Software Requirements Specification (SRS)

Freelancing Website

Name	**Email**
TAIEB-EZZRAIMI Cherif	cherif.taiebezzraimi@ensia.edu.dz
HEZAM AYA EL ATRA	aya.elatra.hezam@ensia.edu.dz
SAIGHI DOUNIA	dounia.saighi@ensia.edu.dz
BRAHMI IDRIS AHMED	ahmed.idris.brahmi@ensia.edu.dz
MIR RANIA	rania.mir@ensia.edu.dz

3rd Year ENSIA Students

Group 9, Team 3

October 17, 2025

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) describes the functional and non-functional requirements for a web-based **Freelancing Website** targeted primarily at the Algerian market and neighboring regions. The platform connects clients (individuals, startups, companies) with freelancers (students, independent professionals) to collaborate on digital projects, internships, and short-term contracts in a secure and efficient way.

This SRS is intended for the development team, UX designers, QA/testers, university supervisors, and project stakeholders and will be used as the authoritative reference during design, implementation, testing and deployment.

1.2 Document Conventions

This document follows the IEEE 830-style structure with sections for introduction, overall description, specific system features, external interfaces, non-functional requirements, appendices, and traceability. Requirement IDs use the prefix `FR-` for functional requirements and `NFR-` for non-functional requirements. Goals use `G-` and test cases use `TC-`.

1.3 Intended Audience and Reading Suggestions

Intended readers:

- **Developers** — read sections: Overall Description, System Features, External Interfaces, Database Requirements.
- **Designers / UX** — read: Introduction, Product Features, User Interfaces, Product Perspective.
- **Testers** — read: Functional Requirements, Traceability Matrix, Test Cases (to be produced).
- **Project Managers / Stakeholders** — read: Executive summary (Introduction), Goals and Objectives, Risk Analysis, Deployment.

Suggested reading order: Introduction → Overall Description → System Features → External Non-functional Requirements → Appendices.

1.4 Project Scope

The system is a multi-tenant web application providing:

- User account system with role-based access (Client, Freelancer, Company, Admin).
- Project posting, bidding/proposals, and direct-hire flows.
- Secure messaging and file exchange between parties.
- Payment flows : manual bank / Baridi-mob or CCP receipt verification.
- Profiles, portfolios, reviews, and verification badges.
- Admin tools for dispute resolution, analytics, and moderation.

1.4.1 Problem Definition

Local freelancing platforms in Algeria and the region typically suffer from:

- Lack of secure payment and escrow mechanisms causing trust issues.
- Poor UI/UX and slow performance causing high bounce rates.
- Limited market awareness and user acquisition, causing low liquidity.
- Weak integration with universities and real-world opportunities for students.
- Unsustainable business models (e.g., 0% commission without alternative revenue).

1.4.2 Solution Overview

This platform will address the above by:

- Delivering modern, web-first UI/UX with strong performance targets.
- Partnering with universities to onboard students and run ambassador programs.
- Introducing gamification, badges, and verified profiles to increase trust.
- Offering initial low/no commission incentives for early adopters and well-defined subscription/credits monetization plans for sustainability.

1.5 References

- IEEE 830 Standard for Software Requirements Specification.
- Competitor research: Upwork, Fiverr, Freehali, DZFreelance, SoukWork, Jobbers.
- UX research papers on marketplace liquidity and user retention.

2 Overall Description

2.1 Product Perspective

The system is designed as a web application (single-page application) with a REST/GraphQL API backend and optional WebSocket service for real-time features (chat, notifications). It can be deployed as a single monolith for MVP and later split into microservices for scale.

2.2 Product Features (high-level)

1. **User Authentication & Roles:** Sign up, sign in, OAuth (Google), email verification, password reset.
2. **Freelancer Profiles:** Skills, portfolio, experience, badges, verification.
3. **Project Posting & Hiring:** Post jobs, proposals/bids, direct hire.
4. **Messaging & File Sharing:** Real-time chat, attachments, read receipts.
5. **Payments & Escrow:** Customer and freelancer will agree first on the pricing of the project , then in Progress-Project page details of payments for each piece of project delivered by freelancer and approver by customer will be discussed there.
6. **Reviews & Reputation:** Ratings, feedback, leaderboards, badges.
7. **Admin Dashboard:** User management, dispute resolution, analytics.
8. **Search & Matching:** Filtering, recommendations (future AI).
9. **Notifications:** In-app and email notifications.

2.3 User Classes and Characteristics

Guest: Browse public pages (landing, searchable portfolios).

Client: Post projects, hire freelancers, pay, leave reviews.

