FreelanceFinder: Discovering Opportunities,

Unlocking Potential

Team Members : K.Swapn Preethi

V.Jayanth

K.Dinesh

K.Mounika

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.

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

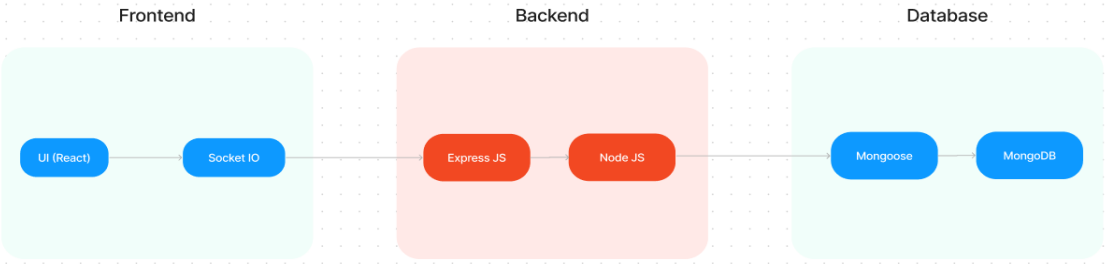
previous marketing materials, further solidifying her design approach. She submits a compelling proposal highlighting her relevant experience and attaching a few samples from her portfolio stored securely within the platform.

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.

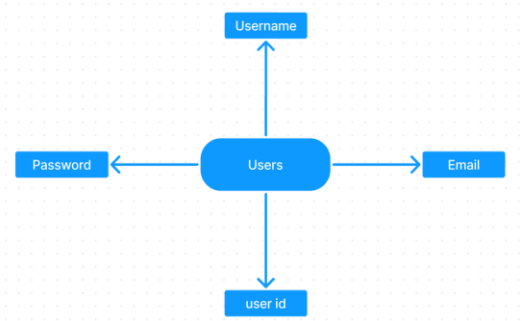
To enhance the user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a real-time and visually appealing interface for users.

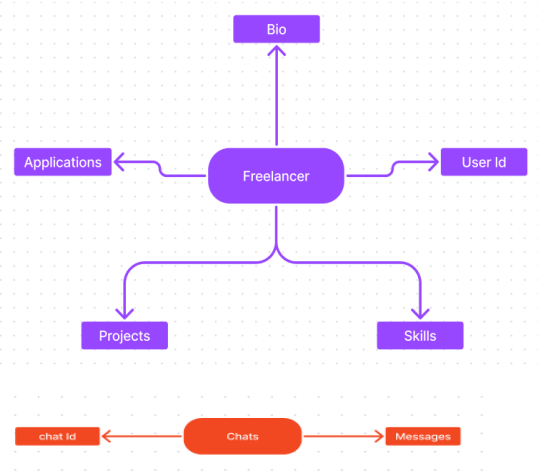
On the backend, we utilize the Express Js framework to manage server-side logic and communication. Express Js provides a robust foundation for handling requests and responses efficiently.

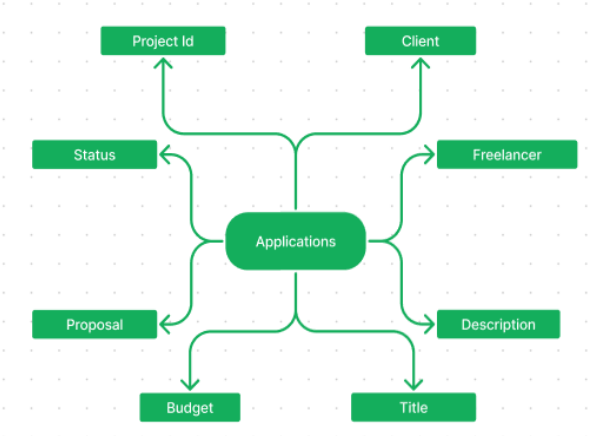
For data storage and retrieval, SB Works relies on MongoDB. MongoDB offers a scalable and efficient solution for storing various data, including user-contributed locations and images. This ensures quick and reliable access to the information needed to enrich the local tourism experience.

