PROJECT REPORT

ON

FreelanceFinder: Discovering Opportunities, Unlocking Potential

BY

Kata Bharath(Team Leader)

Athuru Davidraj(Team Member)

Shaik Davoodh(Team Member)

Raja Dhanasekhar (Team Member)

ABSTRACT

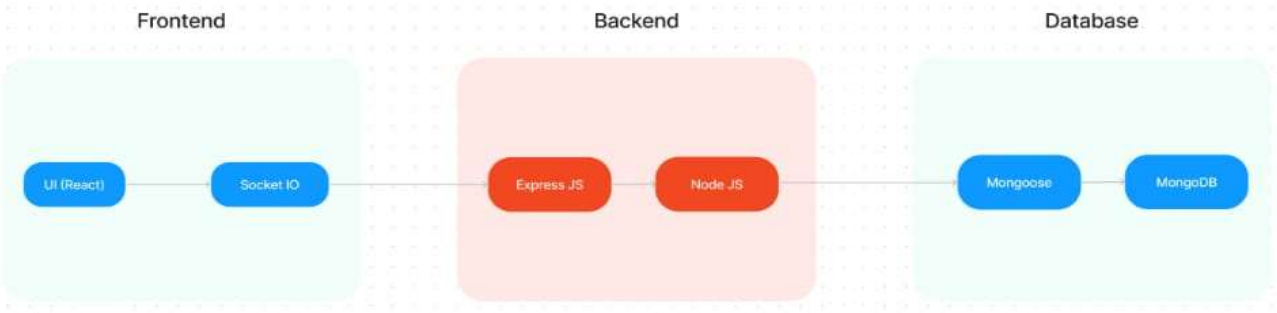
The Freelancer Project Management System is a web application that helps connect freelancers with clients. It is built using the MERN stack — MongoDB, Express.js, React.js, and Node.js. Freelancers can register, view available projects, apply for jobs, and chat with clients. Clients can post projects and manage applications.

The system also includes an admin panel to monitor users and projects. It uses secure login, real-time messaging, and a simple, user-friendly design. This project shows how a full-stack application can be used to manage freelance work more easily and effectively.

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.

