Ministry of Higher Education and Scientific Research
University of Jendouba
Higher Institute of Applied Languages and Computer Science of Béja

Réf : LA-GLSI……./2025

Final Year Project Report

For the obtainment of the

BACHELOR'S DEGREE IN COMPUTER SCIENCE

Subject:

Development of a Diagram Generation Platform

Prepared by:

Askri Souhaieb, Mekni Issam

Supervised by:

Academic Supervisor :Mr. Mohamed Naija

Professional Supervisor :Mrs. Askri Ines

Host Organization : Goodwill Engineering

Academic Year: 2024–2025

Dedication

To my dear parents, No words of gratitude can truly do you justice. You have always been my support, my strength, and my first source of encouragement and generosity. Your efforts, sacrifices, and boundless love have been a guiding light that helped me overcome every challenge. I dedicate this work to you as a token of appreciation and gratitude for the values and principles you have instilled in me.

To my sisters, friends, and loved ones, Thank you for standing by me and for your encouraging words in every moment of weakness. You have always been and continue to be a part of this success. You have all my appreciation and love.

Acknowledgments

This project would not have been possible without the support and assistance of several individuals to whom we extend our deepest gratitude. We dedicate this work to them with great appreciation.

We sincerely thank Professor Mohamed Naija, our supervisor, who spared no effort in guiding and advising us throughout the course of this project, offering invaluable support from beginning to end.

We also express our gratitude to Mrs. Askri Ines for his insightful guidance and helpful remarks during the internship period.

Our thanks also go to all our esteemed professors, especially the members of the examination committee who kindly agreed to evaluate our humble work.

Finally, we extend our heartfelt thanks to everyone who contributed, directly or indirectly, to the completion of this project. May Allah bless everyone with success and barakah.

Contents

Dedication 2

Acknowledgments 3

1 Overview 11

1.1 Introduction . . . 11

1.2 Presentation of the Project Context . . . 11

1.2.1 Problem Statement . . . 11

1.2.2 Existing Solutions . . . 11

1.2.3 Proposed Solution . . . 12

1.3 Methodology Agile and Scrum Framework . . . 12

1.4 Modeling Language . . . 12

1.5 Conclusion . . . 12

2 Project Initiation 13

2.1 Introduction . . . 13

2.2 Requirements Analysis . . . 13

2.2.1 System Actors . . . 13

2.2.2 Core Requirements . . . 14

2.2.2.1 Functional Requirements . . . 14

2.2.2.2 Non-Functional Requirements . . . 14

2.3 Project Management . . . 14

2.3.1 Scrum Organization . . . 14

2.3.2 Global Use Case Diagram . . . 15

2.3.3 Sprint Planning . . . 15

2.4 System Architecture . . . 16

2.4.1 Deployment Overview . . . 16

2.4.2 Technology Stack . . . 16

2.5 Conclusion . . . 17

3 Sprint I 18

3.1 Introduction . . . 18

3.2 Sprint Planning and Objectives . . . 18

3.2.1 Primary Objectives . . . 18
3.2.2 Sprint Backlog . . . 18
3.3 Technology Stack Implementation . . . 19
3.3.1 Core Services Overview . . . 19
3.4 Infrastructure Deliverables . . . 19
3.4.1 Database Schema Design . . . 19
3.4.2 Docker Compose Configuration . . . 20
3.4.3 Application Container Configuration . . . 21
3.4.4 Environment Configuration . . . 22
3.5 Sprint Retrospective . . . 22
3.5.1 Key Achievements . . . 22
3.5.2 Challenges Resolved . . . 22
3.5.3 Future Enhancements . . . 23
3.6 Conclusion . . . 23

4 Sprint II 24
4.1 Introduction . . . 24
4.2 Sprint Planning . . . 24
4.2.1 Objectives . . . 24
4.2.2 Backlog Items . . . 24
4.3 System Analysis . . . 25
4.3.1 Use Case Overview . . . 25
4.3.2 Authentication Use Cases . . . 26
4.3.2.1 Core Authentication Scenarios . . . 26
4.4 System Design . . . 27
4.4.1 Authentication Flow . . . 27
4.5 Implementation Results . . . 28
4.5.1 Landing Page . . . 28
4.5.2 Authentication Interface . . . 28
4.6 Sprint Retrospective . . . 29
4.7 Conclusion . . . 29

5 Sprint III 30
5.1 Introduction . . . 30
5.2 Sprint Planning . . . 30
5.2.1 Objectives . . . 30
5.2.2 Sprint Backlog . . . 31
5.3 Analysis and Design . . . 31
5.3.1 Use Case Analysis . . . 31
5.3.2 Key Use Case Specifications . . . 32

5.3.2.1 Create New Project (UC-3.1) . . . 32
5.3.2.2 View Project (UC-3.2) . . . 32
5.3.2.3 Update Project (UC-3.3) . . . 32
5.3.2.4 Delete Project (UC-3.4) . . . 32
5.3.2.5 Download Compressed (UC-3.5) . . . 32
5.4 System Design . . . 32
5.4.1 Sequence Diagrams . . . 32
5.5 Implementation Results . . . 34
5.5.1 User Interface Screenshots . . . 34
5.6 Sprint Retrospective . . . 36
5.6.1 Achievements . . . 36
5.6.2 Areas for Improvement . . . 36
5.7 Conclusion . . . 36

6 Sprint IV 37
6.1 Introduction . . . 37
6.2 Sprint Planning . . . 37
6.2.1 Objectives . . . 37
6.2.2 Sprint Backlog . . . 38
6.3 System Analysis . . . 39
6.3.1 Use Case Overview . . . 39
6.3.2 Core Features . . . 39
6.3.2.1 Diagram Management . . . 39
6.3.2.2 Workspace Management . . . 40
6.4 System Design . . . 41
6.4.1 Process Flow . . . 41
6.4.2 Key Sequence Diagrams . . . 42
6.4.2.1 Diagram Creation Process . . . 42
6.4.2.2 AI-Assisted Editing . . . 42
6.5 Implementation Results . . . 43
6.5.1 Core Interfaces . . . 43
6.5.2 Workspace Environment . . . 44
6.6 Sprint Retrospective . . . 45
6.6.1 Achievements . . . 45
6.6.2 Challenges & Solutions . . . 45
6.7 Conclusion . . . 45

7 Sprint V 46
7.1 Introduction . . . 46
7.2 Sprint Planning . . . 46

7.2.1 Objectives and Backlog 46
7.3 System Analysis 47
7.3.1 Use Case Overview 47
7.3.2 Community Interaction Features 48
7.3.2.1 Key Use Cases Description 48
7.3.3 Profile Management Features 48
7.4 System Design 49
7.4.1 Key Sequence Diagrams 49
7.5 Implementation Results 52
7.5.1 Community Features 52
7.5.2 Profile Management 53
7.6 Sprint Retrospective 53
7.6.1 Achievements 53
7.6.2 Areas for Improvement 53
7.6.3 Future Actions 54
7.7 Conclusion 54

8 General Conclusion 55
8.1 Summary of Achievements 55
8.2 Challenges Faced 55
8.3 Future Perspectives 56

List of Figures

1.1 UML Official Logo 12

2.1 Global Use Case Diagram 15
2.2 Deployment Architecture 16

3.1 Database Entity Relationship Diagram 20

4.1 Sprint II Use Case Diagram 25
4.2 Refined Authentication Use Case 26
4.3 OAuth Authentication Sequence 27
4.4 Responsive Landing Page 28
4.5 OAuth Sign-In Interface 28

5.1 Sprint III Use Case Diagram 31
5.2 Refined Project Management Use Cases 31
5.3 Create Project Sequence 33
5.4 View Project Sequence 33
5.5 Download Project Diagrams Sequence 34
5.6 Home page with project overview 34
5.7 Project creation interface 35
5.8 Project details and management 35
5.9 Project deletion confirmation 35

6.1 Sprint IV Use Case Diagram 39
6.2 Diagram Management Use Cases 39
6.3 Workspace Management Use Cases 40
6.4 Diagram Editing Activity Flow 41
6.5 Create New Diagram Sequence 42
6.6 AI Chat Integration Sequence 42
6.7 Diagram Creation Interface 43
6.8 Interactive Code Editor with Real-time Preview 43
6.9 AI Assistant Integration 44
6.10 Split-View Workspace - Secondary View 44

7.1 Use Case Diagram for Sprint V 47

7.2 Community Interaction Use Cases 48
7.3 Profile Management Use Cases 48
7.4 Community Exploration Flow 49
7.5 Project Commenting Flow 50
7.6 Profile Editing Flow 51
7.7 Community Exploration Interface 52
7.8 Comment System Implementation 52
7.9 Profile Management Interface 53

List of Tables

2.1 Scrum team roles 14
2.2 Sprint planning overview 15
2.3 Core technology stack with icons 16

3.1 Infrastructure services and technologies 19

4.1 Sprint II Backlog 25

5.1 Sprint Backlog 31

6.1 Sprint IV Backlog 38

7.1 Sprint V Product Backlog 47

Chapter 1

Overview of the Project

1.1 Introduction

The evolution of software development has increased the demand for efficient design tools. UML diagrams play a vital role by bridging conceptual design and implementation, improving stakeholder communication and offering standardized documentation.

Traditional approaches to diagramming face challenges due to manual effort and technical complexity. With the rise of AI and LLM technologies, automating diagram generation is now feasible.

This project proposes a platform combining natural language accessibility with the precision of textual UML generation, democratizing the process while maintaining professional standards.

1.2 Presentation of the Project Context

1.2.1 Problem Statement

UML creation using GUI and textual tools presents several challenges:

GUI-Based Tools: Difficult to master, time-consuming, limited collaboration, and weak version control integration.

Textual Tools: Require syntax knowledge (e.g., PlantUML, Mermaid), lack real-time feedback, and pose debugging difficulties.

Integration Issues: Poor workflow integration and limited automation.

1.2.2 Existing Solutions

AI-Based Tools: Use LLM to generate diagrams from user input, but often lack accuracy.

Conversational Tools: Tools like ChatUML provide fast feedback but lack support for complex cases.

Limitations: Existing tools lack full AI integration, offer inconsistent quality, and miss collaborative/community features.

1.2.3 Proposed Solution

Core Idea: The platform combines LLM with PlantUML to interpret natural language and generate accurate diagrams.

Key Features: Real-time validation, collaborative editing, version control support, and a marketplace for sharing templates.

Architecture: Microservices separate LLM, generation, and UI layers for scalability.

Advantages: Professional-quality output, community-oriented design, and user-friendly interfaces.

1.3 Methodology Agile and Scrum Framework

Agile promotes iterative development and adaptability, ideal for evolving AI projects. Scrum enhances Agile through defined roles (Product Owner, Scrum Master, Development Team), events (Planning, Daily, Review, Retrospective), and artifacts (Product Backlog, Sprint Backlog, Increment).

This framework ensures regular inspection, collaboration, and adaptation, supporting continuous improvement throughout the development process.

1.4 Modeling Language

Modeling languages like UML provide standard notations for representing system structure and behavior.

Figure 1.1: UML Official Logo

UML is widely adopted due to its standardization, tooling support, and flexibility across domains, unlike domain-specific languages like BPMN or SysML.

1.5 Conclusion

This project addresses key limitations in UML generation by integrating LLM with PlantUML. The proposed platform enhances usability, collaboration, and automation through a scalable architecture. By leveraging AI, it democratizes diagramming while maintaining professional quality and precision.

Chapter 2

Project Initiation

2.1 Introduction

This project develops a comprehensive PlantUML-based diagramming platform combining individual productivity tools with community collaboration features. The web-based solution enables creating, editing, and sharing PlantUML diagrams while fostering collaborative learning environments.

The platform targets developers, software architects, system designers, and educational institutions requiring efficient technical diagram creation and visual documentation tools.

Key objectives include:

- Developing user-friendly web interface for PlantUML diagram creation
- Implementing robust project and diagram management
- Creating collaborative workspace with AI-powered assistance
- Building community platform for sharing and discovering diagrams
- Ensuring scalable architecture with authentication and administration

The project follows agile development using Scrum framework for iterative development and continuous feedback integration.

