# SB Works

Freelancing Platform

SDLC Phase 2

# REQUIREMENTS ANALYSIS REPORT

| | |
|---|---|
| **Project Name** | SB Works — Online Freelancing Platform |
| **Phase** | Phase 2: Requirements Analysis |
| **Prepared By** | I. Sai Ganesh — Lead Developer |
| **Date** | January 2024 |
| **Version** | 1.0 |

# 1. Introduction to Requirements Analysis

## 1.1 What is Requirements Analysis?

Requirements Analysis is the second phase of the Software Development Life Cycle. After planning what we want to build in Phase 1, we now dig deeper into the specifics of what the software must actually do. Think of it like an architect taking a vague idea — "I want a house" — and turning it into a detailed blueprint that specifies every room, door, window, and wall.

In this phase, we ask the question: "What exactly should the system do for each type of user?" We gather these requirements from the perspective of each user role — Admin, Client (Owner), and Freelancer — and document them in a structured way so they can be handed to the design and development phases.

> ■ Requirements are divided into two types: Functional Requirements (what the system does — features and capabilities) and Non-Functional Requirements (how the system behaves — performance, security, usability). Both types are documented in this report.

## 1.2 How Requirements Were Gathered

Since this is a solo development project, requirements were gathered through the following methods:

- Competitive Analysis: Studying existing freelancing platforms (Fiverr, Upwork, Freelancer.com) to understand what features users expect.
- User Journey Mapping: Walking through the experience of each type of user from first login to project completion.
- Pain Point Identification: Noting what existing platforms do poorly and designing SB Works to do it better.
- Technical Feasibility Review: Checking that each requirement can realistically be built with the MERN stack within the project timeline.
- Stakeholder Self-Review: As the developer and primary user, reviewing requirements from all three user perspectives.

# 2. Functional Requirements

## 2.1 Admin User Requirements

| ID | Requirement | Priority | Implemented |
|---|---|---|---|
| FR-A01 | Admin can log in using phone number and OTP | High | Yes |
| FR-A02 | Admin can view a list of all registered users | High | Yes |
| FR-A03 | Admin can search users by name, email, or phone | Medium | Yes |
| FR-A04 | Admin can approve, reject, or pend any user account | High | Yes |
| FR-A05 | Admin can view all projects posted on the platform | Medium | Yes |
| FR-A06 | Admin can view all proposals across all projects | Medium | Yes |
| FR-A07 | Admin can add, edit, and delete project categories | High | Yes |
| FR-A08 | Admin dashboard shows platform-wide statistics | Medium | Yes |
| FR-A09 | Admin account is created automatically by seed script | High | Yes |

## 2.2 Client (Owner) User Requirements

| ID | Requirement | Priority | Implemented |
|---|---|---|---|
| FR-C01 | Client can register with phone number, set name, email, choose Owner role | High | Yes |
| FR-C02 | Client can create a new project with title, description, category, budget, deadline, tags | High | Yes |
| FR-C03 | Client can view all their posted projects in a table | High | Yes |
| FR-C04 | Client can edit or delete any of their projects (if no freelancer assigned) | Medium | Yes |
| FR-C05 | Client can toggle a project between OPEN and CLOSED status | Medium | Yes |
| FR-C06 | Client can view all proposals submitted to their projects | High | Yes |
| FR-C07 | Client can accept one proposal to hire a freelancer | High | Yes |
| FR-C08 | Client can reject unwanted proposals with one click | High | Yes |
| FR-C09 | Client can view submitted work files and descriptions | High | Yes |
| FR-C10 | Client can approve completed work or request revisions with written feedback | High | Yes |

| | | | |
|---|---|---|---|
| FR-C11 | Client can leave a 1-5 star rating and written review after work is approved | High | Yes |
| FR-C12 | Client can chat in real-time with freelancers via the built-in chat system | High | Yes |
| FR-C13 | Client receives real-time notifications for new proposals and work submissions | Medium | Yes |

# 2. Functional Requirements (Continued)

## 2.3 Freelancer User Requirements

| ID | Requirement | Priority | Implemented |
|---|---|---|---|
| FR-F01 | Freelancer can register with phone number, set name, email, choose Freelancer role | High | Yes |
| FR-F02 | Freelancer can browse all open projects in a grid with search functionality | High | Yes |
| FR-F03 | Freelancer can submit a proposal with bid amount, delivery time, and cover letter | High | Yes |
| FR-F04 | System prevents submitting more than one proposal per project per user | High | Yes |
| FR-F05 | Freelancer can track all their proposals and see current status (Pending/Accepted/Rejected) | High | Yes |
| FR-F06 | Freelancer can view their active projects (accepted proposals) | High | Yes |
| FR-F07 | Freelancer can submit completed work with a description and optional file attachment | High | Yes |
| FR-F08 | Freelancer can resubmit work if the client requests revisions | Medium | Yes |
| FR-F09 | Freelancer can see client feedback when revision is requested | Medium | Yes |
| FR-F10 | Freelancer can chat in real-time with clients via built-in chat system | High | Yes |
| FR-F11 | Freelancer receives real-time notifications for proposal decisions and work approvals | Medium | Yes |
| FR-F12 | Freelancer profile shows average rating and total review count | Medium | Yes |

