Namal University, Mianwali

Department of Computer Science

System Design Report

Upwork Proposal Assistant

A Browser Extension for Intelligent Freelance Proposal Management

CSC-225 – Software Engineering

Complex Computing Problem – Milestone 3

Prepared By:

Group Leader: Tayyab Shahzad
Team Member: Kashif
Team Member: Najiba

Supervised By:

Asiya Batool
Department of Computer Science
Namal University, Mianwali

Contents

Chapter 1

Introduction

This chapter provides an overview of the System Design Report for the Upwork Proposal Assistant, outlining the purpose, scope, and organization of this document. It establishes the context for the detailed design specifications that follow.

1.1 Purpose of this Design Document

The primary purpose of this System Design Report is to translate the approved Software Requirements Specification (SRS) into a comprehensive, implementable system design. This document serves as a bridge between requirements analysis and system implementation, providing detailed blueprints for developers, testers, and stakeholders.

1.1.1 Document Objectives

This design document aims to:

- **Translate Requirements into Design:** Convert functional and non-functional requirements from the SRS into concrete design artifacts including architectural components, class structures, and interaction models.
- **Establish Design Decisions:** Document and justify key architectural and design decisions made during the system design phase, ensuring traceability and rationale for future reference.
- **Facilitate Implementation:** Provide developers with clear, unambiguous specifications through visual models (UML diagrams, DFDs) and detailed descriptions that guide the coding process.
- **Enable Validation and Verification:** Establish a foundation for testing by clearly defining system behavior, component interactions, and expected outcomes.
- **Support Stakeholder Communication:** Serve as a communication tool between technical teams, project managers, requirement providers, and academic evaluators.
- **Ensure Quality and Consistency:** Maintain adherence to IEEE software engineering standards (IEEE Std 1016-2009) and industry best practices for software design documentation.

1.1.2 Intended Audience

This document is prepared for multiple stakeholders:

- **Development Team:** Software engineers who will implement the system based on these design specifications.
- **Quality Assurance Team:** Testers who will validate system behavior against design models.
- **Requirement Provider:** Client stakeholder who will verify that the design meets their expectations.
- **Course Instructor and Evaluators:** Academic assessors evaluating the quality and completeness of design work.
- **Project Managers:** Individuals responsible for planning implementation phases and resource allocation.
- **Future Maintainers:** Developers who may need to enhance or modify the system in the future.

1.2 Project Overview

1.2.1 System Context

The **Upwork Proposal Assistant** is a browser extension designed to revolutionize the freelance proposal workflow on the Upwork platform. In the highly competitive landscape of online freelancing, professionals face significant challenges in efficiently 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 15-20 minutes per job application analyzing client history, reading reviews, and writing customized proposals.
- **Information Overload:** Upwork job listings contain extensive data 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.

- **Resource Wastage:** Upwork's "Connect" system charges freelancers for each application, making poor targeting costly.
- **Template Management:** Successful freelancers 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 provides:

1. **Automated Client Analysis:** Extracts and analyzes client profile data including spending history, hire rates, and feedback patterns to generate a normalized Client Reliability Score.
2. **AI-Powered Proposal Generation:** Leverages large language models to create customized proposal drafts tailored to job requirements and client characteristics.
3. **Template Management and Analytics:** Provides sophisticated template organization with performance tracking and automated ranking based on success metrics.
4. **Decision Support System:** Offers intelligent recommendations on whether to apply to specific jobs based on multi-tier analysis.
5. **Historical Analytics:** Maintains comprehensive proposal history with visual analytics for continuous improvement.

1.2.4 Key System Features

Based on the approved SRS, the system delivers nine core functional capabilities:

- FR-001: Automated client data extraction from Upwork job pages
- FR-002: Client reliability scoring with visual indicators
- FR-003: Intelligent recommendation engine (Apply/Skip/Strong Apply)
- FR-004: Template library management with performance ranking
- FR-005: Proposal history tracking and analytics visualization
- FR-006: User profile and skills configuration
- FR-007: AI tone and style customization for proposals
- FR-008: Administrative system management and monitoring
- FR-009: Secure data synchronization across devices

1.3 Scope of Design

1.3.1 Design Coverage

This design document comprehensively covers:

- **Behavioral Models:** Use case diagrams, data flow diagrams (Levels 0-2), sequence diagrams, and activity diagrams representing system behavior and interactions.
- **Structural Models:** Class diagrams and component diagrams illustrating system architecture and organization.
- **Traceability:** Complete mapping between functional requirements and design artifacts ensuring full coverage.
- **User Interface Design:** Paper-based and interactive Figma prototypes demonstrating system interfaces and user workflows.
- **Design Rationale:** Documented assumptions, constraints, and key design decisions with justifications.

