"SB Works - Freelancing Application MERN"

1. Introduction

SB Works is an innovative and user-friendly freelancing platform designed to bridge the gap between clients seeking skilled professionals and freelancers offering their expertise. Developed using the **MERN** stack **(MongoDB, Express.js, React.js, Node.js)**, SB Works streamlines the freelancing process through a modern, intuitive, and efficient web-based interface.

This platform empowers clients to post a wide range of projects—from creative assignments to technical tasks—while enabling freelancers to explore opportunities and bid based on their skills and interests. The system promotes transparent interactions by allowing clients to review freelancer profiles, assess past work, and choose the most suitable candidate.

Key Features of SB Works:

- Simple and secure project posting and bidding system
- Real-time chat and notifications
- Admin supervision to maintain platform integrity and user trust
- Seamless work submission and feedback process

The platform ensures secure transactions, maintains quality assurance, and fosters a professional and collaborative environment for both clients and freelancers. Admins play a vital role in overseeing activity, resolving disputes, and ensuring compliance with platform guidelines.

SB Works aims to become the go-to destination for businesses and independent professionals, supporting smooth, secure, and productive freelance engagements.

2. Project Overview

■ Purpose

The purpose of the **SB Works** project is to develop a secure, responsive, and user-friendly

freelancing web platform that enables **clients** and **freelancers** to connect, collaborate, and manage projects efficiently in one place.

The platform is designed to:

- Simplify the **freelancing workflow** from project posting to submission.
- Ensure **transparency and professionalism** in client–freelancer interactions.
- Offer a **centralized system** for bidding, communication, submission, and feedback.
- Provide **admin-level monitoring** to maintain trust, security, and quality control.

SB Works aims to support the modern gig economy by giving both clients and freelancers a robust, scalable platform to work together productively and securely.

■ **Features:** Below are the key features and core functionalities of SB Works:

- **User Registration & Authentication**
  Separate roles: Client, Freelancer, and Admin
  Secure login and registration using password encryption (bcrypt)
- **Project Posting and Bidding**
  Clients can post new projects with descriptions and deadlines.
  Freelancers can browse available projects and submit bids.
- **Profile Management**
  Both clients and freelancers can maintain a public profile with relevant details like skills, project history, etc.
- **Project Assignment and Submission**
  Clients assign work to selected freelancers.
  Freelancers can submit completed work via the platform.
- **Real-time Chat System**
  Enables direct communication between freelancers and clients using Socket.IO.
- **Admin Panel**
  Admin can monitor users, projects, and communication.
  Handles conflict resolution and enforces policies.
- **Feedback and Ratings**
  Clients can review and rate freelancers after project completion.
  Helps in maintaining quality and building trust.
- **Responsive UI/UX**
  Clean and interactive user interface using React.js, Bootstrap, and Material UI.
  Works across devices with a mobile-friendly design.
- **Notifications and Updates**
  Users receive real-time updates on project status, messages, and other activities.

3. Architecture

Frontend

The frontend of SB Works is developed using **React.js**, which is a powerful JavaScript library used for building user interfaces. The entire user experience—such as login, registration, project posting, bidding, chatting, and dashboard views—is handled on the frontend.

React components are divided into different folders like:

- **Components** – for reusable UI elements like headers, forms, and cards.
- **Pages** – for route-based screens like Home, Profile, Dashboard.
- **Context** – for managing global state such as user authentication and project data.

React Router is used for page navigation, and Axios is used to make HTTP requests to the backend. Styling is managed using **Bootstrap** and **Material UI**, which makes the interface responsive and user-friendly.

Backend

The backend is built using **Node.js** and **Express.js**. It serves as the core of the application, handling all business logic, routing, and data processing.

The backend handles:

- User authentication (register, login)
- Project posting and retrieval
- Bidding system
- Chat communication (with the help of Socket.IO)
- Admin operations like monitoring and conflict resolution

All backend routes are connected to the database using **Mongoose**, which acts as a bridge between Node.js and MongoDB.

Database

The database used is **MongoDB**, which is a NoSQL database that stores data in JSON-like format (called BSON).

The main collections (tables) in MongoDB are:

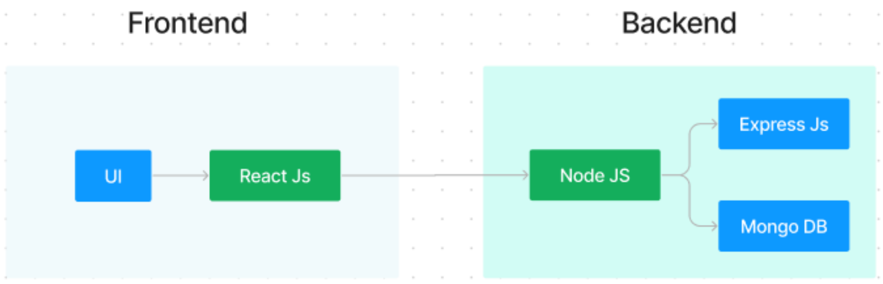
- **Users** – stores details like name, email, password, role (client/freelancer/admin)
- **Projects** – stores project details like title, description, budget, client ID, and status

- **Applications** – stores freelancer bids for projects, with details like freelancer ID, bid amount, and proposal
- **Chats** – stores messages between users
- **Messages** – stores individual messages with timestamps and sender/receiver IDs

Mongoose schemas define how data is structured in each collection and make it easy to perform operations like create, read, update, and delete (CRUD).

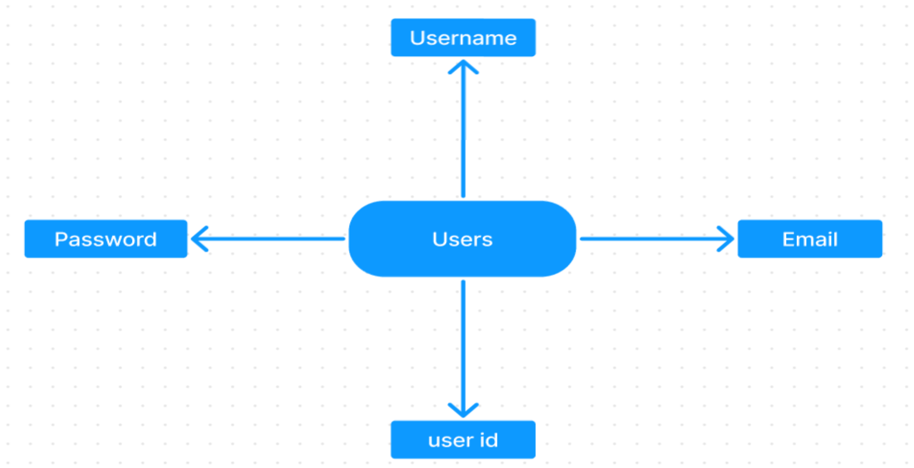
The backend connects to MongoDB using a connection string, and all data interactions are handled through models and controllers.

