Software Requirements Specification (SRS)

Upwork Proposal Assistant

A Browser Extension for Intelligent Freelance Proposal Management

Prepared By:

University Project Team
Department of Computer Science
Namal University, Mianwali

Document Version: 1.0

Date: January 5, 2026

Prepared For:
CSC-225 – Software Engineering
Complex Computing Problem – Milestone 2

This document conforms to IEEE Std 830-1984 guidelines for Software Requirements Specifications

Revision History

Version	**Date**	**Description**	**Author**
0.1	Nov 15, 2025	Initial draft created after RP Meeting 1	Project Team
0.5	Nov 28, 2025	Added functional requirements and diagrams	Project Team
1.0	Jan 5, 2026	Final version with complete NFRs and validation	Project Team

Table 1: Document Revision History

Contents

Chapter 1

Introduction

1.1 Purpose

1.1.1 Document Purpose

The primary purpose of this Software Requirements Specification (SRS) is to provide a comprehensive and definitive description of the Upwork Proposal Assistant system. This document explicitly defines all functional and non-functional requirements to ensure that the final product aligns with the expectations of the Requirement Provider (RP) and meets the academic standards established by Namal University's Department of Computer Science.

1.1.2 Intended Audience

This SRS is designed to serve multiple stakeholders throughout the software development lifecycle:

- **Development Team:** Software engineers and developers who will design, implement, and test the system. This document provides them with a clear technical roadmap including architectural constraints, interface specifications, and performance benchmarks.

- **Project Managers:** Individuals responsible for planning, scheduling, and resource allocation. The SRS enables accurate effort estimation and milestone definition based on the scope and complexity of requirements.

- **Quality Assurance Team:** Testers and validation engineers who will verify that the implemented system meets all specified requirements. Each requirement in this document is written to be verifiable and measurable.

- **Requirement Provider (RP):** The client or stakeholder who commissioned this system. This document serves as a formal contract defining what will be delivered, establishing clear boundaries between in-scope and out-of-scope functionality.

- **Academic Evaluators:** Faculty members and instructors evaluating this project for CSC-225 Software Engineering. The document demonstrates comprehensive understanding of requirements engineering principles and IEEE standards compliance.
- **Maintenance Personnel:** Future developers who may need to enhance or modify the system. The traceability and clarity of requirements will facilitate efficient maintenance operations.

1.1.3 Document Benefits

By adhering to IEEE Std 830-1984 guidelines, this SRS provides several specific benefits:

1. **Establishes Contractual Baseline:** Creates a formal agreement between stakeholders on system functionality, reducing the risk of scope creep and misaligned expectations during development.
2. **Reduces Development Costs:** Early identification of requirements conflicts, ambiguities, and technical constraints minimizes expensive redesign and recoding efforts during later phases.
3. **Enables Accurate Estimation:** Provides a realistic foundation for cost estimation, resource planning, and schedule development based on clearly defined deliverables.
4. **Facilitates Validation and Verification:** Each requirement is specified in a measurable, testable format, enabling systematic quality assurance processes and acceptance testing.
5. **Supports System Evolution:** The modular organization and comprehensive traceability enable efficient modification and enhancement of the system throughout its operational lifetime.
6. **Improves Communication:** Serves as a single source of truth, eliminating ambiguity and ensuring all stakeholders share a common understanding of system behavior.

1.2 Scope

1.2.1 Product Name and Context

The system to be developed is the **Upwork Proposal Assistant**, a specialized browser extension designed to optimize the workflow of professional freelancers operating on the

Upwork platform. In the current competitive landscape of online freelancing, professionals spend significant time analyzing potential clients, crafting customized proposals, and managing their application history.

1.2.2 Problem Statement

Current freelance workflow challenges include:

- **Time Inefficiency:** Freelancers spend an average of 15-20 minutes per job application analyzing client history, reading reviews, and writing customized proposals.
- **Information Overload:** Upwork job listings contain extensive data (client spending history, hire rates, previous feedback, job requirements) that must be manually processed and evaluated.
- **Inconsistent Quality:** Without systematic analysis tools, freelancers may submit proposals to low-quality clients or fail to identify red flags in client behavior patterns.
- **Resource Wastage:** Upwork's "Connect" system charges freelancers for each application. Applying to clients with poor hiring records wastes both Connects and time.
- **Template Management:** Successful freelancers maintain libraries of proposal templates but lack tools to track which templates perform best or systematically improve their approach.

1.2.3 Solution Overview

The Upwork Proposal Assistant addresses these challenges through intelligent automation and data-driven decision support. The system will reduce proposal preparation time by approximately 60% through the following core capabilities:

Automated Client Analysis

The system shall automatically extract and analyze client profile data including:

- Total spending history and budget patterns
- Historical hire rate percentages
- Previous freelancer feedback and sentiment analysis
- Payment verification status and geographic location

- Job posting patterns and response rates

This analysis will be synthesized into a normalized ”Client Reliability Score” (1-100 scale) displayed prominently within the Upwork interface, enabling freelancers to make informed decisions about where to invest their limited Connects.

AI-Powered Proposal Generation

Leveraging large language models (LLMs), the system shall generate customized proposal drafts by:

- Parsing job description requirements and matching them to the freelancer's skill profile
- Adapting tone and style based on client characteristics (corporate vs. startup, technical vs. non-technical)
- Incorporating relevant portfolio examples and past project successes
- Maintaining professional quality while reducing writing time from 15 minutes to under 2 minutes

Template Management and Analytics

The system shall provide sophisticated template management including:

- Storage and categorization of proposal templates
- Performance tracking (views, interviews, hires per template)
- Automated ranking based on success metrics
- Historical analytics showing proposal effectiveness trends over time

1.2.4 System Boundaries

To ensure compliance with Upwork's Terms of Service and maintain ethical operation, the system explicitly **will NOT**:

- Automatically submit proposals without explicit user confirmation
- Circumvent Upwork's Connect payment system
- Violate Upwork's rate limiting or anti-automation policies
- Store or transmit Upwork user credentials (authentication handled via OAuth where possible)

- Scrape data from Upwork for purposes other than assisting the authenticated user
- Share or aggregate data across multiple users without explicit consent

The system functions strictly as an "Assistant" – augmenting human decision-making rather than replacing it. All final decisions regarding proposal submission remain with the freelancer.

1.2.5 Target Metrics and Success Criteria

Upon successful implementation, the system shall achieve:

- **Time Reduction:** Average proposal preparation time reduced from 15 minutes to 6 minutes or less (60% improvement)
- **Decision Quality:** 80% of users report improved ability to identify high-quality clients
- **Resource Efficiency:** 50% reduction in wasted Connects on low-probability applications
- **User Satisfaction:** System Usability Scale (SUS) score of 75 or higher
- **Performance:** 99.5% of analysis operations completed within 3 seconds

1.2.6 Benefits and Objectives

The successful deployment of this system will provide:

1. **Competitive Advantage:** Freelancers can respond to job postings faster and more effectively than competitors using manual processes.
2. **Income Optimization:** By avoiding low-quality clients and targeting high-value opportunities, freelancers can increase their effective hourly rate.
3. **Professional Development:** Analytics and performance tracking enable freelancers to systematically improve their proposal strategies over time.
4. **Stress Reduction:** Automation of tedious tasks reduces cognitive load and allows freelancers to focus on high-value creative work.
5. **Scalability:** The system enables freelancers to efficiently manage higher volumes of applications without proportional time investment.

