

PROJECT WORK REPORT ON
Freelance Finder: Discovering Opportunities,Unlocking Potential

Team ID:LTVIP2025TMId53643

Team size: 4

Team leader :Madhu Dasi

Team member :M Sai Surya Nithik

Team member: M Devisatyasri

Team member: M Bhanu

UNDER THE GUIDENCE OF

Dr Shaik Salma Begum



SESHADRI RAO GUDLAVALERU ENGINEERING COLLEGE

DECLARATION

We here by declare that

- The work contained in this report is original and has been done by us under the guidance of our supervisor(s).
- The work has not been submitted to any other institute for obtaining any degree or diploma.
- We have followed the guidelines provided by the institute in preparing there port.
- We have conformed to the norms and guidelines given in the ethical code of conduct of the ins titute.
- When ever we have used materials (data, theoretical analysis, figures, and text)from other sources, we have given due credit to them by citing in the text of the report and giving their details in the references.

Team ID:LTVIP2025TMId53643

FULL STACK DEVELOPMENT with MERN

Project Documentation Format

1. Introduction

- **Project Title:** [Your Project Title]
- **Team Members:** List team members and their roles

2. Project Overview

- **Purpose:** Briefly describe the purpose and goals of the project.
- **Features:** Highlight key features and functionalities.

3. Architecture

- **Frontend:** Describe the frontend architecture using React.
- **Backend:** Outline the backend architecture using Node.js and Express.js.
- **Database:** Detail the database schema and interactions with MongoDB.

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.

5. Folder Structure

- **Client:** Describe the structure of the React frontend.
- **Server:** Explain the organization of the Node.js backend.

6. Running the Application

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

7. API Documentation

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

8. Authentication

- Explain how authentication and authorization are handled in the project.
- Include details about tokens, sessions, or any other methods used.

9. User Interface

- Provide screenshots or GIFs showcasing different UI features.

10. Testing

- Describe the testing strategy and tools used.

11. Screenshots or Demo

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

12. Known Issues

- Document any known bugs or issues that users or developers should be aware of.

13. Future Enhancements

- Outline potential future features or improvements that could be made to the project.

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.

Description

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.

PROJECT OVERVIEW

Purpose:

The purpose of Freelance Finder is to create a dynamic online platform that connects freelancers with clients. It enables skilled professionals to find job opportunities and clients to hire talent efficiently across various domains such as design, development, writing, and more.

✦✦ Key Features:

1. User Authentication

- Secure sign-up and login using JWT tokens
- Role-based access for freelancers and clients

2. Freelancer & Client Profiles

- Freelancers can create detailed profiles showcasing their skills, experience, and portfolio
- Clients can post jobs, view applicants, and hire suitable candidates

3. Job Posting & Bidding System

- Clients can create job posts with deadlines and budgets
- Freelancers can apply/bid on jobs with proposals

4. Search & Filter

- Advanced filtering to find freelancers or jobs by skills, category, budget, or rating

5. Messaging System

- In-app chat between freelancers and clients for better communication

6. Dashboard & Notifications

- Personalized dashboards for freelancers and clients
- Real-time updates and notifications for job status, applications, and messages

7. Responsive Design

- Fully mobile-friendly UI using Bootstrap and React.js

8. Admin Panel

- Admin can manage users, jobs, disputes, and platform analytics

➤ Tech Stack Breakdown:

Layer	Technology
Frontend	React.js, Bootstrap, HTML, CSS, JavaScript
Backend	Node.js, Express.js
Database	MongoDB
Authentication	JWT, bcrypt
Hosting (optional)	Render / Vercel (Frontend), MongoDB Atlas(DB),Railway/Render(Backend)

Project Architecture

1. Frontend (Client-Side)

- Built using React.js, styled with Bootstrap and custom CSS. Key UI components include:
- Home Page – Showcases the platform's features and value.
- Login/Register Pages – For freelancers and clients.
- Dashboard – Personalized area for users to manage profiles, projects, and messages.
- Search & Filter – Browse freelancers/projects by skill, category, location, etc.
- Project Bidding/Apply – Freelancers can apply to posted jobs.

> Frontend Stack:

- B React.js + Bootstrap + Axios (for API calls) + React Router

2. Backend (Server-Side)

- Built using Node.js with Express.js framework. Key modules include:
- Authentication – JWT-based login/registration system.
- User Management – Profiles for freelancers and clients.
- Project Management – CRUD operations for jobs/projects.
- Chat System (optional) – Real-time or asynchronous messaging using Socket.IO or REST APIs.

> Backend Stack:

- Node.js + Express.js + JWT + bcrypt (for password encryption)

3. Database (NoSQL)

- Managed using MongoDB with collections such as:
- users (freelancers, clients)
- projects (job posts, bids)

- applications (job applications)
- messages (for chats)

> **Tools/ODM:**

- Mongoose for schema design and interaction with MongoDB.

Features Summary

- ✓ User Authentication (Freelancer/Client)
- ✓ Project Posting & Bidding
- ✓ Profile Customization
- ✓ Project Search & Filtering
- ✓ Real-Time Notifications (Optional)
- ✓ Chat Between Clients and Freelancers (Optional)

Goal

- To provide a seamless platform where freelancers can find genuine opportunities and clients can hire the right talent — unlocking true potential on both sides.

Scenario based case-study

Sarah, a recent graduate with a degree in graphic design, is eager to showcase her skills and build a strong freelance portfolio. She stumbles upon SB Works while searching for online freelancing opportunities.

Finding the Perfect Project: Impressed by the user-friendly interface, Sarah browses through various project categories. She discovers a project posted by a local bakery, "Sugar Rush," seeking a logo redesign. The project description details the bakery's brand identity and target audience, giving Sarah a clear understanding of the client's needs.