1.3.2 Design Boundaries

This document **does not** include:

- Detailed implementation code or algorithms
- Database schema design (physical implementation details)
- Deployment and infrastructure configuration
- Test cases and test plans (covered in separate testing documentation)
- Project management artifacts (schedules, budgets, resource allocation)

1.3.3 System Boundaries

The system operates within the following boundaries:

- **Platform:** Browser extension (Chrome, Firefox, Edge, Brave)
- **Integration:** Upwork platform (web interface only, no mobile)
- **User Base:** Professional freelancers and system administrators
- **Functionality:** Decision support and automation (not autonomous application submission)
- **Compliance:** Upwork Terms of Service, GDPR, browser extension policies

1.4 Design Methodology

1.4.1 Design Approach

The system design follows a **model-driven, object-oriented approach** with emphasis on:

1. **Requirements Traceability:** Every design element traces directly to one or more functional requirements from the SRS, ensuring complete coverage and preventing scope creep.

2. **Iterative Refinement:** Design models were developed iteratively with stakeholder feedback from two formal Requirement Provider meetings:

    - Meeting 1: Paper-based prototype review and feedback
    - Meeting 2: Interactive Figma prototype validation and refinement

3. **Top-Down Decomposition:** System functionality is decomposed hierarchically from high-level context (Level 0 DFD) through detailed process models (Level 2 DFD), ensuring logical organization.

4. **Object-Oriented Principles:** Class design follows SOLID principles with appropriate use of inheritance, composition, and encapsulation.

5. **Separation of Concerns:** Clear separation between presentation layer (browser extension UI), business logic (analysis and generation), and data persistence (storage and synchronization).

1.4.2 Modeling Standards

All diagrams conform to:

- **UML 2.5 Notation:** Use case, sequence, activity, class, and component diagrams follow Unified Modeling Language standards.
- **DFD Conventions:** Data flow diagrams follow Yourdon-DeMarco notation with proper balancing across levels.
- **IEEE Std 1016-2009:** Software Design Description standards for documentation structure and content.

1.4.3 Design Tools

The following tools were utilized:

- **Diagram Creation:** Draw.io, Lucidchart, Visual Paradigm

- **Prototyping:** Figma for interactive UI/UX prototypes
- **Documentation:** LaTeX for professional report formatting
- **Version Control:** GitHub for collaborative design artifact management

1.5 Document Organization

This design document is organized into the following chapters:

- **Chapter 1: Introduction** – Provides context, purpose, and overview of the design document (current chapter).
- **Chapter 2: Design Assumptions and Constraints** – Documents assumptions made during design and constraints that limit design choices.
- **Chapter 3: Key Design Decisions** – Explains and justifies critical architectural and design decisions including DFD decomposition strategy, class relationships, and functionality distribution.
- **Chapter 4: System Design Diagrams** – Presents all behavioral and structural models:
  - Use Case Diagram
  - Data Flow Diagrams (Levels 0-2)
  - Sequence Diagrams
  - Activity Diagrams
  - Class Diagram
  - Component Diagram
- **Chapter 5: GitHub and Figma Links** – Provides access to project repository and interactive prototypes.

1.5.1 Reading Guide

For Developers: Focus on Chapters 3, 4, and 5 for implementation guidance.

For Testers: Review Chapters 4 and 5 to understand expected system behavior.

For Stakeholders: Chapters 1, 2, and 5 provide high-level overview and validation materials.

For Academic Evaluators: All chapters demonstrate comprehensive design methodology and IEEE compliance.

1.6 References

This design document is based on and references the following:

1. **Software Requirements Specification (SRS)** – Upwork Proposal Assistant, Version 1.0, January 5, 2026. This document defines all functional and non-functional requirements upon which this design is based.
2. **IEEE Std 1016-2009** – IEEE Standard for Information Technology—Systems Design—Software Design Descriptions.
3. **IEEE Std 830-1998** – IEEE Recommended Practice for Software Requirements Specifications.
4. **UML 2.5 Specification** – Object Management Group (OMG), Unified Modeling Language standard.
5. **Yourdon-DeMarco DFD Notation** – Structured analysis methodology for data flow diagrams.
6. **Upwork Platform Documentation** – Upwork Terms of Service and Developer Guidelines.
7. **Browser Extension Standards** – Chrome Extension Manifest V3, Firefox WebExtensions API.

1.6.1 External Links

- **SRS Document:** Available in project GitHub repository
- **Requirement Provider Meeting Minutes:** Linked in GitHub repository
- **Design Diagrams:** All diagrams available in `/Design_Diagrams` folder

Chapter 2

Design Assumptions and Constraints

2.1 Design Assumptions

2.1.1 User Roles and Characteristics Assumptions

The system design assumes the existence of two primary user roles:

- **Freelancer (Primary User):**
  The primary users are professional freelancers who actively use the Upwork platform to apply for jobs. It is assumed that users:
  - Have basic familiarity with web browsers and browser extensions.
  - Understand the standard Upwork workflow (viewing jobs, analyzing clients, submitting proposals).
  - Possess at least a minimal professional profile including skills, experience, and portfolio items.
  - Are capable of reviewing AI-generated content and making final decisions manually.
- **Administrator (Secondary/Internal Role):**
  The administrator role is assumed to be limited to internal system maintenance tasks such as monitoring system health, managing AI API keys, and reviewing global usage metrics. Administrators are assumed to have technical expertise and are not typical end-users of the system.

No assumptions are made about user technical expertise beyond basic computer literacy. The system is explicitly designed to assist decision-making rather than replace human judgment, aligning with the SRS system boundary definitions.