2.2 Requirements Analysis

2.2.1 System Actors

Primary Actors:

- **User**: Authenticated individuals with full platform access including workspace management and community interaction

Secondary Actors:

- **AI System**: Intelligent assistant providing code editing assistance
- **PlantUML Server**: External service for diagram rendering

2.2.2 Core Requirements

2.2.2.1 Functional Requirements

- **Authentication**: OAuth via Google/GitHub with cross-device persistence
- **Project Management**: Complete CRUD operations, sharing, and bulk export
- **Workspace**: Interactive editor with real-time rendering and AI assistance
- **Community**: Project exploration, commenting, liking, and forking
- **Profile**: User management and public portfolio display

2.2.2.2 Non-Functional Requirements

- **Performance**: Page loads <3s, diagram rendering <5s, 1000 concurrent users
- **Security**: HTTPS/TLS encryption, OAuth 2.0, input validation, vulnerability protection
- **Usability**: Responsive design, WCAG 2.1 Level AA compliance
- **Reliability**: 99.5

2.3 Project Management

2.3.1 Scrum Organization

Role	Member(s)
Product Owner	Issam Mekni
Scrum Master	Issam Mekni
Development Team	Issam Mekni, Souhaieb Askri

Table 2.1: Scrum team roles

Sprint	Focus Area	Duration
I	Infrastructure Setup	2 weeks
II	Authentication & Landing Page	3 weeks
III	Project Management	3 weeks
IV	Diagram & Project Management	3 weeks
V	Community & Profiles	3 weeks

Table 2.2: Sprint planning overview

2.3.2 Global Use Case Diagram

2.3.3 Sprint Planning

The project spans 14 weeks across 5 focused sprints:

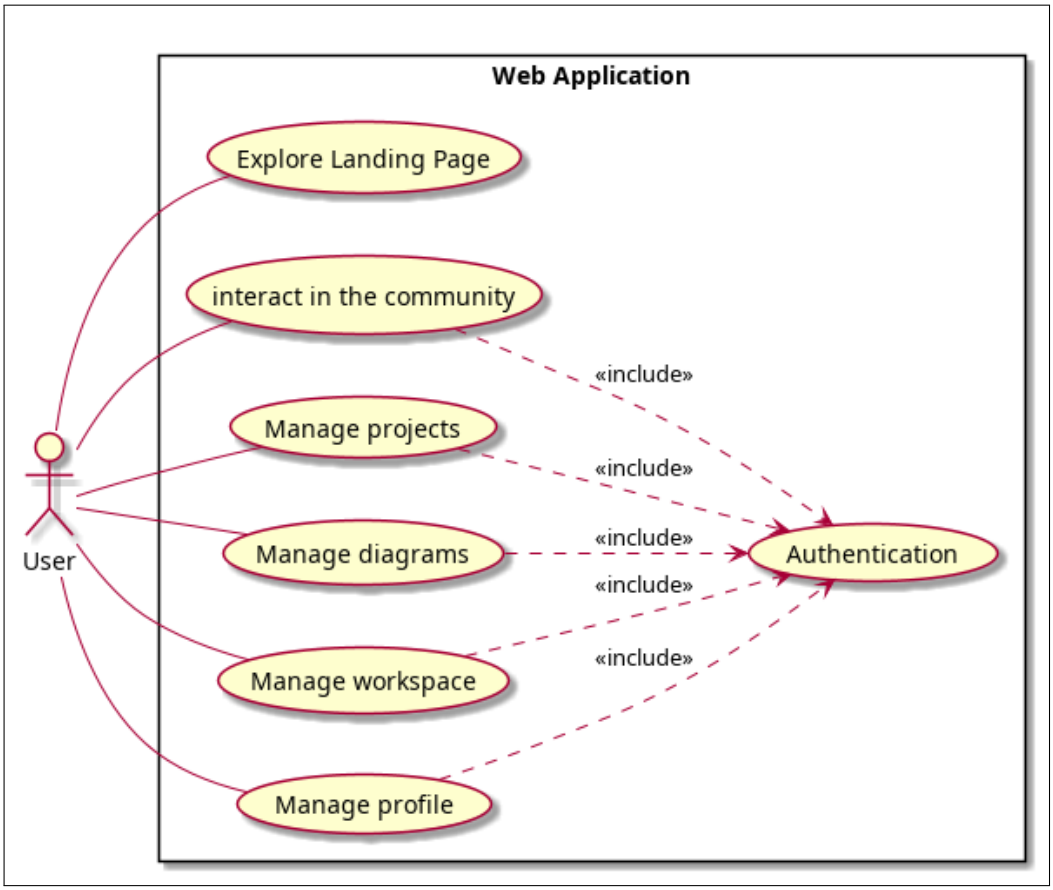

Figure 2.1: Global Use Case Diagram

2.4 System Architecture

2.4.1 Deployment Overview

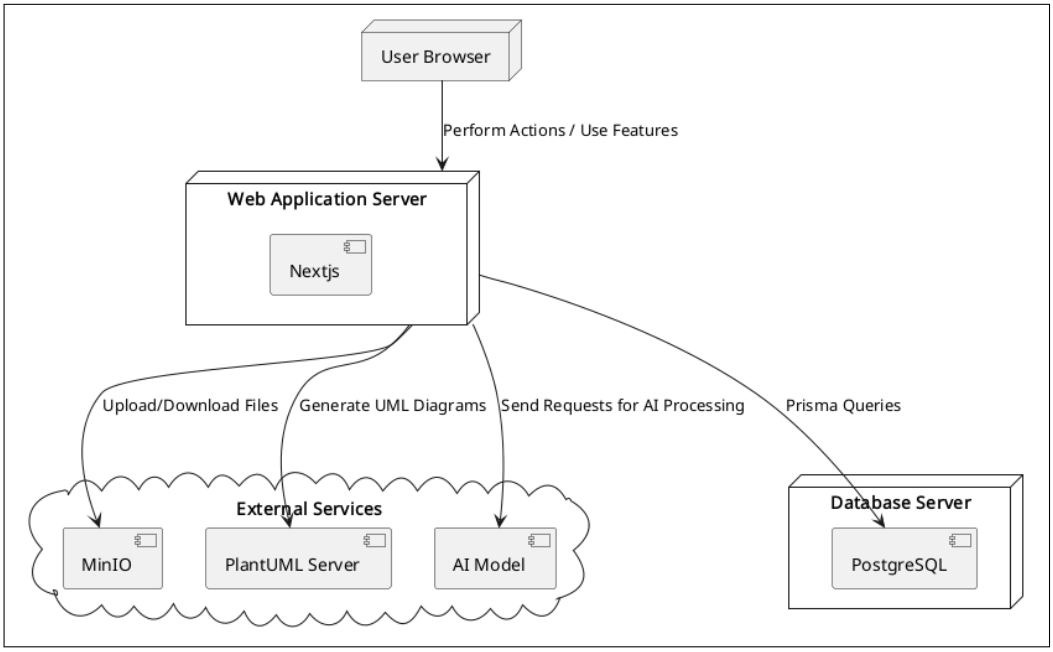

Figure 2.2: Deployment Architecture

2.4.2 Technology Stack

Category	Technologies
Frontend	Next.js, React, TypeScript, Tailwind CSS
Backend	Node.js, NextAuth.js, Prisma ORM
Database	PostgreSQL
AI Integration	LangChain
Deployment	Docker, MinIO
Development	Git, GitHub, VSCodium, Linux

Table 2.3: Core technology stack with icons

2.5 Conclusion

The project initiation phase successfully established a comprehensive foundation through systematic requirement analysis, stakeholder identification, and strategic Scrum-based planning. The structured approach ensures focused development on core functionality while maintaining flexibility for future enhancements.

Key achievements include clear actor identification, comprehensive requirement specification, prioritized product backlog, realistic sprint planning, and established project management framework. This foundation positions the project for successful progression through technical architecture design and implementation phases.

Chapter 3

Study and Implementation of Sprint I: Infrastructure Setup

3.1 Introduction

Sprint I establishes a robust, scalable infrastructure foundation using modern DevOps practices and Docker containerization. This sprint creates the development environment and core services supporting the entire application ecosystem, ensuring consistency across environments and facilitating deployment.

3.2 Sprint Planning and Objectives

3.2.1 Primary Objectives

- Establish containerized development environment using Docker Compose
- Set up PostgreSQL database with data persistence
- Configure MinIO object storage for file management
- Deploy PlantUML server for automated diagram generation
- Create Next.js application with Prisma ORM integration
- Implement NextAuth.js authentication with Google OAuth
- Configure secure environment variables and service networking

3.2.2 Sprint Backlog

1. Database Setup (PostgreSQL with persistent storage)
2. Object Storage (MinIO S3-compatible service)

3. Diagram Service (PlantUML server deployment)
4. Web Application (Next.js with TypeScript/Tailwind CSS)
5. ORM Configuration (Prisma with PostgreSQL)
6. Environment Configuration and Integration Testing

3.3 Technology Stack Implementation

3.3.1 Core Services Overview

Service	Technology	Purpose
Database	PostgreSQL 16	ACID-compliant relational database with persistent storage
Object Storage	MinIO	S3-compatible file storage with web management interface
Diagram Service	PlantUML Server	Automated diagram generation from markup
Web Framework	Next.js 14	Full-stack React framework with TypeScript support
ORM	Prisma	Type-safe database access layer
Containerization	Docker Compose	Service orchestration and environment consistency

Table 3.1: Infrastructure services and technologies

3.4 Infrastructure Deliverables

3.4.1 Database Schema Design

The database implements a normalized schema supporting core application functionality:

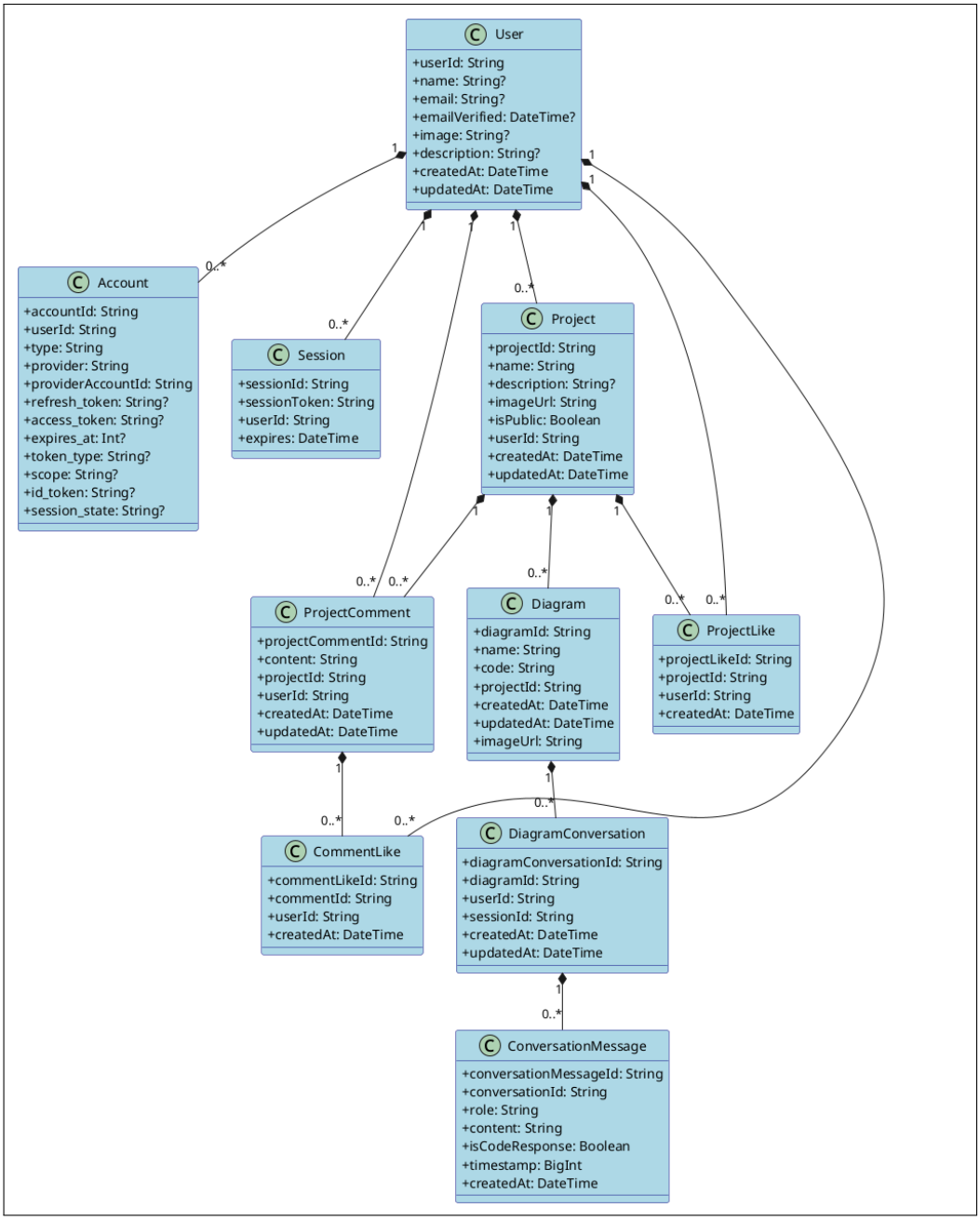

Figure 3.1: Database Entity Relationship Diagram

3.4.2 Docker Compose Configuration

The complete infrastructure is orchestrated through Docker Compose:

Listing 3.1: Docker Compose Services Configuration

```
services:
  postgres:
    image: postgres:16
    container_name: my_postgres
    environment:
      POSTGRES_USER: user
```

```
      POSTGRES_PASSWORD: password
      POSTGRES_DB: database
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  minio:
    image: minio/minio
    container_name: minio
    ports:
      - "9000:9000"   # API
      - "9001:9001"   # Web UI
    volumes:
      - ./minio-data:/data
    environment:
      MINIO_ROOT_USER: minioadmin
      MINIO_ROOT_PASSWORD: minioadmin
    command: server /data --console-address ":9001"

  plantuml:
    image: plantuml/plantuml-server
    container_name: plantuml_server
    ports:
      - "3030:8080"
    restart: unless-stopped

  web-app:
    build: .
    container_name: nextjs_app
    ports:
      - "3000:3000"
    depends_on:
      - postgres
      - minio
      - plantuml
    environment:
      - DATABASE_URL=postgresql://user:password@postgres:5432/database
      - PLANTUML_SERVER=http://plantuml:8080
```

3.4.3 Application Container Configuration

Listing 3.2: Next.js Application Dockerfile

```
FROM node:20-alpine

```

```
WORKDIR /app

COPY package*.json ./
COPY prisma ./prisma/

RUN npm install
RUN npx prisma generate

COPY . .

EXPOSE 3000

CMD ["npm", "run", "dev"]
```

3.4.4 Environment Configuration

Essential environment variables for secure operation:

Listing 3.3: Environment Variables

```
GOOGLE_CLIENT_SECRET=**********************
GOOGLE_CLIENT_ID=**********************
NEXTAUTH_URL=http://localhost:3000
NEXTAUTH_SECRET="**********************"
DATABASE_URL=postgresql://user:password@postgres:5432/database
PLANTUML_SERVER=http://localhost:3030
GEMINI_API_KEY=**********************
```

3.5 Sprint Retrospective

3.5.1 Key Achievements

- Complete containerized infrastructure with service integration
- Working authentication system with Google OAuth
- Database schema design and Prisma ORM integration
- Secure environment configuration and networking
- Comprehensive documentation and testing

3.5.2 Challenges Resolved

- **Port Conflicts**: Resolved through careful port mapping documentation

- **Container Networking**: Fixed connection strings and service naming
- **Security Management**: Implemented secure environment variable handling
- **Schema Synchronization**: Coordinated Prisma migrations in containers

3.5.3 Future Enhancements

- Health checks implementation for all services
- Logging and monitoring solutions integration
- Automated backup systems for database
- SSL/TLS configuration for production readiness
- CI/CD pipeline preparation

3.6 Conclusion

Sprint I successfully established a comprehensive, production-ready development infrastructure using containerization best practices. The integration of PostgreSQL, MinIO, PlantUML, and Next.js creates a robust foundation supporting all planned application features.