Freelancer: Create profile, submit proposals, accept work, deliver files.

Company: Create company profile, post internships, invite freelancers.

Admin: Manage users, moderate content, resolve disputes, configure platform settings.

2.4 Operating Environment

- **Client:** Modern browsers (Chrome, Firefox, Edge, Safari) on desktop and mobile. Responsive web design.
- **Server:** Linux server(s) (Ubuntu), Node.js runtime (recommended) or Django/Python alternative.
- **Database:** PostgreSQL (recommended) with Redis for caching and session/real-time pub/sub.
- **Storage:** Cloud object storage (AWS S3 or compatible) for uploaded files.
- **Third-party:** Payment gateways (Stripe/PayPal), email provider (SendGrid, Mailgun), SMS/OTP provider (optional).

2.5 Design and Implementation Constraints

- Project timeline for MVP: 3 months.
- Budget constraints — initial hosting on cost-effective platforms (e.g., Heroku/Render/Vercel + managed DB).
- Compliance with privacy best practices (GDPR-like considerations) for user data handling.
- Performance target: first contentful paint under 1.5s on typical connection; page load under 3s.

2.6 User Documentation

Planned documentation:

- User Guide (for Clients and Freelancers).
- Admin Guide.
- FAQ and Knowledge Base.
- Short tutorial videos / onboarding flows (Optional).

2.7 Assumptions and Dependencies

- Users have Internet access and modern browsers.
- Payment API providers maintain uptime and contractual access.
- Email/SMS providers deliver messages reliably.
- Universities and partners may cooperate for the ambassador program (optional).

3 System Features

3.1 System Feature 1 – User Registration and Authentication

Description and Priority

Secure account creation and login with role assignment. Priority: High.

Stimulus/Response Sequences

User submits registration form → system validates input → sends email verification → account activated.

Functional Requirements

- **FR-1.1:** Support email/password signup.
- **FR-1.2:** Support OAuth signup (Google).
- **FR-1.3:** Send email verification.
- **FR-1.4:** Password reset via secure token or OTP.
- **FR-1.5:** Role-based registration and access (Freelancer, Client, Company, Admin).
- **FR-1.6:** Session management using secure cookies or JWT with refresh tokens.

3.2 System Feature 2 – Freelancer Profile Management

Description and Priority

Freelancers create a public profile with portfolio and verifications. Priority: High.

Functional Requirements

- **FR-2.1:** Create/edit profile (photo, bio, skills, hourly rate).
- **FR-2.2:** Upload portfolio items (images, links, files).
- **FR-2.3:** View project history, earnings, and reviews.

3.3 System Feature 3 – Project Posting and Bidding

Description and Priority

Clients post jobs and freelancers can bid or be hired directly. Priority: High.

Functional Requirements

- **FR-3.1:** Clients create project postings (title, details, budget, deadline, visibility).
- **FR-3.2:** Freelancers submit proposals with price and message.
- **FR-3.3:** Clients can shortlist proposals and accept or decline.
- **FR-3.4:** Clients can direct-hire a freelancer without a public bidding phase.
- **FR-3.5:** Notifications for new proposals, acceptances, declines, job offers.

3.4 System Feature 4 – Messaging and Collaboration

Description and Priority

Real-time communication between users with file exchange. Priority: High.

Functional Requirements

- **FR-4.1:** One-to-one chat (WebSocket) with message persistence.
- **FR-4.2:** File attachments (size limits configurable).
- **FR-4.3:** Typing indicators and read receipts (optional).
- **FR-4.4:** Chat history related to a project (to be discussed in project progress page).

3.5 System Feature 5 – Payments

Description and Priority

Handle payments, escrow, commissions, and manual local payments. Priority: High.

Functional Requirements

- **FR-5.1:** Support manual local payment (bank transfer/BaridiMob/Edahabia) with receipt upload in project-progress page.
- **FR-5.2:** Record transaction histories and generate statements.
- **FR-5.3:** Apply platform commissions (configurable) and support subscription/credit models.

3.6 System Feature 6 – Reviews, Ratings and Reputation

Description and Priority

After project completion both parties can leave feedback to build reputation. Priority: High.

Functional Requirements

- **FR-6.1:** Clients rate freelancers and leave textual reviews.
- **FR-6.2:** Display aggregated rating and verified badges.
- **FR-6.3:** Provide mechanisms to contest reviews (admin mediation).

3.7 System Feature 7 – Search, Filtering, and Recommendations

Description and Priority

Find freelancers and projects with advanced filters and recommendation engine (future enhancement). Priority: Medium.

