# **ProLancer -** Freelancing Platform Project Report

### 1. Project Overview

ProLancer is a full-stack freelancing platform where clients can post projects and freelancers can apply, chat in real-time, and submit completed work. The platform supports user authentication, project management, real-time chat using Socket.IO, and submission tracking.

### 2. Features Implemented

* User registration and login (Client/Freelancer)
* Project posting and application system
* Real-time chat with Socket.IO
* Freelancer application approval by clients
* Project status updates (Ongoing -> Completed)
* File submission system for completed work
* Project filtering by status
* Skill tagging and budget management

### 3. Tech Stack

* Frontend: React.js, Tailwind CSS
* Backend: Node.js, Express.js
* Database: MongoDB (with Mongoose)
* Realtime: Socket.IO- Authentication: JWT + LocalStorage

### 4. Architecture Diagram

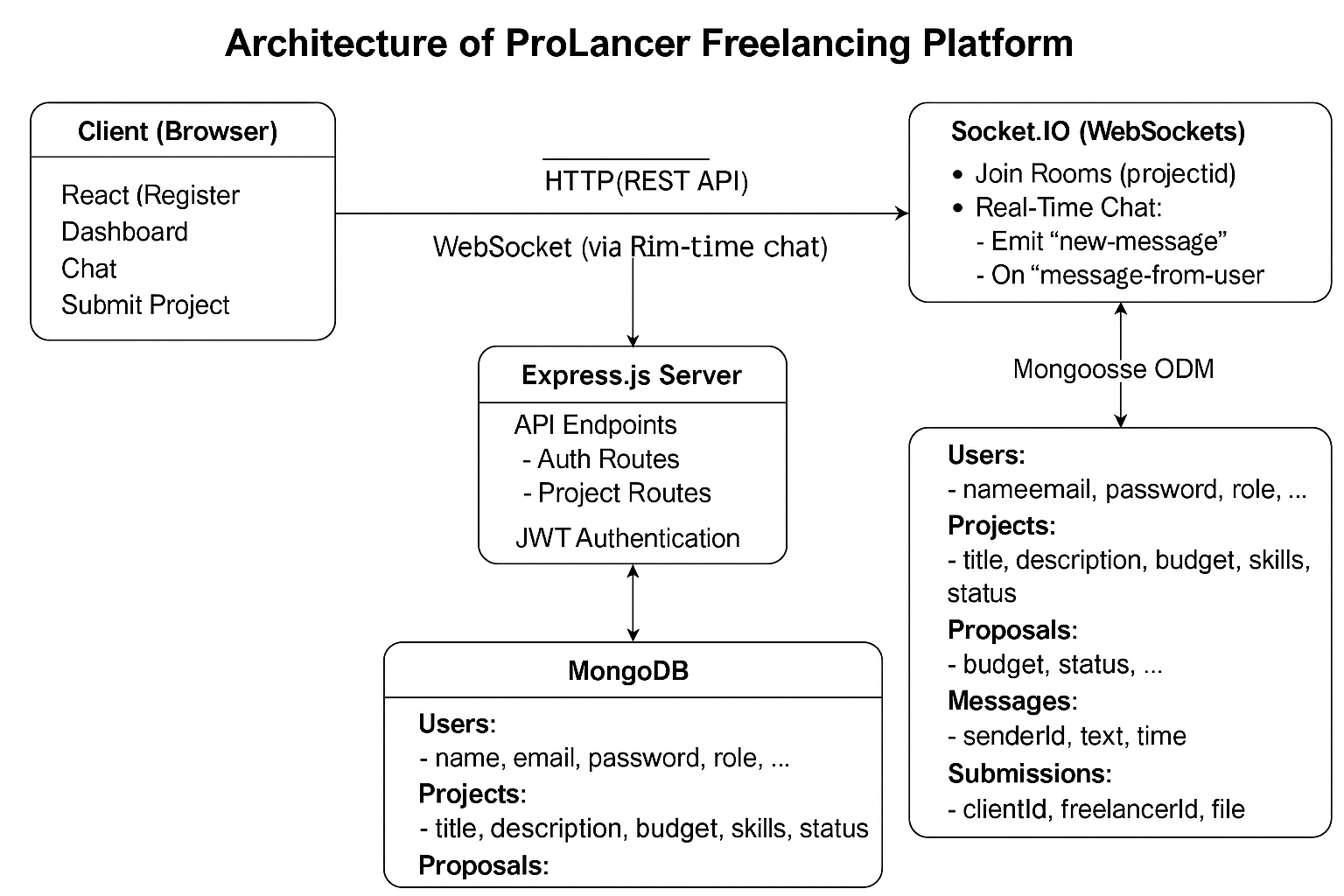
The architectural diagram of the **ProLancer Freelancing Platform** shows how the different components communicate to provide a seamless user experience.

The **frontend**, built with **React.js**, is responsible for the user interface. It interacts with the **backend** (developed using **Node.js** and **Express.js**) using both **REST APIs** and **WebSocket connections** (via **Socket.IO**).