Bidding with Confidence: Confident in her design skills, Sarah dives into the project details. SB Works allows her to review the bakery's

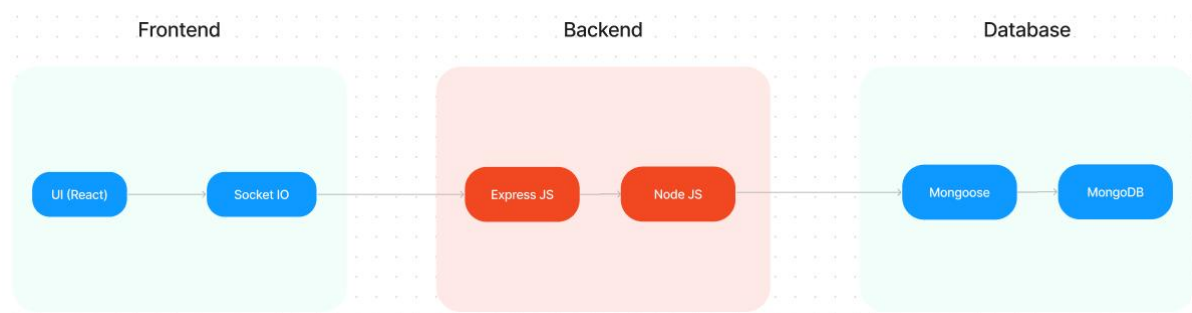
previous marketing materials, further solidifying her design approach. She submits a compelling proposal highlighting her relevant experience and attaching a few samples from her portfolio stored securely within the platform.

Communication & Collaboration: "Sugar Rush" selects Sarah's proposal based on her impressive portfolio and competitive pricing. SB Works facilitates seamless communication between Sarah and the bakery, allowing them to discuss project specifics and refine the design direction through an integrated chat system.

Delivery & Feedback: Once finalized, Sarah submits her logo design through the SB Works platform. "Sugar Rush" can review the design, provide feedback, and request minor revisions if needed. SB Works fosters a collaborative environment where both parties can work towards achieving the desired outcome.

Building a Thriving Career: Following a successful project completion and a glowing review from "Sugar Rush," Sarah's profile on SB Works gains traction. The positive experience encourages her to actively seek new projects on the platform. With a growing portfolio and strong client testimonials, Sarah is well on her way to establishing a thriving freelance career on SB Works.

TECHNICAL 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.

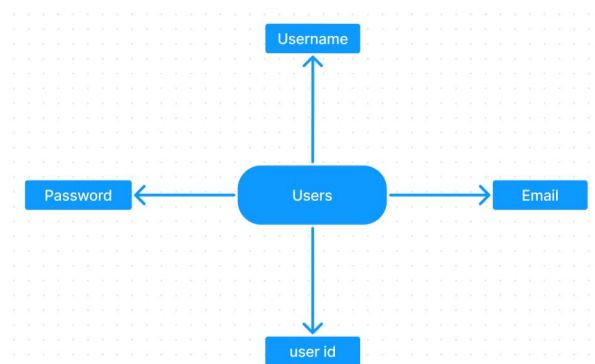
To enhance the user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a real-time and visually appealing interface for users.

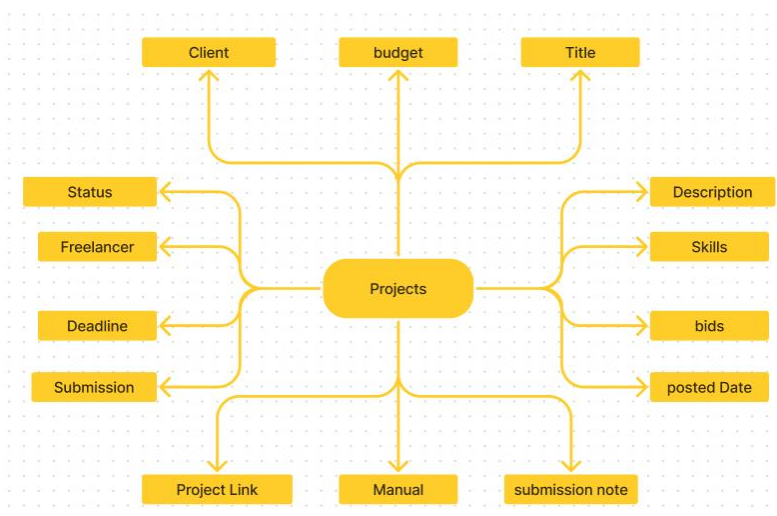
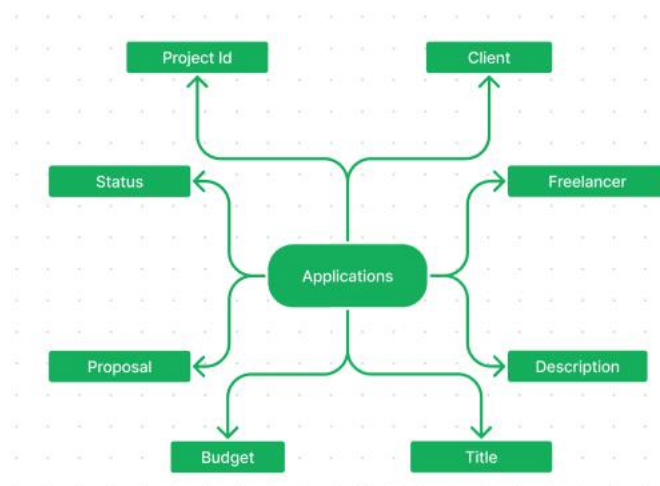
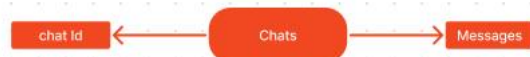
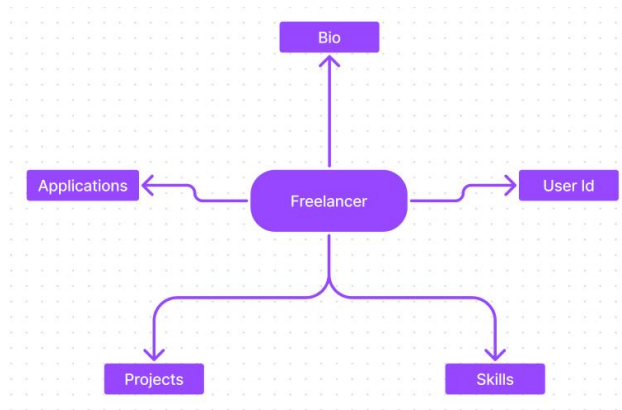
On the backend, we utilize the Express Js framework to manage server-side logic and communication. Express Js provides a robust foundation for handling requests and responses efficiently.