2.1.2 Deployment Environment Assumptions

The system design assumes that the Upwork Proposal Assistant is deployed as a browser extension operating in modern web browsers. The following assumptions apply:

- Users access Upwork through supported browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, etc.

- The extension is built according to Manifest V3 specifications, ensuring compatibility with current browser security models.
- The user environment includes:
  - Stable internet connectivity for AI-based proposal generation.
  - Sufficient system resources to support lightweight background processing.
- Backend services are deployed on a cloud-based infrastructure (e.g., AWS or GCP), providing scalability, availability, and geographic distribution.

The design assumes no dependency on operating-system-specific APIs, making the system OS-agnostic at the application level.

2.1.3 Usage Pattern Assumptions

The system is designed under the assumption that:

- Users interact with the extension on-demand, primarily when viewing Upwork job postings.
- Typical usage involves:
  - Viewing a job post
  - Triggering client analysis
  - Reviewing a reliability score
  - Generating and refining a proposal draft
- The system is not used continuously but in short, focused sessions aligned with job application workflows.
- AI-powered proposal generation is invoked selectively rather than excessively to avoid unnecessary API usage.

It is assumed that users prefer fast feedback, minimal UI interruptions, and the ability to manually edit generated proposals before submission.

2.1.4 Data Availability Assumptions

The system design assumes that:

- Required client and job data are available within the Upwork job page DOM and can be extracted using stable selectors.

- The structure of Upwork job pages remains reasonably consistent over time.
- External AI services (e.g., LLM APIs) are available and responsive during normal operation.
- Users provide accurate and up-to-date profile information (skills, experience, preferences) for effective proposal personalization.

The system does not assume access to private Upwork APIs or sensitive user credentials, in compliance with platform policies.

2.2 Design Constraints

2.2.1 Technology Stack Constraints

The design is constrained by the selected technology stack, which includes:

- Browser Extension Technologies: JavaScript, HTML, CSS
- Extension APIs: WebExtensions API (Manifest V3)
- Backend: RESTful APIs implemented using server-side technologies
- AI Integration: External AI service APIs
- Database: Cloud-hosted NoSQL database (e.g., MongoDB)

These constraints limit the system to technologies that are:

- Secure
- Widely supported
- Compatible with browser extension security policies

2.2.2 Platform Constraints (Browser Extension Limitations)

As a browser extension, the system is constrained by:

- Sandboxed execution environments
- Restricted access to system-level resources
- Limited background processing capabilities
- Strict permission declarations required by browsers
- Content script limitations when interacting with third-party websites

These constraints influenced the separation of responsibilities between content scripts, background scripts, and backend services.

2.2.3 Performance Constraints

Performance requirements defined in the SRS impose strict constraints, including:

- UI responsiveness within defined time limits
- Analysis and proposal generation within acceptable latency thresholds
- Limited CPU and memory usage to prevent browser degradation

These constraints guided the decision to offload computationally expensive tasks (e.g., AI processing) to backend services rather than executing them locally.

2.2.4 Security Constraints

Security-related constraints include:

- Mandatory encryption for data in transit (TLS)
- Secure local storage of sensitive data
- No storage of user credentials
- Compliance with data privacy principles and anonymization requirements

These constraints restrict how data is stored, processed, and transmitted, directly influencing backend API and storage design.

2.2.5 Business and Timeline Constraints

The system design is constrained by:

- Academic project timelines
- Limited development resources
- Requirement to strictly follow the approved SRS
- Evaluation criteria defined in Project Milestone 3

These constraints required prioritization of core functionalities and avoidance of scope creep.

Chapter 3

Key Design Decisions

This section explains the rationale behind critical design choices, demonstrating traceability between requirements and design artifacts.

3.1 DFD Decomposition Strategy

3.1.1 Process Breakdown Rationale

The Data Flow Diagrams were decomposed incrementally from Level 0 to Level 2 to manage system complexity. High-level processes were broken down based on:

- Functional responsibility
- Data transformation boundaries
- Logical separation of concerns

For example, proposal generation was decomposed into data extraction, skill matching, prompt construction, AI interaction, and result presentation.

3.1.2 Balancing Approach

Balancing was ensured by maintaining consistency of inputs and outputs across DFD levels. Each child diagram preserved the data flows of its parent process, ensuring:

- No data loss
- No introduction of undocumented processes
- Full traceability to functional requirements

3.2 Class Relationship Decisions

3.2.1 Association, Aggregation, and Composition

- Association was used where objects interact without ownership (e.g., User and Job).

- Aggregation was applied where objects logically belong together but can exist independently (e.g., User and Template).
- Composition was used where lifecycle dependency exists (e.g., Proposal and ProposalDraft).

These choices reflect real-world ownership semantics and reduce coupling.

3.2.2 Inheritance Decisions

Inheritance was used selectively to avoid overgeneralization. Shared attributes and behaviors were abstracted into base classes (e.g., User $\rightarrow$ Freelancer/Admin) only where reuse was meaningful.

