Final Report: SB Works - Freelancing Platform

Team ID: LTVIP2025TMID59358

Tech Stack: MERN (MongoDB, Express.js, React.js, Node.js)

# 1. Introduction

SB Works is a freelancing platform that connects clients with skilled freelancers. It offers an intuitive interface for project posting, bidding, and streamlined collaboration. With a dedicated admin team ensuring security and smooth communication, SB Works aims to be the go-to platform for both clients and freelancers.

# 2. Scenario-Based Case Study

Sarah, a recent graduate in graphic design, discovers SB Works while searching for freelancing opportunities. She finds a project by "Sugar Rush" bakery needing a logo redesign. She confidently submits a proposal and gets selected. Through SB Works' integrated chat, they collaborate. Upon final submission, Sarah receives positive feedback and builds her freelance portfolio further on the platform.

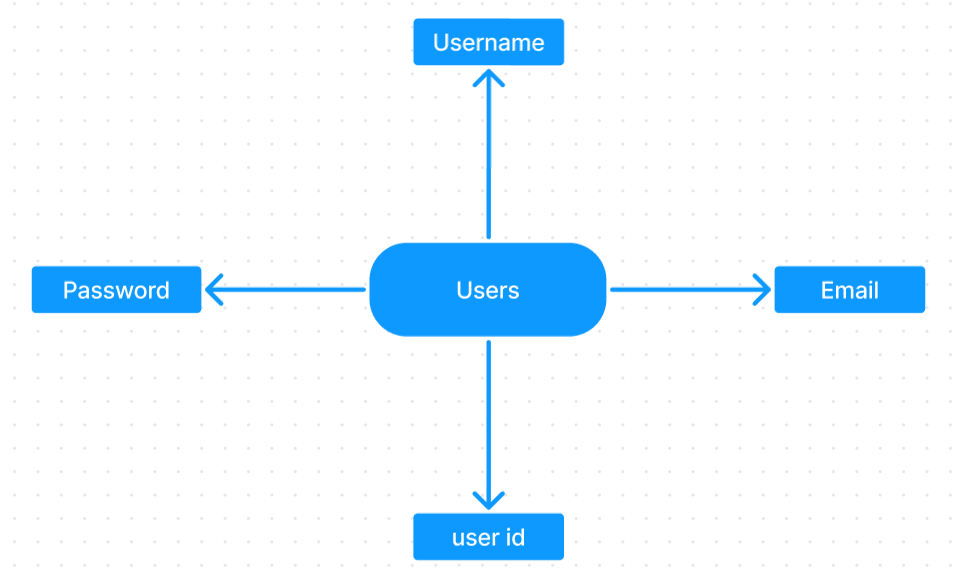
# 3. Technical Architecture

SB Works follows a client-server architecture. The frontend, built in React.js, utilizes Bootstrap and Material UI. Axios handles HTTP communication with the Express.js backend, which connects to MongoDB via Mongoose. The application supports RESTful APIs for user, project, and communication management.

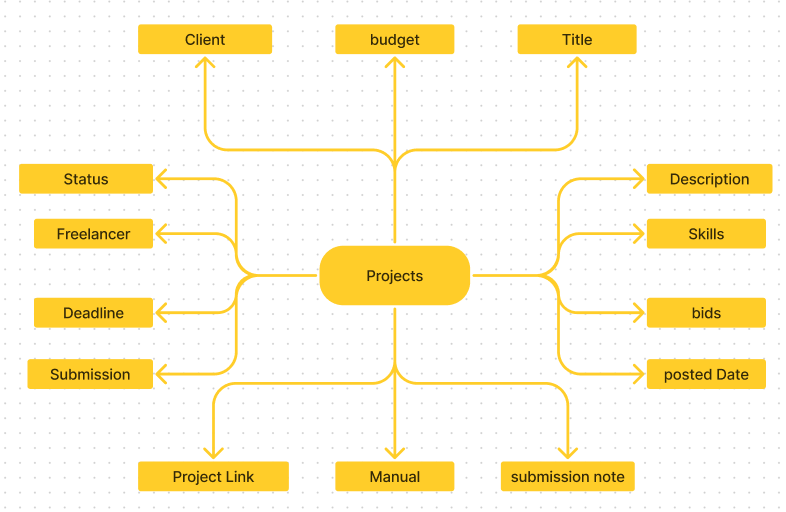
# 4. ER Diagram

The platform includes the following entities:

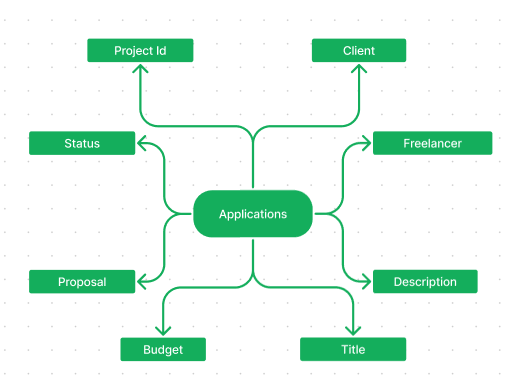
* - User (Client/Freelancer)



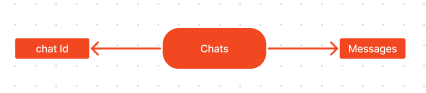
* - Project



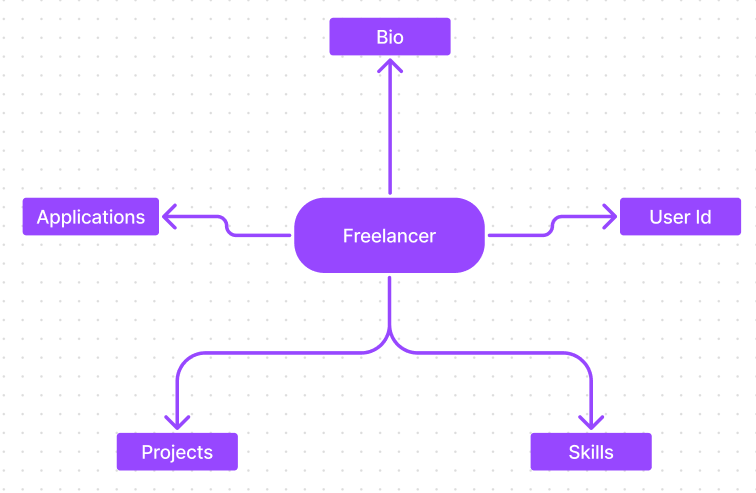
* - Application (Proposal)



* - Chat



* - Admin



# 5. Project Structure

Frontend: Built using React.js with modular components for user profiles, projects, chat system, and dashboards.

