FULL STACK DEVELOPMENT WITH MERN

1. Introduction

Project Title: FreelanceFinder: Discovering Opportunities, Unlocking Potential

Team Members:

BHEEMARASETTY TRIVEDH[Team Leader]
MEDE SAI LIKESH REDDY[Team member]
KAKARLA SREEKALA[Team member]
KOMATI S N R RAJAMANI DURGA[Team member]

Team Members and Roles:

Name	Role
BHEEMARASETTY TRIVEDH *(Team Leader)*	Full Stack Developer, Project Architect, GitHub Manager
MEDE SAI LIKESH REDDY	Backend Developer, API Integration, MongoDB Modeling
KAKARLA SREEKALA	Frontend Developer, UI/UX Designer, React Component Development
KOMATI S N R RAJAMANI DURGA	Tester & Documentation Lead, Functional Testing, Report Preparation

Team Overview:

Each member has contributed uniquely to the successful development of the FreelanceFinder platform. The team collaboratively ideated, designed, developed, and tested a full-stack MERN application that facilitates seamless interaction between clients and freelancers.

2. Project Overview

2.1Purpose:

The purpose of the SB Works Freelance Platform is to create a seamless and interactive environment where clients and freelancers can connect, collaborate, and manage projects efficiently. It is designed to bridge the gap between job providers and skilled professionals by offering a digital space that supports project posting, bidding, assignment, communication, and delivery. The platform is built to simplify the freelance workflow, ensure transparency, and maintain smooth communication from project initiation to final submission and approval.

2.1.2Key Features:

1 User Authentication Secure registration and login for clients, freelancers, and admins.

2 Project Posting Clients can post new freelance projects with details like description, skills, budget, and deadline.

3 Proposal Bidding System Freelancers can submit proposals with bid amount, estimated time, and description.

4 Application Management Clients can view proposals, accept or reject bids, and assign projects to freelancers.

5 Project Dashboard Freelancers and clients have separate dashboards to track assigned and applied projects.

6 Real-Time Chat System Socket.io-powered chat for seamless communication between clients and freelancers.

7 Project Submission Freelancers can submit project links, manuals, and descriptions for client review.

8 Client Review Clients can approve or reject submissions; approved submissions complete the project.

9 Admin Panel Admin can monitor project counts, user activity, and overall application performance.

10 Status Tracking Clear status updates for each project: Posted, Assigned, Completed, etc.

3. System Architecture

3.1Client Layer (Frontend)

Technology: React.js

Users: Clients, Freelancers, Admin

Role:

1. Provides an interactive user interface for functionalities such as registration, login, project posting, bidding, chatting, and file uploads.
2. Manages user sessions and authentication tokens using localStorage.
3. Utilizes **Axios** for making HTTP requests to backend APIs.
4. Integrates **Socket.IO client** for real-time messaging and notifications.

3.2Server Layer (Backend API)

Technology: Node.js with Express.js

Responsibilities:

1. Manages RESTful API endpoints for user authentication, project creation, application management, submissions, and admin tasks.
2. Uses **Socket.IO** to manage real-time events for chat and notifications.
3. Implements middleware for validating requests and enforcing role-based access control.
4. Organized modularly with separate routes for users, projects, and administrative operations.
5. Ensures secure operations with environment variables and proper error handling.

3.3Real-Time Layer (Chat System)

Technology: Socket.IO

Functions:

1. Enables real-time communication between freelancers and clients via dedicated project chat rooms.
2. Messages are transmitted instantly and broadcasted across connected user interfaces.
3. Chat functionality is activated only after a project has been assigned.
4. Messages are stored in MongoDB to maintain a persistent chat history for each project.

3.4Data Layer (Database)

Technology: MongoDB

Role:

1. **Serves as the central data store for the entire platform.**
2. **Stores collections for users (clients, freelancers, admins), project details, proposals, chat messages, and file submissions.**
3. **Utilizes Mongoose ODM for defining schemas and interacting with the database.**
4. **Enables real-time data updates through efficient querying, ensuring the platform reflects the latest actions and statuses across sessions.**

4. Setup Instructions

1. **Install Prerequisites :** Node.js, npm, MongoDB (local or Atlas)
2. **Clone Repository : https://github.com/trivedh2316/FreelanceFinder-Discovering-Opportunities-Unlocking-Potential**
3. **Start MongoDB :** Run mongod (or connect Atlas DB)
4. **Setup Backend:** cd server npm install node index .js
5. **Setup Frontend:** cd client npm install npm start
6. **Access the App:** Frontend: http://localhost:3000
7. Backend API: http://localhost:6001
8. **Socket.IO:** Real-time chat works only for assigned projects.

