SB Works

Freelancing Platform

SDLC Phase 1

PROJECT PLANNING REPORT

Project Name	SB Works — Online Freelancing Platform
Phase	Phase 1: Project Planning
Prepared By	Project Team — I. Sai Ganesh (Lead Developer)
Date	January 2024
Version	1.0
Status	Completed

1. Introduction

1.1 Purpose of This Report

This document is the official Project Planning Report for the SB Works Freelancing Platform. It was written at the very beginning of the project, before any code was written. The purpose of this report is to clearly define what we are going to build, why we are building it, who will use it, and how we plan to go about the work. Think of this as the blueprint that guided every decision made throughout the project.

1.2 What is SB Works?

SB Works is a web-based freelancing marketplace — a platform where people who need work done (clients) can connect with skilled professionals (freelancers) to get projects completed. The idea came from platforms like Fiverr and Upwork, but SB Works is built entirely from scratch using modern web technologies, giving us full control over every feature.

The name "SB Works" reflects the idea that skillful work gets done here. The platform covers the entire lifecycle of a freelance project: posting the job, submitting proposals, hiring, doing the work, submitting deliverables, reviewing, rating, and real-time communication — all in one place.

> ■ Project Vision: To build a fully functional, production-ready freelancing marketplace using the MERN stack (MongoDB, Express.js, React.js, Node.js) that demonstrates all core features of a real-world web application including authentication, real-time communication, file uploads, and role-based access control.

1.3 Background and Motivation

The freelancing industry has grown rapidly over the past decade. Millions of professionals worldwide offer their services online, and businesses of all sizes hire remote talent. However, existing platforms charge high commission fees (up to 20%) and have complex interfaces. SB Works was conceived as a clean, straightforward alternative that a developer can understand, modify, and deploy independently.

From a learning and demonstration perspective, building SB Works provides hands-on experience with the complete MERN stack, including: database design, REST API development, React frontend architecture, real-time features via Socket.io, JWT-based authentication, and file handling. These are exactly the skills demanded in today's web development job market.

2. Project Scope and Objectives

2.1 Project Objectives

The following objectives were set during the planning phase and used to measure project success at completion:

1. Build a three-role system (Admin, Client/Owner, Freelancer) with proper access control and separate dashboards for each role.

2. Implement a complete project lifecycle — from posting a project to hiring a freelancer, submitting work, reviewing deliverables, and leaving a star rating.

3. Create a real-time chat system so clients and freelancers can communicate directly within the platform without needing external tools.

4. Build an OTP-based phone authentication system — users log in with their phone number and a one-time code, eliminating the need for passwords.

5. Implement file upload functionality so freelancers can attach their completed work files when submitting deliverables.

6. Build an admin panel that allows a platform administrator to manage users, approve accounts, and oversee all activity.

7. Design a clean, responsive user interface using Bootstrap 5 and Material UI that works on desktops and tablets.

8. Write clean, well-structured code with proper separation of concerns (models, controllers, routes, middlewares, services).

2.2 Project Scope

The table below defines what is included in the project (in-scope) and what was deliberately left out to keep the project focused (out-of-scope):

In-Scope (Will Be Built)	Out-of-Scope (Not Included)
OTP phone login and JWT authentication	Payment gateway / actual money transfer
Three role-based dashboards	Mobile native app (iOS/Android)
Project posting and management	Video calls or screen sharing
Proposal submission and bidding	Email notification system
Real-time chat via Socket.io	Social media login (Google, GitHub)
File upload for work submissions	Search engine optimization (SEO)
Star ratings and written reviews	Multi-currency support
Admin user management panel	Subscription or premium plans
Real-time notifications	Dispute resolution system

10 project categories via seed data	Public API for third-party apps

3. Stakeholders and Team

3.1 Project Stakeholders

Stakeholders are everyone who has an interest in the project — either because they contributed to it, or because they will use or be affected by it.

