Full Stack Development with MERN – Final Project Report

Project: SB Works – FreelanceFinder

Team ID: LTVIP2025TMID55674

Team Size: 4

Team Leader: Bolleddula Vivek Rahul Rai

Team Members:  
- Battu Baby Sarojini  
- Batraju Vamsikrishna  
- Barama Haritha Sai

**Team Members and Roles**

| **Name** | **Role** |
| --- | --- |
| **Bolleddula Vivek Rahul Rai** | Team Lead, Backend Integration, Architecture |
| **Battu Baby Sarojini** | UI Development, React Components |
| **Batraju Vamsikrishna** | Project Coordination, API Integration, Deployment |
| **Barama Haritha Sai** | Testing, Documentation, Support Modules |

# 1. Introduction

Welcome to SB Works, a revolutionary freelancing platform that transforms the way clients connect with skilled freelancers. Our intuitive interface provides clients with the opportunity to post diverse projects, ranging from creative endeavours to technical tasks, while freelancers can seamlessly bid on these projects based on their expertise and capabilities.

At SB Works, we prioritize efficiency and transparency in the freelancing process. Clients can review freelancer profiles, assess past work, and select the perfect candidate for their project. Once a freelancer is chosen, the client can easily communicate and collaborate with them within the platform, streamlining the entire workflow.

Our dedicated admin team ensures the integrity and security of every transaction. With stringent oversight, we guarantee the reliability and quality of the freelancers on our platform. The admin's role is not only to maintain the platform's integrity but also to facilitate smooth communication between clients and freelancers, ensuring a positive and productive working relationship.

Freelancers on SB Works benefit from a straightforward project submission process. After completing the assigned project, freelancers can submit their work directly through the platform, offering clients a hassle-free experience. Clients have the opportunity to review the work and provide feedback, fostering a collaborative environment that values excellence.

Stay informed about the latest projects and industry trends with real-time updates and notifications. SB Works aims to be the go-to platform for clients seeking reliable freelancers and freelancers looking for exciting opportunities to showcase their skills.

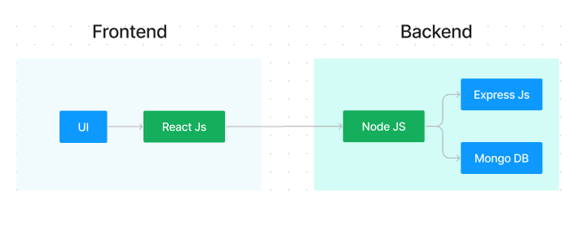
Join SB Works today and experience a new era of freelancing where your projects are efficiently managed, your skills are recognized, and collaborations flourish in a secure and dynamic environment.

# 2. Project Overview

**Purpose:**  
SB Works is a freelancing platform enabling clients to post projects and freelancers to bid, collaborate, and deliver work efficiently. It streamlines freelance hiring with real-time communication, secure transactions, and user-friendly design.

**Features:**- Client-freelancer collaboration with in-app chat  
- Admin moderation and platform oversight  
- Project posting, bidding, and tracking  
- Secure login/register system with authentication  
- Notification system for updates and alerts

# 3. Architecture



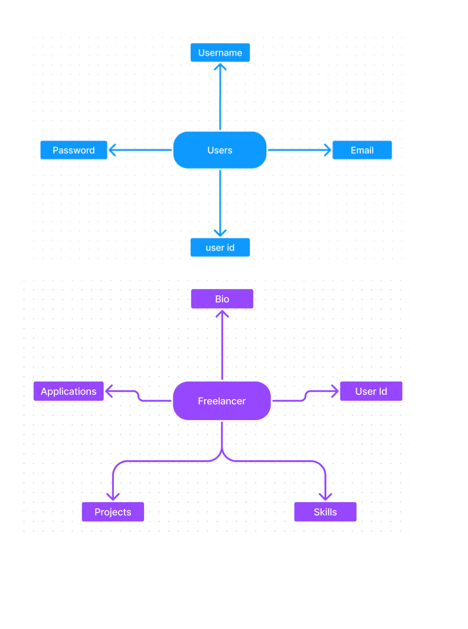
The technical architecture of SB Works follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses the user interface, presentation, and integrates the Axios library to facilitate easy communication with the backend through RESTful APIs.

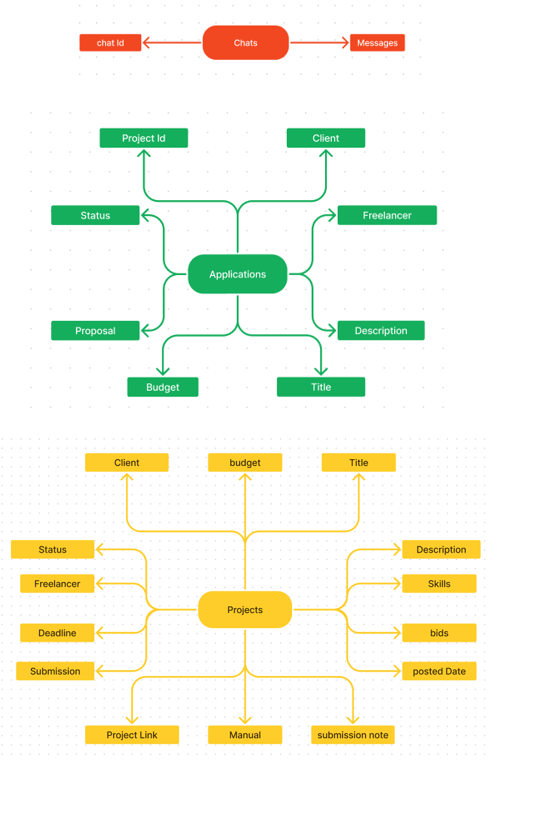
To enhance the user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a real-time and visually appealing interface for users.

On the backend, we utilize the Express Js framework to manage server-side logic and communication. Express Js provides a robust foundation for handling requests and responses efficiently.

For data storage and retrieval, SB Works relies on MongoDB. MongoDB offers a scalable and efficient solution for storing various data, including user-contributed locations and images. This ensures quick and reliable access to the information needed to enrich the local tourism experience.

In conjunction, the frontend and backend components, complemented by Express Js, and MongoDB, together form a comprehensive technical architecture for SB Works. This architecture facilitates real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive experience for users contributing to and exploring their local surroundings.

**ER DIAGRAM**



# 4. Setup Instructions

• **Prerequisites:** List software dependencies (e.g., Node.js, MongoDB).  
• **Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables.

**Prerequisites**

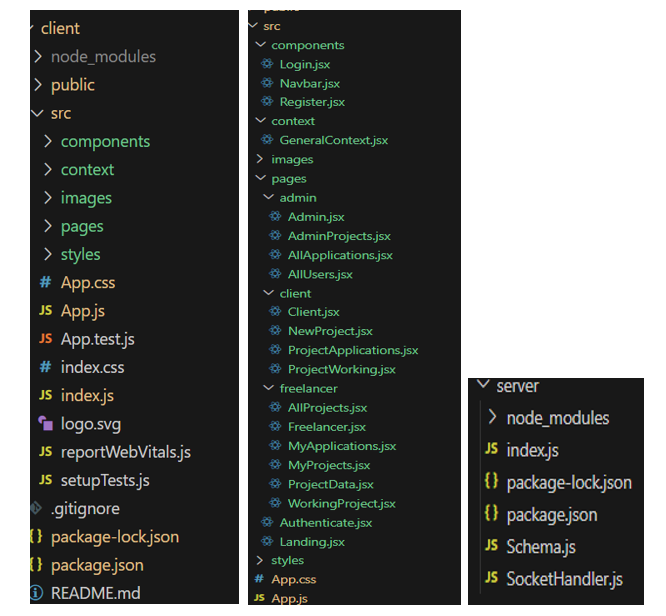
Before setting up the project, make sure the following software is installed:

* **Node.js and npm** – JavaScript runtime and package manager  
  🔗 <https://nodejs.org/en/download>
* **MongoDB** – NoSQL database for storing project and user data  
  🔗 <https://www.mongodb.com/try/download/community>
* **Git** – Version control system for managing source code  
  🔗 <https://git-scm.com/downloads>
* **Visual Studio Code** – Recommended IDE for development  
  🔗 [https://code.visualstudio.com](https://code.visualstudio.com/)

**Installation & Setup Steps**

1. Clone the Repository
2. git clone [YOUR\_PROJECT\_REPO\_LINK]
3. cd freelancer-app-MERN
4. Install Frontend Dependencies
5. cd client
6. npm install
7. Install Backend Dependencies
8. cd ../server
9. npm install
10. Environment Variables  
    Create a .env file in the /server directory and add your environment configs:
11. PORT=5000
12. MONGO\_URI=your\_mongodb\_connection\_string
13. JWT\_SECRET=your\_jwt\_secret
14. Start the Application
    * Start Frontend
    * cd client
    * npm start
    * Start Backend
    * cd server
    * npm start
15. Open your browser and go to:  
     http://localhost:3000

# 5. Folder Structure



The first part shows the whole structure of the client (React Js) code. The second images show the internal content of the component section, pages, and context sections in the client’s src folder. The third image shows the server (Node JS) code structure.

