FREELANCING APPLICATION USING MERN STACK

A Project Done By

SELVAHARINI L	211121104047
AKSHAYA S	211121104005
MADHUMITHA	211121104034
HARITHA VARSHINI V	211121104020

STUDENTS OF

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

MADHA ENGINEERING COLLEGE

KUNDRATHUR

CHENNAI-69

FREELANCING APPLICATION USING MERN STACK

A Project Done By

SELVAHARINI L	**211121104047**
AKSHAYA S	**211121104005**
MADHUMITHA	**211121104034**
HARITHA VARSHINI V	**211121104020**

STUDENTS OF

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

MADHA ENGINEERING COLLEGE

KUNDRATHUR

CHENNAI-69

ACKNOWLEGDEMENT

First of all we pay our grateful thanks to the chairman **Ln.Dr. S.Peter** for introducing the Engineering College in Kundrathur

We would like to thank the Director **Er.A.Prakash,** for giving us support and valuable suggestion for our project

It is with great pleasure and privilege we express our sincere thanks and gratitude to **Dr.Ponnusamy R.P, M.Tech., Ph.D.,** Principal, for the spontaneous help rend to us during our study in this college

We express our sincere thanks to **Mrs. Er. B Kalpana Sattu., B.Tech(IT)., BS Psychology., MS Criminology., ME CSE,** Head of the Computer Science Department and our project Co-ordinator **Mr. M.Navin bharathi., M.Tech(IT).,** for the Good will fostered towards and for their guidance during the execution of this project.

It is a great privilege to express our sincere thanks to our Internal Guide **Mr. M.Navin bharathi., M.Tech(IT).,** and we acknowledge our indebtedness to her for the encouragement valuable suggestions and clear tireless guidance given to us on the preparation and execution of this project

It is a great privilege to express our sincere thanks to **SMART INTERNZ** and **NAANMUDHALVAN** team.

We would like to thank all the teaching and non-teaching **STAFF MEMBERS & Friends** of the Computer Science Engineering Department for giving the support and valuable suggesions for our Project work.

BONAFIDE CERTIFICATE

Certified that this project report titled " **FREELANCING APPLICATION**" is the Bonafide work of **"SELVAHARINI.L (211121104047) , "HARITHA VARSHINI.V" (211121104020), "AKSHAYA.S" (211121104005) , " MADHUMITHA" (211121104034)** who carried out the project. Work under my supervision.

SIGNATURE	SIGNATURE
Er. B KALPANA SATTU., B.Tech(IT)., BS Psychology., MS Criminology., ME CSE	**Mr. M.NAVIN BHARATHI., M.Tech(IT)**
HEAD OF THE DEPARTMENT Computer Science Engineering	ASSOCIATE PROFESSOR Computer Science Engineering
Madha Engineering College,	Madha Engineering College,
Kundrathur, Chennai - 600 069	Kundrathur, Chennai - 600 069

Submitted for the examination held on ______________

Internal Examiner External Examiner

ABSTRACT

The freelancing web application, developed using the MERN (MongoDB, Express.js, React, Node.js) stack, is designed to connect freelancers and clients on a dynamic and scalable platform. This project addresses the growing demand for online freelancing platforms by providing a user-friendly interface, real-time features, and secure payment processing.

The system facilitates seamless job posting, application management, and real-time communication between users. MongoDB ensures efficient data storage and retrieval, while Node.js and Express.js handle backend operations and API management. React powers the frontend with a responsive and interactive user interface, enhancing the overall user experience.

This project report outlines the entire development lifecycle, from requirements analysis and system design to implementation, testing, and deployment. It also explores the challenges encountered, such as performance optimization, data consistency, and security, along with the solutions implemented.

With a modular architecture and cloud-ready infrastructure, the freelancing web application is scalable and adaptable for future enhancements. This platform serves as a modern solution for bridging the gap between talent and opportunity, empowering users in the global freelancing ecosystem.

Table of Content

1. INTRODUCTION

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.

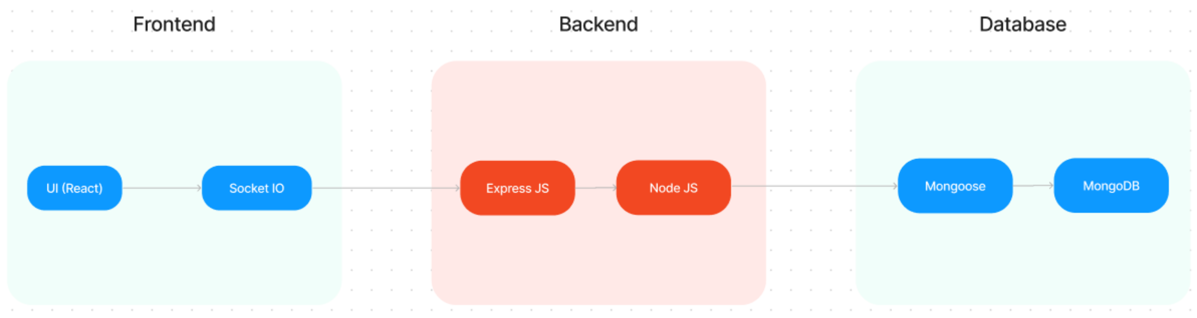
Bidding with Confidence: Confident in her design skills, Sarah dives into the project details. SB Works allows her to review the bakery's previous marketing materials, further solidifying her design approach. She submits a compelling proposal highlighting her relevant experience and attaching a few samples 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.

2. 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.

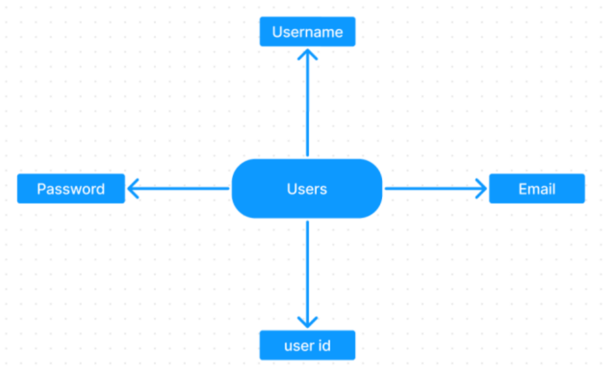
To enhance the user experience, the frontend leverages the Bootstrap and Material UI libraries, creating a real-time and visually appealing interface for users.

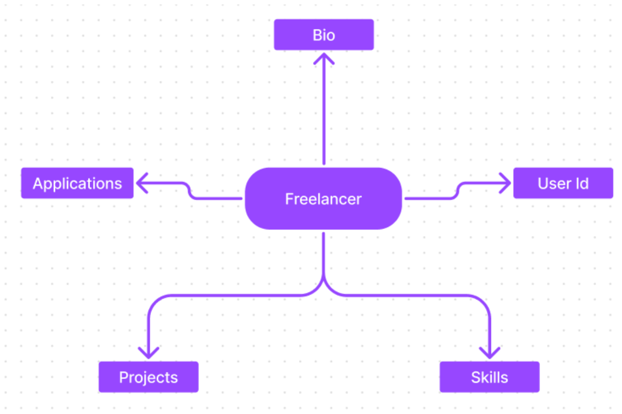
On the backend, we utilize the Express Js framework to manage server-side logic and communication. Express Js provides a robust foundation for handling requests and responses efficiently.

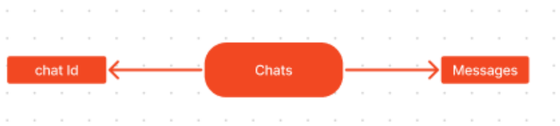
For data storage and retrieval, SB Works relies on MongoDB. MongoDB offers a scalable and efficient solution for storing various data, including user-contributed locations and images. This ensures quick and reliable access to the information needed to enrich the local tourism experience.

