UniSupervisor

IEEE 1074 Standard
Software Development Life Cycle Process
Analysis & Design Document

Version 1.0

Date: October 08, 2025

A Comprehensive Project Management System
for Academic Supervision

Table of Contents

1. Executive Summary

UniSupervisor is a comprehensive project management system designed to streamline academic supervision and project coordination between students and supervisors. This document presents the complete analysis and design following the IEEE 1074 Standard for Software Development Life Cycle Processes. The system addresses critical challenges in academic project management, including communication gaps, feedback delays, and administrative overhead.

The platform provides a centralized environment where students can browse available projects, register their interests, submit their work, and receive timely feedback from supervisors. Supervisors benefit from streamlined workflows for posting projects, reviewing submissions, and managing multiple student projects simultaneously. The system incorporates modern software engineering practices, including domain-driven design, service-oriented architecture, and robust data persistence.

2. Software Development Selection Process

2.1 Development Methodology Selection

The UniSupervisor project adopts an Agile development methodology, specifically following Scrum principles with two-week sprints. This methodology was selected based on several key factors: iterative development enables rapid prototyping and continuous refinement based on stakeholder feedback; risk mitigation through incremental releases; and regular stakeholder engagement through sprint reviews and demonstrations.

2.2 Technology Stack Selection

Component	Technology	Justification
Frontend	Java Swing	Rich desktop UI, cross-platform
Backend	Java Domain Logic	Strong typing, enterprise reliability
Database	MySQL	ACID compliance, scalability
Architecture	Layered/Domain-Driven	Separation of concerns
Testing	JUnit	Industry standard for Java
Version Control	Git/GitHub	Distributed VCS, collaboration

3. Initiation Process

3.1 Project Vision and Goals

Vision: To create a seamless digital platform that transforms academic project supervision from a fragmented, email-dependent process into a structured, transparent, and efficient workflow.

Primary Goals:

• Enable students to easily discover and register for projects aligned with their interests

• Provide supervisors with centralized tools to manage multiple student projects

• Facilitate timely submission and feedback cycles

• Maintain comprehensive audit trails of all project activities

• Reduce administrative overhead through automated notifications

3.2 Stakeholder Analysis

Students: Require intuitive interfaces for browsing projects, easy submission mechanisms, and clear visibility into feedback. They need confidence that their work is properly recorded.

Supervisors: Need efficient tools to post projects, review submissions in batch, provide structured feedback, and monitor student progress. Time efficiency is critical.

Administrators: Require system-wide visibility, user management capabilities, and mechanisms to ensure compliance with institutional policies.

4. Project Planning Process

4.1 Project Schedule

The project follows a phased approach spanning 32 weeks, organized into distinct stages that align with the IEEE 1074 process framework. The Gantt chart below illustrates the timeline, dependencies, and parallel workstreams:

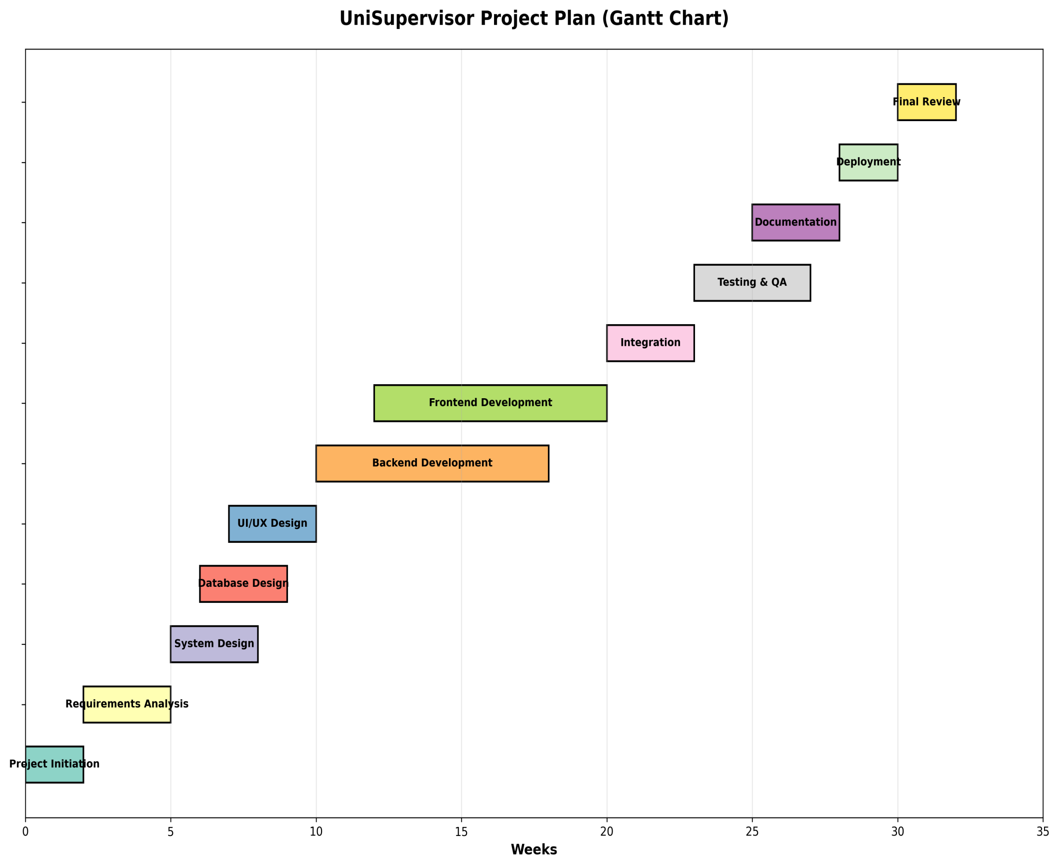

Figure 1: UniSupervisor Project Plan (Gantt Chart)

The schedule incorporates buffers for testing and integration. Database design overlaps with system design to enable early prototyping. Frontend and backend development proceed in parallel after initial architecture is established.

4.2 Resource Planning

Development Team: 3 full-stack developers, 1 database specialist, 1 UI/UX designer

Infrastructure: Development servers, MySQL database instances (dev, staging, production), CI/CD pipeline using GitHub Actions

5. Project Monitoring and Control Process

5.1 Progress Tracking

Sprint Metrics: Velocity tracking, burn-down charts, and completed story points provide quantitative measures of team productivity.

Code Quality Metrics: Automated static analysis, test coverage monitoring (target: >80%), and code review completion rates ensure technical debt remains manageable.

Milestone Reviews: Formal reviews at major milestones ensure alignment with stakeholder expectations.

5.2 Risk Management

Risk	Probability	Impact	Mitigation Strategy
Database performance degradation	Medium	High	Index optimization, caching
Scope creep	High	Medium	Strict change control
Security vulnerabilities	Medium	High	Automated scanning, updates
Developer attrition	Low	High	Documentation, knowledge sharing

6. Pre-Development Processes

6.1 Concept Exploration

Initial concept exploration involved interviews with 15 students and 8 supervisors to understand current pain points. Key findings:

• 68% of students reported delays in receiving feedback (average: 12 days)

• Supervisors spend ~6 hours/week on administrative tasks

• Email-based tracking led to 23% of submissions being temporarily misplaced

• Students desired more transparency into supervisor availability

6.2 Feasibility Analysis

Technical Feasibility: All required technologies are mature and well-documented. The development team possesses necessary expertise.

Economic Feasibility: Development costs estimated at 6 person-months. Expected ROI from reduced administrative burden (200 hours/semester saved).

Operational Feasibility: Deployment requires minimal infrastructure. Training requirements are low given intuitive UI design.

7. Development Processes: Requirements Analysis

7.1 Use Case Model

The use case model captures all primary interactions between actors (students and supervisors) and the UniSupervisor system. Each use case represents a discrete goal that an actor can accomplish:

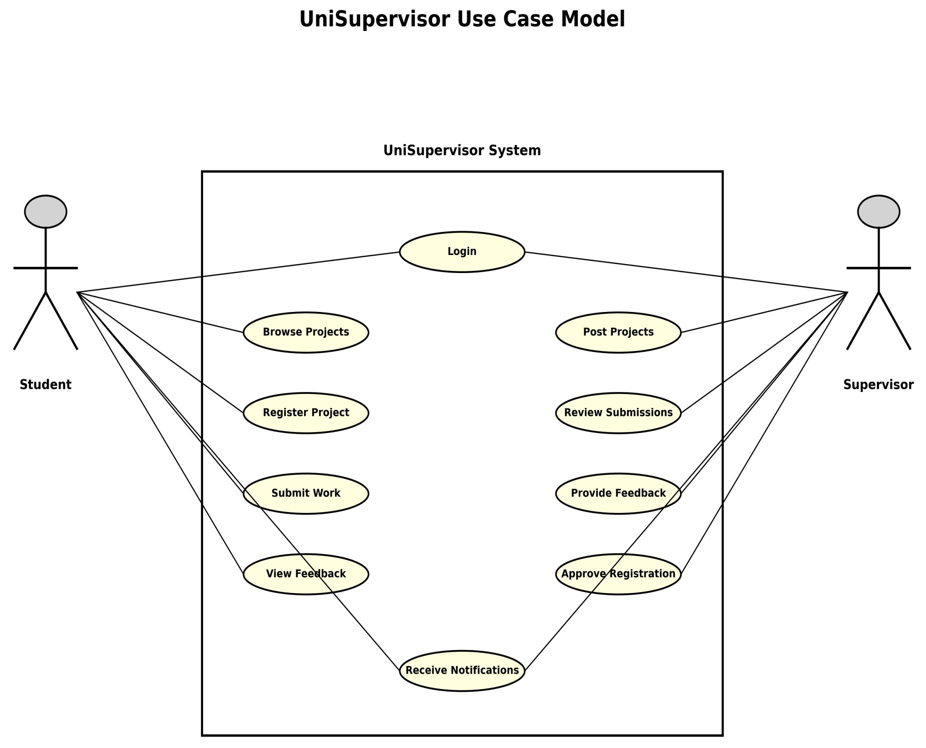

Figure 2: UniSupervisor Use Case Model

7.2 Functional Requirements

Student Requirements:

FR-S1: Browse all available projects with filtering

FR-S2: Register interest in a project, pending approval

FR-S3: Upload work products (PDF, ZIP) up to 50MB

FR-S4: View submission history with timestamps

FR-S5: Receive real-time notifications for feedback

Supervisor Requirements:

FR-V1: Post new projects with title and description

FR-V2: Review and approve/reject registration requests

FR-V3: View all submissions from assigned students

FR-V4: Provide structured feedback with ratings

FR-V5: Download submitted work products for offline review

7.3 Non-Functional Requirements

Category	Requirement	Target Metric
Performance	Page load time	< 2 seconds
Performance	File upload (10MB)	< 5 seconds
Scalability	Concurrent users	500+
Availability	System uptime	99.5%
Security	Password encryption	BCrypt with salt
Usability	User training time	< 15 minutes

8. Development Processes: Design

8.1 System Architecture

UniSupervisor employs a layered architecture that enforces separation of concerns and facilitates independent evolution of different system aspects. The architecture comprises five distinct layers:

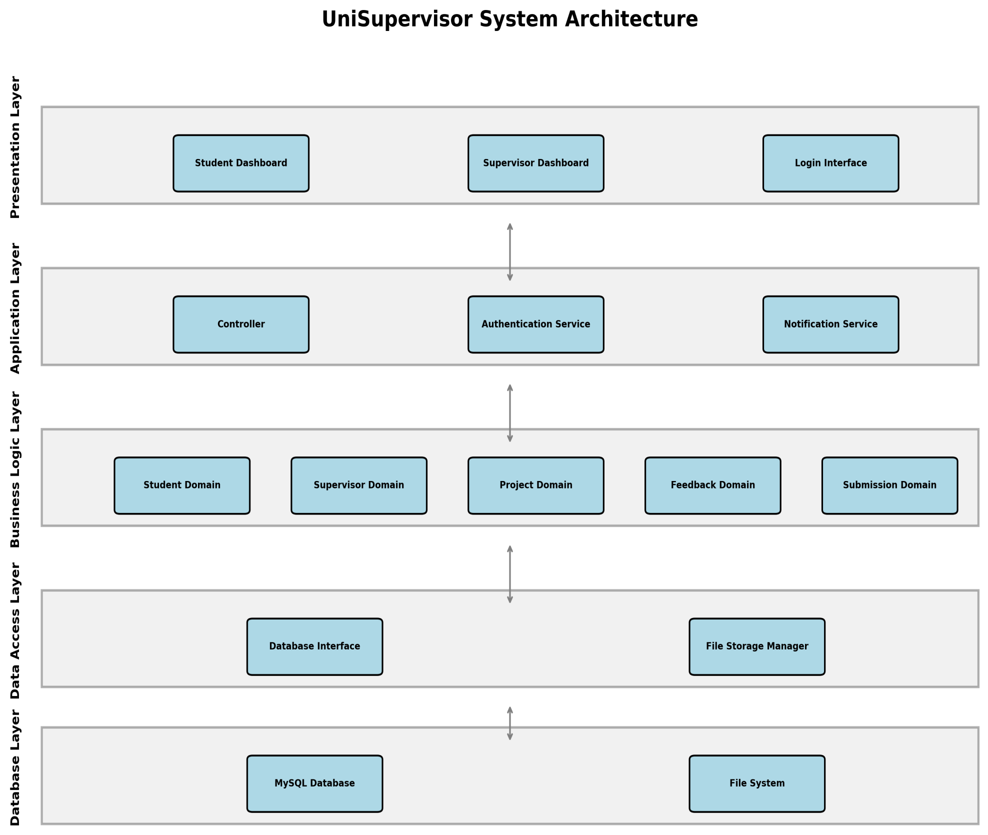

Figure 3: UniSupervisor System Architecture

Presentation Layer: Java Swing interfaces with role-specific dashboards for students and supervisors.

Application Layer: Controller and cross-cutting services (Authentication, Notifications).

Business Logic Layer: Domain-specific logic organized by bounded contexts.

Data Access Layer: Database Interface and File Storage Manager.

Database Layer: MySQL database and file system for binary artifacts.

8.2 Component Diagram

The component diagram illustrates the major structural elements and their dependencies. Components communicate through well-defined interfaces:

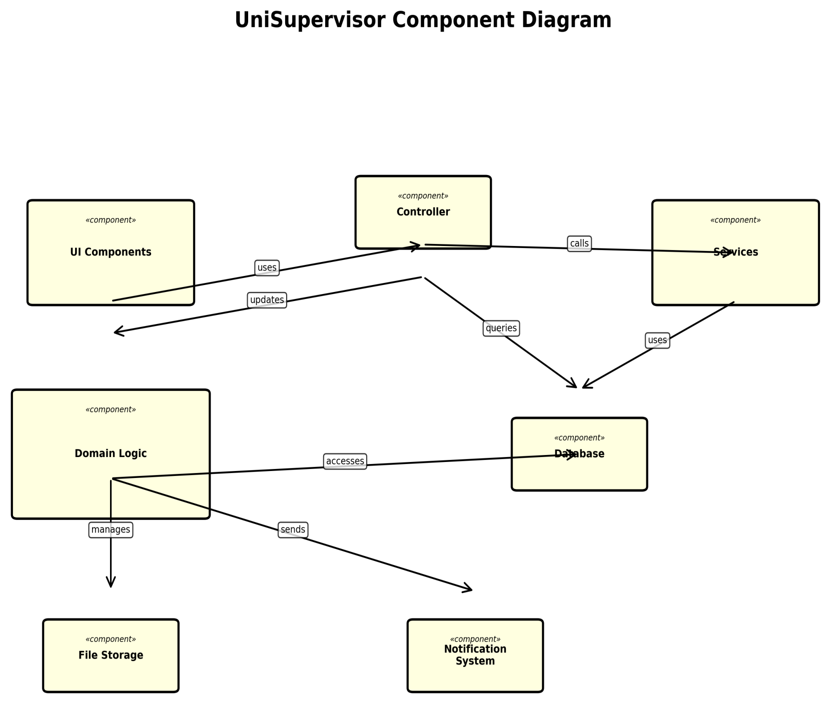

Figure 4: UniSupervisor Component Diagram

8.3 Data Model

The Entity Relationship Diagram captures core domain entities and their relationships. The model is normalized to 3NF to eliminate data redundancy:

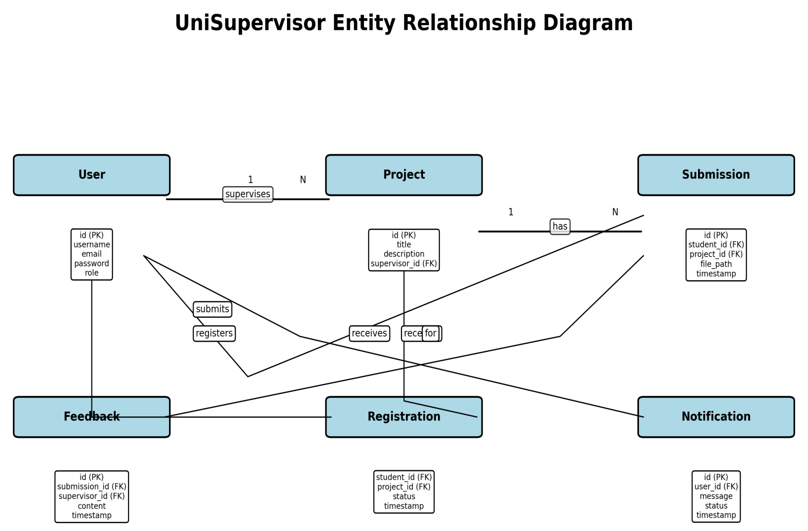

Figure 5: UniSupervisor Entity Relationship Diagram

Key Entities: User (system users with roles), Project (supervisory projects), Submission (student work submissions), Feedback (supervisor feedback), Registration (student-project enrollment), Notification (system-generated notifications).

8.4 Class Diagram

The class diagram presents the object-oriented design, showing domain classes, service classes, and repository interfaces. This design follows SOLID principles and emphasizes loose coupling:

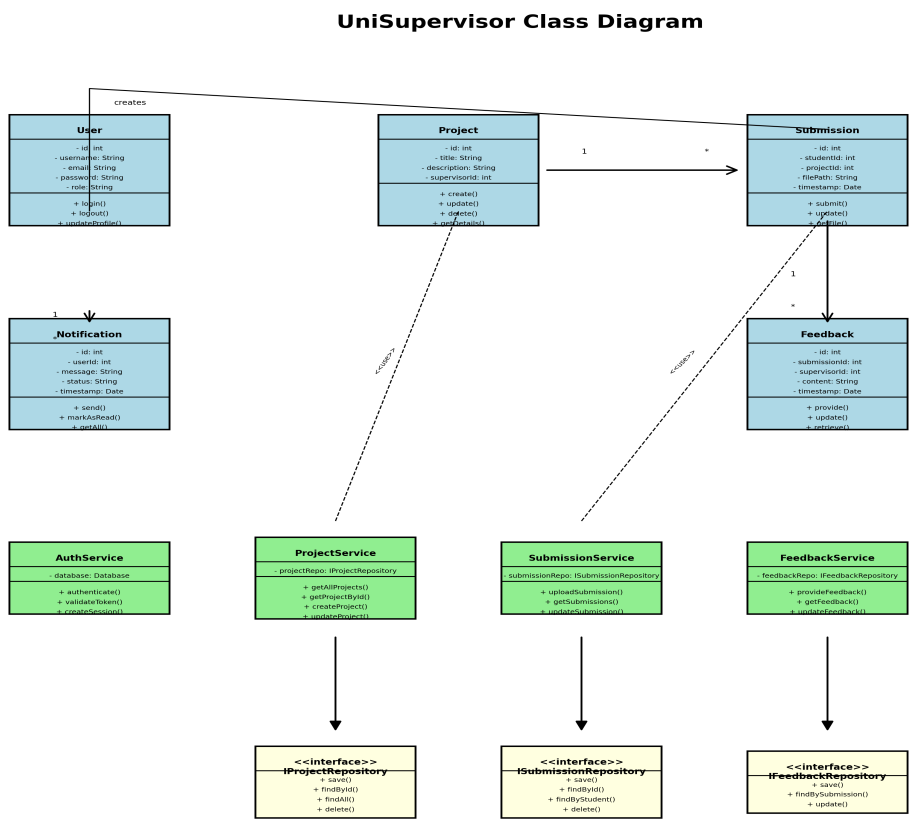

Figure 6: UniSupervisor Class Diagram

Domain Classes: User, Project, Submission, Feedback, and Notification represent core business entities. Each class encapsulates its data and behavior.

Service Classes: AuthService, ProjectService, SubmissionService, and FeedbackService implement application-level operations.