TECHNICAL ARCHITECTURE

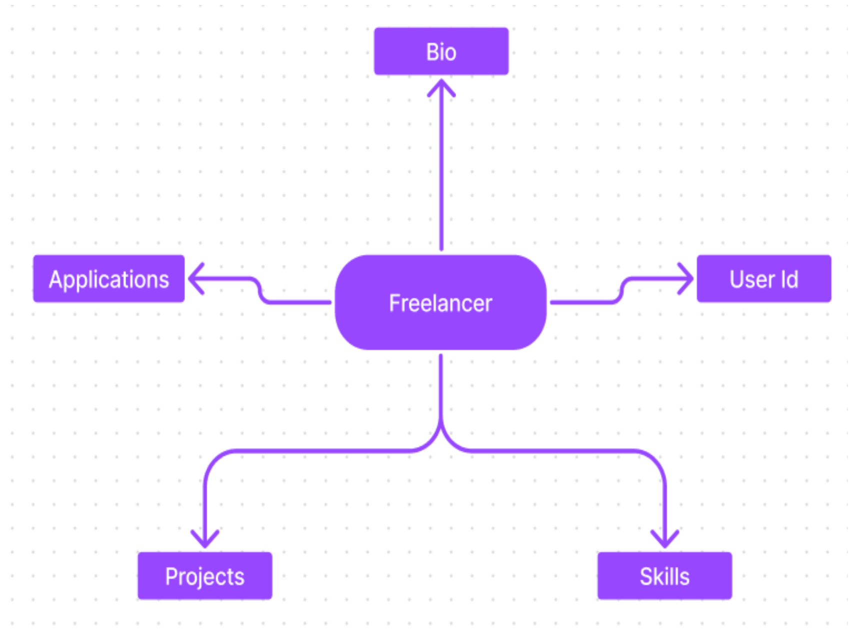


ER DIAGRAM

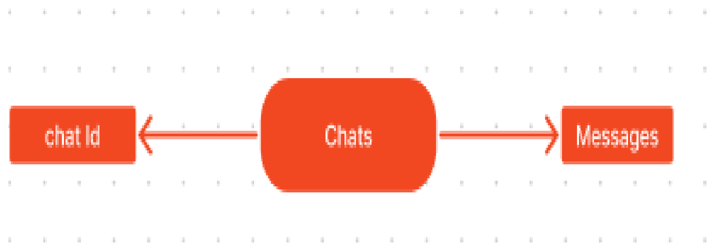
● The **Users** entity is the foundational part of the SB Works platform. It stores key login and identification details such as Username, Email, Password, and a unique User ID. This entity helps differentiate between different user roles (freelancer, client, or admin) and facilitates secure access to the platform



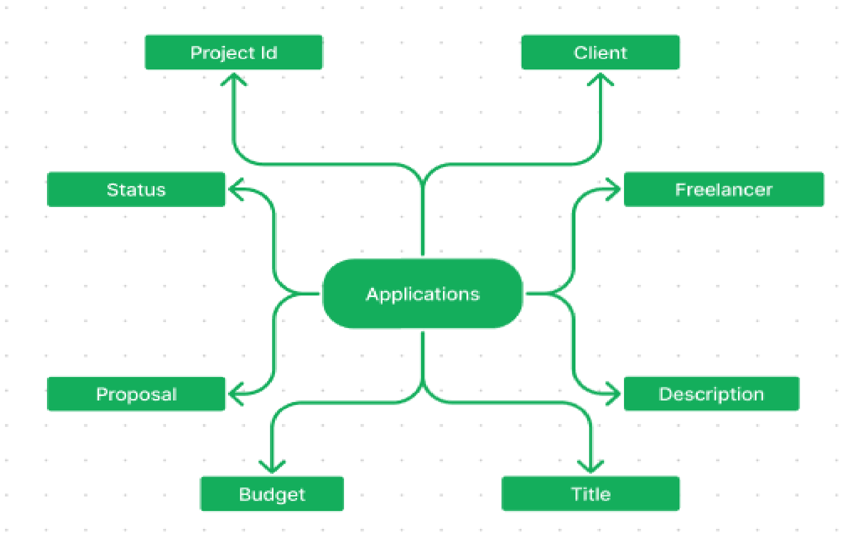
- The **Freelancer** entity extends the base user by adding detailed freelancing attributes. It includes fields like Bio (description), Skills, and maintains associations with Projects (assigned/completed) and Applications (bids submitted). It links to the Users entity via User ID and helps track a freelancer's professional profile and activity



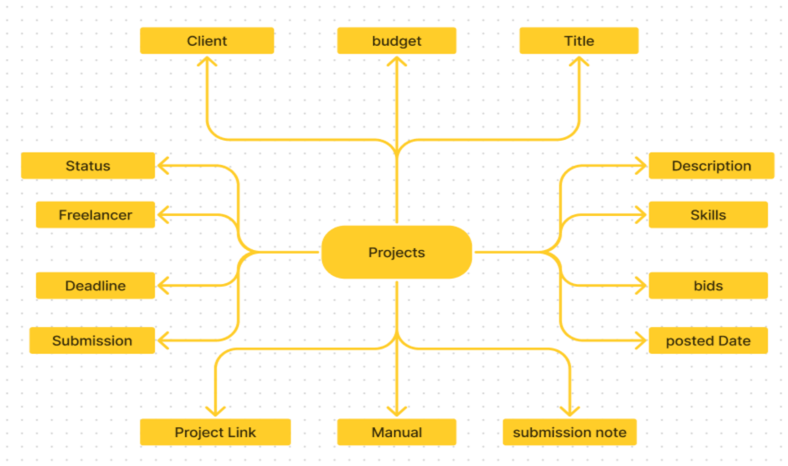
● The **Chats** entity enables direct communication between clients and freelancers. Each chat is identified using a unique Chat ID and stores Messages. This structure ensures all discussions related to a project are centralized and accessible, supporting real-time collaboration and history tracking.



● The **Applications** entity manages all project bids made by freelancers. It contains detailed information including Project ID, Client, Freelancer, Title, Description, Budget, Proposal, and Status. This structure allows clients to review and accept or reject proposals, ensuring organized and fair project allocation.



● The **Project** entity represents all the jobs or tasks posted by clients on the SB Works platform. It includes fields such as Title, Description, Budget, Skills Required, Client ID, Posted Date, and Status (e.g., Available, Assigned, Completed). It may also include optional submission-related fields like Project Link, Manual Link, and Submission Description. Each project connects with multiple applications (bids), and upon assignment, links to a freelancer, enabling complete tracking from posting to completion.



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.