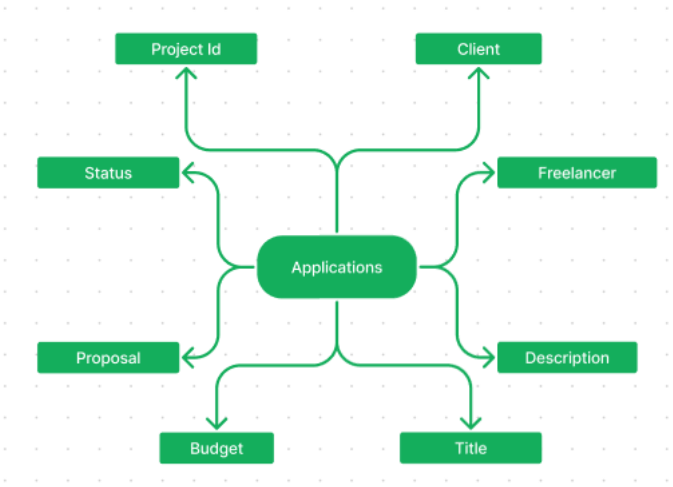
In conjunction, the frontend and backend components, complemented by Express Js, and MongoDB, together form a comprehensive technical architecture for SB Works. This architecture facilitates real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive experience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

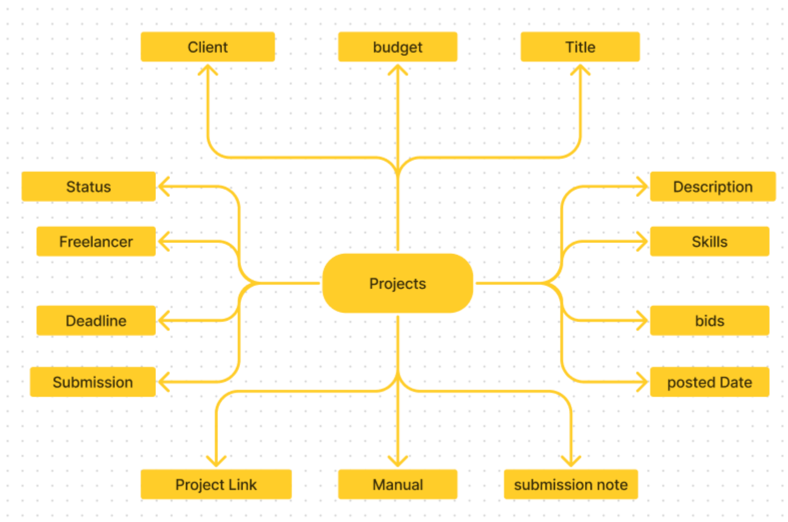
2.2 ER Diagram:











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.

2.3. Frontend Architecture (React.js):

The frontend is designed using React.js, a JavaScript library that allows for building reusable and responsive user interfaces. Key aspects include:

- **Component-Based Structure:**
  UI is divided into reusable components like ProductCard, Navbar, and Cart. Each component handles specific tasks, improving maintainability.

- **State Management:**
  Context API or Redux is used for managing application state, such as user authentication and cart data.

- **Routing:**
  React Router manages navigation between pages, such as Home, Product Details, and Checkout, without full-page reloads.

- **Styling:**
  Styling is implemented using CSS, SCSS, or libraries like Material-UI or Bootstrap for responsiveness.

2.4. Backend Architecture (Node.js + Express.js):

The backend handles business logic, API endpoints, and communication with the database. Key aspects include:

- **RESTful APIs:**
  APIs are built using Express.js to handle HTTP requests and responses.
  - Example:
    - GET /api/products: Fetch all products.
    - POST /api/orders: Place an order.
- **Middleware:**
  Custom middleware is implemented for tasks like request validation, error handling, and authentication using JWT.
- **Scalability:**
  Node.js's asynchronous and event-driven nature ensures the backend can handle multiple concurrent requests.

2.5. Database Architecture (MongoDB):

MongoDB, a NoSQL database, is used for flexible and scalable data storage.

- **Collections:**

  The database consists of the following key collections:

  - Users: Stores user credentials, profile data, and order history.
  - Products: Stores product details like name, price, category, and availability.

- Orders: Tracks orders, including user ID, product IDs, and order status.
- **Cart:** Temporary storage for a user's cart items.
- **Schema Design:**
  Documents are structured to allow efficient querying and updates. For instance, orders store product references instead of embedding entire product details to minimize data duplication.

3. SETUP INSTRUCTION

Prerequisites:

- Node.js (v14 or higher)
- MongoDB (locally or via a cloud service like MongoDB Atlas)
- Git
- npm or yarn

Installation:

1. **Clone the repository:**

```
git clone <repository-url>
cd <repository-folder>
```

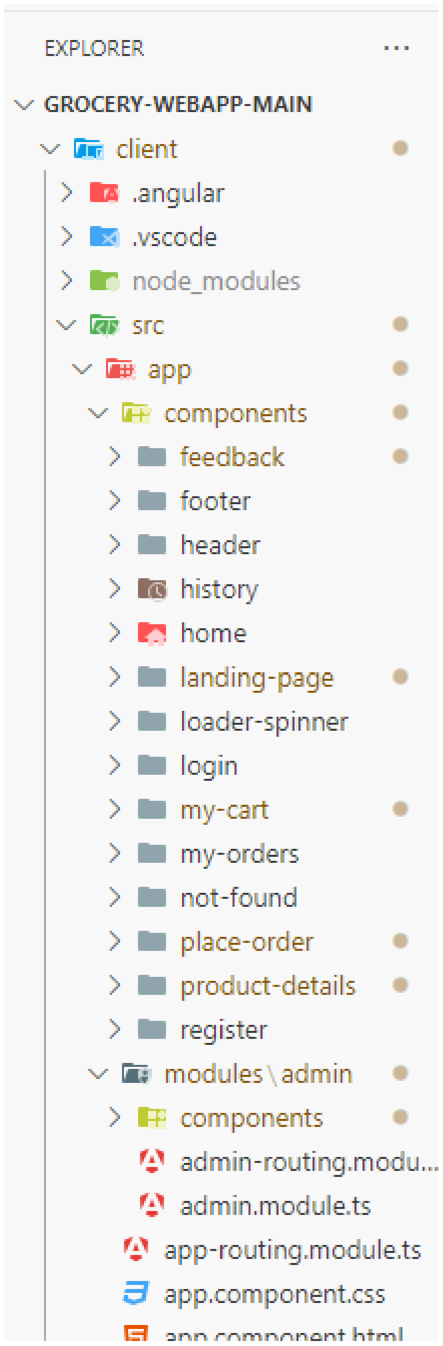
2. **Install dependencies for both frontend and backend:**

```
cd client
npm install
cd ../server
npm install
```

3. **Set up environment variables:**

- Create a .env file in the server directory.
- Add variables such as database connection string, JWT secret, etc.

Project Structure



```
EXPLORER                                ···
∨ GROCERY-WEBAPP-MAIN
  ∨ client                               ●
    > .angular
    > .vscode
    > node_modules
    ∨ src                                ●
      ∨ app                              ●
        ∨ components                     ●
          > feedback                     ●
          > footer
          > header
          > history
          > home
          > landing-page                 ●
          > loader-spinner
          > login
          > my-cart                      ●
          > my-orders
          > not-found
          > place-order                  ●
          > product-details              ●
          > register
        ∨ modules \ admin                ●
          > components                   ●
            admin-routing.modu...
            admin.module.ts
          app-routing.module.ts
          app.component.css
          app.component.html
```

Fig 1.3 Project Structure

This structure assumes an Angular app and follows a modular approach.

Here's a brief explanation of the main directories and files:

- src/app/components: Contains components related to the customer app, such as register, login, home, products, my-cart, my-orders, placeorder, history, feedback, product-details, and more.
- src/app/modules: Contains modules for different sections of the app. In this case, the admin module is included with its own set of components like add-category, add-product, dashboard, feedback, home, orders, payment, update-product, users, and more.
- src/app/app-routing.module.ts: Defines the routing configuration for the app, specifying which components should be loaded for each route.
- src/app/app.component.ts, src/app/app.component.html, `src.

<u>**Running the Application:**</u>

1. Start the Backend

The backend server is powered by Node.js and runs on the port specified in the .env file (default: 5000).

1. Open a terminal.
2. Navigate to the server directory:

```
cd se
rver
```

3. Start the backend server:

```
npm start
```

2. Start the Frontend

The frontend is built using React.js and runs on port 3000 by default.

1. Open a new terminal.
2. Navigate to the client directory:

```
cd client
```

3. Start the frontend development server:

```
npm start
```

4. The React application will open in your default browser. If it doesn’t, you can manually open: **http://localhost:3000**

4. 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

5. SYSTEM ANALYSIS

1. Introduction to System Analysis

System analysis is the process of studying and understanding the current environment to design a new, improved system. For the freelancing web application, the analysis involves evaluating the existing systems, identifying user needs, and defining functional and non-functional requirements to ensure the application fulfills its intended purpose.

2. Existing System Analysis

The existing freelancing platforms face several challenges, including:

- **High Commission Rates**: Platforms often charge freelancers and clients high fees, reducing their profitability.
- **Limited Customization**: Users have limited control over their profiles, job postings, or search filters.
- **Lack of Real-Time Features**: Communication and updates are often delayed, affecting collaboration.
- **Scalability Issues**: Many platforms struggle to handle increasing numbers of users and projects effectively.