The infrastructure demonstrates modern DevOps practices with proper security, scalability, and documentation. This solid foundation enables the development team to focus on business logic and user features in subsequent sprints, ensuring both current development needs and future scaling requirements are met.

Chapter 4

Study and Implementation of Sprint II: Authentication & Landing Page

4.1 Introduction

Sprint II focuses on developing the authentication system and landing page using NextAuth.js and Prisma ORM. This sprint establishes essential user management capabilities and creates an intuitive entry point for the application, building upon the foundation from previous iterations.

4.2 Sprint Planning

4.2.1 Objectives

- Implement secure OAuth authentication (Google, GitHub)
- Develop cross-device session management
- Create responsive landing page
- Establish database schema with Prisma ORM
- Ensure seamless sign-in/sign-out functionality

4.2.2 Backlog Items

The MoSCoW prioritization method categorizes requirements into four levels: **Must have** (critical features essential for project success), **Should have** (important features that add significant value but aren't critical), **Could have** (nice-to-have features that can be included if time and resources allow), and **Won't have** (features explicitly excluded from current scope).

This method helps teams focus on delivering the most valuable features first while maintaining clear expectations about what will and won't be included in each release.

Table 4.1: Sprint II Backlog

ID	User Story	Priority	Status
1	As a user, I want to sign in with Google account	M	Completed
2	As a user, I want to sign in with GitHub account	M	Completed
3	As a user, I want to stay authenticated across devices	M	Completed
4	As a user, I want to sign out from my account	S	Completed
5	As a visitor, I want to explore the landing page	M	Completed

4.3 System Analysis

4.3.1 Use Case Overview

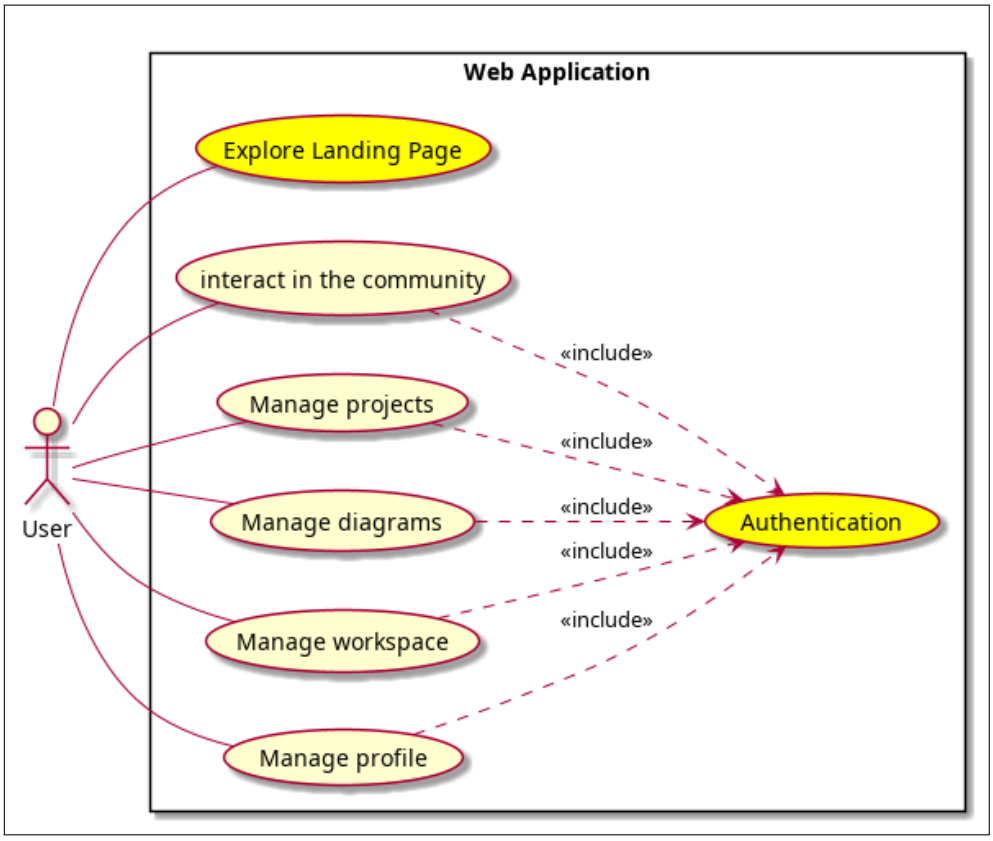

Figure 4.1: Sprint II Use Case Diagram

4.3.2 Authentication Use Cases

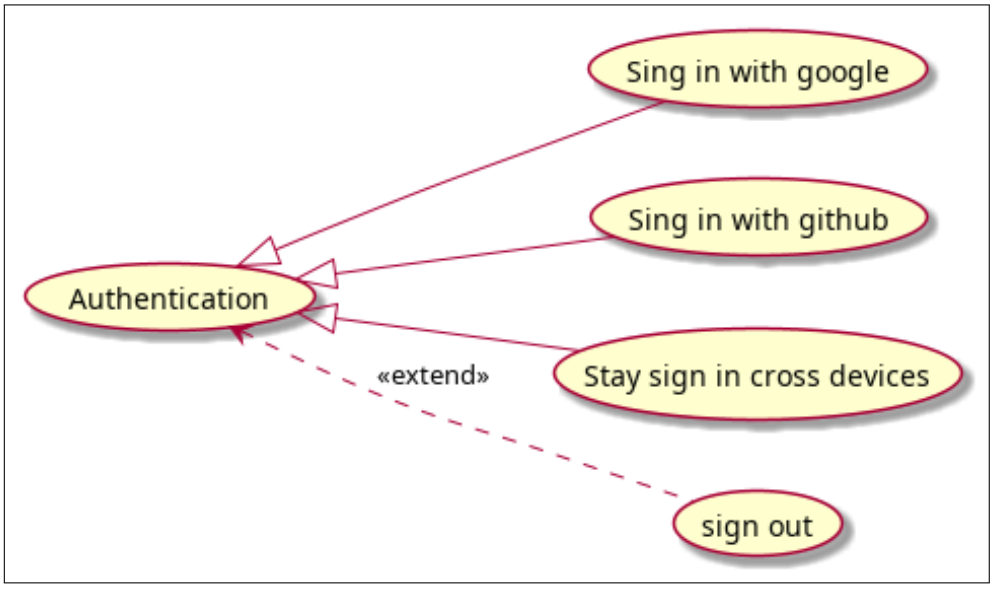

Figure 4.2: Refined Authentication Use Case

4.3.2.1 Core Authentication Scenarios

OAuth Sign-In Process:

1. User clicks OAuth provider button (Google/GitHub)
2. System redirects to provider's authorization page
3. User authorizes application access
4. Provider returns authorization code
5. System validates and creates user session
6. User is redirected to dashboard

Cross-Device Authentication: Session persistence is maintained through secure tokens allowing users to access the application across multiple devices without re-authentication, with automatic session validation and expiration handling.

Secure Sign-Out: Session termination involves token invalidation, cookie clearing, and secure redirection to the landing page.

4.4 System Design

4.4.1 Authentication Flow

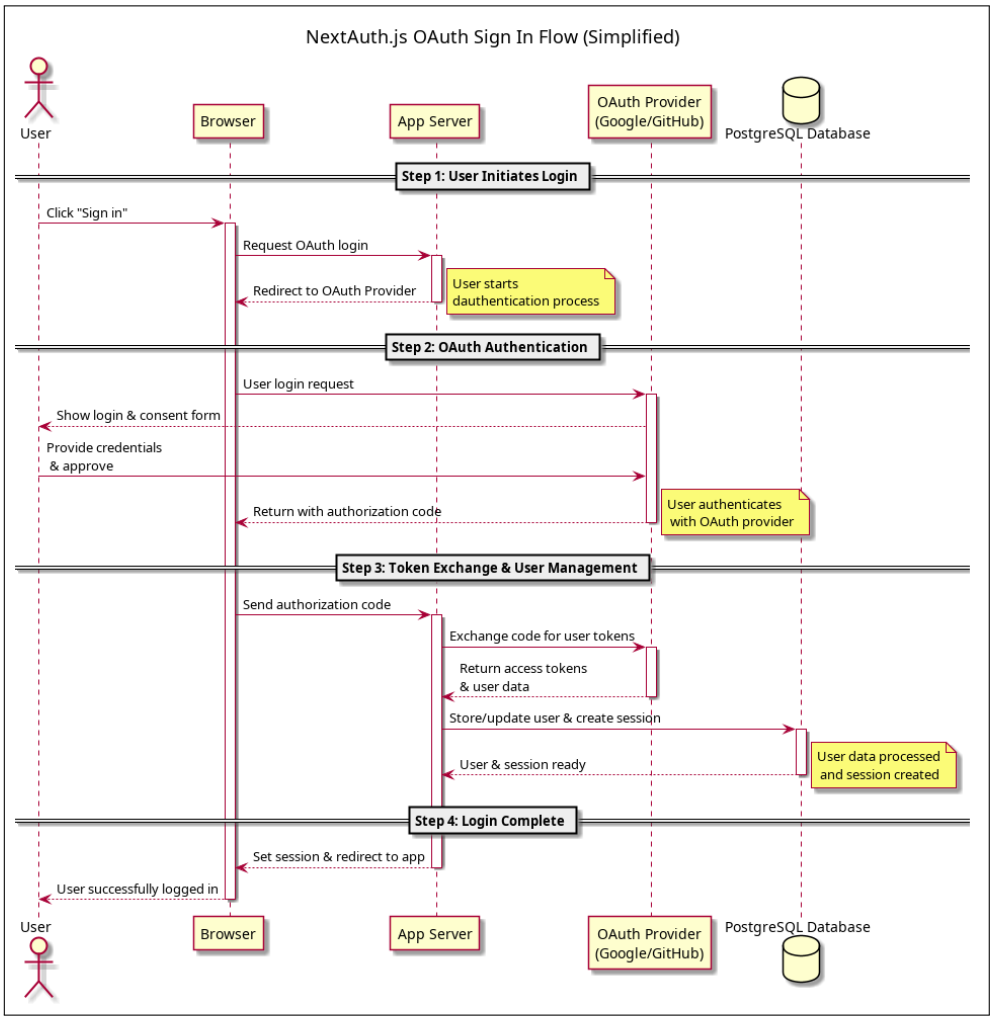

Figure 4.3: OAuth Authentication Sequence

4.5 Implementation Results

4.5.1 Landing Page

Figure 4.4: Responsive Landing Page

Modern design featuring clear value proposition, feature highlights, and prominent call-to-action elements optimized for user engagement and conversion.

4.5.2 Authentication Interface

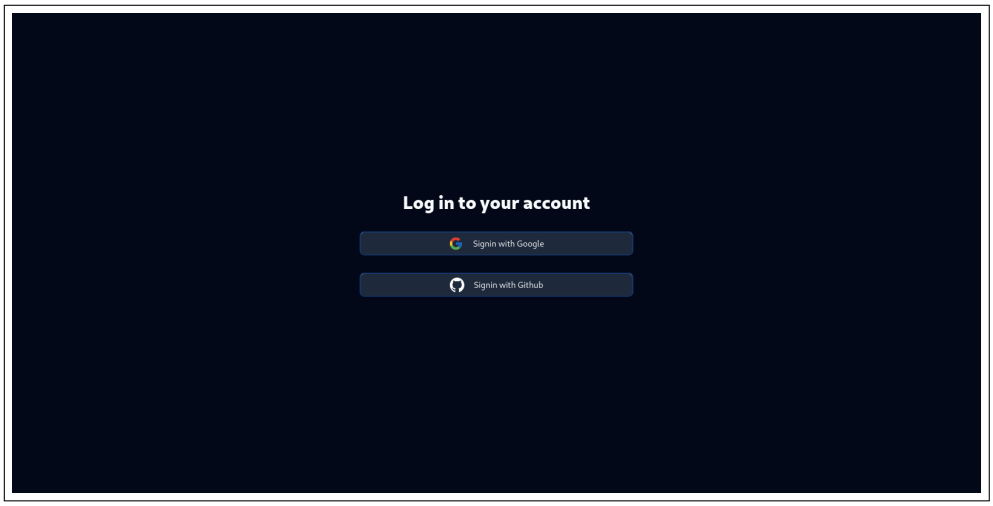

Figure 4.5: OAuth Sign-In Interface

Clean, user-friendly authentication interface supporting multiple OAuth providers with consistent branding and accessibility standards.

4.6 Sprint Retrospective

Achievements:

- Successful OAuth integration with Google and GitHub
- Robust cross-device session management
- Responsive landing page with high conversion potential
- Secure authentication flow with proper error handling

Challenges Resolved:

- OAuth configuration complexities across environments
- Session persistence optimization
- Cross-browser compatibility testing

4.7 Conclusion

Sprint II successfully established the authentication infrastructure and user entry point. The implementation of NextAuth.js with OAuth providers and Prisma database management provides a secure, scalable foundation for user management. The responsive landing page effectively communicates value while guiding user engagement. These achievements create a solid foundation for subsequent development phases, with robust security and optimal user experience.

Chapter 5

Sprint III: Project Management Implementation

5.1 Introduction

Sprint III implements comprehensive project management functionality within the UML diagram platform. This sprint introduces essential features enabling users to organize, manage, and maintain UML projects effectively, serving as the foundation for user workflow organization with capabilities for project creation, modification, visualization, and data export.