* **REST APIs** are used for standard operations like user registration/login, project posting, and data fetching.
* **Socket.IO** handles **real-time messaging**, allowing clients and freelancers to chat instantly. Each project chat is organized using Socket.IO rooms based on projectId.

The **backend** processes requests, handles authentication using **JWT tokens**, and manages business logic for projects, applications, and messaging. It also updates and retrieves data from the **MongoDB** database using **Mongoose**, which provides a structured way to define and query the data.

Overall, this architecture ensures both secure and responsive interactions between users, supports real-time communication, and maintains data integrity across all components.

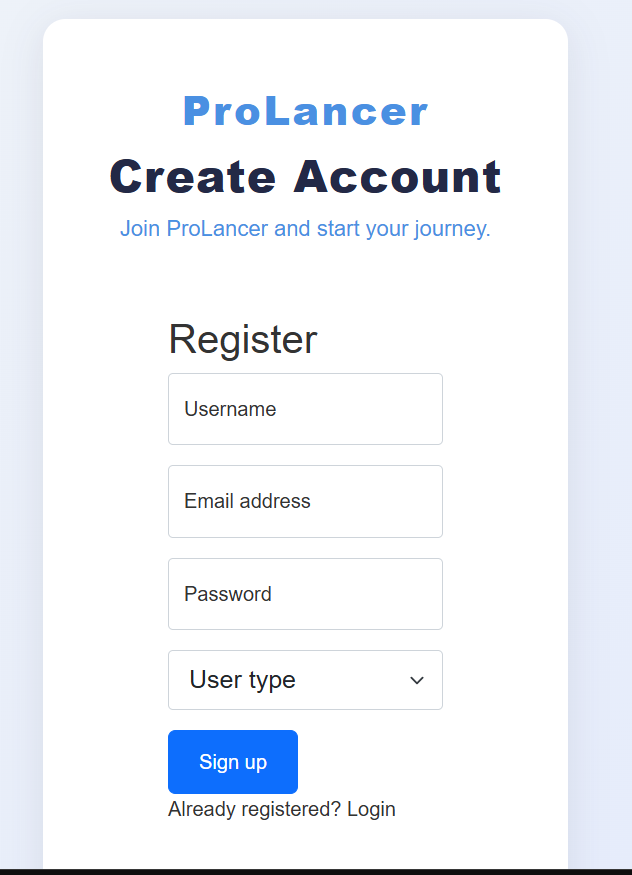


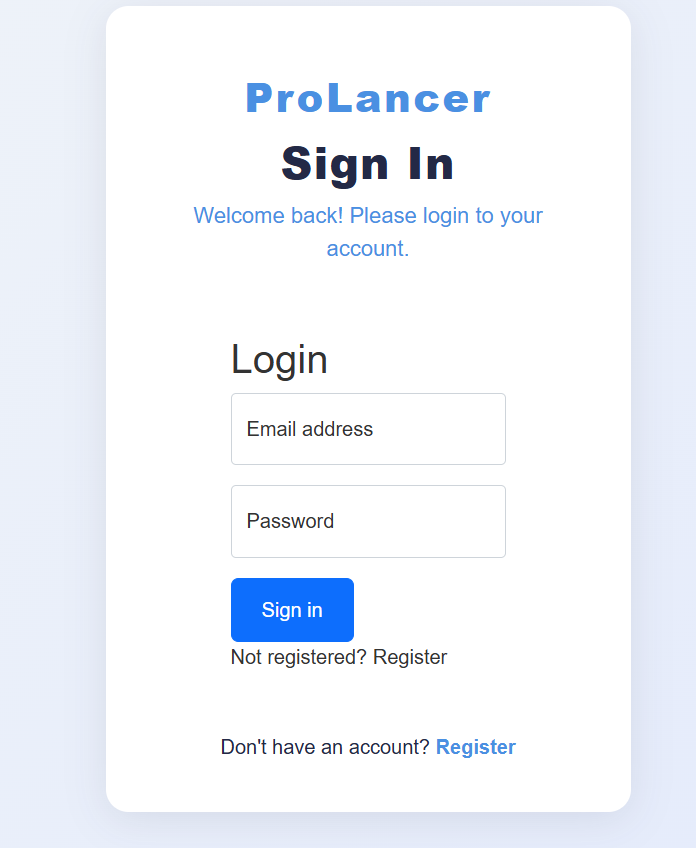
### 5. Screenshots

#### Landing Page

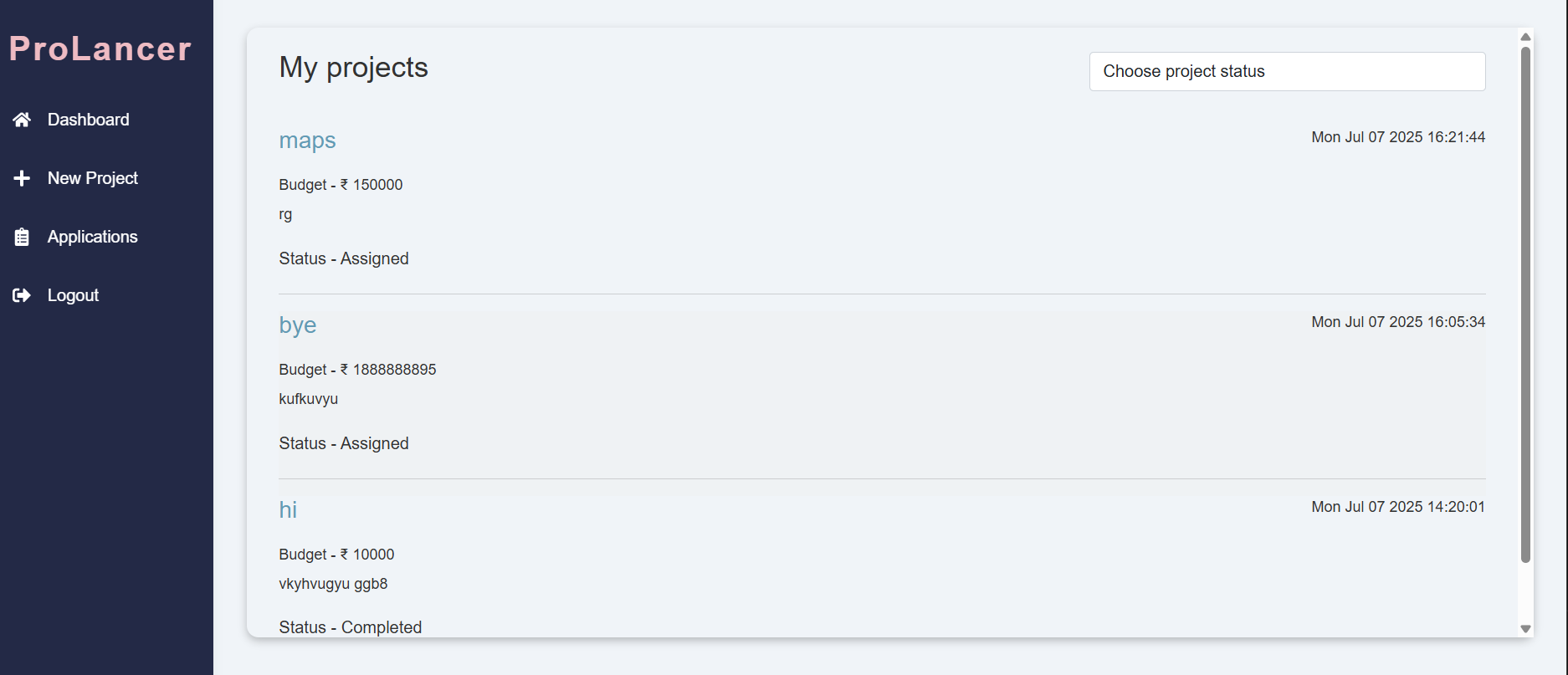
