

# Full Stack Development with MERN

## Project Documentation format

### 1. INTRODUCTION:

Freelancing 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.

#### ❖ PROJECT TITLE:

Freelancing finder: Discovering Opportunities, Unlocking Potentials.

#### ❖ TEAM MEMBERS:

Team Leader: Ch. Srinu – Frontend development

Team member: Syed. Muskan - Backend development

Team member: Shaik. Sharmila - Testing & deployment

Team member: Mannem. Yamini- Final documentation

### 2. PROJECT OVERVIEW:

Freelance Finder is a next-generation freelancing platform designed to revolutionize how clients and freelancers connect, collaborate, and succeed. With an intuitive interface, clients can post a wide range of projects—from creative ventures to technical challenges—while freelancers bid based on their expertise. The platform emphasizes transparency and efficiency, allowing clients to review profiles, evaluate past work, and seamlessly communicate with selected freelancers throughout the project lifecycle. A dedicated admin team ensures the integrity and security of every transaction, fostering a professional environment built on trust.

Freelancers enjoy a streamlined process for submitting completed work, while clients benefit from a hassle-free review and feedback system that encourages collaboration and excellence. Real-time updates and notifications keep both parties informed about project status and industry trends. At Freelance Finder, we're committed to helping clients discover the right talent and enabling freelancers to unlock their full potential in a dynamic and secure freelancing ecosystem.

#### ❖ PURPOSE:

The purpose of the Freelance Finder project is to build a next-generation freelancing platform that transforms the way clients and freelancers connect, collaborate, and succeed. By offering an intuitive, secure, and transparent environment, the platform enables clients to easily find and work with skilled professionals, while empowering freelancers to showcase their expertise and access meaningful opportunities. Freelance Finder is committed to streamlining project

workflows, fostering trust, and unlocking the full potential of both clients and freelancers in a dynamic digital ecosystem.

#### ❖ FEATURES:

**Project Posting & Bidding:** Clients post diverse projects; freelancers bid based on skills and experience.

**Profile & Portfolio Management:** Freelancers showcase expertise; clients review profiles before hiring.

**Built-in Communication:** Real-time messaging enables smooth collaboration.

**Work Submission & Feedback:** Freelancers submit work easily; clients review and rate completed tasks.

**Admin Support & Oversight:** Platform integrity ensured through admin monitoring and dispute resolution.

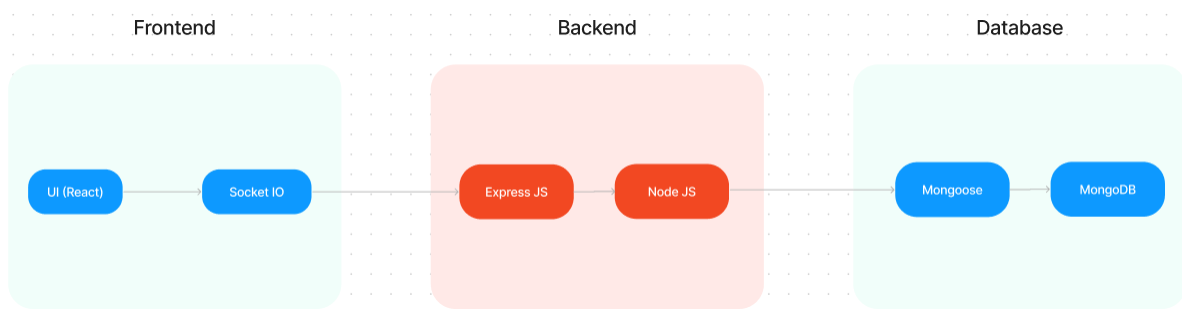
**Notifications & Updates:** Users receive real-time alerts on project status and new opportunities.

**Secure Payments:** Protected transactions with milestone or escrow options.

**Search & Filters:** Smart tools help clients find the right talent quickly.

### 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.



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.