3.2.3 Interface Design Decisions

Interfaces were introduced to abstract external services such as AI providers, allowing:

- Vendor independence
- Easier testing
- Future extensibility

3.3 Functionality Distribution

3.3.1 Multiple Sequence Diagrams

Multiple sequence diagrams were created to:

- Avoid overcrowded diagrams
- Clearly represent distinct use cases
- Improve readability and traceability

Each sequence diagram maps to a specific functional requirement.

3.3.2 Multiple Activity Diagrams

Activity diagrams were separated by workflow type (analysis, generation, template management) to:

- Clearly show decision points
- Model parallel activities
- Improve understanding of user-system interaction

3.3.3 Component Separation Rationale

The system was divided into components (UI, analysis engine, AI service, storage) to:

- Reduce coupling
- Improve maintainability
- Support scalability

3.4 Architecture Decisions

3.4.1 Browser Extension Architecture

A three-part extension architecture (content scripts, background scripts, UI components) was chosen to comply with Manifest V3 and ensure security.

3.4.2 Backend API Design Decisions

RESTful APIs were selected for:

- Stateless communication
- Simplicity
- Scalability
- Compatibility with browser-based clients

3.4.3 Database Design Decisions

A NoSQL database was chosen to support:

- Flexible data models
- Rapid schema evolution
- Efficient storage of semi-structured data

Chapter 4

System Design Diagrams

4.1 Use Case Diagram

Diagram Explanation

This diagram represents the functional behavior of the Upwork Proposal Assistant System from the user's perspective. The primary actor is the Freelancer, who can log in, view client profiles, analyze client reliability, generate and edit proposals, save templates, view proposal history, and receive job application recommendations. The system interacts with the Upwork Platform to retrieve job and client information and with an AI Text Service to generate proposals. An Administrator actor is responsible for managing users, templates, and monitoring system usage. The diagram clearly illustrates how different users interact with the system and its external services.