5. Folder Structure

5.1Client (React Frontend) :

The client directory contains the entire frontend built with React.js.It is organized as follows:

1. **public/** – Contains static files like index.html and images.
2. **src/** – Main source code folder:
3. **components/** – Reusable components such as buttons, cards, input forms.
4. **pages/** – Page-level components (e.g., Home, Login, AllProjects, MyProjects).
5. **context/** – React context for managing global state (user, socket, etc.).
6. **styles/** – All CSS styles specific to components and pages.
7. **App.jsx** – Main application file managing routing and layout.
8. **package.json** – Lists frontend dependencies and scripts.

5.2Server (Node.js + Express Backend):

The server directory is built with Node.js, Express.js, and MongoDB.

It is organized to maintain a clean and scalable structure:

1. **models/** – Mongoose schemas for User, Project, Application, Chat.
2. **routes/** – API route files that handle HTTP requests.
3. **controllers/** – Functions that handle business logic and interact with models.
4. **SocketHandler.js** – Manages Socket.IO logic for real-time messaging.
5. **index.js** – Entry point of the server; sets up Express, MongoDB connection, and sockets.
6. **package.json** – Lists backend dependencies and scripts.

6. Running the Application

To run the project locally, follow these steps for both the frontend and backend servers:

1. Frontend (React) :
2. Open a terminal.
3. Navigate to the client folder: cd client
4. Start the React development server: npm start
5. The frontend will run at: http://localhost:3000
6. Backend (Node.js + Express) :
7. Open another terminal window.
8. Navigate to the server folder: cd server
9. Start the backend server: node index.js
10. The backend will run at: http://localhost:6001

7. API Documentation

7.1User APIs:

POST /register

Registers a new user.

Request Body:

{

"name": "John Doe",

"email": "john@example.com",

"password": "123456",

"role": "freelancer"

}

POST /login

Authenticates a user.

Request Body:

{

"email": "john@example.com",

"password": "123456"

}

7.2Project APIs:

POST /post-project

Allows a client to post a new freelance project.

Request Body:

{

"title": "Build Portfolio Website",

"description": "React-based portfolio site",

"budget": 5000,

"skills": ["React", "CSS"],

"clientId": "abc123"

}

GET /fetch-projects

Returns a list of all available projects.

GET /fetch-project/:id

Returns detailed info for a specific project.

7.3Application (Proposal) APIs :

POST /make-bid

Allows a freelancer to apply to a project.

Request Body:

{

"clientId": "abc123",

"freelancerId": "def456",

"projectId": "proj789",

"bidAmount": 4500,

"estimatedTime": "4 days",

"proposal": "I can deliver this on time with great quality."

}

GET /fetch-applications

Returns all applications for the logged-in client.

GET /approve-application/:id

Client approves a specific proposal.

GET /reject-application/:id

Client rejects a specific proposal.

7.4Chat APIs:

GET /fetch-chats/:projectId

Fetches chat history for a specific project.

Real-time chat is handled via Socket.io.

new-message

join-chat-room

message-from-user

7.5Submission APIs

POST /submit-project

Freelancer submits final project details.

Request Body:

{

"projectId": "proj789",

"projectLink": "https://github.com/myproject",

"manualLink": "https://drive.com/manual",

"submissionDescription": "This is the final version."

}

GET /approve-submission/:projectId

Client approves submitted project.

GET /reject-submission/:projectId

Client rejects submission and allows resubmission.

.

8. Authentication Flow

The authentication system ensures secure access and role-based routing for Freelancers, Clients, and Admins. It includes registration, login, token-based session management, and protected routes.