1.3 Definitions, Acronyms, and Abbreviations

API	Application Programming Interface: A set of protocols and tools that enables communication between different software components. In this system, APIs facilitate interaction between the browser extension, backend server, and external AI services.
Client Reliability Score	A normalized numerical value (1-100) calculated by the system to represent the overall trustworthiness and quality of an Upwork client based on historical data analysis.
Connect	Upwork's internal currency used by freelancers to submit proposals. Each job application costs a specific number of Connects (typically 1-6), and freelancers must purchase additional Connects when their allocation is exhausted.
CSS Selector	A pattern-matching syntax used to identify and extract specific HTML elements from web pages. The system uses CSS selectors to scrape client data from Upwork's Document Object Model.
DOM	Document Object Model: The hierarchical tree structure representing a web page's content. The browser extension manipulates the DOM to inject UI elements and extract job data.
Extension Manifest	A JSON configuration file (manifest.json) that defines a browser extension's permissions, capabilities, scripts, and metadata. This system uses Manifest V3 specification.
Freelancer	The primary end-user of this system; a professional service provider who seeks and completes projects through the Upwork platform.
JSR	Job Success Rate: An Upwork metric representing the percentage of contracts a freelancer completes successfully, weighted by project value and client feedback.
LLM	Large Language Model: An artificial intelligence system trained on vast text corpora capable of generating human-like text. This system uses LLMs (such as GPT-4 or Claude) for proposal generation.

NFR	Non-Functional Requirement: A specification of system quality attributes (performance, security, usability) rather than specific behaviors or functions.
OAuth	Open Authorization: An industry-standard protocol for secure delegated access, potentially used for Upwork API authentication if direct integration becomes available.
Proposal	A formal written application submitted by a freelancer to a client's job posting, typically including an introduction, relevant experience, project approach, pricing, and call-to-action.
RP	Requirement Provider: The individual or entity responsible for defining system goals, validating requirements, and accepting final deliverables. In academic context, this may be a faculty advisor or external client.
REST API	Representational State Transfer Application Programming Interface: An architectural style for web services using HTTP methods (GET, POST, PUT, DELETE) for data exchange between the extension and backend server.
SRS	Software Requirements Specification: This formal document describing all functional and non-functional requirements for the system being developed, conforming to IEEE Std 830-1984 guidelines.
TBD	To Be Determined: A placeholder indicating that specific requirement details are not yet finalized and require future resolution by a specified milestone or meeting.
TLS	Transport Layer Security: A cryptographic protocol ensuring secure, encrypted communication between the browser extension and backend servers, protecting user data during transmission.
Upwork Platform	The online marketplace connecting freelancers with clients seeking professional services. The system interfaces with Upwork's web application but operates independently.
Use Case	A specific scenario describing how an actor (user or external system) interacts with the system to achieve a particular goal, including preconditions, main flow, and expected outcomes.

1.4 References

1. ANSI/IEEE Std 830-1984, *IEEE Guide to Software Requirements Specifications.* Institute of Electrical and Electronics Engineers, 1984.
2. IEEE Std 729-1983, *IEEE Standard Glossary of Software Engineering Terminology.* Institute of Electrical and Electronics Engineers, 1983.
3. Namal University Department of Computer Science, *CSC-225 Software Engineering: Complex Computing Problem – Milestone 2 Guidelines*, November 2025.
4. Upwork Inc., *Upwork Terms of Service*, Available at: https://www.upwork.com/legal, Accessed December 2025.
5. Upwork Inc., *Upwork API Documentation*, Available at: https://developers.upwork.com/, Accessed December 2025.
6. Mozilla Developer Network, *Browser Extensions API Documentation*, Available at: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions, Accessed December 2025.
7. Google Chrome Developers, *Chrome Extension Manifest V3 Migration Guide*, Available at: https://developer.chrome.com/docs/extensions/mv3/, Accessed December 2025.
8. OpenAI, *GPT-4 API Reference Documentation*, Available at: https://platform.openai.com/docs/ Accessed December 2025.
9. Anthropic, *Claude API Documentation*, Available at: https://docs.anthropic.com/, Accessed December 2025.
10. World Wide Web Consortium (W3C), *Web Content Accessibility Guidelines (WCAG) 2.1*, Available at: https://www.w3.org/WAI/WCAG21/quickref/, June 2018.
11. Internet Engineering Task Force (IETF), *RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3*, August 2018.
12. European Parliament and Council, *General Data Protection Regulation (GDPR)*, Regulation (EU) 2016/679, April 2016.

1.5 Document Overview

This Software Requirements Specification is organized into eight primary chapters designed to provide comprehensive coverage of all system aspects while maintaining readability and logical flow.

1.5.1 Document Structure

Chapter 1: Introduction (Current Chapter) establishes the document's purpose, defines the system scope, provides essential terminology, and outlines the overall document organization. This chapter ensures all readers share a common understanding of objectives and context.

Chapter 2: Overall Description provides high-level context for the system including its relationship to other products, general capabilities, user characteristics, operating environment, design constraints, and fundamental assumptions. This chapter bridges the gap between project vision and detailed requirements.

Chapter 3: External Interface Requirements specifies all touchpoints between the system and external entities including user interfaces, hardware platforms, software dependencies, and communication protocols. This chapter ensures complete integration planning.

Chapter 4: Functional Requirements contains the core behavioral specifications – what the system shall do. Each requirement is uniquely identified, written in verifiable format, and organized by functional area for clarity and traceability.

Chapter 5: Non-Functional Requirements defines quality attributes and constraints including performance benchmarks, security standards, usability criteria, maintainability guidelines, and legal compliance requirements. These specifications ensure the system meets operational excellence standards.

Chapter 6: System Models and Diagrams presents visual representations including Use Case Diagrams and Context Diagrams that illustrate system boundaries, actor interactions, and data flows. These models complement textual requirements with graphical clarity.

Chapter 7: Appendices contains supporting materials including sample data formats, UI mockups, API specifications, meeting minutes summaries, and glossaries. This supplementary information aids understanding without cluttering primary requirement sections.

Chapter 8: Index and Traceability provides cross-referencing tools including a comprehensive index of key terms and a requirements traceability matrix linking functional requirements to non-functional constraints and test cases.

1.5.2 Reading Guidance

For Project Managers and Executives: Focus on Chapters 1, 2, and 5 to understand scope, objectives, and quality commitments without technical implementation details.

For Developers and Architects: Chapters 3, 4, and 6 provide the technical specifications required for system design and implementation, including interface contracts and behavioral requirements.

For Quality Assurance Teams: All chapters are relevant, with particular emphasis on Chapters 4 and 5 where each requirement specifies its verification method and acceptance criteria.

For Academic Evaluators: The document demonstrates comprehensive requirements engineering practices, IEEE standard compliance, and systematic software development methodology throughout all chapters.

1.5.3 Document Maintenance

This SRS is maintained under version control with all changes tracked in the Revision History table. Modifications to requirements must follow the formal change control process:

1. Change requests are documented with rationale and impact analysis
2. Stakeholders review and approve changes in formal review meetings
3. Approved changes are incorporated with updated version numbers
4. Affected downstream artifacts (design documents, test plans) are updated accordingly
5. All modifications maintain backward traceability to source requirements

The document is reviewed and revalidated at major project milestones to ensure continued alignment with stakeholder needs and technical feasibility.

Chapter 2

Overall Description

2.1 Product Perspective

2.1.1 System Context

The Upwork Proposal Assistant exists within a complex ecosystem of interconnected technologies and platforms. Understanding this context is essential for proper system design and integration planning.