## 2.4 Authentication System Requirements

| ID | Requirement | Description |
|---|---|---|
| FR-AU01 | OTP Generation | System generates a 6-digit OTP when user enters phone number |
| FR-AU02 | OTP Delivery (Dev) | In development, OTP is printed to the backend server console |
| FR-AU03 | OTP Delivery (Prod) | In production, OTP is sent via Twilio SMS API |
| FR-AU04 | OTP Expiry | OTP expires after 90 seconds — user must request a new one |
| FR-AU05 | JWT Cookies | Login sets signed HttpOnly access token (1 day) and refresh token (1 year) |

| | | |
|---|---|---|
| FR-AU06 | Auto Refresh | Access token is automatically refreshed before expiry |
| FR-AU07 | Role Redirect | After login, user is redirected to their role-specific dashboard |
| FR-AU08 | Admin Seeding | Admin account with permanent OTP (123456) is created by seed script |

# 3. Non-Functional Requirements

Non-functional requirements define how the system behaves — its quality attributes, performance standards, and security expectations. These are just as important as functional requirements because a system can have all the right features but still fail if it is slow, insecure, or difficult to use.

| Category | Requirement | Acceptance Criterion |
|---|---|---|
| Performance | API response time for standard requests | Under 500ms for most endpoints |
| Performance | Real-time chat message delivery | Under 100ms via Socket.io WebSocket |
| Security | Authentication tokens | JWT stored in HttpOnly signed cookies (not localStorage) |
| Security | Password-free login | No passwords stored — OTP-based authentication only |
| Security | Route protection | All protected routes check token validity and user role |
| Security | File uploads | Only specific file types accepted, size limits enforced |
| Usability | Responsive design | Works correctly on desktop screens (1024px and above) |
| Usability | Error messages | All errors shown as clear toast notifications (React Hot Toast) |
| Usability | Loading states | All API calls show loading indicator while fetching |
| Reliability | Database connection | App shows clear error if MongoDB is not reachable |
| Reliability | Graceful Twilio fallback | If Twilio fails, OTP is logged to console in dev mode |
| Maintainability | Code structure | MVC pattern — models, controllers, routes clearly separated |
| Scalability | Database | MongoDB Atlas with cloud hosting for production scalability |
| Portability | Windows support | Application runs on Windows 10/11 without modification |

# 4. Use Case Summary and Phase Conclusion

## 4.1 Primary Use Cases

A use case describes a specific interaction between a user and the system. The main use cases for SB Works are:

| Use Case | Actor | Trigger | Outcome |
|---|---|---|---|
| UC-01: Login with OTP | Any User | User enters phone number | User receives OTP, logs in, redirected to dashboard |
| UC-02: Post a Project | Client | Client clicks "Post Project" | Project created in OPEN status, visible to freelancers |
| UC-03: Browse Projects | Freelancer | Freelancer visits Browse page | Sees all open projects, can search by keyword |
| UC-04: Submit Proposal | Freelancer | Freelancer clicks "Submit Proposal" | Proposal saved, client notified in real-time |
| UC-05: Hire Freelancer | Client | Client clicks "Accept" on a proposal | Freelancer hired, other proposals auto-rejected |
| UC-06: Submit Work | Freelancer | Freelancer clicks "Submit Work" | Work + file uploaded, client notified |
| UC-07: Approve Work | Client | Client clicks "Approve Work" | Project marked complete, freelancer notified |
| UC-08: Leave Review | Client | Client submits star rating + comment | Review saved, freelancer rating updated |
| UC-09: Real-time Chat | Any User | User opens Chat page | Messages sent and received instantly via Socket.io |
| UC-10: Manage Users | Admin | Admin visits Users page | Can approve, reject, or pend any user account |

## 4.2 Phase 2 Summary

Phase 2 — Requirements Analysis — produced a comprehensive list of functional and non-functional requirements that served as the contract between planning and development. With 39 functional requirements documented across all three user roles and 14 non-functional requirements covering performance, security, usability, and maintainability, the development team had a clear, unambiguous specification to build against.

> ■ Phase 2 Outcome: All requirements were reviewed, prioritized (High/Medium/Low), and confirmed feasible within the MERN stack. The requirements document was signed off before moving to Phase 3: System Design. All 39 functional requirements were successfully implemented by the end of the project.