Ref : 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:

Souhaieb Askri, Issam Mekni

Supervised by:

Academic Supervisor   :Mr. Mohamed Naija

Professional Supervisor   :Mrs. Ines Askri

Host Organization   : Goodwill Engineering

Academic Year: 2024–2025

Dedication

To my beloved parents,

Words cannot express the depth of my gratitude for everything you have given me. You have been my foundation, my inspiration, and my unwavering source of strength throughout this journey. Your endless sacrifices, unconditional love, and constant belief in my abilities have shaped who I am today. This work stands as a humble tribute to your dedication and the invaluable lessons you have taught me. Thank you for being my guiding stars.

To my cherished sisters, dear friends, and all my loved ones,

Your presence in my life has been a blessing beyond measure. Through every triumph and challenge, you have stood beside me with encouragement and faith. Your support has been the wind beneath my wings, lifting me higher when I needed it most. This achievement belongs to all of us, for you are an integral part of this success story. My heart overflows with appreciation and love for each of you.

Souhaieb Askri

To my dear parents,

No amount of words can capture the magnitude of what you mean to me. You have been my pillars of strength, my sources of wisdom, and my greatest champions. Your relentless support, boundless patience, and selfless love have illuminated my path through every obstacle. The values you instilled in me and the sacrifices you made have brought me to this moment. I dedicate this work to you with profound gratitude and immense pride in being your son.

To my wonderful sisters, treasured friends, and beloved family,

Thank you for being the constants in my life's equation. Your encouragement during difficult times and your joy in my successes have meant the world to me. You have celebrated my victories and lifted me up during my struggles. This accomplishment is as much yours as it is mine, for you have all contributed to making it possible. With all my love and deepest appreciation.

Issam Mekni

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. Ines Askri* for her 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.

Souhaieb Askri & Issam Mekni

Contents

CONTENTS

CONTENTS

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

General Introduction

The rapid evolution of software development has increased demand for efficient design tools that can keep pace with accelerated development cycles. UML diagrams serve as essential visual representations that bridge the gap between conceptual design and implementation, facilitating clear communication among stakeholders and providing standardized documentation approaches that reduce project ambiguity and technical debt.

However, traditional UML diagramming approaches present significant productivity barriers. Manual diagram creation is time-consuming and error-prone, requiring specialized knowledge of UML syntax and conventions. Existing tools often demand steep learning curves and technical expertise, creating bottlenecks in the design process and excluding non-technical stakeholders from meaningful participation in system design discussions.

Recent advances in artificial intelligence and natural language processing have opened new possibilities for automating diagram generation, fundamentally transforming how developers and designers create visual system representations. These technologies enable the conversion of natural language descriptions into structured visual models, potentially eliminating the technical barriers that have traditionally limited UML adoption.

This project addresses the critical need for an intelligent platform that combines the precision of formal diagram specifications with the accessibility of natural language input. By enabling users to describe system designs in plain language while automatically generating professionally compliant UML diagrams, this approach democratizes UML creation for developers, students, and project managers regardless of their technical background, while maintaining the rigorous standards required for professional software development documentation.

Chapter 2

Overview of the Project

2.1 Introduction

This chapter introduces the host organization Goodwill Engineering and presents the project context, including the problem statement and proposed AI-driven UML generation solution. It also outlines the Agile methodology used for development and concludes with the project's expected outcomes.

2.2 Overview of the Host Organization

Goodwill Engineering is a Tunisian company specialized in developing software solutions. It was founded in 2011 by consultants with over 10 years of experience in SAGE solutions. As a growing IT services and engineering company (SSII), Goodwill is the official and authorized distributor of SAGE solutions in Tunisia, making it a key partner for many small and medium-sized enterprises. The company has extensive expertise in building customized systems for the finance and insurance sectors, with a strong focus on Business Intelligence, Big Data, analytical accounting, and payroll management. Goodwill also provides integrated solutions in system integration and ERP systems, aiming to support digital transformation and enhance operational efficiency and decision-making within organizations.

2.3 Presentation of the Project Context

2.3.1 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.

2.3.2 Existing Solutions

AI-Based Tools: Use LLM to generate diagrams from user input, but often lack accuracy.Tools like ChatUML[1] and DiagrammingAi[2] provide fast feedback but lack support for complex cases.