4. Setup Instructions

Prerequisites

Before setting up the SB Works application, make sure the following software dependencies are installed on your system:

1. **Node.js and npm (Node Package Manager)**
   - Required to run JavaScript on the backend and install packages
   - Download: https://nodejs.org/en/download
2. **MongoDB**
   - Required to store application data (users, projects, messages)
   - Download: https://www.mongodb.com/try/download/community
3. **Git**
   - Version control tool used to clone the project repository
   - Download: https://git-scm.com/downloads
4. **Visual Studio Code** (or any preferred code editor)
   - Download: https://code.visualstudio.com/download

Installation Steps

Follow these steps to run the project on your local system:

1. Clone the GitHub Repository:

```
git clone https://github.com/Pawan-o3/Freelancing.git
cd Freelancing
```

2. Install Dependencies for Frontend:

```
cd client
npm install
```

3. Install Dependencies for Backend:

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

4. Start the Backend Server:

```
node index.js
```

5. Start the Frontend Development Server:

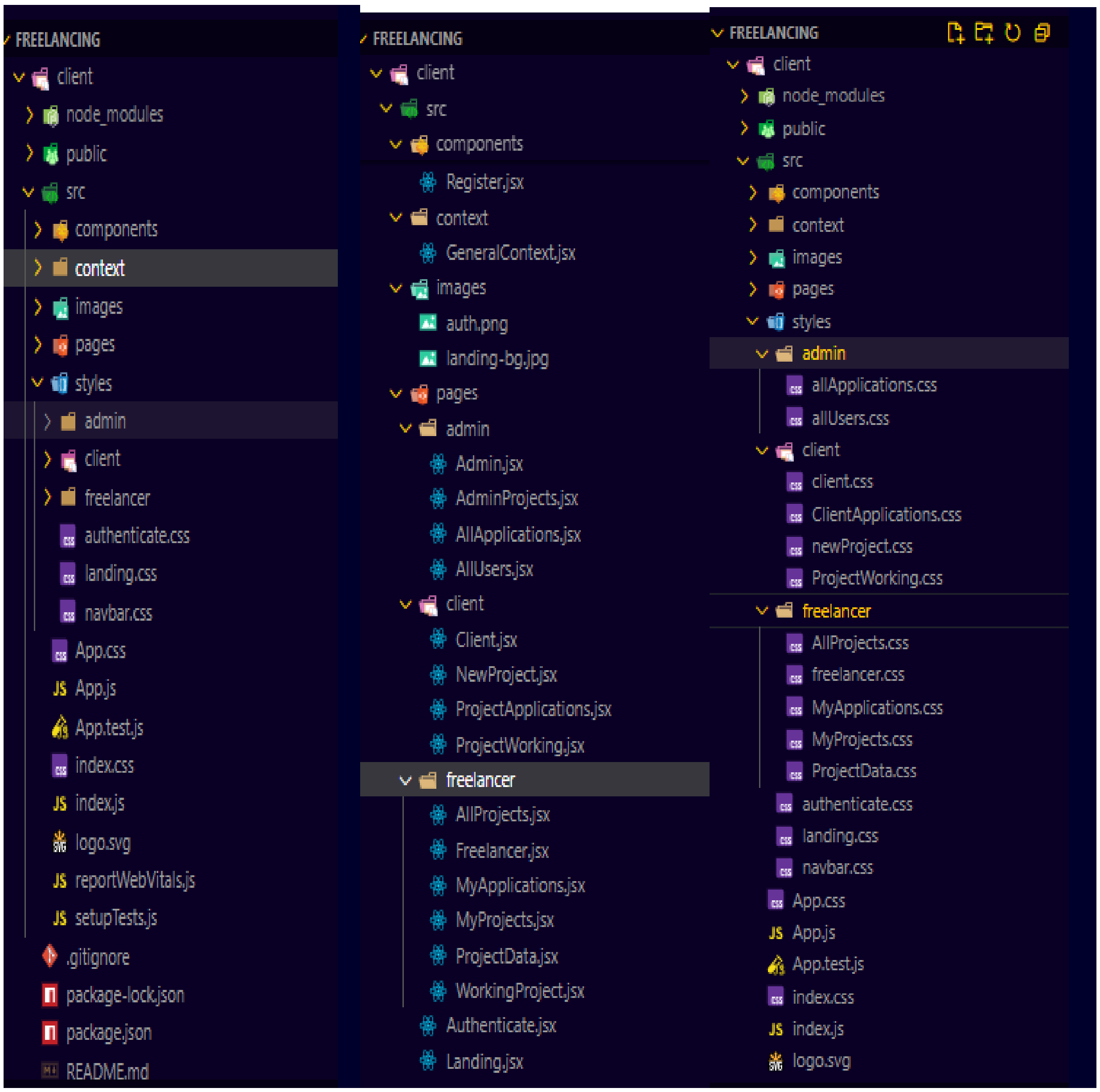
```
cd ../client
npm start
```

6. Access the Application:
Open your browser and go to: http://localhost:3000

5. Folder Structure

Client (Frontend - React.js)

The **client** folder contains the entire frontend code built using **React.js**. Below is a description of the main folders and files inside the `client` directory



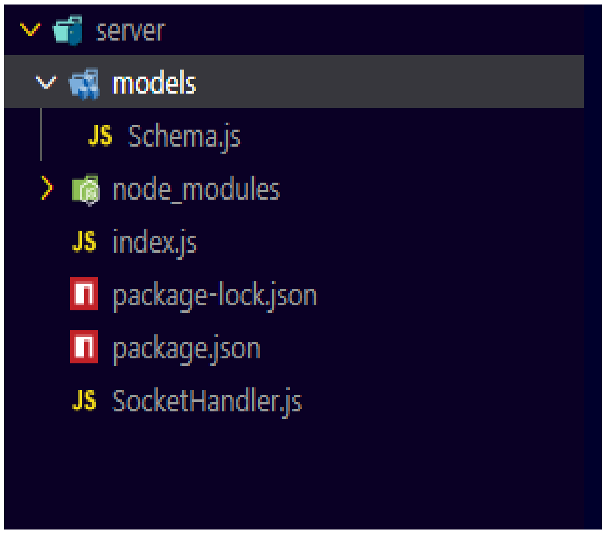
```
FREELANCING
∨ client
  › node_modules
  › public
  ∨ src
    › components
    › context
    › images
    › pages
    ∨ styles
      › admin
      › client
      › freelancer
        authenticate.css
        landing.css
        navbar.css
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
    README.md
```

```
FREELANCING
∨ client
  ∨ src
    ∨ components
        Register.jsx
    ∨ context
        GeneralContext.jsx
    ∨ images
        auth.png
        landing-bg.jpg
    ∨ 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
```

```
FREELANCING
∨ client
  › node_modules
  › public
  ∨ src
    › components
    › context
    › images
    › pages
    ∨ styles
      ∨ admin
          allApplications.css
          allUsers.css
      ∨ client
          client.css
          ClientApplications.css
          newProject.css
          ProjectWorking.css
      ∨ freelancer
          AllProjects.css
          freelancer.css
          MyApplications.css
          MyProjects.css
          ProjectData.css
        authenticate.css
        landing.css
        navbar.css
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
```

Server (Backend - Node.js + Express.js)

The **server** folder contains all backend logic, API routes, database connections, and WebSocket configuration using **Express.js** and **Socket.IO**.



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

6. Running the Application

To run the **SB Works – Freelancing Application** on your local machine, follow the steps below. This assumes you have already installed the required dependencies and set up your environment variables as described in the setup instructions.

Step-by-Step Commands

1. Start the Backend Server

Navigate to the `server` directory and run:

```
cd server
npm start
```

- This will start the **Node.js + Express.js** backend server.
- It will run on the default port defined in your `.env` file (usually `http://localhost:5000`).