TECHNICAL ARCHITECTURE

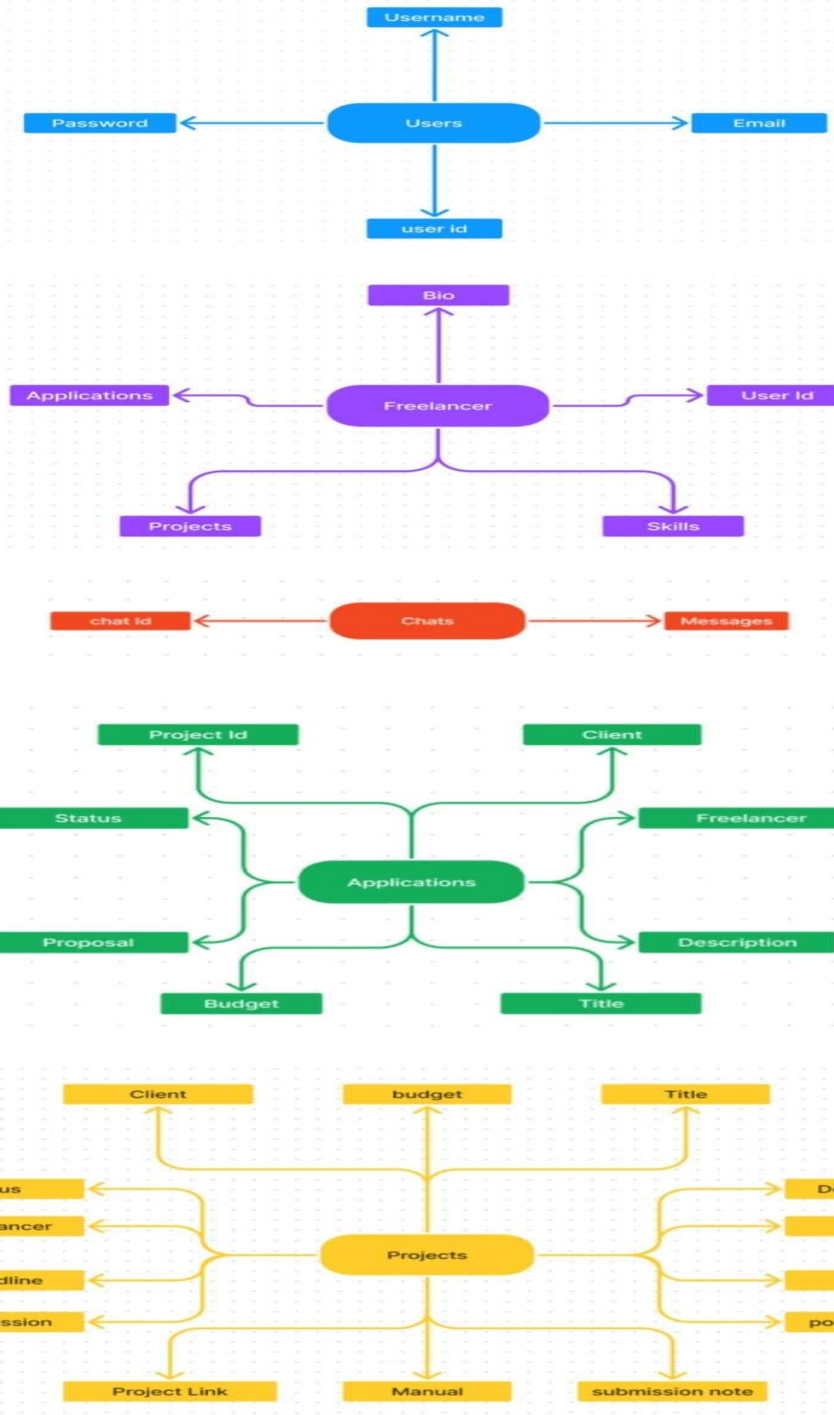


SB Works uses a client-server model. The frontend is the user interface built with Bootstrap and Material UI to make it look good and work smoothly. It uses Axios to send and receive data from the backend using REST APIs.

The backend is built with Express.js, which handles the logic, requests, and responses. For storing data like user locations and images, it uses MongoDB, which is fast and scalable.

Together, the frontend, backend, and database work as a complete system that allows users to easily share and explore local places in real time.

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 uses React.js to build the user interface. It has reusable blocks (called components) for things like profiles, projects, and chat. These blocks are arranged into pages such as "Browse Projects" or "Freelancer Profile." Shared data, like the logged-in user or search filters, is managed using React Context.

On the server side, Node.js handles things like user login, posting projects, and messaging. It uses Mongoose to talk to the MongoDB database, which stores all the data in an organized way.

PRE -REQUISTIC:

- Node.js and npm
- Express.js
- MongoDB
- React.js

✓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.

Application Flow

Freelancer Responsibilities:

Submit Projects: Upload completed, high-quality work.

Follow Guidelines: Make sure the work meets client needs and platform rules.

Communicate Well: Reply to messages, ask questions, and give updates.

Meet Deadlines: Finish work on time.

Be Professional: Stay respectful and cooperative.

Check Quality: Ensure the work is correct and error-free.

Client Responsibilities:

Clear Instructions: Write detailed project descriptions.

Respond Quickly: Answer freelancer questions and give feedback.

Pay on Time: Make payments once work is done properly.

Give Feedback: Help freelancers improve with useful reviews.

Admin Responsibilities:

Monitor Data: Keep all user data safe and accurate.

Enforce Rules: Make sure everyone follows platform rules.

Solve Problems: Handle disputes fairly and quickly.

Help Users: Support and guide users when needed.

Maintain Platform: Keep improving and fixing the system.

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.

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

Milestone 2: Backend Development

1. Project Setup

Create the project folder and run npm init.

Install required packages: express, mongoose, body-parser, and cors.

2. Database Setup

Use MongoDB (locally or through MongoDB Atlas).

Create collections:

Users – Stores user details and account type.

Projects – Stores project info, budget, and skills.

Applications – Holds proposals, freelancer rates, and portfolio links.

Chat – Saves messages for each project.

Freelancer – Extra details like skills, experience, and ratings.

3. Express Server

Set up an Express server to handle all API requests.

Use body-parser and cors for handling requests and security.

4. API Routes

Create separate files for different routes:

Users – Register, login, update profile.

Projects – Post projects, list all, view details.

Applications – Submit/view proposals.

Chat – Send and receive messages.

Freelancers – View/update profiles.

5. Data Models

Use Mongoose to create schemas and models:

User, Project, Application, Chat, and Freelancer (extends User).

Build full CRUD operations to manage data.

6. User Authentication

Use JWT or sessions for login security.