**1. Setting the Stage:**

The SB Works frontend thrives on React.js. To get started, we'll:

* Create the initial React application structure.
* Install essential libraries for enhanced functionality.
* Organize project files for a smooth development experience.
* This solid foundation ensures an efficient workflow as we bring the SB Works interface to life.

**2. Crafting the User Experience:**

Next, we'll focus on the user interface (UI). This involves:

* Designing reusable UI components like buttons, forms, and project cards.
* Defining the layout and styling for a visually appealing and consistent interface.
* Implementing navigation elements for intuitive movement between features.
* These steps will create a user-friendly experience for both freelancers and clients.

**3. Bridging the Gap:**

The final stage connects the visual interface with the backend data. We'll:

* Integrate the frontend with SB Works' API endpoints.
* Implement data binding to ensure dynamic updates between user interactions and the displayed information.

This completes the frontend development, bringing the SB Works platform to life for users.

Reference video: <https://drive.google.com/file/d/16o7iRxHfaKKHJkep1ixKs86C3nfNACVQ/view?usp=sharing>  
Backend: Node.js with Express.js handles API routing and server-side logic.

**1.Project Setup:**

* Create a project directory and initialize it using npm init.
* Install required dependencies like Express.js, Mongoose, body-parser, and cors.

**2. Database Configuration:**

* Set up a MongoDB database (locally or using a cloud service like MongoDB Atlas).
* Create collections for:
* Users (storing user information, account type)
* Projects (project details, budget, skills required)
* Applications (freelancer proposals, rate, portfolio link)
* Chat (communication history for each project)
* Freelancer (extended user details with skills, experience, ratings)

**3. Express.js Server:**

* Create an Express.js server to handle HTTP requests and API endpoints.
* Configure body-parser to parse request bodies and cors for cross-origin requests.

**4. API Routes:**

* Define separate route files for user management, project listing, application handling, chat functionality, and freelancer profiles.
* Implement route handlers using Express.js to interact with the database:
* User routes: registration, login, profile management.
* Project routes: project creation, listing, details retrieval.
* Application routes: submit proposals, view applications.
* Chat routes: send and receive messages within projects.
* Freelancer routes: view and update profiles, showcase skills.

**5. Data Models:**

* Define Mongoose schemas for each data entity:
* User schema
* Project schema
* Application schema
* Chat schema
* Freelancer schema (extends User schema with skills, experience)
* Create Mongoose models to interact with the MongoDB database.
* Implement CRUD operations for each model to manage data.

**6. User Authentication:**

* Implement user authentication using JWT or session-based methods.
* Create routes and middleware for user registration, login, and logout.
* Use authentication middleware to protect routes requiring user authorization (e.g., applying for projects).

**7. Project Management:**

* Allow clients to post projects with details and budget.
* Enable freelancers to browse projects, search by skills, and submit proposals.
* Implement a system for clients to review applications and choose freelancers.

**8. Secure Communication & Collaboration:**

* Integrate a secure chat system within projects for communication between clients and freelancers.
* Allow file attachments and feedback exchange to facilitate collaboration.

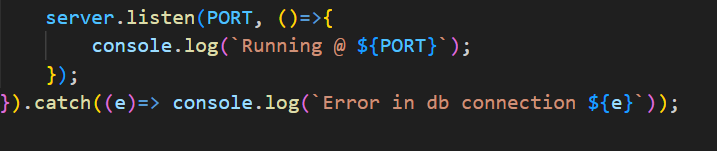
**9. Admin Panel (Optional):**

* Implement an admin panel with functionalities like:
* Managing users
* Monitoring project updates and applications
* Accessing transaction history

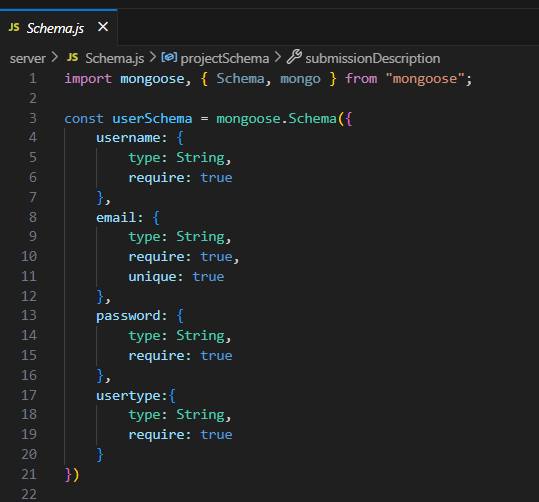
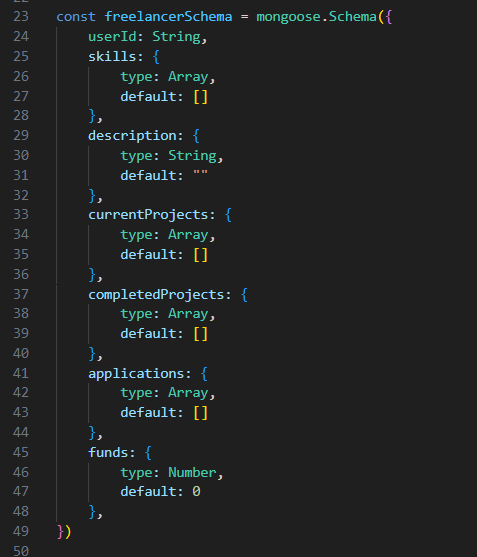
Reference video: <https://drive.google.com/file/d/1zrOMSp6svjH1tRcul3b442XVPNyKTSp4/view?usp=sharing>  
Database: MongoDB stores user, project, chat, and application data.