3. Proposed System

The proposed system is a modern freelancing web application built using the MERN stack, offering the following solutions:

- **Cost Efficiency**: Lower transaction fees for users compared to existing platforms.
- **Enhanced Customization**: User-friendly dashboards and advanced search filters.
- **Real-Time Interaction**: Features like instant messaging and real-time notifications.
- **Scalable Infrastructure**: A robust architecture designed to accommodate future growth.

4. Feasibility Study

- **TechnicalFeasibility**:

  The MERN stack (MongoDB, Express.js, React, Node.js) is a proven technology for building dynamic and scalable web applications. Each component of the stack supports the proposed system’s requirements:

- **MongoDB**: Efficient NoSQL database for managing user, job, and transaction data.
- **Express.js**: Lightweight backend framework for building APIs and handling server-side logic.
- **React**: A powerful library for creating responsive and dynamic user interfaces.
- **Node.js**: A fast and scalable runtime for handling multiple concurrent requests.

- **Economic Feasibility**:
  - Open-source tools in the MERN stack reduce development costs.
  - Cloud-based hosting ensures cost-effective scalability.
  - Anticipated revenue from user subscriptions and premium services can offset operational costs.
- **Operational Feasibility**:
  - The application will be user-friendly, addressing the needs of freelancers and clients.
  - Continuous user feedback mechanisms ensure the system adapts to user demands.

5. Functional Requirements

Functional requirements define the features and behavior of the system:

- **User Management**:
  - User registration and login (clients and freelancers).
  - Profile creation and updates.
- **Job Management**:
  - Job posting by clients.
  - Freelancers browsing and applying for jobs.
- **Communication**:
  - Real-time messaging between users.
- **Payments**:
  - Secure payment processing for job transactions.
- **Search and Filters**:
  - Advanced search based on skills, budget, ratings, and location.

6. Non-Functional Requirements

Non-functional requirements define the system's quality attributes:

- **Performance**:
  - Fast response times even with concurrent user actions.
  - Real-time notifications with minimal delay.
- **Scalability**:
  - The system should handle increasing users, jobs, and transactions.
- **Security**:
  - Secure user authentication and data encryption.
  - Protection against common vulnerabilities like SQL injection and XSS attacks.
- **Usability**:
  - Intuitive and responsive user interface across devices.
- **Reliability**:
  - High uptime with robust error-handling mechanisms.

7. Use Case Diagram

The use case diagram depicts the primary actors and their interactions with the system:

Actors:

- **Freelancer**: Registers, creates a profile, searches for jobs, applies for jobs, communicates with clients, and receives payments.
- **Client**: Registers, posts jobs, manages applications, communicates with freelancers, and processes payments.
- **Admin**: Manages users, jobs, and monitors platform activity.

Key Use Cases:

1. User Registration and Login.
2. Job Posting and Management.
3. Freelancers Applying for Jobs.
4. Real-Time Messaging.
5. Payment Processing and History.

8. SWOT Analysis

Strengths:

- Open-source MERN stack ensures cost efficiency.
- Real-time features enhance user collaboration.

Weaknesses:

- Initial development may take longer due to real-time functionality.

Opportunities:

- Potential to attract a global user base.
- Integration of advanced AI-based features like job recommendations.

Threats:

- Competition from established freelancing platforms.
- Security threats like data breaches or unauthorized access.

6. 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:

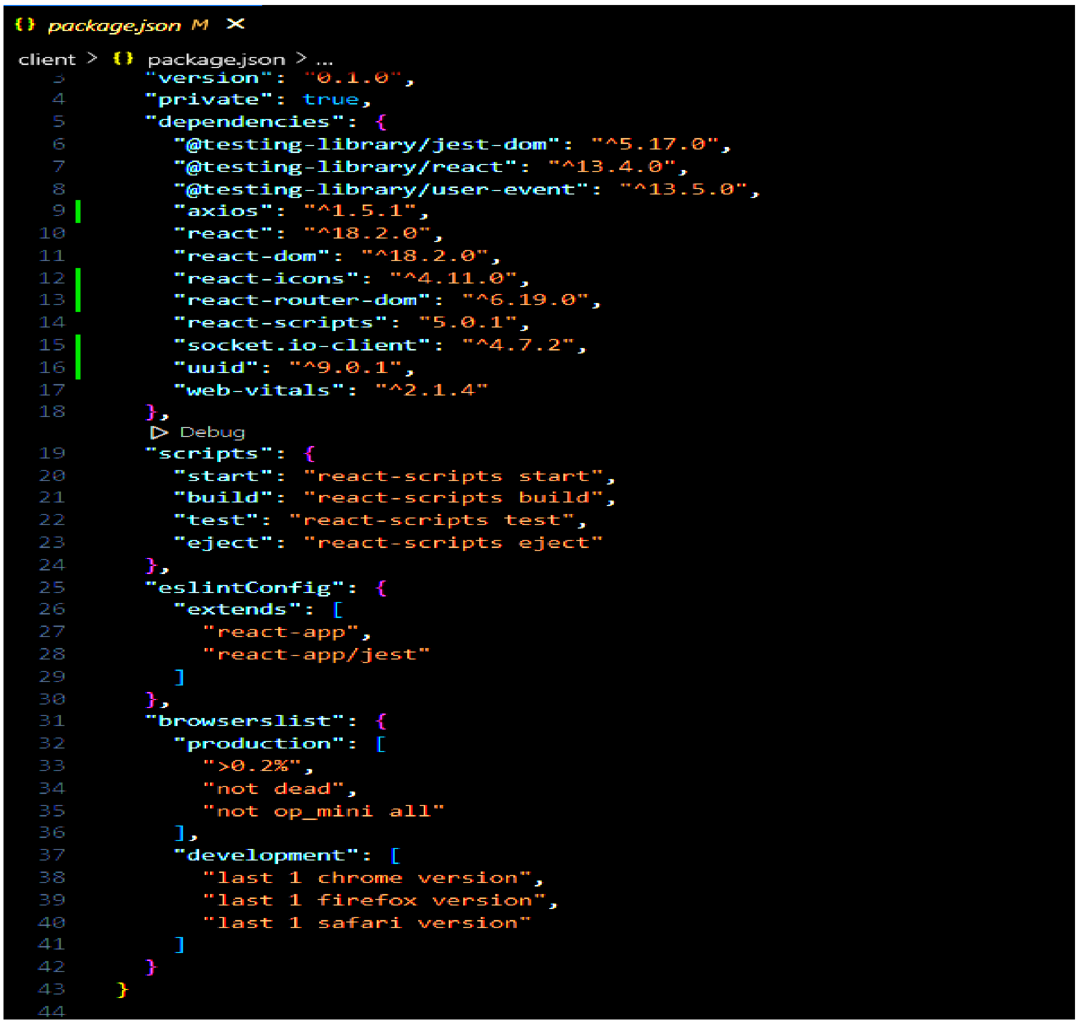
- React
- Bootstrap
- Material UI
- Axios
- react-bootstrap

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.

