Project Report for Modern Application Development – I Project

Author: Piyush Gupta

Roll Number: 23f2000400

Student email ID: 23f2000400@ds.study.iitm.ac.in

I'm a BS student with a strong interest in programming and data science. Working on this project was a really fun and rewarding experience. There were definitely times when I got stuck, but finding solutions and overcoming those challenges felt amazing.

Description

In this project, I was tasked with building a flashcard app using HTML, CSS, and Bootstrap for the design; Flask for backend development; RESTful APIs for communication; SQLAlchemy for database management; and other necessary modules. The app includes a login/signup page to securely store usernames and passwords of users.

Technologies used:

1. **Flask**:
   Used to build the web application, serving as the core framework for routing and handling HTTP requests.
2. **Flask-SQLAlchemy**:
   An extension of Flask, used for managing database connections and performing operations with the SQLite database efficiently.
3. **Flask-Login**:
   An extension of Flask, responsible for managing user sessions, including logging in and out, remembering users, and storing active user data securely.
4. **Datetime**:
   A Python module used for time stamping operations within the application, such as logging the time when users access flashcards.
5. **Security**:
   Implemented for securing usernames and passwords to ensure user data protection.
6. **Requests**:
   A Python module used for handling HTTP requests and responses, facilitating smooth communication between the frontend and backend.

DB Schema Design:

1. Users Table:
   - Each user has attributes such as id (primary key), email, username, password, and other personal details.
   - A user can either be a customer or a professional, defined by the role attribute.
2. Customers and Professionals Tables:

- The customers table links each customer to the users table via a user_id foreign key.
- The professionals table also links to the users table via a user_id foreign key and contains professional-specific details such as service_type, experience, rating, and total_ratings.

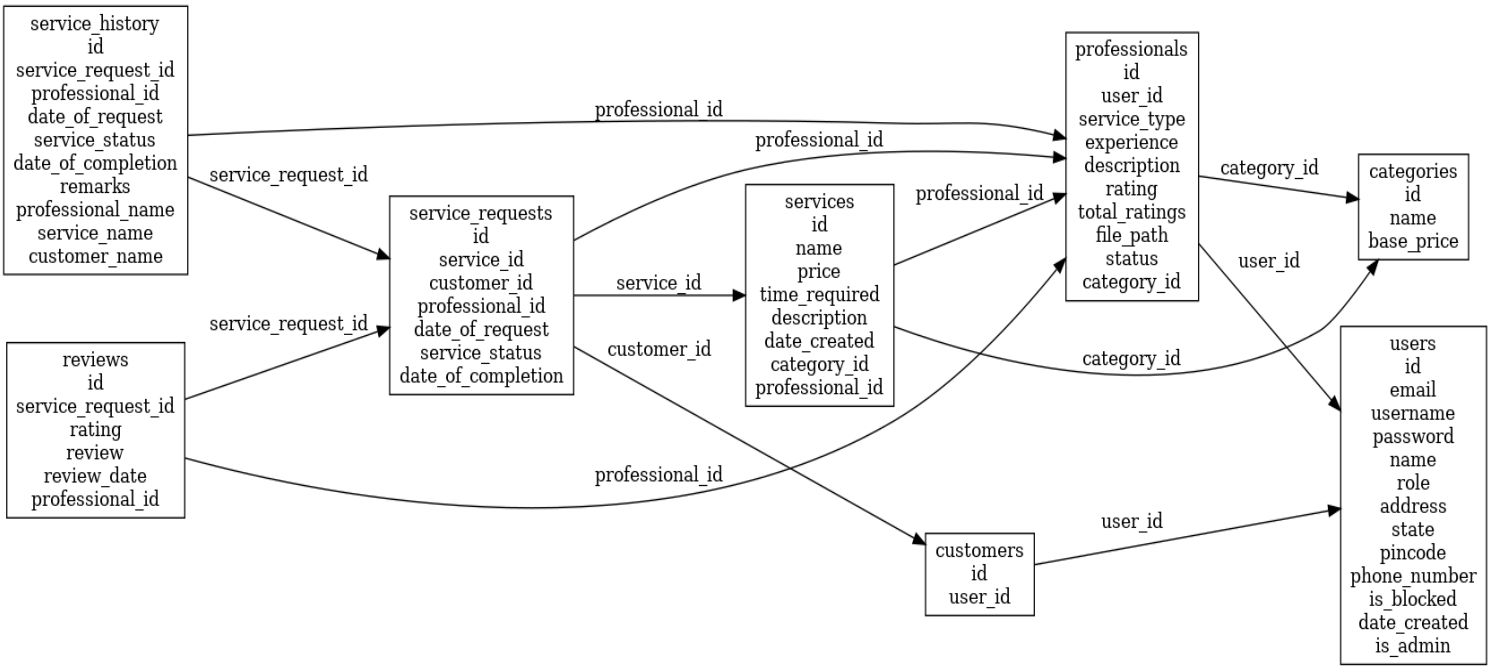
3. Categories Table:
   - Each category (e.g., cleaning, plumbing) has attributes such as id, name, and base_price.
   - It establishes a relationship with professionals through the category_id foreign key.
4. Services Table:
   - Services belong to specific categories and are linked to professionals.
   - Attributes include id (primary key), name, price, time_required, description, and date_created.
5. Service Requests Table:
   - This table tracks requests for services by customers.
   - It has attributes such as id (primary key), service_id (foreign key to services), customer_id (foreign key to customers), and professional_id (foreign key to professionals).
   - Additional details include service_status, date_of_request, and date_of_completion.
6. Reviews Table:
   - Linked to the service_requests table via the service_request_id foreign key. ○ Captures customer feedback with attributes such as rating, review, review_date, and professional_id.
7. Service History Table:
   - Logs historical data about services provided. ○ Includes service_request_id, date_of_request, date_of_completion, and other relevant details like remarks.

Relationships:

- One-to-Many:
  - A professional can offer multiple services, and a service can have multiple requests.
  - A customer can make multiple service requests.
- Many-to-One:
  - Each service request is associated with one specific service.

- Many-to-Many:
    - The schema accommodates customer and professional interactions, where multiple professionals can serve multiple customers through service requests.

This is the ER Diagram:



Architecture and Features

- The project code is organized into multiple files for modularity: ○ Models: Contains the database models, such as User, Customer, Professional, Service, ServiceRequest, and Review.
    - Routes: Defines the application endpoints and implements functionalities for users, services, and reviews.
    - Static: Contains static resources, like images for the UI,professional uploaded documents.
    - Templates: Contains HTML files for different pages, such as login, registration, dashboard, and review submission.
    - App Initialization:
- The `app.py` file serves as the entry point for the application.
- Key configurations include:
    - `SQLALCHEMY_DATABASE_URI`: Connects to SQLite database named `household.db`.
    - `SQLALCHEMY_TRACK_MODIFICATIONS`: Disabled to save resources.
    - `SECRET_KEY`: Used for securing sessions and cookies.

- **Features**:
    - **Role-Based Access**:

        - Admins, professionals, and customers have distinct capabilities.
    - **User Authentication**:
        - Login, registration, and session management.
    - **Service Management**:
        - Customers can book, cancel, or review services.
        - Professionals can manage incoming requests and provide services.
    - **Review System**:
        - Customers can leave reviews and ratings for professionals.
    - **Dashboard**:
        - Displays service requests, status, and progress for admin, customers and professionals.

Video Folder Gdrive Link:
https://drive.google.com/drive/folders/159roup8wY6R1HnibosGuUoiA-82lGtuA?usp=drive_link