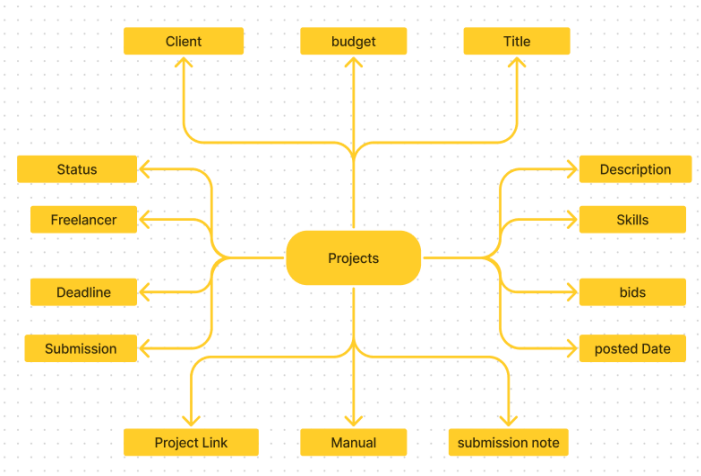
In conjunction, the frontend and backend components, complemented by Express Js, and MongoDB, together form a comprehensive technical architecture for SB Works. This architecture facilitates real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive experience for users contributing to and exploring their local surroundings.