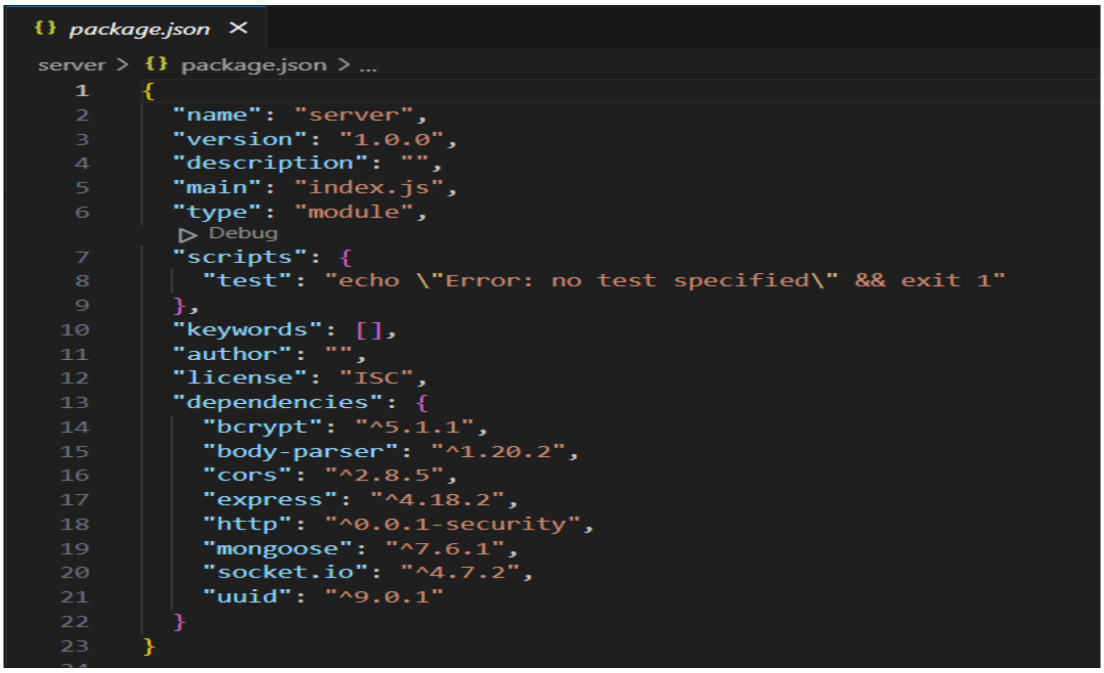


{} package.json M ✕

client > {} package.json > ...

```
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "axios": "^1.5.1",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-icons": "^4.11.0",
    "react-router-dom": "^6.19.0",
    "react-scripts": "5.0.1",
    "socket.io-client": "^4.7.2",
    "uuid": "^9.0.1",
    "web-vitals": "^2.1.4"
  },
  ▷ Debug
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

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



{} package.json ×

server > {} package.json > ...

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "type": "module",
  ▷ Debug
  "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"
  }
}
```

7. 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:

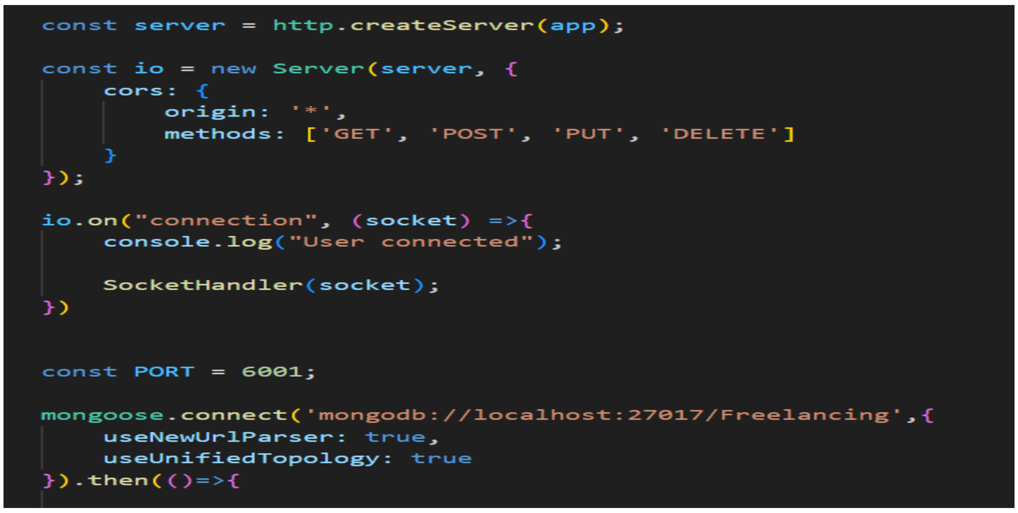
- Integrate a secure chat system within projects for communication between clients and freelancers.
- Allow file attachments and feedback exchange to facilitate collaboration.

9. Admin Panel (Optional):

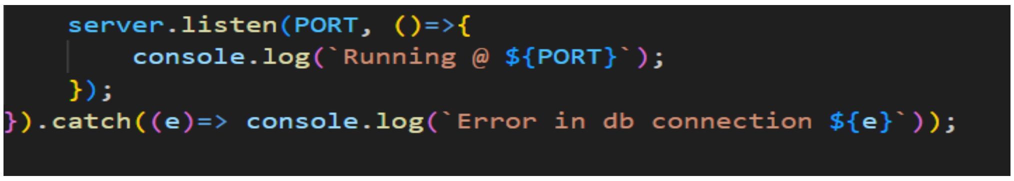
- Implement an admin panel with functionalities like:
- Managing users
- Monitoring project updates and applications
- Accessing transaction history

8. 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.
-
-

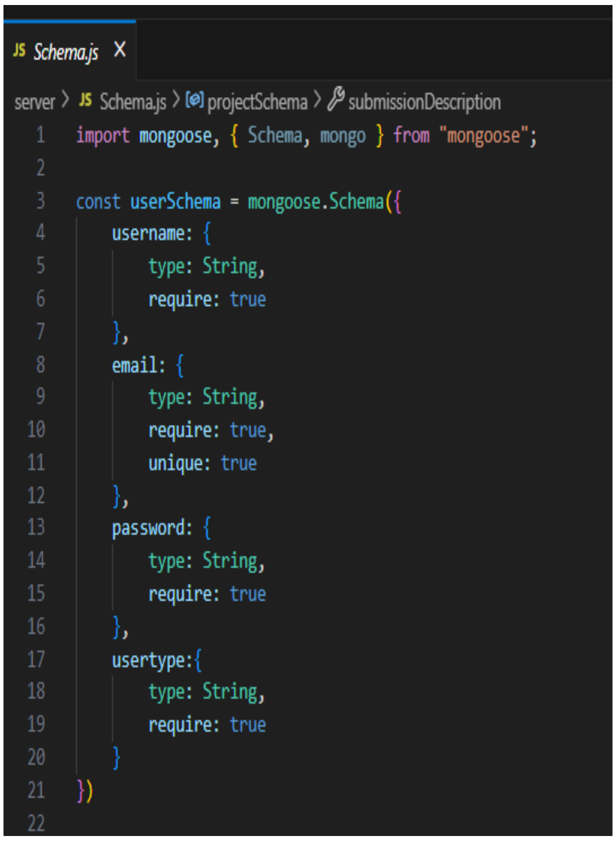


```
const server = http.createServer(app);

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(()=>{
```

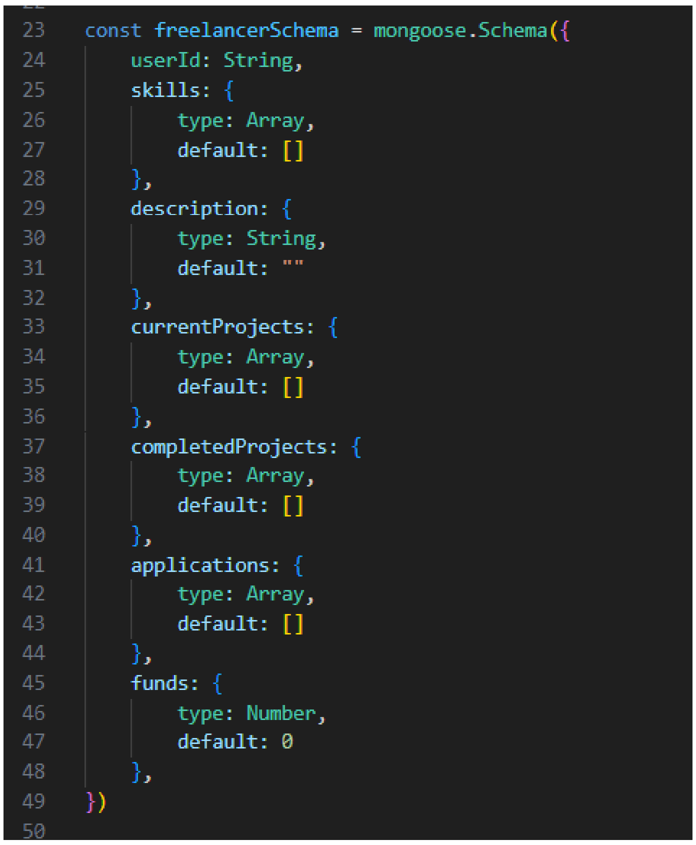


```
    server.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });
}).catch((e)=> console.log(`Error in db connection ${e}`));
```