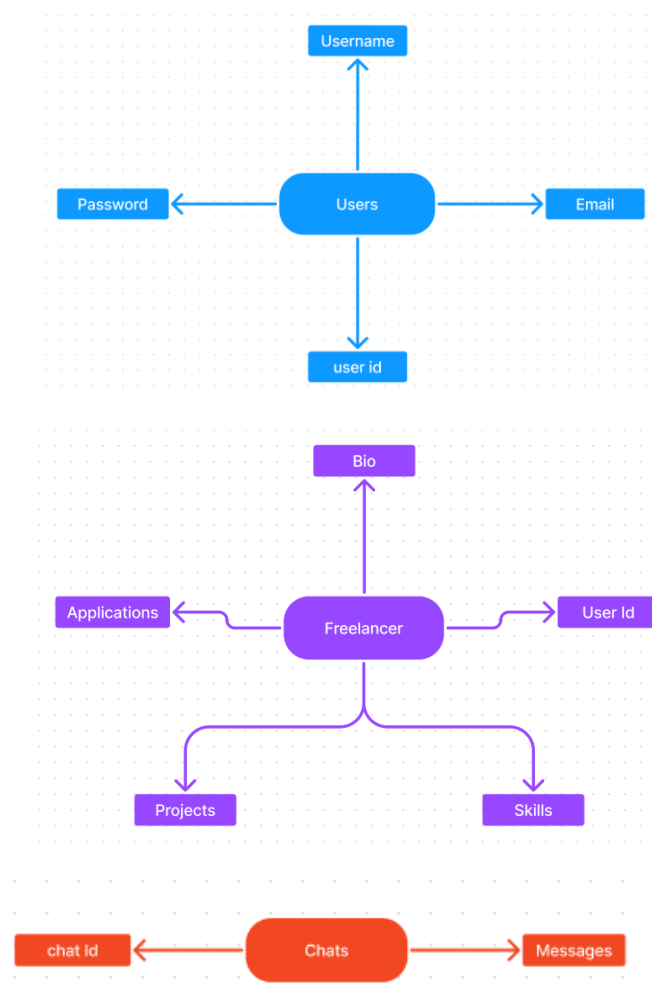
To enhance the user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a real-time and visually appealing interface for users.

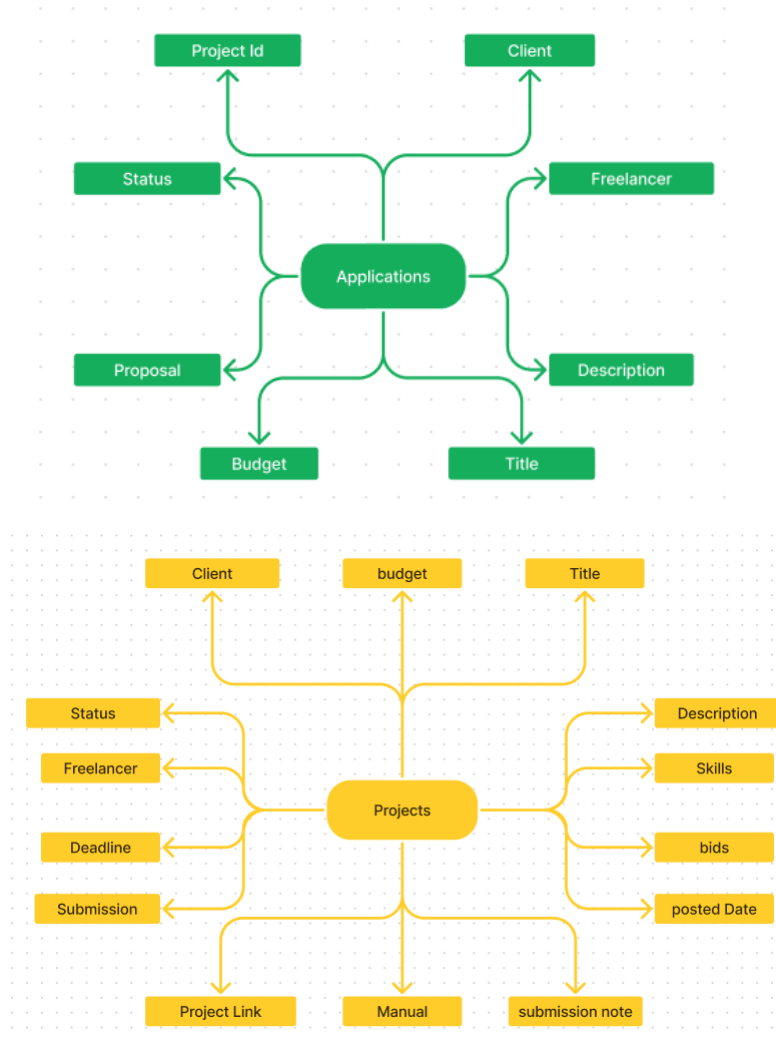
On the backend, we utilize the Express Js framework to manage server-side logic and communication. Express Js provides a robust foundation for handling requests and responses efficiently.

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:





Freelancer finder connects clients with skilled freelancers through a user-friendly platform. Clients can post projects with details and browse freelancer profiles to find the perfect match. Freelancers can submit proposals, collaborate with clients through secure chat, and securely submit work for review and payment. An admin team ensures quality and communication, making Freelancer Finder a go-to platform for both clients and freelancers.

## 4. SETUP INSTRUCTIONS:

To set up the Freelance Finder project locally, first clone the repository and navigate into the project folder. Go to the client folder, install dependencies with `npm install`, and start the frontend using `npm start` or `npm run dev`. Then, in the server folder, run `npm install` and create a `.env` file with variables like `PORT`, `MONGO_URI`, and `JWT_SECRET`. Use MongoDB Atlas for the database if needed. Start the backend with `npm run dev`. The frontend will run at `http://localhost:3000` and the backend API at `http://localhost:5000/api`. Make sure Node.js and npm are installed. For deployment, Vercel and Render are good options.