5.2 Sprint Planning

5.2.1 Objectives

Sprint III aims to implement a complete project management system allowing users to efficiently organize UML diagram projects through:

- Project creation with customizable parameters
- Intuitive project browsing and viewing capabilities
- Secure project modification features
- Safe project deletion with confirmations
- Robust export system for compressed project diagrams

5.2.2 Sprint Backlog

Table 5.1: Sprint Backlog

ID	User Story	Priority
US 3.1	Create new projects to organize UML diagrams systematically	M
US 3.2	Read and view project details to access existing work	M
US 3.3	Update project details to maintain current information	M
US 3.4	Delete projects to manage workspace efficiently	M
US 3.5	Download project diagrams compressed in zip format	M

5.3 Analysis and Design

5.3.1 Use Case Analysis

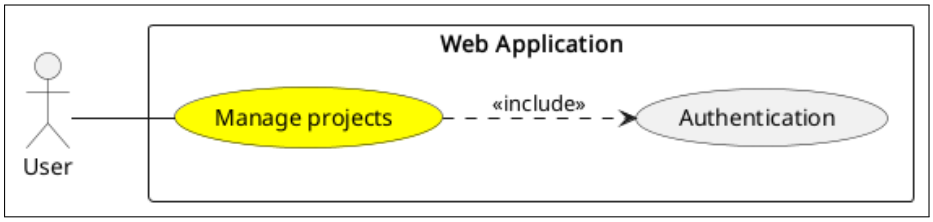

Figure 5.1: Sprint III Use Case Diagram

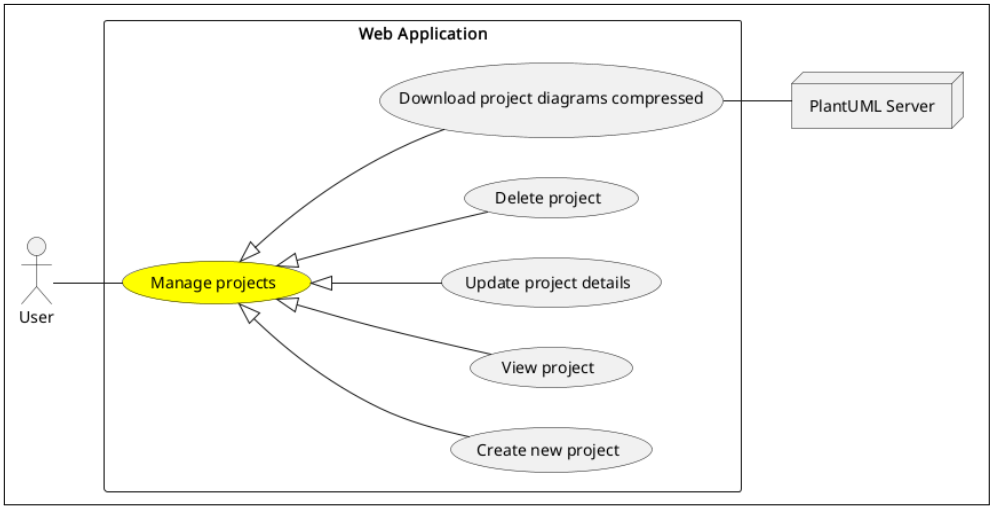

Figure 5.2: Refined Project Management Use Cases

5.3.2 Key Use Case Specifications

5.3.2.1 Create New Project (UC-3.1)

Main Flow: User accesses creation form → enters project details → selects type-/settings → system validates → creates project → displays confirmation → redirects to dashboard.

Alternative Flows: Invalid input triggers validation errors; system errors maintain form data.

5.3.2.2 View Project (UC-3.2)

Main Flow: User accesses project list → selects project → system retrieves details → displays organized layout → enables diagram navigation.

Alternative Flows: Empty projects show appropriate messages; access errors redirect or show permissions.

5.3.2.3 Update Project (UC-3.3)

Main Flow: User accesses edit interface → modifies fields → submits changes → system validates → saves to database → confirms success.

Alternative Flows: No changes or validation errors handled appropriately.

5.3.2.4 Delete Project (UC-3.4)

Main Flow: User selects deletion → system shows confirmation → user confirms → system removes data → cleans resources → confirms deletion.

Alternative Flows: Cancellation or shared project warnings handled safely.

5.3.2.5 Download Compressed (UC-3.5)

Main Flow: User accesses export → selects formats → initiates download → system generates diagrams → creates zip → downloads file.

Alternative Flows: Large projects show progress; selective export options available.

5.4 System Design

5.4.1 Sequence Diagrams

The following diagrams illustrate system component interactions:

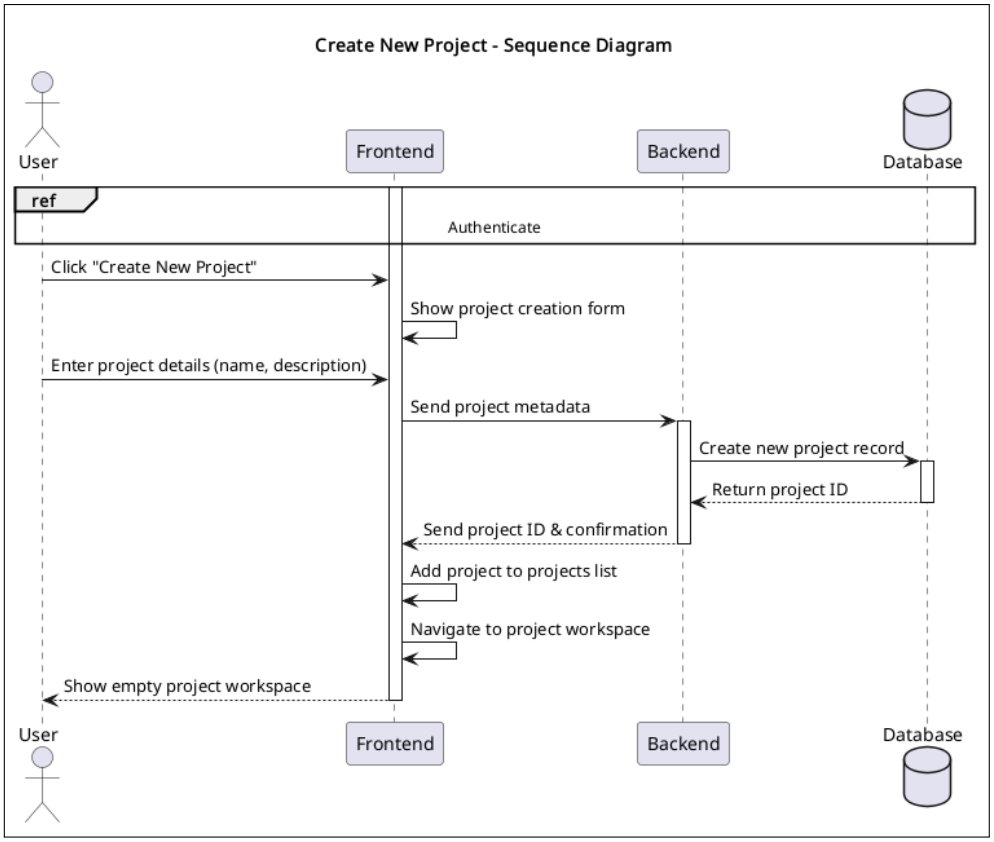

Figure 5.3: Create Project Sequence

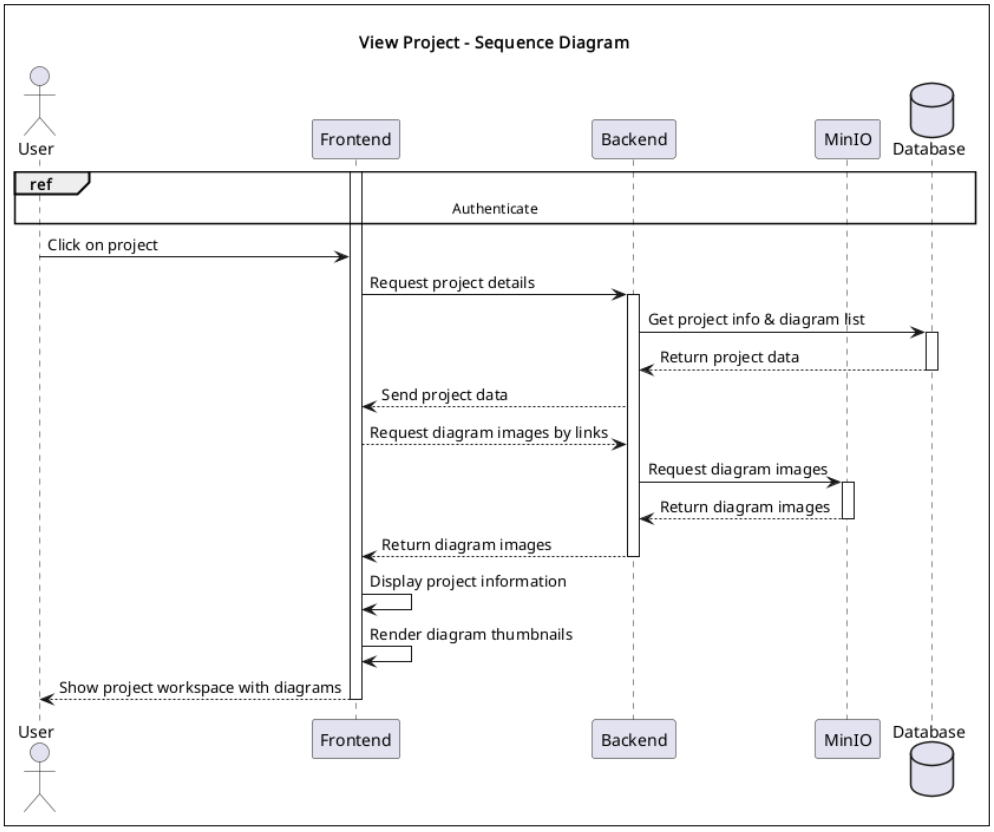

Figure 5.4: View Project Sequence

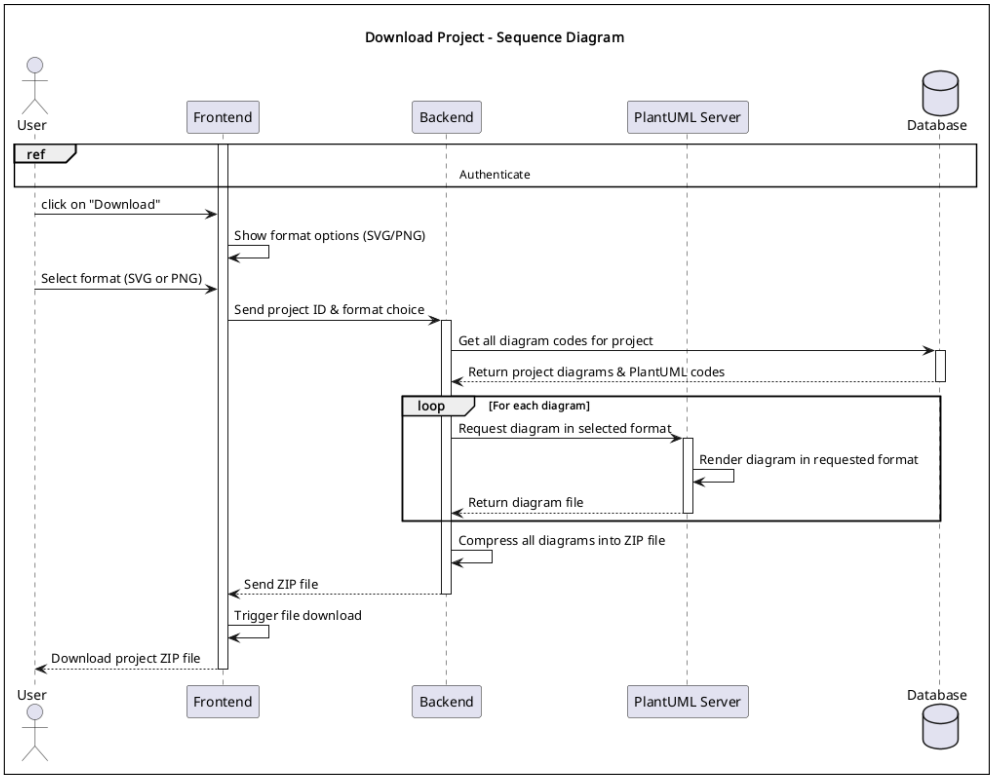

Figure 5.5: Download Project Diagrams Sequence

5.5 Implementation Results

5.5.1 User Interface Screenshots

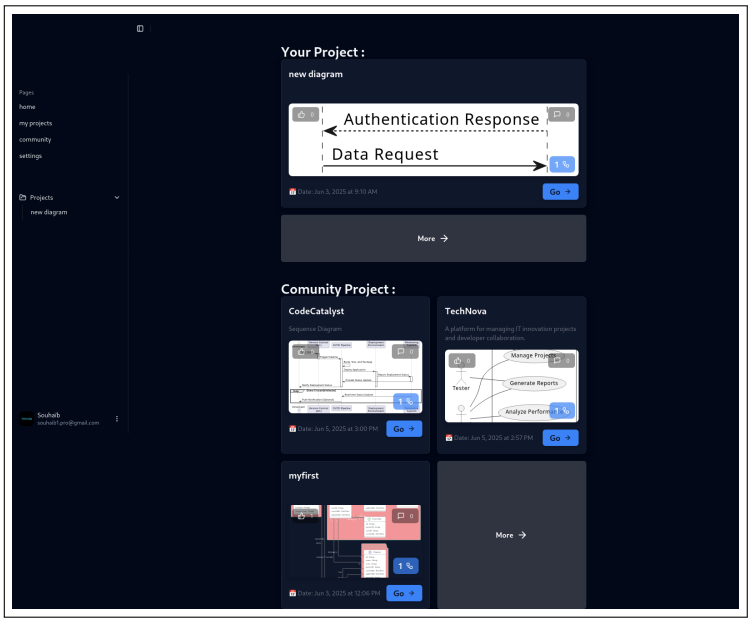

Figure 5.6: Home page with project overview

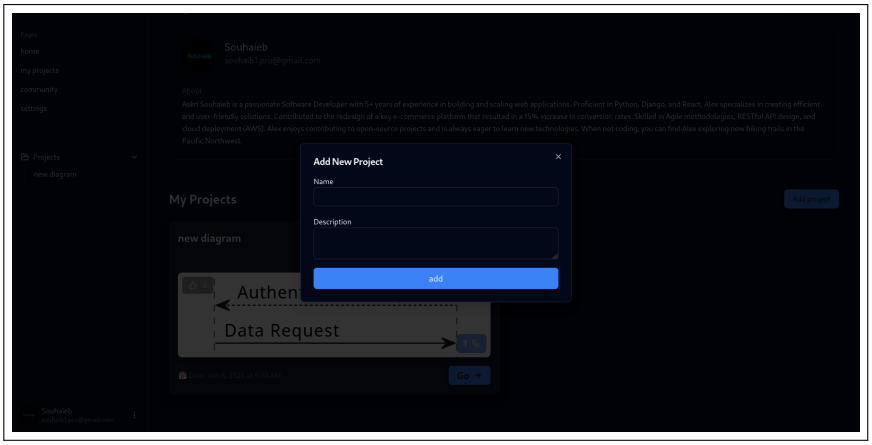

Figure 5.7: Project creation interface

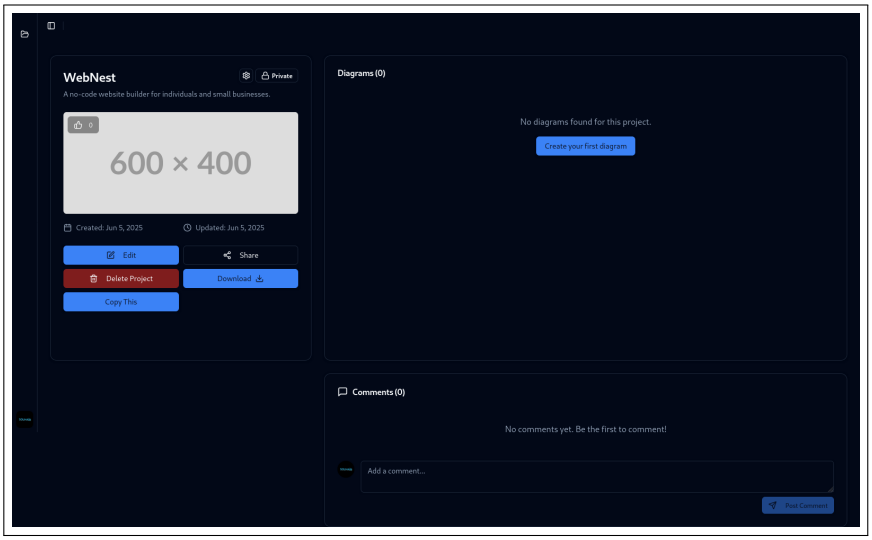

Figure 5.8: Project details and management

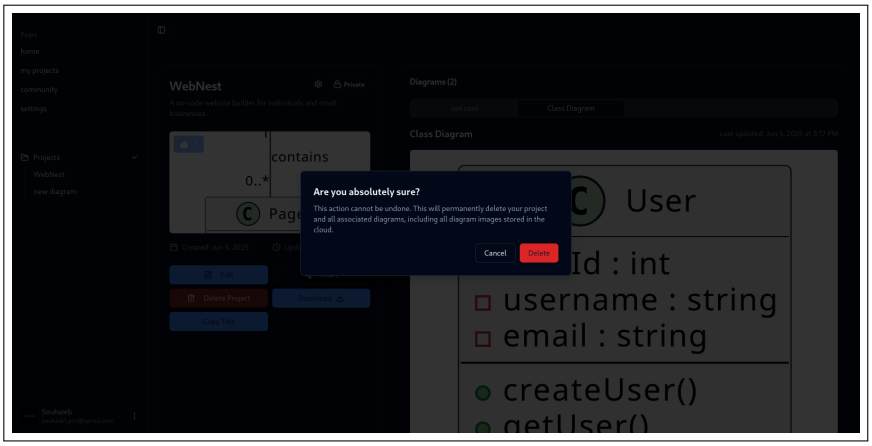

Figure 5.9: Project deletion confirmation

5.6 Sprint Retrospective

5.6.1 Achievements

Sprint III successfully delivered comprehensive project management capabilities exceeding initial expectations:

- Complete CRUD operations implementation
- Robust export functionality with multiple formats
- Intuitive user interfaces with validation
- Strong technical execution and requirement analysis

5.6.2 Areas for Improvement

- Enhanced error handling mechanisms
- Performance optimization for large projects
- Extended sharing capabilities

5.7 Conclusion

Sprint III established a robust project management foundation providing users with comprehensive tools for organizing and managing UML projects effectively. The implemented features deliver significant value through intuitive interfaces, reliable functionality, and scalable architecture. This sprint positions the platform for continued growth and enhanced collaboration capabilities, demonstrating the team's ability to deliver complex functionality while maintaining high quality standards.

Chapter 6

Sprint IV: Diagram & Workspace Management

6.1 Introduction

Sprint IV focuses on implementing core diagram and workspace management functionality, representing a significant milestone in developing a comprehensive diagramming tool. This sprint delivers diagram lifecycle management and advanced workspace features including AI-assisted editing and interactive code editing capabilities.

6.2 Sprint Planning

6.2.1 Objectives

Primary objectives include implementing comprehensive diagram CRUD operations, establishing robust workspace environment with split-view functionality, integrating AI assistance for diagram editing, and ensuring seamless PlantUML server integration for rendering.

6.2.2 Sprint Backlog

Table 6.1: Sprint IV Backlog

Feature	ID	User Story	Priority
Manage Diagrams	4.1	As a user, I want to create a new diagram so that I can visualize my ideas.	M
	4.2	As a user, I want to view my diagram so that I can review my work.	M
	4.3	As a user, I want to update diagram details so that I can improve my designs.	M
	4.4	As a user, I want to delete a diagram so that I can remove unwanted content.	M
	4.5	As a user, I want to download diagram images in appropriate formats so that I can use them in other applications.	S
Manage Workspace	5.1	As a user, I want to control and split views in my workspace so that I can work efficiently.	S
	5.2	As a user, I want to edit diagram code in an interactive editor so that I can create diagrams efficiently.	M
	5.3	As a user, I want to chat with an AI model to edit diagram code so that I can get assistance with complex diagrams.	C
	5.4	As an AI system, I need to respond to user requests and help edit diagram code so that users can create better diagrams.	C
	5.5	As a PlantUML Server, I need to render diagram code into diagram images so that users can visualize their work.	M

6.3 System Analysis

6.3.1 Use Case Overview

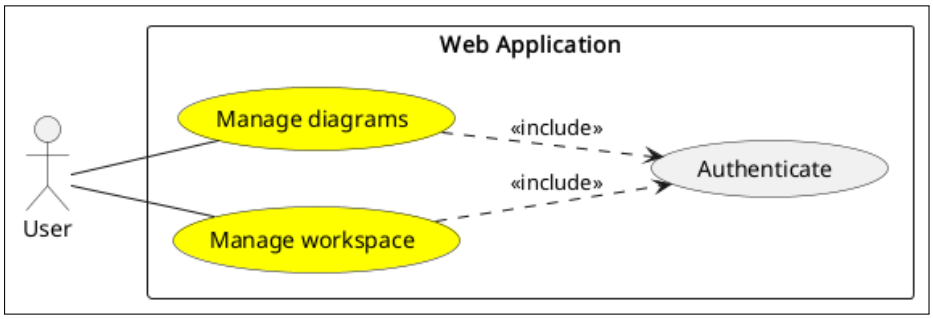

Figure 6.1: Sprint IV Use Case Diagram

6.3.2 Core Features

6.3.2.1 Diagram Management

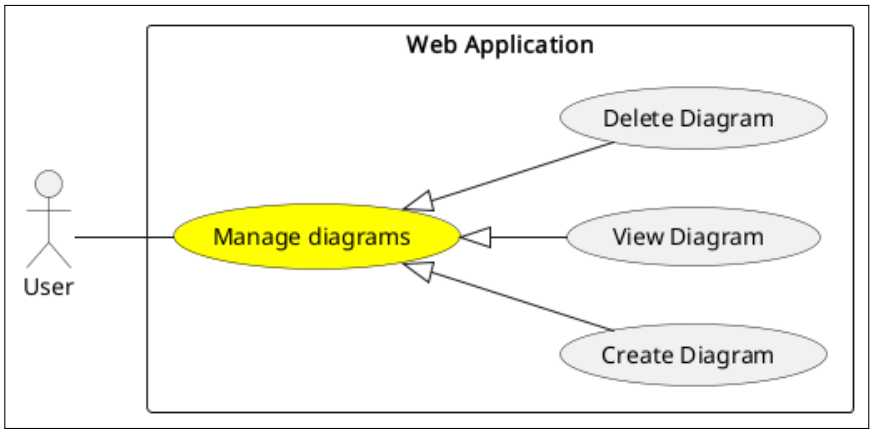

Figure 6.2: Diagram Management Use Cases

Key operations include:

- **Create Diagram**: User initiates new diagram creation with name and type selection
- **View/Edit Diagram**: Interactive editor with real-time preview and validation
- **Delete Diagram**: Secure deletion with confirmation dialog
- **Download Diagram**: Export in multiple formats (PNG, SVG)

6.3.2.2 Workspace Management

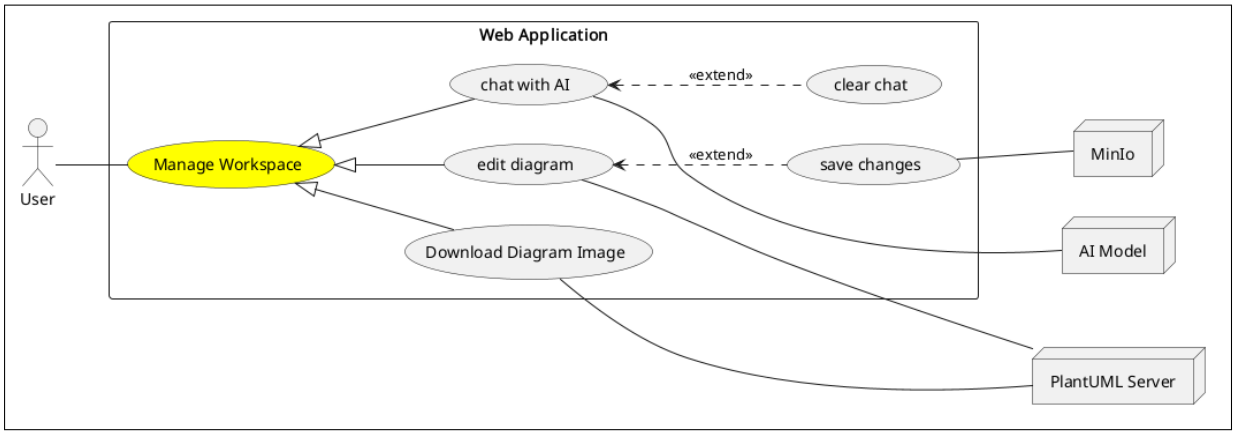

Figure 6.3: Workspace Management Use Cases

Core workspace features:

- **Load Workspace**: Initialize user environment with diagrams and preferences
- **Edit Diagram**: Interactive code editor with syntax highlighting
- **AI Chat**: Natural language assistance for diagram creation and troubleshooting
- **Save Changes**: Persistent storage with validation and error handling