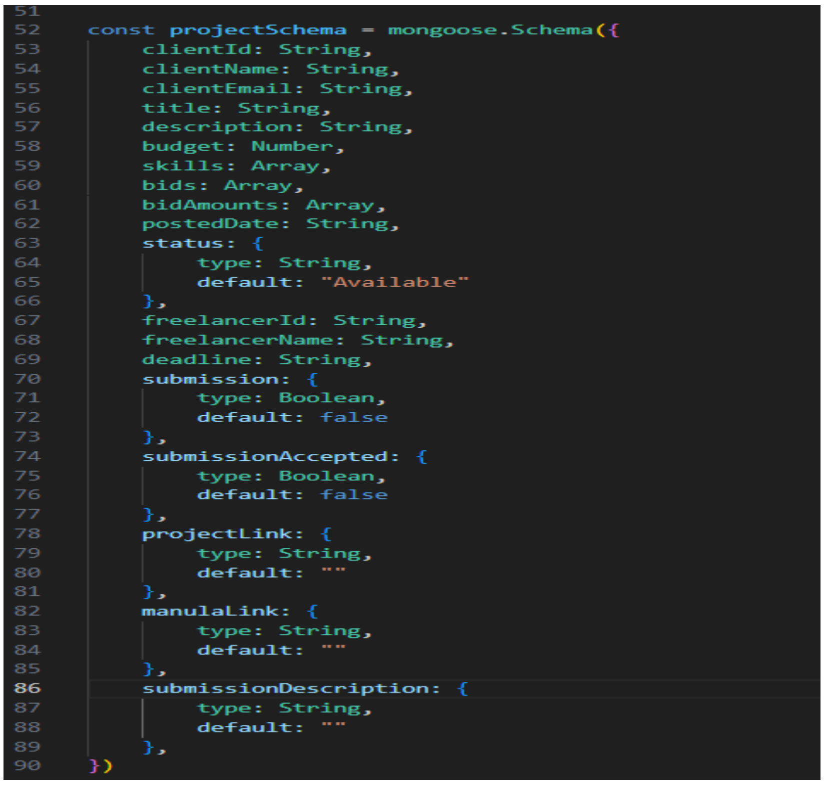
The Schemas for the database are given below



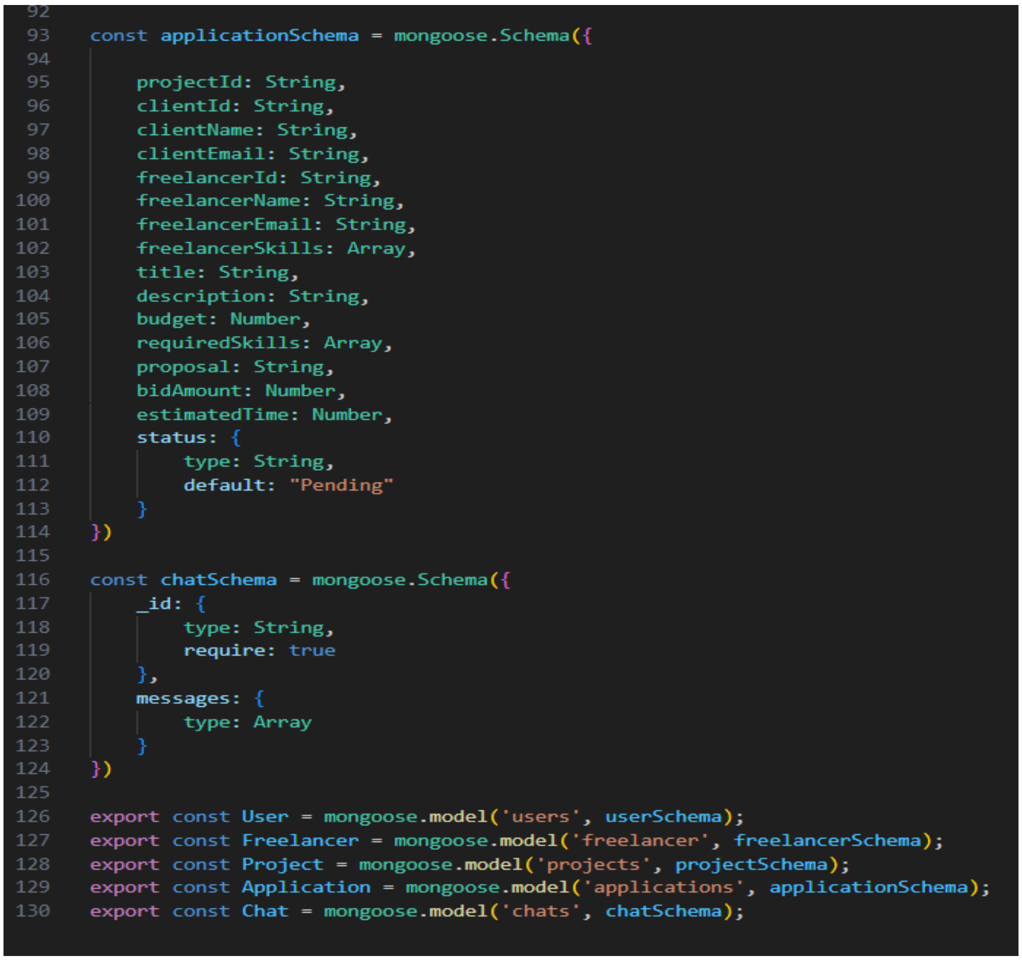
```javascript
import mongoose, { Schema, mongo } from "mongoose";

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
    }
})

```



```javascript
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: ""
    },
})
```



```
const applicationSchema = mongoose.Schema({

    projectId: String,
    clientId: String,
    clientName: String,
    clientEmail: String,
    freelancerId: String,
    freelancerName: String,
    freelancerEmail: String,
    freelancerSkills: Array,
    title: String,
    description: String,
    budget: Number,
    requiredSkills: Array,
    proposal: String,
    bidAmount: Number,
    estimatedTime: Number,
    status: {
        type: String,
        default: "Pending"
    }
})

const chatSchema = mongoose.Schema({
    _id: {
        type: String,
        require: true
    },
    messages: {
        type: Array
    }
})

export const User = mongoose.model('users', userSchema);
export const Freelancer = mongoose.model('freelancer', freelancerSchema);
export const Project = mongoose.model('projects', projectSchema);
export const Application = mongoose.model('applications', applicationSchema);
export const Chat = mongoose.model('chats', chatSchema);
```

8.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.

9. TESTING OVERVIEW

Introduction to Testing

Testing is an essential phase of the software development life cycle, ensuring that the application functions as expected and meets the specified requirements. It helps identify and resolve any defects or issues in the application before deployment. For this freelancing web application developed using the MERN stack (MongoDB, Express.js, React, Node.js), testing was carried out at various stages, covering different aspects like individual components, integration, user interactions, and overall system performance.

Types of Testing Implemented

1. **Unit Testing**

   Unit testing is performed on individual components or functions within the system to verify that each part works correctly. In the context of this project:

   - **Frontend (React)**: React components, such as user registration forms, login forms, and job listing displays, were tested to ensure they render and function as intended.
   - **Backend (Node.js/Express)**: API endpoints and business logic were tested to ensure that they return the correct responses under different conditions.
   - **Database (MongoDB)**: Individual database queries were tested to ensure that data was being correctly retrieved, inserted, and updated in MongoDB.

2. **Integration Testing**

   Integration testing ensures that different modules of the application work together as expected. For the freelancing platform:

   - Testing was done to verify the integration between the React frontend, Node.js/Express backend, and MongoDB database.
   - For example, when a user registers, the data is passed from the frontend form to the backend API, which stores it in the database. The integration tests ensure that this data flow works seamlessly without issues.

3. **System Testing**

   System testing involves testing the entire application in an environment that simulates real-world usage. It checks if the full system works as expected, ensuring that all components work together in a functional application.

    - Functional testing covered key workflows such as user registration, job posting, freelancer applications, messaging between clients and freelancers, and payment processing.
    - The system's performance under different user loads was also tested, especially in handling simultaneous user actions like job postings and applications.

4. **User Acceptance Testing (UAT)**

   UAT is a crucial phase where end-users (freelancers and clients) test the application to ensure it meets their expectations and is usable in real-world scenarios. In this project:
    - A sample group of users (freelancers and clients) were asked to perform common tasks, such as creating a profile, posting a job, applying for a job, and making payments.
    - Their feedback was collected to refine the user interface and fix usability issues.

Testing Process

The testing process followed a structured approach:

1. **Test Planning**: Initially, a test plan was created, outlining the testing goals, scope, methodologies, tools, and timelines.
2. **Test Case Creation**: Test cases were designed for each functionality of the application. Each test case included the test objective, input, expected result, and actual result.
3. **Execution**: The test cases were executed using manual and automated testing tools.
4. **Bug Reporting and Resolution**: Any defects or issues discovered during testing were logged, and the development team worked to resolve them. Once issues were fixed, the affected tests were rerun to ensure the problems were addressed.
5. **Retesting**: After resolving defects, retesting was performed to verify that fixes did not introduce new issues.