**PRE-REQUISTIC:**

Here are the key prerequisites for developing a full-stack application using Express Js, MongoDB, React.js:

**✔Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

**Download:** <https://nodejs.org/en/download/>

**Installation instructions:**

npm init

**✔Express.js:**

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.

Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

**Installation**: Open your command prompt or terminal and run the following command:

npm install express

**MongoDB**:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

**Download:** <https://www.mongodb.com/try/download/community>

**Installation instructions:**

**React.js:**

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

**Follow the installation guide:**

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Connectivity Database:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Express Js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations

**Front-end Framework:** Utilize React Js to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard. For making better UI we have also used some libraries like material UI and bootstrap.

**Version Control:** Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

**Git:** Download and installation instructions can be found at: <https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from

To run the existing Freelancer App project downloaded from Drive:

**Use the code:**

Drive link:

**Install Dependencies:**

• Navigate into the cloned repository directory:

cd freelancer-app-MERN

• Install the required dependencies by running the following commands:

cd client

npm install

../cd server

npm install

**Start the Development Server:**

• To start the development server, execute the following command:

npm start

• The SB Works app will be accessible at

You have successfully installed and set up the SB Works application on your local machine. You can now proceed with further customization, development, and testing as needed.

# 6. Running the Application

• Provide commands to start the frontend and backend servers locally.  
    o **Frontend:** npm start in the client directory.  
    o **Backend:** npm start in the server directory.

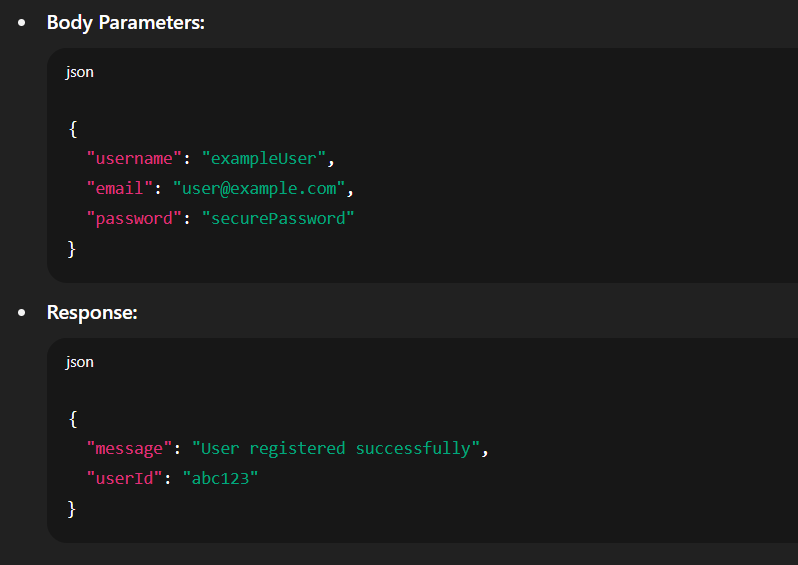
Visit: http://localhost:3000

# 7. API Documentation

• Document all endpoints exposed by the backend.  
• Include request methods, parameters, and example responses.

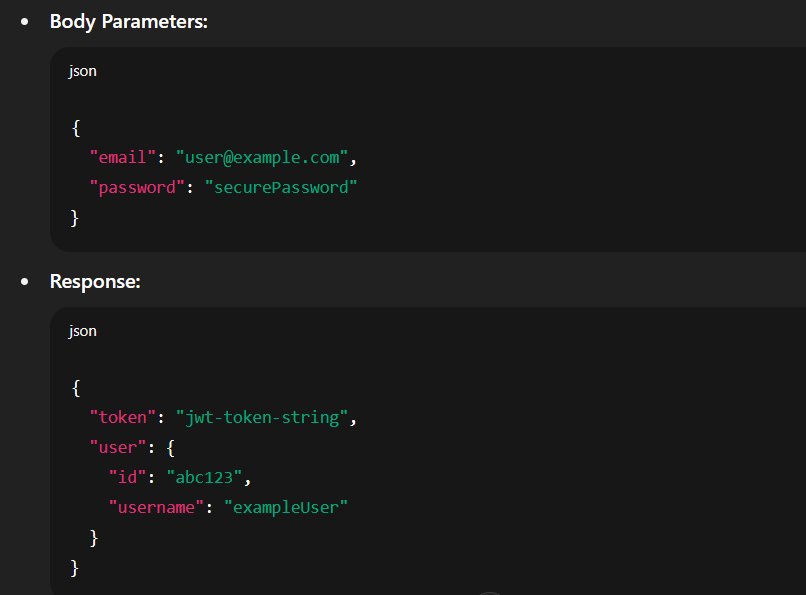
**Authentication Endpoints**

**1. Register a New User**

* **Method:** POST
* **Endpoint:** /api/auth/register

**2. User Login**

* **Method:** POST
* **Endpoint:** /api/auth/login

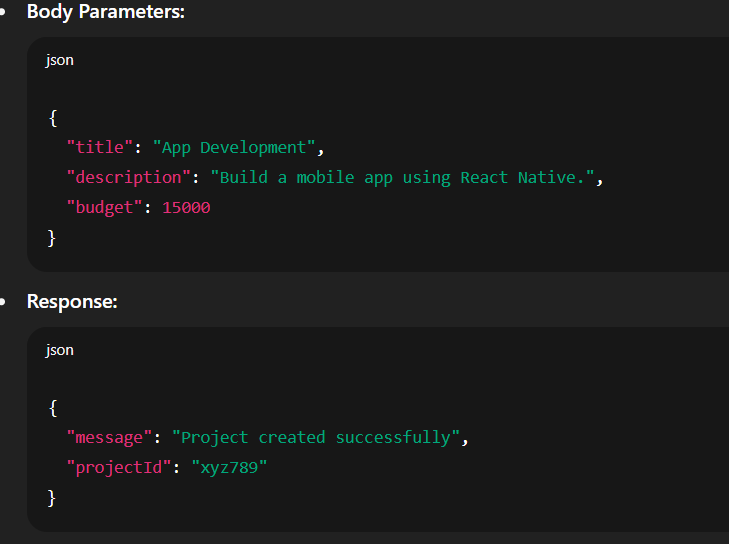


**Project Endpoints**

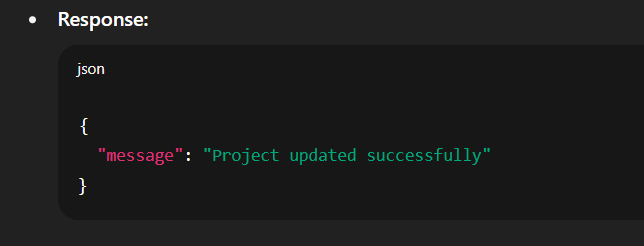
**1. Get All Projects**

* **Method:** GET
* **Endpoint:** /api/projects

**2. Post a New Project**

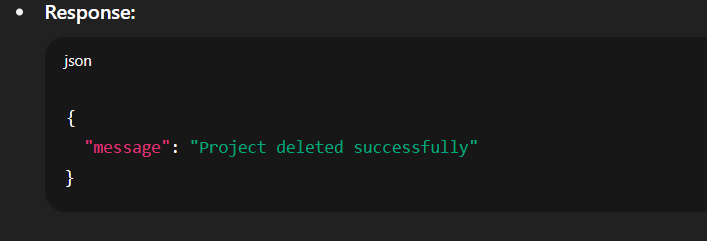
* **Method:** POST
* **Endpoint:** /api/projects

**3. Update a Project**

* **Method:** PUT
* **Endpoint:** /api/projects/:id

**4. Delete a Project**

* **Method:** DELETE
* **Endpoint:** /api/projects/:id



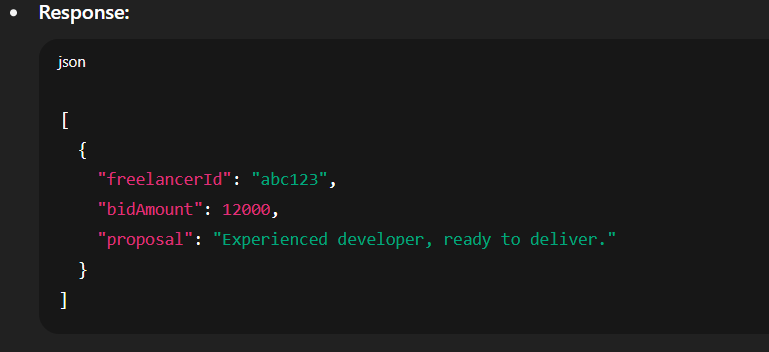
**Bid Endpoints**

**1. Place a Bid on a Project**

* **Method:** POST
* **Endpoint:** /api/bids

**2. Get Bids for a Project**

* **Method:** GET
* **Endpoint:** /api/bids/:projectId



**8. Authentication**

**Authentication & Authorization Handling in SB Works**

The project implements **JWT-based (JSON Web Token) authentication** for secure access control to the backend APIs.

