MAD 1 Project:

Customer Service Management Services

Project Report

Video Presentation link: https://youtu.be/DTwRuz0ulJ8

Project Title

Service Request Management System

Abstract

The Service Request Management System is a web-based platform designed to bridge customers and service professionals. It allows customers to request services, service professionals to manage and accept requests, and administrators to oversee the system. The platform includes features such as service request creation, review and rating functionality, user management (blocking and unblocking users), and profile updates.

Table of Contents

---

1. Introduction

The Service Request Management System simplifies the process of connecting customers with qualified service professionals. The system provides a seamless interface for creating, managing, and reviewing service requests. It also ensures accountability through user reviews and administrative control.

---

2. Objectives

- Provide an efficient way for customers to request services.
- Allow service professionals to manage and fulfill requests.
- Facilitate administrator control over users and services.
- Incorporate feedback mechanisms for service improvement.
- Maintain a secure, role-based system for customers, professionals, and admins.

---

3. System Architecture

The system is designed as a **3-tier architecture**:

1. **Presentation Layer**: HTML, CSS, Bootstrap for user-friendly interfaces.
2. **Application Layer**: Flask framework to handle business logic.
3. **Database Layer**: SQLite for storing and retrieving data.

---

4. Key Features

Customer Features:

- Create and edit service requests.

- View assigned service professionals.
- Submit reviews for completed services.
- Manage their profiles.

Service Professional Features:

- Accept or reject service requests.
- Close completed service requests.
- View feedback provided by customers.
- Update their profiles.

Admin Features:

- Add, edit, and delete services.
- Approve or block/unblock users (customers and professionals).
- Monitor reviews and feedback.
- Oversee the entire system.

---

5. Modules

5.1 User Management

- Customer registration and login.
- Service Professional registration and login.
- Admin login.

5.2 Service Request Management

- Customers can create, edit, and close service requests.
- Service professionals can accept and close assigned requests.

5.3 Review System

- Customers can provide reviews and ratings for completed services.

- Reviews are displayed to both professionals and admins.

5.4 Administrative Control

- Admins can manage users and services.
- Admins can monitor system activity through reviews and feedback.

---

6. Implementation

Frontend:

- HTML templates were designed using **Bootstrap** for responsiveness.
- Dynamic content rendering was achieved through **Flask's Jinja2 templating**.

Backend:

- Built using **Flask**, a lightweight Python web framework.
- Routes and models were implemented to handle CRUD operations.
- Flask-Login ensures secure session management for role-based access.

Database:

- SQLite database stores information about users, services, service requests, and reviews.
- SQLAlchemy ORM is used for database interactions.

---

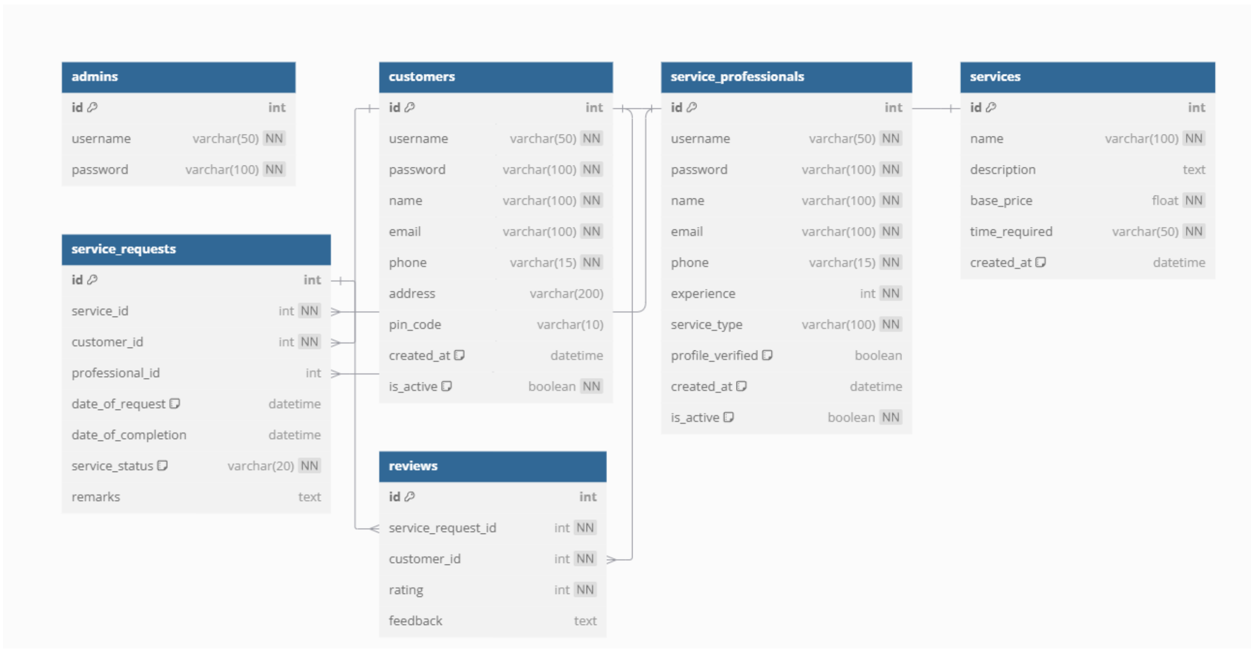
7. Database Design

Tables:

1. **Users (Customers, Service Professionals, Admins)**:
    - Fields: `id`, `name`, `email`, `phone`, `role`, `is_active`, etc.
2. **Services**:
    - Fields: `id`, `name`, `description`, `base_price`, `time_required`.

3. **Service Requests**:
   - Fields: `id`, `customer_id`, `professional_id`, `service_id`, `status`, `remarks`, `date_created`, `date_completed`.
4. **Reviews**:
   - Fields: `id`, `service_request_id`, `customer_id`, `rating`, `feedback`.

- In details



8. Technologies Used

- **Frontend**: HTML, CSS, Bootstrap, Jinja2
- **Backend**: Flask, Python
- **Database**: SQLite, SQLAlchemy
- **Authentication**: Flask-Login
- **Web Server**: Werkzeug (development)

9. Testing and Debugging

- Conducted unit testing for routes and models.
- Tested role-based access control for secure operations.
- Debugged common issues such as `IntegrityError` and routing errors during development.
- Verified CRUD operations for service requests and reviews.

---

10. Future Enhancements

- **Advanced Search**: Implement filters for professionals and services.
- **Notifications**: Add email or SMS notifications for service updates.
- **Analytics Dashboard**: Provide analytics for admin insights (e.g., most popular services).
- **Payment Integration**: Allow payments for service requests within the platform.

---

11. Conclusion

The application is basic but does most of the work that is required as per the services, thank you for going through my project.