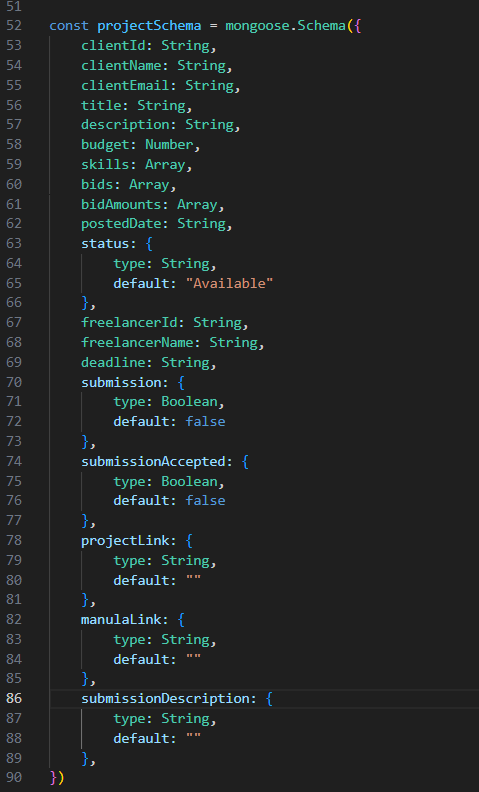
* Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
* Create a database and define the necessary collections for users, freelancer, projects, chats, and applications.
* Connect the database to the server with the code provided below.

****

****

The Schemas for the database are given below

**  **

****

****

Security: JWT-based authentication and role-based access control.

* **Folder setup:**

Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

* Client folders.
* Server 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

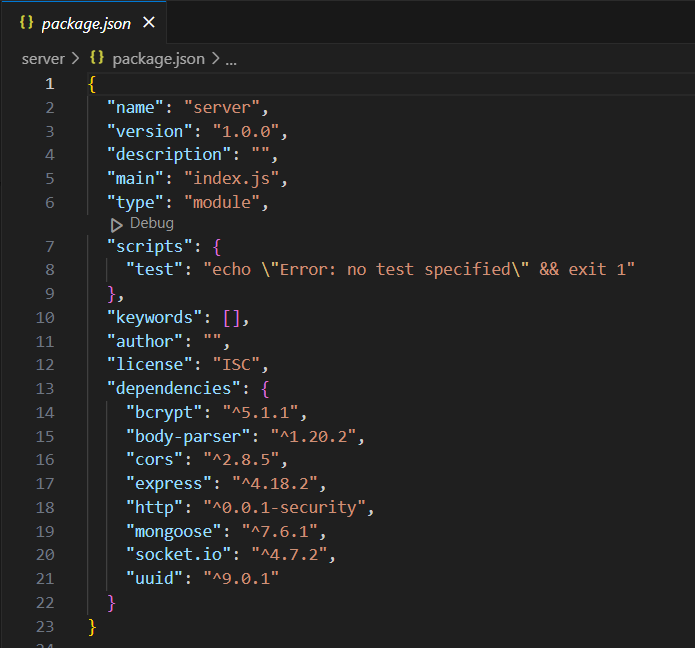
For backend, we use:

* Express Js
* Node JS
* MongoDB
* Mongoose
* Cors
* Bcrypt

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.



# 6. Prerequisites

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: <https://nodejs.org/en/download/package-manager/>

?**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: <https://docs.mongodb.com/manual/installation/>

?**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: <https://reactjs.org/docs/create-a-new-react-app.html>

?**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.

?**Database Connectivity**: 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 <https://code.visualstudio.com/download>

To run the existing Freelancer App project downloaded from Drive:

Use the code:

Drive link: <https://drive.google.com/drive/folders/10mSn2lMTaVMDWWFNjeJjiOLfmcD3-87C?usp=sharing>

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 [http://localhost:3000](http://localhost:3000/)  
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.