ER DIAGRAM


Username
Password
Users
Email
user id

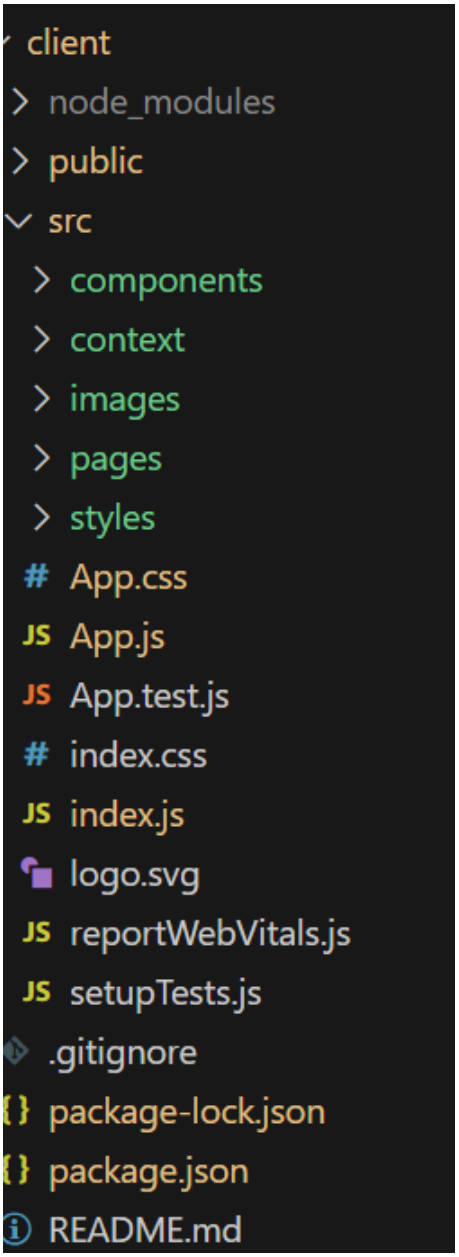

Bio
Applications
Freelancer
User Id
Projects
Skills
chat Id
Chats
Messages

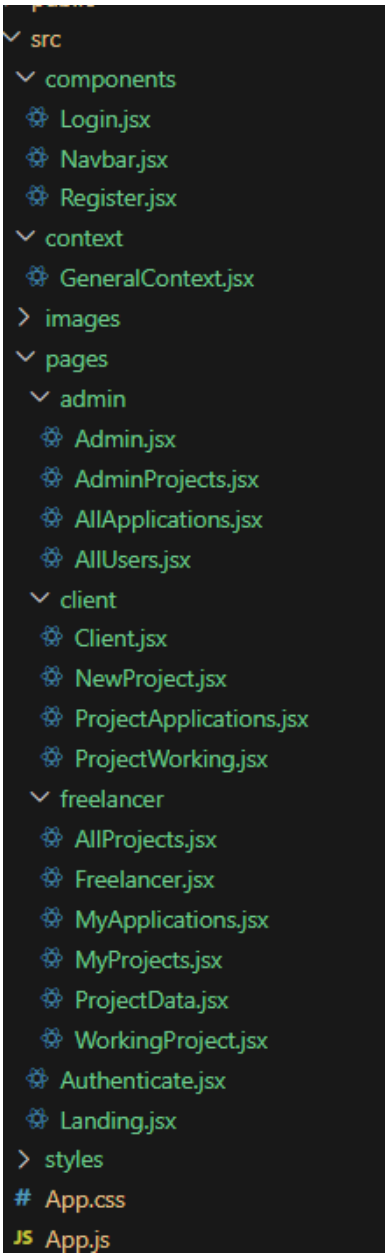

Project Id
Client
Status
Freelancer
Applications
Proposal
Description
Budget
Title

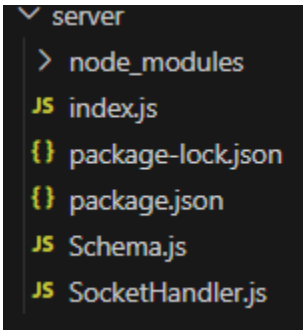


SB Works connects clients with skilled freelancers through a user-friendly platform. Clients can post projects with details and browse freelancer profiles to find the perfect match. Freelancers can submit proposals, collaborate with clients through secure chat, and securely submit work for review and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

PROJECT STRUCTURE




src
components
Login.jsx
Navbar.jsx
Register.jsx
context
GeneralContext.jsx
images
pages
admin
Admin.jsx
AdminProjects.jsx
AllApplications.jsx
AllUsers.jsx
client
Client.jsx
NewProject.jsx
ProjectApplications.jsx
ProjectWorking.jsx
freelancer
AllProjects.jsx
Freelancer.jsx
MyApplications.jsx
MyProjects.jsx
ProjectData.jsx
WorkingProject.jsx
Authenticate.jsx
Landing.jsx
styles
App.css
App.js


server
node_modules
index.js
package-lock.json
package.json
Schema.js
SocketHandler.js

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.

Software Requirements:

Node.js and npm

Express.js

MongoDB

React.js

HTML, CSS, and JavaScript:

Database Connectivity

Front-end Framework

Version Control

Application flow

Freelancer Responsibilities:

• Project Submission: Freelancers are responsible for submitting completed and high-quality work for the assigned projects through the platform.
• Compliance: Ensure that the submitted work adheres to client requirements, industry standards, and any specific guidelines outlined by the platform.
• Effective Communication: Actively engage in communication with clients, promptly responding to messages, asking clarifying questions, and providing updates on the project progress.
• Time Management: Manage time effectively to meet project deadlines and deliver work in a timely manner.
• Professionalism: Conduct oneself professionally by maintaining a respectful and cooperative attitude with clients and fellow freelancers.
• Quality Assurance: Deliver work that is accurate, well-executed, and free from errors to maintain client satisfaction.

Client Responsibilities:

• Clear Project Description: Provide a detailed and comprehensive project description, including deliverables, desired outcomes, and any specific requirements.
• Timely Communication: Respond promptly to freelancer inquiries, providing necessary information and feedback in a timely manner.
• Payment Obligations: Fulfill the agreed-upon payment terms promptly and fairly upon satisfactory completion of the project.
• Feedback and Evaluation: Provide constructive feedback and evaluate the freelancer's performance, helping them improve and providing valuable insights.

Admin Responsibilities:

- Data Oversight: As an admin, one of your key responsibilities is to monitor and ensure the integrity and security of all data on the platform
- Policy Enforcement: Admins play a crucial role in enforcing platform policies, guidelines, and ethical standards.
- Conflict Resolution: In the event of disputes or issues within the community, it is the admin's responsibility to address them promptly and impartially
- User Support and Communication: Admins should provide support and guidance to users on the platform
- Platform Maintenance and Improvement: Admins are responsible for the overall maintenance and improvement of the research platform