## ❖ 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.

### ✓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.

### ✓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.

**✓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.

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

### 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
```

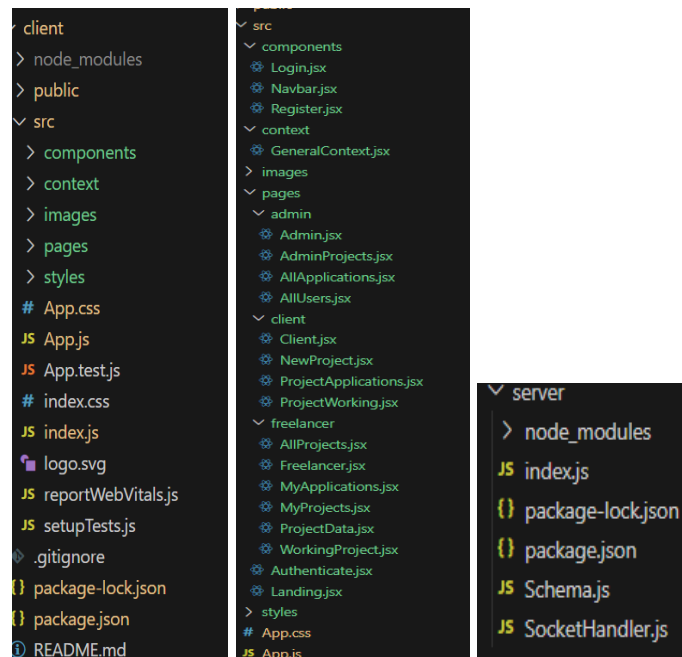
Start the Development Server:

- To start the development server, execute the following command:  
npm start

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.

## 5. FOLDER STRUCTURE:

SB Works leverages React.js for the user interface. The client-side code likely consists of reusable components for profiles, projects, and chat, assembled into pages like project browsing or freelancer profiles. Shared data like user info or search filters might be managed with React Context. On the server side, Node.js handles API requests for user management, project actions, and communication. Mongoose models ensure structured interaction with the MongoDB database. This breakdown provides a foundational understanding of SB Works' architecture.



## 6. RUNNING THE APPLICATION:

### ❖ Backend Development

#### 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

## **❖ Frontend development**

### **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.

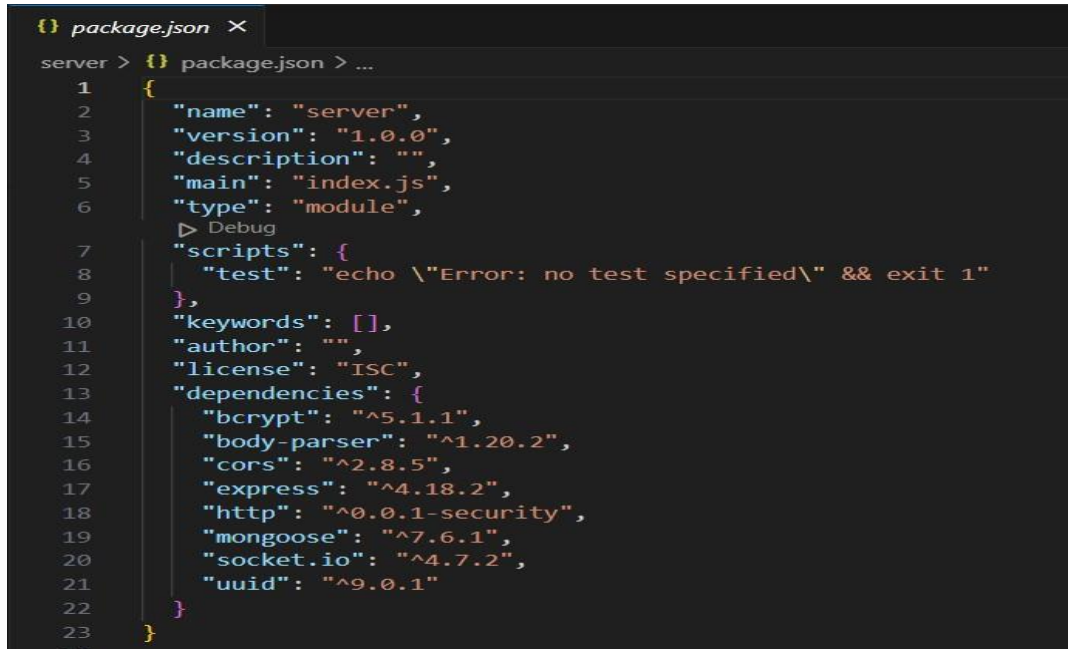


## 7. API DOCUMENTATION:

### ☐ Authentication APIs