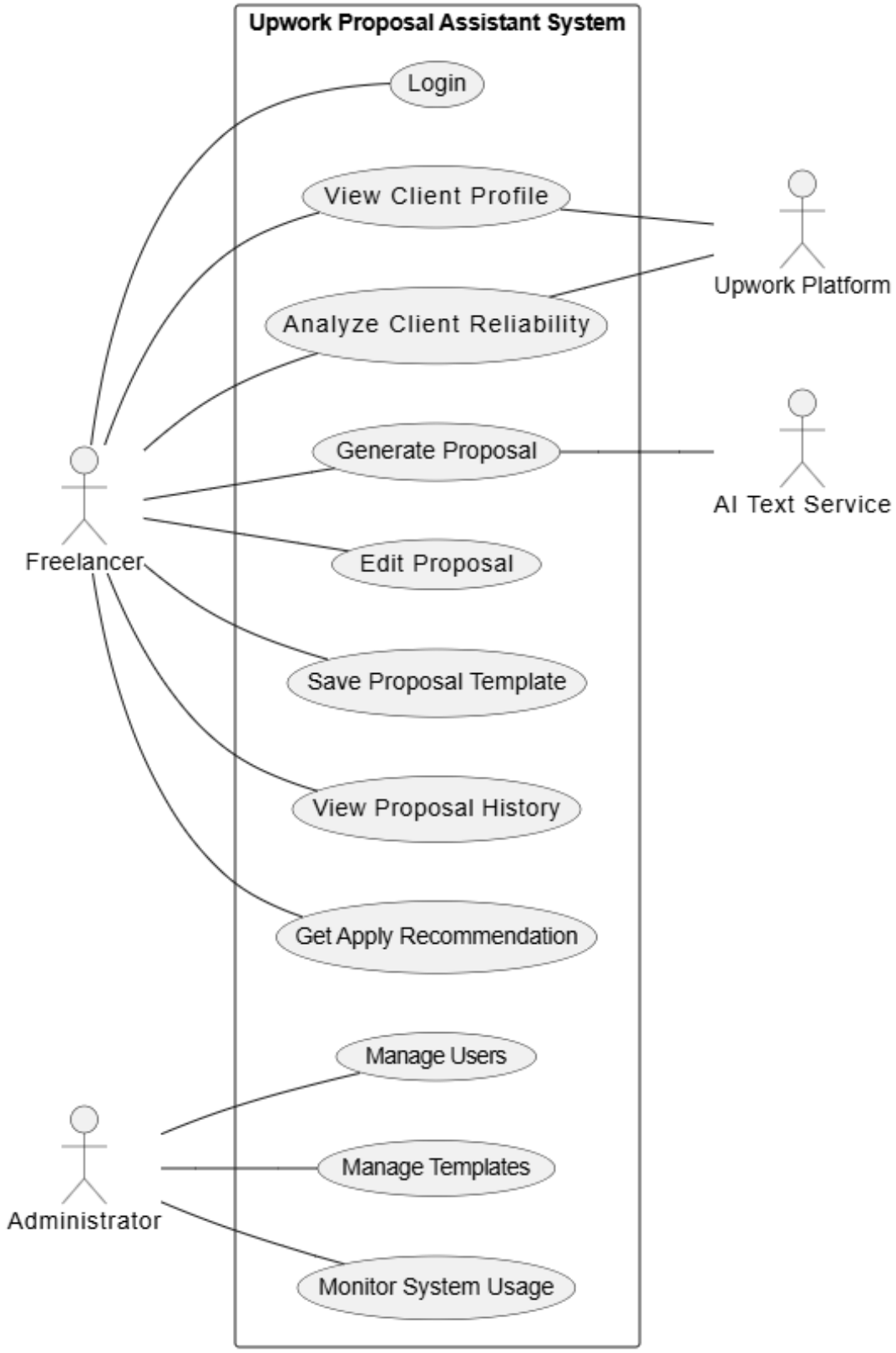

Figure 4.1: Use Case Diagram of the Upwork Proposal Assistant

4.2 Data Flow Diagrams

4.2.1 Context Diagram (Level 0)

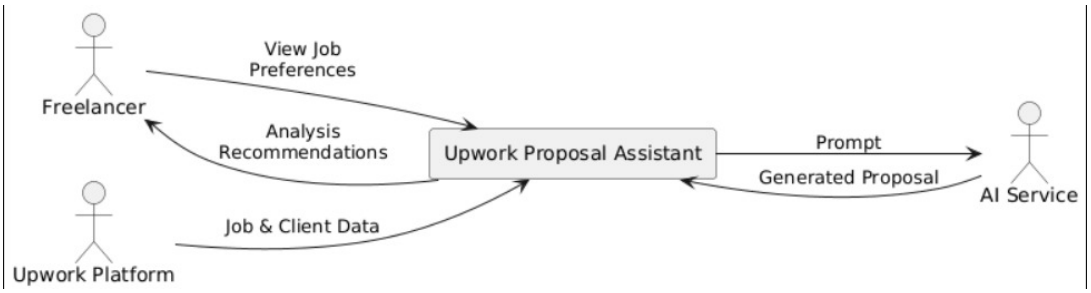

Figure 4.2: Context Diagram (Level 0 DFD)

Diagram Explanation

The Level 0 DFD represents the system as a single process and shows its interaction with external entities such as the freelancer, admin, and the Upwork platform. It provides a high-level overview of data entering and leaving the system.

4.2.2 Level 1 DFD

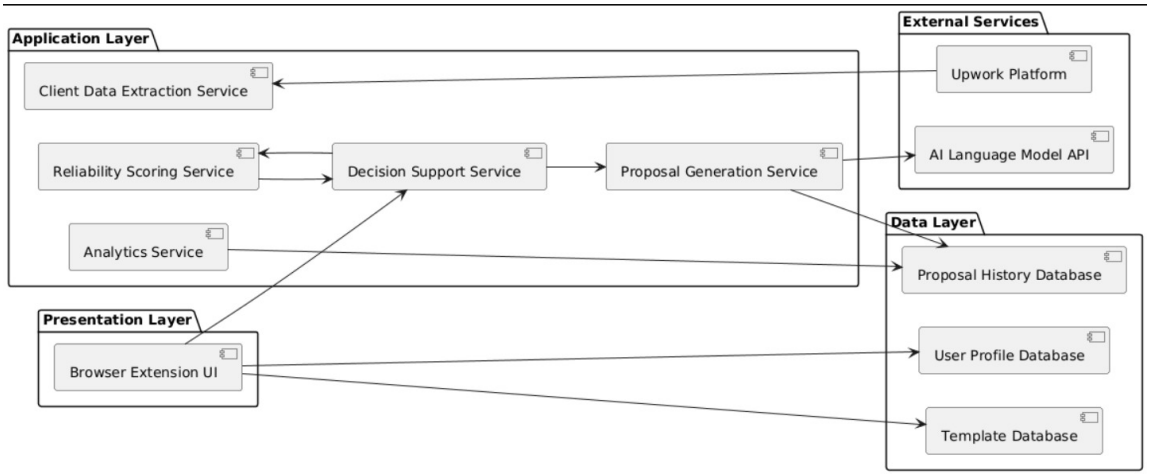

Figure 4.3: Data Flow Diagram (Level 1)

Diagram Explanation

This diagram decomposes the main system into major sub-processes such as client analysis, proposal generation, template handling, and analytics. It explains how data flows between internal processes and data stores.