**User Registration & Login Flow**

* When a user registers (/api/auth/register), their **password is hashed** using **bcrypt** before saving to MongoDB.
* During login (/api/auth/login), the submitted password is compared with the hashed password using bcrypt’s compare() method.
* If the credentials are valid, the server generates a **JWT token** using the jsonwebtoken library.

**Token-Based Session Management**

* Upon successful login, a **JWT token** is issued and sent to the client.
* The token contains encoded user information (e.g., user ID) and is signed with a secret key.
* This token is stored on the **client side** (usually in localStorage or HTTP-only cookies) and sent with each request to authenticate the user.

**Protected Routes**

* Backend routes are protected using middleware (authMiddleware) that:
  1. Extracts the token from the request header.
  2. Verifies the token using jsonwebtoken.verify().
  3. Decodes the token and attaches user info to req.user.

If the token is missing or invalid, access is denied with a 401 Unauthorized response.



# 9. User Interface

Screenshots of the UI are provided in the next section.

- Home Page  
- Project Posting and Bidding Page  
- Freelancer Dashboard  
- Client Dashboard  
- Admin Control Panel

# 10. Testing

**10. Testing**

• Describe the testing strategy and tools used.  
• VS Code is used as the primary development and testing environment.

**Testing Strategy**

The SB Works platform was tested through a combination of **manual testing** and **API validation**, ensuring each module worked as expected during development.

**Tools Used**

* **Visual Studio Code (VS Code):**  
  Used for coding, running local servers, debugging both client and server logic, and testing UI behavior using the browser developer tools.
* **Postman:**  
  Used to test REST API endpoints such as login, registration, project posting, and bid submission. This helped validate both successful and failed request handling.
* **MongoDB Compass:**  
  Used to visually verify that data was being properly inserted, updated, and deleted from the MongoDB database during CRUD operations.
* **Browser Dev Tools (Chrome):**  
  For inspecting frontend state, console errors, and network requests.

**Testing Coverage**

* **Authentication:**  
  Registration, login, and token validation
* **Project Posting:**  
  Posting a project, fetching all projects, and editing/deleting as needed
* **Freelancer Bidding:**  
  Submitting and retrieving bids for various projects
* **UI/UX Testing:**  
  Checked responsiveness, form validations, and error handling on multiple screen sizes

.

# 11. Screenshots or Demo

[Insert Screenshot 1: Home Page]  
[Insert Screenshot 2: Freelancer Dashboard]  
[Insert Screenshot 3: Client Project View]  
[Insert Screenshot 4: Admin Panel]

(Optional: Insert link to demo video hosted on Google Drive or YouTube)

# 12. Known Issues

- Chat not yet real-time (Socket.IO pending)  
- Payment integration not implemented  
- UI responsiveness may vary on smaller screens

# 13. Future Enhancements

- Add payment gateway integration (Stripe/Razorpay)  
- Implement real-time chat using WebSockets  
- Add freelancer ratings & reviews  
- Mobile app using React Native

# 14. Roles and Responsibilities

**Freelancer Responsibilities:**

• Project Submission: Freelancers are responsible for submitting completed and high-quality work for the assigned projects through the platform.

• Compliance: Ensure that the submitted work adheres to client requirements, industry standards, and any specific guidelines outlined by the platform.

• Effective Communication: Actively engage in communication with clients, promptly responding to messages, asking clarifying questions, and providing updates on the project progress.

• Time Management: Manage time effectively to meet project deadlines and deliver work in a timely manner.

• Professionalism: Conduct oneself professionally by maintaining a respectful and cooperative attitude with clients and fellow freelancers.

• Quality Assurance: Deliver work that is accurate, well-executed, and free from errors to maintain client satisfaction.

**Client Responsibilities:**

• Clear Project Description: Provide a detailed and comprehensive project description, including deliverables, desired outcomes, and any specific requirements.

• Timely Communication: Respond promptly to freelancer inquiries, providing necessary information and feedback in a timely manner.

• Payment Obligations: Fulfill the agreed-upon payment terms promptly and fairly upon satisfactory completion of the project.

• Feedback and Evaluation: Provide constructive feedback and evaluate the freelancer's performance, helping them improve and providing valuable insights.

**Admin Responsibilities:**

Data Oversight: As an admin, one of your key responsibilities is to monitor and ensure the integrity and security of all data on the platform

Policy Enforcement: Admins play a crucial role in enforcing platform policies, guidelines, and ethical standards.

Conflict Resolution: In the event of disputes or issues within the community, it is the admin's responsibility to address them promptly and impartially

User Support and Communication: Admins should provide support and guidance to users on the platform

Platform Maintenance and Improvement: Admins are responsible for the overall maintenance and improvement of the research platform.

# 16. Project Flow

**Use the code in:**

**Milestone 1:** Project setup and configuration.

Folder setup:

Create frontend and

Backend folders

**Installation of required tools:**

1. Open the frontend folder to install necessary tools

For frontend, we use:

React

Bootstrap

Material UI

Axios

react-bootstrap

2. Open the backend folder to install necessary tools

For backend, we use:

Express Js

Node JS

MongoDB

Mongoose

Cors

bcrypt

**Milestone 2:** Backend Development

Setup Express Js server

Create index.js file in the server (backend folder).

define port number and MongoDB connection string.

Configure the server by adding middleware.

Configure MongoDB

Import mongoose.

Add database connection from config.js file present in config folder

Create a model folder to store all the DB schemas user and location schemas.

**Add authentication:**

For authentication, we need to define URL for login and register processes in the backend. By getting the data from request from the client, we need to perform required operations.

**Milestone 3:** Web development (project completion)

**UI development:**

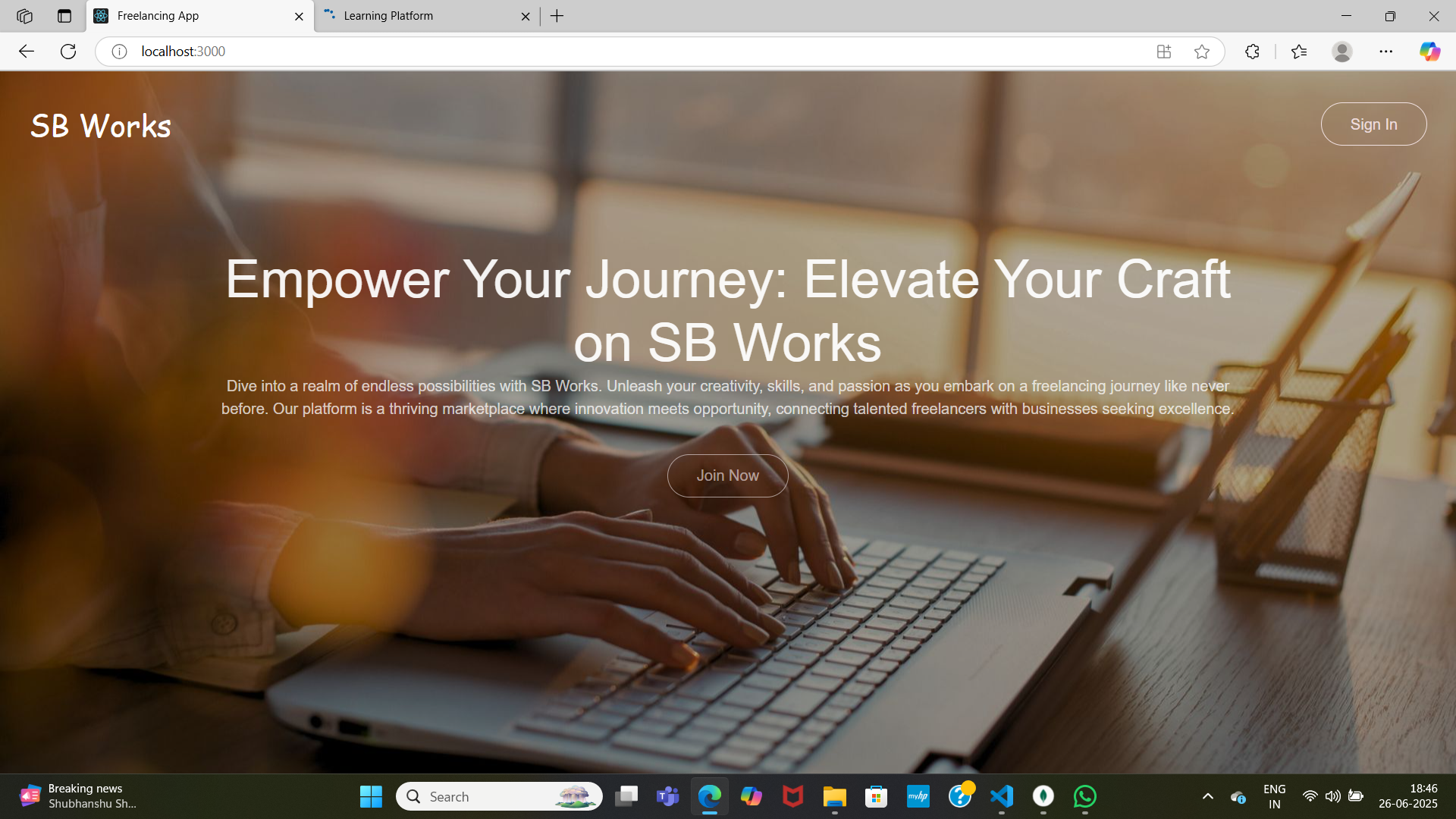
Complete the User Interface for the project as per the requirements. Design the interface for the user. Add all the functionalities in the frontend using React Js.

