QUICK FIX

Scope

Project Goal:To develop a web-based platform that connects users with various service providers, including plumbers, carpenters, electricians, salons, and tutors.

Project Boundaries:

In-Scope:

User registration and login

Service provider registration and login

Service provider profile display

Booking and order management

Payment processing

Role-based security

Out-of-Scope:

Real-time chat or messaging features

Geolocation-based service matching

A mobile app

Integration with third-party scheduling tools

Functional Requirements

1. User Registration and Login:

Allow users to create accounts with basic information (name, email, password).

Implement secure login mechanisms.

2. Service Provider Registration and Login:

Allow service providers to create profiles, including details like name, services offered, experience, and contact information.

Implement secure login mechanisms for service providers.

3. Service Provider Profile Display:

Display service provider profiles with relevant information, such as name, services, ratings, and reviews.

Allow users to search for service providers based on specific criteria.

4. Booking and Order Management:

Enable users to book services from registered service providers.

Track the status of orders (pending, in progress, completed).

Allow users to cancel or modify bookings.

Notify service providers of new bookings.

5. Payment Processing:

Integrate a secure payment gateway to handle transactions.

Require payment before booking a service.

Provide options for different payment methods (e.g., credit card, digital wallets).

6. Role-Based Security:

Enforce different access levels for users and service providers.

Restrict certain functionalities based on user roles.

7. Service Provider Visibility:

Hide service provider details until the order is completed.

Prevent users from booking the same service provider multiple times.

Non-Functional Requirements

1. Performance:

Ensure the website loads quickly and responds efficiently to user interactions.

Optimize database queries and network requests.

2. Security:

Implement robust security measures to protect user data and prevent unauthorized access.

Use encryption for sensitive information.

Regularly update and patch the system to address vulnerabilities.

3. Scalability:

Design the system to handle increasing user and service provider loads.

Consider using cloud-based infrastructure for scalability.

4. Reliability:

Ensure high availability and minimal downtime.

Implement redundancy and backups to protect against data loss.

5. Usability:

Provide an intuitive and user-friendly interface.

Use clear and concise language.

Offer helpful guidance and tooltips.

Technology Used:-

Backend:

Spring Boot

Spring Reactive WebFlux

Spring Security

Spring Data JPA

PostgreSQL (or other suitable database)

Payment gateway integration (e.g., Stripe, PayPal)

Frontend:

React

JavaScript

CSS

HTML

High-Level Design

Components:

User Service: Handles user registration, login, and profile management.

Service Provider Service: Handles service provider registration, login, profile management, and order management.

OrderService: Manages booking, order status, and payment processing.

Security Service: Implements role-based access control and authentication.

Payment Gateway Integration: Handles payment transactions.

Interactions:

1. User registers or logs in.
2. User searches for a service provider.
3. User selects a service provider and books a service.
4. OrderService processes the booking and payment.
5. Service provider receives a notification.
6. Service provider completes the service.
7. User updates the order status as completed.

Low-Level Design

Data Model:

User table: id, name, email, password, role (user, service provider)

Service Provider table: id, name, services, experience, contact, rating

Order table: id, user_id, service_provider_id, status, payment_status

Review table: id, user_id, service_provider_id, rating, comment

API Endpoints:

User endpoints: register, login, profile

Service Provider endpoints: register, login, profile, update_profile

Order endpoints: create, update_status, cancel

Payment endpoints: process_payment

Security:

Use Spring Security for authentication and authorization.

Implement JWT token-based authentication.

Database:

Use PostgreSQL or another suitable database.

Create indexes for frequently queried fields.

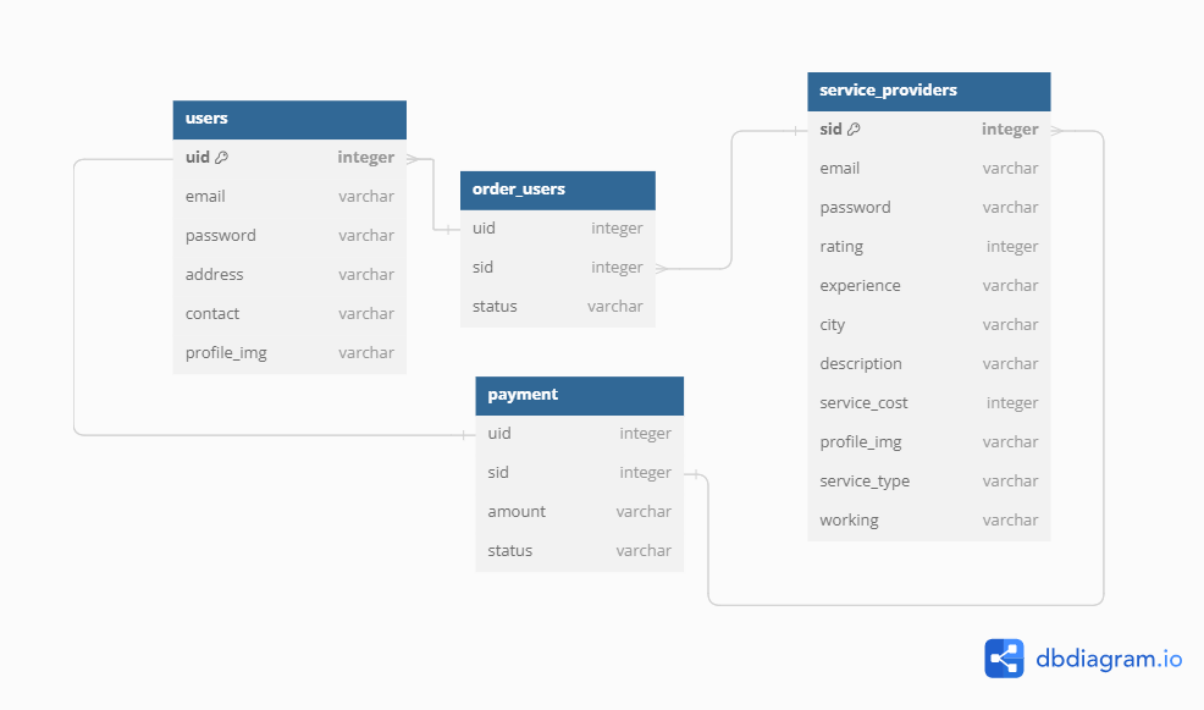
Frontend:

Use React to build the user interface.