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

2.3.3 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.

2.4 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.

2.5 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 3

Project Initiation

3.1 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.

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

3.2 Requirements Analysis

3.2.1 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

3.2.2 Core Requirements

3.2.2.1 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

3.2.2.2 Non-Functional Requirements

- **Performance**: Page loads <3s, diagram rendering <5s, real-time syntax highlighting and rendering

- **Security**: HTTPS/TLS encryption, OAuth 2.0 authentication, input validation, XSS/CSRF protection, secure code execution sandboxing

- **Usability**: Responsive design across devices, WCAG 2.1 Level AA accessibility compliance, intuitive visual interface design

- **Editor Experience**: Syntax highlighting, intelligent autocomplete with context awareness, real-time error detection

- **SEO & Discoverability**: Server-side rendering (SSR) for search engine optimization, semantic HTML structure

3.3 Project Management

3.3.1 Scrum Roles

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

Table 3.1: Scrum team roles

3.3.2 Product Backlog

The product backlog represents a prioritized list of features and requirements derived from stakeholder needs and market analysis. Each backlog item follows the user story format and includes priority classification using MoSCoW method (Must have, Should have, Could have, Won't have this time).

Table 3.2: Product Backlog with User Stories

ID	Feature	Sub-ID	User Story	Priority
1	Authentication	1.1	As a user; I want to authenticate using my Google .	M
		1.2	As a user; I want to authenticate using my GitHub account.	M
		1.3	As a user; I want to stay authenticated across multiple devices so that I can access my account anywhere.	S
		1.4	As a user; I want to log out from my account .	M
2	Explore Landing Page	2.1	As a user; I want to explore the landing page so that I can understand the platform's features and benefits.	M
3	Manage Projects	3.1	As a user; I want to create a new project so that I can organize my diagrams.	M
		3.2	As a user; I want to view my projects.	M
		3.3	As a user; I want to update project details so that I can keep information current.	M
		3.4	As a user; I want to delete a project.	M
		3.5	As a user; I want to download project diagrams as images in a compressed ZIP file.	S
		3.6	As a user; I want to share my project with others.	S
4	Manage Diagrams	4.1	As a user; I want to create a new diagram.	M
		4.2	As a user; I want to view my diagram.	M
		4.3	As a user; I want to update diagram details.	M
		4.4	As a user; I want to delete a diagram.	M
5	Manage Workspace	5.1	As a user; I want to edit diagram code in an interactive editor.	M
		5.2	As a user; I want to chat with an AI model to edit diagram code.	C
		5.3	As an AI system; I need to respond to user requests and help edit diagram code.	C
		5.4	As a PlantUML Server; I need to render diagram code into diagram images.	M
6	Community Interaction	6.1	As a user; I want to explore the community.	S

Table 3.2 – continued from previous page

ID	Feature	Sub-ID	User Story	Priority
		6.2	As a user; I want to comment on projects.	C
		6.3	As a user; I want to like/unlike projects.	C
		6.4	As a user; I want to share projects.	C
		6.5	As a user; I want to update my comments.	C
		6.6	As a user; I want to delete my comments.	C
		6.7	As a user; I want to like/unlike comments.	C
		6.8	As a user; I want to copy community projects to my workspace.	S
7	Profile Management	7.1	As a user; I want to edit my profile.	S
		7.2	As a user; I want to view public projects on profiles.	S