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.
- Server folders

- **Installation of required tools:**

1. Open the frontend folder to install necessary tools
   For frontend, we use:

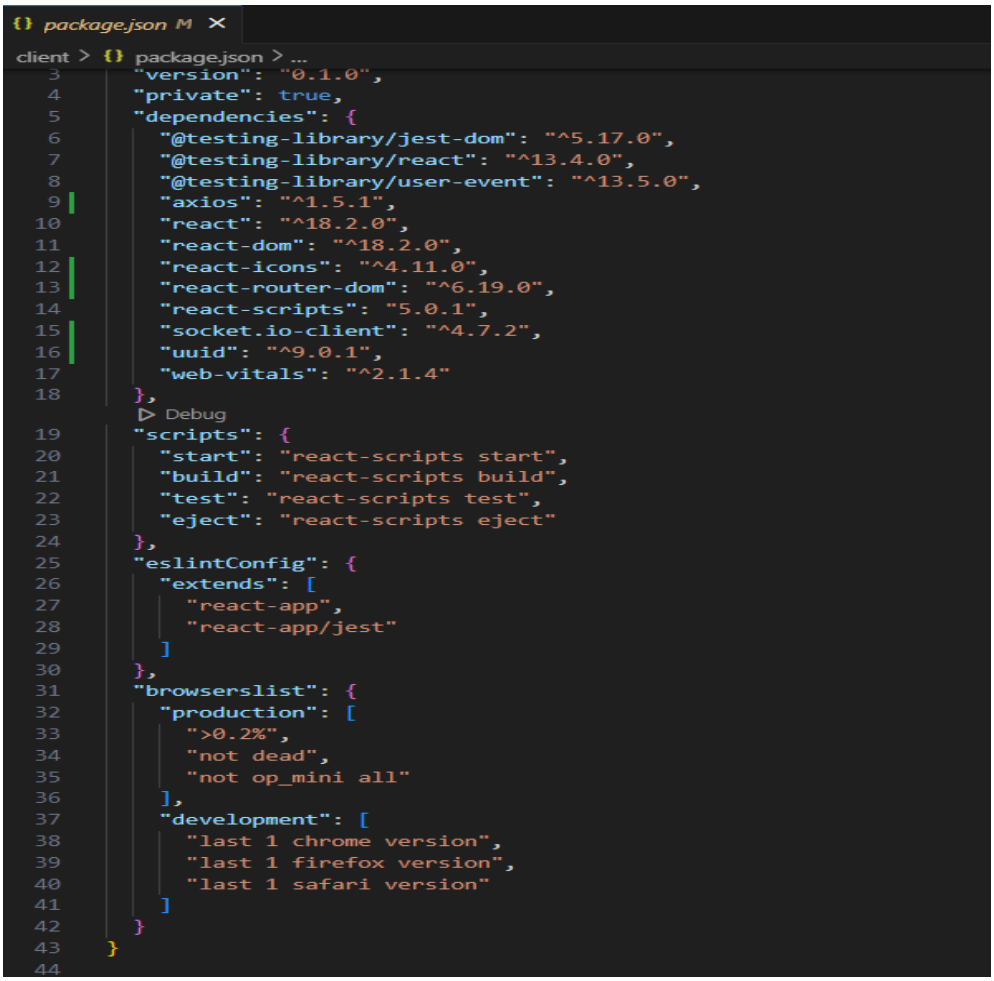
- React
- Bootstrap
- Material UI
- Axios
- react-bootstrap

