Report: Household Services App - V2

Submitted by: Jyotiraditya Saha
Email: 21f2000759@ds.study.iitm.ac.in

Overview

The Household Services App - V2 is a multi-user platform designed to offer home servicing solutions. It includes three primary user roles: **Admin**, **Service Professionals**, and **Customers**. The app provides a comprehensive solution for managing services, professionals, and customer interactions.

Entities and Relationships

1. User

- **Attributes**:
  id, email, name, password, address, pincode, phone_number, service_type, experience, document_path, active, approved, last_login_at, current_login_at, last_login_ip, current_login_ip, login_count, fs_uniquifier.
- **Relationships**:
  - Many-to-Many with **Role** through **RolesUsers**.
  - One-to-Many with **ServiceRequest** (as customer and professional).
  - One-to-Many with **Review** (as customer and professional).

2. Role

- **Attributes**:
  - id, name.
- **Relationships**:
  - Many-to-Many with **User** through **RolesUsers**.

3. RolesUsers

- **Attributes**:
  - id, user_id, role_id.
- **Purpose**: Association table for the many-to-many relationship between **User** and **Role**.

4. Category

- **Attributes**:
  - id, name, description.
- **Relationships**:
  - One-to-Many with **Service**.

5. Service

- **Attributes**:
  - id, name, description, price, category_id.
- **Relationships**:
  - Many-to-One with **Category**.
  - One-to-Many with **ServiceRequest**.

6. ServiceRequest

- **Attributes**:
  - id, customer_id, professional_id, service_id, request_date, status, customer_review.
- **Relationships**:
  - Many-to-One with **User** (as customer and professional).
  - Many-to-One with **Service**.

7. Review

- **Attributes**:

    - id, customer_id, professional_id, rating, review_text.
- **Relationships**:
    - Many-to-One with **User** (as customer and professional).

Frameworks and Libraries

1. **Flask**: Lightweight WSGI framework used for routing, requests, and responses.
2. **Flask-Security**: Manages user authentication, role-based access, and security features.
3. **Flask-Restful**: Extends Flask to build RESTful APIs.
4. **Flask-CORS**: Enables Cross-Origin Resource Sharing.
5. **Flask-SQLAlchemy**: Integrates SQLAlchemy for ORM functionality.
6. **Celery**: Asynchronous task queue for background job management.
7. **Redis**: Caching Layer and backend job management.
8. **Werkzeug**: Library for secure filename handling and other WSGI utilities.
9. **Caching**: Used to optimize performance by storing frequently accessed data.
10. **SQLAlchemy**: Used for database interactions via Flask-SQLAlchemy.
11. **JSON**: Handles JSON data. Implicitly used via Flask's jsonify.

API Endpoints

1. User Management

- **POST** /signup/customer
- **POST** /signup/professional
- **POST** /signin

2. Category Management

- **GET** /all_categories
- **GET** /category
- **POST** /add_category

3. Service Management

- **GET** /services
- **POST** /services
- **DELETE** /services/<int:service_id>
- **PUT** /services/<int:service_id>
- **GET** /services/category/<int:category_id>

4. Service Requests

- **POST** /request_service
- **GET** /service_requests
- **POST** /accept_service_request
- **POST** /reject_service_request
- **POST** /complete_service_request
- **POST** /cancel_service_request

5. Professional Management

- **GET** /professionals
- **POST** /approve/professional
- **POST** /block

6. Customer Management

- **GET** /customers
- **GET** /customer/profile
- **PUT** /customer/profile

7. Service Rating and Review

- **POST** /rate_professional
- **GET** /professional/<int:professional_id>/rating

8. Search Functionality

- **POST** /search_services

9. File Management

- **GET** /static/uploads/<filename>

10. Task Scheduling

- **GET** /remind_professionals_to_complete_requests
- **POST** /trigger-csv-export

Conclusion

The **Household Services App - V2** is a robust and scalable solution that effectively bridges the gap between customers and service professionals while offering powerful management tools for admins. The app leverages a well-structured database, modern frameworks, and advanced features like background tasks and caching to ensure an optimized user experience.

Presentation URL:
https://drive.google.com/file/d/1v_s1Rv9gGi1OBgv85U3vQSUjpNcBWq13/view?usp=sharing