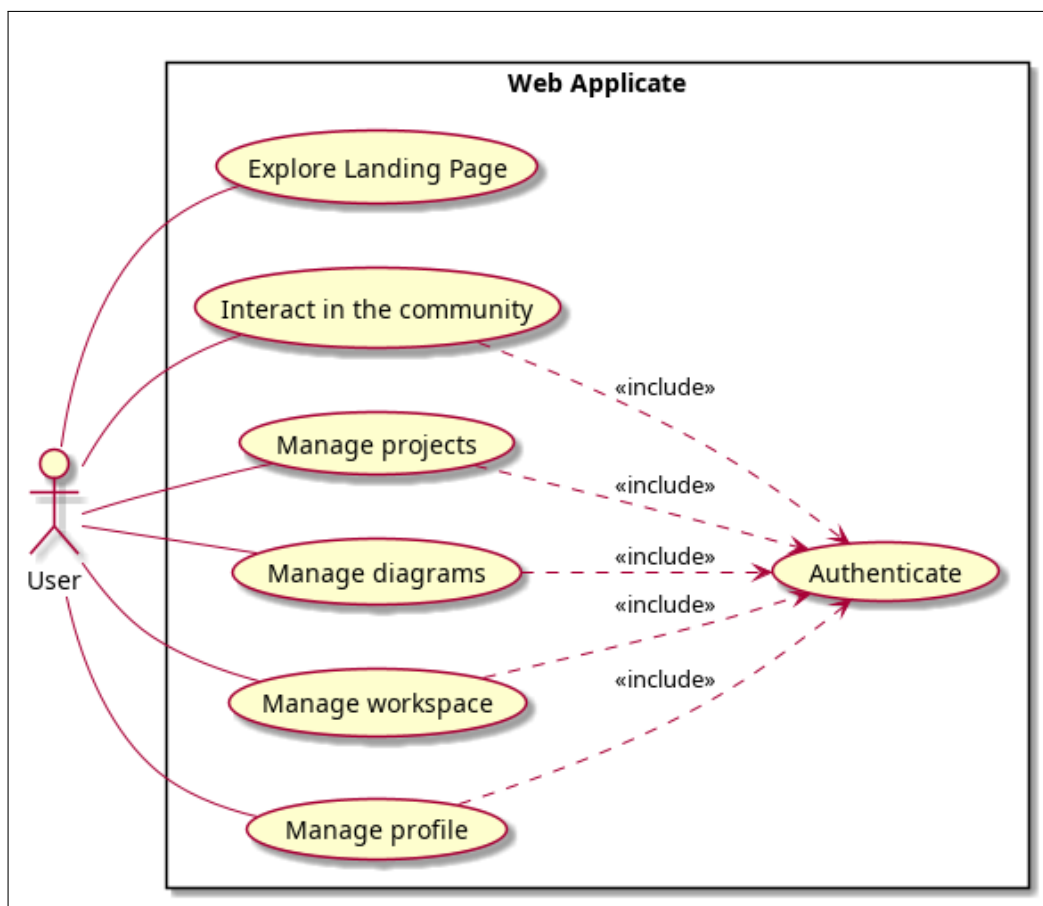### 3.3.3   Global Use Case Diagram

Figure 3.1: Global Use Case Diagram

3.3.4   Sprint Planning

The project is organized into six strategic sprints, each focusing on specific functional areas and building upon previous deliverables. The total project duration is designed to fit within 3.5 months (14 weeks) with efficient resource allocation and parallel development activities.

Sprint	Focus Area	Backlog Features	Weeks
0	Infrastructure Setup	N/A	2
1	Authentication and Landing Page	1,2	3
2	Project Management	3	3
3	Diagram and Project Management	4,5	3
4	Community Interaction and Profiles	6 ,7	3