The system is neither a standalone product nor a component of a larger pre-existing software suite. Instead, it functions as an **augmentation layer** that integrates with existing platforms and services. The primary integration point is the Upwork web platform, accessed through standard web browsers (Chrome, Firefox, Edge, Brave). The system does not replace or fundamentally alter Upwork's functionality; rather, it enhances the user experience by providing additional analytical and automation capabilities.

2.1.2 Architectural Overview

The Upwork Proposal Assistant employs a three-tier distributed architecture:

Tier 1: Client-Side Browser Extension

The front-end component runs entirely within the user's web browser as a packaged extension conforming to Manifest V3 specifications. This component is responsible for:

- **DOM Manipulation:** Injecting visual overlays, badges, and sidebars into Upwork pages without disrupting native functionality
- **Data Extraction:** Parsing HTML structure using CSS selectors and XPath queries to extract client data, job descriptions, and user profile information
- **Local Storage Management:** Caching frequently accessed data (templates, user preferences) in browser local storage for offline access
- **Event Handling:** Listening for user interactions (button clicks, page navigation) and triggering appropriate backend requests

- **Real-Time UI Updates:** Displaying analysis results, proposal drafts, and notifications without requiring page refreshes

The extension maintains minimal footprint (under 5MB packaged size) to ensure fast installation and updates. All computationally intensive operations (AI inference, sentiment analysis) are delegated to the backend tier to prevent browser performance degradation.

Tier 2: Backend Application Server

A Node.js-based REST API server deployed on cloud infrastructure (AWS, Google Cloud, or Azure) handles business logic and orchestrates external service integrations. Core responsibilities include:

- **Authentication and Authorization:** Validating user sessions using JWT (JSON Web Tokens) and enforcing role-based access control
- **Data Processing Pipeline:** Executing client analysis algorithms, sentiment scoring, and reliability calculations
- **AI Service Integration:** Managing API calls to LLM providers (OpenAI, Anthropic) with request queuing, retry logic, and error handling
- **Analytics Aggregation:** Processing proposal history data to generate performance reports and success metrics
- **Cache Management:** Implementing Redis-based caching to reduce database load and improve response times
- **Rate Limiting:** Enforcing usage quotas to prevent abuse and manage API costs

The backend architecture follows microservices principles, with distinct modules for analysis, generation, authentication, and analytics. This modularity enables independent scaling, testing, and deployment of individual components.

Tier 3: Data Persistence Layer

A MongoDB database cluster provides persistent storage for:

- **User Profiles:** Account information, skill configurations, preferences, and subscription tiers
- **Proposal Templates:** User-created and system-provided templates with associated metadata (success rates, usage counts, tags)

- **Historical Data:** Complete archive of generated proposals, client analyses, and decision outcomes for analytics
- **System Configuration:** API keys, feature flags, rate limits, and operational parameters

MongoDB was selected for its flexible schema design, horizontal scalability, and native JSON document storage aligning with JavaScript-based application architecture.

2.1.3 External System Dependencies

The Upwork Proposal Assistant interfaces with several external systems:

Upwork Web Platform (Primary External System)

- **Interface Type:** Indirect DOM-based interaction (no official API integration)
- **Data Flow:** Unidirectional read-only access to public job posting data
- **Dependency Risk:** High – Changes to Upwork's HTML structure may break data extraction
- **Mitigation Strategy:** Implement adaptive CSS selectors with fallback patterns and automated regression testing

Large Language Model APIs (AI Services)

- **Primary Provider:** OpenAI GPT-4 or Anthropic Claude (configurable)
- **Interface Type:** RESTful HTTPS API using JSON payloads
- **Data Flow:** Bidirectional – System sends context prompts; AI returns generated text
- **Dependency Risk:** Medium – Service outages impact proposal generation but not analysis features
- **Mitigation Strategy:** Implement provider failover and graceful degradation to manual templates

Browser Extension Stores

- **Platforms:** Chrome Web Store, Firefox Add-ons, Microsoft Edge Add-ons
- **Interface Type:** Manual upload and automated update distribution

- **Dependency Risk:** Low – Only affects distribution, not operation
- **Compliance Requirements:** Must adhere to each platform's content policies and technical requirements

2.1.4 System Boundaries and Constraints

Technical Boundaries

The system operates exclusively within the boundaries of:

- Modern web browsers supporting JavaScript ES6+ and Web Extensions API
- Upwork's publicly accessible job posting pages (does not access authenticated areas beyond user's own account)
- HTTPS-encrypted communication channels for all external data transmission
- User-granted browser permissions (no unauthorized data access)

Functional Boundaries

The system explicitly **does not**:

- Guarantee hiring success (provides decision support only)
- Automate bid placement or contract acceptance
- Modify Upwork's native functionality or appearance permanently
- Store Upwork account credentials (uses session-based authentication)
- Share individual user data across accounts without explicit consent

Legal and Ethical Boundaries

The system respects:

- Upwork's Terms of Service regarding automated access
- GDPR and data protection regulations for user privacy
- Intellectual property rights (no plagiarism in AI-generated content)
- Fair use principles in data analysis and processing

2.1.5 Operational Perspective

Deployment Model

The system follows a hybrid deployment approach:

- **Extension:** Distributed through official browser extension marketplaces with automatic update mechanisms
- **Backend:** Deployed on containerized cloud infrastructure (Docker/Kubernetes) with auto-scaling capabilities
- **Database:** Managed MongoDB Atlas cluster with automated backups and point-in-time recovery

Operational Lifecycle

The typical user journey spans:

1. **Discovery and Installation:** User finds extension in browser store, reviews permissions, and installs
2. **Onboarding:** First-run configuration wizard collects profile data and preferences
3. **Daily Usage:** User browses Upwork jobs; system provides real-time analysis and proposal assistance
4. **Continuous Improvement:** System learns from user feedback and proposal outcomes to refine recommendations
5. **Subscription Management:** User may upgrade tiers for advanced features (premium AI models, unlimited analyses)

Maintenance and Support

System maintenance encompasses:

- **Proactive Monitoring:** 24/7 automated health checks on API endpoints, database connections, and third-party service availability
- **Reactive Support:** User-reported issues triaged through in-app feedback system and support ticketing
- **Continuous Updates:** Monthly feature releases and weekly security patches delivered via automatic extension updates
- **Data Integrity:** Daily automated backups with 30-day retention and quarterly disaster recovery drills

2.2 Product Functions

This section provides a high-level summary of system capabilities organized by functional area. Detailed specifications for each function appear in Chapter 4 (Functional Requirements).

2.2.1 Client Intelligence and Analysis

Automated Profile Evaluation (FR-001)

The system shall autonomously extract and analyze client historical data from Upwork job postings including total spending, hire rate percentages, payment verification status, and geographic location. This data shall be normalized into a Client Reliability Score (1-100 scale) using weighted algorithms that prioritize hiring history (40%), spending patterns (30%), and review sentiment (30%).

Sentiment Analysis of Client Reviews (FR-001)

The system shall perform natural language processing on previous freelancer feedback to identify patterns indicating positive or negative client experiences. Red-flag keywords (e.g., "unresponsive," "payment issues," "scope creep") shall be automatically detected and factored into reliability calculations.

Competitive Analysis (FR-003)

The system shall evaluate the number of existing applicants, average interview rates, and job posting recency to estimate the probability of the user's proposal being viewed and considered. This competitive intelligence enables strategic decision-making about Connect allocation.

2.2.2 Intelligent Proposal Generation

AI-Driven Draft Creation (FR-002)

Leveraging large language models, the system shall generate customized proposal drafts by synthesizing job description requirements with the user's professional profile. The generated text shall maintain professional quality, incorporate relevant experience examples, and adapt tone based on client characteristics.

