**Project Report: Smart Service Connect**

**1. Title of the Project:** Smart Service Connect – A Full Stack Web Application for Booking Local Services

**2. Abstract:** Smart Service Connect is a full stack web application that bridges the gap between users who seek reliable service providers (such as electricians, plumbers, carpenters) and professionals who offer these services. The system enables users to register, browse services, book appointments, and provide feedback, while service providers can manage their listings and accept bookings. Built using the MERN stack (MongoDB, Express.js, React.js, Node.js), the application aims to provide a modern, scalable, and secure platform to make service discovery and management seamless.

**3. Problem Statement:** Finding reliable, nearby service providers is often difficult. Existing platforms are fragmented and fail to provide verified listings, real-time availability, and integrated booking systems. On the other side, local service providers struggle to reach a larger customer base due to a lack of digital presence. This project aims to solve both problems through a centralized digital platform that simplifies discovery, booking, and communication.

**4. Objectives of the Project:**

* To develop a user-friendly platform for service seekers to find and book local service providers.
* To create provider dashboards where professionals can manage bookings and services.
* To implement secure user authentication.
* To store and manage user, provider, booking, and review data in an organized NoSQL database.
* To offer a scalable backend API that serves frontend requests efficiently.

**5. Literature Survey:** Platforms like UrbanClap (now Urban Company), Sulekha, and JustDial offer service listings. However, they often:

* Do not provide real-time booking and availability.
* Have cluttered interfaces or limited verification mechanisms.
* Offer minimal backend flexibility for providers.

This project focuses on:

* Role-based access (User/Provider)
* Real-time booking and service filtering
* A simpler, modern UI using React.js

**6. System Architecture:** The architecture of Smart Service Connect follows the MVC pattern and is divided into:

* **Frontend (React.js):** Handles user interface, service listing, forms, dashboards.
* **Backend (Node.js + Express.js):** API handling, business logic, authentication.
* **Database (MongoDB):** NoSQL storage of users, services, bookings, and reviews.

**Architectural Flow:**

User / Provider Browser (React Frontend)

|

REST API Requests (Axios)

|

Node.js + Express.js Backend

|

MongoDB Database

**7. Module Description:**

**7.1 User Module:**

* Register/Login
* Browse available services
* Book a service
* View booking history
* Leave reviews

**7.2 Service Provider Module:**

* Register/Login
* Add service listings
* Manage bookings
* View feedback

**7.3 Booking Module:**

* Create new booking entries
* Track status (Pending, Confirmed, Completed)
* Connect users and providers

**7.4 Review Module:**

* Allow users to rate and review services
* Store and fetch reviews

**7.5 Admin Module (Optional/Future Scope):**

* Monitor all activity
* Moderate listings and reviews

**7.6 Authentication Module:**

* JWT-based secure login and session management
* Role-based route protection

**8. Frontend and Backend Details:**

**Frontend (React.js):**

* Component-based structure (Home, Login, Dashboard, Booking Form, Reviews)
* React Router for navigation
* Axios for API communication
* TailwindCSS/CSS for styling

**Backend (Node.js + Express.js):**

* RESTful API development
* Mongoose models for MongoDB collections
* Controllers for handling logic
* JWT Authentication middleware
* Environment configurations for database, port, etc.

**9. Database Schema (MongoDB Collections):**

**User Collection:**

* name
* email
* password (hashed)
* role (user/provider)

**Service Collection:**

* serviceType
* providerId
* description
* availability

**Booking Collection:**

* userId
* providerId
* date/time
* status

**Review Collection:**

* userId
* providerId
* rating
* comment

**10. Software and Tools Used:**

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| React.js | Frontend framework |
| Node.js | Backend runtime environment |
| Express.js | Backend framework |
| MongoDB | NoSQL database |
| Mongoose | MongoDB object modeling |
| JWT | Authentication and authorization |
| Postman | API testing |
| Git & GitHub | Version control |
| Visual Studio Code | Code editor |
| npm | Package manager |

**11. Features and Functionality:**

* Secure login and registration using JWT
* Role-based dashboards for users and providers
* Real-time service booking system
* Provider management system
* Booking history and review module
* RESTful APIs with modular codebase

**12. Future Scope:**

* Payment gateway integration (e.g., Razorpay)
* Admin dashboard for moderation
* Push notifications and reminders
* Live chat between users and providers
* AI-based service suggestions

**13. Conclusion:** Smart Service Connect simplifies the traditionally manual and chaotic process of finding and managing local services. By providing a reliable, secure, and interactive digital platform, the project enhances access for users and empowers providers. Through modern technologies and scalable architecture, the application can be extended to meet future real-world demands.

**14. References:**

* https://reactjs.org
* https://expressjs.com
* https://nodejs.org
* https://www.mongodb.com
* https://jwt.io
* https://postman.com