Below is the extracted text content from the provided images of your Software Requirements Specification (SRS) for the Freelancer Marketplace project:

Software Requirements Specification (SRS)

Freelancer Marketplace

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a comprehensive, unambiguous, and meticulously detailed definition of all functional, non-functional, security, and technical requirements for the Freelancer Marketplace. This document serves as the official blueprint for all project stakeholders—including business analysts, project managers, solution architects, developers, quality assurance engineers, and the end-users (freelancers, clients, and administrators). It aims to clearly articulate the system's capabilities, user interactions, and technical specifications to ensure a shared understanding of the project's goals. The primary objective is to create a modern, scalable, and secure web application that facilitates connections between clients seeking services and freelancers offering their skills, streamlining the entire project lifecycle from project posting to completion and management. This SRS will serve as the single source of truth, guiding development, facilitating rigorous testing, and ensuring the final product perfectly aligns with the business objectives of both freelancers and clients.

1.2 Scope

The Freelancer Marketplace is a sophisticated, two-tier web application designed to connect clients and freelancers. Its scope is rigorously defined to provide a clear boundary for the project and its core business functionalities.

In-Scope:

- **User Management:** A highly secure, token-based authentication system for both freelancers and clients to register, log in, and manage their profiles. This includes account verification and password reset functionality.
- **Profile Management:** Full CRUD (Create, Read, Update, Delete) capabilities for user profiles. This includes managing professional history, skills, portfolio, and rates for freelancers, and company details for clients.
- **Project Management (Client):** A complete suite of tools for clients to create new projects, post job requirements, and manage project details.
- **Freelancer Search and Discovery:** A powerful search engine that allows clients to search for and browse freelancer profiles based on skills, rates, and other criteria.
- **Project Bidding and Assignment:** A formal mechanism for freelancers to apply to projects with a proposal and for clients to review applications and assign a freelancer to a project.
- **Project Status Tracking:** Functionality for both clients and freelancers to view and update the status of an assigned project throughout its lifecycle.

- **In-app Messaging:** A simple messaging feature that allows clients and freelancers to communicate about projects after an application has been submitted or a project has been assigned.
- **Admin Dashboard:** A secure administrative portal for managing users and projects.
- **Notifications System:** A system for sending automated notifications to users for key events (e.g., new project applications, status changes).
- **RESTful API:** A meticulously designed, versioned, and secure backend API built with Spring Boot to handle all business logic, data persistence, and user authentication.
- **Responsive User Interface:** A modern, highly-responsive Single-Page Application (SPA) developed with React.js that provides an intuitive user experience on all devices.

Out-of-Scope:

- **Payment Processing:** The system will not handle any financial transactions, invoicing, or payment processing between clients and freelancers. This is assumed to be handled outside the platform.
- **Dispute Resolution:** The system will not provide any tools or features for mediating disputes between clients and freelancers.
- **Rating and Review System:** The system will not include a public rating and review system for freelancers or clients.
- **Advanced Communication:** The system will not feature video conferencing or file sharing within messages.
- **Multi-Admin Roles:** The system is designed for a single administrative role. Support for multiple administrative roles with different permissions is out of scope.

1.3 Definitions, Acronyms, and Abbreviations

- **API:** Application Programming Interface.
- **CRUD:** Create, Read, Update, Delete. The four fundamental operations.
- **DTO:** Data Transfer Object.
- **HTTPS:** Hypertext Transfer Protocol Secure.
- **JWT:** JSON Web Token. A stateless authentication mechanism.
- **REST:** Representational State Transfer.
- **SPA:** Single-Page Application.
- **UI/UX:** User Interface / User Experience.
- **RBAC:** Role-Based Access Control.
- **JPA:** Java Persistence API.
- **BCrypt:** A password-hashing function.
- **WCAG:** Web Content Accessibility Guidelines.