4.2.3 Level 2 DFD

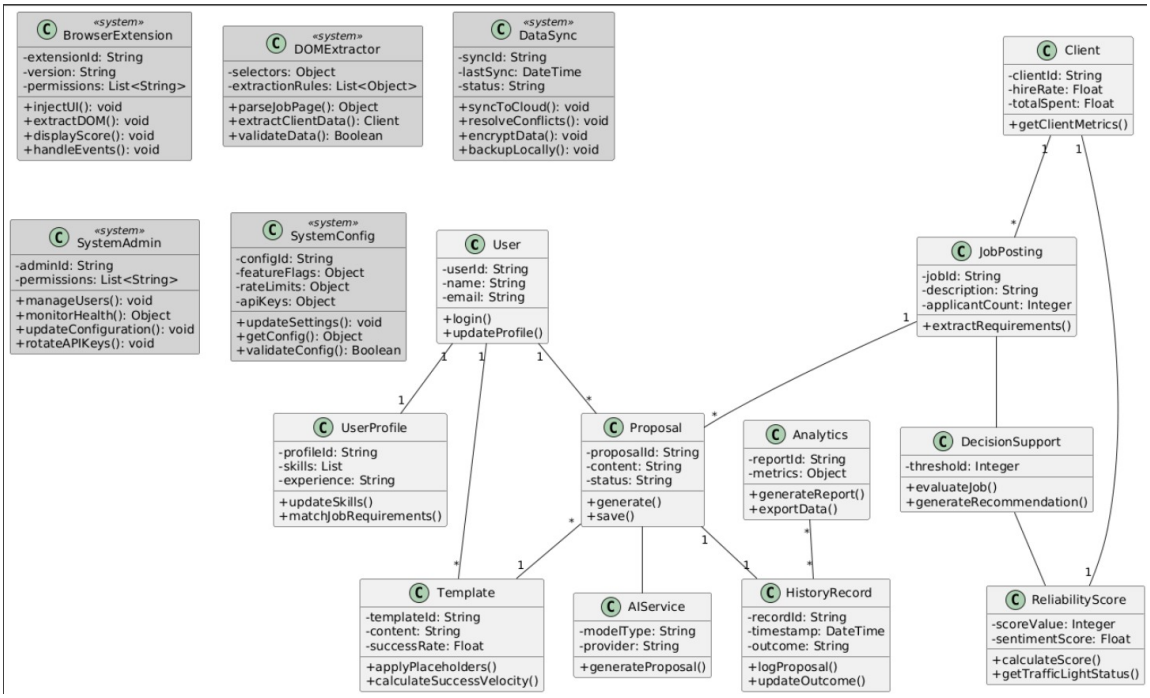

Figure 4.4: Data Flow Diagram (Level 2)

Diagram Explanation

The Level 2 DFD further breaks down critical processes into detailed steps, providing a deeper understanding of internal data transformations and decision-making logic within the system.

4.3 Sequence Diagrams

Diagram Explanation

This sequence diagram demonstrates the time-ordered interaction between system components during the proposal generation process. It shows how user actions trigger system services and how responses are generated.