# 7. Application Flow

**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.

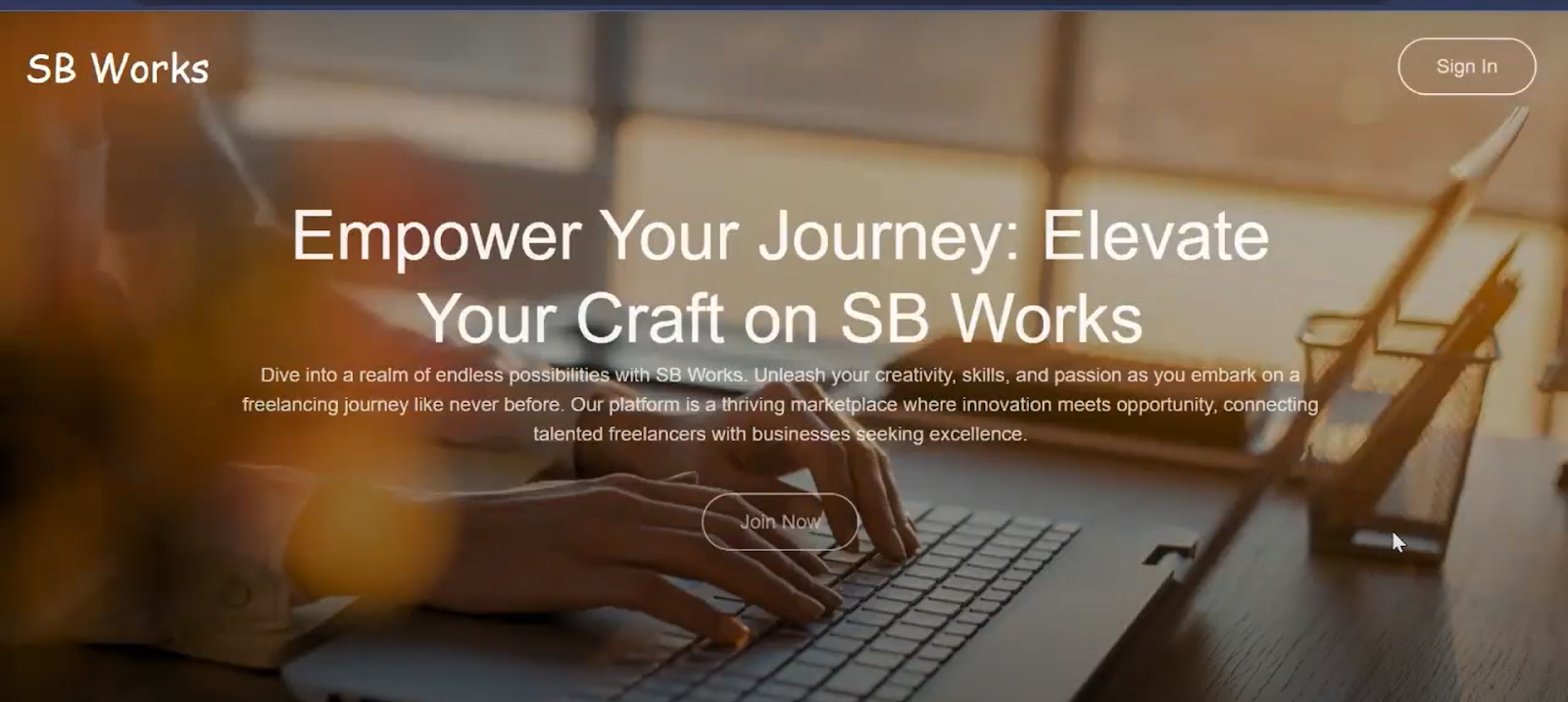
**Admin Responsibilities:**

* 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

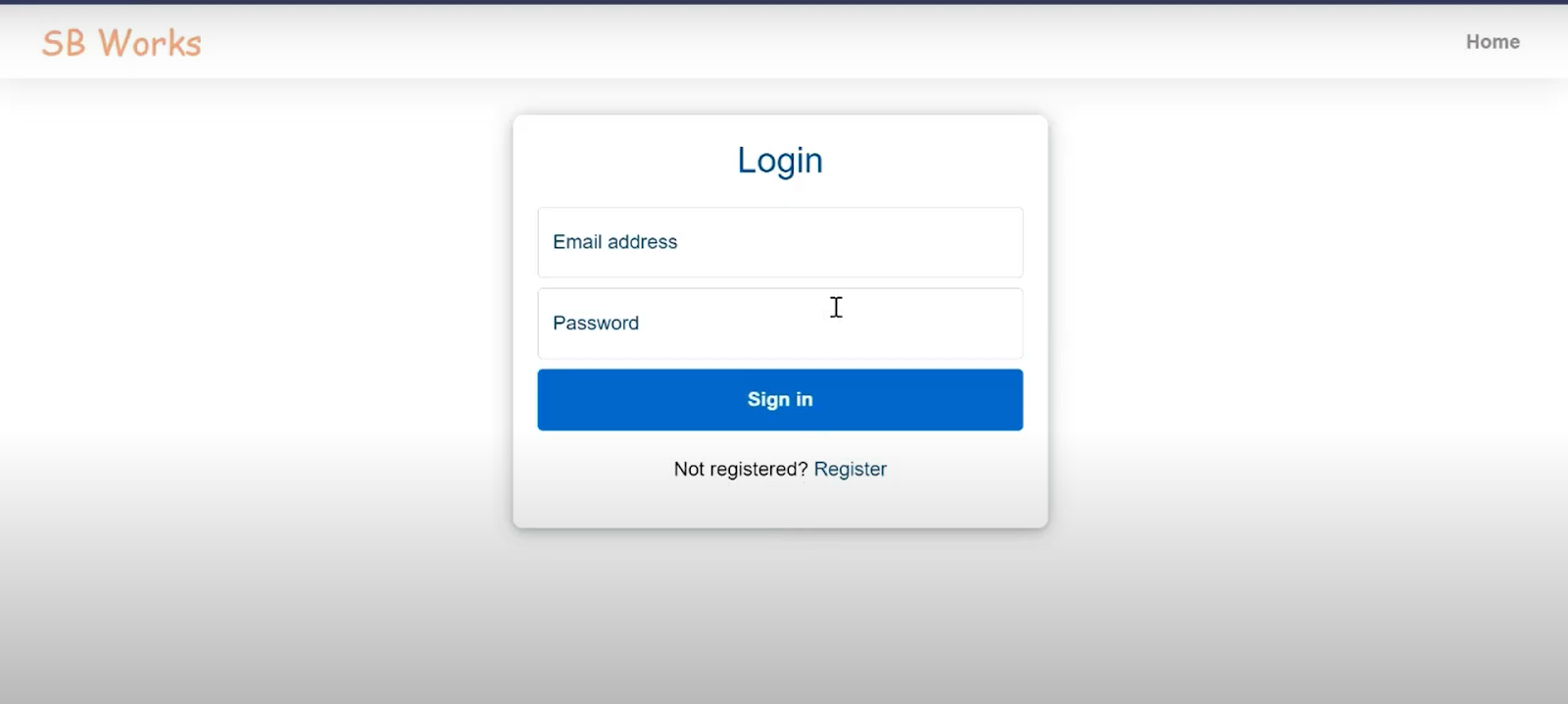
# 8. Project Milestones

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the images provided below.

