IITM Modern Application Development - 1

Household Services

Naganathan M R - 23f2005531

Description:

- This project is a **multi-user platform** designed to provide comprehensive home servicing and solutions, catering to three distinct roles: **Admin**, **Service Professionals**, and **Customers**.
- The **Admin** acts as a super user with root access, overseeing user activities, approving service professionals post-verification, creating services with base pricing, and managing fraud prevention through blocking functionalities.
- **Service Professionals** register to offer specialized services in categories like plumbing, cleaning, or electrical work. They can accept or reject customer requests and build credibility through customer reviews.
- **Customers** can search for services based on their needs and location, book professionals, and provide feedback post-service.

Database Design:

Entity Identification:

The following entities and their respective attributes were identified for the given problem statement,

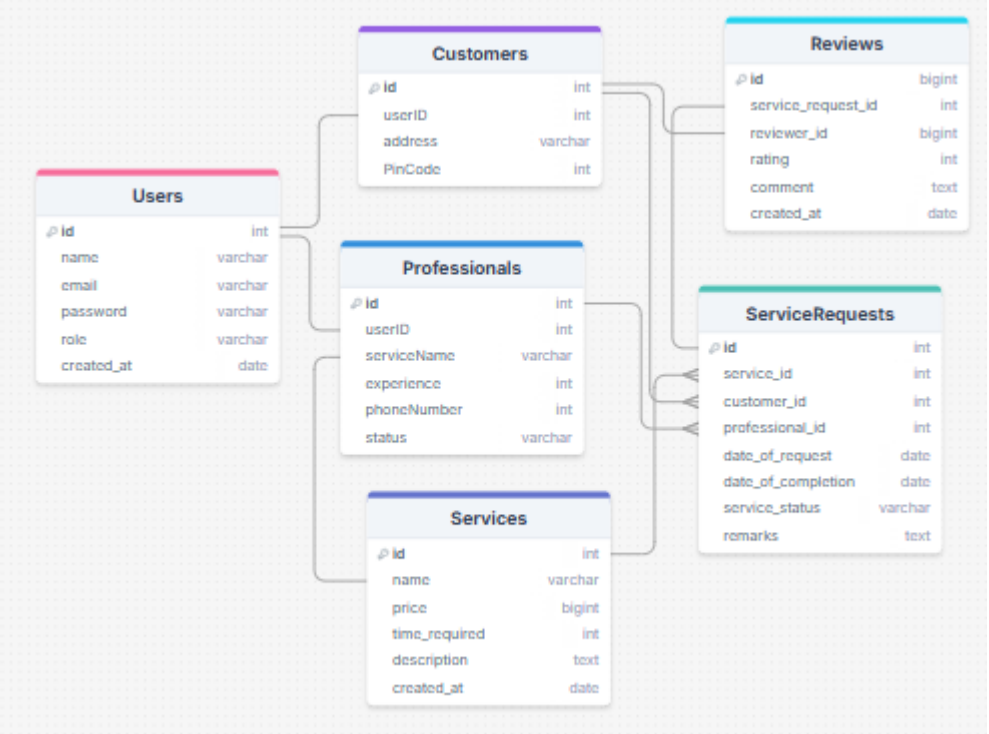
- **Users:** id, name, email, password, role, created_at
- **Services:** id, name, price, time_required, description, created_at
- **Professionals:** id, userID, serviceName, experience, phoneNumber, status
- **Customers:** id, userID, address, PinCode
- **ServiceRequests**: id, service_id, customer_id, professional_id, date_of_request, date_of_completion, service_status, remarks
- **Reviews**: id, service_request_id, reviewer_id, rating, comment, created_at

Relationship Identification:

- Professionals offer Services.
- Professionals link to Users via userID and are associated with Services by serviceName.
- Customers place Service Requests:
- ServiceRequests link a Customer to a Service using customer_id and service_id, respectively.
- Service Requests are fulfilled by Professionals:
- A ServiceRequest can optionally link to a Professional using professional_id (assigned upon acceptance).
- Customers review Services:
- Reviews are tied to a completed ServiceRequest using service_request_id and linked to the reviewer_id (a User).

Schema:

Considering the above entity model, schema diagram has been constructed,



Tech Stack:

• **Flask** – Flask is a backend framework for python. It includes flexible core functionality and an extensive ecosystem of supported modules like Flask-**sqlite3** for database access, Flask-Login for session management, and Flask-RESTful for API development.

• **SQLite** – SQLite is an embedded database engine written in C. It's self-contained, i.e. it doesn't require a separate server process.

• **Jinja 2** – Jinja is a fast, expressive, extensible templating engine. Special placeholders in the template allow writing code similar to Python syntax. Then the template is passed data to render the final document.

API Design:

• **GET request** – For rendering html pages and return data from sqlite server.
• **POST request** – For add and edit operations on the database. Used in adding service requests, services, users, editing services, profile, etc...