Project Report

Student Details

Name: Rajdeep Chatterjee

Roll Number: 21F1006551

Email Id.: 21f1006551@ds.study.iitm.ac.in

Course: Modern Application Development II

Project Details

Problem Statement: The project entails developing a household services application, run by an administrator, that acts as a platform for clients to connect with service providers. The application gives the administrator the ability to oversee user behaviour, authorise professionals, and oversee services. Customers can use it to submit, review, and close service requests, and service providers can accept or reject them according to their availability.

Approach: The approach focused on designing a multi-role application with the following roles: Admin, Service Professional, and Customer. Key functionalities include managing service requests, creating a seamless interface for role-specific operations, and implementing scheduled backend jobs for reminders and reporting.

Frameworks and Libraries

- **Backend:**
  - Flask: API development for handling request/response cycles and role-based access control.
  - SQLite: Database management for structured data storage.
  - Redis: Caching and background job management.
  - Celery: Task scheduling and management for batch jobs and reminders.
- Frontend:
  - VueJS: Frontend interface for dynamic user interaction.
  - Bootstrap: Styling and responsive design.

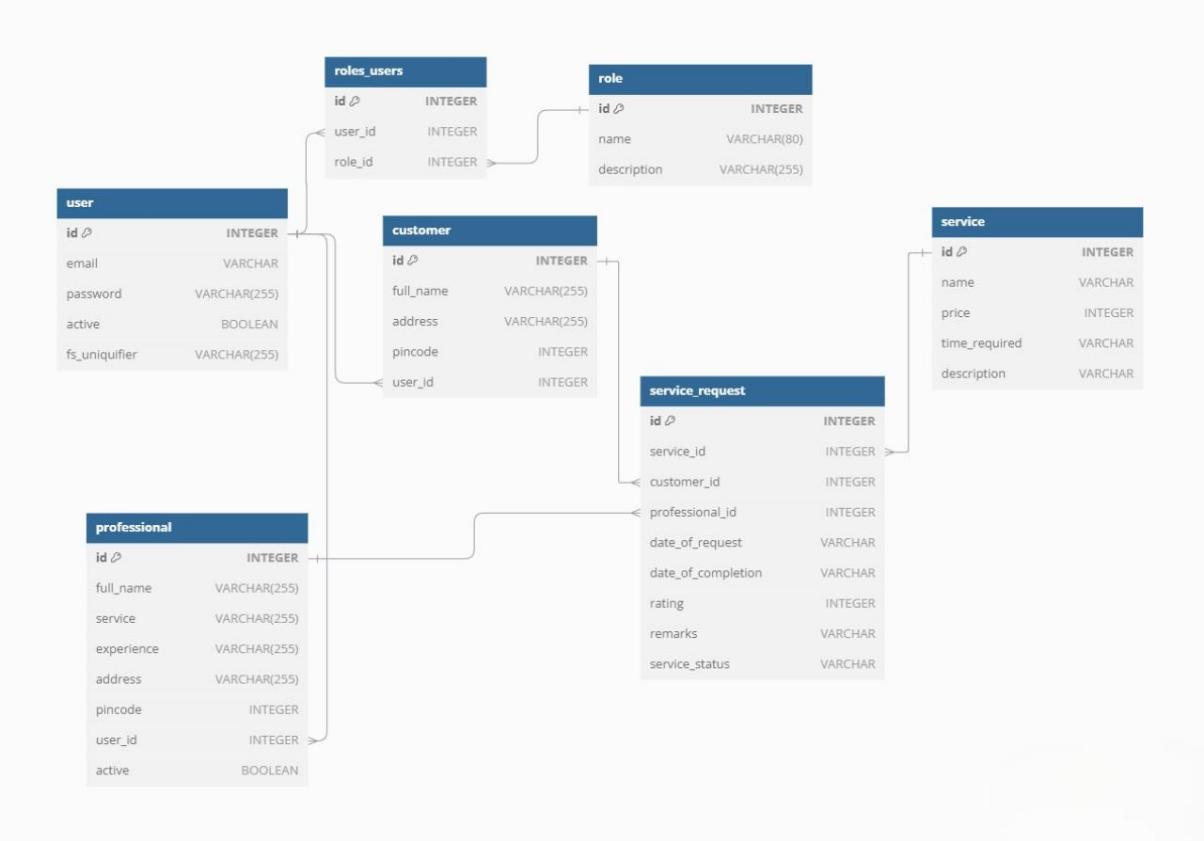
ER Diagram

The database's ER diagram, which highlights the main entities and their connections, is shown below.

Tables:

- user: Stores user details and login information.
- role: Stores user roles like admin, customer, and professional.
- roles_users: Association table connecting users to roles.
- service: Stores service offerings (e.g., plumbing, AC servicing).

- customer: Stores customer profile details.
- professional: Stores professional profile details.
- service_request: Stores customer service requests, connecting customers and professionals.



API Resource Endpoints

1. Service Management

- **Retrieve All Services**
    - **Endpoint**: GET /api/services
    - **Authorization**: Any authenticated user
    - **Description**: Returns a list of all services. The response is cached for 50 seconds to enhance performance.
- **Create a New Service**
    - **Endpoint**: POST /api/services
    - **Authorization**: Admin only
    - **Description**: Creates a new service with specified attributes like name, price, time required, and description.

- **Update Existing Service**
    - **Endpoint**: GET /api/update/service/<int:id>
    - **Authorization**: Admin only
    - **Description**: Retrieves details of a specific service for updating.
    - **Endpoint**: POST /api/update/service/<int:id>
    - **Authorization**: Admin only
    - **Description**: Updates an existing service with provided details.

2. Customer Management

- **Retrieve All Customers**
    - **Endpoint**: GET /api/customers
    - **Authorization**: Admin only
    - **Description**: Returns a list of all customers.
- **Add a New Customer**
    - **Endpoint**: POST /api/customers
    - **Authorization**: Open
    - **Description**: Registers a new customer and creating a user account for it.

3. Professional Management

- **Retrieve All Professionals**
    - **Endpoint**: GET /api/professionals
    - **Authorization**: Admin only
    - **Description**: Returns a list of all professionals.
- **Add a New Professional**
    - **Endpoint**: POST /api/professionals
    - **Authorization**: Open
    - **Description**: Registers a new professional, creates a user account, and associates the professional profile with it.

4. Service Request Management

- **Retrieve All Service Requests**
    - **Endpoint**: GET /api/request/service
    - **Authorization**: Any authenticated user
    - **Description**: Returns a list of all service requests, along with available services.

- **Create a Service Request**
    - **Endpoint**: POST /api/request/service
    - **Authorization**: Customer
    - **Description**: Creates a new service request for a specific service.

5. Manage Service Requests (Professional)

- **Reject a Service Request**
    - **Endpoint**: GET /api/accept/service-request/<int:id>
    - **Authorization**: Professional
    - **Description**: Allows a professional to reject a specific service request.
- **Accept a Service Request**
    - **Endpoint**: POST /api/accept/service-request/<int:id>
    - **Authorization**: Professional
    - **Description**: Allows a professional to accept a specific service request.

6. Customer-Specific Service Requests

- **Retrieve Customer's Service Requests**
    - **Endpoint**: POST /api/service-request/customer
    - **Authorization**: Customer
    - **Description**: Returns a list of service requests created by a specific customer.

7. Close a Service Request

- **Close Service Request**
    - **Endpoint**: POST /api/close/service-request/<int:id>
    - **Authorization**: Professional
    - **Description**: Marks a service request as completed, records a rating and any remarks, and updates the completion date.

Presentation Video

Drive Link: Modern Application II Project Presentation Video