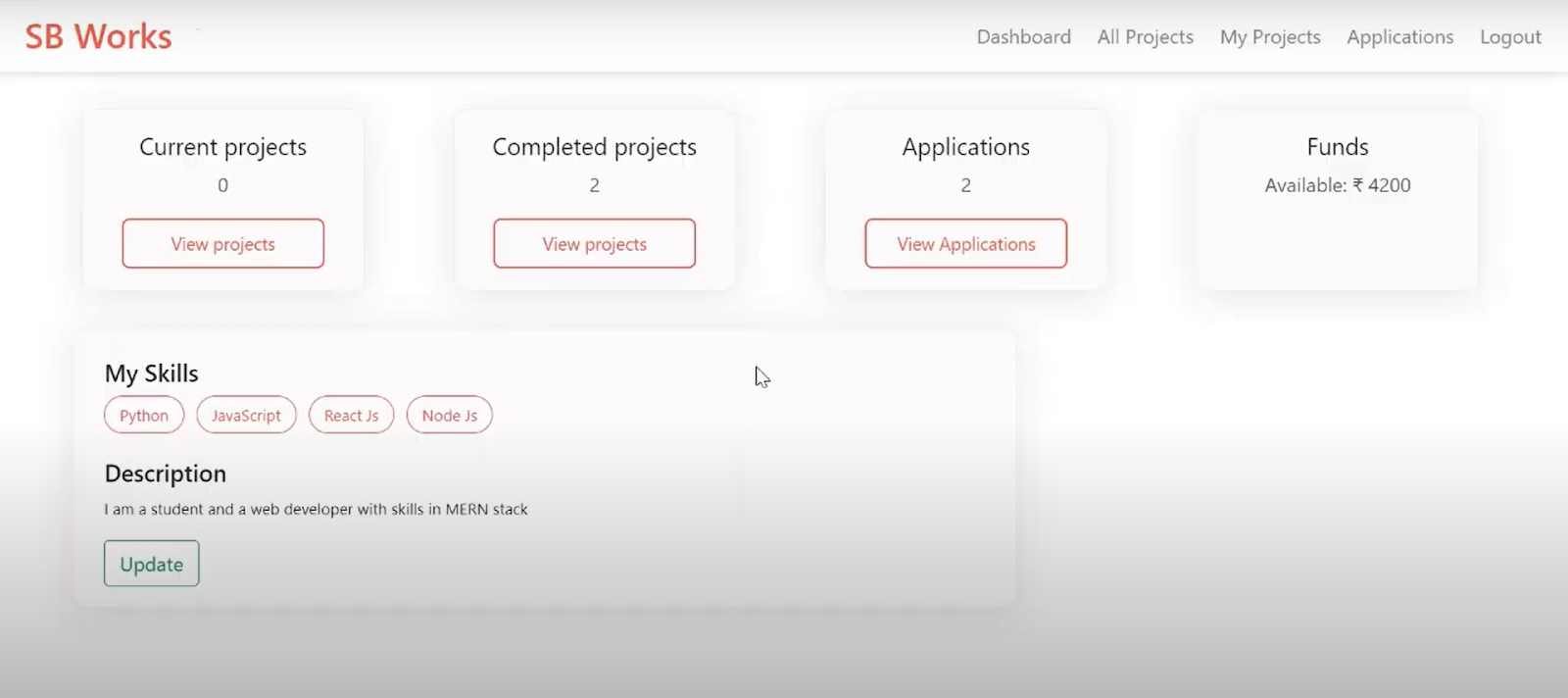
**Landing page:**

****

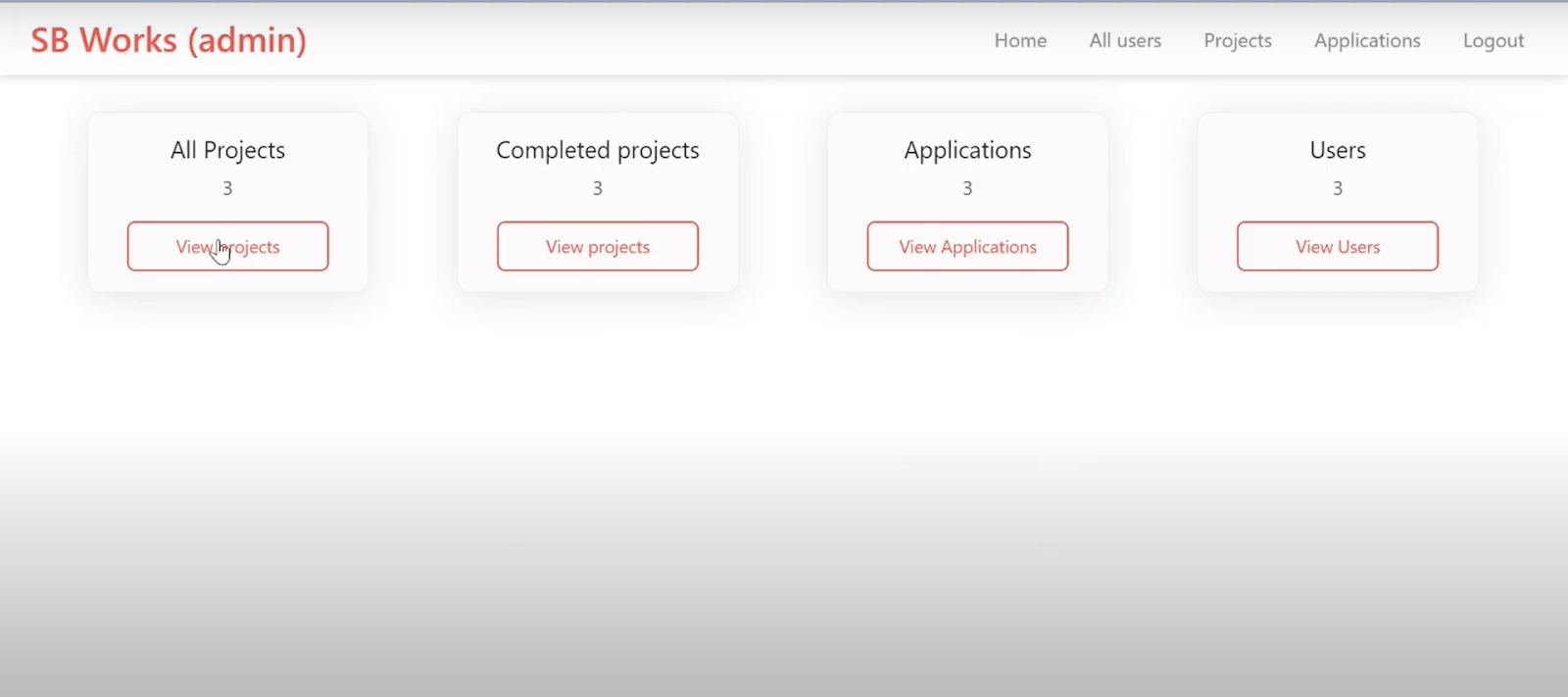
**Authentication:**

****

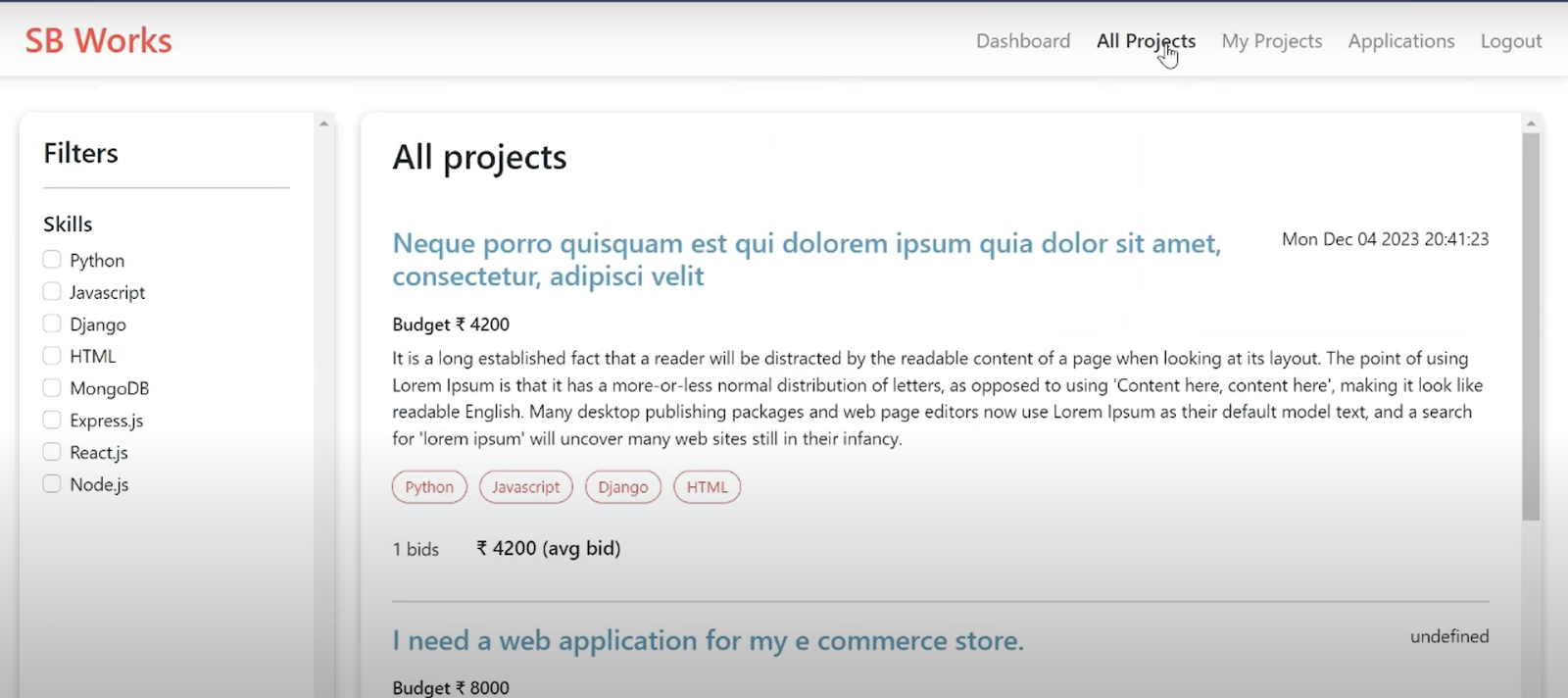
**Freelancer dashboard:**

****

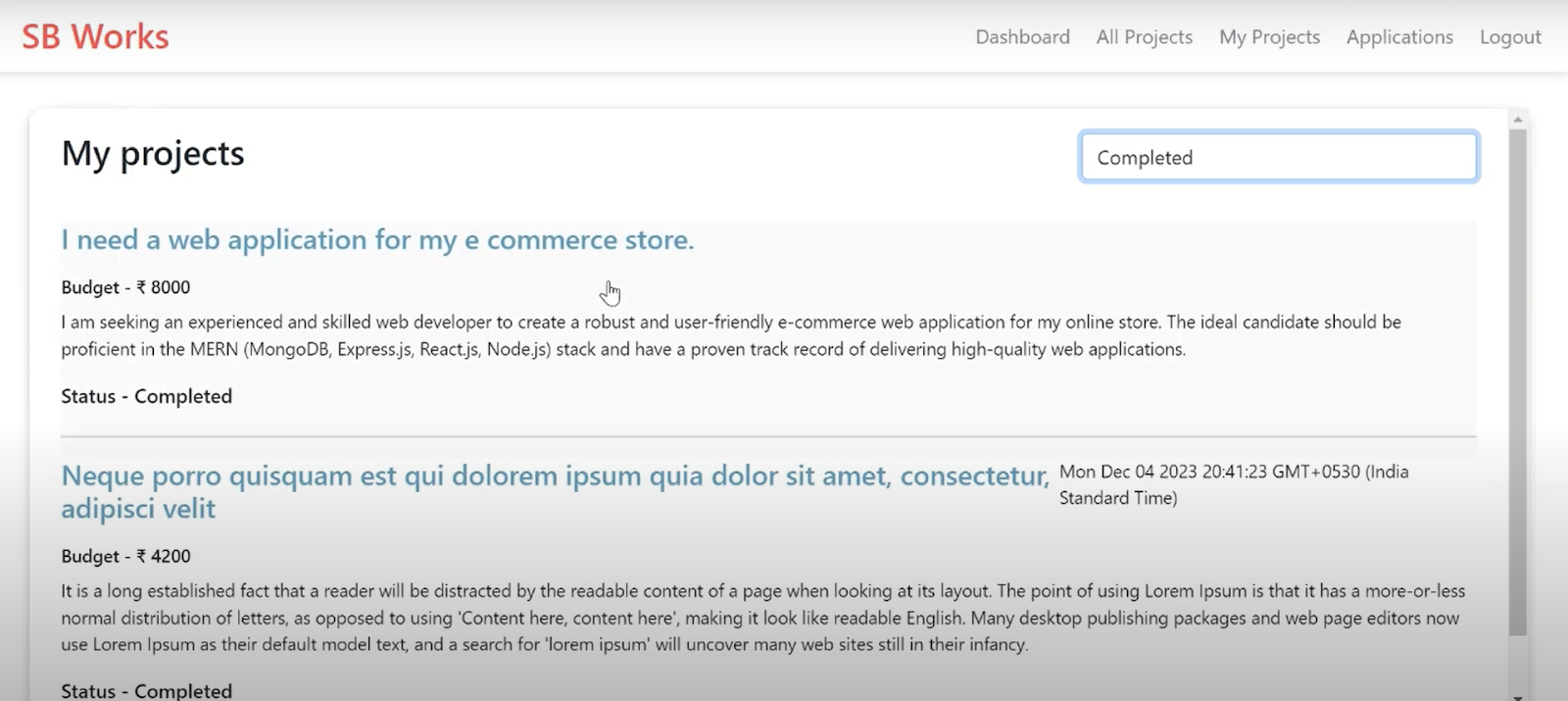
