**Household Services Application - V2 Report**

**Name: Hariom Pandey  
College: IITM  
Diploma Level Student  
Roll No: 21F3002112**

**Project Overview**

**The Household Services Application is a multi-user platform designed to provide comprehensive home servicing solutions. It includes three main roles: Admin, Service Professionals, and Customers. The application is built using the following technologies:**

* **Backend: Flask for API development, Flask\_restfull**
* **Frontend: VueJS for UI**
* **Database: SQLite for data storage**
* **Caching: Redis for caching**
* **Background Jobs: Redis and Celery for batch jobs**
* **Styling: Bootstrap for HTML generation and styling**

**Core Functionalities**

1. **Admin Dashboard:**
   * **Admin login redirects to the dashboard.**
   * **Manage users (customers/service professionals).**
   * **Approve service professionals after verification.**
   * **Block users based on fraudulent activity or poor reviews.**
2. **Service Management:**
   * **Create, update, and delete services.**
3. **Service Request Management:**
   * **Customers can create, edit, and close service requests.**
4. **Search Functionality:**
   * **Customers can search for services by name or location.**
   * **Admin can search for professionals to manage them.**
5. **Service Request Actions:**
   * **Service professionals can view, accept, or reject service requests.**
6. **Backend Jobs:**
   * **Daily reminders for service professionals.**
   * **Monthly activity reports sent via email.**
   * **User-triggered CSV export of service requests.**

**Difficulties Faced**

1. **Role-Based Access Control (RBAC):**
   * **Implementing secure authentication and authorization for different user roles was challenging, especially ensuring that the admin has full access while restricting service professionals and customers.**
2. **Database Relationships:**
   * **Designing the database schema to effectively manage relationships between users, services, and service requests required careful planning to avoid redundancy and ensure data integrity.**
3. **Asynchronous Tasks:**
   * **Setting up Celery with Redis for background jobs was complex, particularly in ensuring that tasks were executed correctly and that results were handled properly.**
4. **Frontend Integration:**
   * **Integrating VueJS with Flask APIs required a good understanding of both frameworks, especially in managing state and handling asynchronous requests.**
5. **Error Handling:**
   * **Implementing robust error handling and user feedback mechanisms in both the frontend and backend to enhance user experience.**

**📚 ER Diagram for Household Service Application**

**1. User (Table)**

* **id (PK)**
* **email (Unique, Not Null)**
* **password (Not Null)**
* **fs\_uniquifier (Unique, Not Null)**
* **active (Boolean, Not Null)**
* **full\_name (Not Null)**
* **address**
* **pincode**
* **experience (Default -1)**
* **service\_type (FK -> Service.service\_type)**
* **service\_name (Default 'None')**
* **description**
* **date\_of\_creation**
* **user\_rating (Default 0)**
* **is\_verified (Default False)**
* **is\_blocked (Default False)**

**Relationships:**

* **Many-to-Many with Role through UserRoles**
* **One-to-Many with ServiceRequest as customer\_service\_requests**
* **One-to-Many with ServiceRequest as professional\_service\_requests**

**2. Role (Table)**

* **id (PK)**
* **name (Unique, Not Null)**
* **description**

**Relationships:**

* **Many-to-Many with User through UserRoles**

**3. UserRoles (Association Table)**

* **id (PK)**
* **user\_id (FK -> User.id)**
* **role\_id (FK -> Role.id)**

**4. Service (Table)**

* **id (PK)**
* **service\_type (Unique, Not Null)**
* **base\_price (Default 1000)**
* **description**
* **time\_required**

**Relationships:**

* **One-to-Many with ServiceRequest as service\_requests**

**5. ServiceRequest (Table)**

* **id (PK)**
* **service\_id (FK -> Service.id)**
* **service\_type (Stores string for history)**
* **customer\_id (FK -> User.id)**
* **professional\_id (FK -> User.id)**
* **status (Default 'Requested')**
* **date\_of\_completion**
* **date\_of\_request (Default today’s date)**
* **remarks**
* **service\_rating**
* **customer\_rating**

**API Routes for Household Services Application**

**User Authentication**

**1. User Login**

* **Endpoint:** **/api/login**
* **Method:** **POST**
* **Description:** Authenticates a user and returns a token along with user details.

**2. User Registration**

* **Endpoint:** **/api/register**
* **Method:** **POST**
* **Description:** Registers a new user (customer or service professional) and returns a success message.

**3. User Info**

* **Endpoint:** **/userinfo**
* **Method:** **GET**
* **Description:** Retrieves the authenticated user's information.

**4. User Role**

* **Endpoint:** **/api/user-role**
* **Method:** **GET**
* **Description:** Returns the role of the logged-in user (admin, service professional, or customer).

**5. User Logout**

* **Endpoint:** **/logout**
* **Method:** **POST**
* **Description:** Logs out the authenticated user.

**Service Management**

**6. Get Professionals for Service**

* **Endpoint:** **/api/service/<int:service\_id>/professionals**
* **Method:** **GET**
* **Description:** Retrieves a list of professionals available for a specific service.

**7. Book Service**

* **Endpoint:** **/api/book-service**
* **Method:** **POST**
* **Description:** Allows a customer to book a service by providing service and professional IDs.

**Service Request Management**

**8. Get Service History**

* **Endpoint:** **/api/customer/service-history**
* **Method:** **GET**
* **Description:** Retrieves the service history for the authenticated customer.

**9. Get Ongoing Services**

* **Endpoint:** **/api/customer/ongoing-services**
* **Method:** **GET**
* **Description:** Retrieves ongoing service requests for the authenticated customer.

**📝 Conclusion for Household Service Application**

The **Household Service Application** is designed to provide a seamless platform that connects customers with skilled professionals for various household services. The application ensures:

✅ **Efficient Service Management:**  
Admins can create, update, delete, and manage services efficiently. Professionals are verified through document verification, ensuring trust and reliability for customers.

✅ **User Role Management:**  
Multiple roles (Admin, Customer, and Professional) with appropriate access levels ensure that users have the right permissions and can interact with the system securely.

✅ **Service Request Tracking:**  
Customers can place service requests, track their status, and rate professionals based on service quality. Professionals can manage service requests and update progress, ensuring smooth workflow management.

✅ **Admin Verification and Monitoring:**  
Admins verify users, block/unblock them if necessary, and maintain a high level of security and trust in the system.

✅ **Enhanced User Experience:**  
With features like user ratings, remarks, and service history, the application aims to improve customer satisfaction by offering personalized and quality services.

✅ **Scalable and Flexible Design:**  
The application is built with Flask, VueJS, and SQLite, making it lightweight yet powerful. Redis and Celery ensure background task handling for smooth performance.

In conclusion, the **Household Service Application** bridges the gap between customers and service professionals by providing a reliable, efficient, and secure platform, ensuring a satisfactory user experience for all stakeholders.

Thankyou.