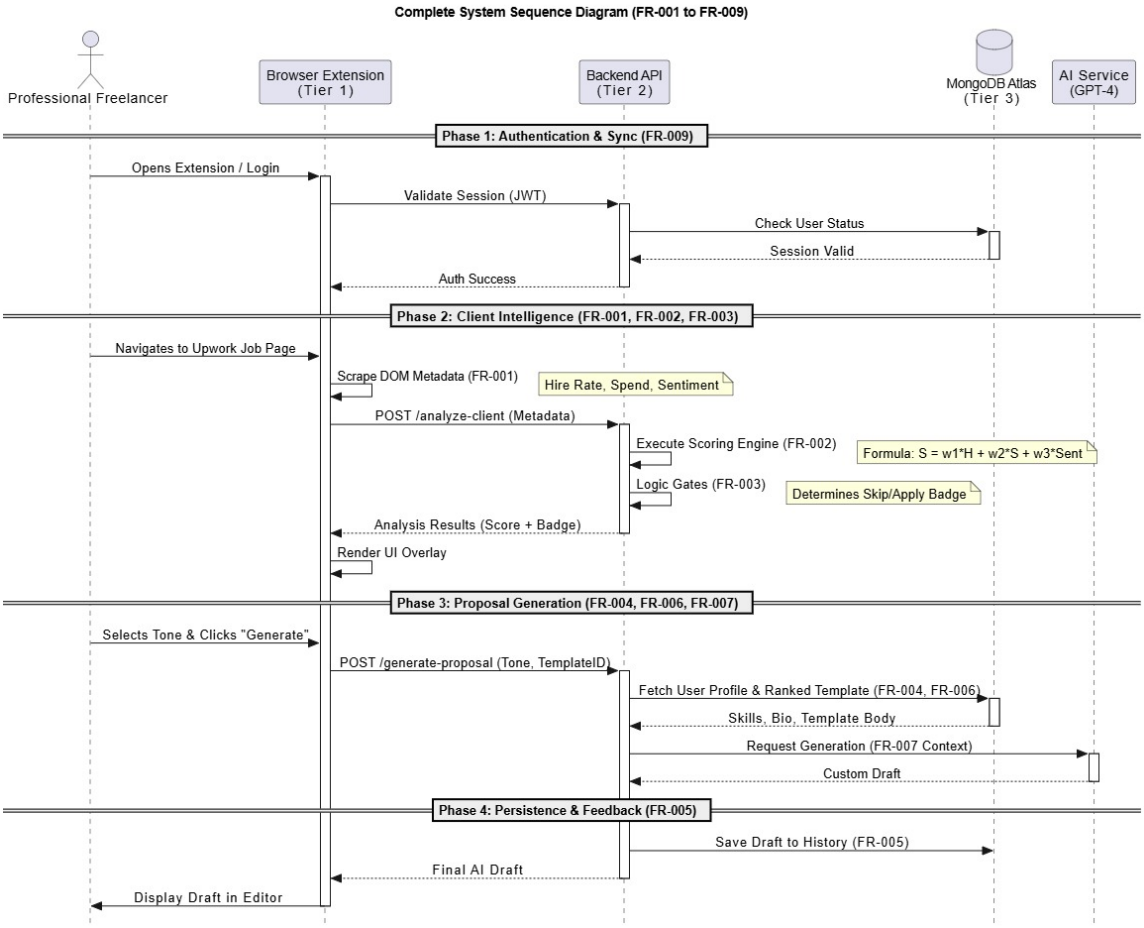

Figure 4.5: Sequence Diagram for Proposal Generation Workflow

4.4 Activity Diagrams

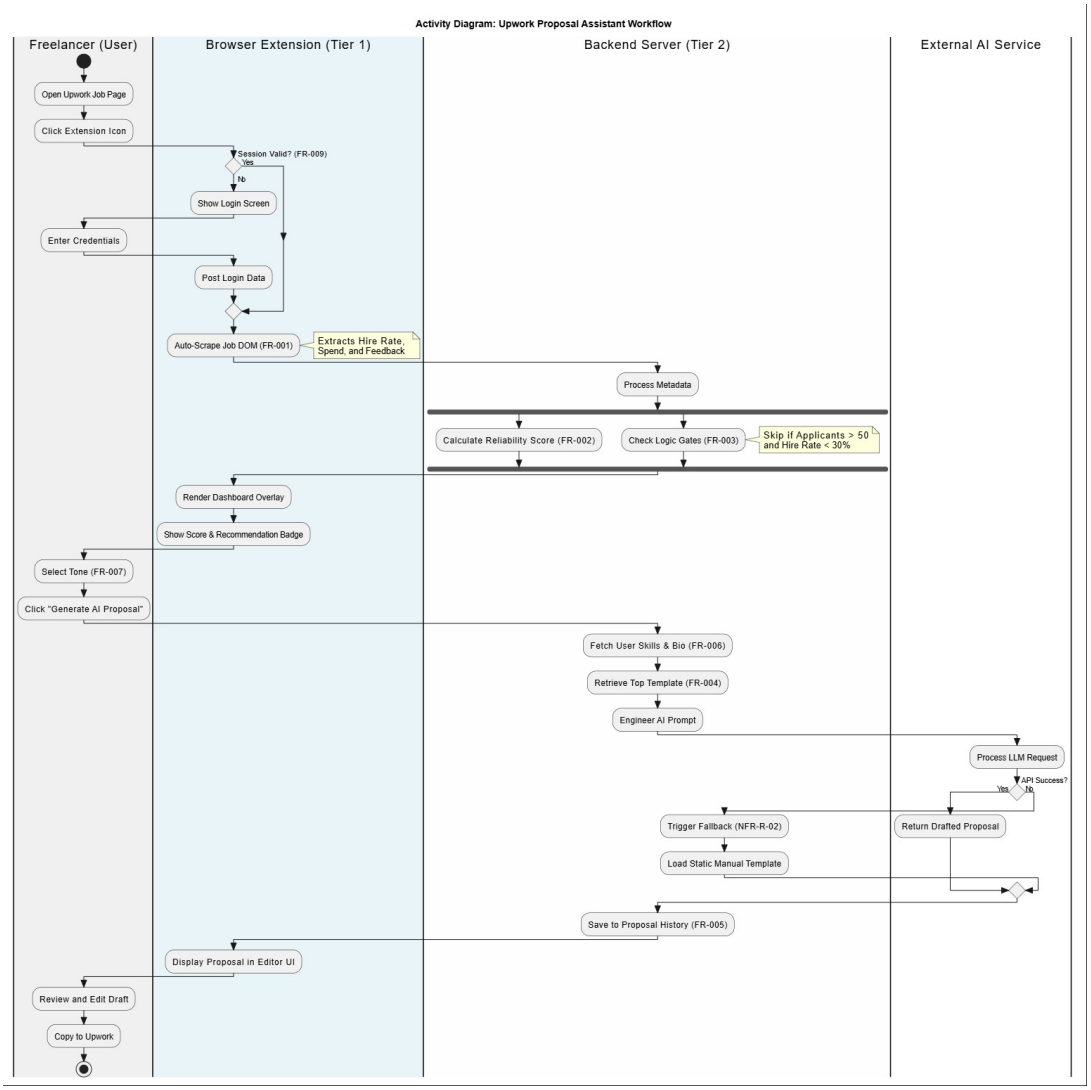

Figure 4.6: Activity Diagram of User Interaction Flow

Diagram Explanation

The activity diagram represents the overall workflow of a freelancer using the system, from job selection to final proposal submission. It highlights decision points and parallel activities.

4.5 Class Diagram

Diagram Explanation

This class diagram defines the static structure of the system by showing classes, attributes, methods, and relationships. It reflects object-oriented design principles used in the system.