1.4 References

- IEEE Std 830-19981
- OWASP Top 10 - 20212
- The Spring Framework Documentation3
- React.js Official Documentation4
- RFC 7519: The standard defining JWTs.5
- WCAG 2.1 AA: Standard guidelines for web accessibility.6

1.5 Overview

This SRS is a highly structured and detailed document.

- **Section 1 (Introduction):** Provides the foundational context, project scope, and key definitions.
- **Section 2 (Overall Description):** Presents a high-level architectural view, outlines the system's major functions, defines the user classes, and specifies the technical and operational environment.
- **Section 3 (Specific Requirements):** The core of the document, containing an exhaustive breakdown of all functional, non-functional, and security requirements.
- **Section 4 (Appendices):** Contains supplementary materials, including detailed data models (ERD), a complete use case model, and a high-level system architecture diagram.

---

2. Overall Description

2.1 Product Perspective

The Freelancer Marketplace is a standalone, two-tier application following a client-server architecture.

- **Front-end Tier (Client):** A dynamic, responsive, and cross-browser compatible SPA built on React.js. This layer is responsible for all user-facing interactions, data presentation, and managing application state. It communicates with the backend exclusively via a secure REST API.
- **Back-end Tier (Server):** A robust, stateless application built with Spring Boot. It encapsulates all business logic, data validation, security, and persistence logic. The back-end is designed to be highly scalable and resilient.
- **Data Tier (Database):** A relational database management system, such as MySQL or PostgreSQL, that provides persistent storage. The back-end is the only component authorized to interact with this tier, ensuring data integrity.

2.2 Product Functions

The system's core functions are logically grouped into the following modules:

- **Authentication Module:** Handles user login, token generation, and authorization.

- **Profile Management Module:** Provides tools for both user types to create and update their profiles.
- **Project Management Module:** Enables clients to create, view, and manage their projects.
- **Search and Discovery Module:** Provides a search engine for clients to find suitable freelancers.
- **Application and Assignment Module:** Facilitates the process of freelancers applying to projects and clients assigning them.
- **Project Status Tracking Module:** Allows both parties to monitor the progress of projects.
- **Messaging Module:** Facilitates communication between clients and freelancers.
- **Admin Module:** Manages users and projects.
- **Notification Module:** Sends automated alerts for key events.

2.3 User Classes and Characteristics

- **Client:**
    - Permissions: Can register, log in, manage their profile, create and post new projects, browse freelancers, and assign projects.
    - Characteristics: Individual or business seeking to hire a freelancer for a project, assumed to be familiar with project management concepts.
- **Freelancer:**
    - Permissions: Can register, log in, manage their professional profile, browse projects, apply to projects, and update project status.
    - Characteristics: Individual with a specific skill set (e.g., web development, graphic design) seeking project-based work.
- **Admin:**
    - Permissions: Privileged access to manage user and project data, deactivate user accounts, and resolve issues.
    - Characteristics: Designated staff member responsible for platform integrity.

2.4 Operating Environment

The system is a distributed application with specific technology and deployment requirements.

2.5 Design and Implementation Constraints

- **Security:** System must adhere to OWASP Top 10 guidelines; protected endpoints use JWT; all passwords hashed with BCrypt (min work factor 12); all data in transit uses HTTPS.
- **Technology Stack:** Technologies are non-negotiable (React.js, Spring Boot, MySQL/PostgreSQL).
- **API Versioning:** API versioned (e.g., /api/v1/).

- **Responsiveness:** UI must be responsive, WCAG 2.1 AA compliant.
- **Data Integrity:** All state-changing database operations must be transactional.

Frontend Environment:

- Framework: React.js (v18.x+)
- Languages: TypeScript, HTML5, CSS3
- Styling: Tailwind CSS
- Browser Support: Latest stable Chrome, Firefox, Safari, Edge

Backend Environment:

- Framework: Spring Boot (v3.x+)
- Languages: Java 17+
- Persistence: Spring Data JPA with Hibernate
- Build Tool: Maven

Database Environment:

- Type: Relational (MySQL 8.x+ or PostgreSQL)
- Deployment: Docker containers on cloud provider

---

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces

UI is the primary point of interaction for all users; design is clean, intuitive, and consistent.

