# **SB Works Documentation**

## **1. Introduction**

### **1.1 Overview**

SB Works is a MERN stack-based freelancing platform connecting clients and freelancers. The platform facilitates job postings, proposal submissions, hiring, and payments.

### **1.2 Key Features**

* User Authentication (Clients and Freelancers)
* Job Postings and Search
* Proposal Management
* Messaging System
* Payments and Escrow
* Review and Ratings
* Dashboard Analytics

### **1.3 Tech Stack**

* **Frontend:** React.js
* **Backend:** Node.js, Express.js
* **Database:** MongoDB
* **Authentication:** JWT
* **Payment Gateway:** Stripe/PayPal
* **Hosting:** AWS/Heroku

## **2. Project Setup**

### **2.1 Prerequisites**

* Node.js >= 14
* MongoDB (local or cloud instance)
* Git
* IDE (e.g., VSCode)

### **2.2 Installation**

* **Clone Repository**

git clone <repository-url>  
cd SB-Works

* **Install Dependencies**
* **Backend:**

cd backend  
npm install

* **Frontend:**

cd ../frontend  
npm install

* **Setup Environment Variables** Create .env files in the **backend** and **frontend** directories with the following keys:
* **Backend**:

makefile

Copy code

PORT=5000  
MONGO\_URI=<your-mongo-uri>  
JWT\_SECRET=<your-jwt-secret>  
STRIPE\_SECRET\_KEY=<stripe-key>

* **Frontend**:

arduino

Copy code

REACT\_APP\_API\_URL=http://localhost:5000

* **Run the Application**
* Start Backend:

cd backend  
npm run dev

* Start Frontend:

cd frontend  
npm start

## **3. Application Architecture**

### **3.1 MERN Architecture Overview**

* **MongoDB**: Stores users, jobs, proposals, messages, and payments data.
* **Express.js**: Handles backend API routes.
* **React.js**: Manages UI/UX.
* **Node.js**: Executes server-side logic and connects to the database.

### **3.2 Folder Structure**

#### **Backend**

backend/

│── models/

│ ├── User.js

│ ├── Job.js

│ ├── Proposal.js

│── routes/

│ ├── authRoutes.js

│ ├── jobRoutes.js

│ ├── proposalRoutes.js

│── controllers/

│ ├── authController.js

│ ├── jobController.js

│── middlewares/

│── utils/

│── server.js

Frontend

frontend/

│── src/

│ ├── components/

│ │ ├── Header.jsx

│ │ ├── JobCard.jsx

│ ├── pages/

│ │ ├── Home.jsx

│ │ ├── Dashboard.jsx

│ ├── context/

│ ├── App.js

│── public/

│── package.json

## **4. Features & Functionalities**

### **4.1 User Authentication**

* **Login/Sign-Up**: Email and password-based, JWT authentication.
* **Role-Based Access**: Separate dashboards for clients and freelancers.

### **4.2 Job Management**

* **Clients**:
  + Create, edit, and delete job postings.
  + View proposals for their jobs.
* **Freelancers**:
  + Search and filter jobs by category or budget.
  + Submit proposals to jobs.

### **4.3 Proposal System**

* Freelancers can:
  + Submit detailed proposals.
  + View the status of proposals (Accepted/Rejected).
* Clients can:
  + Accept/reject proposals.
  + Hire a freelancer.

### **4.4 Messaging**

* Real-time chat between clients and freelancers.
* Messages linked to specific jobs.

### **4.5 Payment Gateway**

* Clients can make payments via Stripe/PayPal.
* Escrow functionality to hold funds until job completion.
* Automatic payouts for freelancers after client approval.

### **4.6 Reviews and Ratings**

* Freelancers and clients can review and rate each other post-project.

### **4.7 Dashboard Analytics**

* Clients: Active jobs, completed jobs, and spend summary.
* Freelancers: Proposals sent, jobs won, and earnings summary.

## **5. API Documentation**

### **5.1 Authentication**

#### **POST /api/auth/register**

* **Request:**

json

Copy code

{  
 "name": "John Doe",  
 "email": "[john@example.com](mailto:john@example.com)",  
 "password": "password123",  
 "role": "freelancer"  
}

* **Response:**

json

Copy code

{  
 "success": true,  
 "token": "<jwt-token>"  
}

#### **POST /api/auth/login**

* **Request:**

json

Copy code

{  
 "email": "[john@example.com](mailto:john@example.com)",  
 "password": "password123"  
}

* **Response:**

json

Copy code

{  
 "success": true,  
 "token": "<jwt-token>"  
}

### **5.2 Jobs**

#### **GET /api/jobs**

* Fetch all jobs with filters.

#### **POST /api/jobs**

* Create a new job (Client role only).

## **6. Deployment**

### **6.1 Frontend Deployment**

* Use Vercel or Netlify for React.js deployment.

bash

Copy code

npm run build

### **6.2 Backend Deployment**

* Use Heroku or AWS for Node.js.
* Ensure MongoDB connection uses a cloud service like MongoDB Atlas.

### **6.3 Environment Variables in Production**

Set all sensitive keys in the hosting environment.

## **7. Future Enhancements**

* AI-powered job matching.
* Calendar scheduling for interviews.
* Mobile app development with React Native.