Project Report for Final Project

Modern Application Development - I

Author

Student Name: Swastik Garg
Roll Number: 22f3000908
Email: 22f3000908@student.onlinedegree.iitm.ac.in

I am a college student deeply interested in application development and web technologies. This project has been a valuable opportunity to hone my skills in Flask, database management, and user-centric design.

Description

The **Household Services Application** is a multi-user platform designed to connect customers with service professionals while allowing administrators to manage users and services effectively. This app utilizes modern technologies like Flask, SQLAlchemy, and Bootstrap to provide a seamless and responsive experience.

The primary features of the platform include:

1. **User Authentication**: Separate login and dashboard functionality for Admins, Customers, and Workers.
2. **Service Management**: Admins can create, update, and delete services.
3. **Request Handling**: Customers can create service requests, which professionals can accept or reject.
4. **Search Functionality**: Users can search for services by name or location.
5. **Ratings and Reviews**: Customers can review completed services, enhancing trust and transparency.

Technologies Used

1. **Flask**: Framework for backend logic and routing.
2. **Flask-SQLAlchemy**: Extension for managing database connections and ORM.
3. **Flask-Login**: User session management for login/logout functionalities.
4. **Bootstrap**: Responsive frontend framework for styling and layout.
5. **SQLite**: Lightweight database for storing user, service, and request information.
6. **Werkzeug**: Utilities for URL routing and security.

Database Schema Design

Users Table:

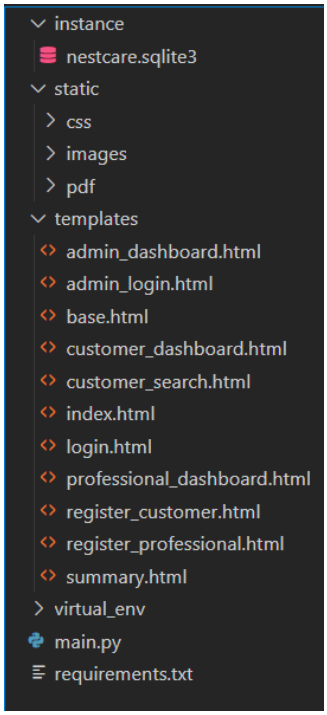
- **id**: Primary Key
- **name**: User's name (Unique)
- **email**: User's email (Unique)
- **password**: Hashed password
- **role**: User role (admin, customer, professional)
- **service_id**: Foreign Key linked to Services

Services Table:

- **id**: Primary Key
- **name**: Name of the service
- **description**: Service details
- **base_price**: Service price
- **time_required**: Time needed for service

ServiceRequest Table:

- **id**: Primary Key
- **service_id**: Foreign Key linked to Services
- **customer_id**: Foreign Key linked to Users
- **professional_id**: Foreign Key linked to Users
- **status**: Request status (e.g., requested, closed)



Controllers Used

- @hsapp.route("/", methods=["GET", "POST"])
- @hsapp.route("/nc_login", methods=["GET", "POST"])
- @hsapp.route("/login", methods=["GET", "POST"])
- @hsapp.route("/register_customer", methods=["GET", "POST"])
- @hsapp.route("/register_professional", methods=["GET", "POST"])
- @hsapp.route("/admin", methods=["GET", "POST"])
- @hsapp.route("/admin/create_service", methods=["GET", "POST"])
- @hsapp.route("/admin/edit_service/<int:service_id>", methods=["GET", "POST"])
- @hsapp.route("/admin/delete_service/<int:service_id>", methods=["GET", "POST"])
- @hsapp.route("/customer", methods=["GET", "POST"])
- @hsapp.route("/customer/create_service_request/<int:service_id>", methods=["GET", "POST"])
- @hsapp.route("/customer/edit_service_request/<int:service_request_id>", methods=["GET", "POST"])
- @hsapp.route("/customer/delete_service_request/<int:service_request_id>", methods=["GET", "POST"])
- @hsapp.route("/customer/close_service_request/<int:service_request_id>", methods=["GET", "POST"])
- @hsapp.route("/professional", methods=["GET", "POST"])
- @hsapp.route("/professional/accept_request/<int:service_request_id>", methods=["GET", "POST"])
- @hsapp.route("/professional/reject_request/<int:service_request_id>", methods=["GET", "POST"])
- @hsapp.route("/customer/search", methods=["GET", "POST"])
- @hsapp.route("/admin/summary", methods=["GET", "POST"])

API Design

- **User API**: Handles login and registration for all user types.
- **Admin API**: Manages services and user moderation.
- **Service API**: Retrieves and displays available services based on search criteria.
- **Request API**: Manages service requests created by customers and assigned to workers.

Architecture and Features

The project is structured as follows:

1. **Backend**:
    - app.py: Initializes Flask, database, and login manager.
    - models.py: Defines database tables and relationships.
    - routes.py: Implements logic for user actions and interactions.
2. **Frontend**:
    - HTML templates organized in the templates/ directory.
    - Static assets (CSS, images) placed in the static/ folder.
3. **Key Features**:
    - Separate dashboards for Admins, Customers, and Workers.
    - Admins can approve or block users based on behavior.
    - Workers can view and act on assigned requests.
    - Customers can search for services and submit reviews.

Video Presentation

Link: SWASTIK_GARG_MAD1_PROJECT - Made with Clipchamp.mp4

https://drive.google.com/file/d/1qGD0kn6K66XJvRw11fW_qkb69wYyzDzQ/view?usp=drive_link

Video Structure:

1. Introduction.
2. Problem statement explanation.
3. Implementation details.
4. Application demo, showcasing login/signup, CRUD operations, and other functionalities.