- **Login/Registration Page:** Secure page for authentication and new account creation ("Freelancer" or "Client" role).
- **Dashboard:** Personalized landing page.
- **Freelancer Profile Page:** For freelancers to create/edit profiles (skills, portfolio, rates).
- **Project Creation Form:** Multi-step form for posting new projects.
- **Project Listing Page:** Paginated table of projects.
- **Freelancer Search Page:** Clients can search/browse freelancer profiles with filters.
- **Project Detail Page:** Dedicated page for single project details.
- **Project Status Tracking Page:** For both parties to update/track assigned project status.
- **Messaging Interface:** Direct messaging between clients/freelancers.
- **Admin Dashboard:** Admin portal for management.

3.2 Functional Requirements (Detailed)

FR1: User Registration and Authentication

- **Description:** Secure, straightforward process for new users to register and log in.
- **Sub-Requirements:**
    - FR1.1: Registration allows choosing "Freelancer" or "Client" role.
    - FR1.2: Registration form captures name, email, password, user role.
    - FR1.3: Passwords securely stored using BCrypt (work factor ≥12).
    - FR1.4: Login endpoint authenticates user and issues JWT on success.

FR2: Freelancer Profile Management

- **Description:** Freelancers can create/update professional profiles.
- **Sub-Requirements:**
    - FR2.1: Profile must include skills, portfolio, hourly rate, professional bio.
    - FR2.2: Freelancers can add multiple skills/portfolio items.
    - FR2.3: System validates all data before saving.

FR3: Project Management (Client)

- **Description:** Clients can create, edit, manage projects.
- **Sub-Requirements:**
    - FR3.1: Project creation form includes title, description, budget, skills.
    - FR3.2: System displays list of projects for each client.
    - FR3.3: Clients can edit/delete projects in "Open" or "Draft" status.

FR4: Freelancer Search and Discovery

- **Description:** Clients can find freelance talent.
- **Sub-Requirements:**
    - FR4.1: Search page allows querying freelancers by skills, hourly rate, location.
    - FR4.2: Results must show matching freelancers with profile summary.
    - FR4.3: Search must be paginated and sortable by relevance/rate.

FR5: Project Application and Assignment

- **Description:** Freelancers apply to projects; clients assign them.
- **Sub-Requirements:**
    - FR5.1: Freelancer can view and apply to open projects with a proposal.
    - FR5.2: Client can view all applications for their projects.

- FR5.3: Client can assign project to a freelancer, locking the project.

FR6: Project Status Tracking

- **Description:** Track assigned project status.
- **Sub-Requirements:**
  - FR6.1: Project must have status lifecycle: "Open," "Assigned," "In Progress," "Completed," "Canceled."
  - FR6.2: Freelancer can change status from "Assigned" to "In Progress."

Below is the full Software Requirements Specification (SRS) for your Freelancer Marketplace, combining all the extracted and structured content from your uploaded images. You requested it as a PDF, but my current environment cannot generate PDFs directly. Instead, here is the complete SRS in text format for you to easily copy and convert to PDF using any text editor or online tool.

Software Requirements Specification (SRS): Freelancer Marketplace

---

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to provide a comprehensive, unambiguous, and detailed definition of all functional, non-functional, security, and technical requirements for the Freelancer Marketplace. It is the blueprint for all stakeholders—business analysts, project managers, architects, developers, quality assurance, and end-users (freelancers, clients, admins). Its aim is to create a scalable, secure web application streamlining connections between clients and freelancers for project posting, completion, and management.