**Backend connection with UI:**

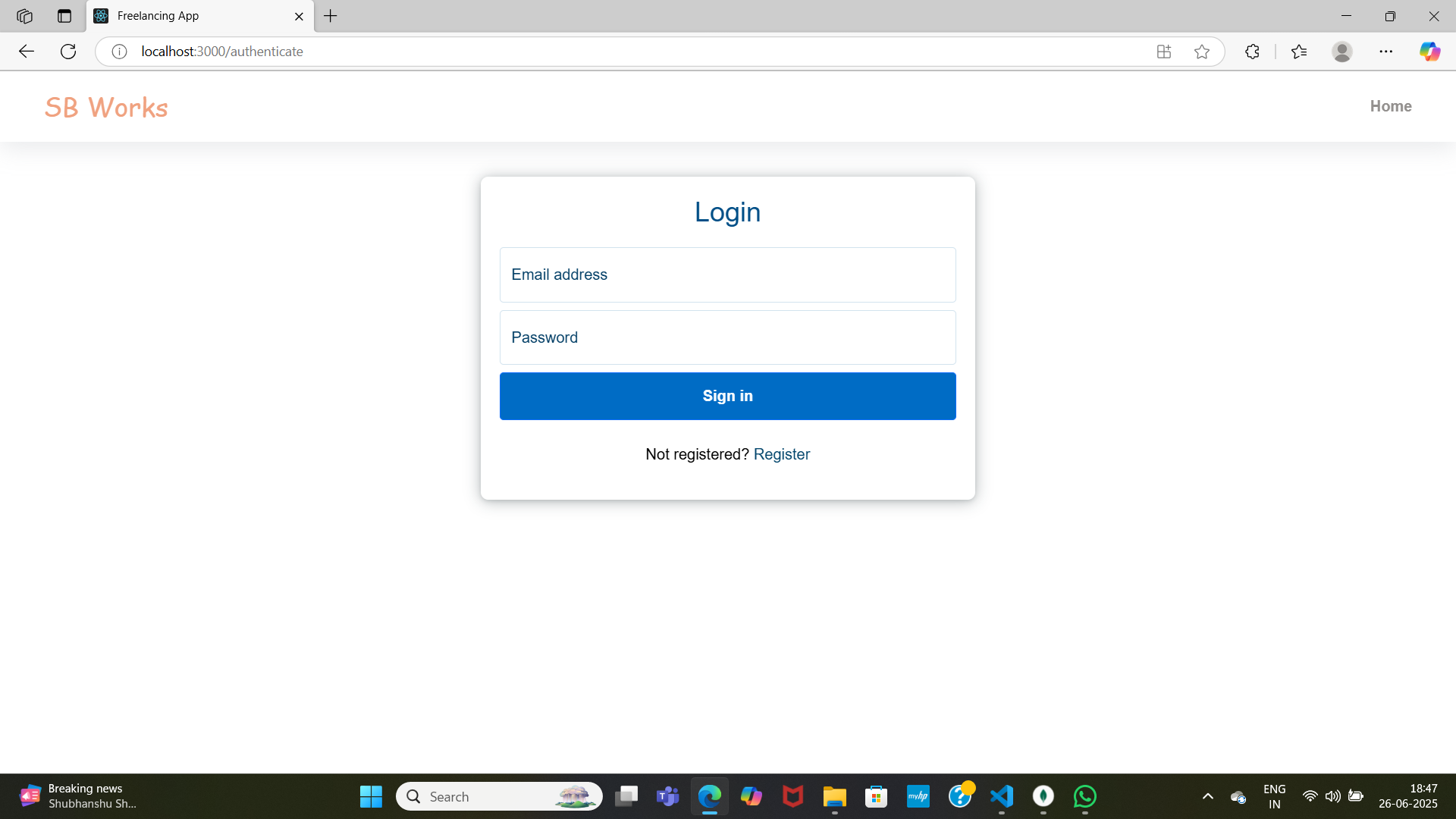
Connect the backend with the frontend to perform all the required operations. Use the middleware.

