Modern Application Development – I

Household Services App

# Author

Name: S Guru Prasad

Roll Number: 21f2000673

Email Id: [21f2000673@ds.study.iitm.ac.in](mailto:21f2000673@ds.study.iitm.ac.in)

[Project Video link](https://drive.google.com/file/d/1dGq6sTM1TVhey5qAdzwOxPiFwLrhhQL5/view?usp=sharing)

I am a recent Computer Science graduate, having completed my studies in 2024. I thoroughly enjoyed working on this project, tackling challenges along the way. It was a fantastic journey filled with both fun and valuable learning experiences.

# Description

1. This project implements a multi-user Flask web application for home service automation, enabling interaction between Admin, Customers, and Service Professionals.
2. Admin has privileges to manage users, services, and service requests.
3. Customers can sign up, request services, review service professionals, and view their service history.
4. Service Professionals sign up to offer services, wait for admin approval, and handle service requests from customers.
5. The project focuses on streamlined service assignment and management, including request tracking, reviews, and location filtering.

# Technologies used

* **Flask:** Core framework for building the web application.
* **Flask SQLAlchemy:** ORM for database operations and schema modeling.
* **Flask Login:** Managing user authentication and session handling.
* **SQLite:** Lightweight database for data storage.
* **Flask Blueprint:** Modularizing the app for roles like Admin, Customer, and Service Professional.
* **Bootstrap:** Frontend framework for responsive UI.
* **Jinja2:** Template engine for dynamic HTML rendering.

# DB Schema Design

The database schema contains the following main entities and relationships:

1. **Admin:** Manages the platform, users, and services.
2. **Customer:** Can request services and submit reviews.
3. **Service Professional:** Offers specific services and handles requests.
4. **Service:** Defined by the admin, with price and description.
5. **Service Request:** Links customers and professionals for services.
6. **Review:** Customers can rate and comment on services by professionals.

**• Service ↔ Service Professional:** One-to-many (A service type is linked to multiple professionals).

**• Service ↔ Service Request:** One-to-many (A service can be requested multiple times).

**•** **Customer ↔ Service Request:** One-to-many (A customer can make multiple requests).

**•** **Service Professional ↔ Service Request:** One-to-many (A professional can accept multiple requests).

**•** **Service Request ↔ Review:** One-to-one (Each request has a review).

**•** **Customer ↔ Review:** One-to-many (A customer can leave multiple reviews).

**•** **Service Professional ↔ Review:** One-to-many (A professional can receive multiple reviews).

A detailed database schema diagram is available below:

# 

ER Diagram

# Architecture and Features

## Architecture:

* Modularized using Flask Blueprint to separate functionality for Admin, Customers, and Service Professionals.
* SQLite database integrates with Flask SQLAlchemy for ORM operations.
* Templating via Jinja2 ensures dynamic rendering of pages.

## Features:

### Admin:

* Add, edit, and delete services (if not assigned).
* Approve or block customers and professionals.
* View and filter service requests based on status.

### Customer:

* Sign up, login, and request services.
* Filter professionals by service type, location, and pin code.
* View service history and ongoing requests.
* Provide ratings and reviews for professionals.

### Service Professional:

* Sign up with a service offering and wait for admin approval.
* Accept or reject service requests.
* Track ongoing requests and update status.
* This project effectively integrates Flask's backend capabilities with Bootstrap for a functional and user-friendly application.