### 

#### Sign in Page

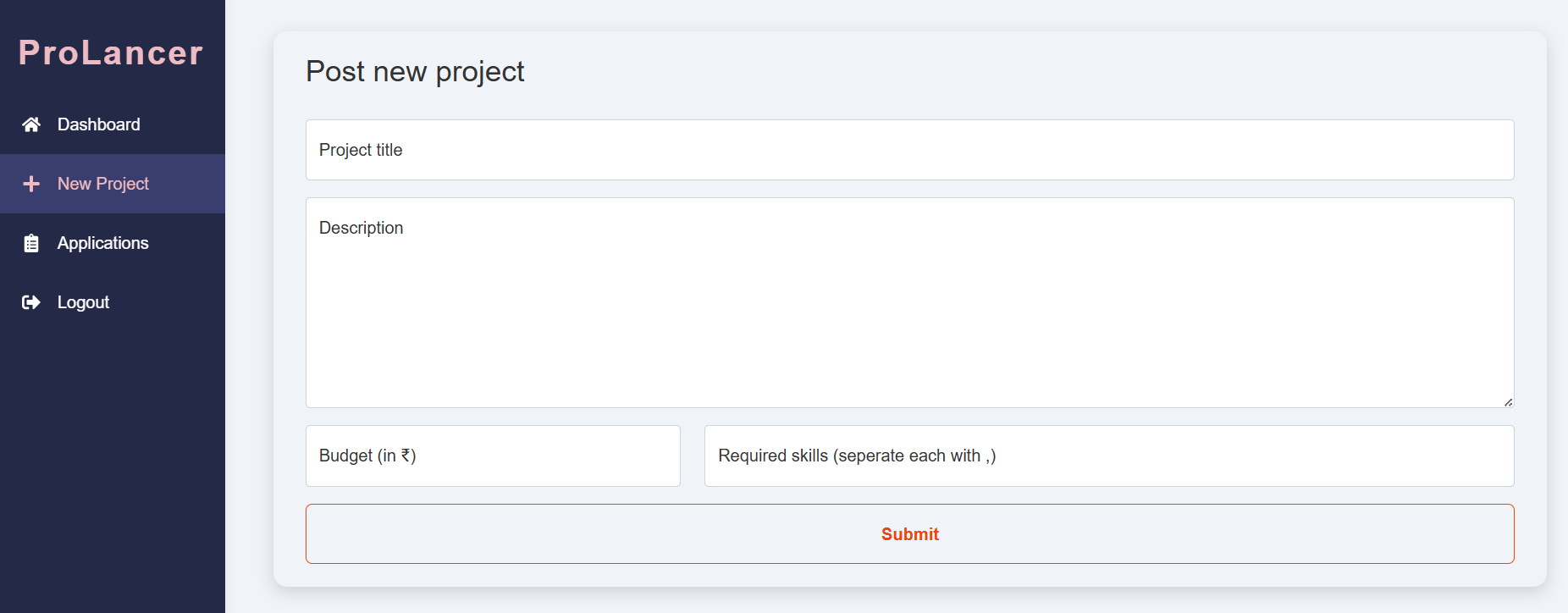




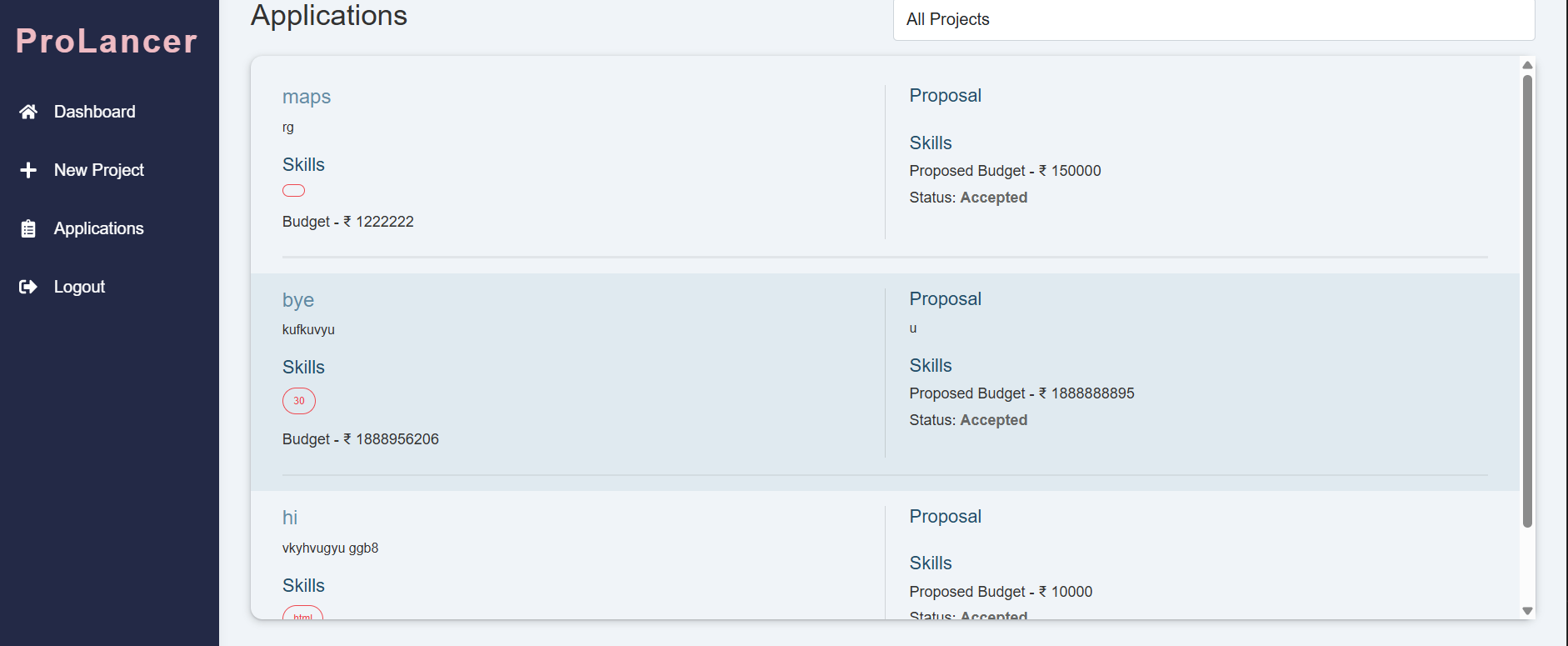
#### Client Dashboard



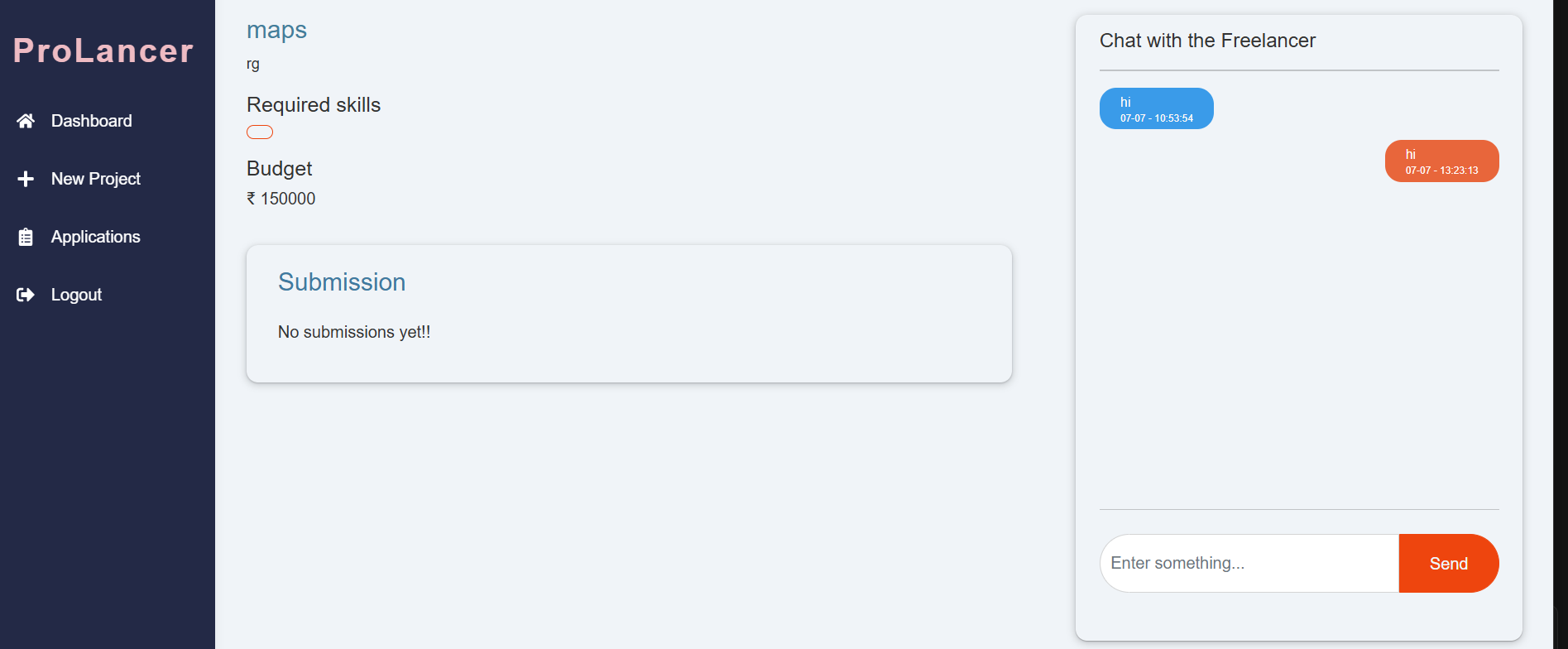
#### Client Posting new project page



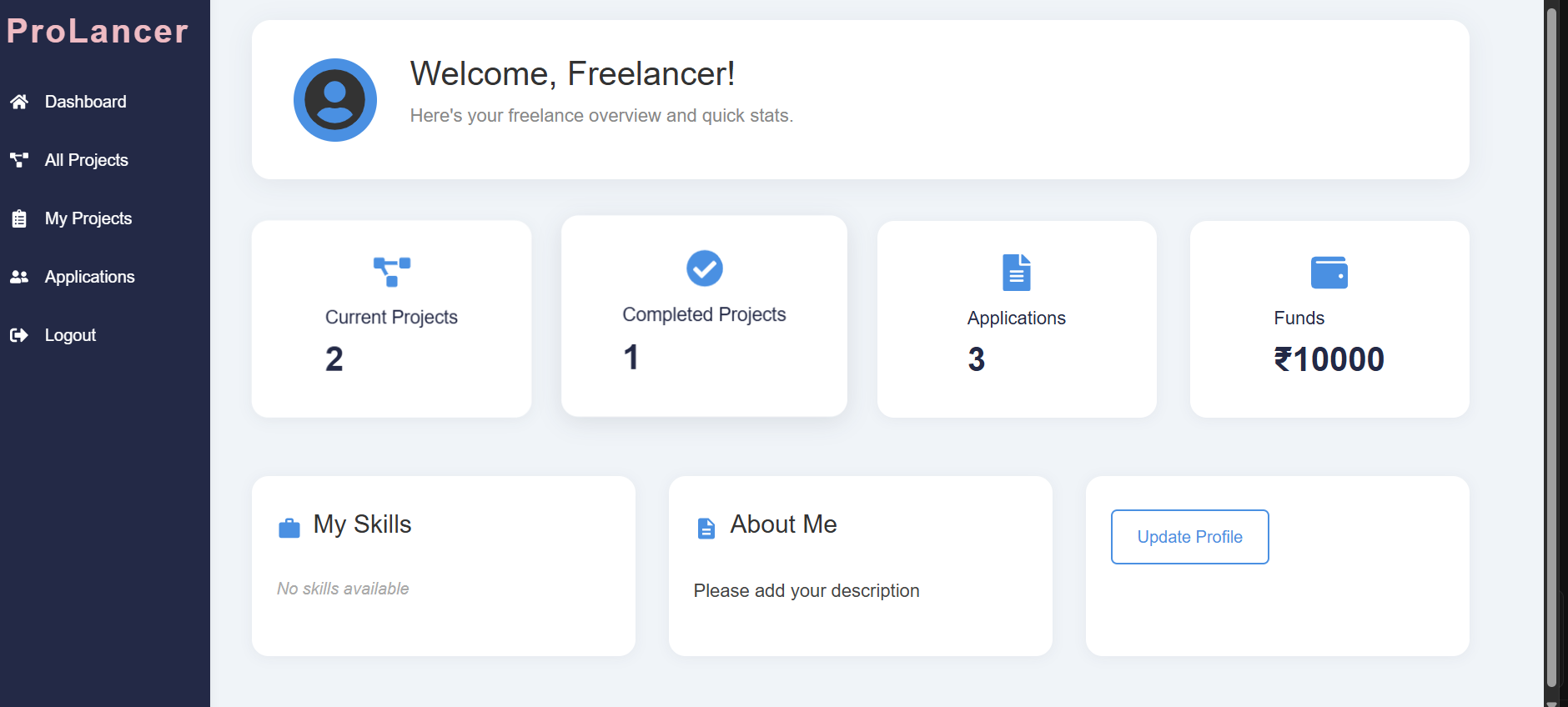
#### Applications Overview of Freelancers