Stakeholder	Role	Interest / Involvement
I. Sai Ganesh	Lead Developer / Project Owner	Designed, built, and delivered the entire system
Admin Users	Platform Administrator	Manage users and oversee platform health
Client (Owner) Users	End User — Client	Post projects, hire freelancers, review work
Freelancer Users	End User — Freelancer	Find work, submit proposals, deliver projects
Academic Evaluators	Assessors	Evaluate the project for academic or professional purposes

3.2 Project Team

This project was developed as a solo full-stack development project. All roles were handled by one developer:

Role	Responsibility	Technology Area
Full Stack Developer	Backend API, database design, server logic	Node.js, Express, MongoDB
Frontend Developer	React components, UI design, routing	React, Bootstrap, Material UI
System Architect	Database schema, API design, Socket.io	Architecture & Design
QA Tester	Manual testing of all features and flows	Testing & Validation
DevOps	Environment setup, deployment config	Configuration & Setup

3.3 Project Timeline Overview

The project was planned across 7 SDLC phases. The timeline below shows the estimated duration of each phase:

Phase	Activity	Duration	Status
Phase 1	Project Planning	3 days	Completed
Phase 2	Requirements Analysis	4 days	Completed
Phase 3	System Design	5 days	Completed
Phase 4	Implementation (Coding)	21 days	Completed
Phase 5	Testing & Quality Assurance	5 days	Completed
Phase 6	Deployment & Configuration	2 days	Completed

Phase 7	Maintenance & Documentation	Ongoing	Active

4. Risk Assessment and Resources

4.1 Risk Identification and Mitigation

Every project has risks — things that could go wrong and delay or derail the work. We identified these risks during planning and created mitigation strategies for each:

Risk	Likelihood	Impact	Mitigation Strategy
Real-time chat implementation complexity	Medium	High	Use Socket.io library — well documented with clear examples
JWT authentication bugs causing security issues	Low	High	Follow industry best practices, use signed HttpOnly cookies
MongoDB schema design mistakes	Medium	Medium	Plan schema carefully, use Mongoose validation
File upload size and type issues	Medium	Low	Use Multer with file size and type restrictions
Windows development environment issues	High	Medium	Document all setup steps, test on Windows explicitly
Scope creep (adding too many features)	High	Medium	Stick to defined in-scope items, defer extras to Phase 7
Third-party SMS API (Twilio) costs	Medium	Low	Build dev mode that prints OTP to terminal instead of SMS
Database connection failures in production	Low	High	Use MongoDB Atlas with automatic failover

4.2 Resources Required

Resource Type	Specific Resource	Purpose
Hardware	Development PC (Windows 10)	Coding, testing, and running the application
Software	Node.js 18+, npm	Backend runtime and package manager
Software	VS Code Editor	Code writing and debugging
Software	MongoDB Compass / Atlas	Database management and viewing
Software	Postman	API testing during development
Software	Git / GitHub	Version control and code backup
Cloud	MongoDB Atlas (Free Tier)	Cloud database hosting
Cloud	Twilio (Optional)	SMS delivery for OTP (production only)
Library	React, Express, Mongoose, Socket.io	Core development frameworks

■ Planning Outcome: By the end of Phase 1, we had a clear understanding of what to build, who would use it, what risks to watch out for, and a realistic timeline. This planning document served as the north star for all subsequent phases of development.

5. Technology Decisions and Phase Summary

5.1 Why MERN Stack?

During planning, a key decision was which technology stack to use. We evaluated several options and chose the MERN stack for the following reasons:

Technology	Why Chosen	Alternative Considered
MongoDB	Flexible schema for varying project/proposal structures	PostgreSQL, MySQL
Express.js	Minimal, fast Node.js web framework with middleware support	Fastify, Hapi.js
React.js	Component-based UI, huge ecosystem, single-page app	Vue.js, Angular
Node.js	JavaScript everywhere — same language front and back	Python (Django), PHP
Socket.io	Easiest real-time WebSocket library, handles fallbacks	WebSocket raw, Pusher
Bootstrap 5	Rapid responsive UI with ready-made components	Tailwind CSS, Bulma
MongoDB Atlas	Free cloud DB, no local install required	Local MongoDB, PlanetScale
JWT + Cookies	Stateless auth, secure HttpOnly cookies vs localStorage	Sessions, OAuth

5.2 Phase 1 Summary and Sign-Off

Phase 1 — Project Planning — was completed successfully. All objectives were defined, the technology stack was chosen, risks were identified with mitigation plans, and a realistic timeline was established. The project team was ready to move into Phase 2: Requirements Analysis.

> ■ Phase 1 Deliverable: This Project Planning Report, which serves as the formal documentation of all decisions made during the planning phase. It was reviewed and approved before Phase 2 commenced.

5.3 Lessons Noted During Planning

- Breaking down the project into SDLC phases from the start helped avoid confusion and kept work organized.
- Writing down scope clearly (in-scope vs out-of-scope) prevented feature creep during development.
- Identifying the Twilio SMS cost risk early led to the smart dev-mode OTP solution that saved time and money.
- Planning the three-role architecture early made database and API design much cleaner in later phases.
- Estimating more time than needed for each phase is always better than running out of time.