Context-Aware Customization (FR-002, FR-007)

The system shall analyze job descriptions to identify key requirements, preferred skills, and client priorities. Proposal generation shall emphasize matching qualifications and de-emphasize irrelevant experience, maximizing perceived fit.

Style and Tone Adaptation (FR-007)

Users shall configure proposal voice settings (Professional, Friendly, Technical, Concise) that influence AI generation parameters. The system shall maintain stylistic consistency while ensuring authenticity and avoiding generic template language.

2.2.3 Template and Content Management

Template Library Management (FR-004)

The system shall provide a comprehensive repository for storing, organizing, and retrieving proposal templates. Each template shall support dynamic placeholders (e.g., [Client_Name], [Project_Budget]) that automatically populate from job posting data.

Performance-Based Ranking (FR-004)

Templates shall be automatically ranked using a "Success Velocity" metric calculated as: $V = \frac{Interviews}{TotalUses} \times 100$. High-performing templates shall be promoted in selection interfaces, while consistently underperforming templates shall be flagged for revision.

Template Optimization Recommendations (FR-004)

The system shall analyze successful vs. unsuccessful proposals to identify effective keywords, optimal length ranges, and structural patterns. Users shall receive actionable suggestions for improving template performance.

2.2.4 Historical Analytics and Performance Tracking

Comprehensive Proposal Archive (FR-005)

The system shall maintain a complete historical record of all generated proposals, associated client analyses, and application outcomes (views, interviews, hires). This archive enables longitudinal performance analysis and strategy refinement.

Success Metrics Dashboard (FR-005)

Users shall access visual analytics including:

- Weekly proposal volume trends (line graphs)
- Client reliability distribution of applied jobs (histogram)
- Template performance comparison (bar charts)

- Interview-to-application conversion rates (pie charts)
- Average Connect cost per interview (calculated metric)

Trend Analysis and Insights (FR-005)

The system shall identify patterns such as:

- Time-of-day correlations with proposal success
- Client characteristics associated with higher interview rates
- Keyword effectiveness in different job categories
- Seasonal variations in market competitiveness

2.2.5 User Profile and Skill Configuration

Professional Identity Management (FR-006)

Users shall define their professional profile including:

- Biographical summary and career highlights
- Technical skill inventory with proficiency levels
- Portfolio links and work sample references
- Hourly rate ranges and availability preferences
- Specialization areas and target industries

Skill-Job Matching (FR-006)

The system shall cross-reference job requirements against the user's configured skills to calculate a compatibility percentage. Applications to jobs where the user lacks 50

Dynamic Profile Updates (FR-006)

As users gain experience and complete projects, the system shall suggest profile updates to reflect new capabilities and accomplishments, ensuring proposal generation utilizes current information.

2.2.6 Decision Support and Recommendations

Multi-Tier Analysis Engine (FR-003)

The system shall apply sophisticated logic gates to generate application recommendations:

- **Tier 1 (Resource Protection):** Flag jobs with 50+ applicants and client hire rate below 30
- **Tier 2 (Quality Scoring):** Categorize jobs as Strong Apply (score 80+), Caution (50-79), or Skip (below 50)
- **Tier 3 (Urgency Detection):** Identify time-sensitive opportunities (posted within 30 minutes, high hire rate) with "Urgent Action" badges

Reasoning Transparency (FR-003)

Each recommendation shall include a plain-language explanation of the underlying rationale (e.g., "Client has 90

2.2.7 System Administration and Maintenance

Global Template Repository (FR-008)

Administrators shall manage a curated library of high-performing templates shared across the user base (with appropriate anonymization). Community-sourced templates undergo quality review before publication.

System Health Monitoring (FR-008)

Administrative dashboards shall display real-time metrics including:

- API response time percentiles (p50, p95, p99)
- Error rates by endpoint and failure type
- Database query performance and connection pool utilization
- Third-party service availability (AI APIs, external dependencies)

Configuration Management (FR-008)

Administrators shall control system-wide settings including:

- AI model selection and prompt engineering parameters
- Rate limiting thresholds and usage quotas per tier

- Feature flags for gradual rollout of new capabilities
- API key rotation for security compliance

2.2.8 Data Synchronization and Persistence

Cross-Device Sync (FR-009)

User data (profiles, templates, settings) shall synchronize across multiple devices in near-real-time using event-driven architecture. Conflict resolution shall prioritize most-recent-write with user override options.

Offline Capability (FR-009)

The system shall cache essential data locally, enabling limited functionality (template access, manual proposal drafting) during network outages. Upon reconnection, pending operations shall queue for automatic synchronization.

Data Security and Encryption (FR-009)

All data transmission shall employ TLS 1.3 encryption. Data at rest shall be encrypted using AES-256. User-specific encryption keys shall derive from account credentials to ensure only authorized access.

2.3 User Classes and Characteristics

The Upwork Proposal Assistant serves a diverse user base with varying technical proficiency, freelancing experience, and usage patterns. Understanding these user classes ensures appropriate interface design, feature prioritization, and documentation strategies.

2.3.1 Primary User Class: Professional Freelancers

Novice Freelancers

- **Characteristics:** Less than 6 months active on Upwork; limited proposal writing experience; uncertain about client evaluation; typically possess 1-2 specialized skills
- **Technical Proficiency:** Basic computer literacy; comfortable with web browsers and common applications; may struggle with advanced configuration
- **Primary Needs:** Step-by-step guidance; clear explanations of client metrics; pre-built template options; confidence-building feedback

- **Usage Frequency:** Daily during initial ramp-up phase; seeking to build profile and reputation
- **Pain Points:** Fear of wasting Connects; uncertainty about competitive positioning; lack of benchmark data
- **System Adaptations:** Simplified onboarding wizard; contextual help tooltips; conservative recommendation defaults; extensive tutorial content

Intermediate Freelancers

- **Characteristics:** 6-24 months Upwork experience; established JSR (Job Success Rate) 85
- **Technical Proficiency:** Comfortable with browser extensions; willing to customize settings; moderate understanding of analytics
- **Primary Needs:** Efficiency gains; template performance optimization; competitive intelligence; time-saving automation
- **Usage Frequency:** 3-5 times per week for targeted applications; strategic rather than volume-based approach
- **Pain Points:** Time constraints balancing project work with new business development; difficulty scaling application volume without quality degradation
- **System Adaptations:** Advanced filtering options; keyboard shortcuts; batch analysis features; customizable workflows

Expert Freelancers

- **Characteristics:** 2+ years platform experience; Top Rated or Top Rated Plus badges; selective about opportunities; specialized in high-value niches; annual earnings $75k+$
- **Technical Proficiency:** Advanced users comfortable with APIs, automation tools, and data analysis; may request programmatic access
- **Primary Needs:** Deep analytics; predictive modeling; integration with CRM tools; ROI optimization; market intelligence
- **Usage Frequency:** Weekly for highly selective applications; focus on quality over quantity
- **Pain Points:** Distinguishing genuinely premium opportunities from inflated budgets; maintaining competitive edge as market matures

- **System Adaptations:** Advanced analytics dashboard; API access for custom integrations; white-label options; priority support

2.3.2 Secondary User Class: System Administrators

Technical Administrators

- **Responsibilities:** Backend infrastructure management; database optimization; API integration maintenance; security monitoring
- **Technical Proficiency:** Expert-level software engineering; DevOps and cloud platform expertise; security best practices knowledge
- **Primary Needs:** Comprehensive logging; performance monitoring dashboards; automated alerting; deployment automation tools
- **System Adaptations:** Admin control panel with system health metrics; log aggregation tools; configuration management interfaces; automated backup and recovery tools