```
{ package.json M X
client > {} package.json > ...
3  "version": "0.1.0",
4  "private": true,
5  "dependencies": {
6    "@testing-library/jest-dom": "^5.17.0",
7    "@testing-library/react": "^13.4.0",
8    "@testing-library/user-event": "^13.5.0",
9    "axios": "^1.5.1",
10   "react": "^18.2.0",
11   "react-dom": "^18.2.0",
12   "react-icons": "^4.11.0",
13   "react-router-dom": "^6.19.0",
14   "react-scripts": "5.0.1",
15   "socket.io-client": "^4.7.2",
16   "uuid": "^9.0.1",
17   "web-vitals": "^2.1.4"
18 },
19   > Debug
20   "scripts": {
21     "start": "react-scripts start",
22     "build": "react-scripts build",
23     "test": "react-scripts test",
24     "eject": "react-scripts eject"
25   },
26   "eslintConfig": {
27     "extends": [
28       "react-app",
29       "react-app/jest"
30     ],
31   },
32   "browserslist": {
33     "production": [
34       ">0.2%",
35       "not dead",
36       "not op_mini all"
37     ],
38     "development": [
39       "last 1 chrome version",
40       "last 1 firefox version",
41       "last 1 safari version"
42     ]
43   }
44 }
```

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

A screenshot of a code editor window titled 'package.json'. The editor shows the content of the package.json file for a project named 'server'. The file is a JSON object with the following properties: 'name' is 'server', 'version' is '1.0.0', 'description' is an empty string, 'main' is 'index.js', 'type' is 'module', 'scripts' is an object with a 'test' property set to 'echo \"Error: no test specified\" && exit 1', 'keywords' is an empty array, 'author' is an empty string, 'license' is 'ISC', and 'dependencies' is an object listing several dependencies with version constraints: 'bcrypt' (^5.1.1), 'body-parser' (^1.20.2), 'cors' (^2.8.5), 'express' (^4.18.2), 'http' (^0.0.1-security), 'mongoose' (^7.6.1), 'socket.io' (^4.7.2), and 'uuid' (^9.0.1). The code is displayed with line numbers from 1 to 23 on the left margin.

