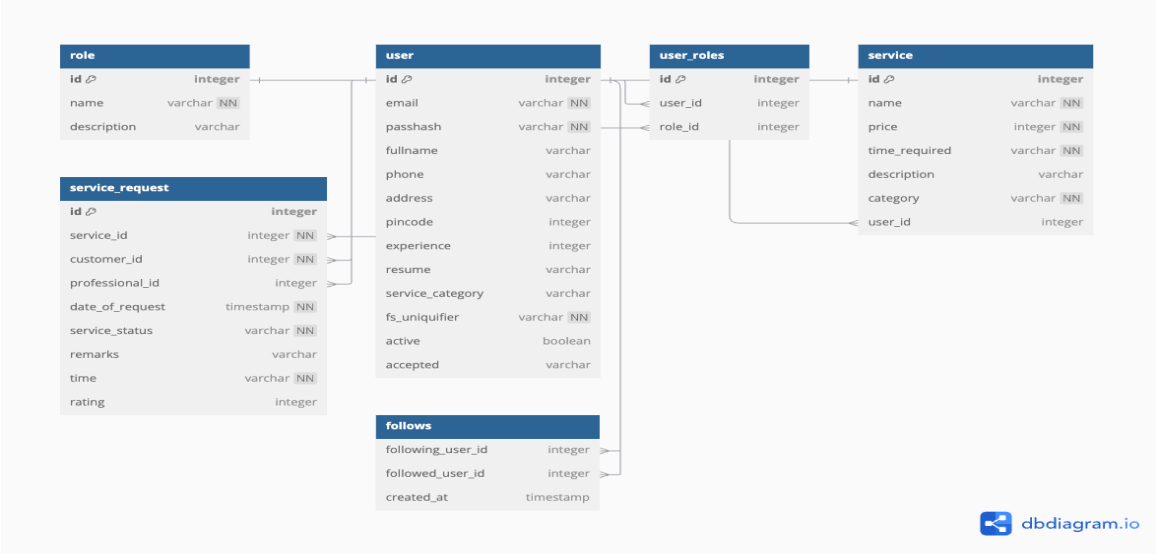
API Design

The API is designed to facilitate a Household Services platform with the following main features:

- **Auth**: The API ensures secure authentication using token-based authentication. Users (Service Professionals, Customers, and Admin) are required to authenticate using tokens to access their respective functionalities.
- **Admin (Admin/Manager)**: The Admin has access to perform CRUD operations on services, manage service requests, and approve or block service professionals. The Admin can also download service-related data in CSV format.
- **Service Professional**: Service professionals can view and manage the services they offer, and can accept or reject service requests. They can also mark a service request as completed once it’s finished.
- **Customer**: Customers can search for available services based on various filters (like category), create service requests, and track their progress. They can also leave feedback after the service is completed.

Database Schema Design

The Household Services platform's database focuses on three main entities: **User**, **Service**, and **ServiceRequest**. **User** stores information like email, phone, and roles (via **UserRoles**), with multiple roles per user. **Service** tracks services offered by professionals, including details like name, price, and category. **ServiceRequest** manages customer requests, linking users (customer and professional) to services, with attributes like status, time, and remarks. The schema supports one-to-many relationships (users to services, services to requests) and many-to-many relationships (users to roles). The design uses **Flask-Security** for secure authentication and role-based access control.



Architecture and Features

The project follows an iteration of the MVC model and follows the fundamental idea of separation of concerns. The models for each of the tables are defined in the `models` file, the files which host the API endpoints and the usage of controllers with authentication validations are stored in the `routes` file. APIs designed specifically for Services and ServiceRequests are stored in 'resources' file. All of these files that perform backend functionalities are stored in the backend folder. The frontend folder contains vue js components, pages and router files that are required for user friendly accessing and operations.
All these folders are collectively part of the `Code` folder. The SQLite database is stored in the folder called `db`.

Features implemented in the application are:

- Proper login and signup page for users, and admin using RBAC
- Validation and authenticated access using token-based authentication provided by Flask Security package
- CRUD on services and service requests.
- Requesting, accepting and closing of service requests by customers and professionals accordingly.
- Searching for services using various parameters like service name.
- Displaying all the services and their details and also professional details and their details to the customer.

Video Link: Click Here