2. Start the Frontend Server

Open a new terminal window or tab. Navigate to the `client` directory and run:

```
cd client
npm start
```

- This will start the **React.js** frontend development server.
- It will run on `http://localhost:3000` by default.

Access the Application

Once both servers are running:

- Open your browser and visit: **http://localhost:3000**

You should now see the **SB Works platform homepage**. From here, you can register, log in, post projects, bid on them, chat, and test the full flow of the application.

7. API Documentation

The backend of SB Works exposes several RESTful API endpoints to support features like user authentication, project posting, bidding, and communication.

1. Register a New User

- **URL**: `/api/register`
- **Method**: `POST`
- **What it does**: Creates a new user (client or freelancer)

Data to send:

```
{
  "name": "Pawan",
  "email": "pawan@example.com",
  "password": "123456",
  "role": "freelancer"
}
```

What it returns:

```
{
  "message": "User registered successfully"
}
```

2. Login User

- **URL**: `/api/login`

- **Method**: `POST`
- **What it does**: Logs in an existing user

Data to send:

```
{
  "email": "pawan@example.com",
  "password": "123456"
}
```

What it returns:

```
{
  "message": "Login successful",
  "user": {
    "id": "abc123",
    "name": "Pawan",
    "role": "freelancer"
  }
}
```

3. Post a New Project

- **URL**: `/api/project`
- **Method**: `POST`
- **What it does**: Lets a client post a new project

Data to send:

```
{
  "title": "Create a Website",
  "description": "Need a React-based website",
  "clientId": "user123"
}
```

What it returns:

```
{ "message": "Project created successfully" }
```

4. Get All Projects

- **URL**: `/api/projects`
- **Method**: `GET`
- **What it does**: Shows all the available projects

What it returns:

```
[
  {
    "title": "Create a Website",
    "description": "Need a React-based website"
  },
  {
    "title": "Design a Logo",
    "description": "Looking for a creative logo"
  }
]
```

5. Apply for a Project (Bid)

- **URL**: `/api/apply`
- **Method**: `POST`
- **What it does**: Lets a freelancer bid on a project

Data to send:

```
{
  "freelancerId": "freelancer123",
  "projectId": "project001",
  "proposal": "I will complete it in 3 days",
  "budget": 1000
}
```

What it returns:

```
{  "message": "Application submitted" }
```

6. Chat (Real-time)

- Chat messages are not sent through normal APIs.
- Instead, **Socket.IO** is used for real-time chat.

8. Authentication

Authentication and authorization in the **SB Works** platform are handled on the backend using **basic JWT-less session logic**. Here's how it works based on your project code:

1. Login & Registration

- The **login** and **registration** processes are handled using **Express.js** API routes.
- Passwords are **hashed using `bcrypt`** before saving to the database for security.
- During login:
    - The system matches the email with the database.
    - Then, it compares the password using `bcrypt.compare()`.

2. Password Security (Using bcrypt)

- On registration:
    - Passwords are encrypted with `bcrypt.hash()` before storing them in MongoDB.
- On login:
    - The entered password is compared with the stored hashed password using `bcrypt.compare()`.

This ensures that even if someone gains database access, the original passwords are not visible.

3. Role-Based Access (Client / Freelancer / Admin)

- Every user in the database has a **`role`** field: `"client"`, `"freelancer"`, or `"admin"`.
- This helps control access:
    - Clients can post and review projects.
    - Freelancers can bid and submit work.
    - Admins can monitor the platform and manage users/projects.

Example from DB:

```
{
"name":"Pawan",
  "email": "pawan@example.com",
  "password": "encrypted...",
  "role": "freelancer"
}
```

4. Session Management (No JWT used yet)

- Your current project **does not use JWT tokens** or session cookies for managing sessions.
- After login, the frontend stores **user info (like role, ID)** in **React Context or LocalStorage**.
- This is used to:
  - Show/hide different pages (based on role)
  - Keep the user logged in until the browser is closed

Note: For future improvement, you can add **JWT (JSON Web Tokens)** for secure and token-based authentication.

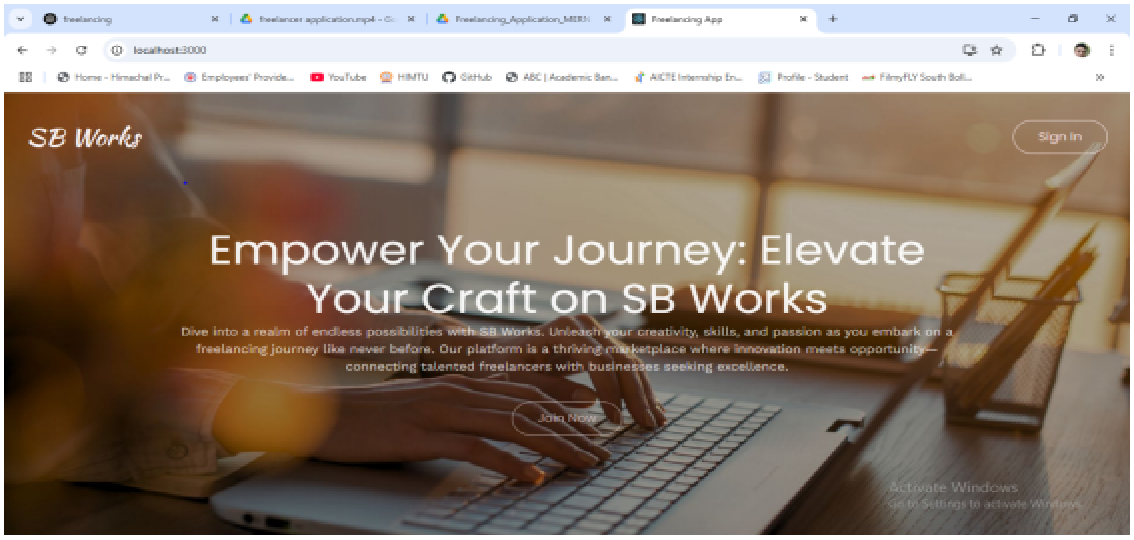
5. Frontend State Handling

- Once login is successful:
  - The user data is saved in the frontend context (`GeneralContext.jsx`)
  - This keeps the user logged in across different pages
- When the user logs out:
  - Context is cleared, and the user is redirected to the login page

9. User Interface

The **SB Works** application offers a clean, responsive, and user-friendly interface built using **React.js**.