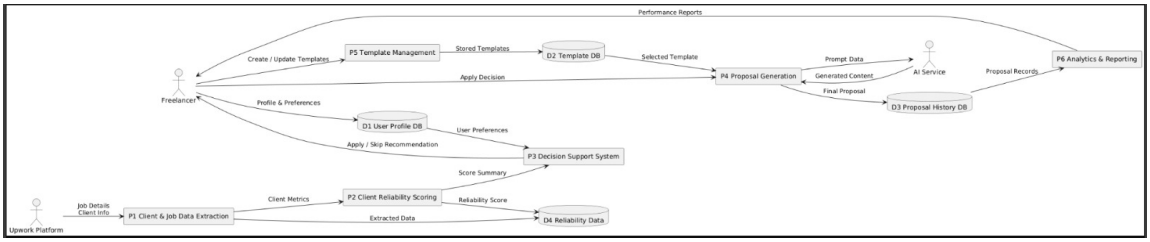

Figure 4.7: Class Diagram of the System

4.6 Component Diagram

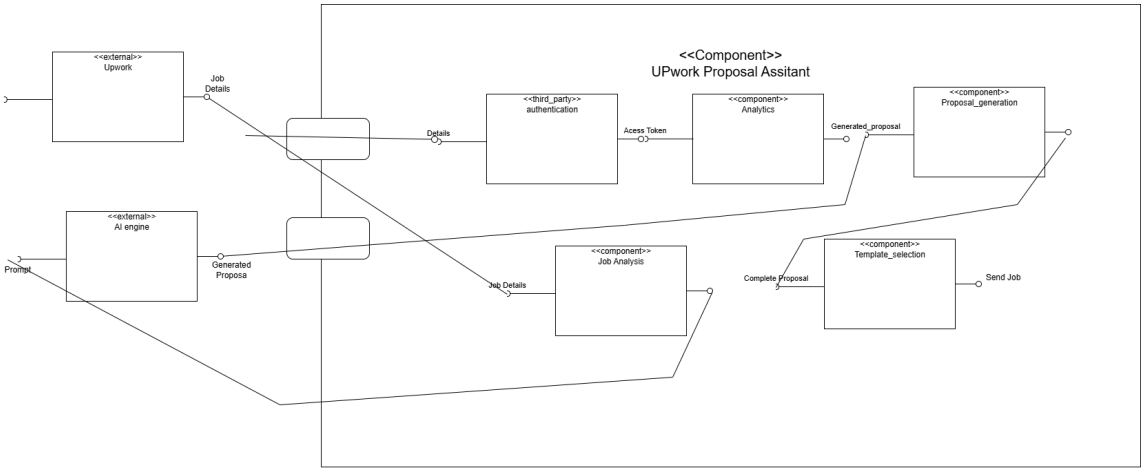

Figure 4.8: Component Diagram of the Upwork Proposal Assistant

Diagram Explanation

This diagram shows the overall architecture of the Upwork Proposal Assistant System and how its major components interact. The system consists of a Browser Extension, a Backend Server, external services, and a Data Layer. The browser extension collects job and client data from the Upwork platform and communicates securely with the backend through a REST API. The backend handles authentication, client analysis, proposal generation, and analytics by coordinating multiple internal engines and an AI service. Data is stored and managed using MongoDB for persistent storage and Redis for fast access and caching. This architecture ensures modularity,

scalability, and efficient data processing.

Chapter 5

GitHub and Figma Links

This chapter provides access to the project repository, interactive prototypes, and related resources.

5.1 GitHub Repository

The complete project, including all design artifacts, diagrams, meeting minutes, and documentation, is maintained in a structured GitHub repository.

5.1.1 Repository Information

- **Repository URL:** https://github.com/iamtayyab-shahzad/Software-engineering-p
- **Repository Type:** Public
- **Last Updated:** January 2026

5.1.2 Key Repository Contents

- **Design_Report/:** Contains this document in PDF format and LaTeX source files
- **Design_Diagrams/:** All UML and DFD diagrams in high-resolution PNG format
- **Traceability/:** Requirements-Design Traceability Matrix
- **Prototypes/:** Paper prototype photos and Figma prototype screenshots
- **Meeting_Minutes/:** Documentation of both RP meetings including videos
- **SRS/:** Approved Software Requirements Specification document
- **README.md:** Project overview and quick navigation guide

5.2 Figma Interactive Prototypes

Two comprehensive interactive prototypes have been developed in Figma to demonstrate the complete system functionality from both user perspectives.

5.2.1 Freelancer-Side Prototype

Purpose: Demonstrates the complete freelancer workflow including job analysis, proposal generation, template management, and analytics.

- **Prototype URL:** https://output-badger-16575296.figma.site/
- **Access:** Public (no login required)
- **Interactive Features:** Clickable navigation, form interactions, modal dialogs

Key Screens Included:

- Extension popup and main dashboard
- Job analysis interface with client scoring
- Recommendation display (Apply/Skip/Strong Apply)
- Proposal generator with AI tone settings
- Template library and template editor
- Analytics dashboard with charts and metrics
- User profile configuration
- Settings and synchronization controls

5.2.2 Admin-Side Prototype

Purpose: Demonstrates administrative functionality including system monitoring, user management, and global template curation.

- **Prototype URL:** https://omen-easel-40982277.figma.site/
- **Access:** Public (no login required)
- **Interactive Features:** Dashboard navigation, data tables, configuration panels

Key Screens Included:

- Admin dashboard with system health metrics
- User management interface
- Global template library and curation tools