```
1 {
2   "name": "server",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "type": "module",
7   "scripts": {
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "bcrypt": "^5.1.1",
15    "body-parser": "^1.20.2",
16    "cors": "^2.8.5",
17    "express": "^4.18.2",
18    "http": "^0.0.1-security",
19    "mongoose": "^7.6.1",
20    "socket.io": "^4.7.2",
21    "uuid": "^9.0.1"
22  }
23 }
```

The backend of the Freelance Finder project exposes a range of RESTful API endpoints grouped into several core modules: authentication, user management, job posting, bidding, and messaging. Authentication endpoints include `/api/auth/register` for user registration and `/api/auth/login` for logging in and obtaining a JWT token. User-related endpoints such as `/api/users/profile` (GET and PUT) allow authenticated users to view and update their profiles. Clients can create job listings using the `/api/jobs` POST endpoint, retrieve all jobs via `/api/jobs`, and get specific job details through `/api/jobs/:id`. Freelancers can place bids on jobs using the `/api/bids` endpoint and clients can view all bids on a specific job using `/api/bids/job/:jobId`. The platform also supports real-time communication through messaging endpoints, with `/api/messages` enabling users to send messages and `/api/messages/:userId` allowing them to retrieve message history with a specific user. These endpoints form the foundation of the Freelance Finder backend, supporting all critical functionalities for user interaction and workflow.

## 8. AUTHENTICATION:

Authentication and authorization in the Freelance Finder project are handled using `*JWT (JSON Web Token)*` based security. When a user successfully logs in through the `POST /api/auth/login` endpoint, the server generates a signed JWT token and returns it to the client. This token contains encoded user information (such as user ID and role) and is stored client-side—typically in `*localStorage*` or `*HTTP-only cookies*`—depending on the frontend setup.

Each time the client makes a protected API request (like fetching a profile or posting a job), it must include the token in the `**Authorization header**` as:

Authorization: Bearer <your\_jwt\_token>

The backend middleware verifies the token on every request to ensure it's valid and not expired. If valid, it allows access and attaches the authenticated user to the request context. If invalid or missing, the API returns a 401 Unauthorized response.

There are *\*no server-side sessions\** maintained—authentication is *\*stateless\**. This approach allows scalable and efficient handling of user authorization across endpoints, with role-based checks (like client vs freelancer) to enforce access control where necessary.

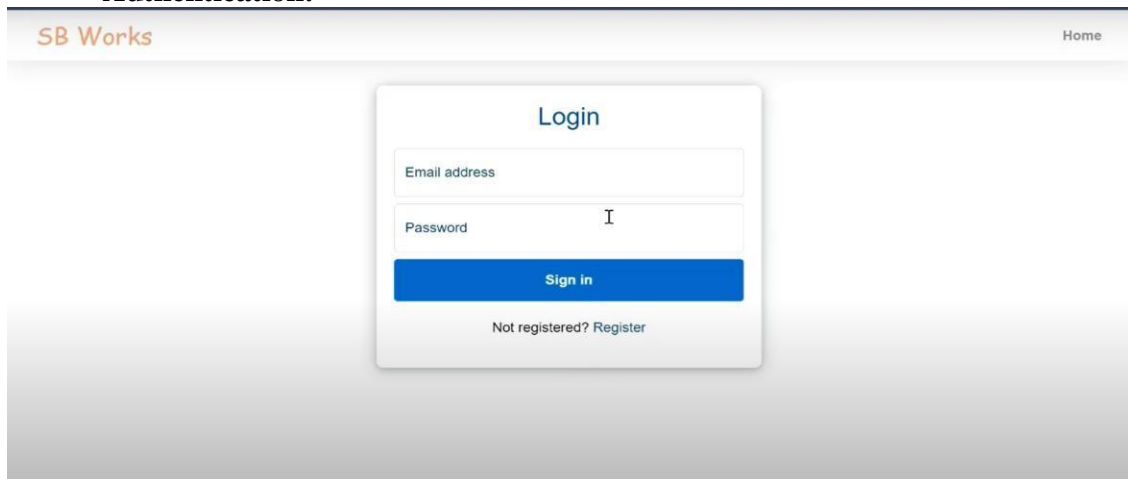
## 9. USER INTERFACE:

- Providing screenshots or GIFs showcasing different UI features.

### Landing page:



### Authentication:



## 10. TESTING:

- Describe the testing strategy and tools used.

### G Testing Strategy:

To ensure the reliability and quality of the application, a combination of **unit testing**, **integration testing**, and **manual testing** was used:

**1. Unit Testing:**

- Focused on testing individual functions, models, and API controllers (e.g., user authentication, profile updates).
- Ensured each component works as expected in isolation.

**2. Integration Testing:**

- Verified end-to-end functionality of APIs (e.g., user registration to project hiring).
- Checked interactions between components like database, middleware, and routes.

**3. Manual Testing (UI & UX):**

- Tested all core user flows (e.g., login, posting projects, hiring freelancers).
- Used for quick verification during development and identifying UI/UX bugs.

✓ **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.

```
package.json M X
client > {} package.json > ...
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "@testing-library/jest-dom": "^5.17.0",
7     "@testing-library/react": "^13.4.0",
8     "@testing-library/user-event": "^13.5.0",
9     "axios": "^1.5.1",
10    "react": "^18.2.0",
11    "react-dom": "^18.2.0",
12    "react-icons": "^4.11.0",
13    "react-router-dom": "^6.19.0",
14    "react-scripts": "5.0.1",
15    "socket.io-client": "^4.7.2",
16    "uuid": "^9.0.1",
17    "web-vitals": "^2.1.4"
18  },
19  "scripts": {
20    "start": "react-scripts start",
21    "build": "react-scripts build",
22    "test": "react-scripts test",
23    "eject": "react-scripts eject"
24  },
25  "eslintConfig": {
26    "extends": [
27      "react-app",
28      "react-app/jest"
29    ]
30  },
31  "browserslist": {
32    "production": [
33      ">0.2%",
34      "not dead",
35      "not op_mini all"
36    ],
37    "development": [
38      "last 1 chrome version",
39      "last 1 firefox version",
40      "last 1 safari version"
41    ]
42  }
43 }
44
```

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

