Project Documentation:

SB Works (FreelanceFinder)

1. Introduction

SB Works is a freelancing platform designed to connect clients with skilled freelancers, facilitating project posting, bidding, communication, and project completion. The platform prioritizes efficiency and transparency.

2. Key Features and Functionality

- **For Clients:**
  - Post diverse projects (creative to technical).
  - Review freelancer profiles and past work.
  - Select the ideal candidate for their project.
  - Communicate and collaborate with freelancers within the platform.
  - Review submitted work and provide feedback.
  - Receive real-time updates and notifications.
- **For Freelancers:**
  - Bid on projects based on expertise.
  - Benefit from a straightforward project submission process.
  - Submit work directly through the platform.
  - Showcase skills and build a portfolio.
- **Admin Team:**
  - Ensures integrity and security of transactions.
  - Guarantees reliability and quality of freelancers.
  - Facilitates smooth communication between clients and freelancers.
  - Monitors data integrity and security.

    - Enforces platform policies.
    - Resolves conflicts.
    - Provides user support.
    - Responsible for platform maintenance and improvement.

3. Scenario-Based Case Study: Sarah's Journey

Sarah, a graphic design graduate, uses SB Works to build her freelance portfolio.

- **Finding the Perfect Project:** Sarah finds a logo redesign project from "Sugar Rush" bakery, with a detailed description of brand identity and target audience.
- **Bidding with Confidence:** She reviews the bakery's previous marketing materials and submits a proposal with her relevant experience and portfolio samples.
- **Communication & Collaboration:** "Sugar Rush" selects Sarah, and they use the platform's integrated chat system for communication and refining the design.
- **Delivery & Feedback:** Sarah submits the logo design through the platform, and "Sugar Rush" provides feedback, allowing for revisions.
- **Building a Thriving Career:** After successful completion and a positive review, Sarah's profile gains traction, encouraging her to seek more projects and build her freelance career on SB Works.

4. Technical Architecture

SB Works follows a client-server model.

- **Frontend:**
    - User interface, presentation layer.
    - Built with React.js.
    - Integrates Axios for communication with the backend via RESTful APIs.
    - Leverages Bootstrap and Material UI for a real-time, visually appealing interface.
    - Client-side code consists of reusable components and pages (e.g., profiles, projects, chat).
    - Shared data managed with React Context.
    - Uses

socket.io-client for real-time communication.

- **Backend:**
    - Manages server-side logic and communication.
    - Utilizes Express.js framework for robust request and response handling.
    - Node.js handles API requests for user management, project actions, and communication.
    - Includes

bcrypt for security, body-parser for request body parsing, and cors for cross-origin requests.

    - http and socket.io are also used.

- **Database:**
    - Relies on MongoDB for data storage and retrieval.
    - Offers a scalable and efficient solution for various data.
    - Mongoose models ensure structured interaction with the database.
    - Collections include: Users, Projects, Applications, Chat, and Freelancer.

Architecture Diagram: The provided diagram illustrates the flow: Frontend (UI (React), Socket IO) -> Backend (Express JS, Node.JS) -> Database (Mongoose, MongoDB).

5. ER Diagram (Entity-Relationship Diagram)

The ER diagrams outline the relationships between key entities:

- **Users:** username, password, email, user_id.
- **Freelancer:** Bio, Applications, User Id, Projects, Skills.
- **Chats:** chat Id, Messages.
- **Applications:** Project Id, Client, Freelancer, Status, Proposal, Budget, Description, Title.
- **Projects:** Client, budget, Title, Description, Skills, bids, posted Date, Status, Freelancer, Deadline, Submission, Project Link, Manual, submission note.

6. Project Structure

The project is divided into client (frontend) and server (backend) directories.

- **Client Folder Structure:**

- node_modules
- public
- src
    - components (Login.jsx, Navbar.jsx, Register.jsx)
    - context (GeneralContext.jsx)
    - images
    - pages
        - admin (Admin.jsx, AdminProjects.jsx, AllApplications.jsx, AllUsers.jsx)
        - client (Client.jsx, NewProject.jsx, ProjectApplications.jsx, ProjectWorking.jsx)
        - freelancer (AllProjects.jsx, Freelancer.jsx, MyApplications.jsx, MyProjects.jsx, ProjectData.jsx, WorkingProject.jsx)
        - Authenticate.jsx
        - Landing.jsx
    - styles
    - App.css
    - App.js
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - reportWebVital.js
    - setupTests.js
- .gitignore
- package-lock.json
- package.json

  - README.md

- **Server Folder Structure:**
  - node_modules
  - index.js
  - package-lock.json
  - package.json
  - Schema.js
  - SocketHandler.js

7. Prerequisites

To develop a full-stack application using Express.js, MongoDB, and React.js, the following are required:

- **Node.js and npm:** JavaScript runtime environment for server-side code.
  - Download:

https://nodejs.org/en/download/

  - Installation instructions:

https://nodejs.org/en/download/package-manager/

- **Express.js:** Fast and minimalist web application framework for Node.js.
  - Installation:

npm install express

- **MongoDB:** Flexible and scalable NoSQL database.
  - Download:

https://www.mongodb.com/try/download/community

  - Installation instructions:

https://docs.mongodb.com/manual/installation/

- **React.js:** JavaScript library for building user interfaces.
  - Installation guide:

https://reactjs.org/docs/create-a-new-react-app.html

- **HTML, CSS, and JavaScript:** Basic knowledge is essential for application structure, styling, and client-side interactivity.
- **Database Connectivity:** Use a MongoDB driver or Mongoose (ODM) to connect the Express.js server with MongoDB for CRUD operations.
- **Front-end Framework:** React.js is utilized for the user-facing part, along with Material UI and Bootstrap for UI enhancement.
- **Version Control:** Git is recommended for collaboration and tracking changes.
    - Download:

https://git-scm.com/downloads

- **Development Environment:** Code editor or IDE (e.g., Visual Studio Code).
    - Visual Studio Code Download:

https://code.visualstudio.com/download

8. Running the Project

To run the existing SB Works application:

1. **Download:** Obtain the project from the Drive link: https://drive.google.com/drive/folders/10mSn2lMTaVMDWWFNjeJjiOLfmcD3-87C?usp=sharing
2. **Install Dependencies:**
    - Navigate into the cloned repository directory:

cd freelancer-app-MERN

    - Install frontend dependencies:

cd client then npm install

    - Install backend dependencies:

cd server then npm install

3. **Start Development Server:** Execute npm start
4. **Access Application:** The SB Works app will be accessible at http://localhost:3000

9. Application Flow (Responsibilities)

- **Freelancer Responsibilities:**
    - Project Submission: Submit completed, high-quality work through the platform.
    - Compliance: Adhere to client requirements and platform guidelines.
    - Effective Communication: Respond to messages, ask clarifying questions, and provide project updates.
    - Time Management: Meet project deadlines.
    - Professionalism: Maintain a respectful and cooperative attitude.
    - Quality Assurance: Deliver accurate, well-executed, and error-free work.
- **Client Responsibilities:**
    - Clear Project Description: Provide detailed project descriptions and requirements.
    - Timely Communication: Respond promptly to freelancer inquiries.
    - Payment Obligations: Fulfill agreed-upon payment terms.
    - Feedback and Evaluation: Provide constructive feedback to freelancers.
- **Admin Responsibilities:**
    - Data Oversight: Monitor data integrity and security.
    - Policy Enforcement: Enforce platform policies and ethical standards.
    - Conflict Resolution: Address disputes promptly and impartially.
    - User Support and Communication: Provide support and guidance to users.
    - Platform Maintenance and Improvement: Oversee overall platform maintenance and improvement.

10. Project Setup and Configuration

- **Folder Setup:** Create client and server folders.
- **Installation of Required Tools:**
    - **Frontend (client folder):** React, Bootstrap, Material UI, Axios, react-bootstrap, react-icons, react-router-dom, react-scripts, socket.io-client, uuid, web-vitals.

- **Backend (server folder):** Express.js, Node.js, MongoDB, Mongoose, Cors, Bcrypt, body-parser, http, socket.io, uuid.

11. Backend Development

1. **Project Setup:** Initialize project with npm, install Express.js, Mongoose, body-parser, and cors.
2. **Database Configuration:** Set up MongoDB (locally or cloud), create collections for Users, Projects, Applications, Chat, and Freelancer.
3. **Express.js Server:** Create an Express.js server to handle HTTP requests and API endpoints, configure body-parser and cors.
4. **API Routes:** Define separate route files for user management, project listing, application handling, chat functionality, and freelancer profiles. Implement route handlers to interact with the database for CRUD operations.
5. **Data Models:** Define Mongoose schemas for User, Project, Application, Chat, and Freelancer. Create Mongoose models and implement CRUD operations.
6. **User Authentication:** Implement JWT or session-based authentication for user registration, login, logout, and protected routes.
7. **Project Management:** Allow clients to post projects and freelancers to browse, search, and submit proposals. Implement client review and freelancer selection.
8. **Secure Communication & Collaboration:** Integrate a secure chat system for client-freelancer communication, enabling file attachments and feedback.
9. **Admin Panel (Optional):** Implement functionalities for managing users, monitoring project updates/applications, and accessing transaction history.

12. Database Development

- **Setup:** Set up a MongoDB database (local or MongoDB Atlas).
- **Collections:** Define collections for users, freelancer, projects, chats, and applications.
- **Connection:** Connect the database to the server using Mongoose.

Database Schemas (Schema.js):

- **userSchema:** username (String, required), email (String, required, unique), password (String, required), usertype (String, required).

- **freelancerSchema:** userId (String), skills (Array, default []), description (String, default ""), currentProjects (Array, default []), completedProjects (Array, default []), applications (Array, default []), funds (Number, default 0).
- **projectSchema:** clientId (String), clientName (String), clientEmail (String), title (String), description (String), budget (Number), skills (Array), bidAmounts (Array), postedDate (String), status (String, default "Available"), freelancerId (String), freelancerName (String), deadline (String), submission (Boolean, default false), submissionAccepted (Boolean, default false), projectLink (String, default ""), manulaLink (String, default ""), submissionDescription (String, default "").
- **applicationSchema:** 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 (String, default "Pending").
- **chatSchema:** _id (String, required), messages (Array).

Mongoose models are exported for each schema (User, Freelancer, Project, Application, Chat).

13. Frontend Development

1. **Setting the Stage:** Create the initial React application structure, install essential libraries, and organize project files.
2. **Crafting the User Experience:** Design reusable UI components (buttons, forms, project cards), define layout and styling, and implement navigation elements.
3. **Bridging the Gap:** Integrate the frontend with SB Works' API endpoints and implement data binding for dynamic updates.

14. Project Implementation & User Interface

After development, the application is run for verification and bug checks.

- **User Interface Snapshots:**
  - Landing page
  - Authentication (Login)
  - Freelancer dashboard (Current projects, Completed projects, Applications, Funds, My Skills, Description)
  - Admin dashboard (All Projects, Completed projects, Applications, Users)

- All projects page (with filters)
- Freelance projects (My projects)
- Applications (My Applications)
- Project page (Project details, Required skills, Budget, Submit the project, Chat with the client)
- New project creation page (Post new project: Project title, Description, Budget, Required skills)