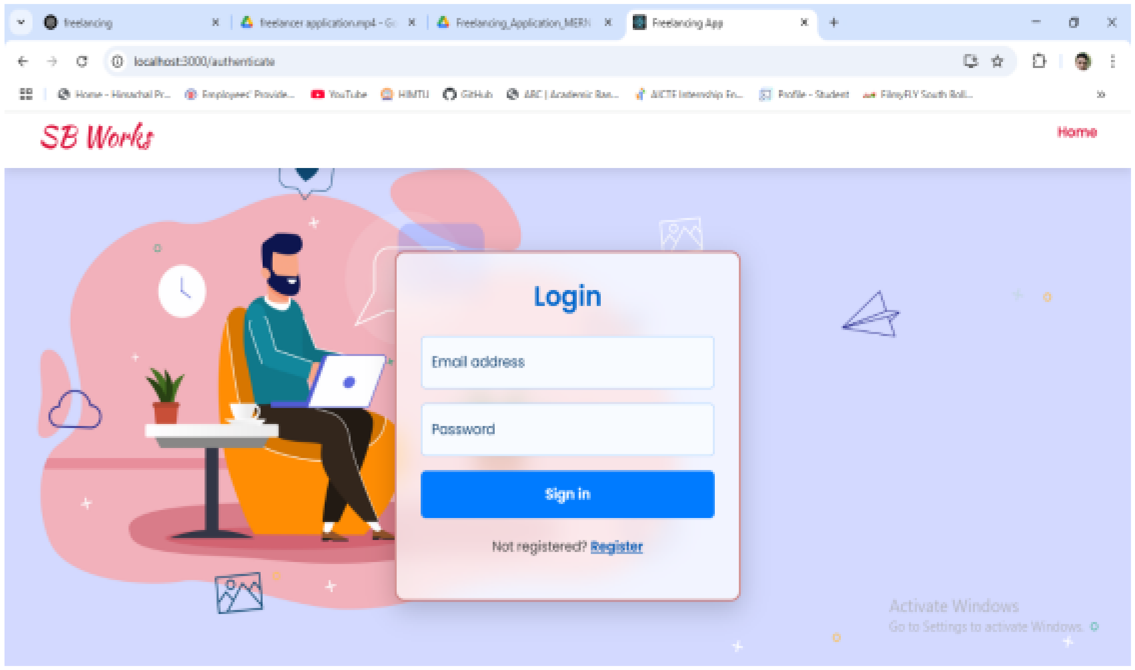
Landing page:



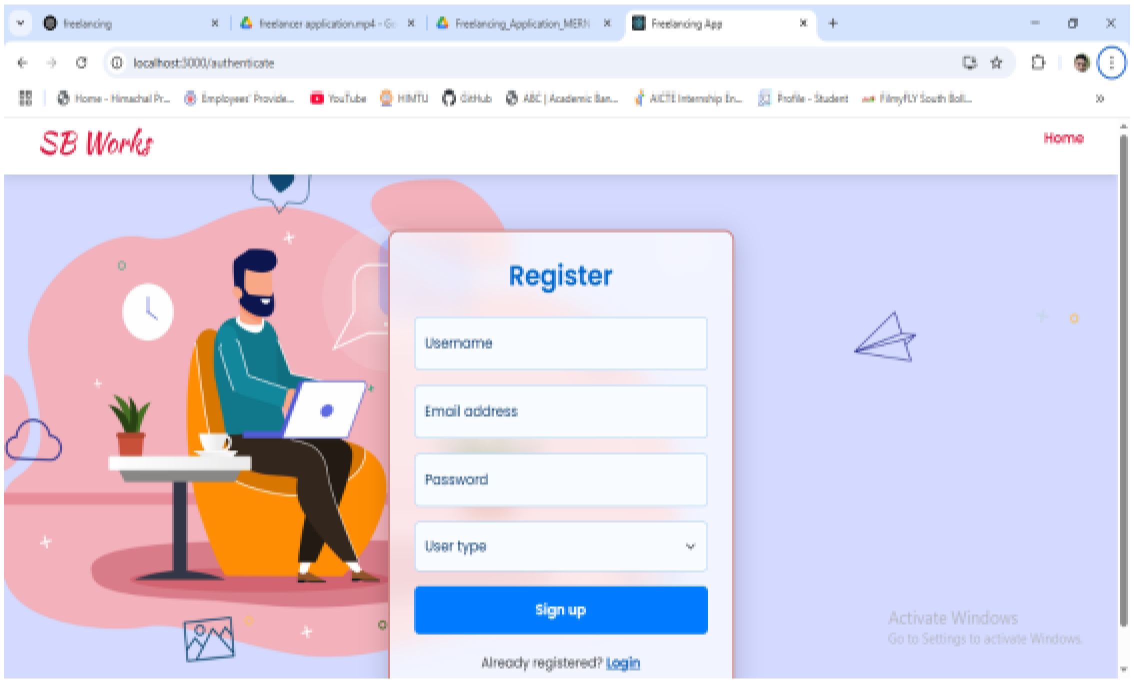
Empower Your Journey: Elevate Your Craft on SB Works

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.

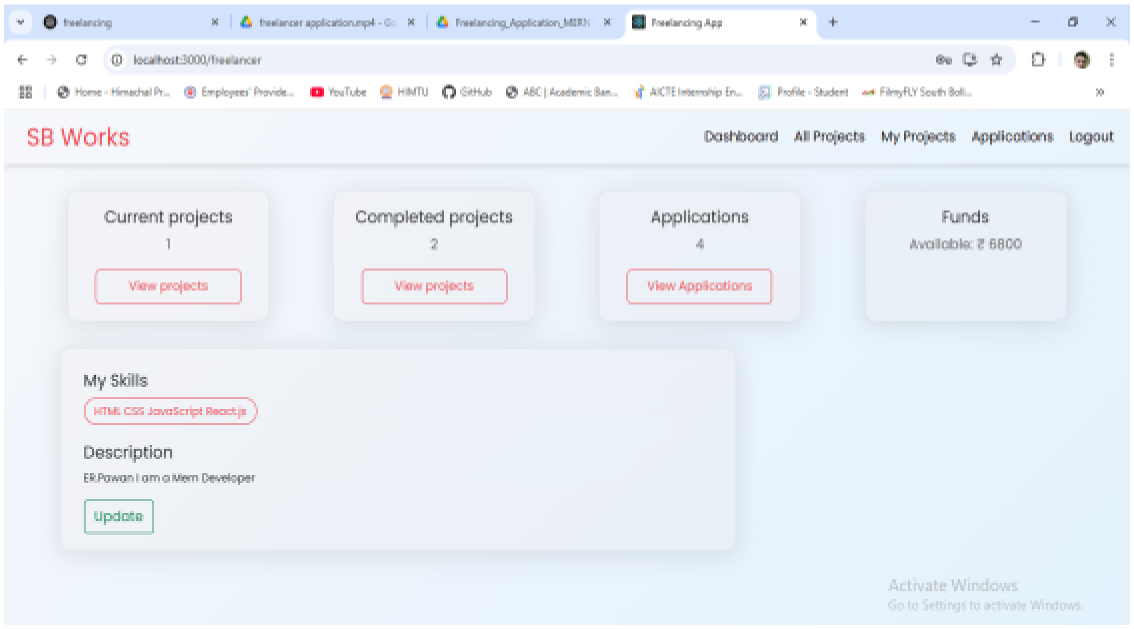
Authentication:



Register:


localhost:3000/authenticate
SB Works
Home
Register
Username
Email address
Password
User type
Sign up
Already registered? Login
Activate Windows
Go to Settings to activate Windows.