1.2 Scope

In-Scope:

- User Management (secure authentication, registration, account verification, password reset)
- Profile Management (CRUD for users: skills, history, portfolio, rates, company details)
- Project Management (clients create, post, and manage projects)
- Freelancer Search and Discovery (advanced search for clients)
- Project Bidding and Assignment (formal application and assignment)
- Project Status Tracking (lifecycle management for clients/freelancers)
- In-app Messaging (project-based chat, after application/assignment)
- Admin Dashboard (user/project management)
- Notifications System (automated alerts for key events)

- RESTful API (Spring Boot)
- Responsive UI (React.js SPA)

Out-of-Scope:

- Payment Processing
- Dispute Resolution
- Rating and Review System
- Advanced Communication (video conferencing, file sharing)
- Multi-Admin Roles (only a single admin role allowed)

1.3 Definitions, Acronyms, Abbreviations

API – Application Programming Interface
CRUD – Create, Read, Update, Delete
DTO – Data Transfer Object
HTTPS – Hypertext Transfer Protocol Secure
JWT – JSON Web Token
REST – Representational State Transfer
SPA – Single-Page Application
UI/UX – User Interface/User Experience
RBAC – Role-Based Access Control
JPA – Java Persistence API
BCrypt – Password hashing
WCAG – Web Content Accessibility Guidelines

1.4 References

- IEEE Std 830-1998: Software Requirements
- OWASP Top 10 - 2021: Web application security
- The Spring Framework Documentation
- React.js Official Documentation
- RFC 7519: JWT standard
- WCAG 2.1 AA: Accessibility guidelines

1.5 Overview

Sections include:

- Introduction (context and scope)
- Overall Description (architecture, user classes, environment)
- Specific Requirements (detailed functions and architecture)
- Appendices (data model, use cases, diagrams)

---

2. Overall Description

2.1 Product Perspective

A two-tier standalone web app with client-server architecture:

- **Front-end (Client):** React.js SPA for user interaction, data display, states (REST API communication).
- **Back-end (Server):** Spring Boot, stateless, all business logic, validation, security, scalable and resilient.
- **Data Tier (Database):** MySQL or PostgreSQL (backend sole access for data integrity).

2.2 Product Functions

Modules:

- Authentication (login, tokens, authorization)
- Profile Management (update profiles)
- Project Management (clients)
- Search and Discovery (client finds freelancers)
- Application and Assignment
- Status Tracking
- Messaging (client ↔ freelancer)
- Admin management
- Notification alerts

2.3 User Classes

- **Client:** Manages profile, posts projects, browses and assigns freelancers.
- **Freelancer:** Manages profile, applies for projects, updates status.
- **Admin:** Privileged access, manages users/projects, deactivates accounts (no access to messages).

2.4 Operating Environment

Distributed app, specific tech stack, and deployment.

2.5 Design and Implementation Constraints

- Security: Strict OWASP Top 10, JWT, BCrypt (factor ≥12), HTTPS.
- Stack: React.js, Spring Boot, MySQL/PostgreSQL.
- API Versioning: /api/v1/
- Responsiveness: WCAG 2.1 AA
- Data Integrity: Transactions on state changes.

Frontend: React.js (18+), TypeScript, HTML5, CSS3, Tailwind CSS, Chrome/Firefox/Safari/Edge support.

Backend: Spring Boot (3+), Java 17+, Spring Data JPA/Hibernate, Maven.

Database: MySQL 8+/PostgreSQL, Docker containers.

---

3. Specific Requirements

3.1 External Interfaces

3.1.1 User Interfaces

Clean, consistent, and intuitive UI:

- Login/Registration: Secure, role choice (Freelancer/Client)
- Dashboard: Personalized post-login info
- Freelancer Profile Page: Edit/view profile, skills, rate
- Project Creation: Multi-step form
- Project Listing: Paginated, searchable table
- Freelancer Search: Clients filter freelancers
- Project Detail: Single project details
- Status Tracking: Update/view status
- Messaging: Project-based chat
- Admin Dashboard: Manage users/projects

3.2 Functional Requirements

FR1: User Registration and Authentication

- Role selection (Freelancer/Client)
- Capture details (name, email, password, role)
- BCrypt hashing and JWT issuance on login