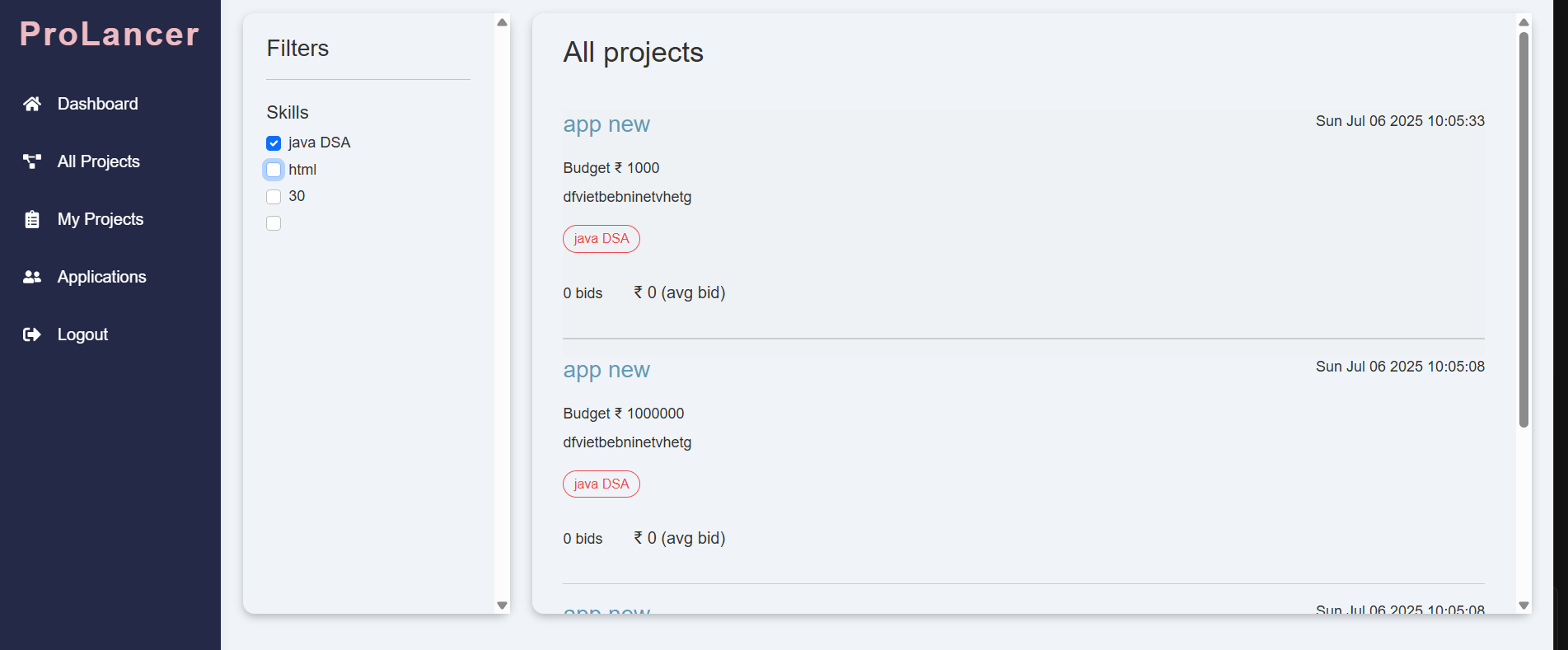
#### Client's Chat Window Alongside Project Details