Freelancer dashboard:



Current projects
1

Completed projects
2

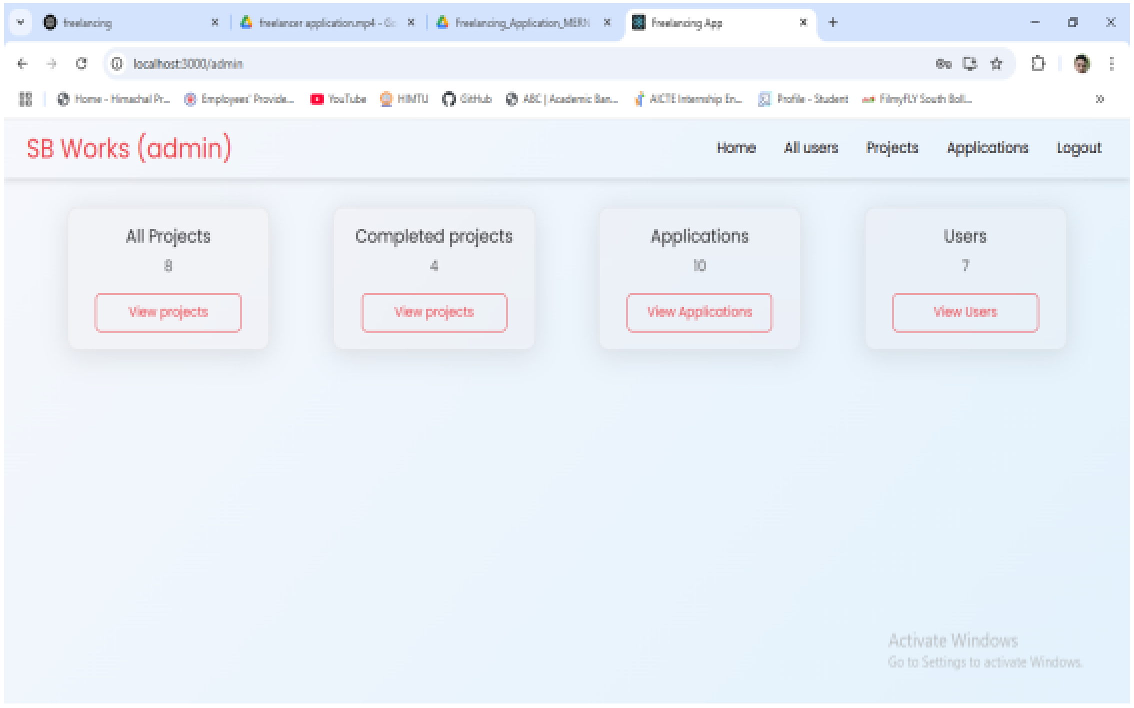
Applications
4

Funds
Available: ₹ 6800

My Skills
HTML CSS JavaScript React.js

Description
ER.Pawan I am a Mern Developer

Admin dashboard:



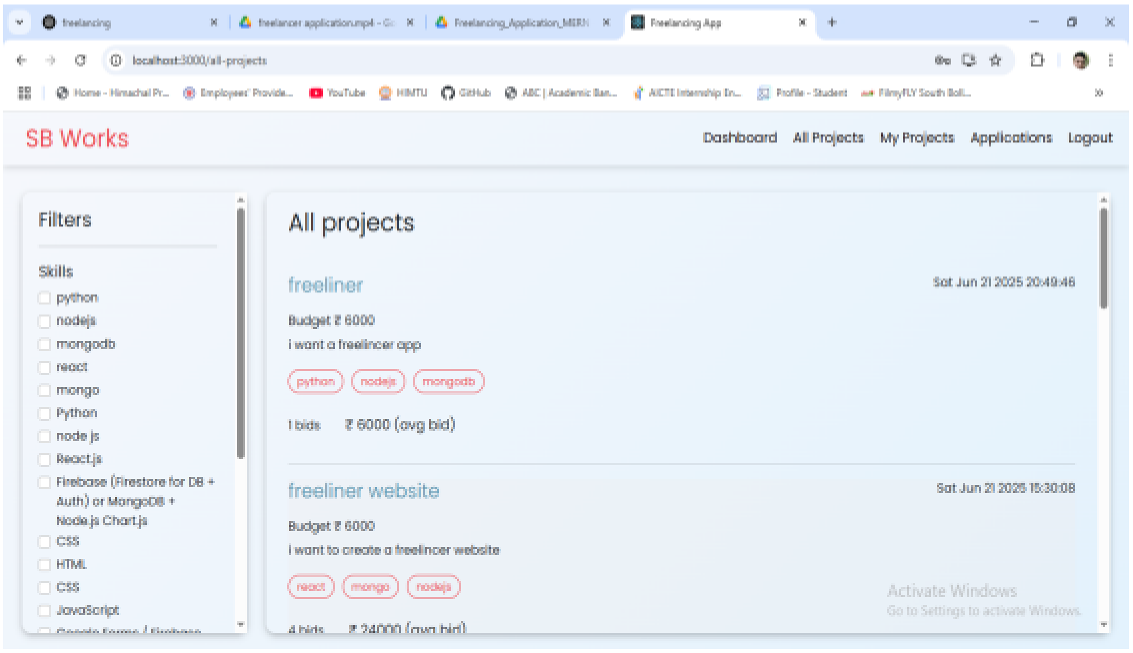
All Projects
8

Completed projects
4

Applications
10

Users
7

All projects:


freelancing
freelancer application.mp4
Freelancing_Application_MERN
Freelancing App
localhost:3000/all-projects
SB Works
Dashboard
All Projects
My Projects
Applications
Logout
Activate Windows
Go to Settings to activate Windows.

Filters

Skills

- python
- nodejs
- mongodb
- react
- mongo
- Python
- node js
- React.js
- Firebase (Firestore for DB + Auth) or MongoDB + Node.js Chart.js
- CSS
- HTML
- CSS
- JavaScript

All projects

freeliner Sat Jun 21 2025 20:49:46

Budget ₹ 6000

i want a freelincer app

python nodejs mongodb

1 bids ₹ 6000 (avg bid)