FR2: Freelancer Profile Management

- Fields: skills, portfolio, hourly rate, bio
- Multiple skills/portfolio items, all data validated

FR3: Project Management (Client)

- Title, description, budget, skills
- Lists client's projects; edit/delete if Open/Draft

FR4: Freelancer Search and Discovery

- Query by skills, rate, location
- Paginated/sortable results, profile summaries

FR5: Project Application and Assignment

- Freelancers view/apply
- Clients see applications
- Assignment locks project

FR6: Project Status Tracking

- Status lifecycle: Open, Assigned, In Progress, Completed, Canceled
- Freelancer can change Assigned → In Progress
- Client changes In Progress → Completed/Canceled
- Both view status on dedicated page

FR7: In-app Messaging

- Private messages after application/assignment
- Simple chat per project, message history
- No deletion or attachments

FR8: User Dashboard with Notifications

- Freelancer: Active projects, sent applications
- Client: Open projects, new applications
- Both: Notification feed for events

FR9: User and Project Search (Admin)

- Search users by name/email
- View all projects/status/user history

FR10: Account Deactivation (Admin)

- Button on admin dashboard
- Deactivation prevents login, hides profile
- Audit log of actions

FR11: Project Application Management

- List freelancers applied
- View applicant profiles
- Accept/reject
- Message applicants

FR12: Project Archiving

- Manually archive Completed/Canceled
- Archived hidden, shown only in Archived Projects

FR13: Profile Photo Upload

- Upload component, validate type/size, resize/store/delete photo

FR14: Password Reset Functionality

- Forgot password link, email reset, single-use timed token

FR15: Project Filter and Search (Freelancer)

- Filters: budget, skills, open projects
- Search by title/description
- Paginated/sortable

FR16: Notifications System

- Alerts to:
    - Client: new application, status changes
    - Freelancer: acceptance/rejection, status changes
- Dashboard and email feed

FR17: Admin Dashboard and Reporting

- User/project counts by role/status
- User activity reports

FR18: Auditing and Logging

- Log all admin deactivations/status changes (ID, action, timestamp)

3.3 Non-Functional Requirements

3.3.1 Security

- JWT authentication, Spring Security RBAC
- TLS/SSL in transit, BCrypt at rest
- Server-side validation for all APIs
- Regular vulnerability scans

3.3.2 Performance

- API response under 1 second for 100 concurrent users
- Horizontal scalability, Redis caching

3.3.3 Reliability

- 99.9% uptime
- Graceful error handling (JSON responses)
- DB transactions for state changes

3.3.4 Maintainability

- Modular codebase, standard directory structure
- 85% backend test coverage
- Dockerized deployment

---

4. Appendices

Appendix A: Data Model (ERD)

users Table: id, email (unique), password_hash, role (ENUM), is_active, created_at, updated_at
freelancer_profiles Table: id, user_id (FK, unique), first_name, last_name, bio, hourly_rate, location, skills (JSON), portfolio (JSON), profile_picture_url
client_profiles Table: id, user_id (FK, unique), company_name
projects Table: id, client_id (FK), title, description, budget, required_skills (JSON), status (ENUM), assigned_freelancer_id (FK), created_at, is_archived
applications Table: id, project_id (FK), freelancer_id (FK), proposal_message, created_at
messages Table: id, sender_id (FK), project_id (FK), content, created_at
notifications Table: id, user_id (FK), message, is_read, created_at
audit_logs Table: id, user_id (FK), action, entity_type, entity_id, timestamp

Appendix B: Use Case Model

C.1 Client Posts a Project

- Actor: Client
- Goal: Create and post project
- Preconditions: Logged in
- Flow: Create Project → Fill form → Validate → Create DB record → Success message
- Postcondition: New project saved

C.2 Freelancer Applies to a Project

- Actor: Freelancer
- Goal: Apply to posted project
- Preconditions: Logged in, project is Open
- Flow: Navigate listing → Apply → Submit proposal → Create Application → Notify client
- Postcondition: New application saved, client notified.