Content Administrators

- **Responsibilities:** Template library curation; user support and training; content moderation; quality assurance
- **Technical Proficiency:** Moderate technical skills; strong understanding of freelancing best practices; excellent communication abilities
- **Primary Needs:** Template management tools; user feedback analysis; content publishing workflows; moderation queues
- **System Adaptations:** Simplified admin interface; bulk editing capabilities; approval workflows; user communication tools

2.3.3 Tertiary User Class: Stakeholders

Business Analysts

- **Interests:** Market trends; feature adoption rates; user retention metrics; revenue optimization
- **System Needs:** Aggregated analytics dashboards; cohort analysis tools; A/B testing frameworks; export capabilities

Compliance Officers

- **Interests:** GDPR compliance; Terms of Service adherence; data retention policies; security audit trails
- **System Needs:** Audit logs; data access reports; consent management tools; right-to-deletion workflows

2.3.4 User Characteristic Summary Matrix

User Class	**Tech Skill**	**Frequency**	**Experience**	**Primary Goal**
				Novice Freelancer
Basic	Daily	0-6 months	Learning	Building Profile
				Intermediate
Moderate	3-5x/week	6-24 months	Efficiency	Growth
				Expert
Advanced	Weekly	2+ years	Optimization	Scale
				Tech Admin
Expert	Daily	N/A	System Stability	Moderate
			Content Admin	
Daily	N/A	Quality	Support	

Table 2.1: User Class Characteristics Summary

2.4 Operating Environment

The Upwork Proposal Assistant must function reliably across diverse computing environments while maintaining consistent performance and user experience. This section defines the technical platforms, configurations, and environmental constraints within which the system operates.

2.4.1 Client-Side Environment Requirements

Supported Web Browsers

The browser extension shall be compatible with:

- **Google Chrome:** Version 100 and later (released April 2022+)
- **Microsoft Edge:** Version 100 and later (Chromium-based versions only)
- **Brave Browser:** Version 1.40 and later

- **Mozilla Firefox:** Version 95 and later (released December 2021+)
- **Opera:** Version 85 and later (Chromium-based)

Rationale: These minimum versions ensure support for Manifest V3, ES2020 JavaScript features, and modern Web Extensions APIs required for DOM manipulation and secure communication. **Browser Limitations:** Safari is explicitly not supported in initial release due to incompatibilities between Safari's extension architecture and Chrome/Firefox standards. Mobile browsers (iOS Safari, Chrome Mobile) are not supported as Upwork's mobile interface differs substantially from desktop.

Operating System Compatibility

The system shall operate on:

- **Windows:** Windows 10 (Version 1909+) and Windows 11
- **macOS:** macOS 12 (Monterey) and later versions
- **Linux:** Ubuntu 20.04 LTS+, Fedora 35+, or any distribution supporting modern Chromium/Firefox builds

The system is operating system agnostic at the application level (no OS-specific APIs used), but OS choice affects browser availability and update mechanisms.

Hardware Requirements

Minimum Configuration:

- **Processor:** Dual-core CPU at 1.6 GHz (Intel Core i3 or AMD equivalent)
- **RAM:** 4 GB system memory (with 2 GB available for browser)
- **Storage:** 50 MB free disk space for extension installation
- **Display:** 1280x720 resolution (HD)
- **Network:** Broadband internet connection (5 Mbps download minimum)

Recommended Configuration:

- **Processor:** Quad-core CPU at 2.4 GHz or better
- **RAM:** 8 GB system memory
- **Storage:** SSD (for faster extension loading)
- **Display:** 1920x1080 resolution (Full HD) or higher

- **Network:** 25 Mbps+ broadband connection

Input Devices: Standard keyboard and mouse/trackpad required. Touchscreen support is provided on compatible devices but not optimized as primary interaction method.

2.4.2 Backend Infrastructure Environment

Cloud Platform Requirements

Backend services shall be deployed on:

- **Primary Platform:** Amazon Web Services (AWS) or Google Cloud Platform (GCP)
- **Compute:** Containerized deployment using Docker containers orchestrated by Kubernetes
- **Auto-Scaling:** Elastic compute instances that scale based on CPU utilization (target: 70
- **Geographic Distribution:** Multi-region deployment (US-East, EU-West, Asia-Pacific) for latency optimization

Database Environment

- **Database System:** MongoDB Atlas (managed cloud service)
- **Cluster Configuration:** M10+ tier with replica sets for high availability
- **Storage:** Minimum 100 GB provisioned IOPS storage with automatic expansion
- **Backup:** Continuous backup with point-in-time recovery to any point within 7 days

Caching Layer

- **Cache System:** Redis 6.0+ for session management and frequent query results
- **Configuration:** In-memory cache with persistence enabled for durability
- **Capacity:** 4-16 GB memory allocation based on user base size

2.4.3 Network and Connectivity Requirements

User Network Requirements

- **Minimum Bandwidth:** 5 Mbps download / 1 Mbps upload for basic functionality
- **Recommended Bandwidth:** 25+ Mbps download / 5+ Mbps upload for optimal AI proposal generation
- **Latency:** Sub-200ms round-trip time to backend servers for responsive UI
- **Stability:** Persistent connection not required; system handles intermittent connectivity gracefully

Security Protocols

- **Encryption:** All communications use TLS 1.3 with strong cipher suites only
- **Certificate Validation:** Extension validates server certificates to prevent MITM attacks
- **Firewall Compatibility:** System operates over standard HTTPS (port 443); no special firewall rules required

2.4.4 External Service Dependencies

AI Service Providers

- **Primary Provider:** OpenAI API (GPT-4) or Anthropic API (Claude)
- **Availability Requirements:** 99.5
- **Fallback:** Secondary AI provider or local template system if primary unavailable

Upwork Platform Dependency

- **Availability:** System functionality contingent on Upwork.com accessibility
- **Version Compatibility:** Must adapt to Upwork's front-end updates without prior notice
- **Access Method:** Standard HTTPS requests to public pages; no private API access required

2.4.5 Development and Testing Environments

Development Environment

- **IDE:** Visual Studio Code or WebStorm
- **Node.js:** Version 18 LTS or later
- **Package Manager:** npm or yarn
- **Version Control:** Git with GitHub/GitLab hosting
- **Local Database:** MongoDB Community Edition or Docker container

Testing Environment

- **Browser Testing:** BrowserStack or Sauce Labs for cross-browser validation
- **API Testing:** Postman or Insomnia for endpoint verification
- **Load Testing:** Apache JMeter or k6 for performance benchmarking
- **CI/CD:** GitHub Actions or Jenkins for automated testing pipeline

2.4.6 Accessibility Environment

The system shall support users with disabilities through:

- **Screen Readers:** JAWS, NVDA, VoiceOver compatibility via semantic HTML and ARIA labels
- **Keyboard Navigation:** Full functionality accessible without mouse (Tab, Enter, Esc shortcuts)
- **Visual Accommodations:** High contrast mode support; respects OS-level zoom settings (up to 200
- **Color Blindness:** Information conveyed through shape and text labels in addition to color

2.5 Design and Implementation Constraints

This section identifies restrictions and limitations that constrain design choices and implementation approaches. These constraints arise from technical standards, business policies, legal requirements, and platform limitations.