8.1User Registration (POST /register)

8.1.1 Users (Freelancer or Client) sign up by submitting:

- **Name**
- **Email**
- **Password**
- **Role (client or freelancer)**

8.1.2 The backend performs the following:

- **Validates the input fields.**
- **Checks for existing users with the same email.**
- **Hashes the password using bcrypt.**
- **Stores the user in the MongoDB database.**

Security Measures:

- **Passwords are never stored in plain text.**
- **Only essential information is returned after successful registration.**

8.2 User Login (POST /login)

- **Users provide email and password to authenticate.**
- **The backend performs:**
  - **Email lookup in the database.**
  - **Password verification using bcrypt.compare.**
  - **On success, a JWT (JSON Web Token) is generated and sent back.**

Response includes:

- **JWT Token**
- **userId**
- **usertype (e.g., "client", "freelancer")**

These details are stored in localStorage on the frontend to:

- **Maintain session across page reloads.**

- **Enable role-based navigation (navigate(/client) or navigate(/freelancer)).**

8.2 Authorization & Protected Routes

8.2.1 Backend routes (e.g., /api/projects, /api/applications) are protected using middleware that:

1. **Verifies the JWT token from request headers.**
2. **Decodes the token to identify the user.**
3. **Rejects unauthorized requests.**

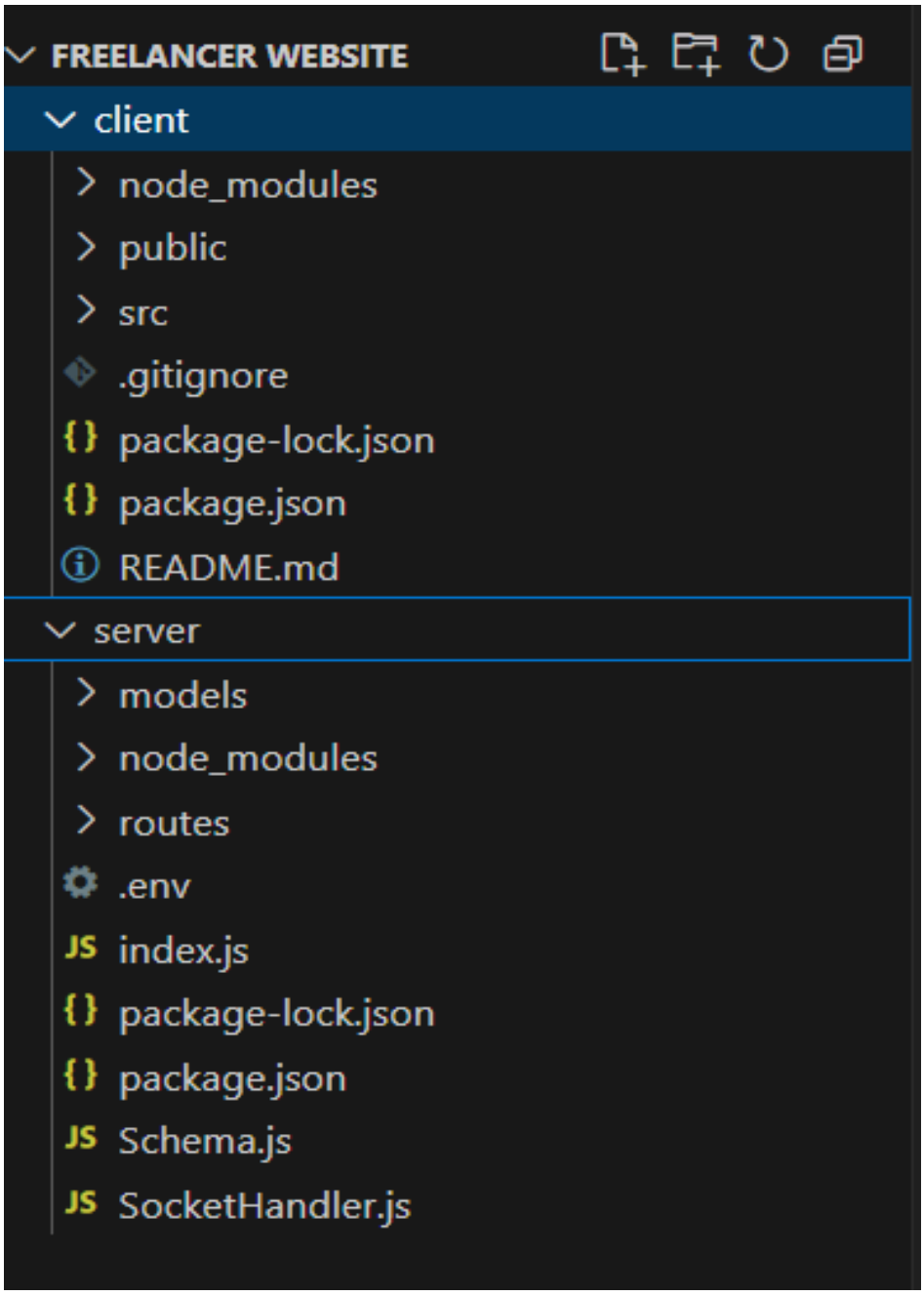
9.Testing

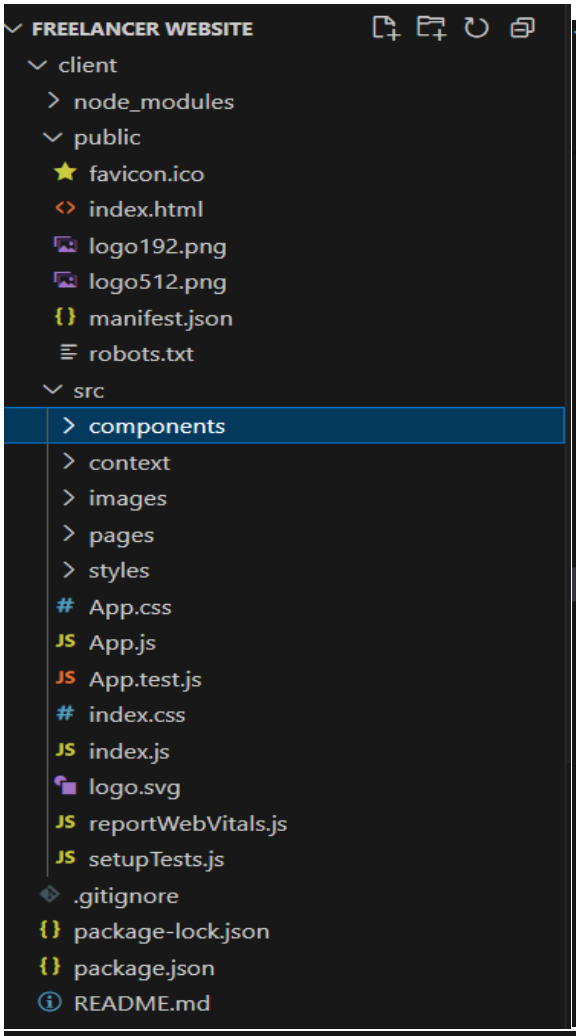
Testing Strategy : The project follows a manual and functional testing strategy to ensure all user flows and features work as expected across different roles (Client, Freelancer, and Admin).

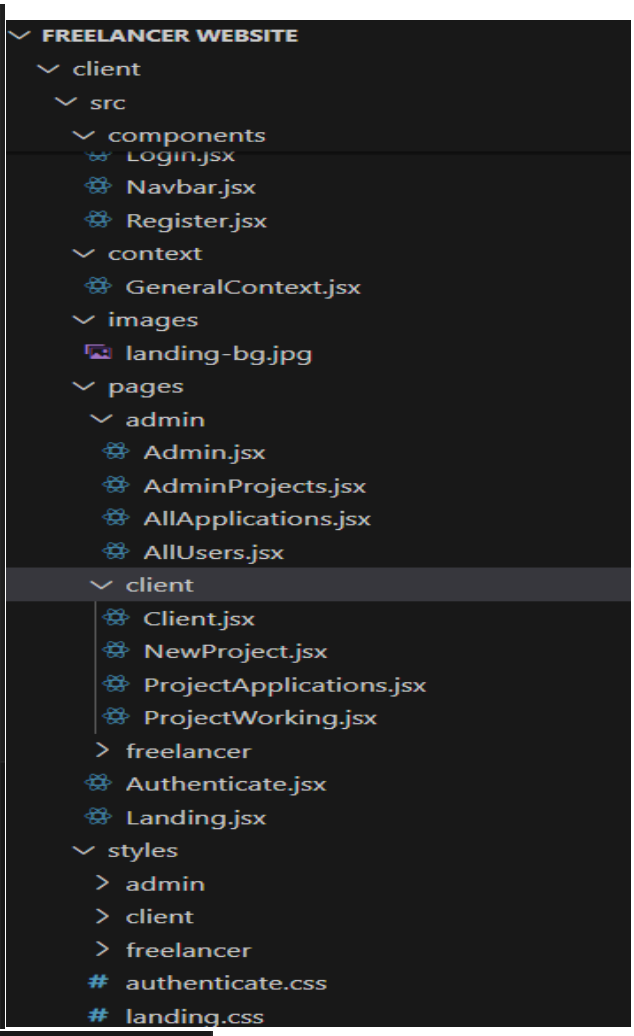
- **Unit Testing:** Core functions and utility logic were tested manually during development.
- **Integration Testing:** Chat integration with socket.io was tested for real-time communication between client and freelancer. Project submission and approval workflows were validated from both ends.
- **End-to-End Testing:** Complete workflows were tested including: Project posting → Proposal submission → Approval → Submission → Final approval. Chat and socket connections between users.
- **UI/UX Testing:** Manual testing was done across devices (desktop + mobile) to ensure responsive layout and smooth navigation.