6.4 System Design

6.4.1 Process Flow

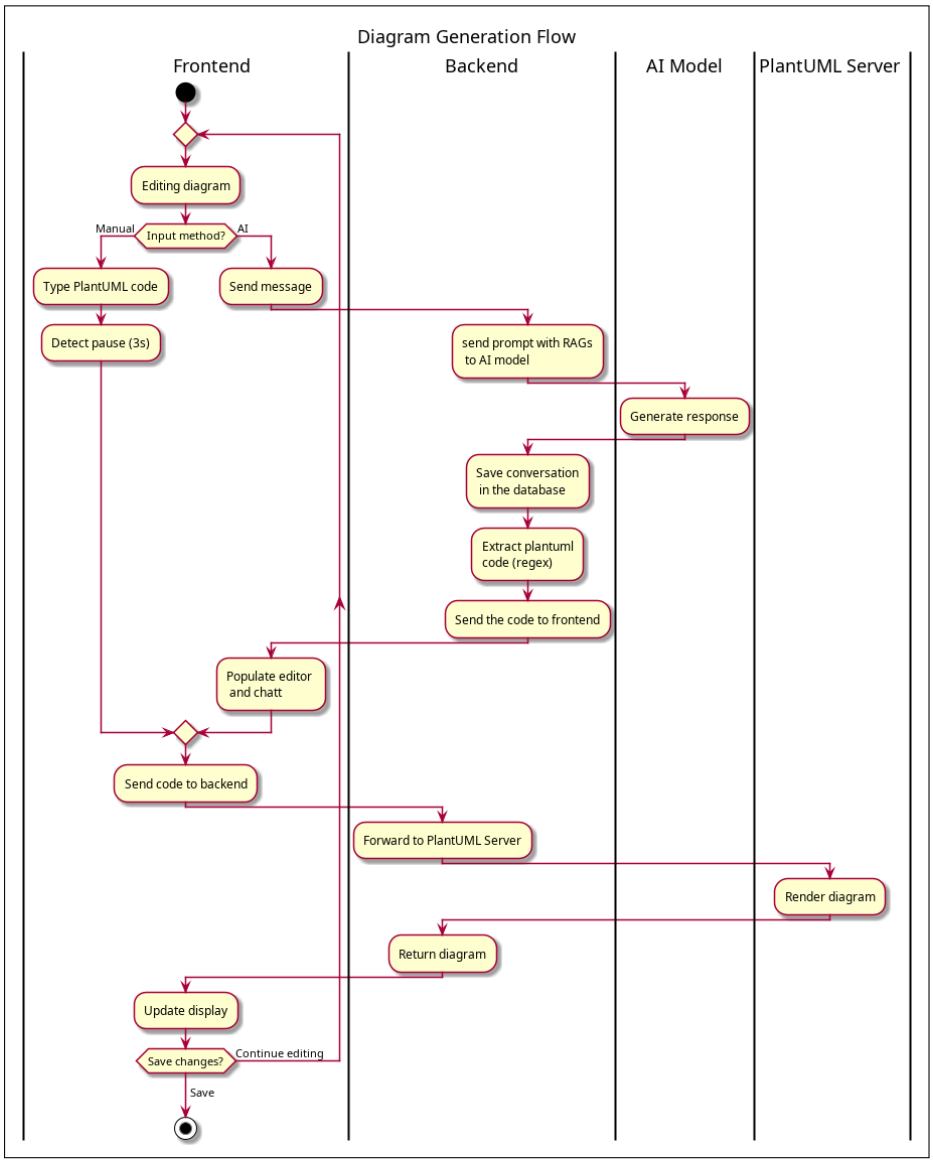

Figure 6.4: Diagram Editing Activity Flow

6.4.2 Key Sequence Diagrams

6.4.2.1 Diagram Creation Process

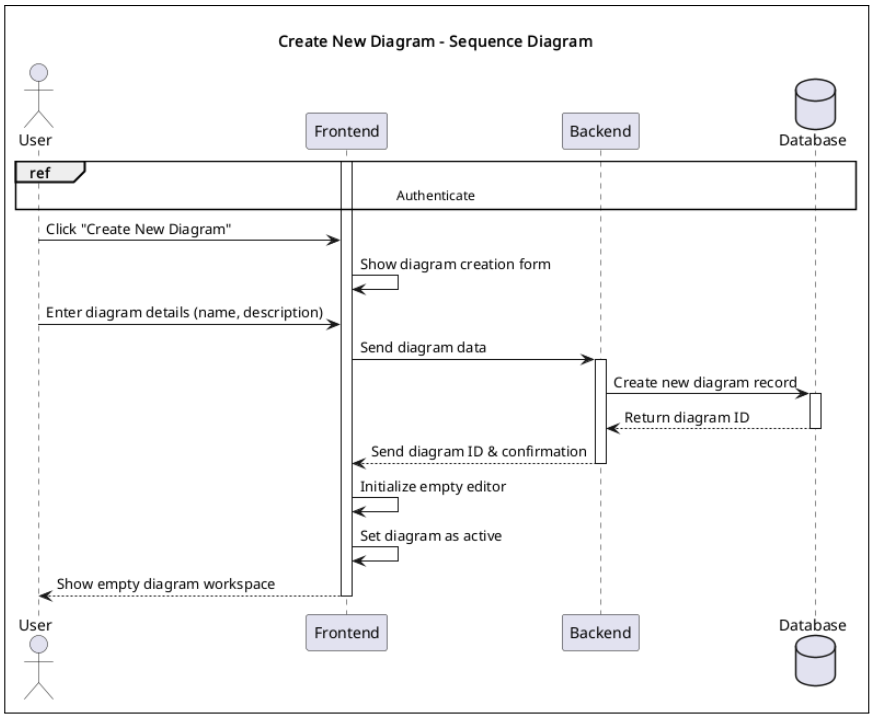

Figure 6.5: Create New Diagram Sequence

6.4.2.2 AI-Assisted Editing

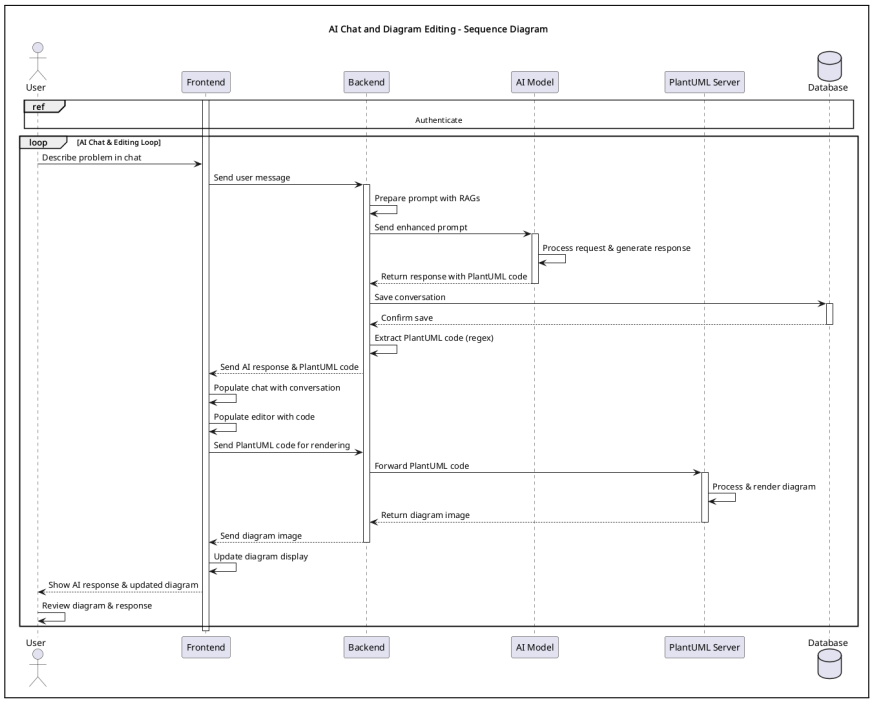

Figure 6.6: AI Chat Integration Sequence

6.5 Implementation Results

6.5.1 Core Interfaces

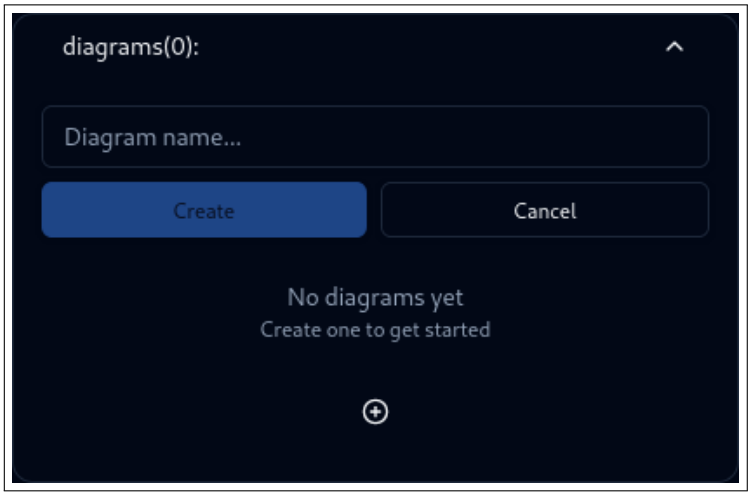

Figure 6.7: Diagram Creation Interface

The diagram creation interface provides intuitive name specification, type selection, and project initialization capabilities.

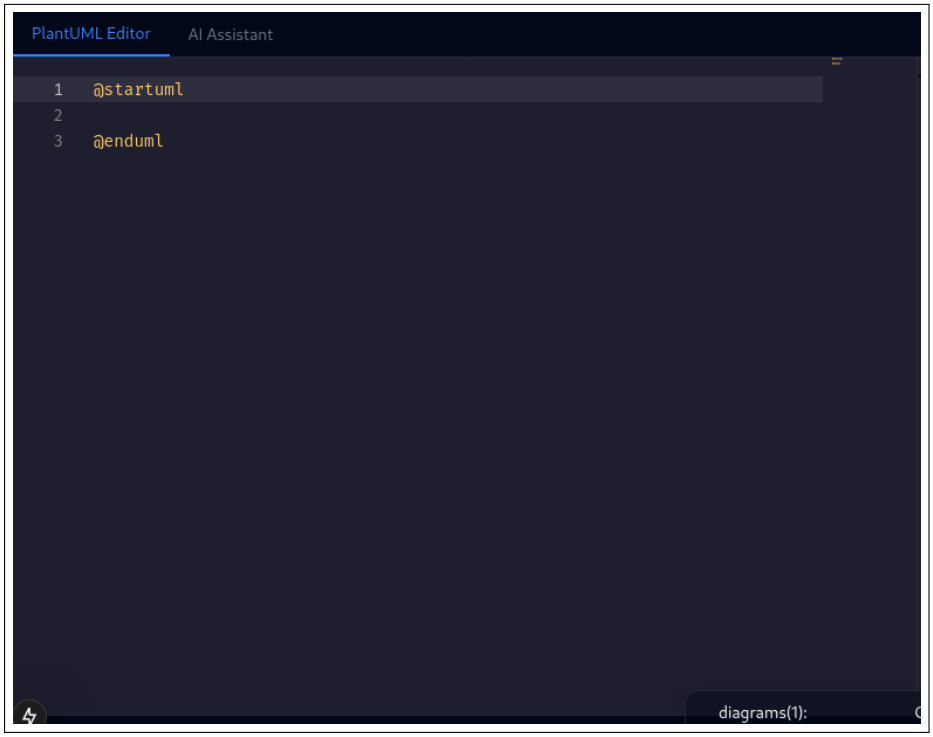

Figure 6.8: Interactive Code Editor with Real-time Preview

The interactive workspace features syntax highlighting, real-time validation, and seamless preview integration for enhanced productivity.

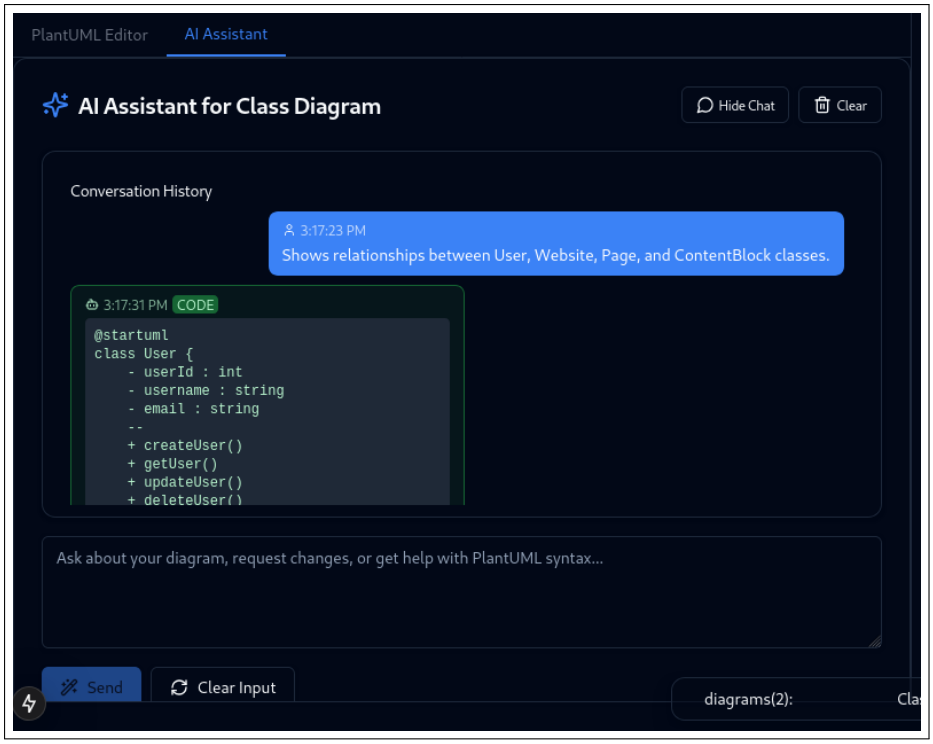

Figure 6.9: AI Assistant Integration

AI assistant provides intelligent support through natural language interaction for diagram creation, code improvement, and troubleshooting assistance.

6.5.2 Workspace Environment

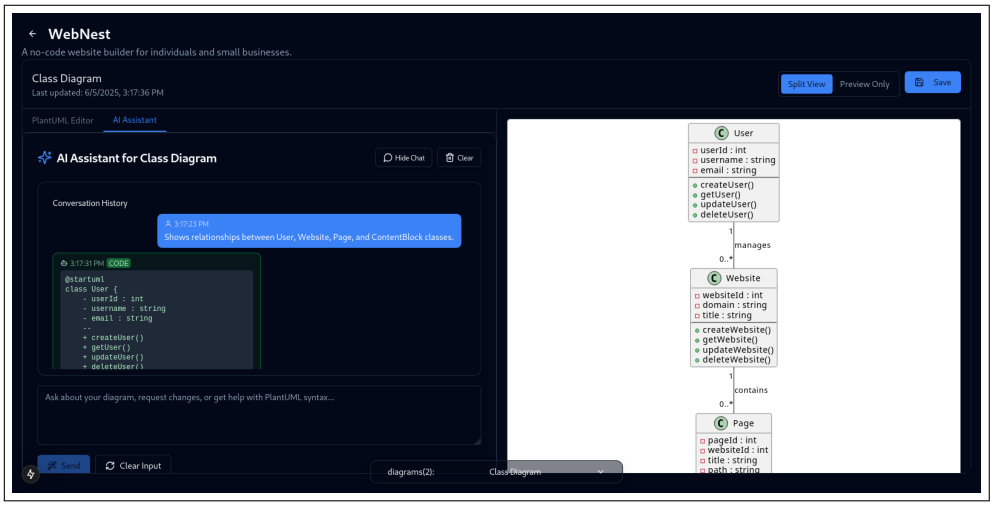

Figure 6.10: Split-View Workspace - Secondary View

The dual-pane workspace enables simultaneous code editing and diagram preview, providing immediate visual feedback and significantly improving user productivity.

6.6 Sprint Retrospective

6.6.1 Achievements

- Successfully implemented complete diagram CRUD operations
- Effective AI assistant integration with natural language processing
- Smooth PlantUML server integration for high-quality rendering
- Real-time editing with immediate visual feedback

6.6.2 Challenges & Solutions

- **Challenge**: AI service integration complexity
- **Solution**: Implemented robust error handling and fallback mechanisms
- **Challenge**: Performance optimization for large diagrams
- **Solution**: Added caching and progressive rendering
- **Challenge**: Browser compatibility for download features
- **Solution**: Implemented cross-browser compatibility layer

6.7 Conclusion

Sprint IV successfully delivered comprehensive diagram and workspace management capabilities that form the application's core foundation. The implementation combines efficient CRUD operations with intelligent AI assistance and interactive editing features, providing users with a powerful and intuitive diagramming platform. The integration of PlantUML server ensures high-quality rendering, while the AI assistant adds significant value for complex diagramming tasks. These achievements establish a solid foundation for future enhancements and advanced features in subsequent development cycles.