2.5.1 Standards Compliance Constraints

IEEE Software Engineering Standards

- The system design and documentation shall conform to IEEE Std 830-1984 for requirements specification
- Testing procedures shall align with IEEE Std 829-1983 for test documentation
- Quality assurance processes shall follow IEEE Std 730-1981 guidelines

Web Standards and Protocols

- All frontend code shall comply with W3C HTML5 and CSS3 specifications
- JavaScript code shall adhere to ECMAScript 2020 (ES11) standard or later
- API communications shall follow REST architectural principles
- Data exchange formats shall use JSON conforming to RFC 8259

Accessibility Standards

- User interfaces shall meet WCAG 2.1 Level AA compliance requirements
- Color contrast ratios shall meet or exceed 4.5:1 for normal text, 3:1 for large text
- Interactive elements shall provide keyboard alternatives to mouse operations

2.5.2 Browser Extension Platform Constraints

Manifest V3 Requirements

- Extension must use Manifest V3 format (Manifest V2 deprecated by Chrome in 2023)
- Background scripts replaced by service workers (no persistent background pages)
- Content Security Policy restricts eval() and inline script execution
- Remote code execution prohibited (all logic must be packaged with extension)

Permission and Security Restrictions

- Extension may only request permissions essential to declared functionality
- Access to user browsing history beyond Upwork domain is prohibited
- Storage quota limited to 10 MB for chrome.storage.local
- Cross-origin requests require explicit host permissions in manifest

Marketplace Policy Compliance

- Chrome Web Store: No obfuscated code; clear privacy policy; no cryptocurrency mining
- Firefox Add-ons: Source code review required; no tracking without disclosure
- Edge Add-ons: Microsoft Store compliance including content rating

2.5.3 Legal and Regulatory Constraints

Data Protection Regulations

- **GDPR Compliance (EU users):**
  - Explicit user consent required before data collection
  - Right to data portability: Export user data in machine-readable format
  - Right to erasure: Permanent deletion within 30 days of request
  - Data processing logs maintained for audit purposes
- **CCPA Compliance (California users):**
  - Disclosure of data collection practices in privacy policy
  - Opt-out mechanism for data sale (though system does not sell data)
  - Non-discrimination for users exercising privacy rights

Upwork Platform Terms of Service

- System shall not automate bid submissions (violates Section 5.2 of Upwork ToS)
- Data scraping must respect Upwork's robots.txt directives
- Rate limiting must prevent excessive server load (maximum 10 requests per minute)
- No circumvention of Upwork's Connect payment system
- User accounts remain subject to Upwork's acceptable use policies

Intellectual Property Constraints

- AI-generated proposals must be reviewed to prevent plagiarism
- System shall not reproduce copyrighted training content verbatim
- User-submitted templates remain property of the user (system gains license to use)
- Upwork trademarks used only for compatibility description (no endorsement implied)

2.5.4 Technical Architecture Constraints

Programming Language Restrictions

- **Frontend:** JavaScript/TypeScript only (as required by browser extension APIs)
- **Backend:** Node.js JavaScript for consistency with frontend (alternative: Python allowed for ML components)
- **Database Queries:** MongoDB Query Language (no SQL injection risk)

Framework and Library Constraints

- Frontend framework limited to lightweight options (React, Vue, Svelte) due to extension size limits
- UI component libraries must not conflict with Upwork's native styles (CSS scoping required)
- External dependencies must be bundled (no CDN loading for security)
- Total bundled extension size shall not exceed 5 MB (Chrome Web Store recommendation)

Database Design Constraints

- MongoDB schema must support horizontal sharding for scalability
- Denormalization acceptable for read-heavy operations (proposal history queries)
- Sensitive fields (API keys, user credentials) must use field-level encryption
- Indexes required on frequently queried fields (user$_i d, timestamp, template_i d$)

2.5.5 Performance and Resource Constraints

Client-Side Resource Limits

- Extension memory footprint shall not exceed 150 MB during active use
- CPU utilization limited to 5
- DOM manipulation operations must complete within one frame (16ms) to prevent jank
- localStorage usage capped at 5 MB to leave room for other extensions

Backend Infrastructure Limits

- API response time must meet p95 latency target of 500ms (95th percentile)
- Database connection pool limited to 100 concurrent connections per instance
- AI API costs constrained to 0.10$perproposalgenerationonaverage$
- Bandwidth costs maintained under 0.05$peruserpermonth$

2.5.6 Security and Privacy Constraints

Authentication and Authorization

- User passwords must never be transmitted or stored by the system
- Session tokens expire after 48 hours of inactivity (configurable by admin)
- Multi-factor authentication required for administrative access
- API keys stored in environment variables (never hardcoded in source)

Data Encryption Requirements

- All data in transit encrypted using TLS 1.3 with Perfect Forward Secrecy
- Data at rest encrypted using AES-256-GCM algorithm
- Encryption keys rotated quarterly and after any suspected compromise
- Backup data encrypted separately with distinct key hierarchy

Audit and Logging Constraints

- Security events logged with immutable audit trail (write-once storage)
- Logs retained for minimum 90 days, maximum 2 years
- Personal identifiable information (PII) redacted from logs per GDPR
- Log access restricted to authorized personnel with change tracking

2.5.7 Business and Operational Constraints

Budget Constraints

- Cloud infrastructure costs capped at $5,000/monthforfirst10,000users$
- AI API usage limited to 50,000 proposals/month on initial budget
- Development team limited to 4 full-time engineers plus 2 part-time QA testers

Timeline Constraints

- Minimum Viable Product (MVP) delivery deadline: April 2026
- Beta testing phase must complete before public launch
- Major version updates constrained to quarterly release schedule

Support and Maintenance Constraints

- User support provided during business hours only (9 AM - 5 PM UTC) initially
- Critical bug fixes delivered within 48 hours
- Feature requests prioritized based on user voting system
- System must support graceful degradation (core features work if AI API fails)

2.5.8 Interoperability Constraints

- System must not interfere with other installed browser extensions
- Must coexist with Upwork's native browser extension (if any)
- API design must allow future mobile app development
- Data export formats must be importable by common productivity tools (CSV, JSON)

2.6 User Documentation

Comprehensive, accessible documentation is essential for user adoption, system maintainability, and support efficiency. This section specifies the documentation artifacts that shall accompany the Upwork Proposal Assistant.

2.6.1 End-User Documentation

Quick Start Guide

A concise, visual guide enabling new users to achieve their first proposal generation within 5 minutes:

- **Format:** Interactive web-based tutorial with embedded screenshots and short video clips
- **Content:** Installation instructions, permission explanations, initial profile setup, first analysis walkthrough
- **Length:** Maximum 1,500 words across 5 screens/pages
- **Accessibility:** Available in-app during first launch; accessible via Help menu thereafter

Complete User Manual

Comprehensive reference documentation covering all system features:

- **Format:** Searchable online knowledge base with categorized articles
- **Structure:** Organized by user task (analyzing clients, generating proposals, managing templates, reviewing analytics)
- **Content Includes:**
  - Feature descriptions with annotated screenshots
  - Step-by-step procedures for common tasks
  - Troubleshooting guides for common issues
  - FAQ section addressing user questions from beta testing
  - Glossary of Upwork and system-specific terminology
- **Maintenance:** Updated within 2 weeks of any feature release

Video Tutorials

Visual learning resources for users who prefer video format:

- **Library:** Minimum 10 tutorial videos (3-5 minutes each) covering core workflows
- **Topics:** Installation, profile setup, client analysis, proposal generation, template management, analytics review
- **Platform:** Hosted on YouTube with closed captions for accessibility
- **Production Quality:** Professional narration, screen recordings with zoom annotations, consistent branding