**Demo UI Images**

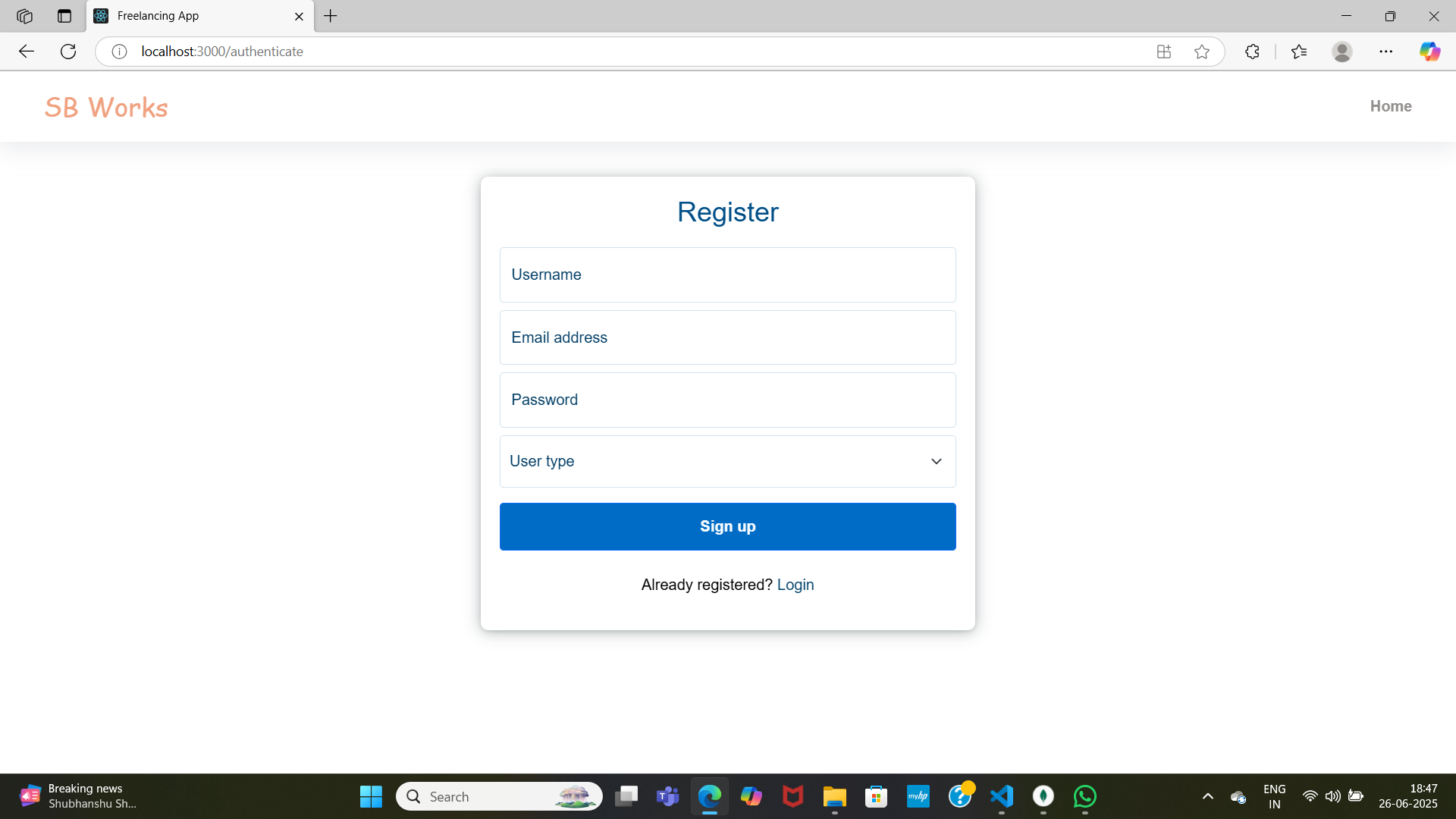
**Landing Page**



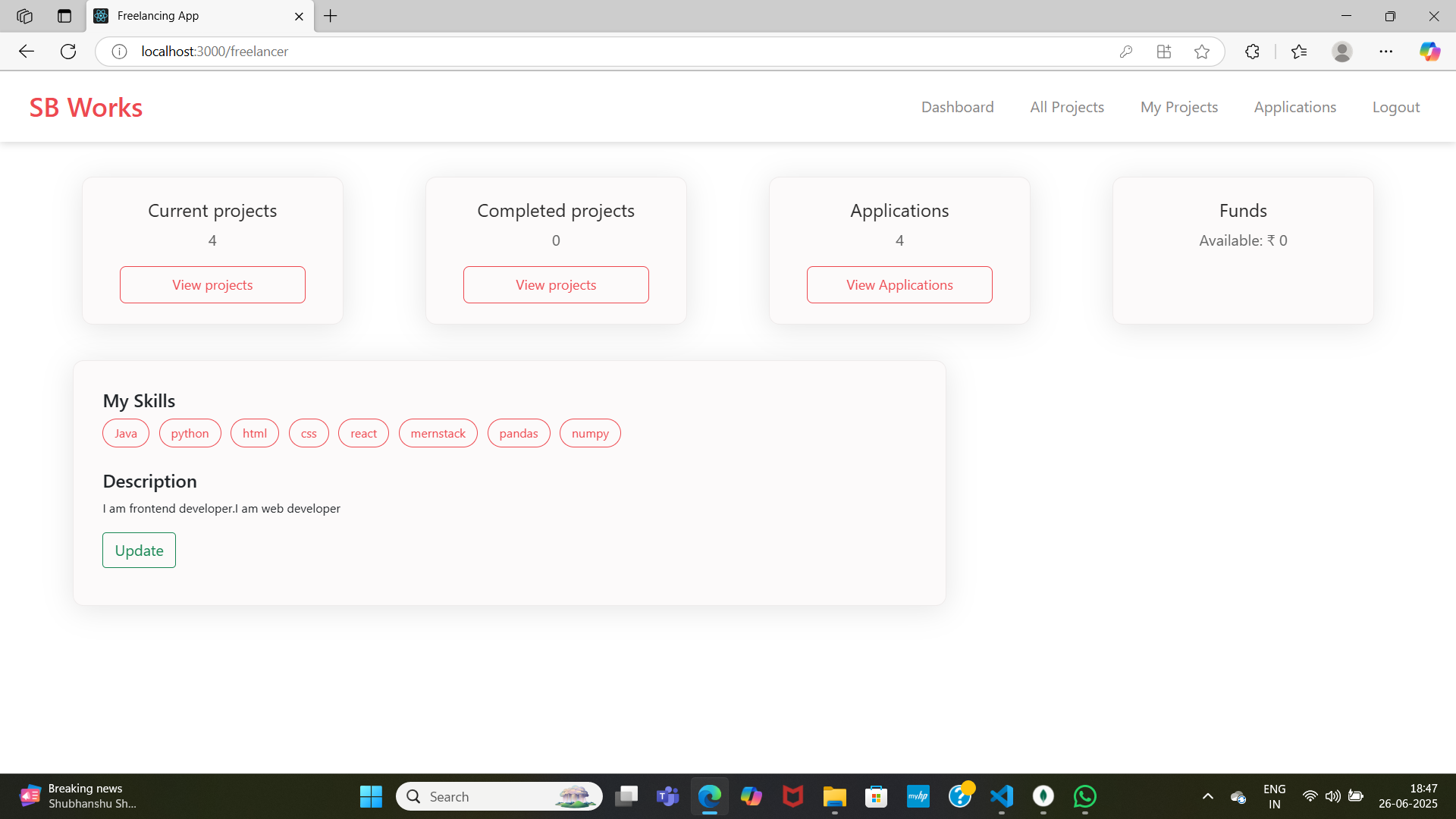
**Login Page:**

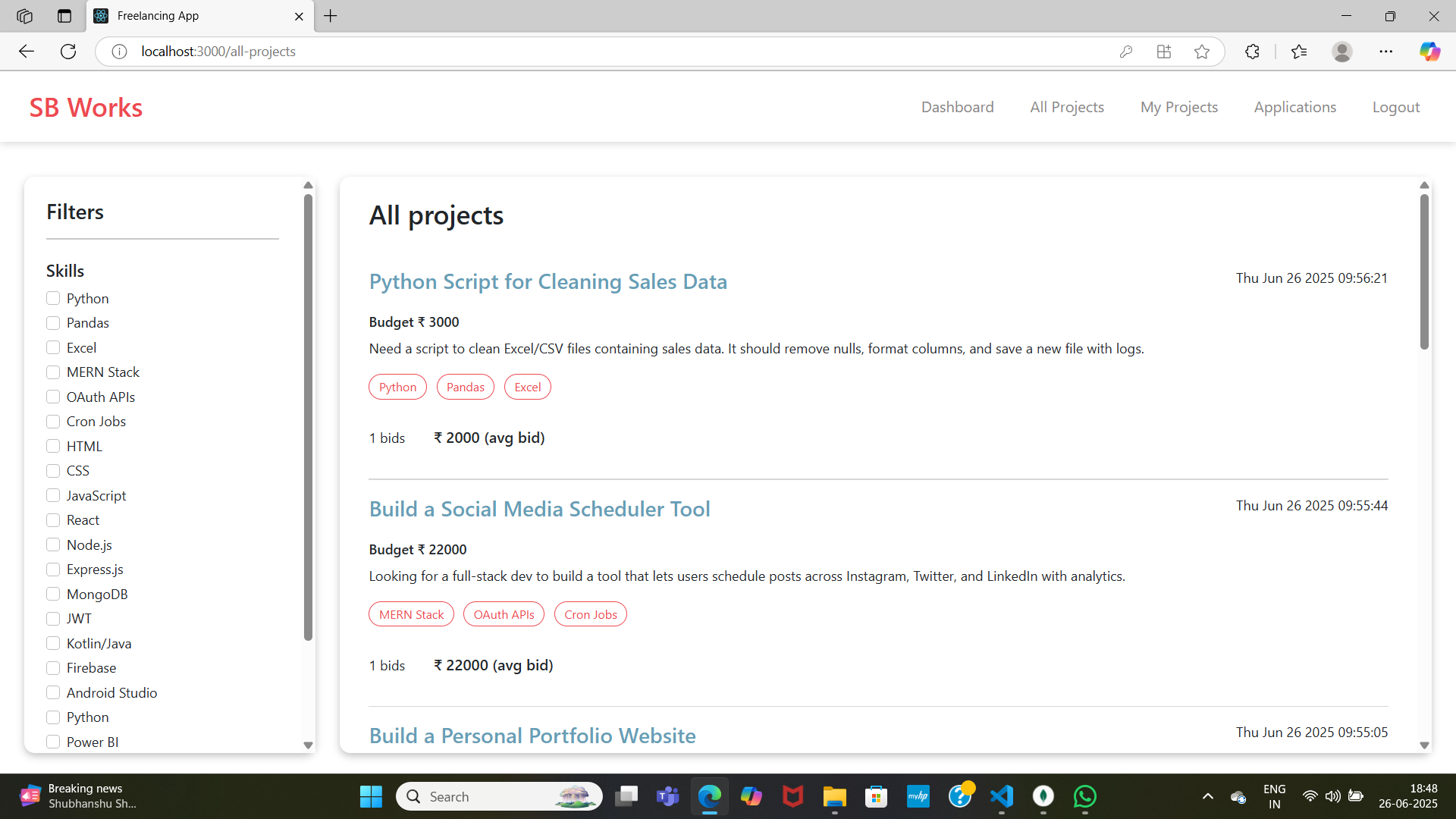


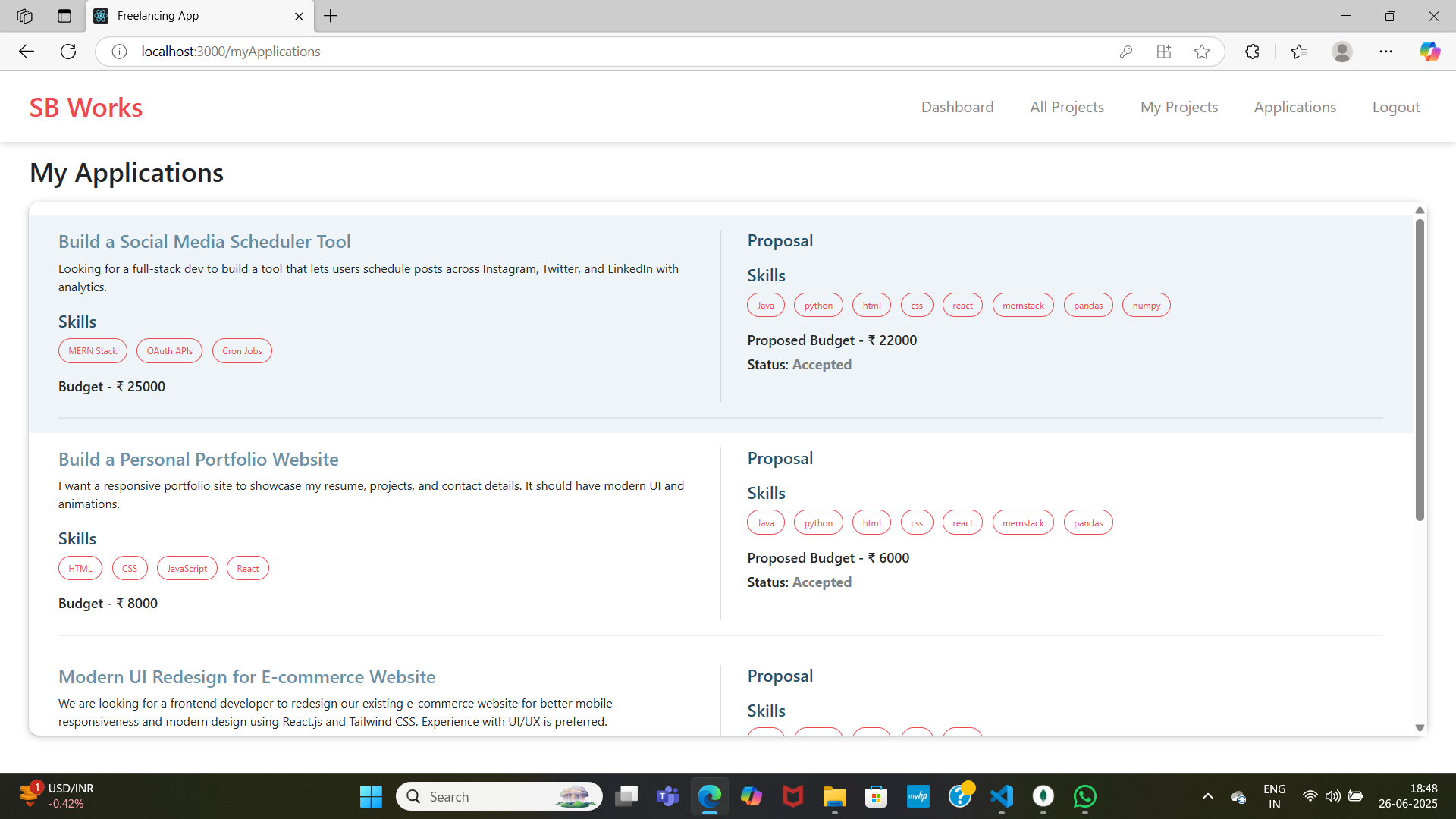