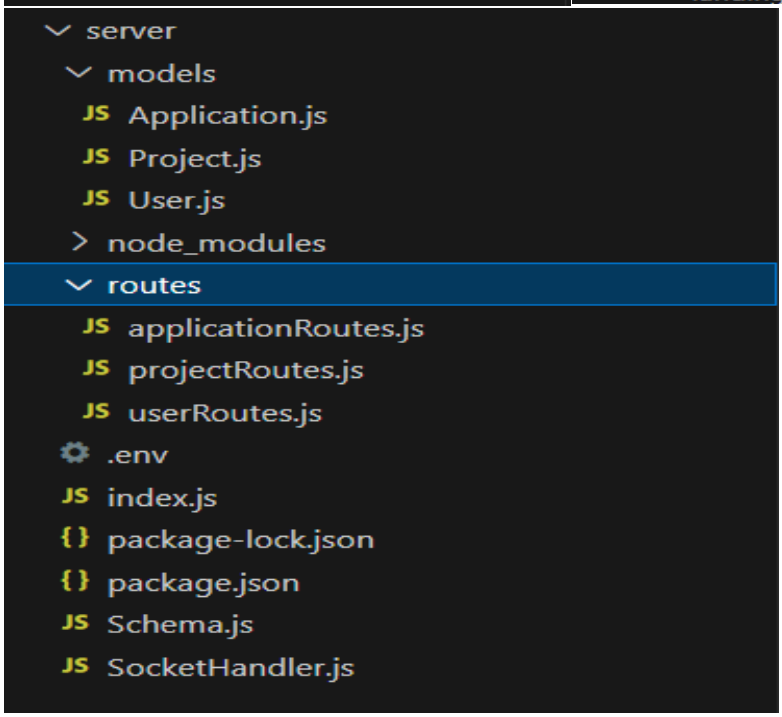
10. Screenshots / Demo

10.1 Folder Structure

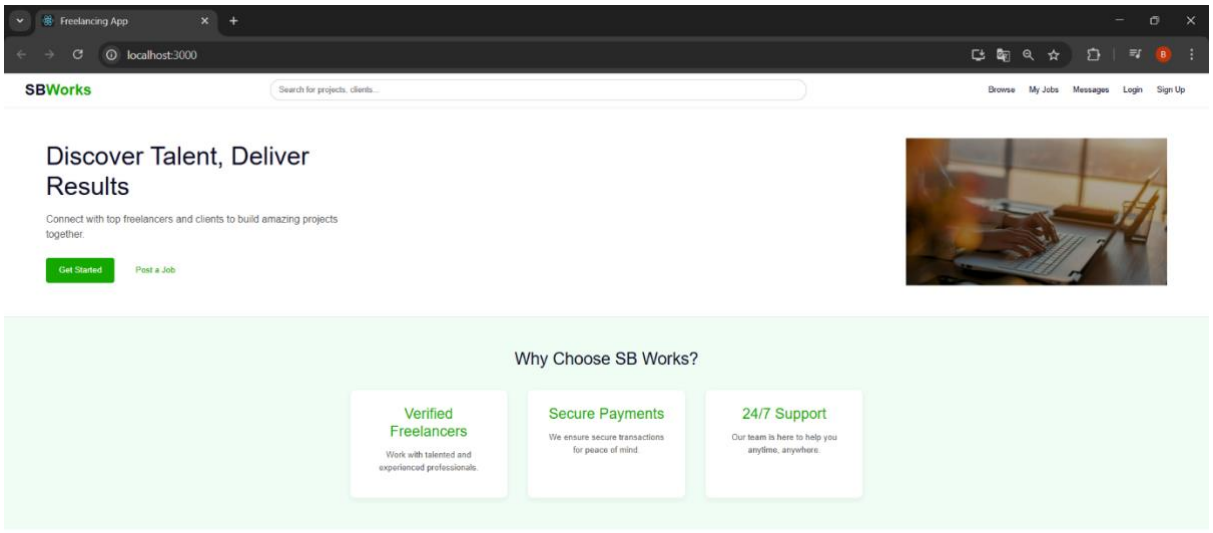


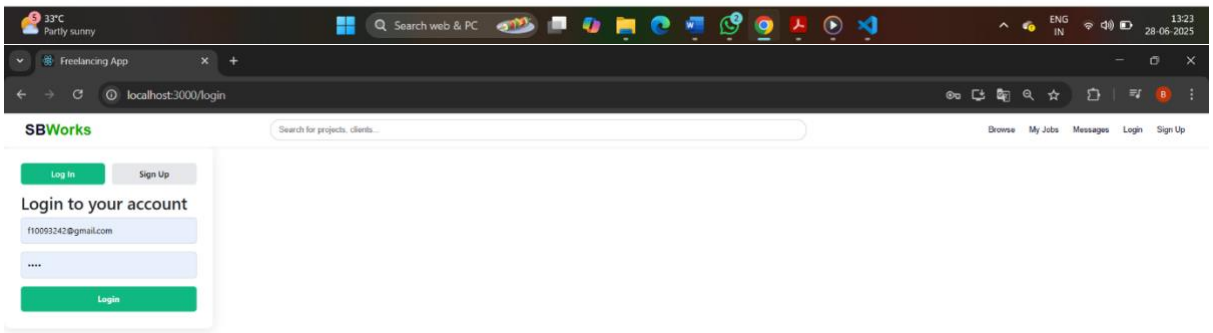

FREELANCER WEBSITE
client
node_modules
public
favicon.ico
index.html
logo192.png
logo512.png
manifest.json
robots.txt
src
components
context
images
pages
styles
App.css
App.js
App.test.js
index.css
index.js
logo.svg
reportWebVitals.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md