Contextual Help

In-application assistance available at point of need:

- **Tooltips:** Hover-triggered explanations for UI elements and settings
- **Help Icons:** Clickable "?" icons opening relevant documentation sections
- **Onboarding Tours:** Optional guided tours highlighting key features for first-time users
- **Smart Suggestions:** Context-aware tips based on user behavior (e.g., "Try filtering by score" when viewing many results)

2.6.2 Technical Documentation

API Documentation

For developers and advanced users requiring programmatic access:

- **Format:** OpenAPI 3.0 specification with interactive documentation (Swagger UI)
- **Content:**
  - Endpoint descriptions with request/response examples
  - Authentication methods and token management
  - Rate limiting policies and error codes
  - Webhook configuration for event notifications
- **Code Examples:** Sample code in JavaScript, Python, and cURL

Developer Guide

For contributors and maintainers working on the codebase:

- **Architecture Overview:** System design diagrams, component interactions, data flow
- **Setup Instructions:** Development environment configuration, dependency installation, local testing
- **Coding Standards:** Style guide, naming conventions, commenting practices
- **Testing Procedures:** Unit test, integration test, and end-to-end test guidelines
- **Deployment Process:** Build pipeline, staging environment, production release checklist

Database Schema Documentation

- Entity-Relationship diagrams showing data model
- Collection/table descriptions with field definitions
- Index strategies and query optimization notes
- Migration procedures for schema updates

2.6.3 Administrative Documentation

Administrator Manual

For system administrators managing backend operations:

- **Dashboard Navigation:** Guide to admin control panel features
- **User Management:** Account administration, role assignment, access control
- **System Monitoring:** Interpreting health metrics, setting up alerts, performance tuning
- **Configuration Management:** Adjusting system settings, feature flags, API keys
- **Backup and Recovery:** Procedures for data backup, restoration, disaster recovery

Operations Runbook

Step-by-step procedures for handling operational scenarios:

- Incident response workflows for service outages
- Scaling procedures for handling traffic spikes
- Security incident response protocols
- Routine maintenance task schedules

Maintenance and Support Documentation

- **Version Control Logs:** Detailed changelogs for every release branch.
- **Issue Tracking:** Procedures for documenting bugs via GitHub Issues or Jira.
- **Security Patching:** Protocol for deploying emergency fixes for identified vulnerabilities.

Chapter 3

Functional Requirements

3.1 Introduction

This chapter defines the fundamental actions the Upwork Proposal Assistant must perform. Each requirement is derived from the Use Case Diagram and is categorized by its specific system module. Every functional requirement (FR) is described in terms of its logic, data inputs, internal processing, and resulting outputs.

3.2 FR-001: Automated Client Data Extraction

3.2.1 Description

The system shall programmatically scrape and parse the Document Object Model (DOM) of the active Upwork job page to extract client-related metadata without manual user intervention.

3.2.2 Inputs

- **Active Tab Context:** The URL of the current Upwork job post.
- **DOM Tree:** The raw HTML structure of the page.

3.2.3 Processing

1. The system shall identify specific CSS selectors associated with client metrics.
2. It shall parse the "Hire Rate," "Total Spent," and "Average Hourly Rate."
3. The system shall extract the last 10 feedback entries to identify recurring sentiment patterns.

3.2.4 Outputs

- **Data Object:** A structured JSON object containing parsed metrics.
- **Status Event:** A trigger sent to the UI to update the loading state.

3.3 FR-002: Client Reliability Scoring Engine

3.3.1 Description

This module processes extracted data to generate a normalized 1-100 score representing the client's quality and trustworthiness.

3.3.2 Inputs

- Client metadata from FR-001.
- Admin-defined weightage constants (w_1, w_2, w_3).

3.3.3 Processing

The system shall calculate the Final Score (S) using a weighted linear combination:

$$S = (w_1 \cdot HireRate) + (w_2 \cdot SpendMetric) + (w_3 \cdot SentimentScore)$$

Where:

- **HireRate**: Normalized value between 0 and 1.
- **SpendMetric**: Logarithmic scale of total dollars spent to prevent outlier bias.
- **SentimentScore**: A value derived from keyword matching in client reviews.

3.3.4 Outputs

- **Visual Score:** A numerical value (1-100) displayed in the extension.
- **Traffic Light Status:** Red (Score < 50), Yellow (50–79), Green (≥ 80).

3.4 FR-003: Freelancer Decision Support System

3.4.1 Description

The DSS acts as the analytical "brain" of the extension. It synthesizes data to provide a final verdict on whether the user should spend "Connects" on a job.

3.4.2 Inputs

- **Aggregated Client Score:** The 1-100 value from FR-002.
- **Job Recency:** Time since post.
- **Applicant Count:** Total current proposals submitted.

3.4.3 Processing

1. **Tier 1 Logic:** If (Applicants $>$ 50) AND (Hire Rate $<$ 30%), output ”Skip.”
2. **Tier 2 Logic:** If (Score $>$ 80) AND (Applicants $<$ 20), output ”Strong Apply.”
3. **Tier 3 Logic:** If (Post Time $<$ 30 mins), add ”Urgent Action” badge.

3.4.4 Outputs

- **Recommendation Badge:** Green (Apply), Yellow (Caution), Red (Skip).
- **Probability Estimate:** Percentage likelihood of the client viewing the proposal.

3.5 FR-004: Proposal Template Management & Ranking

3.5.1 Description

Allows management of a library of templates. The system tracks which templates lead to interviews and ranks them by performance.

3.5.2 Inputs

- **Template Body:** Text containing placeholders like [Client_Name].
- **Performance Data:** User-marked success (Interview/Hire).

3.5.3 Processing

1. **Success Velocity (V):** $V = (Interviews/TotalUses) \times 100$.
2. **Auto-Ranking:** Reorder list based on V value.
3. **Parsing:** Replace brackets with data from FR-001.

3.5.4 Outputs

- **Ranked Library:** A searchable list sorted by performance.
- **Performance Icons:** ”Top Performer” or ”Needs Improvement.”

3.6 FR-005: Proposal History and Analytics

3.6.1 Description

Maintains a record of all generated proposals for user auditing and performance tracking.

3.6.2 Inputs

- **Proposal Text:** Final draft content.
- **Timestamp:** Date of generation.

3.6.3 Processing

1. **Aggregation:** Weekly totals of applications submitted.
2. **Trend Calculation:** Compare current week success rate to previous week.

3.6.4 Outputs

- **Dashboard:** Graphical bar and pie charts.
- **Export:** CSV/PDF generation.

3.7 FR-006: User Profile and Skills Configuration

3.7.1 Description

A knowledge base for the AI, storing the freelancer’s specific background and technical strengths.

3.7.2 Inputs

- **Professional Bio:** Detailed experience summary.
- **Skill Tags:** Keywords like [React], [Python], [SEO].

3.7.3 Processing

1. **Mapping:** Compare user skill tags with job-required skills.
2. **Matching %:** Calculate $M = (UserSkills \cap JobSkills)/JobSkills$.

3.8 FR-007: AI Tone and Style Configuration

3.8.1 Description

Controls the linguistic "voice" of the generated proposals.

3.8.2 Inputs

- **Tone Select:** Professional, Friendly, Technical, or Aggressive.
- **Length:** Short, Medium, or Detailed.

3.8.3 Processing

1. **Prompt Engineering:** Append style constraints to the AI API request.

3.9 FR-008: Administrative System Management

3.9.1 Description

Internal tools for the admin to monitor system health and global template quality.