Functional Requirements

- **FR-7.1:** Keyword search, category, skill tags, location, rating, price filters.
- **FR-7.2:** Display recommended freelancers/projects based on basic heuristics (skills match, recent activity).
- **FR-7.3:** Support saved searches and alerts.

3.8 System Feature 8 – Admin and Moderation Tools

Description and Priority

Administrative functions for platform health, dispute resolution, and analytics. Priority: High.

Functional Requirements

- **FR-8.1:** View and manage users (suspend, verify, delete).
- **FR-8.2:** Manage disputes and refund requests.
- **FR-8.3:** Analytics dashboard: active users, active projects, revenue, daily metrics.

- **FR-8.4:** Configure platform-wide settings (commission rates, file limits).

3.9 System Feature 9 – Notifications and Emailing

Functional Requirements

- **FR-9.1:** In-app notifications for jobs, messages, and actions.
- **FR-9.2:** Email notifications for critical events (welcome, verification, receipts).
- **FR-9.3:** Configurable notification preferences per user.

4 External Interface Requirements

4.1 User Interfaces

The UI will be a modern, responsive single-page application (SPA) built with React. Key considerations:

- Accessible forms and content (WCAG basics).
- Fast navigation, lazy-loading lists, and pagination.
- Design system with reusable UI components and a consistent color palette (to be decided by the design team).

4.2 Hardware Interfaces

No direct hardware interfaces. Users upload files (documents, images) via browser.

4.3 Software Interfaces

- RESTful JSON or GraphQL API between frontend and backend.
- WebSocket (Socket.io or native ws) for chat and real-time notifications.
- Email delivery API (SendGrid/Mailgun).
- Object storage API (S3-compatible).

4.4 Communications Interfaces

All communication over HTTPS/TLS. JSON will be the primary payload format. Authentication using secure JWT or cookie-based sessions (HTTPS-only, Secure, SameSite).

5 Other Nonfunctional Requirements

5.1 Performance Requirements

- **NFR-Perf-1:** Support at least 200 concurrent active users for initial MVP (scale horizontally later).
- **NFR-Perf-2:** Page response time under 2.5s for typical pages; first contentful paint under 1.5s on modern connections.
- **NFR-Perf-3:** File uploads should support resumable uploads for large files.

5.2 Safety Requirements

- Daily backups with weekly off-site snapshot.
- Automatic alerts for data integrity issues and severe errors.

5.3 Security Requirements

- **NFR-Sec-1:** Passwords hashed with bcrypt/argon2 and never stored in plaintext.
- **NFR-Sec-2:** All endpoints protected against injection attacks, CSRF, and XSS.
- **NFR-Sec-3:** Role-based access control (RBAC) for all protected resources.
- **NFR-Sec-4:** Transport layer encryption (HTTPS/TLS v1.2+).
- **NFR-Sec-5:** Regular security audits and penetration testing prior to public launch.

5.4 Maintainability and Scalability

- Modular codebase with clear separation of concerns (API, auth, payments, chat).
- CI/CD pipeline for automated tests and deployment.
- Containerized deployments (Docker) with orchestration path (Kubernetes) for scalability.

5.5 Software Quality Attributes

- **Usability:** Easy onboarding flows and a guided "post a job" wizard.
- **Reliability:** 99.5% uptime target for MVP.
- **Portability:** Cloud-agnostic deployment options.

6 Other Requirements

Database Requirements

Primary tables (initial set):

- `users` (id, name, email, role, hashed_password, verified, profile_id, created_at, ...)
- `profiles` (user_id, bio, skills, hourly_rate, portfolio_refs, verification_status)
- `projects` (id, client_id, title, description, budget, status, created_at, deadline)
- `proposals` (id, project_id, freelancer_id, price, message, status)
- `messages` (id, sender_id, receiver_id, project_id, content, attachments, created_at)
- `payments` (id, payer_id, payee_id, amount, status, method, transaction_ref)
- `reviews` (id, project_id, reviewer_id, reviewee_id, rating, comment)

(Relationships and full ERD to be completed during design phase.)

Logging and Auditing

All critical actions (login attempts, profile verifications, admin actions) must be logged with timestamps and user IDs. Logs should be stored for a configurable retention period and accessible by Admins for forensic purposes.

Future Enhancements

- AI-based recommendations (freelancer-project matching).
- Advanced reporting and business intelligence dashboard.
- Mobile native apps (iOS and Android).
- Multi-language support.