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.

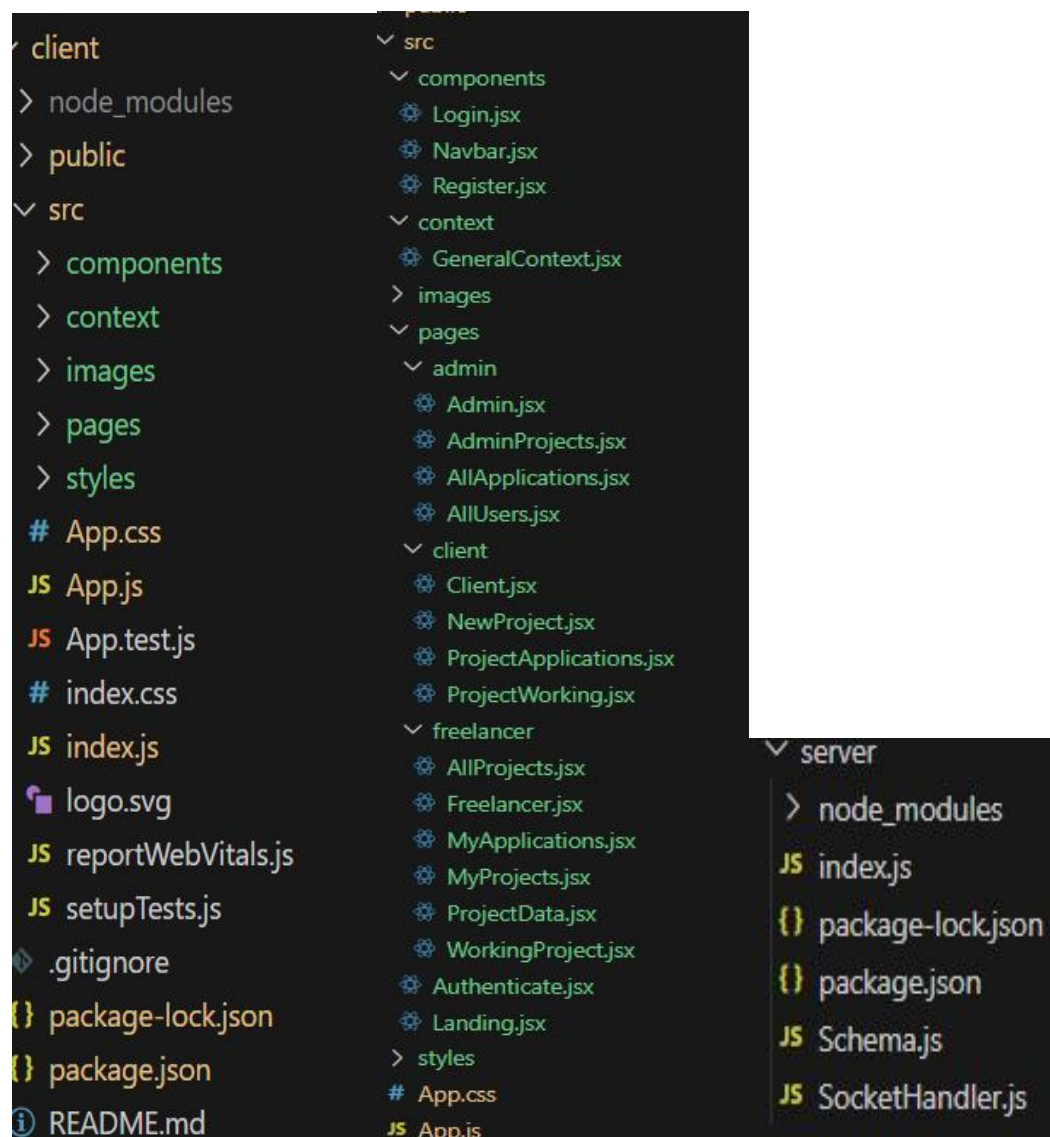
ER DIAGRAM





SB Works 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 SB Works a go-to platform for both clients and freelancers.

PROJECT 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.

SETUP INSTRUCTIONS

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: <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:

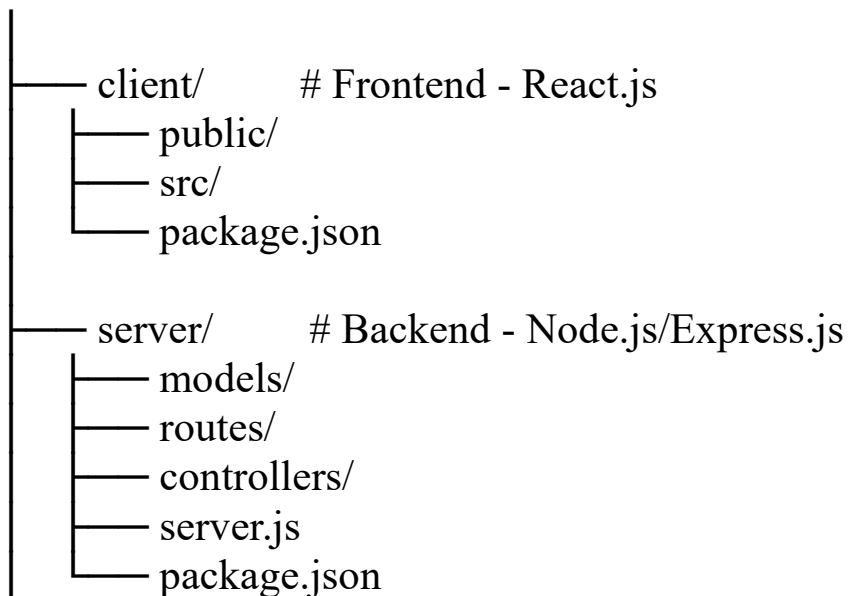
```
npm start
```

- The SB Worksapp will be accessible at <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.