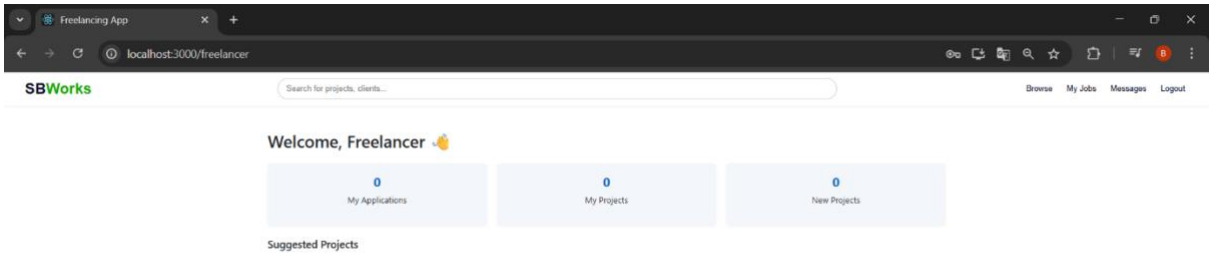

FREELANCER WEBSITE
client
src
components
Login.jsx
Navbar.jsx
Register.jsx
context
GeneralContext.jsx
images
landing-bg.jpg
pages
admin
Admin.jsx
AdminProjects.jsx
AllApplications.jsx
AllUsers.jsx
client
Client.jsx
NewProject.jsx
ProjectApplications.jsx
ProjectWorking.jsx
freelancer
Authenticate.jsx
Landing.jsx
styles
admin
client
freelancer
authenticate.css
landing.css