freeliner website Sat Jun 21 2025 15:30:08

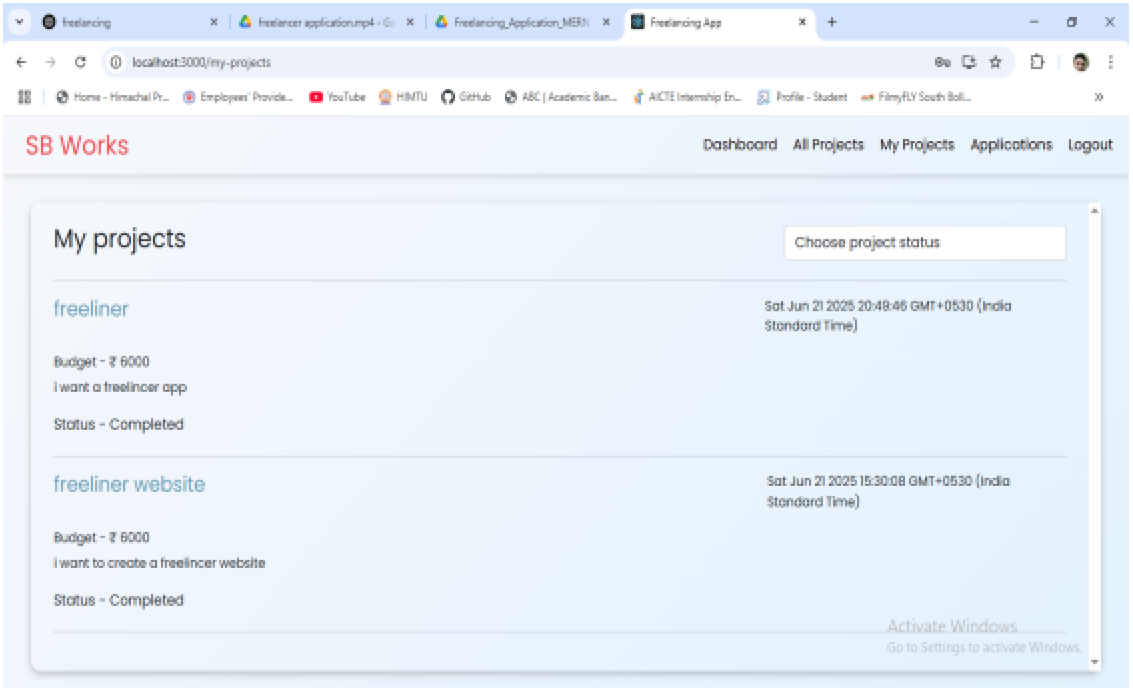
Budget ₹ 6000

i want to create a freelincer website

react mongo nodejs

4 bids ₹ 24000 (avg bid)

Freelance Projects:



My projects

freeliner Sat Jun 21 2025 20:49:46 GMT+0530 (India Standard Time)

Budget - ₹ 6000

i want a freelincer app

Status - Completed

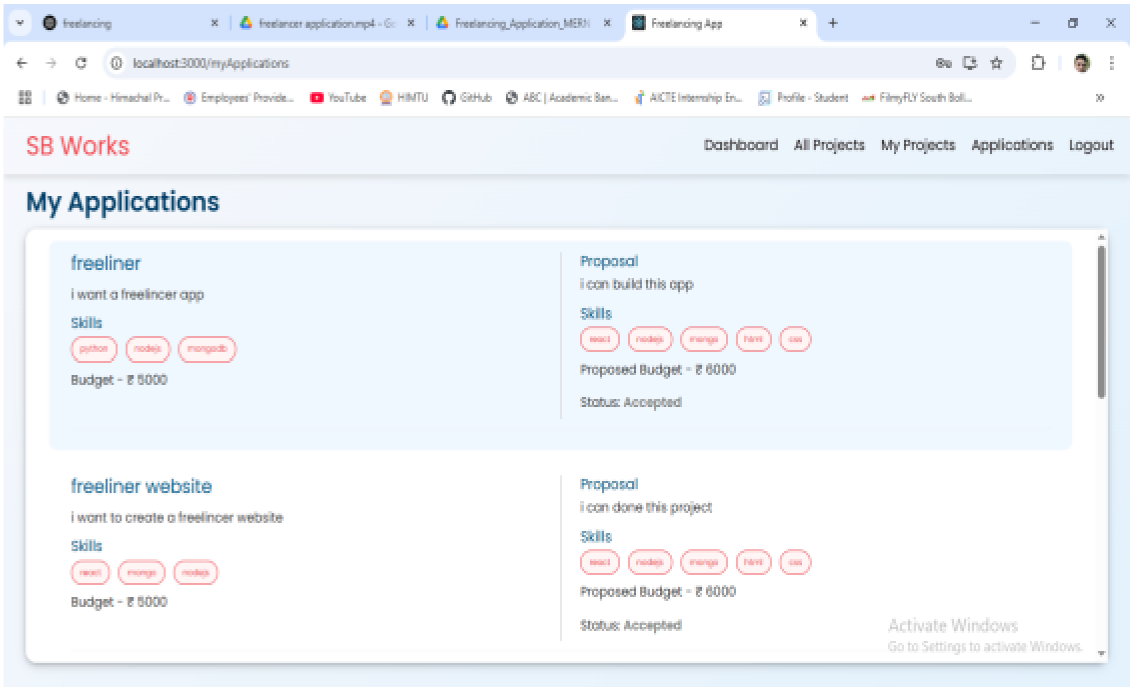
freeliner website Sat Jun 21 2025 15:30:08 GMT+0530 (India Standard Time)

Budget - ₹ 6000

i want to create a freelincer website

Status - Completed

Applications:



My Applications

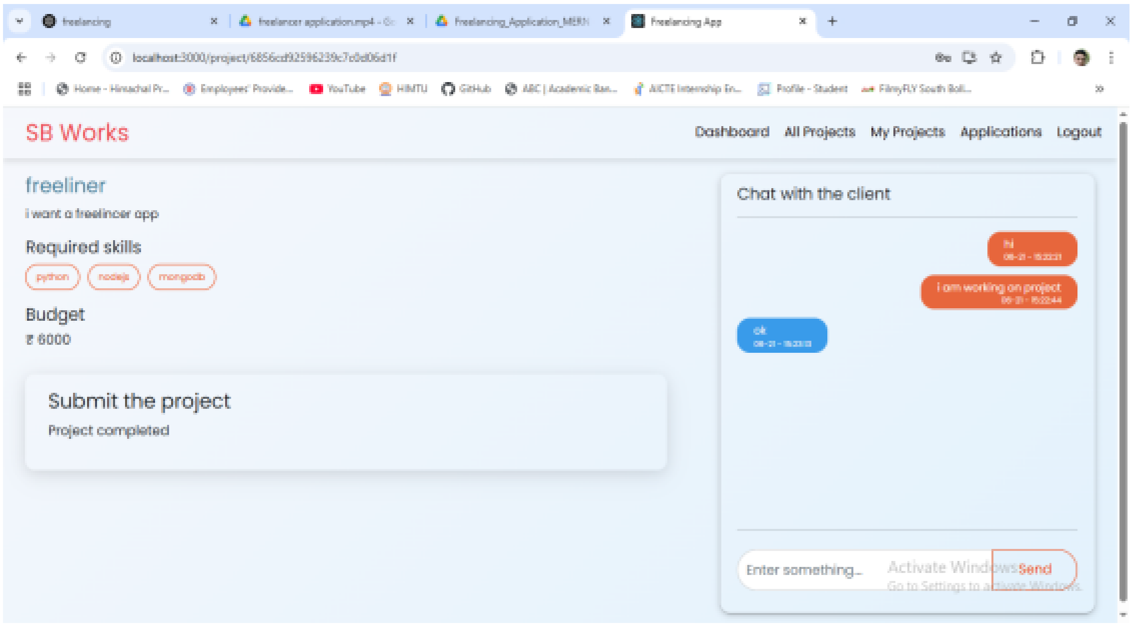
freeliner
i want a freelincer app
Skills
python nodejs mongodb
Budget - ₹ 5000