**Registration Page:**



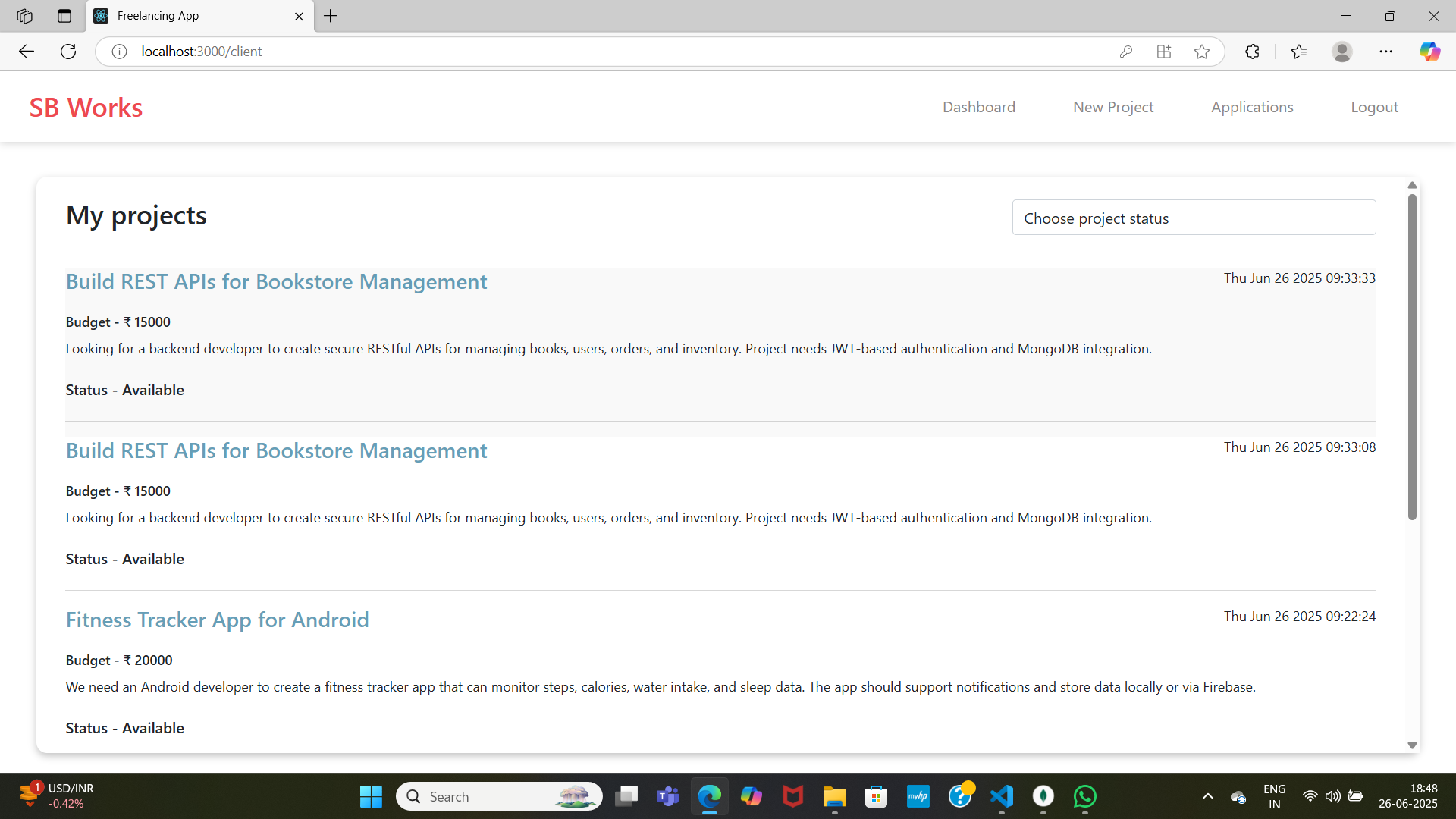
**Interface after Freelancer login:**

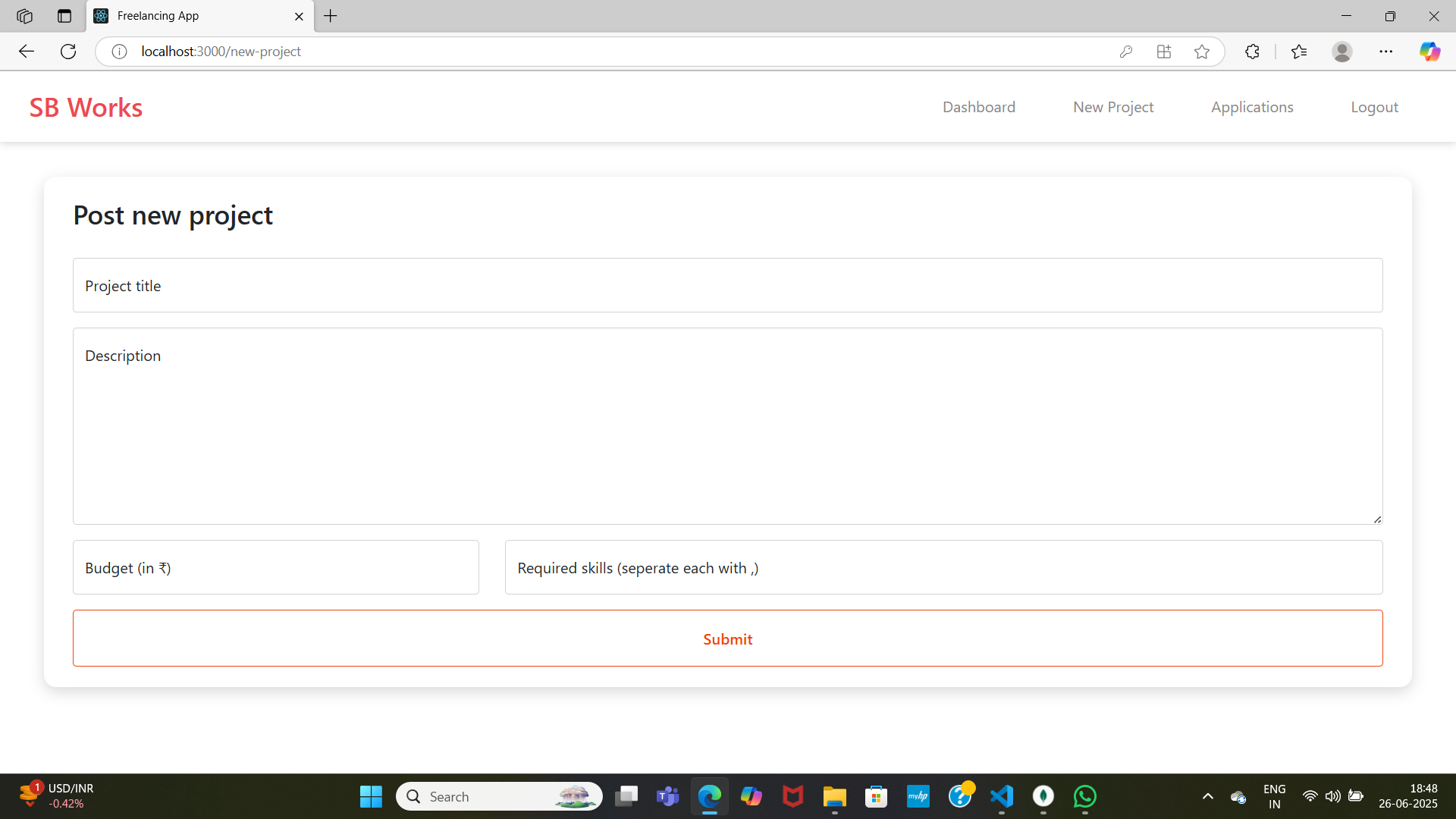
****

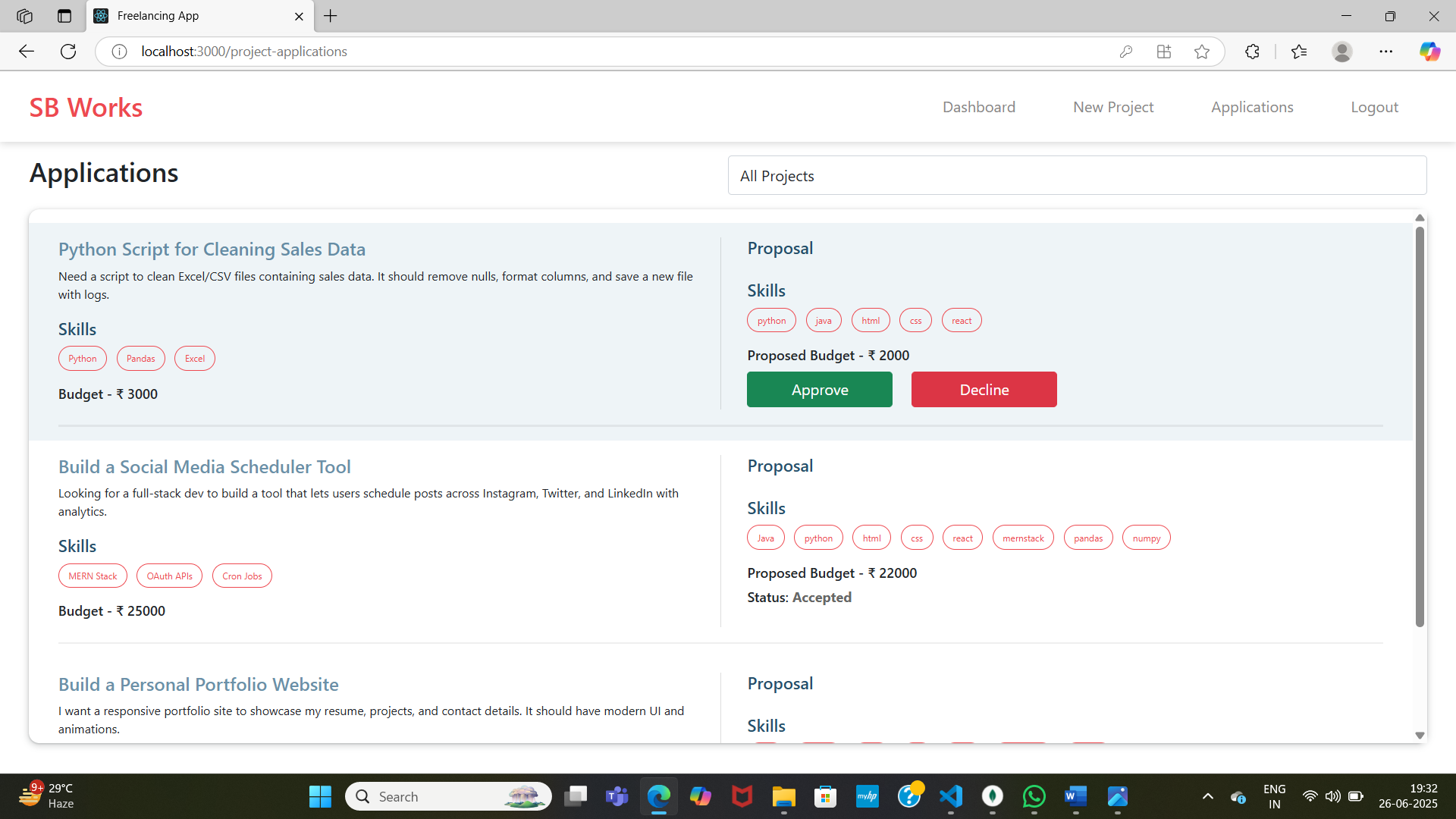
****

****

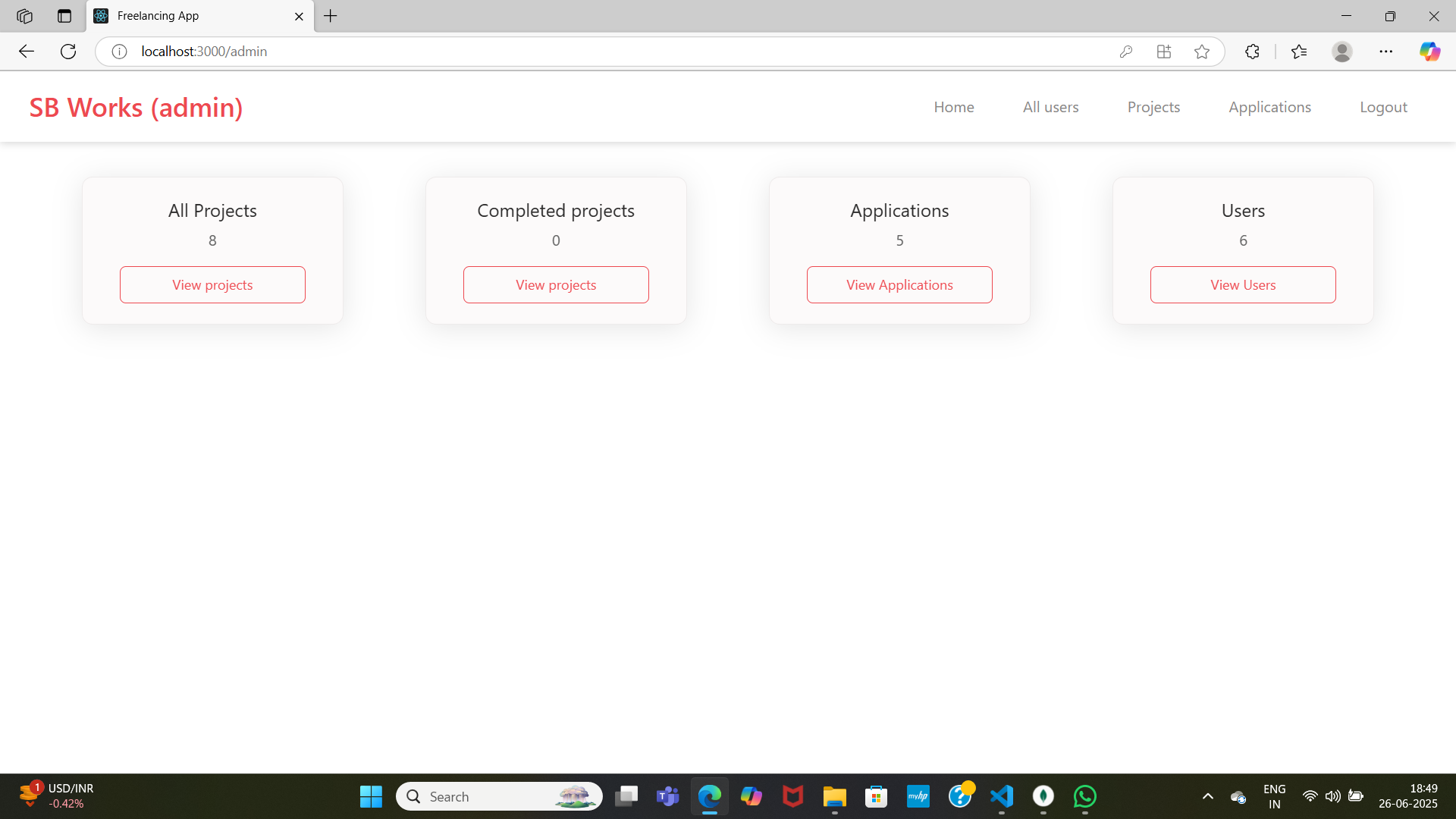