Add middleware to protect routes (e.g., applying to projects).

7. Project Features

Clients can post projects with details.

Freelancers can browse and apply to suitable projects.

Clients review applications and choose freelancers.

8. Chat & Collaboration

Add a secure chat system within each project.

Support sending messages, files, and feedback.

9. Admin Panel (Optional)

Admin can:

Manage users

Monitor projects and applications

View transaction history

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

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.

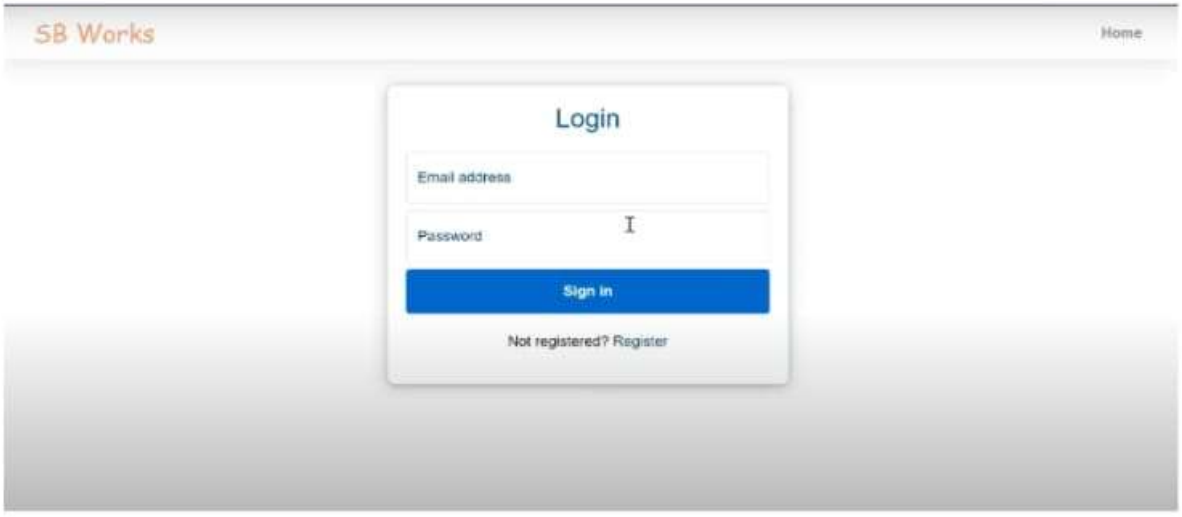
This completes the frontend development, bringing the SB Works platform to life for users.

Milestone 5: Project Implementation

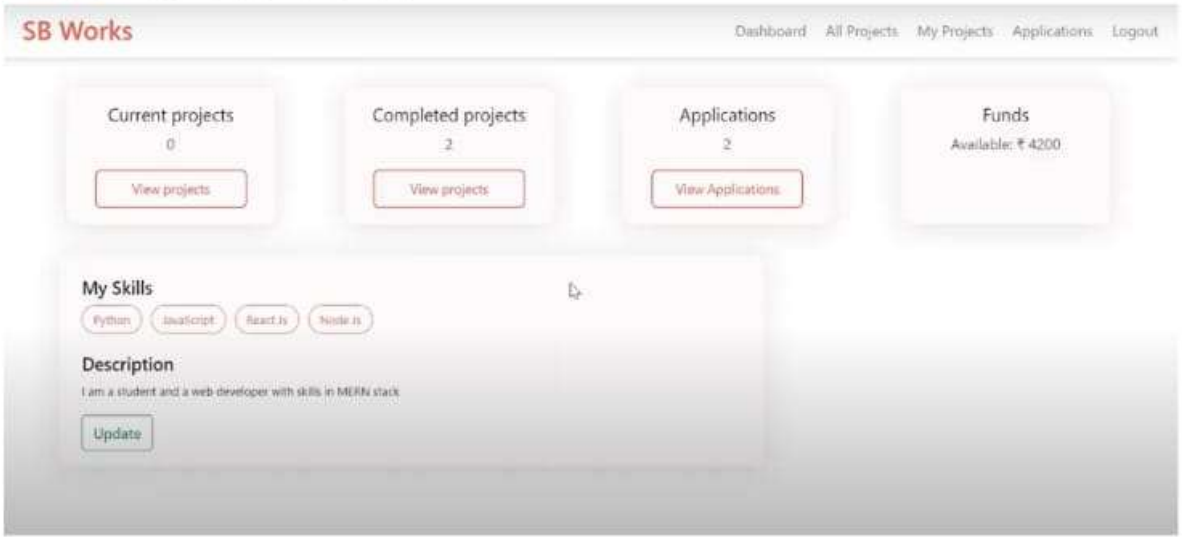
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the images provided below.