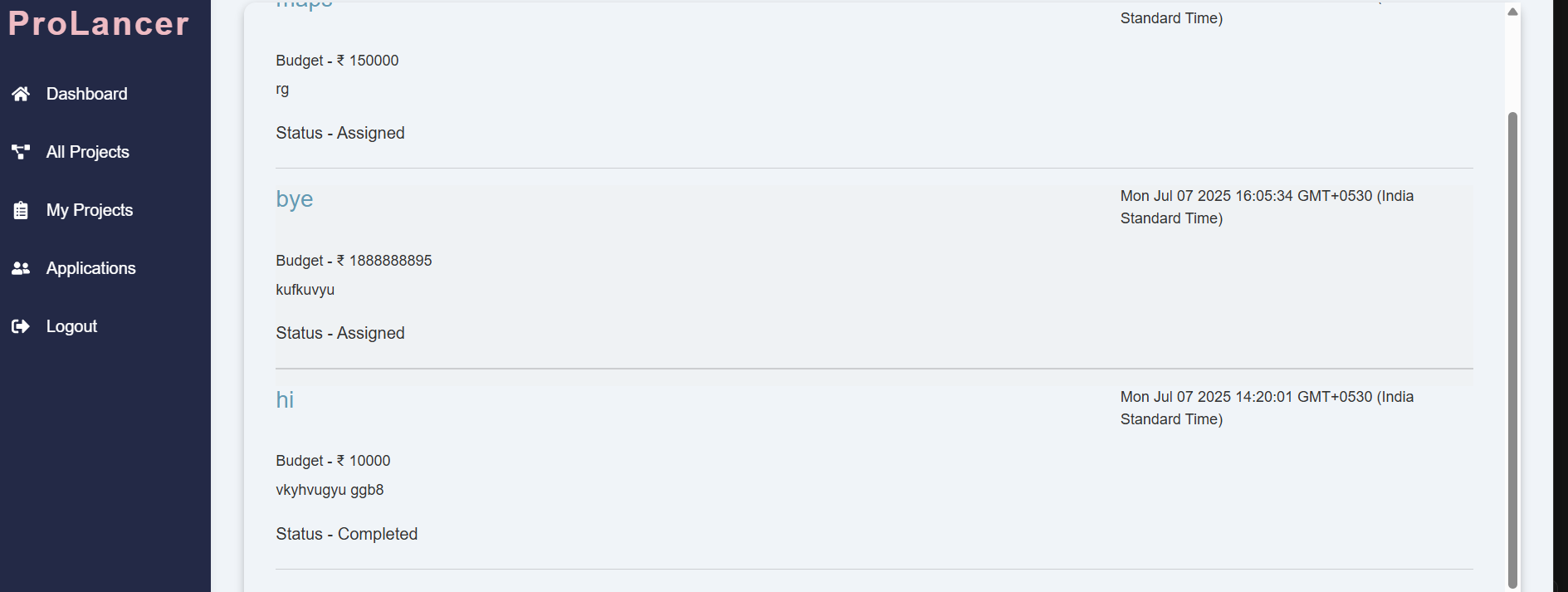
#### Freelancer Dashboard



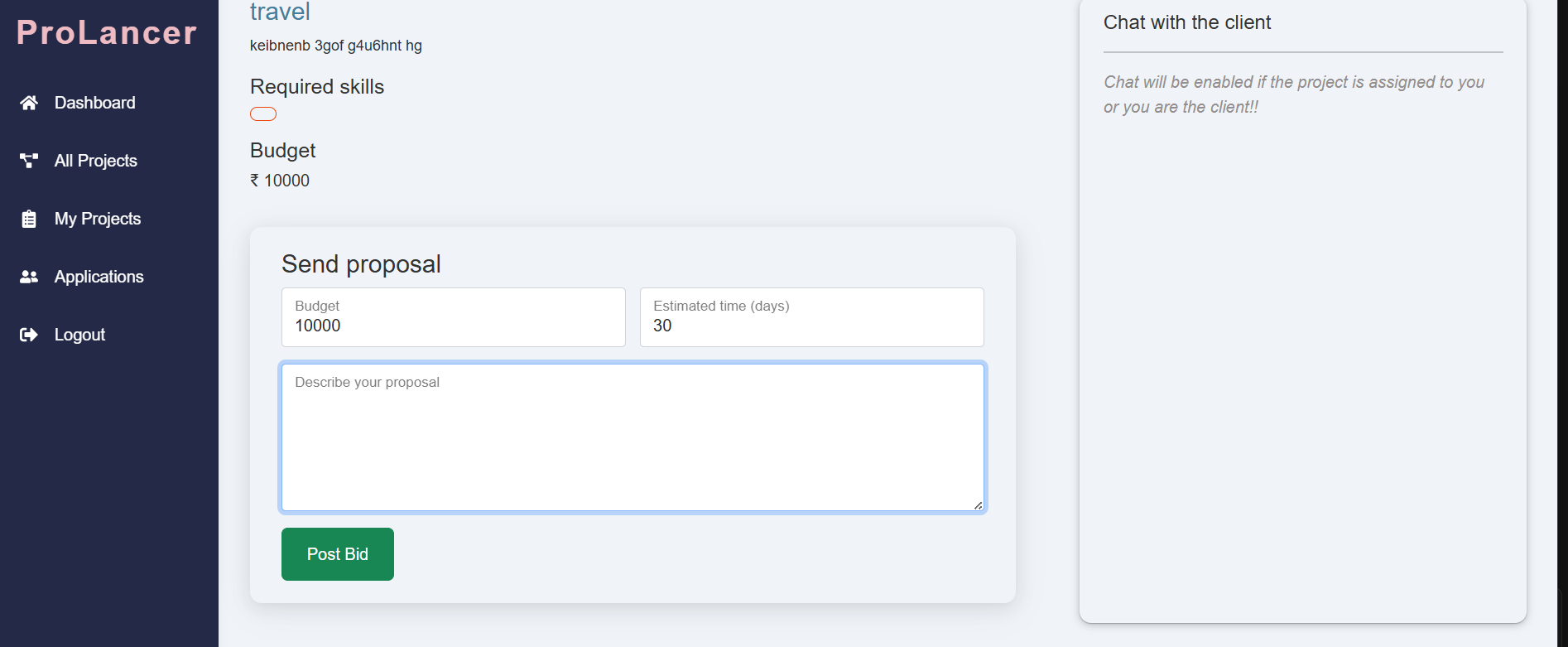
#### Freelancer View – All Projects with Filters