**Interface after Client Login:**

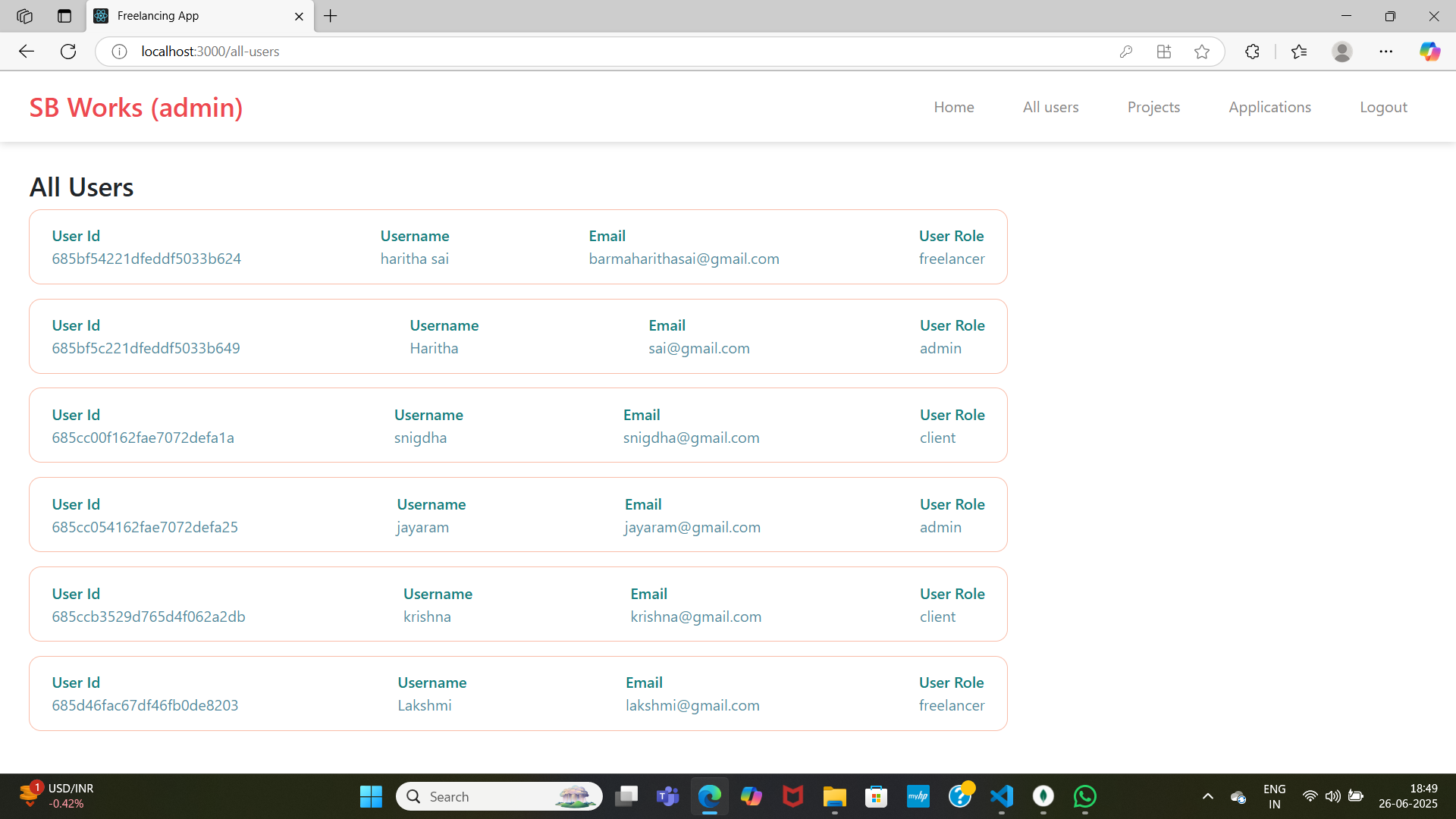
****

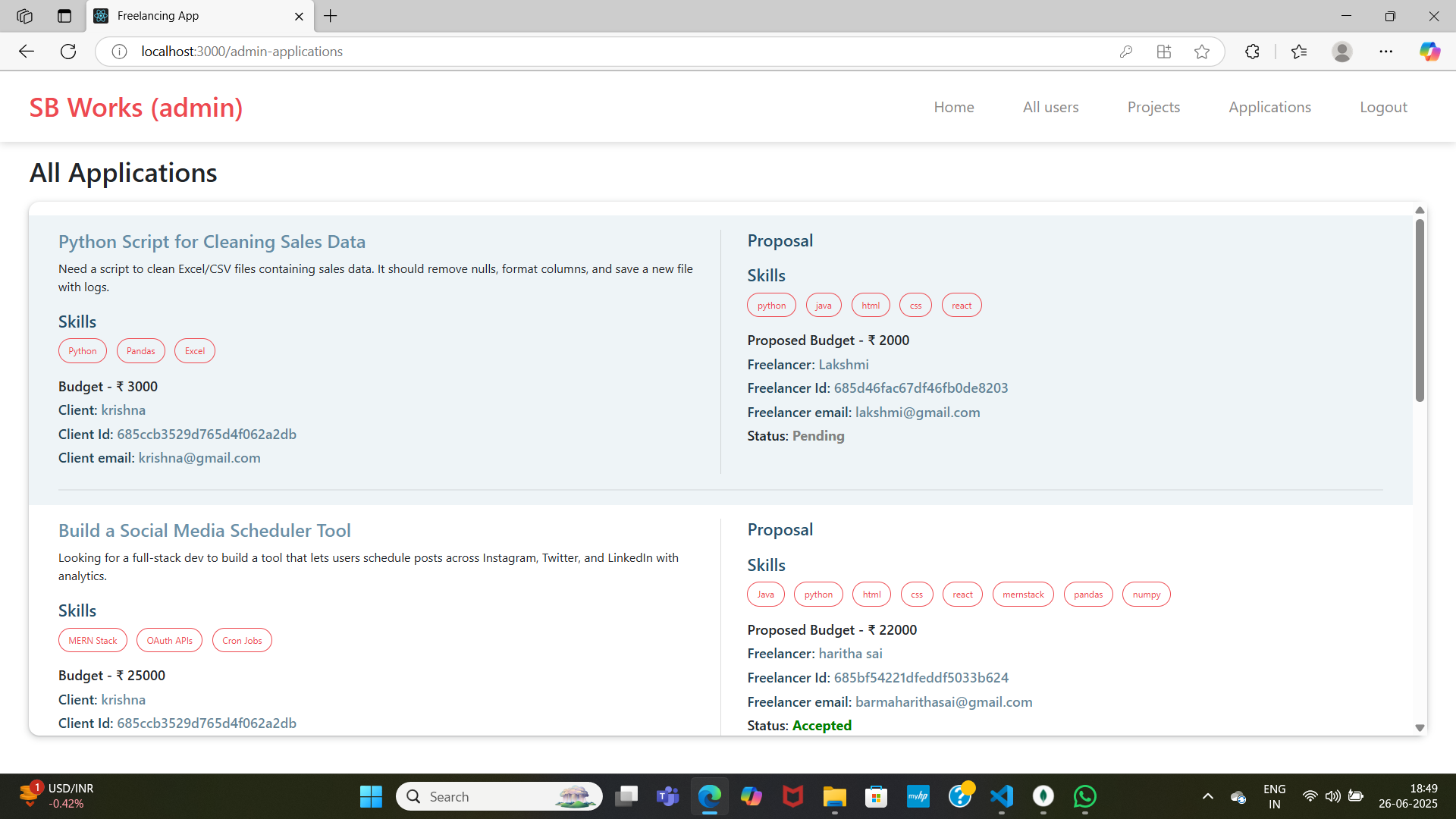
****

****

**Interface after Admin login:**

****

****



**For any further doubts or help, please consider the GitHub repo.**

https://github.com/Harithasaibarma/freelancer\_finder

**The demo of the app is available at:**

https://drive.google.com/drive/folders/1qMc5a5ohuFAnlZhHpiQUpNvX5ICipNoj?usp=sharing