App Dev-2 Project Report

Student Details

- Name: Kushagra Wadhwa
- Roll No: - 21f3002930
- Email: 21f3002930@ds.study.iitm.ac.in

Project Details

- Project Title: -Household Services Management System ~ "Homeyfy"
- Problem Statement: -Develop a web-based application to streamline household services. 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.

Approach to the Problem Statement

 Requirements Analysis and Feature Prioritization:

- Clearly define the core functionalities for the three user roles (Admin, Service Professional, Customer).
- Understand the roles and actions of each actor (e.g., managing users/services for Admin, service acceptance/rejection for Professionals, service search/requests for Customers).

 Architecture Design:

- Define the backend APIs using Flask to handle user authentication (RBAC with Flask Security), service management, and service requests.
- Use SQLite for structured data storage, designing tables with clear relationships (e.g., Services, Service Requests, Users).

 Frontend Development:

- Use Vue.js components for rendering of Admin dashboards, professional profiles, and service requests.

 Caching and Async Tasks:

- Implement Redis for caching frequently accessed data (e.g., service search results) to improve API performance.
- Use Celery with Redis for batch jobs like daily reminders, monthly reports, and CSV exports.

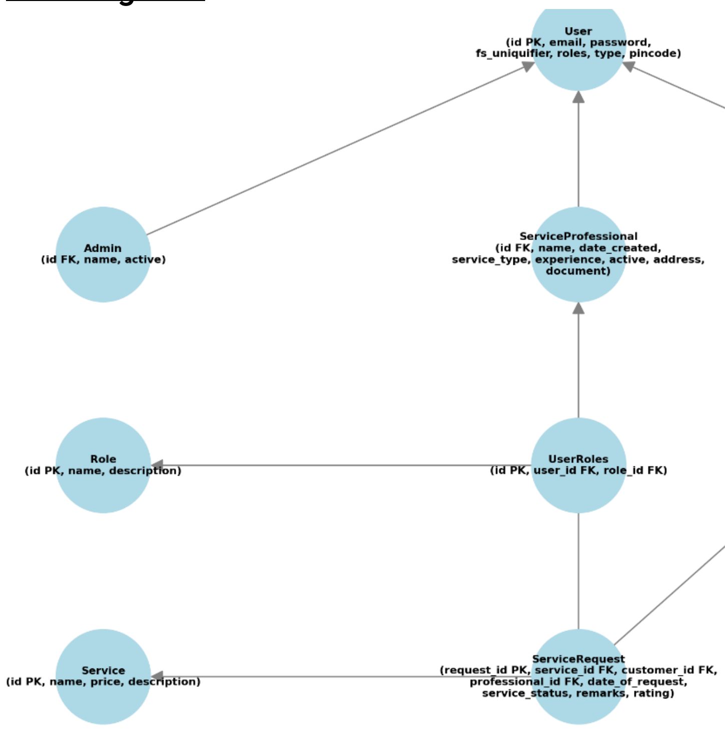
 Scheduled Jobs and Notifications:

- Develop backend tasks using Celery to send reminders and generate monthly activity reports, integrating email or Google Chat webhooks.

Frameworks and Libraries Used

- Flask: Used as the core backend framework for API development.
- SQLAlchemy: Enabled object-relational mapping for database management.
- Celery: Handled background tasks like generating activity reports and sending email notifications.
- Redis: Utilized as an in-memory data structure store and message broker.
- Flask-Mail: Managed email notifications for activities like service confirmations and reminders.
- MailHog: Captured and tested email functionalities during development.
- Flask-SQLAlchemy: Extended SQLAlchemy capabilities within Flask applications.

ER Diagram


User
(id PK, email, password,
fs_uniquifier, roles, type, pincode)
Admin
(id FK, name, active)
ServiceProfessional
(id FK, name, date_created,
service_type, experience, active, address,
document)
Customer
(id FK, name, address, active)
Role
(id PK, name, description)
UserRoles
(id PK, user_id FK, role_id FK)
Service
(id PK, name, price, description)
ServiceRequest
(request_id PK, service_id FK, customer_id FK,
professional_id FK, date_of_request,
service_status, remarks, rating)

Architecture

The architecture follows a modular design with a Flask-based backend serving as the API layer, SQLite for persistent data storage, and Vue.js for the frontend interface. Redis is integrated for caching and managing async batch jobs via Celery. Role-Based Access Control (RBAC) is implemented using Flask Security to restrict access based on user roles (Admin, Service Professional, Customer). Key endpoints include /login and /register for authentication, /admin/dashboard for managing users and services, /services for CRUD operations on services, /service-requests for creating, updating, and closing service requests. Additional support for batch operations like daily/monthly reminders and customer activity reports for generating activity summaries.

Drive Link for Presentation Video

https://drive.google.com/file/d/1B8M62IEN44I0p9CK1iU0echyviY1qu-q/view?usp=sharing

Future Work

1. Expand functionality for real-time notifications using WebSockets.
2. Integrate machine learning for matching service professionals based on past performance and customer preferences.
3. Optimize database queries for handling large-scale user operations.
4. Performance Optimization: Add enhanced caching strategies for frequently accessed data like service lists and professional availability.
5. Improved UI.
6. Integrate Payment Gateway, Customer loyalty, Discounts etc.