Tools Used for Testing

- **Frontend Testing**:
    - **Jest** and **React Testing Library** were used to test React components. These tools help test the components in isolation, simulating user interactions and validating that the component behavior is as expected.
- **Backend Testing**:
    - **Mocha** and **Chai** were used for testing Node.js and Express API endpoints. These tools allow for writing automated tests to verify the API responses and the expected output.
- **Database Testing**:

    - **MongoDB Compass** was used for manually querying the database to ensure data was correctly stored and retrieved.
    - **Mongoose** (the ODM for MongoDB) was tested to ensure correct data validation and schema behavior.
- **End-to-End Testing**:
    - **Postman** was used to test and document the API endpoints, ensuring that the backend and database were integrated properly.
    - **Selenium** could be used for end-to-end testing of the entire user workflow, simulating real-world interactions such as logging in, posting jobs, and applying for jobs.

Challenges Faced During Testing

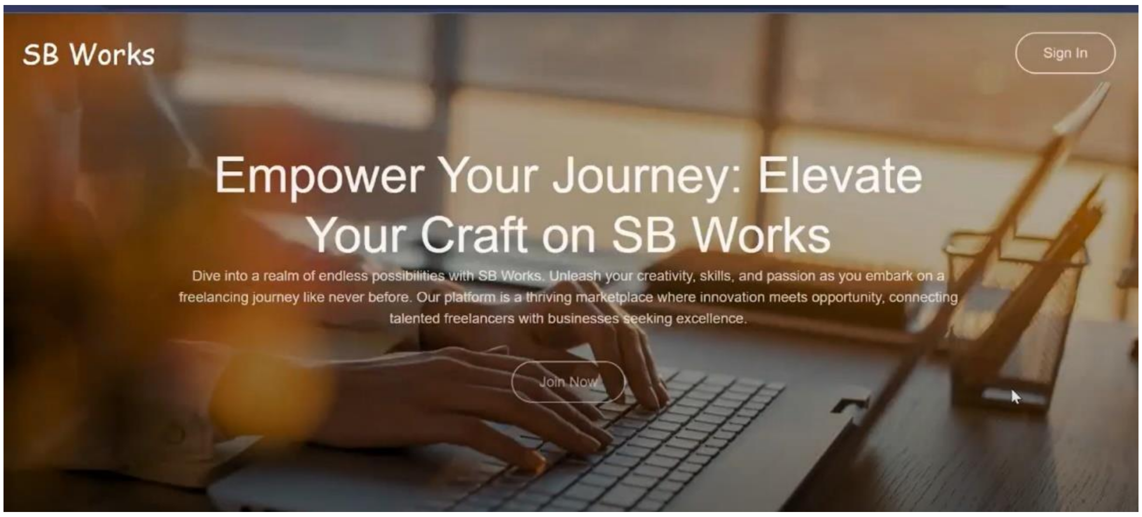
Testing the freelancing platform revealed several challenges:

- **Data Integrity**: Ensuring data consistency between the frontend, backend, and database was challenging. For instance, there were cases where user data submitted via forms was not correctly saved in the database due to errors in API endpoint handling.
- **Performance Issues**: During load testing, the application experienced slower response times when multiple users were interacting with the platform simultaneously. Optimizing database queries and server performance was required to improve speed.
- **Cross-Browser Compatibility**: Ensuring that the application worked seamlessly across different browsers and devices required significant effort in responsive design and testing.

10. USER INTERFACE & SCREENSHOTS

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.

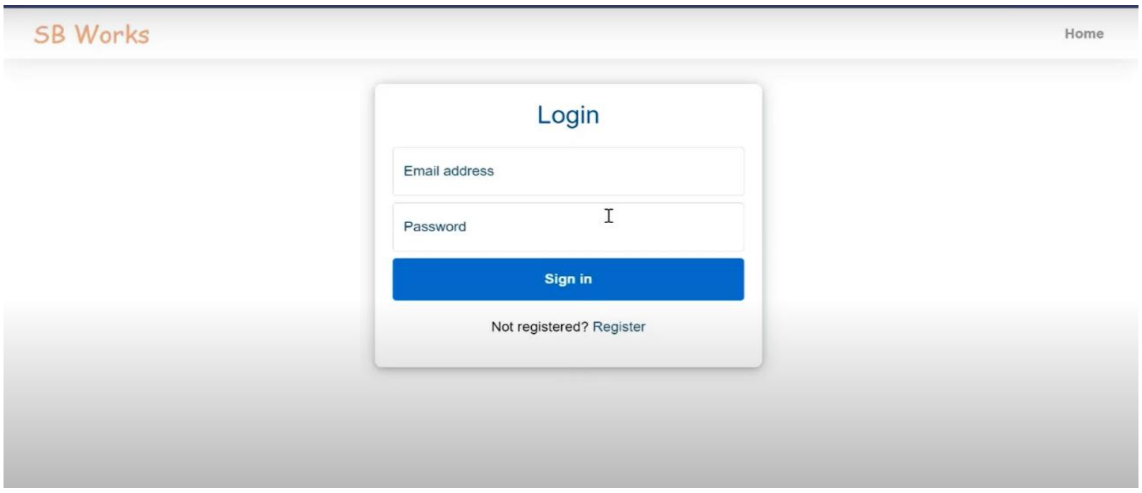
Landing page:



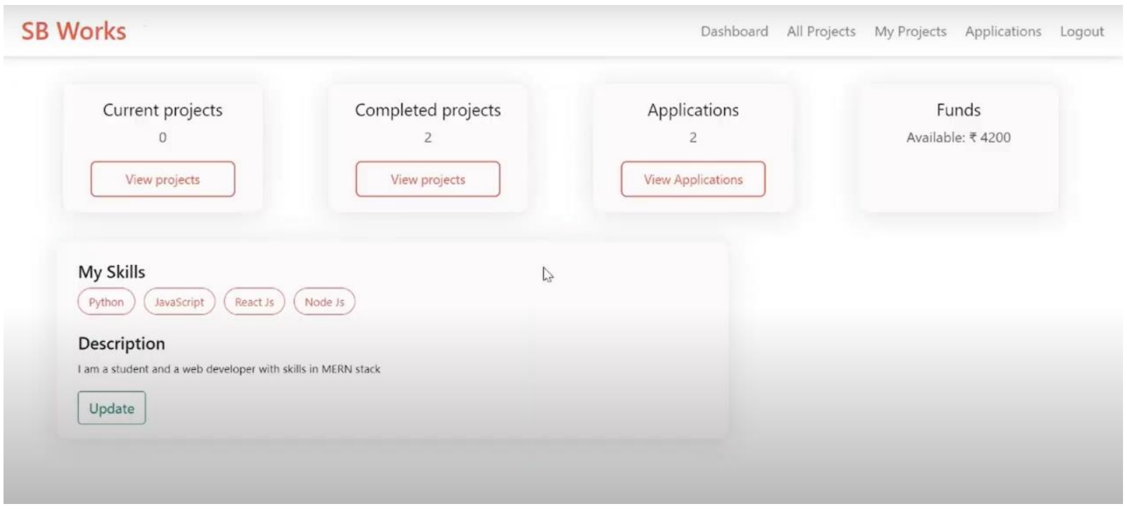
Empower Your Journey: Elevate Your Craft on SB Works

Dive into a realm of endless possibilities with SB Works. Unleash your creativity, skills, and passion as you embark on a freelancing journey like never before. Our platform is a thriving marketplace where innovation meets opportunity, connecting talented freelancers with businesses seeking excellence.

