# Household Services Application - V2
# MAD 2 Project Report

Prakhar Singh
Roll No: 21f3001320
Email: 21f3001320@study.iitm.ac.in

I am a final-year B.Tech student in Electronics and Communication Engineering at Bundelkhand Institute of Engineering and Technology, Jhansi. With a Diploma in Information Technology, I am passionate about Data Science, Drone Technology, Coding, and Electronics, and I enjoy exploring innovative and cutting-edge technologies.

**Description :**

The "Household Services Application - V2" is a multi-user platform connecting customers with service professionals for comprehensive home solutions. Built using SQLite, Flask, and VueJS, it ensures efficient data management, responsive APIs, and an intuitive user interface, facilitating seamless service bookings and administration**.**

**Technologies used :**

1. Flask-SQLAlchemy- to create and manage the relational database for the app
2. Flask security for token-based authentication.
3. Bootstrap is for templates of web pages.
4. SQLite3 will be used to create the app's database structure.
5. Flask- this web application is built on flask.
6. Redis for the caching.
7. Celery worker/beat for task management and scheduling.
8. MailHog for checking and sending mail services

**Database Schema Design:**

The database schema supports a service management app with role-based user management, service categorization, task tracking, and historical logging, ensuring scalability, efficiency, and easy tracking of service requests and statuses. A detailed description of the model is listed below.

1. Role Table :
   ( **id**, name, description) Stores roles for users like admin, customer, or professional.
2. User Table :
   (**id**, name, email, password, fs_uniquifier, active, roles) Represents all users in the application with their basic information.
3. Service :
   ( **id**, name, price, time_required, description, category_id) Represents individual services provided under specific categories.

4. Service Category :
   ( **id**, name, description, tags) Defines categories for services to organize available offerings.
5. Service Request :
   (**id**, service_id, customer_id, professional_id, date_of_request, date_of_completion, service_status, remarks) Manages requests for services by customers and tracks their status
6. User Roles :
   ( **id**, user_id, role_id) Manages many-to-many relationships between users and roles.
7. History :
   ( **id**, customer_id, professional_id, date_of_completion, customer_name, pro_name, service_name, category_name) Logs completed service details for customers and professionals.
8. Professional :
   ( **id**, is_verified, category_id, experience, rating) Extends the User table for professionals with specific attributes.
9. Customer :
   ( **id**, address, phone) Extends the User table for customers with additional details.

**API Design:**

In the project, I have used the Flask-restful module to implement API for CRUD on Login, Category, Services, User Management, Customer Management etc.

Service Management:

- GET `/services` Retrieves a list of services filtered by query or category ID.
- POST `/services` Adds a new service (requires admin privileges).
- PUT `/services/<int:id>` Updates the details of a specific service (requires admin privileges).
- DELETE `/services/<int:id>` Deletes a specific service (requires admin privileges).

Service Request Management:

- GET `/service_requests` Retrieves service requests for the authenticated user.
- POST `/service_requests` Creates a new service request (only customers can create requests).
- PATCH `/service_requests/<int:id>`Updates the status of a service request (accept, close, etc.).
- PUT `/service_requests/<int:id>`Accepts a service request (only professionals can accept requests).

History Management:

- GET `/history` Retrieves service history for the authenticated customer or professional

**Architecture and Features** :

Customer Features:

☑ Search For a Service by name or category
☑ Book a Available Service on the application
☑ Check for the History
☑ Check whether the Professional Accepted his/her request or not and the name of the professional

Admin Feature :

☑ Create a Service Category
☑ Full control on the Professionals verify/Unverify
☑ Create Edit Delete a Service
☑ Ban/Unban Customers based on fraudulent activity
☑ Export the Available Category Data in CSV
☑ Create a new service with a base price

Professional Feature :

☑ Accept the Service requests
☑ Register only for one service Category(as per problem statement) and serve only for that
☑ Only visit the service requests according to their category

Additional Features :

☑ Sending Reminder to Application User
☑ User can Export History as CSV

Admin-mail = admin@study.iitm.ac.in
Password = pass

**VideoURL:**
**https://drive.google.com/file/d/13QtOSdxaI3mggiBKJvs_96dxLbm4tdgS/view?usp=sharing**