➤ Folder Structure

FreelanceFinder/



RUNNING THE APPLICATIONS

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.

Project Flow:

Use the code

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

Demo video:

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

Milestone 1: Project setup and configuration.

✓ Folder setup:

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

- Client folders.
- Serverfolders

✓ Installation of required tools:

1. Open the frontend folder to install necessary tools

For frontend, we use:

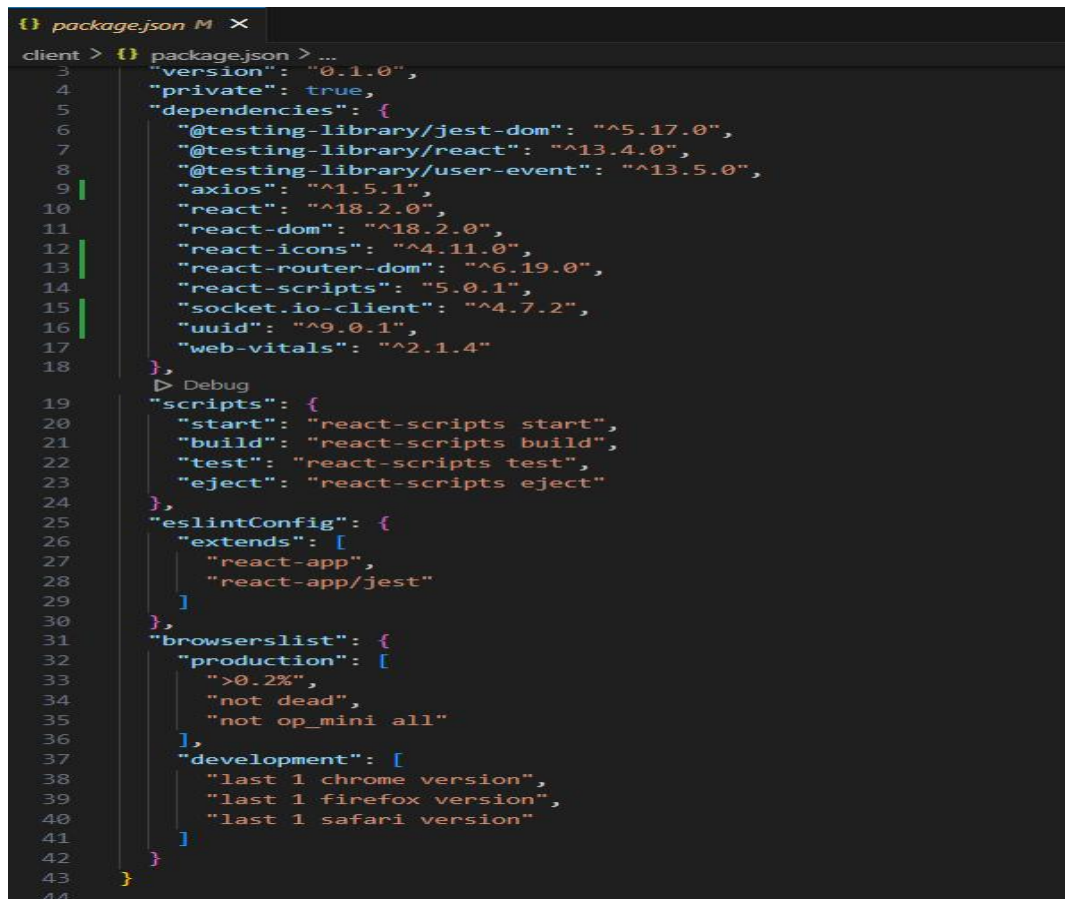
- React
- Bootstrap
- Material UI
- Axios
- react-bootstrap

2. Open the backend folder to install necessary tools

For backend, we use:

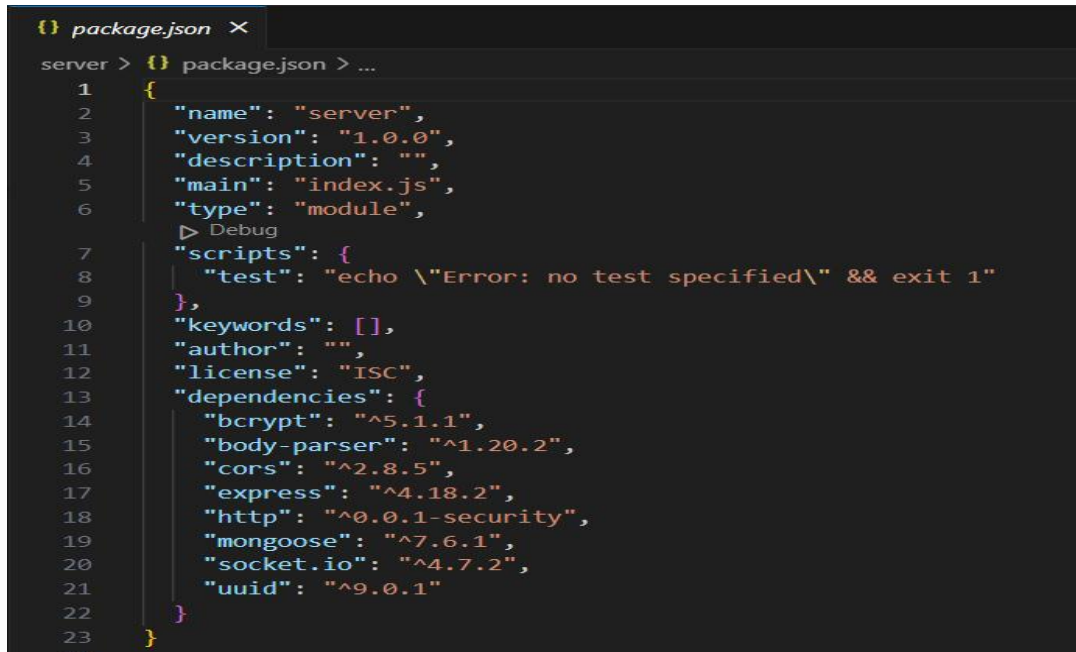
- Express Js
- Node JS
- MongoDB
- Mongoose
- Cors
- Bcrypt