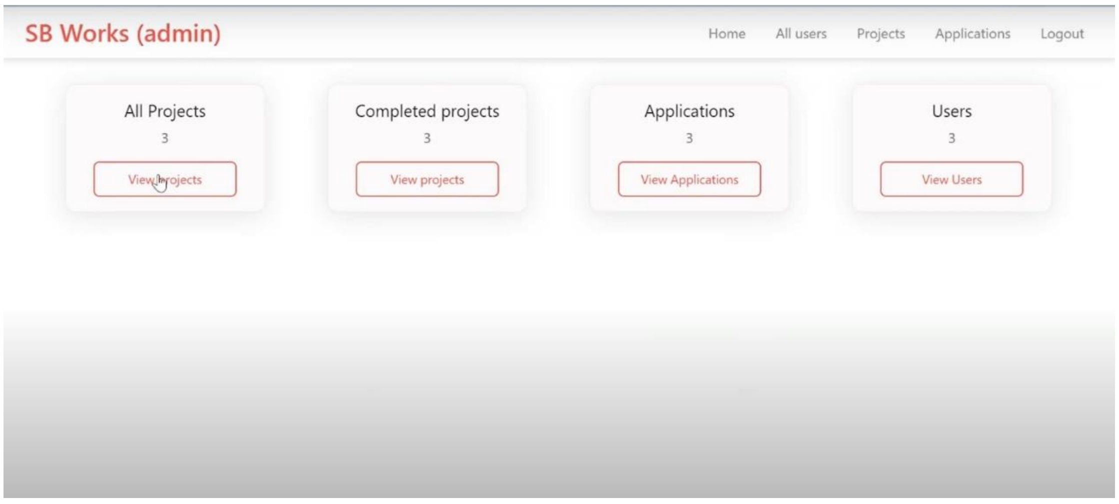
Authentication:



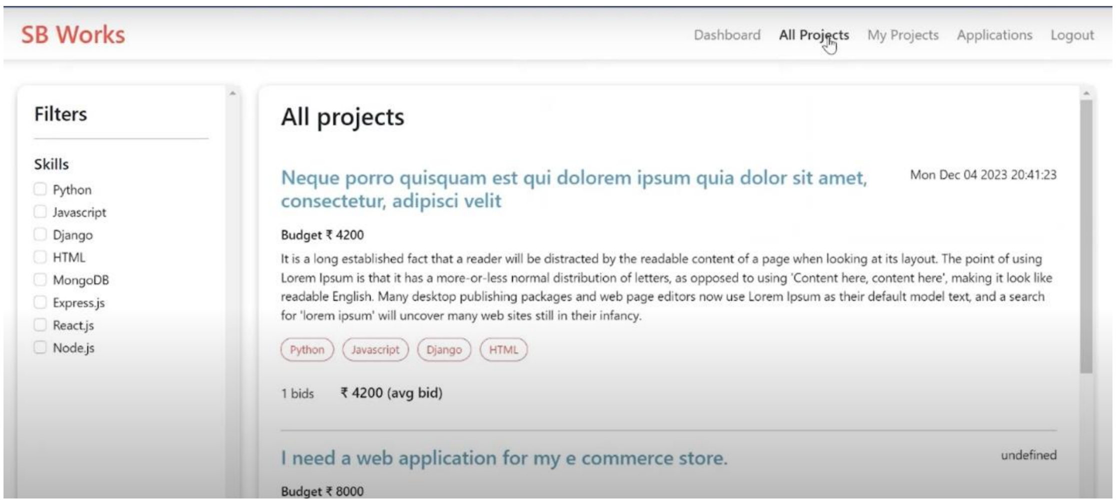
Freelancer dashboard:



Admin dashboard:



All projects:



All projects

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

Mon Dec 04 2023 20:41:23

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.

Python Javascript Django HTML

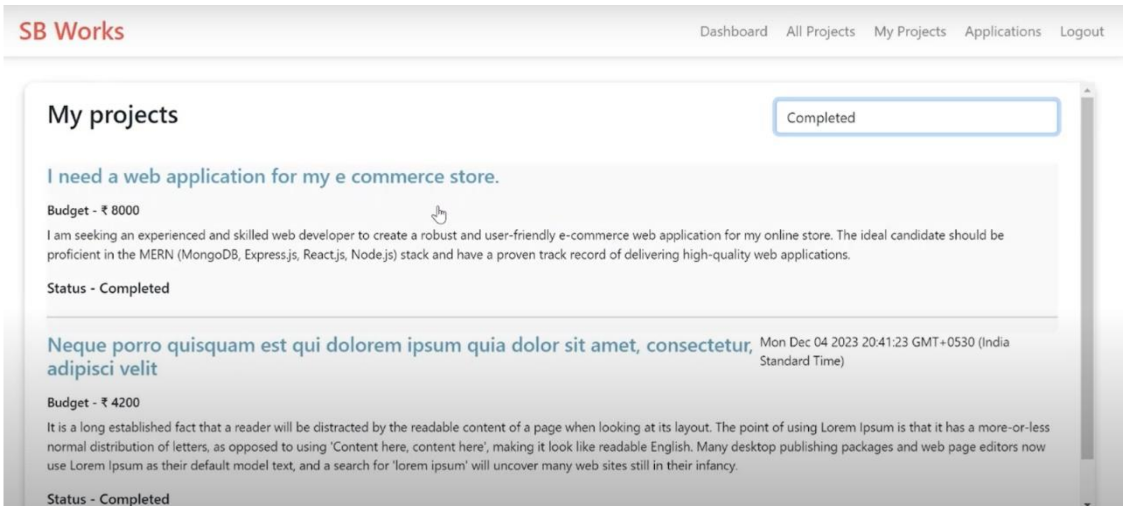
1 bids ₹ 4200 (avg bid)

I need a web application for my e commerce store.

undefined

Budget ₹ 8000

Freelance projects:



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

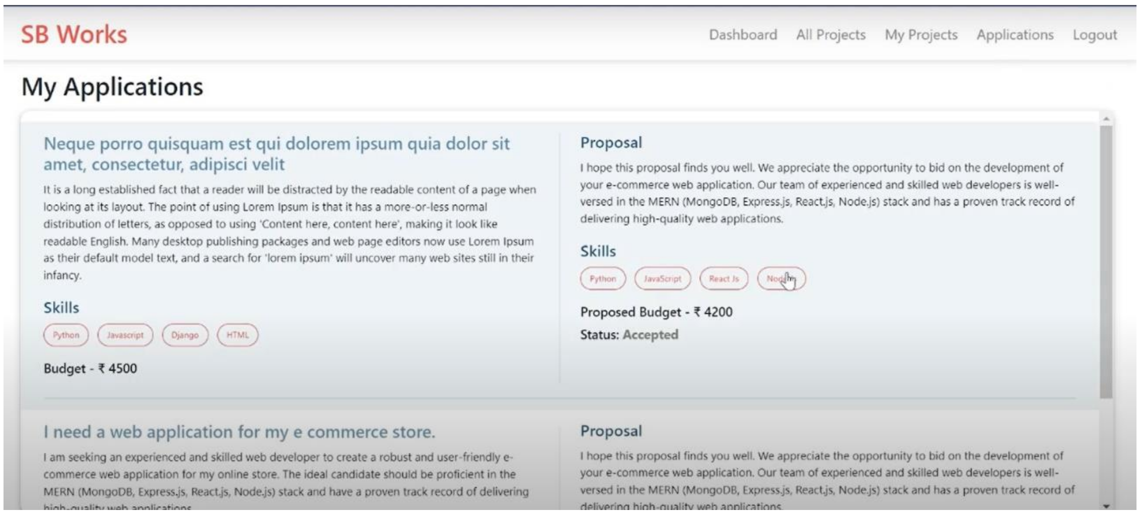
Mon Dec 04 2023 20:41:23 GMT+0530 (India Standard Time)

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:



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

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

Proposed Budget - ₹ 4200

Status: Accepted

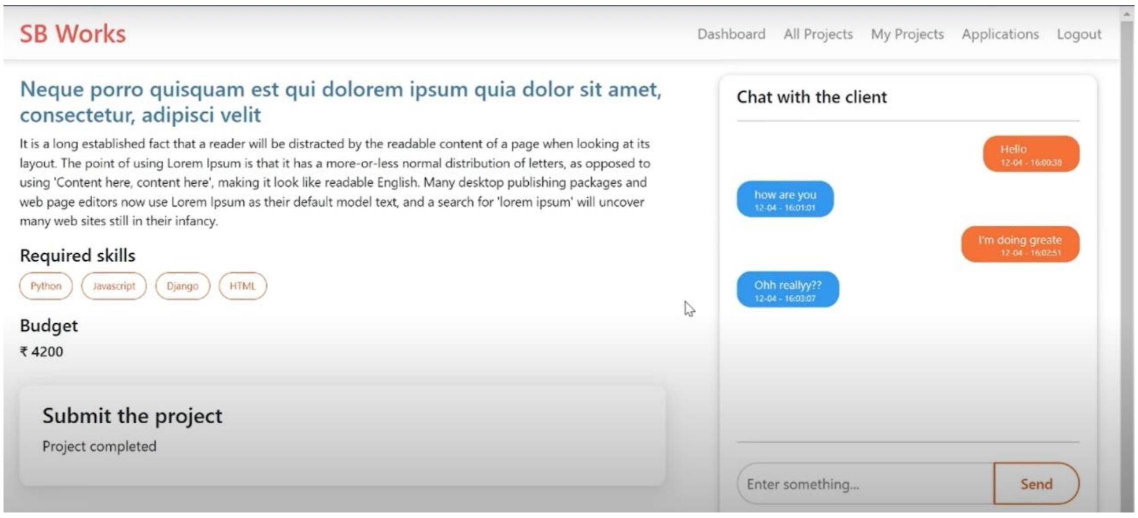
I need a web application for my e commerce store.