**Admin dashboard:**

****

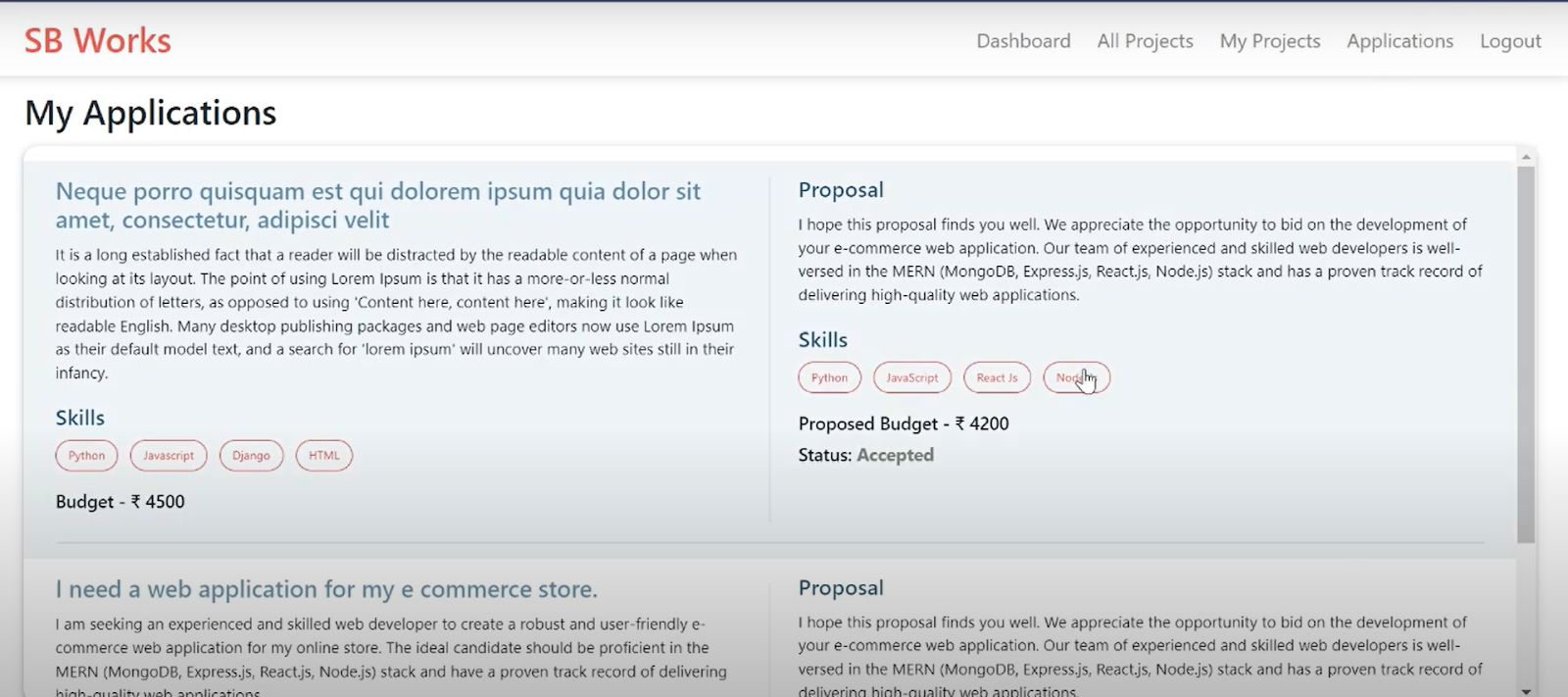
**All projects:**

****

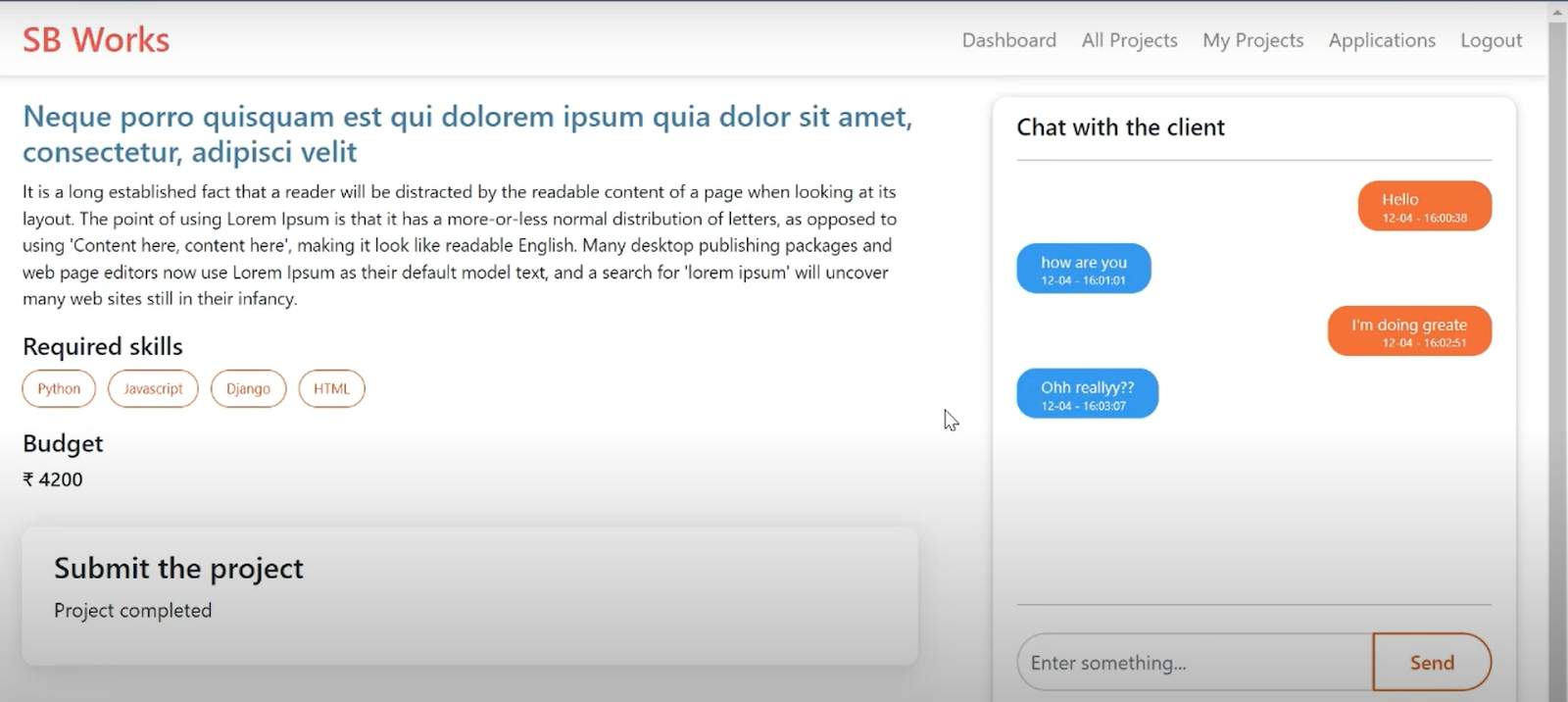
**Freelance projects:**

****

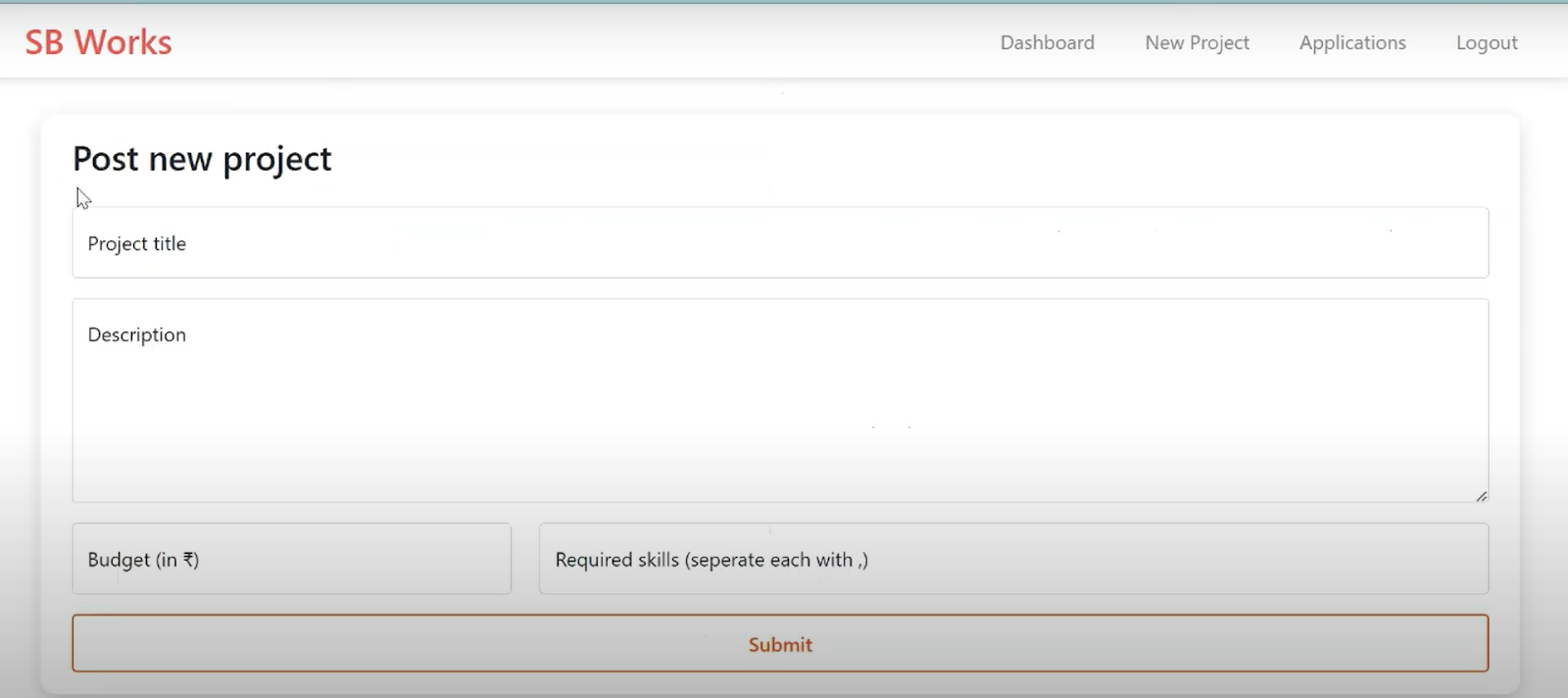
**Applications:**

****

**Project page:**

****

**New project:**

****

# 9. Git Hub link

# 10. Demo link

<https://drive.google.com/file/d/1erdcudF8D00QyHEf0aMKioTAqWa2AjDb/view?usp=sharing>

# 11. Conclusion

SB Works creates a reliable ecosystem for freelancing by integrating modern technologies with an intuitive interface. It ensures efficient communication, transparent project handling, and secure data management, empowering freelancers and clients alike.