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



```
client > {} package.json M X
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 }
```

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' (server), 'version' (1.0.0), 'description' (empty), 'main' (index.js), 'type' (module), 'scripts' (test: echo \"Error: no test specified\" && exit 1), 'keywords' (empty array), 'author' (empty), 'license' (ISC), and 'dependencies' (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, uuid: ^9.0.1). The code is syntax-highlighted and line numbers are visible on the left side of the editor.

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "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",
    "uuid": "^9.0.1"
  }
}
```

Milestone 2: 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

Reference video:

<https://drive.google.com/file/d/1zrOMSp6svjH1tRcul3b442XVPNyKTSp4/view?usp=sharing>

Milestone 3: Database development

- 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.

```
const server = http.createServer(app);

const io = new Server(server, {
  cors: {
    origin: '*',
    methods: ['GET', 'POST', 'PUT', 'DELETE']
  }
});

io.on("connection", (socket) =>{
  console.log("User connected");

  SocketHandler(socket);
})

const PORT = 6001;

mongoose.connect('mongodb://localhost:27017/Freelancing',{
  useNewUrlParser: true,
  useUnifiedTopology: true
}).then(()=>{
```



```

server.listen(PORT, ()=>{
  console.log(`Running @ ${PORT}`);
});
}).catch((e)=> console.log(`Error in db connection ${e}`));

```

The Schemas for the database are given below

```

JS Schemajs X
server > JS Schemajs > [0] projectSchema > submissionDescription
1 import mongoose, { Schema, mongo } from "mongoose";
2
3 const userSchema = mongoose.Schema({
4   username: {
5     type: String,
6     require: true
7   },
8   email: {
9     type: String,
10    require: true,
11    unique: true
12  },
13  password: {
14    type: String,
15    require: true
16  },
17  usertype:{
18    type: String,
19    require: true
20  }
21 })
22
23 const freelancerSchema = mongoose.Schema({
24   userId: String,
25   skills: {
26     type: Array,
27     default: []
28   },
29   description: {
30     type: String,
31     default: ""
32   },
33   currentProjects: {
34     type: Array,
35     default: []
36   },
37   completedProjects: {
38     type: Array,
39     default: []
40   },
41   applications: {
42     type: Array,
43     default: []
44   },
45   funds: {
46     type: Number,
47     default: 0
48   },
49 })
50

```

```

51
52 const projectSchema = mongoose.Schema({
53   clientId: String,
54   clientName: String,
55   clientEmail: String,
56   title: String,
57   description: String,
58   budget: Number,
59   skills: Array,
60   bids: Array,
61   bidAmounts: Array,
62   postedDate: String,
63   status: {
64     type: String,
65     default: "Available"
66   },
67   freelancerId: String,
68   freelancerName: String,
69   deadline: String,
70   submission: {
71     type: Boolean,
72     default: false
73   },
74   submissionAccepted: {
75     type: Boolean,
76     default: false
77   },
78   projectLink: {
79     type: String,
80     default: ""
81   },
82   manulalink: {
83     type: String,
84     default: ""
85   },
86   submissionDescription: {
87     type: String,
88     default: ""
89   },
90 })

```