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:



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

Budget

₹ 4200

Submit the project

Project completed

Chat with the client

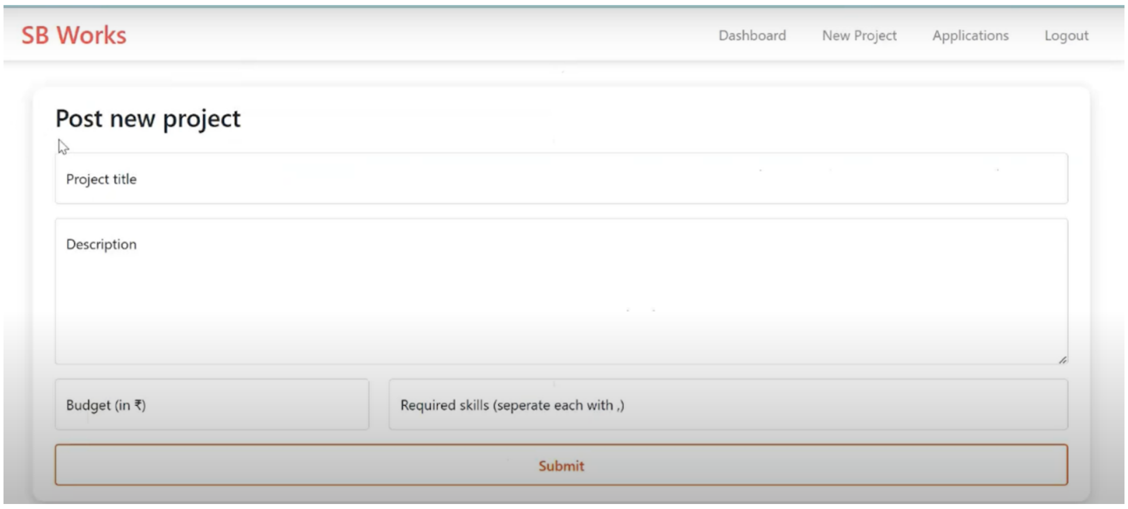
Hello
12-04 - 16:00:38

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

I'm doing greate
12-04 - 16:02:51

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

New project:



Post new project

Project title

Description

Budget (in ₹)

Required skills (seperate each with ,)

Submit

12. CHALLENGES AND SOLUTIONS

1. Data Integrity Issues

Challenge:

Maintaining data consistency between the frontend, backend, and database was complex, especially when handling multiple interconnected features like user profiles, job postings, applications, and payments. For example, incomplete or duplicate data often occurred when simultaneous users accessed or modified the same records.

Solution:

- Implemented **database transaction management** to ensure data consistency across operations.
- Used **Mongoose validation** to enforce strict schema rules, preventing invalid data from being saved.
- Incorporated **real-time syncing** using WebSockets to keep the frontend updated with the latest data changes.

2. Performance Bottlenecks

Challenge:

The application faced slower response times under high user loads, particularly when processing large datasets or running complex queries, such as filtering freelancers based on skills, ratings, and location.

Solution:

- **Indexed critical database fields** to improve query performance in MongoDB.
- Applied **pagination** to reduce the amount of data fetched and displayed at once, optimizing response times.
- Used **load balancers** and optimized backend API calls to handle concurrent user requests more efficiently.

3. Security Vulnerabilities

Challenge:

Securing sensitive user data, such as passwords, payment details, and communications, was critical. Risks included unauthorized access, data breaches, and fraudulent activities.

Solution:

- Implemented **JWT (JSON Web Token)** for secure authentication and session management.
- Used **bcrypt** to hash user passwords before storing them in the database.
- Integrated **HTTPS** to encrypt data during transmission.
- Conducted regular **security audits** and implemented input sanitization to prevent SQL injection, XSS (Cross-Site Scripting), and other vulnerabilities.

4. Cross-Browser and Device Compatibility

Challenge:

Ensuring the application worked seamlessly across various browsers and devices (desktop, mobile, and tablet) posed a challenge due to differences in rendering and user interface behavior.

Solution:

- Used **CSS media queries** and a mobile-first design approach to create a responsive UI.
- Tested the application extensively on multiple browsers and devices using tools like **BrowserStack** to identify and fix compatibility issues.

5. Real-Time Communication Implementation

Challenge:

Integrating a real-time messaging feature between clients and freelancers was challenging in terms of managing performance and ensuring message delivery without delays.

Solution:

- Used **Socket.IO** for real-time, bi-directional communication between the client and server.
- Incorporated a **retry mechanism** to handle message delivery in case of temporary network failures.
- Stored chat histories in MongoDB to allow users to retrieve their past conversations.

6. Payment Gateway Integration

Challenge:

Integrating a secure and reliable payment gateway was complex due to compliance requirements, handling payment errors, and supporting multiple payment methods.

Solution:

- Integrated **Stripe API** for secure payment processing and ease of implementation.
- Built error-handling mechanisms to detect and recover from failed transactions.
- Added support for multiple currencies and payment methods to cater to a global user base.

7. Managing Application State in Frontend

Challenge:

As the application scaled, managing the state for various features, such as user authentication, job posting workflows, and notifications, became increasingly complex.

Solution:

- Used **Redux** for global state management, ensuring a centralized and predictable state across the application.
- Implemented React's **context API** for managing simpler state requirements to reduce complexity.

8. Scalability for Future Growth

Challenge:

Ensuring the application could handle increased users, jobs, and transactions in the future required designing for scalability from the start.

Solution:

- Adopted a **modular architecture** to allow easy addition of features and services.
- Used **cloud-based infrastructure** (e.g., AWS or Azure) to scale backend resources dynamically based on user demand.
- Designed database collections with scalability in mind, avoiding complex relationships and ensuring faster queries.

13.CONCLUSION

The development of the freelancing web application using the MERN (MongoDB, Express.js, React, Node.js) stack has been a comprehensive journey in building a modern, robust, and scalable platform. The project demonstrates the power of full-stack JavaScript development, leveraging the capabilities of the MERN stack to deliver a seamless user experience for both freelancers and clients.

Key outcomes of the project include:

1. **Enhanced User Experience**: The React-based front end ensures a dynamic and interactive interface, allowing users to browse, search, and interact effortlessly.
2. **Efficient Backend Operations**: Node.js and Express.js provide a solid backend framework that efficiently handles user requests and ensures smooth application workflows.
3. **Robust Database Management**: MongoDB's flexibility and scalability enable effective storage and retrieval of user data, projects, and transactions.
4. **Core Functionalities**:
    - User authentication and authorization ensure secure access for both freelancers and clients.
    - A responsive dashboard for job posting, application tracking, and communication.
    - Integration of secure payment gateways for streamlined financial transactions.
5. **Scalability and Maintenance**: The modular architecture allows for easy updates and scalability, ensuring the application can grow to meet increasing user demands.

Future Enhancements

While the current version achieves the project's objectives, there is scope for further development:

- **Advanced Matching Algorithms**: To connect freelancers and clients more effectively based on skills, ratings, and project requirements.
- **Mobile Application Development**: Expanding accessibility via dedicated mobile apps.
- **Enhanced Security**: Incorporating advanced encryption techniques for data protection and fraud prevention.
- **Analytics and Insights**: Offering users insights into performance metrics and market trends.

14.REFERENCES

1. **React Documentation** - https://reactjs.org/docs
2. **Node.js Documentation** - https://nodejs.org/en/docs
3. **Express.js Guide** - https://expressjs.com/en/guide
4. **MongoDB Documentation** - https://www.mongodb.com/docs
5. **JWT Authentication** - https://jwt.io/introduction
6. **Cypress Testing** - https://www.cypress.io
7. **Bootstrap (for responsive design)** - https://getbootstrap.com/docs
8. **Postman (for API testing)** - https://www.postman.com
9. **MERN Stack Tutorials** - https://www.freecodecamp.org/news/mern-stack-tutorial
10. **Full-Stack React, TypeScript, and Node" by David Choi**
11. **MERN Quick Start Guide" by Eddy Wilson Iriarte Koroliova**
12. **Learning React: Modern Patterns for Developing React Apps" by Alex Banks and Eve Porcello**
13. **Node.js Design Patterns" by Mario Casciaro**
14. **MongoDB: The Definitive Guide" by Kristina Chodorow and Michael Dirolf**
15. **Pro MERN Stack: Full Stack Web App Development with Mongo, Express, React, and Node" by Vasan Subramanian**