2. Open the backend folder to install necessary tools

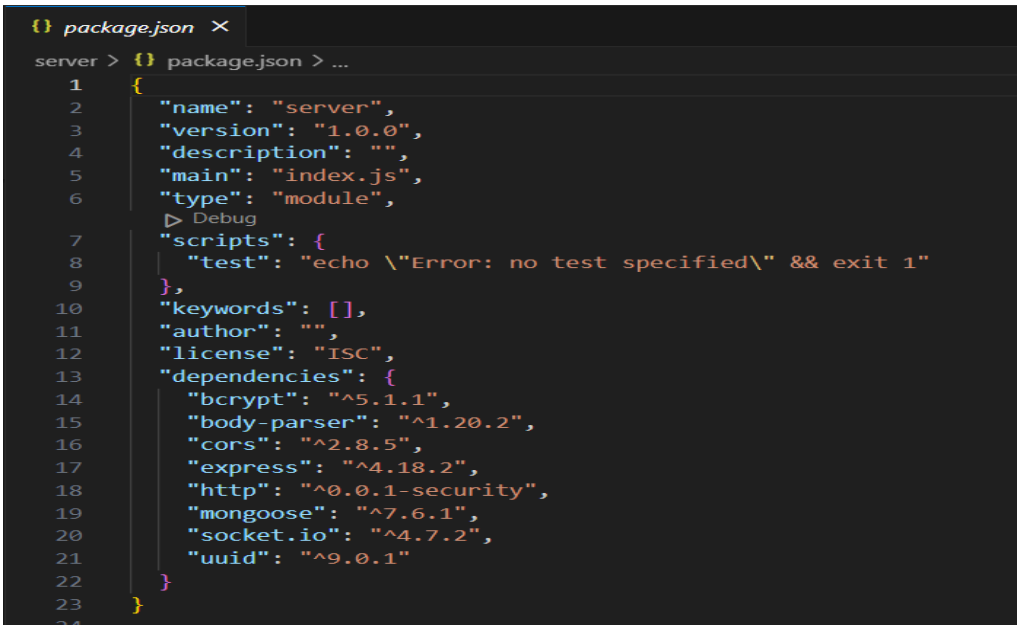
For backend, we use:

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

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



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



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

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

Schema.js

server > Schema.js > projectSchema > submissionDescription

```
import mongoose, { Schema, mongo } from "mongoose";

const userSchema = mongoose.Schema({
    username: {
        type: String,
        require: true
    },
    email: {
        type: String,
        require: true,
        unique: true
    },
    password: {
        type: String,
        require: true
    },
    usertype:{
        type: String,
        require: true
    }
})
```

```
const freelancerSchema = mongoose.Schema({
    userId: String,
    skills: {
        type: Array,
        default: []
    },
    description: {
        type: String,
        default: ""
    },
    currentProjects: {
        type: Array,
        default: []
    },
    completedProjects: {
        type: Array,
        default: []
    },
    applications: {
        type: Array,
        default: []
    },
    funds: {
        type: Number,
        default: 0
    },
})
```

```
const projectSchema = mongoose.Schema({
    clientId: String,
    clientName: String,
    clientEmail: String,
    title: String,
    description: String,
    budget: Number,
    skills: Array,
    bids: Array,
    bidAmounts: Array,
    postedDate: String,
    status: {
        type: String,
        default: "Available"
    },
    freelancerId: String,
    freelancerName: String,
    deadline: String,
    submission: {
        type: Boolean,
        default: false
    },
    submissionAccepted: {
        type: Boolean,
        default: false
    },
    projectLink: {
        type: String,
        default: ""
    },
    manulaLink: {
        type: String,
        default: ""
    },
    submissionDescription: {
        type: String,
        default: ""
    },
})
```

```
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);
```