Chapter 7

Study and Implementation of Sprint V: Community Interaction & Profile Management

7.1 Introduction

Sprint V focuses on building a vibrant community ecosystem and profile management capabilities. This sprint introduces community interaction features enabling users to engage through comments, likes, and collaborative copying mechanisms, alongside robust profile management functionality for showcasing work effectively.

The community features transform the platform from a diagramming tool into a collaborative workspace where users discover, learn from, and build upon each other's work.

7.2 Sprint Planning

7.2.1 Objectives and Backlog

Sprint V objectives include developing community exploration, project interaction features, comment management, project copying functionality, and comprehensive profile management.

Table 7.1: Sprint V Product Backlog

ID	User Story	Feature	Size
6.1	Explore community to discover interesting projects	Community	S
6.2	Visitor community exploration to see platform offerings	Community	S
6.3	Comment on projects to provide feedback	Community	C
6.4	Like/unlike projects to show appreciation	Community	C
6.5	Share projects to promote content	Community	C
6.6-6.8	Update, delete, and like/unlike comments	Community	C
6.9	Copy community projects to workspace	Community	S
7.1	Edit profile to keep information current	Profile	S
7.2	View public projects on profiles	Profile	S

7.3 System Analysis

7.3.1 Use Case Overview

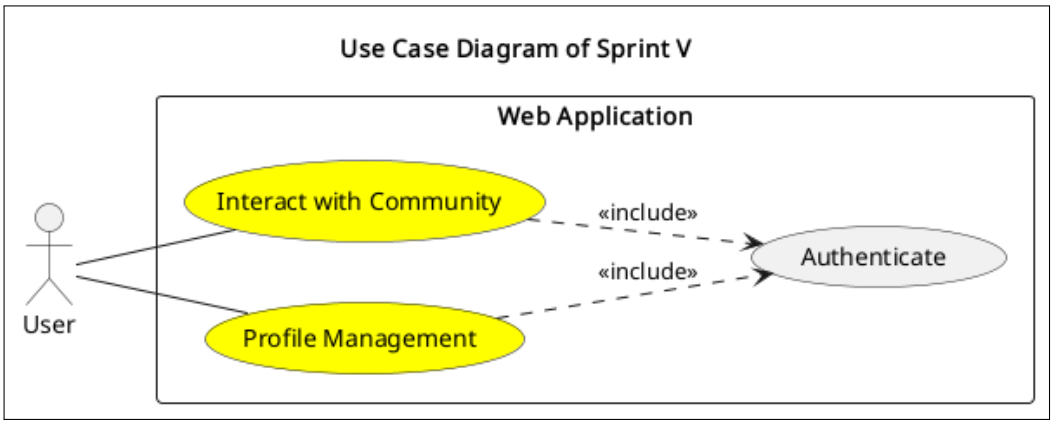

Figure 7.1: Use Case Diagram for Sprint V

7.3.2 Community Interaction Features

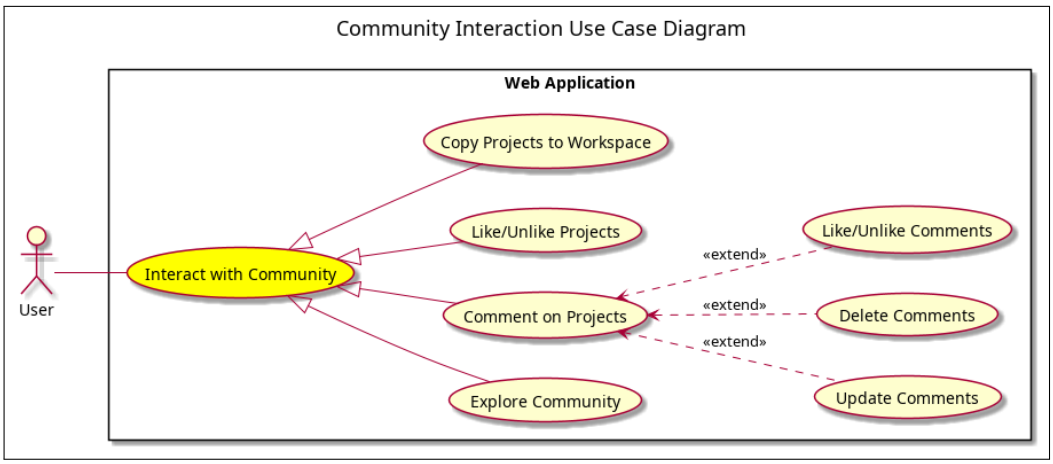

Figure 7.2: Community Interaction Use Cases

7.3.2.1 Key Use Cases Description

Explore Community: Users and visitors browse public projects with filtering and search capabilities to discover interesting content and platform offerings.

Comment Management: Authenticated users can create, update, and delete comments on projects, providing feedback and engaging in community discussions with full CRUD operations.

Project Interactions: Users can like/unlike projects and comments to show appreciation and engage with community content, with real-time UI updates.

Project Copying: Users can copy community projects to their workspace for learning and building upon others' work, with proper attribution and workspace integration.

7.3.3 Profile Management Features

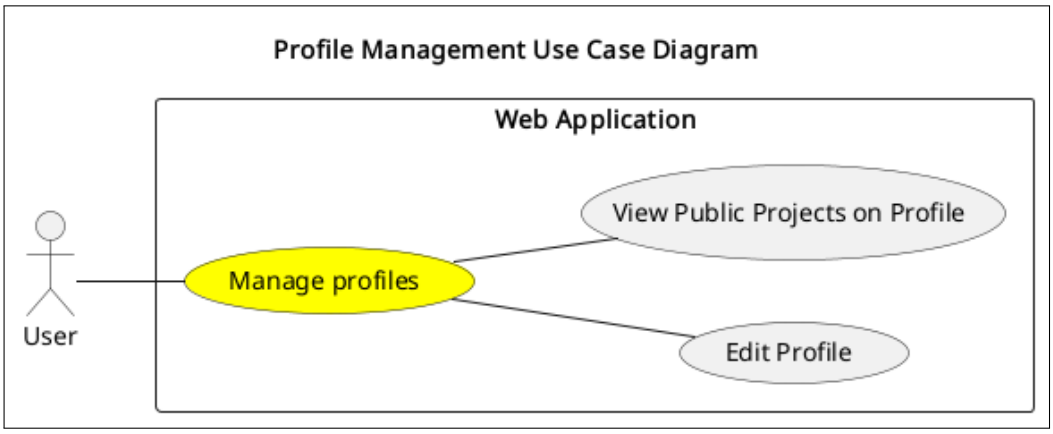

Figure 7.3: Profile Management Use Cases

Edit Profile: Users can update personal information, maintain account details, and manage their public presence with validation and confirmation feedback.

View Public Projects: Users and visitors can explore public projects on user profiles, showcasing portfolios and enabling project discovery through user-centric browsing.