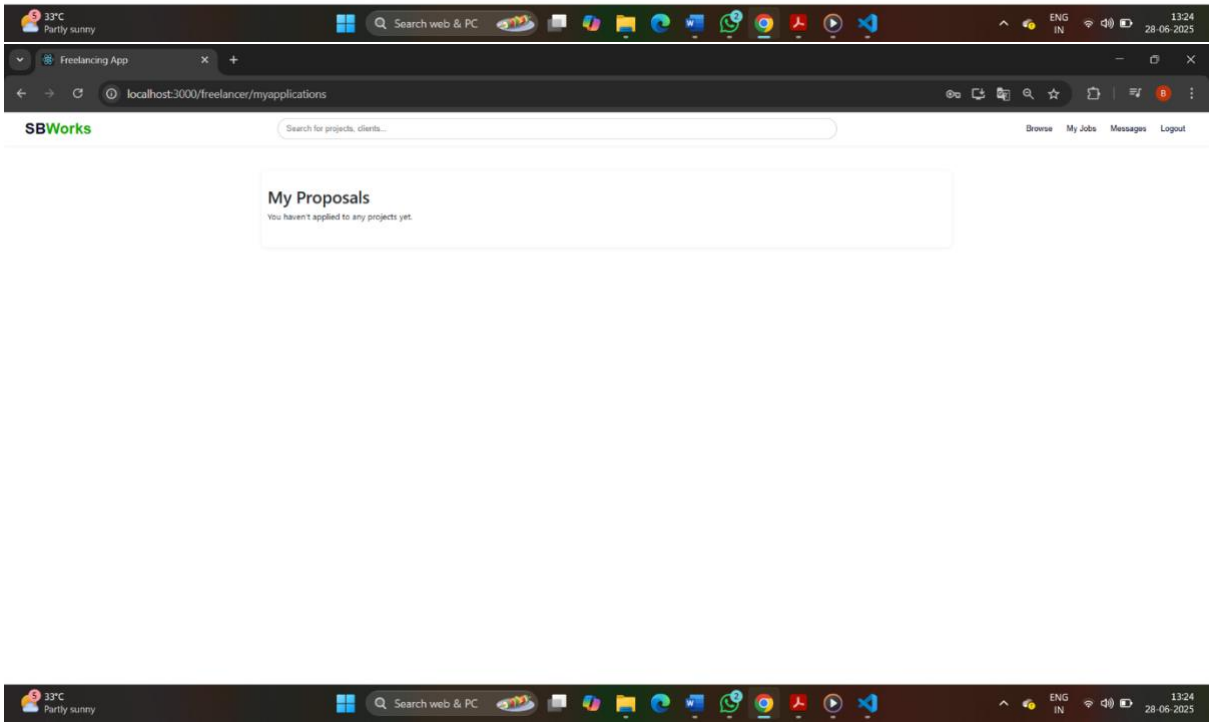

server
models
Application.js
Project.js
User.js
node_modules
routes
applicationRoutes.js
projectRoutes.js
userRoutes.js
.env
index.js
package-lock.json
package.json
Schema.js
SocketHandler.js