```
package.json X
server > {} package.json > ...
1   {
2     "name": "server",
3     "version": "1.0.0",
4     "description": "",
5     "main": "index.js",
6     "type": "module",
7     "scripts": {
8       "test": "echo \"Error: no test specified\" && exit 1"
9     },
10    "keywords": [],
11    "author": "",
12    "license": "ISC",
13    "dependencies": {
14      "bcrypt": "^5.1.1",
15      "body-parser": "^1.20.2",
16      "cors": "^2.8.5",
17      "express": "^4.18.2",
18      "http": "^0.0.1-security",
19      "mongoose": "^7.6.1",
20      "socket.io": "^4.7.2",
21      "uuid": "^9.0.1"
22    }
23  }
24
```

## 11. SCREENSHOTS OR DEMO:

- Providing screenshots or a link to a demo to showcase the application.

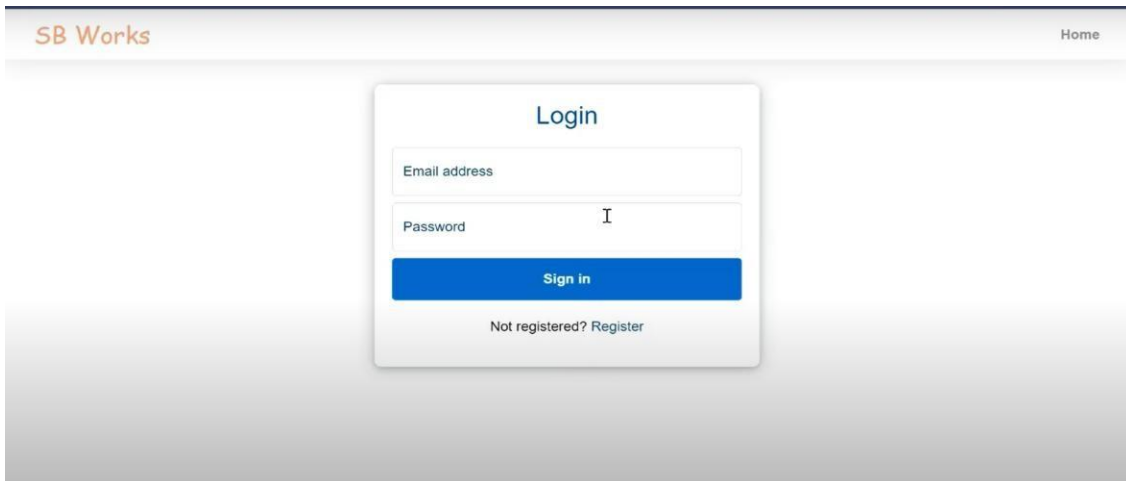
### PROJECT IMPLEMENTATION;

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.

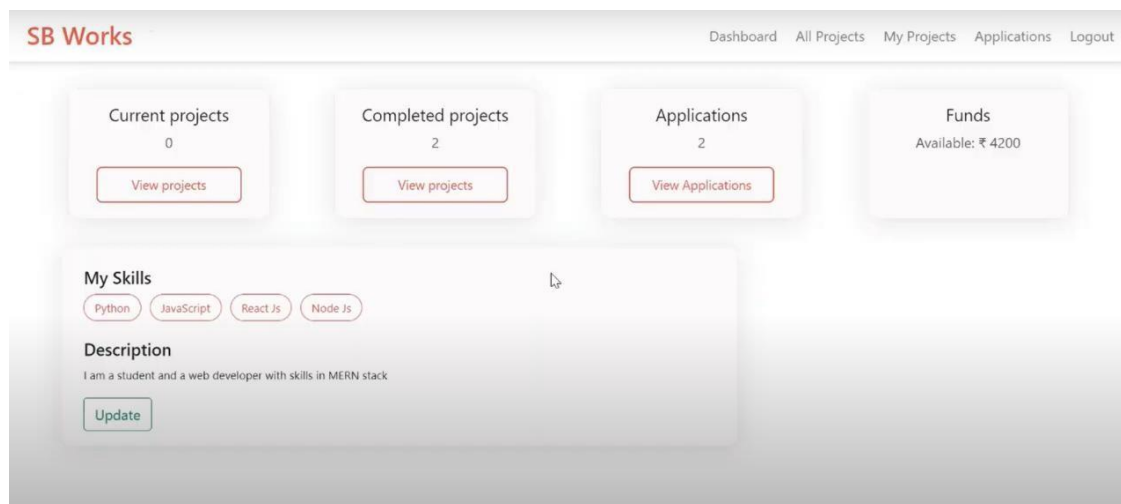
#### Landing page:



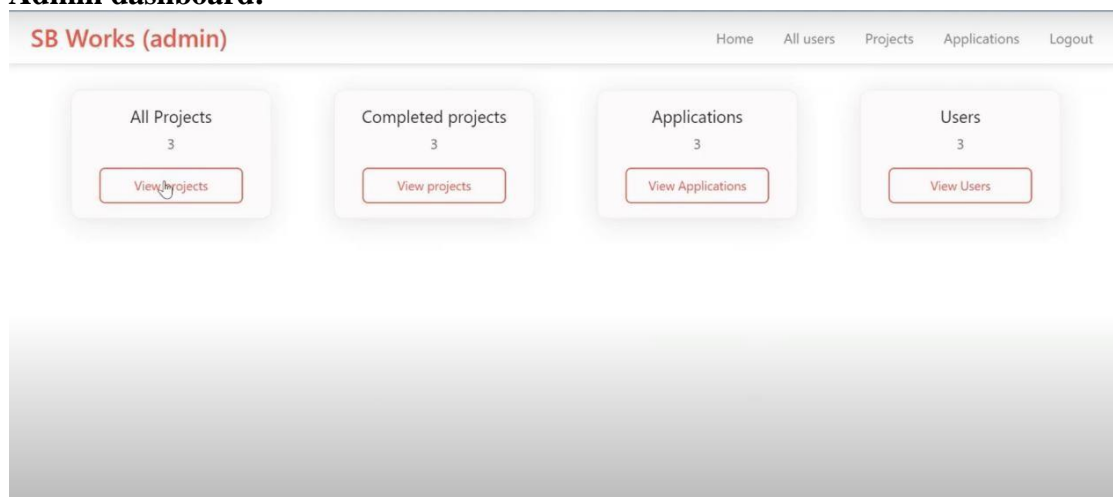
#### Authentication:



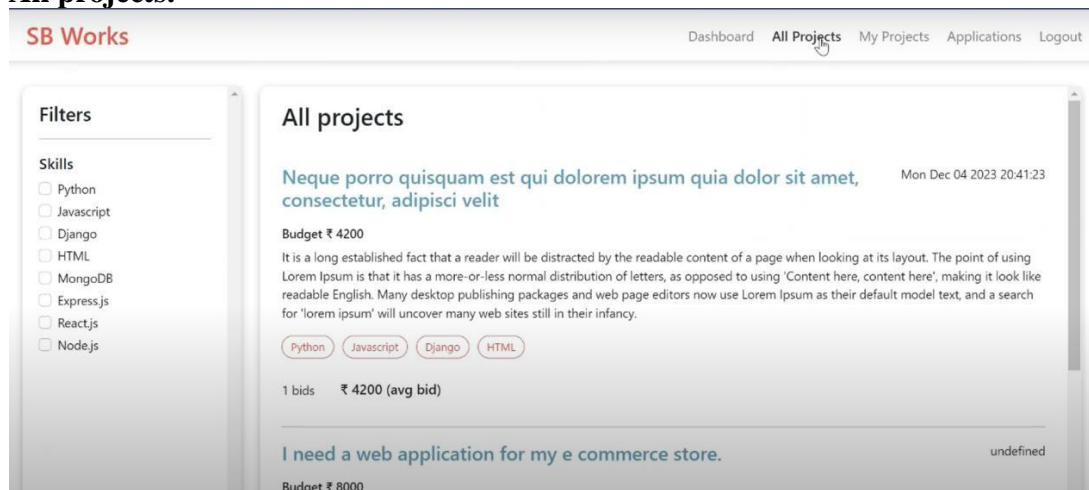
#### Freelancer dashboard:



## Admin dashboard:



## All projects:





## Freelance projects:

**SB Works**

DashboardAll ProjectsMy ProjectsApplicationsLogout

### My projects

Completed

#### I need a web application for my e commerce store.

**Budget - ₹ 8000**

I am seeking an experienced and skilled web developer to create a robust and user-friendly e-commerce web application for my online store. The ideal candidate should be proficient in the MERN (MongoDB, Express.js, React.js, Node.js) stack and have a proven track record of delivering high-quality web applications.

**Status - Completed**

#### Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit

**Budget - ₹ 4200**

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy.

**Status - Completed**

## Applications:

**SB Works**

DashboardAll ProjectsMy ProjectsApplicationsLogout

### My Applications

#### Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy.

**Skills**

Python JavaScript Django HTML

**Budget - ₹ 4500**

#### Proposal

I hope this proposal finds you well. We appreciate the opportunity to bid on the development of your e-commerce web application. Our team of experienced and skilled web developers is well-versed in the MERN (MongoDB, Express.js, React.js, Node.js) stack and has a proven track record of delivering high-quality web applications.

**Skills**

Python JavaScript React Js Node Js