```

92
93 const applicationSchema = mongoose.Schema({
94
95   projectId: String,
96   clientId: String,
97   clientName: String,
98   clientEmail: String,
99   freelancerId: String,
100   freelancerName: String,
101   freelancerEmail: String,
102   freelancerSkills: Array,
103   title: String,
104   description: String,
105   budget: Number,
106   requiredSkills: Array,
107   proposal: String,
108   bidAmount: Number,
109   estimatedTime: Number,
110   status: {
111     type: String,
112     default: "Pending"
113   }
114 })
115
116 const chatSchema = mongoose.Schema({
117   _id: {
118     type: String,
119     require: true
120   },
121   messages: {
122     type: Array
123   }
124 })
125
126 export const User = mongoose.model('users', userSchema);
127 export const Freelancer = mongoose.model('freelancer', freelancerSchema);
128 export const Project = mongoose.model('projects', projectSchema);
129 export const Application = mongoose.model('applications', applicationSchema);
130 export const Chat = mongoose.model('chats', chatSchema);

```

Milestone 4: 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.

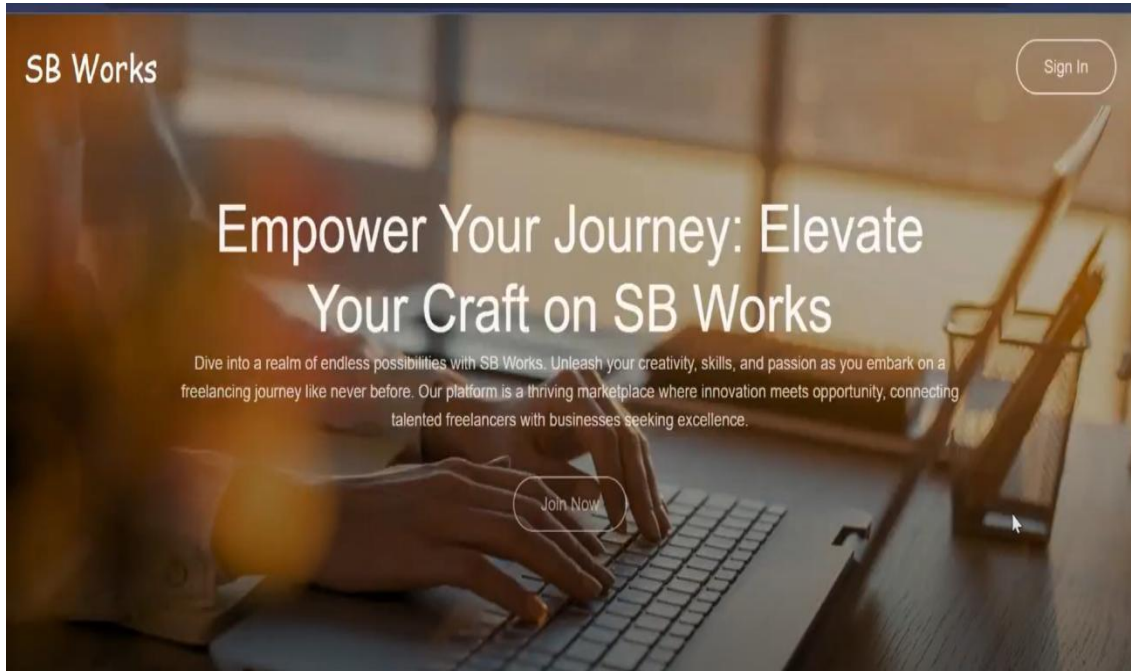
Reference video:

<https://drive.google.com/file/d/16o7iRxHfaKKHJkep1ixKs86C3nfNACVQ/view?usp=sharing>

Milestone 5: 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:

SB Works

Home

Login

Email address

Password

Sign in

Not registered? Register

Freelancer dashboard:

SB Works

DashboardAll ProjectsMy ProjectsApplicationsLogout

Current projects

0

View projects

Completed projects

2

View projects

Applications

2

View Applications

Funds

Available: ₹ 4200

My Skills

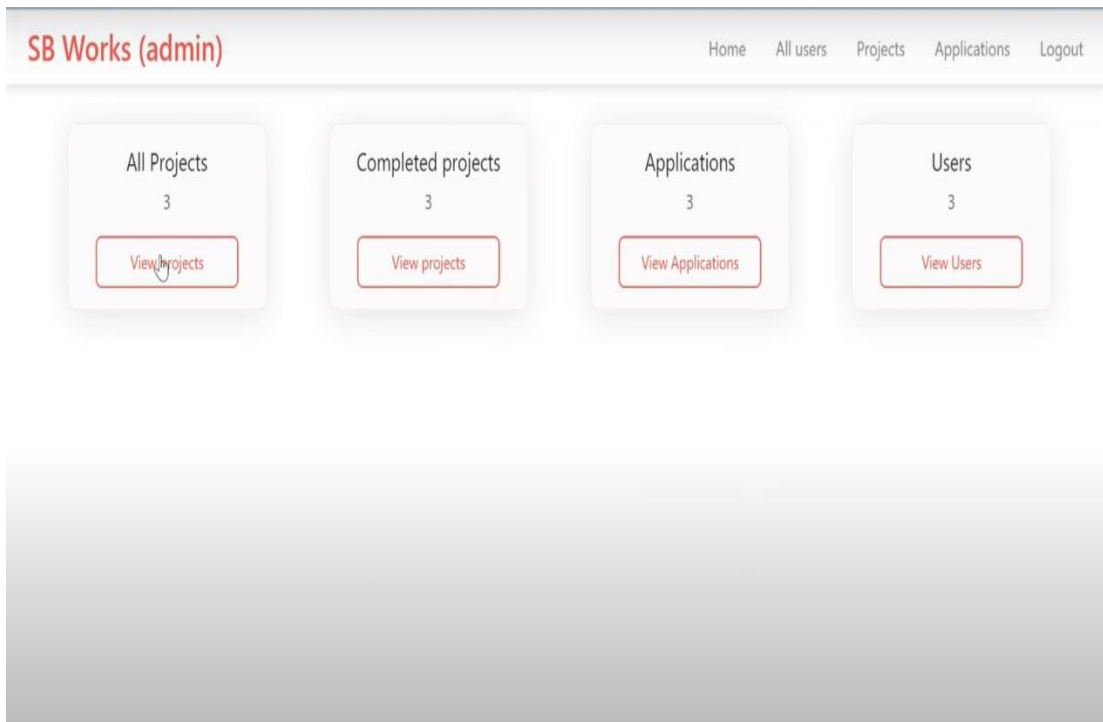
PythonJavaScriptReact JsNode Js

Description

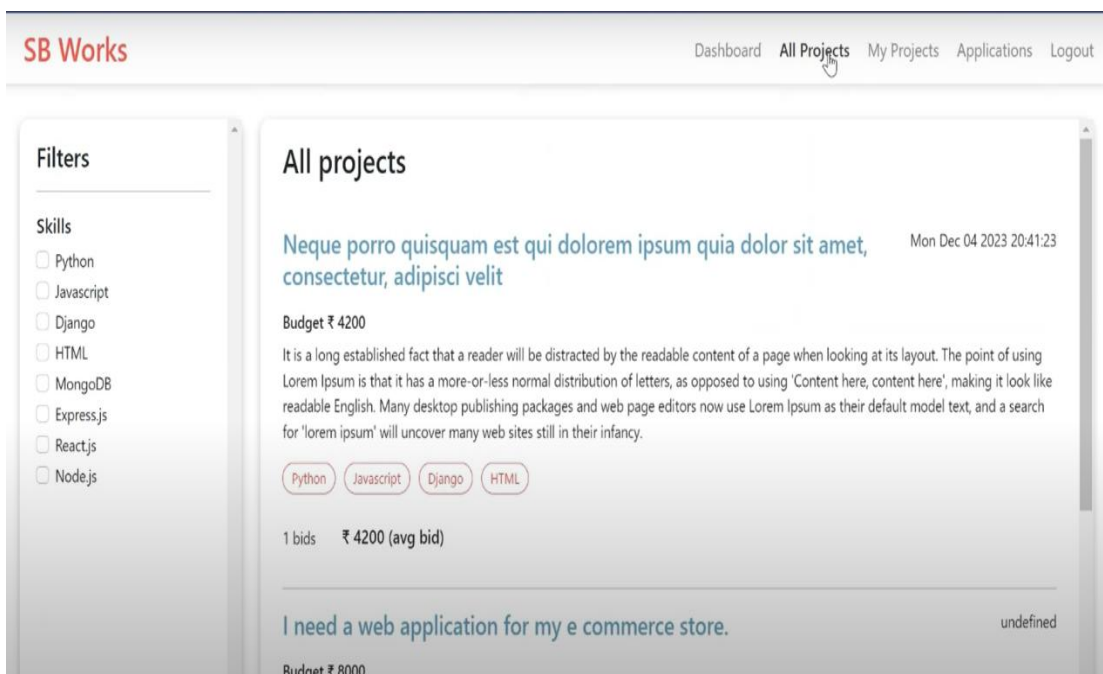
I am a student and a web developer with skills in MERN stack

Update

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

Mon Dec 04 2023 20:41:23 GMT+0530 (India Standard Time)

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

PythonJavaScriptDjangoHTML

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

PythonJavaScriptReact JsNode Js

Proposed Budget - ₹ 4200

Status: Accepted

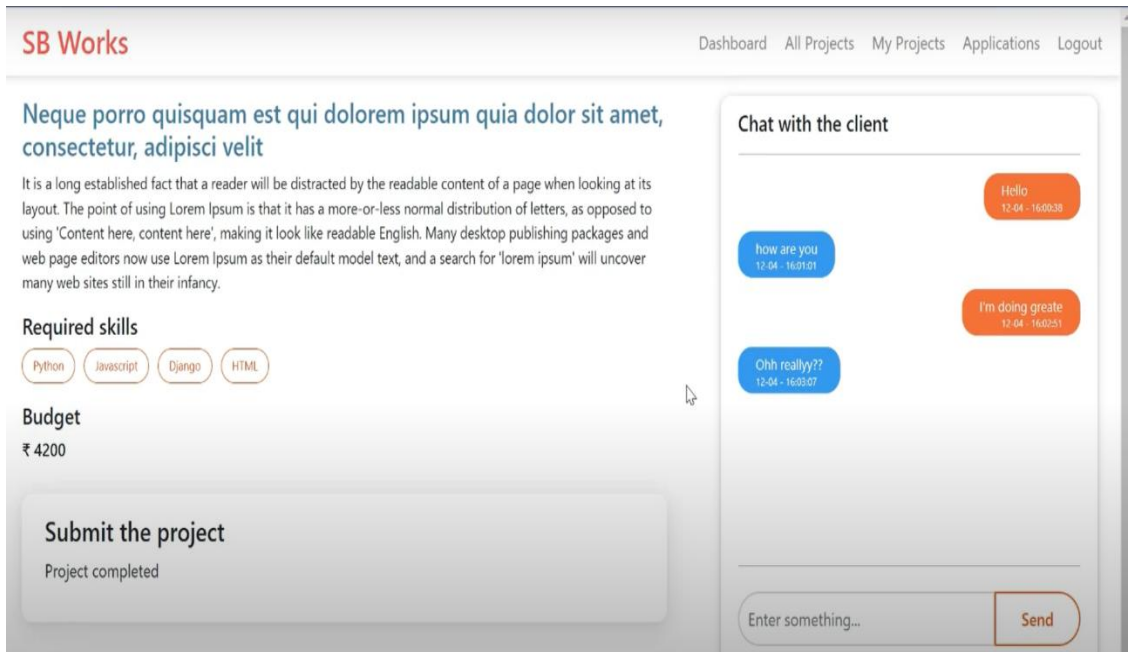
I need a web application for my e commerce store.

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:



New project:

The screenshot shows the 'Post new project' form in the SB Works dashboard. The navigation bar includes Dashboard, New Project, Applications, and Logout. The form is titled 'Post new project' and contains the following fields: 'Project title' (a text input field), 'Description' (a large text area), 'Budget (in ₹)' (a text input field), and 'Required skills (seperate each with ,)' (a text input field). A 'Submit' button is located at the bottom of the form.

For any further doubts or help, please consider the code in the drive link given below,

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

The demo of the app is available at:

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

USER INTERFACE

1. User Authentication UI

- Login Page
- **Fields:** Email, Password
- **Role selection:** Freelancer / Client
- Signup Page
- **Input:** Name, Email, Password, Role
- Validations and confirmation messages
- Forgot Password (Optional)

2. Landing / Home Page

- Welcome message
- Overview of platform features
- CTA buttons: “Find Freelancers”, “Post a Job”, “Join Now”

3. Freelancer Dashboard

Overview of:

- Jobs applied
- Saved jobs