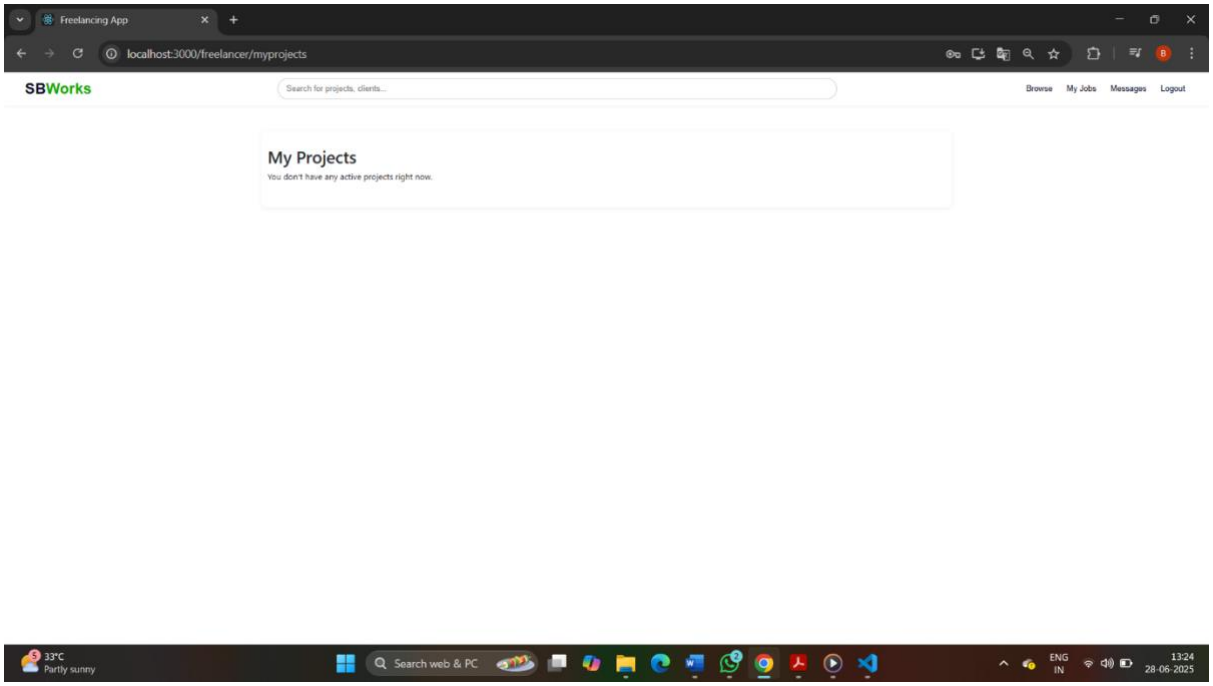
10.2 DEMO PHOTOS






Freelancing App
localhost:3000/freelancer
SBWorks
Search for projects, clients...
Browse
My Jobs
Messages
Logout
Welcome, Freelancer
0
My Applications
0
My Projects
0
New Projects
Suggested Projects


33°C
Partly sunny
Search web & PC
ENG
IN
13:24
28-06-2025
Freelancing App
localhost:3000/freelancer/myapplications
SBWorks
Search for projects, clients...
Browse
My Jobs
Messages
Logout
My Proposals
You haven't applied to any projects yet.
33°C
Partly sunny
Search web & PC
ENG
IN
13:24
28-06-2025


Freelancing App
localhost:3000/freelancer/myprojects
SBWorks
Search for projects, clients...
Browse
My Jobs
Messages
Logout
My Projects
You don't have any active projects right now.
33°C
Partly sunny
Search web & PC
ENG
IN
13:24
28-06-2025

11. Known Issues

Real-Time Chat Delay

Description: The real-time chat feature, powered by Socket.IO, sometimes experiences a slight delay in message delivery between the sender and recipient. This is especially noticeable in low-speed internet environments or during backend restarts. Although messages are eventually delivered, the delay can impact the user experience, particularly for time-sensitive communication.

Missing Validation in Proposal Forms

Description: Currently, the input forms for submitting proposals lack strong validation. For instance, users can submit empty fields or invalid numeric inputs like zero or negative bid amounts. This could result in incomplete or incorrect data being stored in the database, which may confuse the client reviewing the proposals.

No File Upload for Project Manual

Description: The project manual submission feature expects a URL input instead of supporting direct file uploads. Freelancers must first upload the manual to an external service (like Google Drive or Dropbox) and then paste the link. While this workaround is functional, it lacks convenience and may affect the professional experience of users.

Lack of Role-Based Access Protection on Frontend

Description: Although the backend distinguishes between client and freelancer roles, the frontend currently lacks robust role-based access control. This means that users can potentially access

unauthorized pages (e.g., a freelancer opening an admin panel) simply by modifying the URL. This is a security gap and should be addressed with route guards and user-role checks.

UI Not Fully Optimized for Small Devices

Description: Several user interface components, such as the chat interface, proposal forms, and project detail layouts, are not fully responsive on mobile or smaller screen sizes. This results in layout breaking, overflow issues, or content misalignment, which can degrade usability on phones and tablets.

12. Future Enhancements

1 File Upload for Submissions Add support for freelancers to directly upload project files (e.g., ZIP files, PDFs) instead of sharing external links. This would improve usability and make the platform more professional.

2 Advanced Search and Filters Implement advanced search options and filters on the project listing pages. Users could filter by skills, budget range, status, and posted date to easily find relevant projects or freelancers.

3 Role-Based Dashboard Access Improve route protection by enforcing role-based authentication in the frontend. This would ensure that freelancers, clients, and admins can only access permitted sections.

4 In-App Notifications Add real-time notification support to inform users about project updates, proposal responses, submission approvals/rejections, or new messages.

5 Project Deadline & Tracking Allow clients to define deadlines and freelancers to update progress. This could be visualized using a simple timeline or progress bar.

6 Rating & Review System Let clients rate and review freelancers after project completion. This helps build credibility and assists other clients in choosing the right freelancer.

7 Admin Analytics Dashboard Extend the admin panel to include charts and metrics showing active users, posted projects, completion rates, most-used skills, etc., for better system monitoring.

8 Mobile App Version Develop a mobile version of the platform for Android and iOS, enabling users to manage their freelance tasks and communication on the go.

9 Freelancer Portfolio Page Provide freelancers with a public portfolio page where clients can view their skills, completed projects, ratings, and more.

10 Payment Gateway Integration Add payment and escrow support, allowing clients to pay securely and freelancers to get paid only after project approval