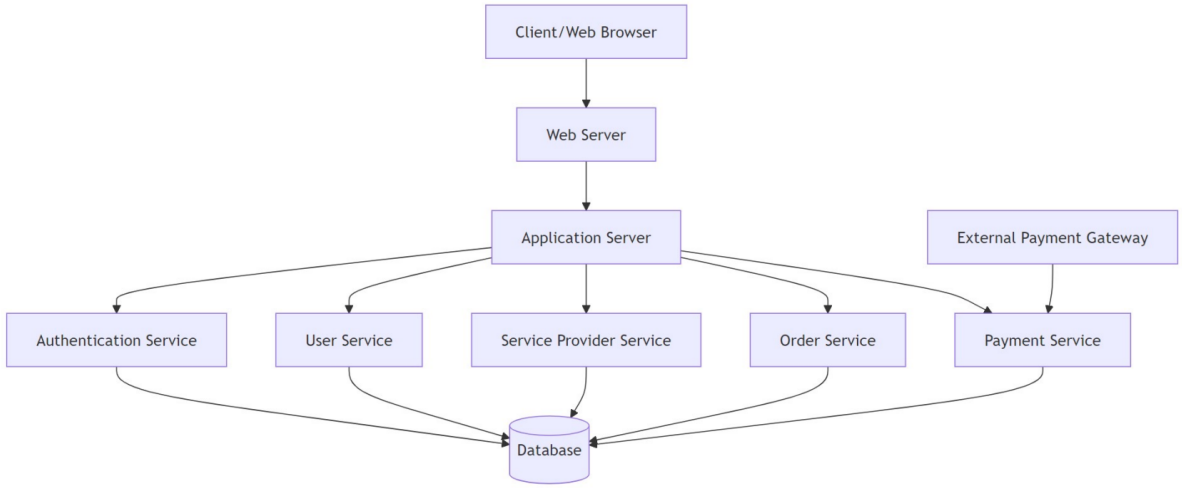
Implement components for user registration, login, service provider search, booking, and order management.

Use state management (e.g., Redux, Context API) to manage application state.

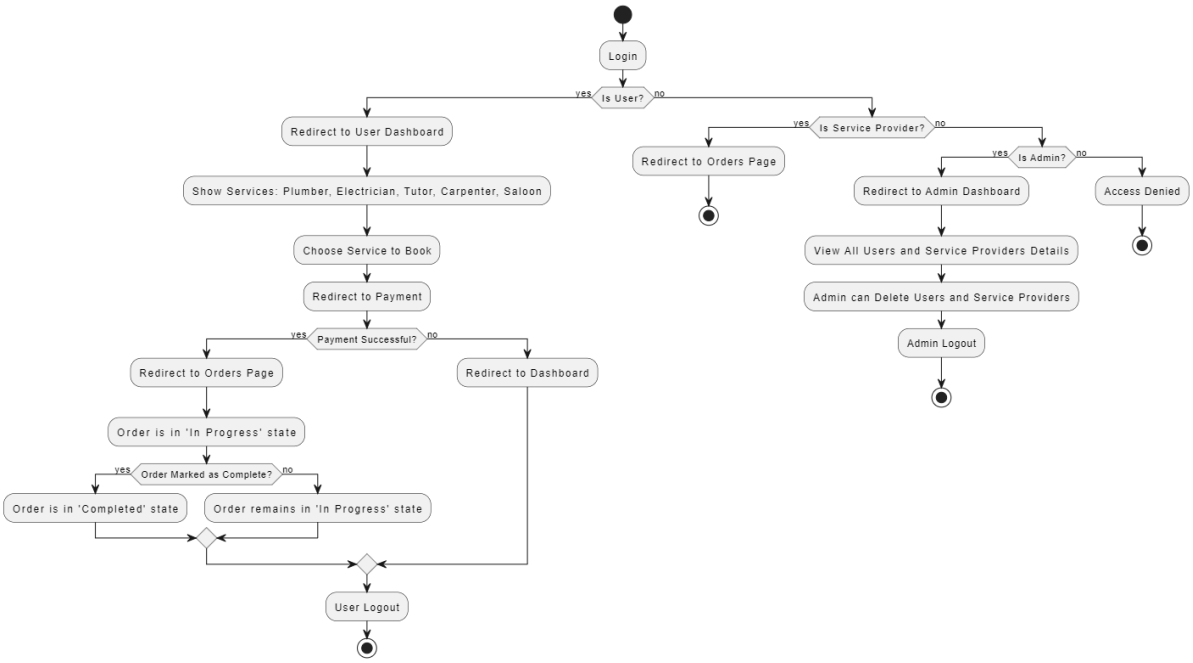
DataBase Diagram:



SYSTEM ARCHITECTURE:



DATA FLOW DIAGRAM



API`s USED:

1.RAZORPAY:-

https://checkout.razorpay.com/v1/checkout.js

used for the payment when the user books a service.

2.Tawk.to

https://embed.tawk.to/66d5e48750c10f7a00a323dd/1i6po9d54

used for chating with the admin..