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 user

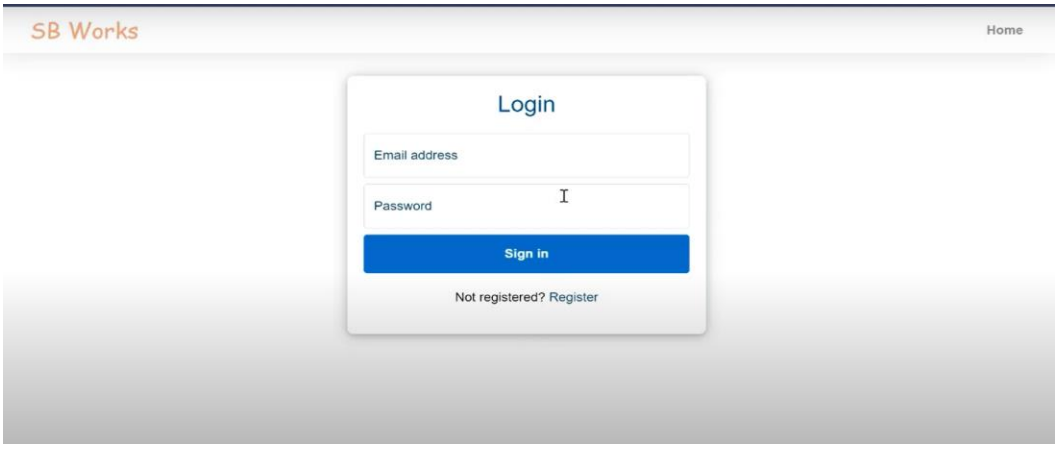
Project Implementation

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in

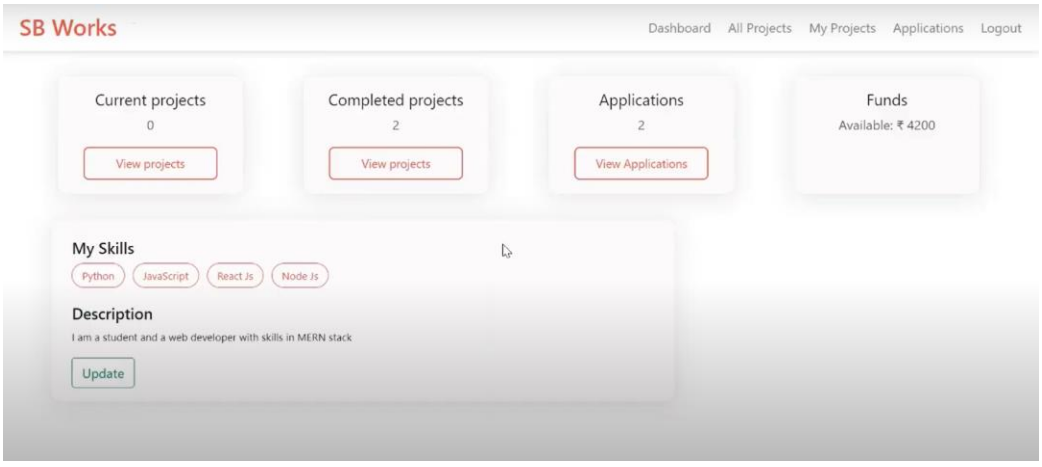
it. The user interface of the application looks a bit like the images provided below.

Landing page:

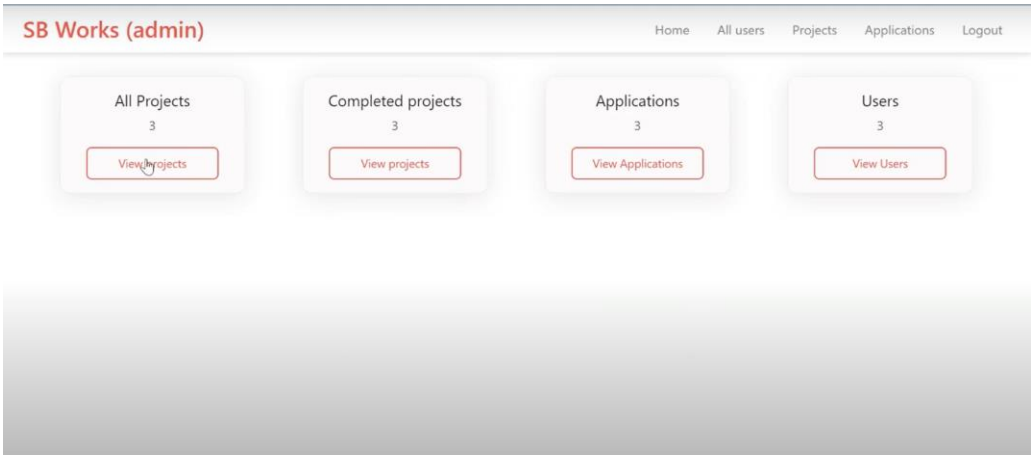
Authentication:



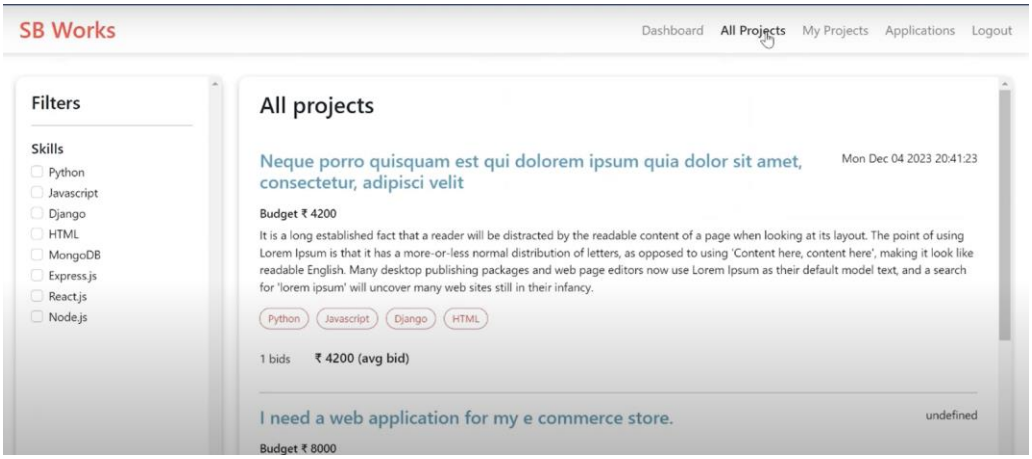
Freelancer dashboard:



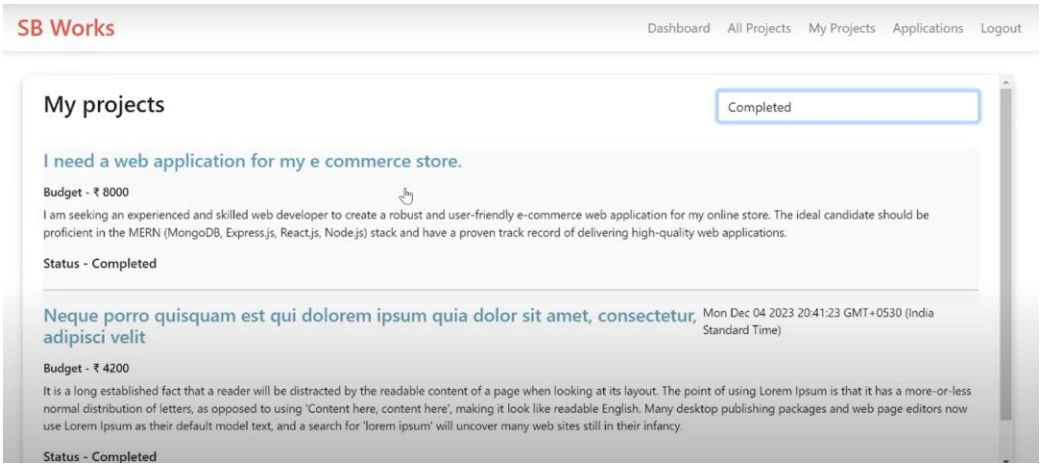
Admin dashboard:



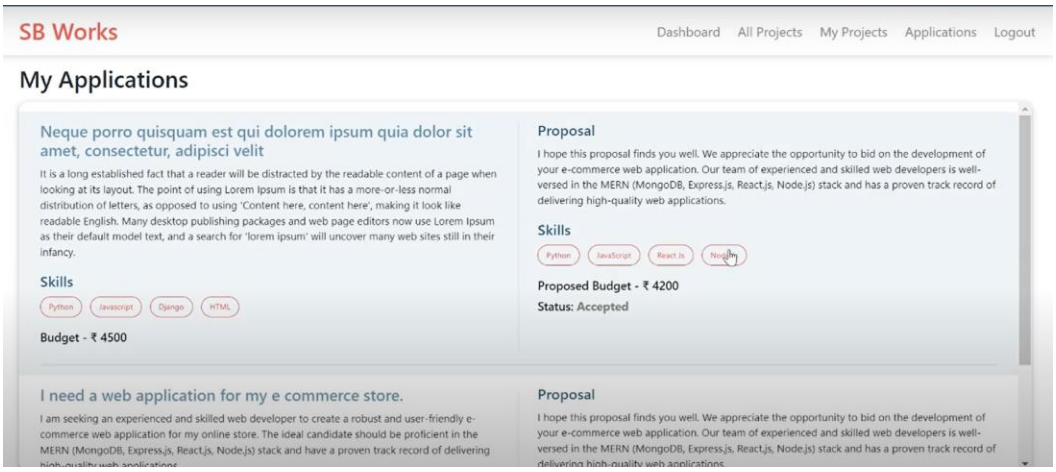
All projects:



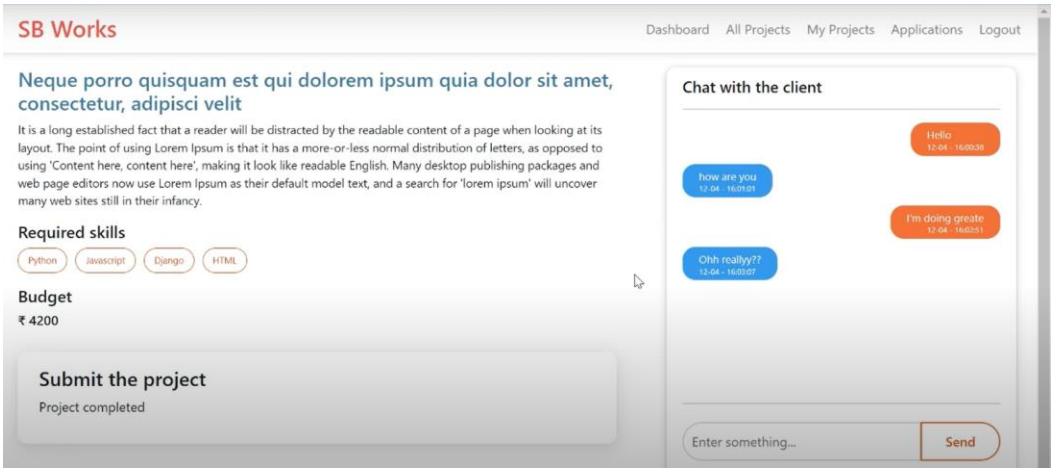
Freelance projects:



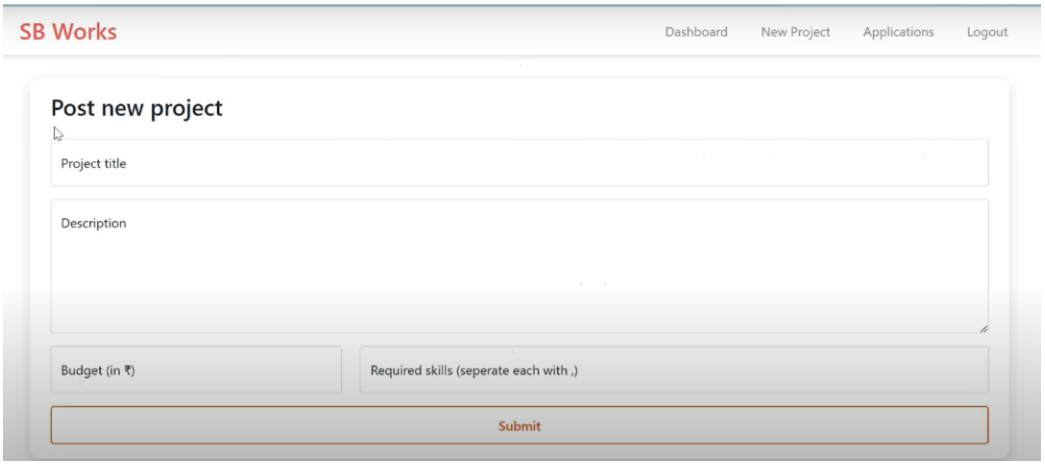
Applications:



Project page:



New project:



Conclusion

FreelanceFinder is more than just a web application—it's a bridge connecting talented freelancers with meaningful opportunities across diverse industries. By streamlining the job search and project hiring process, it empowers individuals to unlock their true potential, work on their own terms, and grow their professional network. With features designed for both clients and freelancers, the platform fosters trust, transparency, and collaboration. As the freelance economy continues to expand, FreelanceFinder stands as a powerful tool to navigate this dynamic landscape, making the process of finding and offering freelance work efficient, rewarding, and future-ready.