Landing page:

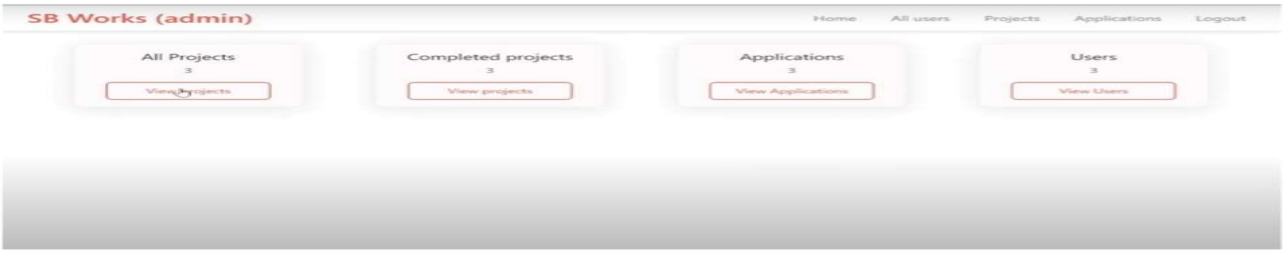
Authentication:



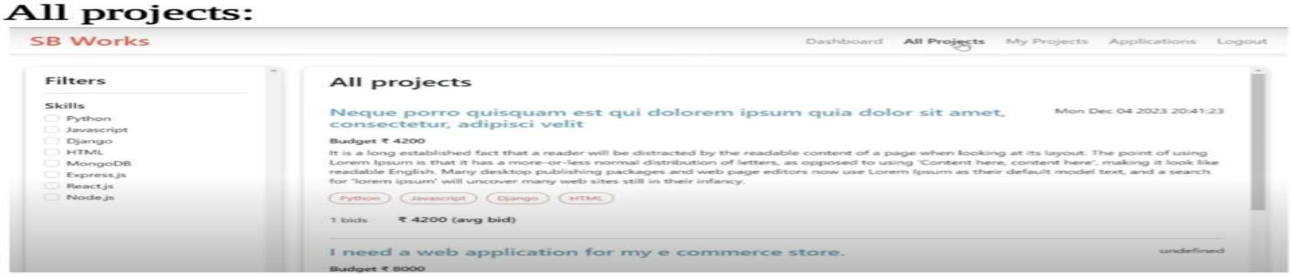
Freelancer dashboard:



Admin dashboard:

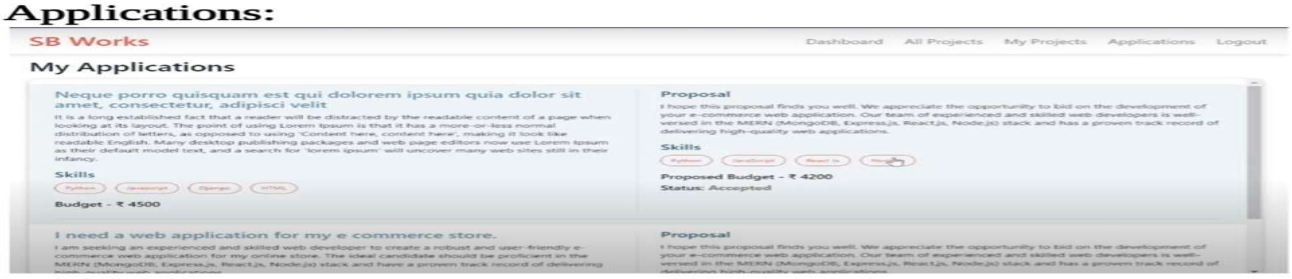


All projects:



Freelance projects:

Applications:



Project page:

New project:

CONCLUSION

The backend of the Freelancer Project System helps manage users, projects, applications, and chats in a smooth and organized way. It uses tools like Node.js, Express.js, and MongoDB to make everything work fast and safely.

With proper APIs, data models, and login security, the system allows freelancers, clients, and admins to use the platform easily. Important features like posting projects, applying, chatting, and admin controls are working well.

This step prepares the system to connect with the frontend and become a full working platform.