- Profile completion status

- Sidebar navigation (Profile, Messages, Jobs, Logout)

4. Client Dashboard

Overview of:

- Jobs posted
- Applications received
- Hired freelancers
- Option to Post New Job

5. Job Listings Page

❖ **List of available jobs with:**Title, Budget, Tags, Posted date

❖ **Filters:**

- Skill, Experience level, Budget range, Location
- “Apply” button for freelancers

6. Job Post Form (Client)

❖ **Inputs:**

- Title, Description, Required Skills, Budget, Deadline
- Form validation and preview option

7. Job Details Page

❖ **Detailed view of job when clicked:**

- Description, Requirements, Budget
- “Apply Now” (Freelancer) or “Edit/Delete” (Client)

8. Profile Page

❖ **Editable for Freelancers:**

- Bio, Skills, Portfolio, Profile picture
- Clients can edit company info or view posted jobs

9. Messaging Interface

- (Optional, but valuable)
- Chat between freelancer and client
- Message history with timestamps
- Unread message indicators

10. Notification Center

❖ **Alerts for:**

- Job application updates
- New messages
- Profile views or shortlists

11. Settings Page

- Change password
- Toggle dark mode / light mode
- Manage notification preferences

❖ 12. Responsive Mobile UI

- All UIs must work well on mobile
- Use Bootstrap grid and collapsible menus
- Mobile-friendly forms and buttons

KNOWN ISSUES

1. Authentication & Session Handling

- Sessions expire unexpectedly after refresh (if JWT token isn't stored or refreshed properly)
- No "Remember Me" option — users must log in every time
- Role-based access control might be bypassed without proper backend checks

2. API Latency / Errors

- Occasionally slow response from the backend if MongoDB queries are not optimized
- No retry mechanism for failed API calls
- Incomplete error handling for API failures (e.g., blank page on server crash)

3. Page Reload / React State Loss

- React app loses state (like filters, search results) on page refresh if not using localStorage or context persist

- Redirection after login might not go to intended dashboard (hardcoded routes)

4. Responsiveness Bugs

- Some Bootstrap components might break on smaller screens
- Mobile view lacks proper alignment for cards and buttons

5. File Upload Issues (If Implemented)

- Resume or portfolio uploads may fail due to:
- Backend file size limit
- No validation for file type (e.g., PDF only)

6. Security Concerns

- Passwords stored without proper hashing (if bcrypt is not used)
- Forms vulnerable to XSS or SQL injection (if input validation is weak)
- No rate-limiting on login route → vulnerable to brute-force attacks

7. Database-Related Issues

- No fallback or handling for MongoDB downtime
- Duplicate entries if unique constraints aren't set (e.g., same job posted multiple times)

8. Messaging System (If Included)

- Real-time chat might lag or lose connection if WebSocket isn't handled properly
- Messages not stored persistently in DB or wrong chat ID pairing

9. UI/UX Gaps

- No success/error feedback on actions (e.g., apply to job, submit form)
- Confusing navigation flow between freelancer and client sections
- No loading indicators during long requests

10. Testing Coverage

- Missing test cases for edge scenarios (e.g., empty job listings, incorrect login)
- Manual testing needed for responsive and accessibility issues

How to Document This

- In your README.md or documentation, add:

Known Issues

- Session expires on page refresh if JWT isn't persisted