Repository Interfaces: IProjectRepository, ISubmissionRepository, and IFeedbackRepository define contracts for data access, enabling independence from persistence technologies.

8.5 Sequence Diagrams

8.5.1 Login Sequence

The login sequence illustrates the authentication flow between UI, Controller, Authentication Domain, and Database:

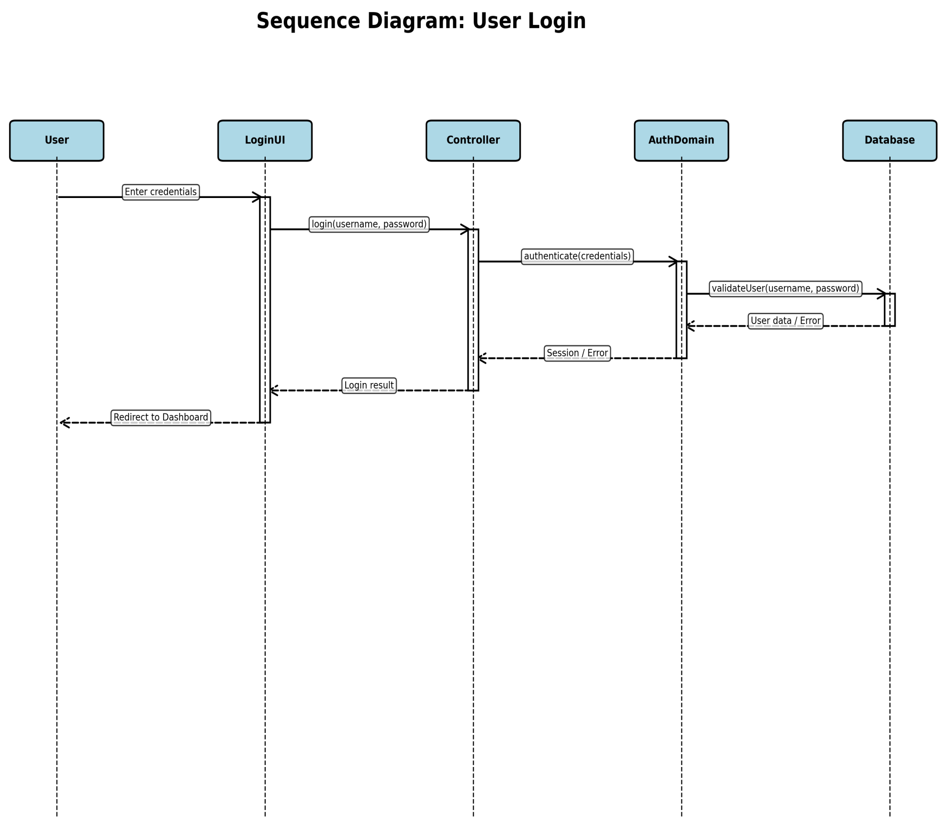

Figure 7: Login Sequence Diagram

The authentication process validates credentials against hashed passwords. Upon success, a session object is created for authorization checks.

8.5.2 Submission and Feedback Sequence

This sequence diagram captures work submission by students and subsequent feedback retrieval:

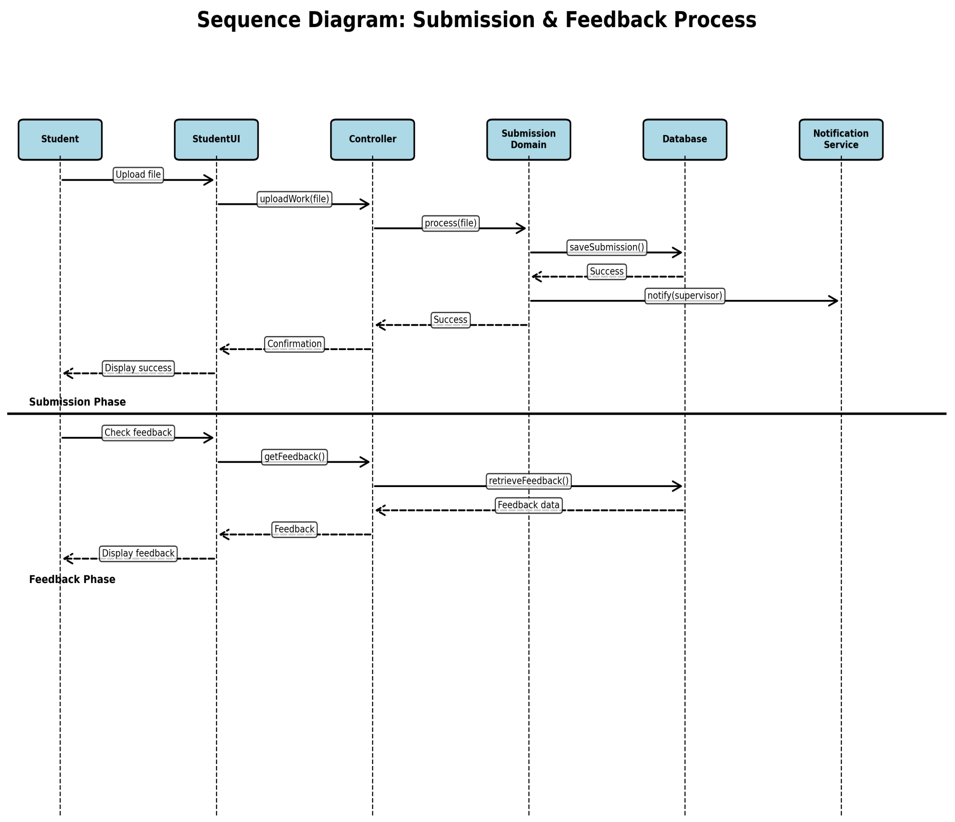

Figure 8: Submission and Feedback Sequence Diagram

File uploads are processed asynchronously. The system validates file types and sizes before accepting. Supervisors are notified via the Notification Service.

8.6 API Design

While the current implementation uses direct method calls, the architecture supports future RESTful API exposure:

Endpoint	Method	Purpose
/auth/login	POST	Authenticate user credentials
/projects	GET	List available projects
/registrations	POST	Register for a project
/submissions	POST	Upload work submission
/submissions/{id}/feedback	GET	Retrieve feedback
/notifications	GET	Get user notifications

8.7 Security Design

Authentication: Passwords hashed using BCrypt with per-user salts. No plaintext passwords stored.

Session Management: Server-side sessions with 30-minute inactivity timeout.

Authorization: Role-based access control (RBAC) enforces permissions on each request.

Input Validation: All inputs validated and sanitized to prevent SQL injection and XSS.

File Upload Security: File type validation, size limits (50MB), and malware scanning.

Audit Logging: All critical operations logged with timestamps and user IDs.

9. Market-Oriented Product Development

9.1 User Journey Map

The user journey map visualizes the student experience from project discovery through feedback receipt, highlighting emotional states, pain points, and opportunities:

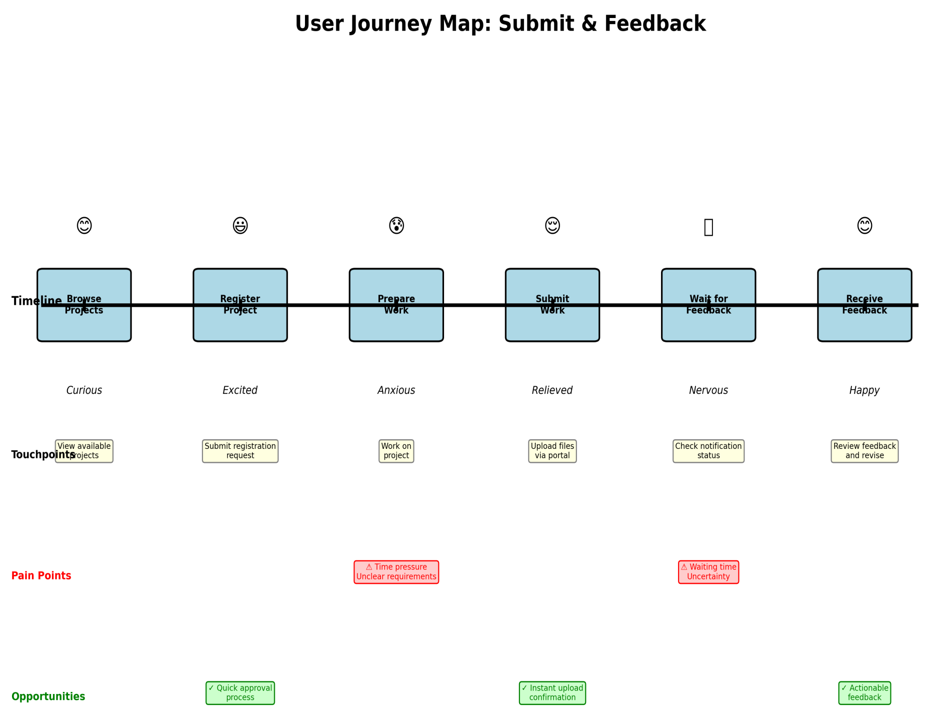

Figure 9: User Journey Map - Submit and Feedback Process

Key Insights:

Anxiety During Preparation: Students experience stress when uncertain about requirements. Mitigation: Detailed project descriptions and example submissions.

Waiting Period Stress: Feedback waiting generates anxiety. Mitigation: SLA notifications and progress indicators.

Feedback Relief: Actionable feedback generates satisfaction. Opportunity: Ensure feedback is constructive and specific.

Instant Confirmations: Quick upload confirmations reduce uncertainty and improve confidence.

9.2 Market Analysis

Target Market: Higher education institutions (universities, colleges) with active student project programs. Primary focus on institutions with 1,000+ students.

Competitive Landscape: Existing solutions include generic LMS platforms (Moodle, Canvas) and project management tools. However, these lack domain-specific features for academic supervision.

Differentiation: UniSupervisor differentiates through academic-specific workflows, supervisor-focused features, and seamless integration with institutional processes.

9.3 Go-to-Market Strategy

Phase 1 - Pilot (Months 1-3): Deploy to single department with 5 supervisors and 50 students. Gather intensive feedback.

Phase 2 - Institutional Rollout (Months 4-9): Expand to entire institution. Provide training and support materials.

Phase 3 - External Marketing (Months 10+): Package as SaaS offering. Develop case studies. Attend academic conferences.

10. Appendices

10.1 Quality Assurance Strategy

Unit Testing: JUnit tests for all business logic. Target: >80% code coverage. Automated in CI/CD pipeline.

Integration Testing: Test database operations, file uploads, and multi-layer interactions.

System Testing: End-to-end testing of complete workflows. Automated using Selenium.

Performance Testing: Load testing with 500 concurrent users. Stress testing to identify breaking points.

Security Testing: Penetration testing, vulnerability scanning (OWASP Top 10).

User Acceptance Testing: Conducted with actual students and supervisors during pilot program.

10.2 Deployment and DevOps

CI/CD: GitHub Actions pipeline automates build, test, and deployment. Every commit triggers automated tests. Successful builds on main branch deploy to staging automatically.

Environment Strategy: Three environments - Development, Staging (mirrors production), and Production. Database migrations managed through version-controlled scripts.

Monitoring: Application logs using SLF4J. Performance metrics through custom dashboards. Alerting for critical errors.

10.3 Maintenance and Support

• Tier 1: User documentation, FAQs, self-service knowledge base

• Tier 2: Email support with 24-hour response SLA for non-critical issues

• Tier 3: Phone support for critical issues with 2-hour response SLA

• Tier 4: Engineering escalation for complex technical issues

Maintenance Windows: Scheduled maintenance every Sunday 2:00-4:00 AM. Users notified 72 hours in advance.

10.4 Future Enhancements

Mobile Applications: Native iOS and Android apps for on-the-go access

Advanced Analytics: Dashboard showing student progress trends and submission patterns

LMS Integration: Single sign-on integration with institutional systems

Plagiarism Detection: Integration with Turnitin or similar services

Video Feedback: Support for supervisors to provide video feedback

Collaborative Features: Enable team projects with multiple students

AI-Assisted Feedback: Suggest feedback templates based on submission content

10.5 Glossary

Term	Definition
Submission	A work product (file) uploaded by a student for supervisor review
Feedback	Structured comments provided by supervisor on a student submission
Registration	The process of enrolling a student in a project under supervisor guidance
Domain	A bounded context in the system encapsulating specific business logic
Repository	An abstraction layer providing data access methods for entities
Session	An authenticated user context maintained server-side for authorization
SLA	Service Level Agreement - guaranteed response/resolution timeframes

--- End of Document ---