7.4 System Design

7.4.1 Key Sequence Diagrams

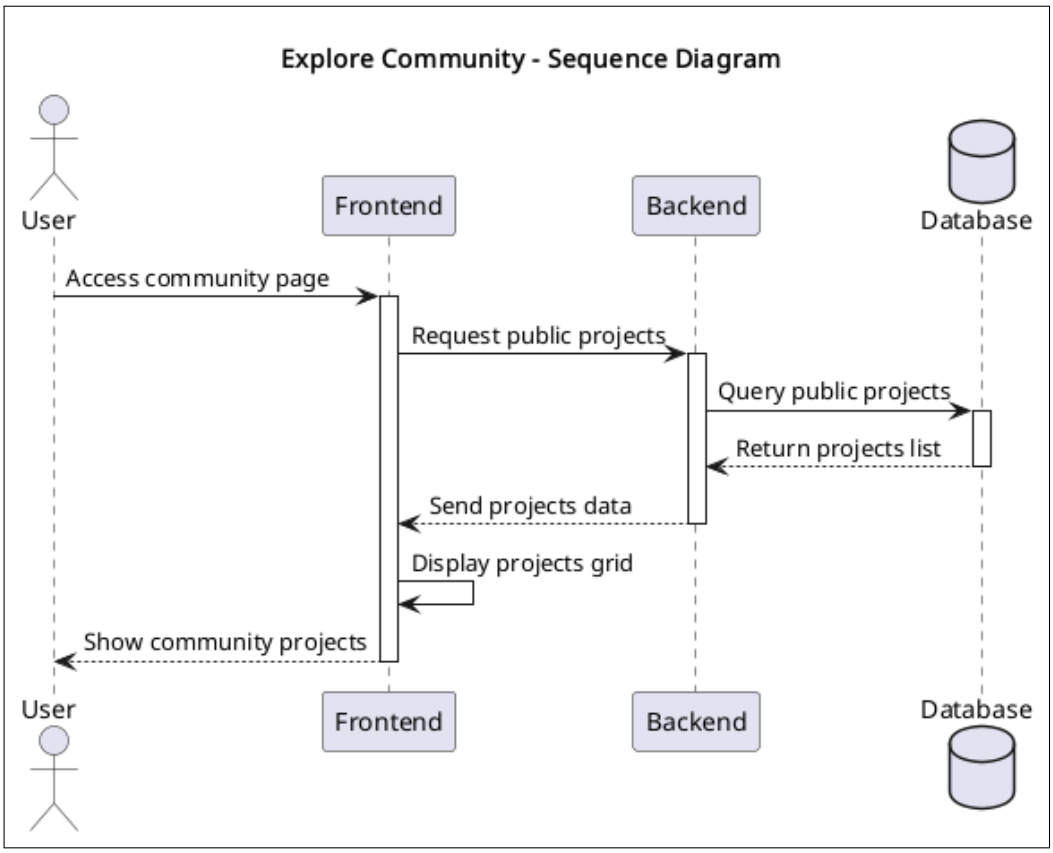

Figure 7.4: Community Exploration Flow

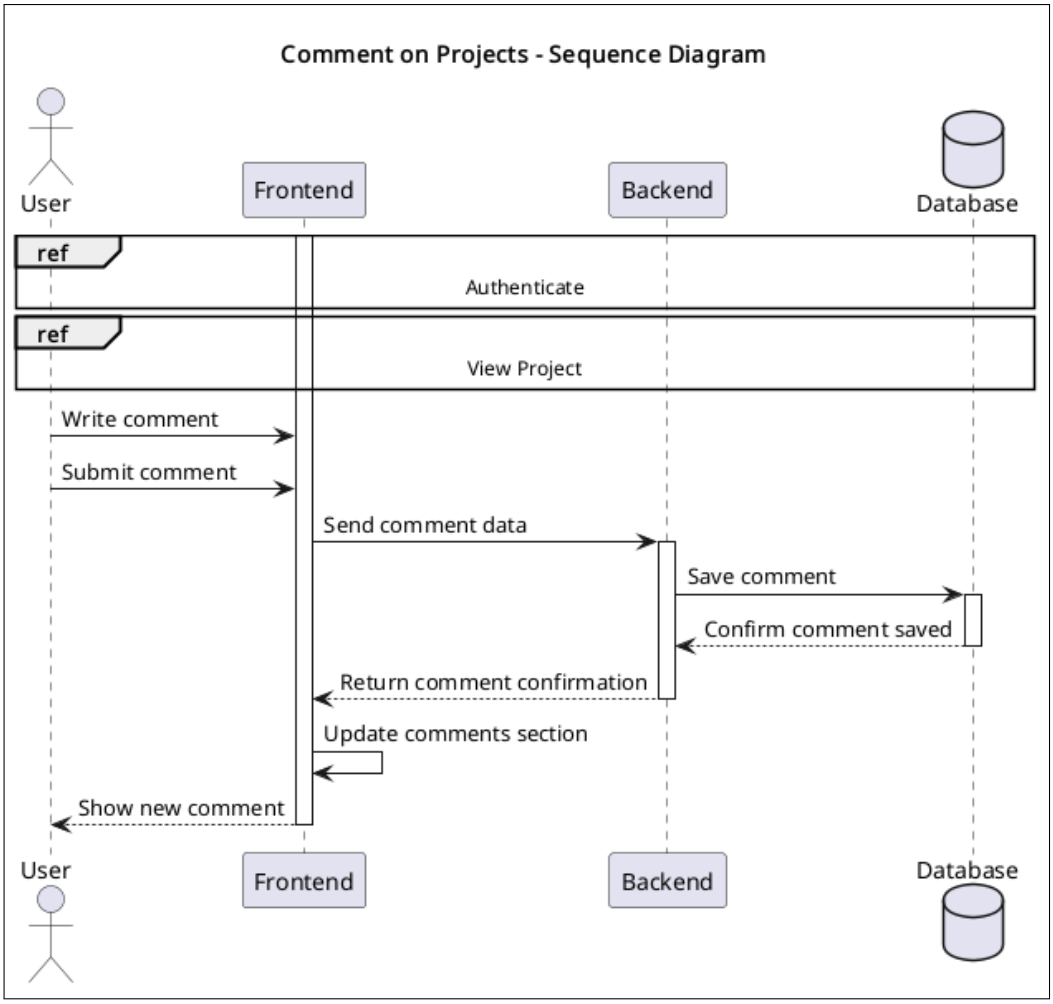

Figure 7.5: Project Commenting Flow

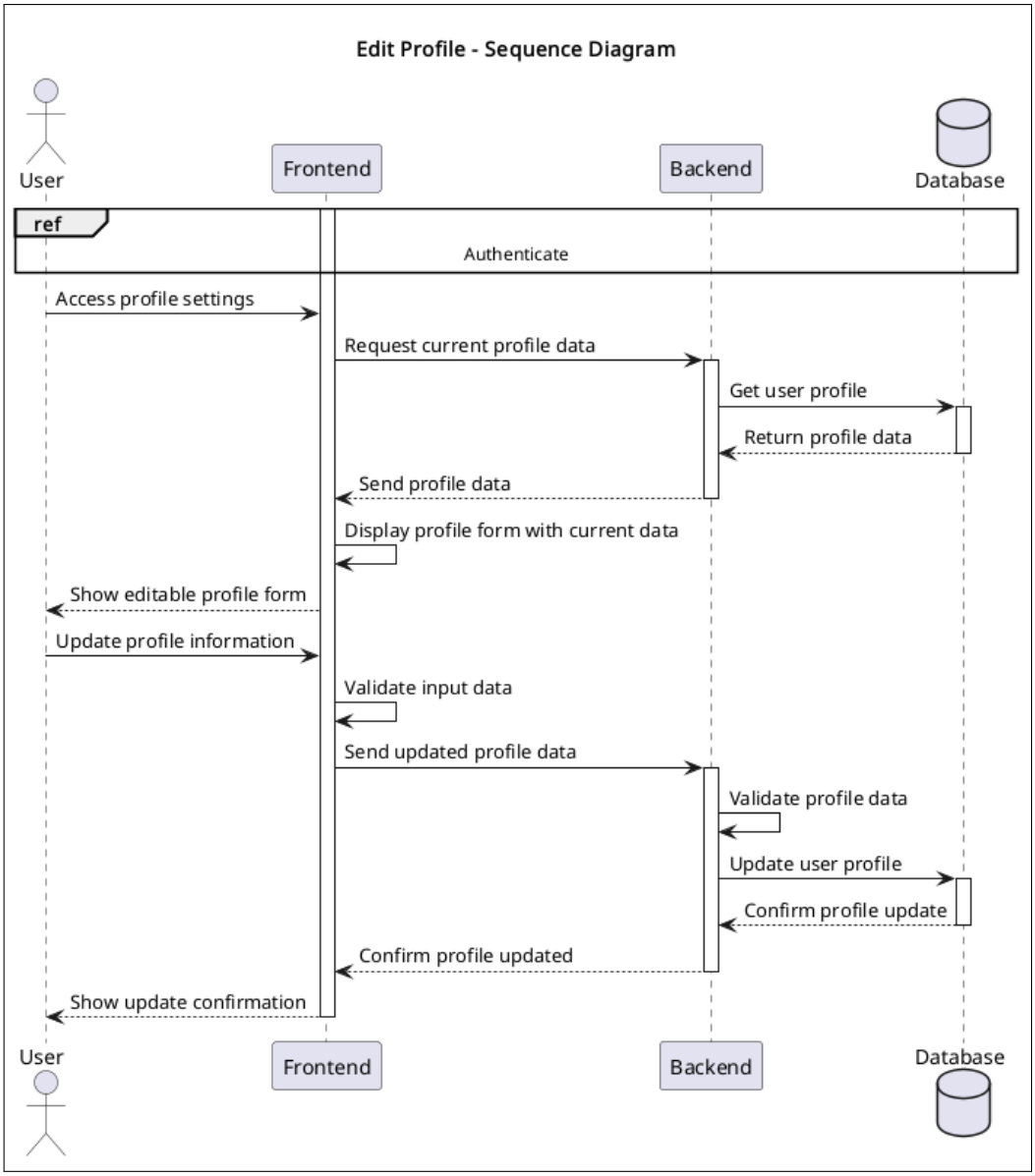

Figure 7.6: Profile Editing Flow

7.5 Implementation Results

7.5.1 Community Features

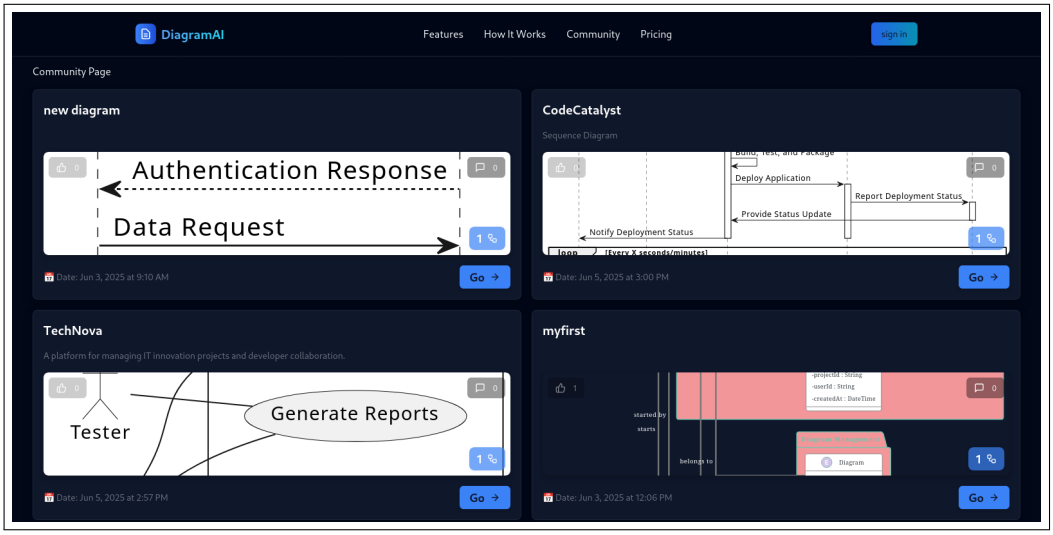

Figure 7.7: Community Exploration Interface

The community interface provides intuitive project discovery with filtering, search capabilities, and clear project previews with essential metadata and engagement metrics.

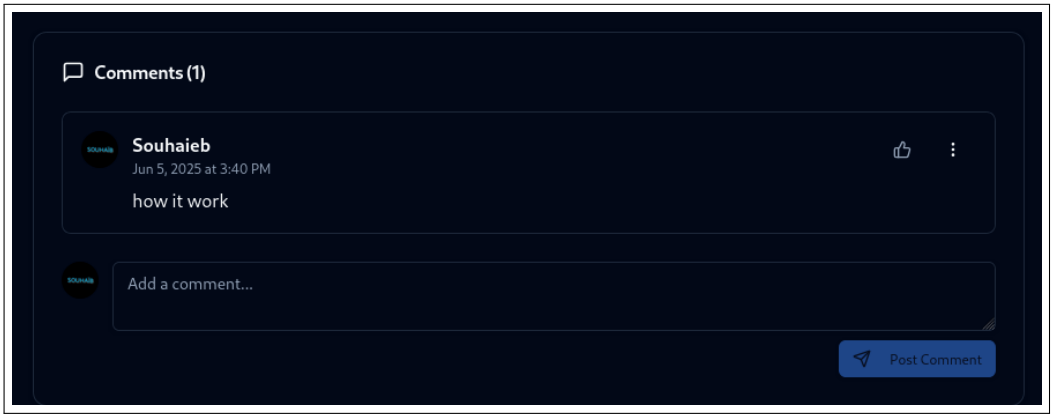

Figure 7.8: Comment System Implementation

The comment system demonstrates comprehensive management including creation, editing, deletion, and interaction features for meaningful project discussions.

7.5.2 Profile Management

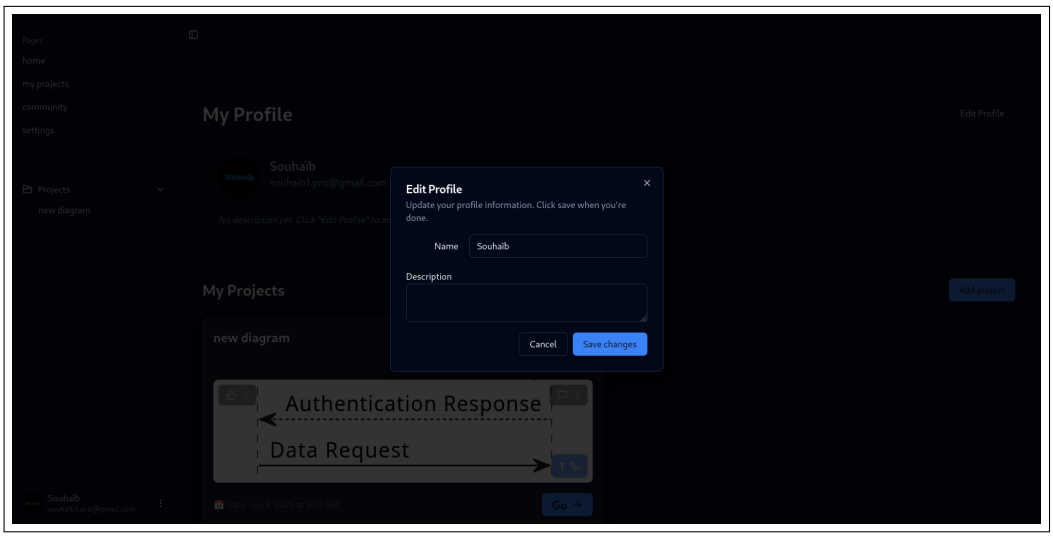

Figure 7.9: Profile Management Interface

The profile interface enables users to maintain account information, update personal details, and manage their public platform presence with validation feedback.

7.6 Sprint Retrospective

7.6.1 Achievements

- Successfully implemented comprehensive community interaction features
- Effective team collaboration and timeline adherence
- Positive user feedback on comment management functionality
- Smooth integration of profile management features

7.6.2 Areas for Improvement

- Enhanced real-time notification system needed
- Performance optimization for large-scale community loading
- Mobile responsiveness improvements required
- User interface refinements based on usability testing

7.6.3 Future Actions

- Implement real-time notifications for community engagement
- Optimize database queries for improved performance
- Conduct comprehensive usability testing sessions
- Enhance mobile user experience across all features

7.7 Conclusion

Sprint V successfully transformed the platform into a collaborative community-driven ecosystem through comprehensive interaction features and robust profile management. The implementation of community exploration, project commenting, liking mechanisms, and profile editing establishes a solid foundation for user engagement and knowledge sharing.

The delivery demonstrates commitment to building a collaborative platform where users learn from each other, share knowledge, and build upon collective expertise, setting the stage for continued platform evolution.

Chapter 8

General Conclusion

8.1 Summary of Achievements

This project successfully delivered a comprehensive intelligent UML diagram generation platform that transforms traditional software modeling approaches. Through five systematic sprints using Scrum methodology, we created an AI-driven solution that bridges the gap between GUI-based tools and complex textual specifications.

Technical achievements include robust containerized infrastructure using PostgreSQL, MinIO, PlantUML, and Next.js, providing scalable foundation with DevOps best practices. OAuth-based authentication through NextAuth.js and Prisma database management ensure secure user management.

The platform delivers comprehensive project management capabilities, enabling users to organize and share UML projects efficiently. Core diagramming features include CRUD operations, intelligent workspace with AI assistance, and high-quality PlantUML rendering integration.

The final sprint transformed the platform into a collaborative community-driven ecosystem with interaction features, project commenting, and profile management, creating a knowledge-sharing environment beyond individual diagramming.

8.2 Challenges Faced

Development encountered significant technical challenges requiring innovative solutions. Technical architecture complexity arose from integrating AI services, PlantUML rendering, and authentication while maintaining performance standards. AI integration presented challenges in context management and ensuring contextually relevant suggestions.

User experience design proved challenging when balancing feature richness with interface simplicity. Performance optimization required sophisticated strategies for handling complex diagram rendering and AI interactions while maintaining browser responsiveness.

8.3 Future Perspectives

The platform establishes excellent foundation for continued innovation in intelligent diagramming tools. Future development will focus on advanced AI capabilities including natural language diagram generation and automated layout optimization.

Enhanced collaboration features will include real-time multi-user editing and integration with development tools. The platform will expand beyond UML to support additional modeling languages with pluggable architecture.

Enterprise features will include advanced user management and role-based access control. Mobile applications will extend accessibility while comprehensive API development will enable third-party integrations and community-driven feature development.

Bibliography

[1] LangChain. (2024). *LangChain – Build LLM-powered Applications*. Retrieved June 9, 2025, from https://www.langchain.com

[2] MinIO. (2024). *MinIO – High Performance Object Storage*. Retrieved June 9, 2025, from https://min.io

[3] PlantUML. (2024). *PlantUML – Open-Source UML Tool*. Retrieved June 9, 2025, from https://plantuml.com

[4] ShadCN UI. (2024). *ShadCN UI – Beautifully Designed UI Components*. Retrieved June 9, 2025, from https://ui.shadcn.com

[5] Docker. (2024). *Docker – Empowering App Development for Developers*. Retrieved June 9, 2025, from https://www.docker.com

[6] NextAuth.js. (2024). *NextAuth.js – Authentication for Next.js*. Retrieved June 9, 2025, from https://next-auth.js.org

[7] Express.js. (2024). *Express – Fast, unopinionated, minimalist web framework for Node.js*. Retrieved June 9, 2025, from https://expressjs.com

[8] LaTeX Project. (2024). *LaTeX – A Document Preparation System*. Retrieved June 9, 2025, from https://www.latex-project.org

[9] Vercel. (2024). *Next.js – The React Framework*. Retrieved June 9, 2025, from https://nextjs.org

[10] Prisma. (2024). *Prisma – Next-generation ORM for Node.js*. Retrieved June 9, 2025, from https://www.prisma.io

[11] Tailwind CSS. (2024). *Tailwind CSS – Rapidly Build Modern Websites*. Retrieved June 9, 2025, from https://tailwindcss.com

[12] Git SCM. (2024). *Git – Distributed Version Control System*. Retrieved June 9, 2025, from https://git-scm.com

[13] Node.js. (2024). *Node.js – JavaScript Runtime*. Retrieved June 9, 2025, from https://nodejs.org

[14] React. (2024). *React – A JavaScript library for building user interfaces*. Retrieved June 9, 2025, from https://reactjs.org

[15] TypeScript. (2024). *TypeScript – JavaScript With Syntax for Types*. Retrieved June 9, 2025, from https://www.typescriptlang.org

[16] Mozilla Firefox. (2024). *Firefox – Fast, Private & Free Web Browser*. Retrieved June 9, 2025, from https://www.mozilla.org/firefox

[17] VSCodium. (2024). *VSCodium – Free/Libre Open Source Software Binaries of VS Code*. Retrieved June 9, 2025, from https://vscodium.com

[18] GitHub. (2024). *GitHub – Where the World Builds Software*. Retrieved June 9, 2025, from https://github.com

[19] Linux Kernel Organization. (2024). *The Linux Kernel Archives*. Retrieved June 9, 2025, from https://kernel.org

[20] PostgreSQL. (2024). *PostgreSQL – The World's Most Advanced Open Source Relational Database*. Retrieved June 9, 2025, from https://www.postgresql.org