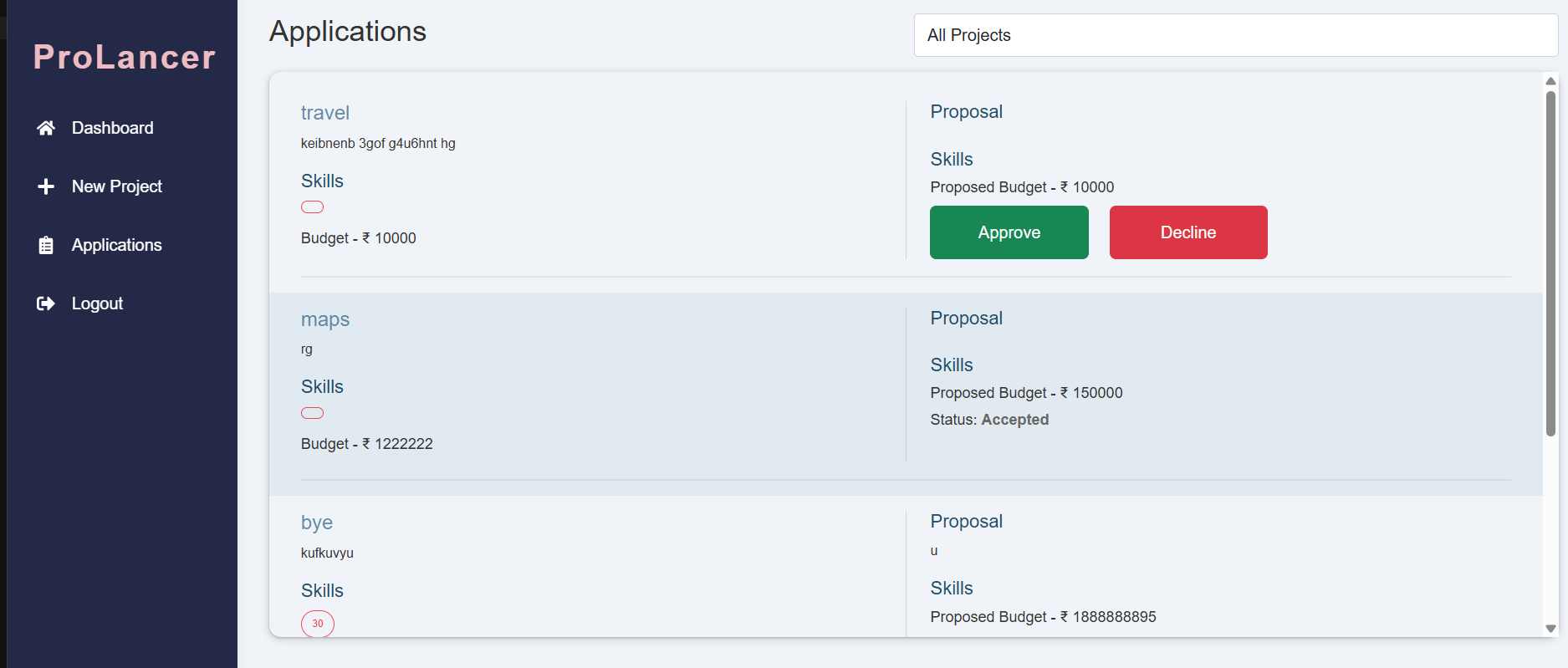
#### Freelancer's My-Projects Section



#### Freelancer Applying for a Project



#### Incoming Applications for a Project From Freelancers



6. How to Run the Project (Setup Guide)

###### 1. Clone the repository:

git clone https://github.com/vamshi329/ProLancer---A-Full-Stack-Freelancing-Platform.git

###### 2. Navigate into backend and install dependencies:

* cd Server
* npm install

###### 3. Navigate into frontend and install dependencies:

* cd ../Client
* npm install

###### 4. Create `.env` files and add:

* MongoDB URI
* JWT Secret
* Port

###### 5. Start servers:

* Backend -> npm run dev
* Frontend -> npm start

7. Challenges Faced

* Setting up stable Socket.IO communication between roles
* Ensuring both freelancer and client receive messages in real time
* Handling older project compatibility for chat feature

### 8. Future Enhancement

* Add typing indicators in chat
* Enable file sharing in messages
* Add notifications for message updates
* Support video calls

9. Conclusion

ProLancer demonstrates how a full-stack platform can connect clients and freelancers in real-time using modern web technologies. The project combines robust backend handling with clean frontend UI and interactive features like chat and submissions