MAD 1 PROJECT (A to Z HOUSEHOLD SERVICES) REPORT

Author:

- Sadakopa Ramakrishnan T
- 23f1001734
- 23f1001734@ds.study.iitm.ac.in
- I am Sadakopa Ramakrishnan, a B.Tech student specialising in Information Technology with a focus on Machine Learning, Data Science, and Software Development at SSN College of Engineering. Passionate about research and hands-on projects, I am also pursuing a Diploma in Programming and Data Science from IIT Madras. With experience in developing applications, managing data, and implementing ML algorithms, I enjoy exploring AI, data-driven solutions, and collaborative work. My recent projects span diverse areas, from machine learning for predictive modelling to multi-user web applications that provide real-world solutions.

Project Overview:

- The Household Services Application is a multi-user platform designed to provide comprehensive home servicing and solutions. It includes role-based access for admins, service professionals, and customers, each with unique functionalities for managing and availing of various home services.

Technologies Used:

1. **Flask:** For developing the backend and managing application logic.
2. **Jinja2 Templates + Bootstrap:** For creating dynamic HTML pages with responsive design.
3. **SQLite:** For data storage and management.

DB Schema Design

Tables used:

1. **User** – Stores details of all users, including admins, service professionals, and customers.
    - **id** --> PK
    - **email** --> unique, not null
    - **password** --> not null
    - **role** --> not null (defines the role as Admin, Customer, or ServiceProfessional)
2. **Customer** – Stores customer-specific details.
    - **id** --> PK
    - **email** --> unique, not null
    - **password** --> not null

    - **full_name** --> not null
    - **address** --> not null
    - **pincode** --> not null
3. **ServiceProfessional** – Stores details of service professionals.
    - **id** --> PK
    - **email** --> unique, not null
    - **password** --> not null
    - **full_name** --> not null
    - service_id --> FK to Service.service_id, not null
    - **experience** --> not null (years of experience)
    - **profile_doc** --> not null (path to profile document)
    - **address** --> not null
    - **pincode** --> not null
    - **date_created** --> default to current date
    - **status** --> not null (status of professional's account)
    - **average_rating** --> computed property for average rating based on associated ratings
4. **Service** – Stores information on available services.
    - **service_id** --> PK
    - **name** --> not null
    - **base_price** --> not null
    - **time_required** --> not null (estimated time for service)
    - **descriptions** --> optional (additional details about the service)
5. **ServiceRequest** – Stores details of service requests made by customers.
    - **serv_req_id** --> PK
    - service_id --> FK to Service.service_id, not null
    - cust_id --> FK to Customer.id, not null
    - professional_id --> FK to ServiceProfessional.id, nullable (assigned professional)
    - **date_of_req** --> default to current date
    - **date_of_completion** --> nullable (completion date of service)
    - **service_status** --> not null (e.g., requested, assigned, completed)
    - **remarks** --> optional (feedback or notes on the service)
6. **Rating** – Stores ratings and reviews provided by customers for services.
    - **id** --> PK
    - service_id --> FK to Service.service_id, not null
    - professional_id --> FK to ServiceProfessional.id, not null
    - customer_id --> FK to Customer.id, not null
    - serv_req_id --> FK to ServiceRequest.serv_req_id, not null
    - **rating** --> not null (numerical rating)
    - **review** --> optional (text review)
    - **rating_status** --> not null (status of the rating)

Architecture and Features

- **Main Program:** The core functionality resides in the `app.py` file, which includes all controllers for handling requests and directing users to the appropriate pages.
- Templates and Application Structure: HTML templates are stored in the templates folder, while the application folder includes database initialization and model definitions. API endpoints are defined within the api.py file, which provides access to services, users, and service requests.

Features:

- 3 separate login methods each for admin, customer and service professionals.
- There is no signup for admin.
- User login and registration are handled by respective controllers that validate form inputs through request.form. If login credentials are incorrect, an error message is displayed.
- Admins can create, edit, or delete services. Input validation is enforced on price and time-related fields to ensure appropriate data is stored.
- Admin can block a customer.
- Admin can view customer profiles or service professional profiles by pressing their ID.
- Admin can search for a professional by filtering with the service or with location.
- Customers can view the services in their homepage which is being created by admin in the dashboard. Customers can press the more details and give a new service request which will be reflected in the admin dashboard for assigning a professional.
- Customers can search for and request specific services by location. If no services match their criteria, a message is displayed.
- Customers can open or close requests and track progress. Service professionals are assigned to handle requests by admin.
- Service professionals view and manage assigned requests and update their status to "completed" upon job completion.
- Customers can rate the service out of 5 and review completed services, visible on the profiles of service professionals.
- Average ratings are computed and displayed in the professional profile to assist other customers in making informed decisions.
- Inputs for creating services, such as base price and required time, are validated to ensure they are correctly formatted.
- If a user tries to request or rate a service without valid data, an error message is displayed. Maximum rating can not exceed 5.
- Admins can view all users, approve service professionals, accept/reject a new professional viewing his document.

- Customers and Service Professionals have an option to edit their professional where they can change their address or set a new password and save which updates the database.
- Service Professional after signing up has to wait till the admin approves their profile. Once the admin has approved the professional joins the platform. If the admin decides to reject, the professional can try to create a new profile.

Video:

- https://drive.google.com/file/d/1kLhHy7-y4uWVfEJyvhBPN8E0_w5fTusY/view?usp=sharing