EasyHome Services Application - Project Report

Student Details

- **Name**: Manish jat
- **Roll Number**: 23f3004152
- **Course**: Modern Application Development I

Project Overview

Objective: It is a multi-user platform for managing household services where users (customers and service professionals) interact under admin supervision.

Roles:

1. **Admin**: Manage services, professionals, and customers.
2. **Service Professional**: Provides specific home services (e.g., plumbing, cleaning) and accepts/reject customer requests.
3. **Customer**:Can search and book services and professionals as per their requirements.

Development Approach

1. **Framework and Libraries**:
    - **Flask** for core application development.
    - **SQLite** as the database engine.
    - **Jinja2 Templates and CSS** for aesthetic views.
    - **SQLAlchemy** for ORM.
    - **Matplotlib** for admin-specific data visualisation.
2. **Authentication & Authorization**:
    - Admin has default credentials (`username: jatmanis1` and `password: 123`).

      For new admin run this commands **`flask create_superuser`**

    - **Session Management**: Flask's session is configured to manage login sessions.
    - **bcrypt**: Used for hashing and managing secure passwords for users.
    - **Role-based Access Decorators**: Decorators to check if a user is logged in and has the correct role to access certain pages.
3. **Features**:
    - **Admin Dashboard**:
        - Manage services, Customers and professionals.

        - View and approve customers and professionals' profiles and block users if needed.
    - **Service Request Management**:
        - Service professionals can accept or reject requests.
        - Customers can create, edit, and close requests.
    - **Customer Review System**:
        - Customers post reviews and feedback upon service completion.
    - **Reporting and Analytics**:
        - Admin can visualize data, such as service counts or performance stats, using plots.

Database Design (ER Diagram)

The code structure suggests an **Entity-Relationship (ER) Diagram** that includes:

1. **Customer**: Registers and creates service requests.
2. **ServiceProfessional**: Represents individual professionals who accept/rejects customer requests.
3. **Service**: Defines the service type, base price, and other details.
4. **ServiceRequest**: Links the customer and professional with a specific service.

Relationships:

- `ServiceRequest` table connects `Customer` and `Professional` as foreign keys.
- Admin oversees all tables without being directly represented in database tables.

Presentation Video

mad1 presentation.mp4