**Proposed Budget - ₹ 4200**

**Status: Accepted**

#### I need a web application for my e commerce store.

I am seeking an experienced and skilled web developer to create a robust and user-friendly e-commerce web application for my online store. The ideal candidate should be proficient in the MERN (MongoDB, Express.js, React.js, Node.js) stack and have a proven track record of delivering high-quality web applications.

#### Proposal

I hope this proposal finds you well. We appreciate the opportunity to bid on the development of your e-commerce web application. Our team of experienced and skilled web developers is well-versed in the MERN (MongoDB, Express.js, React.js, Node.js) stack and has a proven track record of delivering high-quality web applications.

## Project page:

**SB Works**

DashboardAll ProjectsMy ProjectsApplicationsLogout

### Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy.

**Required skills**

Python Javascript Django HTML

**Budget**

₹ 4200

**Submit the project**

Project completed

#### Chat with the client

Hello!  
12-04 - 16:00:30

how are you?  
12-04 - 16:01:03

I'm doing great!  
12-04 - 16:02:51

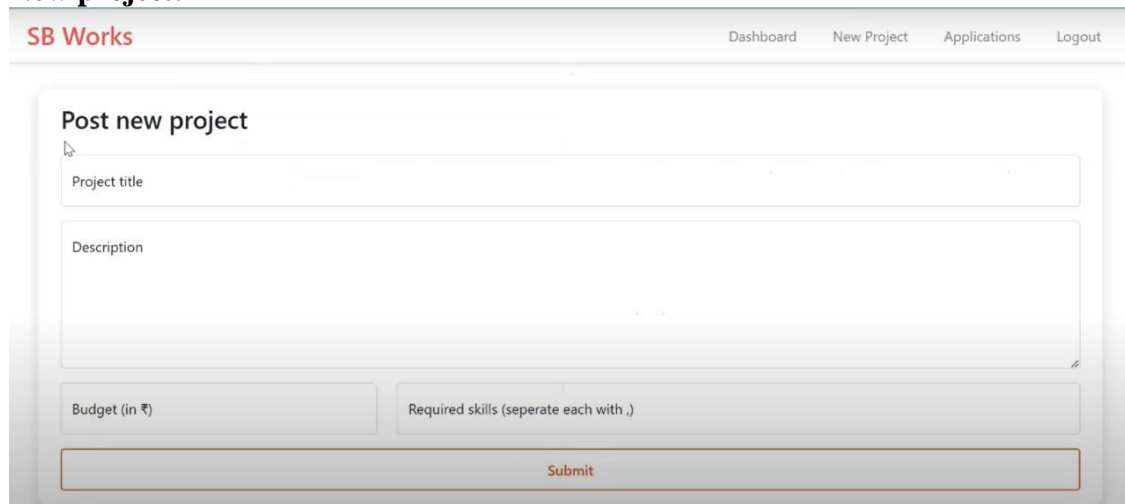
Ohh reallyyyy?  
12-04 - 16:03:07

Enter something...

Send



## New project:



The screenshot shows the 'Post new project' form in the SB Works application. The form is titled 'Post new project' and includes the following fields:

- Project title**: A text input field.
- Description**: A large text area for the project description.
- Budget (in ₹)**: A text input field for the budget.
- Required skills (separate each with ,)**: A text input field for listing required skills.
- Submit**: A button to submit the form.

The form is located on the 'New Project' page, which is part of the SB Works application. The navigation bar at the top includes links for 'Dashboard', 'New Project', 'Applications', and 'Logout'.

## 12. KNOWN ISSUES:

### ◆ **Login Token Expiry Not Handled Well:**

If your login token expires, the system doesn't automatically log you out or show a clear message. You may just see things stop working until you refresh the page or log in again.

### ◆ **Error Messages Are Sometimes Too General:**

When something goes wrong (like a failed API call), the system sometimes says "Something went wrong" without giving details. This can make it hard to understand what the problem is.