3.9.2 Inputs

- **Global Usage Logs:** API latency and success rates.

3.9.3 Processing

1. **Key Rotation:** Update system-wide API keys for AI services.

3.10 FR-009: Data Synchronization and Persistence

3.10.1 Description

Ensures data is not lost across browser sessions or extension reinstalls.

3.10.2 Inputs

- **Session Data:** Local storage JSON.

3.10.3 Processing

1. **AES-256 Encryption:** Encrypt data before sending to cloud backup.

Chapter 4

Non-Functional Requirements

4.1 Introduction

While functional requirements define what the system does, this chapter defines the quality attributes, operational constraints, and performance benchmarks that ensure the system is professional-grade. In adherence to IEEE Std 830-1984, these requirements are quantifiable, measurable, and verifiable.

4.2 Performance Requirements

4.2.1 Speed and Latency

- **NFR-P-01 (UI Responsiveness):** The extension popup and on-page overlays shall render and become interactive within **1.5 seconds** of the trigger event.
- **NFR-P-02 (Analysis Execution):** The Client Reliability Analysis (FR-001/FR-002) shall complete processing and display results within **3.0 seconds** of data extraction.
- **NFR-P-03 (Generation Latency):** AI-based proposal generation shall deliver a draft to the user in under **5.0 seconds**, excluding external network API delays.

4.2.2 Resource Utilization

- **NFR-P-04 (Memory Management):** The browser extension shall not consume more than **180MB of RAM** at peak processing to ensure no degradation of the browser experience.
- **NFR-P-05 (CPU Overhead):** Background script CPU usage shall remain below **3%** during idle states and **10%** during active generation.

4.3 Reliability and Availability

- **NFR-R-01 (Availability):** The system backend and template database shall maintain **99.9% uptime** (Three Nines), measured on a monthly basis.
- **NFR-R-02 (Fault Tolerance):** In the event of an external AI API failure, the system shall provide a fallback to manual templates within **500ms** without crashing the UI.
- **NFR-R-03 (Data Persistence):** User drafts and history shall be automatically cached locally every **30 seconds** to prevent data loss during browser crashes.

4.4 Security and Privacy

- **NFR-S-01 (Encryption):** All data in transit between the extension and the server shall use **TLS 1.3** encryption over HTTPS.
- **NFR-S-02 (Storage Security):** Sensitive user data, including API keys and credentials, shall be stored using **AES-256** encryption at rest.
- **NFR-S-03 (Anonymization):** During data extraction, no sensitive client personal identifiable information (PII) shall be stored on the system backend.

4.5 Usability and Human Factors

- **NFR-U-01 (Learnability):** A new user shall be able to perform a ”Client Analysis” and ”Proposal Generation” within **2 minutes** of installation without reading a manual.
- **NFR-U-02 (Accessibility):** The user interface shall comply with **WCAG 2.1 Level AA** standards, including high-contrast support and screen-reader compatibility.

Chapter 5

Diagrams and Models

5.1 Context Diagram (Level 0 DFD)

The Context Diagram establishes the boundary between the Upwork Proposal Assistant and the external entities. The system is represented by a central process bubble, highlighting its role as an intermediary and data processor.

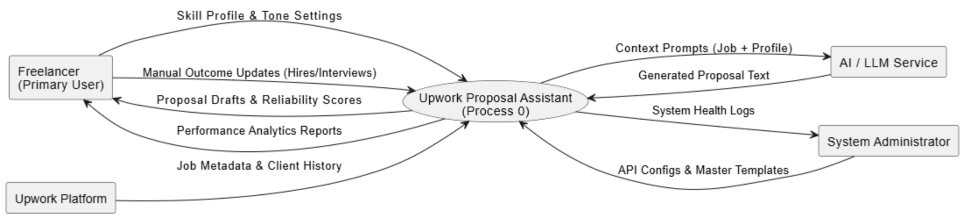
Skill Profile & Tone Settings
Freelancer (Primary User)
Manual Outcome Updates (Hires/Interviews)
Proposal Drafts & Reliability Scores
Performance Analytics Reports
Job Metadata & Client History
Upwork Platform
Upwork Proposal Assistant (Process 0)
Context Prompts (Job + Profile)
Generated Proposal Text
AI / LLM Service
System Health Logs
System Administrator
API Configs & Master Templates

Figure 5.1: Level 0 Context Diagram of the Upwork Proposal Assistant

5.1.1 Entity Interaction Analysis

- **Freelancer:** The primary user who supplies job selections and tone preferences. In return, the system provides real-time reliability scores and proposal drafts.
- **Upwork Platform:** Acts as a data provider. The system extracts job metadata, budget history, and client feedback directly from the site's DOM.
- **AI Service API:** An external computational entity that receives structured prompts from the system and returns high-quality, generated proposal text.
- **System Administrator:** Responsible for managing system configurations, maintaining templates, and monitoring system performance. The administrator does not participate in proposal generation but ensures the reliability, security, and proper operation of the platform.

5.2 Use Case Diagram

The Use Case Diagram defines the functional scope of the system and illustrates how different actors interact with the system's core features.

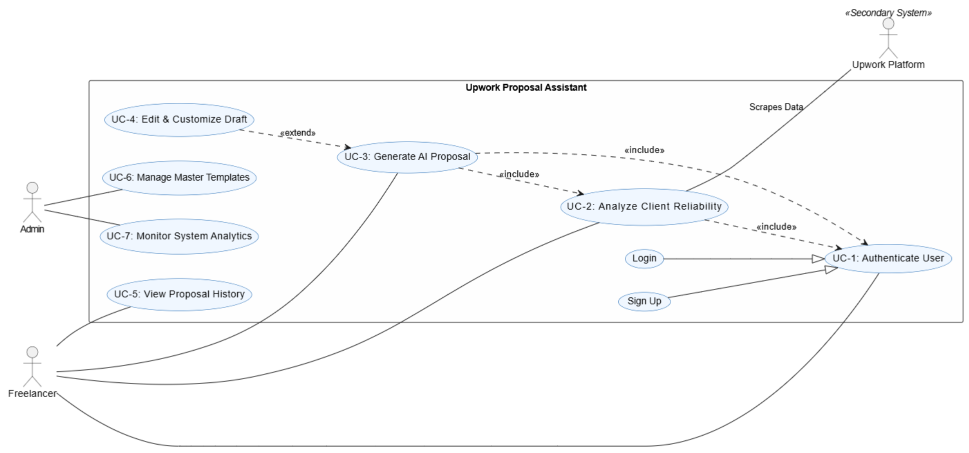
«Secondary System»
Upwork Platform
Upwork Proposal Assistant
Scrapes Data
UC-4: Edit & Customize Draft
«extend»
UC-3: Generate AI Proposal
«include»
«include»
UC-6: Manage Master Templates
Admin
UC-2: Analyze Client Reliability
«include»
UC-7: Monitor System Analytics
Login
UC-1: Authenticate User
UC-5: View Proposal History
Sign Up
Freelancer

Figure 5.2: UML Use Case Diagram showing Actor-System interactions

Chapter 6

References

[1] IEEE Computer Society, ”IEEE Std 830-1984: IEEE Guide to Software Requirements Specifications,” 1984.

[2] Chrome Developers, ”Chrome Extension Manifest V3 Documentation,” Google, 2025.

[3] Upwork Global Inc., ”Upwork API and Terms of Service for Developers,” 2025.

[4] N. Nielsen, ”10 Usability Heuristics for User Interface Design,” 1994.