Proposal
i can build this app
Skills
react nodejs mongo html css
Proposed Budget - ₹ 6000
Status: Accepted

freeliner website
i want to create a freelincer website
Skills
react mongo nodejs
Budget - ₹ 5000

Proposal
i can done this project
Skills
react nodejs mongo html css
Proposed Budget - ₹ 6000
Status: Accepted

Project page:



freeliner
i want a freelincer app
Required skills
python nodejs mongodb
Budget
₹ 6000

Submit the project
Project completed

Chat with the client

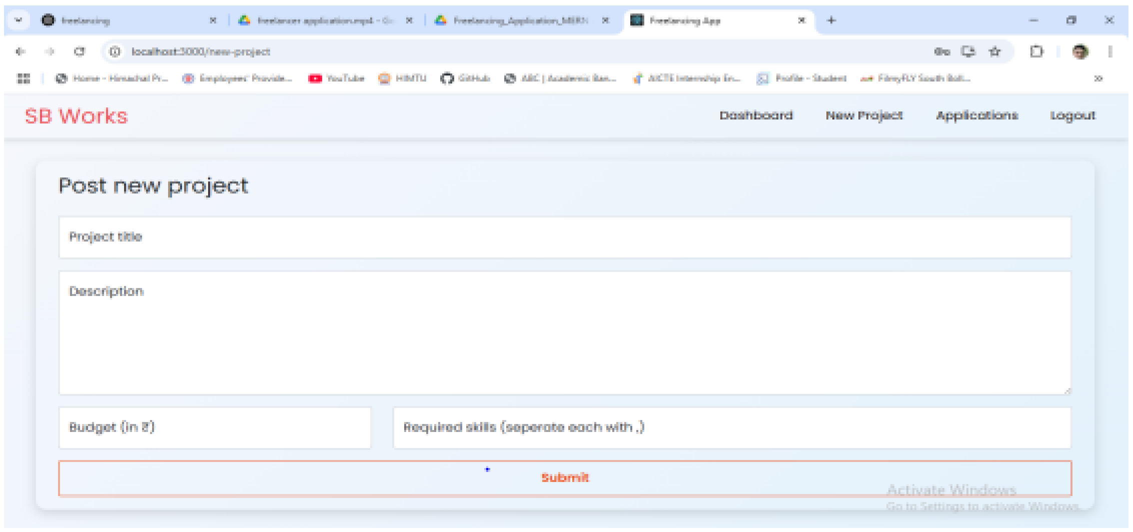
hi
08-21 - 15:22:21

I am working on project
08-21 - 15:22:44

ok
08-21 - 15:23:13

Enter something...

New project:



Post new project

Project title

Description

Budget (in ₹)

Required skills (seperate each with ,)

10. Testing

Testing was an important part of the development of the **SB Works** application to ensure all features worked smoothly and without bugs. The goal was to make sure the platform functions correctly from the user's point of view.

1. Testing Strategy

The application was tested using **manual testing** methods. Each part of the system was checked by performing actual actions like user registration, project posting, applying for projects, and sending messages.

The testing covered:

- **User authentication** (register and login)
- **Project posting and viewing**
- **Bid submission**
- **Real-time chat**
- **Role-based dashboards** (client, freelancer, admin)
- **Data storage and retrieval in MongoDB**

2. Tools Used

Tool	Purpose
Postman	To test and verify all backend API endpoints manually
MongoDB Compass	To view and confirm the data stored in the MongoDB database
Manual Testing	To test the complete flow by using the app as different users

3. Example Test Scenarios

- Register as a client and freelancer
- Login with correct and incorrect credentials
- Post a project and check if it appears in the project list
- Submit a bid as a freelancer and view it in the client dashboard
- Use the chat feature between two users
- Verify project and user data in MongoDB Compass

Testing Conclusion: Testing was done manually to ensure that all the main features are working properly. The combination of Postman, MongoDB Compass, and hands-on testing helped to identify and fix errors during development.

11. Demo

To showcase the working features and user experience of the **SB Works** platform, a recorded video demo has been prepared.

The demo includes:

- User registration and login (Client and Freelancer)
- Posting a project as a client
- Viewing and bidding on a project as a freelancer
- Admin monitoring features
- Real-time chat between client and freelancer
- Project submission.

Demo Video

Link:https://drive.google.com/file/d/1FtU3sDtQrkgpJtNGsJfPH8ytkm1edV1l/view

12. Known Issues

Although the SB Works application is functional and supports core freelancing features, there are a few known issues and limitations that may affect user experience or require attention during future development:

1. No JWT or Token-Based Authentication

- The application currently does not use **JWT** or any secure session/token management.
- Once logged in, user data is stored in **local storage**, which can be manipulated or lost on browser refresh.

2. Limited Form Validation

- Some input forms (like login, project posting) lack strong validation.
- Users can submit incomplete or invalid data unless validated manually.

3. No Profile Editing

- Users cannot update their profile information (like name, email, password) after registration.
- Profile management features are currently not implemented.

4. No Image/File Upload Feature

- The platform does not currently support uploading project files, profile pictures, or attachments during communication.

5. Chat is Basic

- The chat system works using **Socket.IO**, but:
    - It does not support message history storage with timestamps.
    - It may not work properly on network disconnection or page refresh.

6. No Email Verification or Password Reset

- New users can register with any email; there is no verification link sent.
- If a user forgets their password, there's no reset mechanism.

7. Admin Role is Functional but Basic

- The admin panel lacks features like detailed analytics, user ban, or automated conflict handling.

13. Future Enhancements

The SB Works platform is currently functional with essential freelancing features. However, several improvements and new features can be added in the future to enhance usability, security, and overall user experience.

1. JWT Authentication and Secure Sessions

- Implement **JSON Web Tokens (JWT)** for secure login and protected routes.
- Prevent unauthorized access to sensitive data or pages.

2. Profile Management

- Allow users to **update their profile** (name, email, password, skills, etc.).
- Add **profile pictures** and **freelancer portfolios**.

3. File Upload Support

- Enable clients and freelancers to **upload documents, images, or zip files** with projects and submissions.

4. Real-Time Chat Improvements

- Store **message history** in MongoDB with timestamps.
- Show **online/offline status**, typing indicator, and chat notifications.

5. Email Verification & Password Reset

- Send **email verification** after registration.
- Add "**Forgot Password**" and **password reset link** functionality.

6. Reviews and Ratings

- Allow clients to **rate freelancers** after project completion.
- Display average ratings on freelancer profiles.

7. Notification System

- Add in-app and email **notifications** for actions like new bids, project updates, and messages.

8. Admin Panel Enhancements

- Add **user management** tools (ban, edit roles).
- Display **platform analytics** (total users, projects, revenue).

9. Mobile App Version

- Build a **React Native** mobile version of the platform for wider accessibility.

10. Payment Integration

- Integrate **secure payment gateways** like Razorpay, PayPal, or Stripe for transactions between clients and freelancers.