### ◆ **Profile Picture Upload Doesn't Work Yet:**

Even though there's an option to upload a profile picture, the backend doesn't actually save it right now, so your image won't appear.

### ◆ **No Pagination in Job or Bid Lists:**

When there are a lot of jobs or bids, everything is loaded at once. This can make the page slow or hard to use if the data grows too large.

### ◆ **Duplicate Bids Can Happen Sometimes:**

If two freelancers bid on a job at the same moment, the system might save both bids even if they're nearly identical.

### ◆ **No Email Confirmation on Signup:**

Users can sign up without verifying their email. This can lead to fake or duplicate accounts being created.

### ◆ **Wrong Time Shown on Messages:**

Messages always show time in UTC (world time), not your local time. This might confuse users who are in different time zones.

### 13. FUTURE ENHANCEMENTS:

- **Email Verification and Password Reset:**  
Add email verification during signup and a "Forgot Password" feature to enhance security and user trust.
- **Profile Image Upload Support:**  
Implement backend support for uploading and storing user profile pictures using cloud storage (like AWS S3 or Cloudinary).
- **Real-Time Chat with Web Sockets:**  
Upgrade the messaging system to support real-time chat using Web Sockets or Socket.IO for faster and smoother communication.
- **Admin Dashboard:**  
Build an admin panel to monitor user activity, manage reported content, and approve or block suspicious accounts.
- **Pagination and Filtering:**  
Add pagination, sorting, and filtering on job and bid listings to improve performance and usability for users with large datasets.
- **Freelancer Rating & Review System:**  
Allow clients to rate freelancers and leave reviews after completing jobs, helping others make informed decisions.
- **Job Status Workflow:**  
Introduce job status tracking (e.g., Open, In Progress, Completed, Cancelled) to give better project visibility to both clients and freelancers.
- **Notifications System:**  
Add in-app and email notifications for new messages, job updates, and bid responses to keep users engaged and informed.
- **Multi-language Support:**  
Enable localization and translation features to expand access for users from different countries and regions.
- **Mobile App (React Native or Flutter):**  
Develop a mobile version of Freelance Finder to allow users to manage jobs and communicate on the go.