- Poor responsiveness on mobile devices for dashboard layout
- MongoDB may return duplicates if proper indexing isn't set
- No real-time status for applied jobs unless page is refreshed
- API error messages not user-friendly or localize

FUTURE ENHANCEMENT

1. User Authentication Enhancements

- Implement JWT token refresh mechanism
- Add OAuth login options (Google, LinkedIn)
- Enable two-factor authentication (2FA) for added security

2. Real-Time Features

- Integrate WebSockets (e.g., Socket.IO) for real-time:
- Chat between client and freelancer
- Notifications for new jobs, messages, etc.

3. Smart Matching with AI

- ❖ **Use AI/ML algorithms to recommend jobs to freelancers based on:**
 - Skills

- Past applications
- Success rate
- Recommend freelancers to clients based on job requirements

4. Advanced Search & Filters

❖ Add dynamic filtering by:

- Category, budget, experience level
- Implement elastic search for better performance

5. Integrated Messaging System

- Build a chat system inside the app (with unread indicators, timestamps)
- Allow file attachments in messages (resumes, portfolios, etc.)

6. Multi-language & Internationalization

- Add support for multiple languages
- Include currency converter for international clients and freelancers

7. Freelancer Profiles & Portfolios

❖ Allow freelancers to:

- Showcase work samples
- Add social links (GitHub, Behance, LinkedIn)
- Clients can rate freelancers post-project

8. Analytics Dashboard

❖ Create a dashboard for:

- **Freelancers:** Applications, earnings, reviews
- **Clients:** Job postings, applicants, hiring stats

9. Progressive Web App (PWA) or Mobile App

- Convert the web app into a PWA or build native apps with React Native
- Provide mobile notifications and offline access

10. Secure Payment Integration

❖ Integrate Stripe/PayPal for:

- Payments between clients and freelancers
- Escrow system for secure transactions

CONCLUSION:

- **FreelanceFinder: Discovering Opportunities, Unlocking Potential** is a comprehensive full-stack web application designed to bridge the gap between talented freelancers and clients seeking skilled professionals. By leveraging modern web technologies such as HTML, CSS, JavaScript, Bootstrap, React.js, Node.js, and MongoDB, this platform provides a user-friendly, responsive, and scalable solution for freelance job management.
- The application offers distinct dashboards for freelancers and clients, job posting and application functionality, secure authentication, and a smooth user experience. It serves as a practical demonstration of how full-stack technologies can be integrated to build real-world applications that solve meaningful problems.
- Throughout the development process, the project demonstrated:
- Effective use of React for component-based frontend UI

- Efficient Node.js & Express backend for handling RESTful APIs
- Robust data handling with MongoDB
- Modular and maintainable code organization
- Real-world functionality like authentication, role-based access, and CRUD operations
- This project also opens up several possibilities for future enhancements, including real-time chat, AI-based job recommendations, mobile support, and secure payment integration — making it a scalable and production-ready foundation for a full-fledged freelancing platform.
- In conclusion, Freelance Finder not only showcases strong technical skills in full stack development but also demonstrates an understanding of building impactful, user-centric digital platforms.

THANK YOU