7 Goals and Objectives

- **G-1:** Enable trusted collaboration between clients and freelancers with escrow payments.
- **G-2:** Reach 1,000 registered users within 6 months of launch (growth metric).
- **G-3:** Achieve user satisfaction above 90% in onboarding surveys.
- **G-4:** Maintain platform average response time under 2.5 seconds.

8 Stakeholders and Roles

Stakeholder	**Interest / Expectation**
Client	Easy posting, fast hiring, secure payment.
Freelancer	Steady opportunities, clear feedback, secure payments.
Admin	Platform stability, moderation tools, and analytics.
Developer Team	Clear requirements, predictable scope, maintainable architecture.
University Supervisor	Demonstration of team competencies and project viability.

9 Risk Analysis and Mitigation

Risk	**Mitigation Strategy**
Data loss due to server failure	Daily backups, replication, and off-site snapshots.
Low user engagement	Partner with universities, ambassador program, targeted marketing, gamification.
Security breaches	Regular audits, timely patching, least privilege access model, encrypted data at rest.
Unclear regulations around local payments	Consult legal advisors; design flexible payment flows.

10 Traceability Matrix

Req ID	Goal ID	Module	Test Case ID
FR-1.1, FR-1.2, FR-1.3	G-1	Authentication	TC-01 (Auth)
FR-2.1, FR-2.2	G-1	Profile Module	TC-02 (Profile)
FR-3.1, FR-3.2	G-1, G-2	Project Module	TC-05 (Project Posting)
FR-4.1, FR-4.2, FR-4.3	G-1	Messaging Module	TC-07 (Messaging)
FR-5.1, FR-5.2, FR-5.3	G-1	Payments	TC-10 (Escrow and Transactions)
FR-6.1, FR-6.2, FR-6.3	G-1, G-3	Reviews	TC-12 (Feedback and Ratings)
FR-7.1, FR-7.2	G-2	Search Module	TC-15 (Search and Recommendations)
FR-8.1, FR-8.2, FR-8.3	G-4	Admin Dashboard	TC-18 (Admin Control)
FR-9.1, FR-9.2	G-4	Notifications	TC-20 (Alerts and Emails)
NFR-Perf-1, NFR-Perf-2	G-4	Performance Testing	TC-25 (Load Test)
NFR-Sec-1, NFR-Sec-3	G-1	Security	TC-27 (Security Audit)

Table 1: Requirements Traceability Matrix linking Functional/Non-functional Requirements to Goals and Test Cases

11 Appendices

A. Acronyms and Abbreviations

API	Application Programming Interface
DBMS	Database Management System
FR	Functional Requirement
NFR	Non-Functional Requirement
RBAC	Role-Based Access Control
SPA	Single Page Application
UI/UX	User Interface / User Experience
MVP	Minimum Viable Product
JWT	JSON Web Token

B. Tools and Technologies (Proposed Stack)

- **Frontend:** React.js, Tailwind CSS, Redux Toolkit (for state management)
- **Backend:** Node.js with Express.js (or Django as alternative)
- **Database:** PostgreSQL
- **Real-time:** Socket.io
- **Deployment:** Docker, Render/Vercel (for MVP)
- **Version Control:** Git + GitHub
- **Testing:** Jest (unit tests), Postman (API tests)

C. References

- IEEE Std 830-1998: Recommended Practice for Software Requirements Specifications.
- Stripe and PayPal developer documentation.
- Academic and industry UX reports on marketplace design and retention.
- Competitor Analysis: Upwork, Fiverr, Freehali, DZFreelance.

D. Team Responsibilities

Team Member	Responsibilities
Cherif Taieb Ezzraimi	System architecture, backend logic, database schema design, API integration.
Aya El Atra Hezam	Frontend design, React component development, and UI/UX testing.
Dounia Saighi	Documentation lead, test case design, and non-functional requirements validation.
Idris Ahmed Brahmi	Payment integration, security requirements, and admin dashboard implementation.
Rania Mir	User onboarding, profile module, and frontend usability enhancements.

E. Glossary

- **Escrow:** A financial arrangement where a third party holds and regulates payment until the transaction conditions are met.
- **Bid/Proposal:** An offer submitted by a freelancer to perform a project at a specific cost and timeline.
- **MVP:** Minimum Viable Product — the first working version of the application that includes core functionalities.

Conclusion

This Software Requirements Specification serves as the foundational document for the development of the Freelancing Website. It defines all key system features, goals, and constraints needed to guide the design and implementation phases.

Once this SRS is approved, the next steps include:

- System Design Document (SDD) preparation — translating requirements into architectural and technical design.
- Database schema modeling (ERD).
- UI wireframing and design prototyping.
- Development phase initiation based on priority modules.

End of Document.