Table 3.3: Scrum Sprint Planning with Estimated Durations

3.3.5   Class Diagram

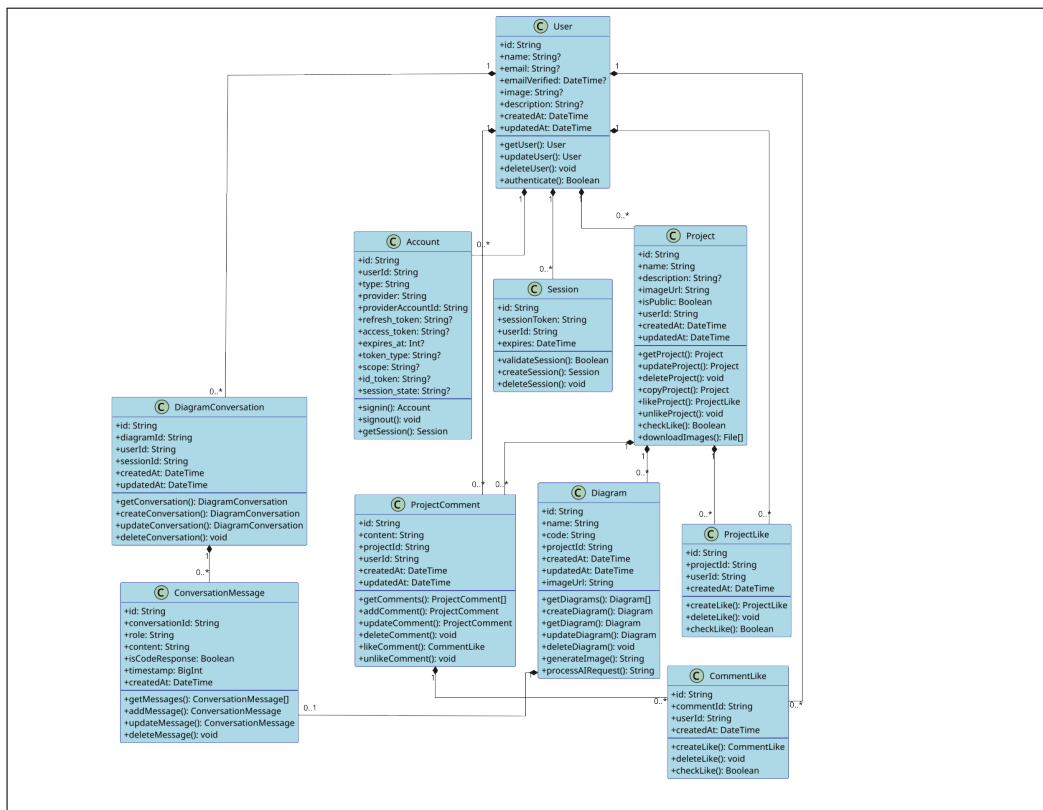The database implements a normalized schema supporting core application functionality:

Figure 3.2: Class Diagram
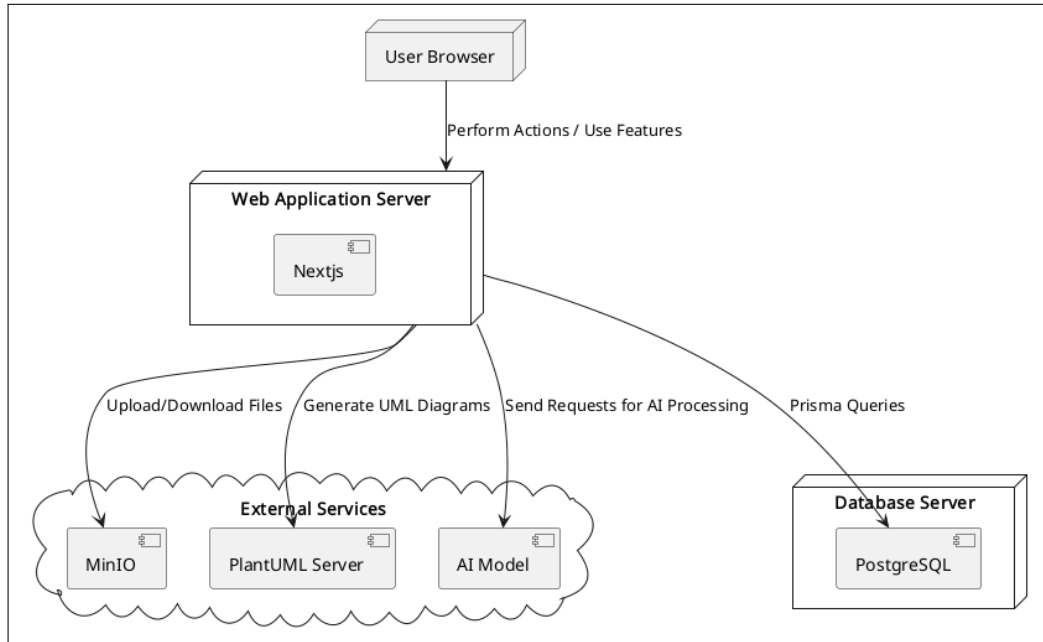
3.4 System Architecture

3.4.1 Deployment Overview

Figure 3.3: Deployment Architecture

3.4.2 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 3.4: Core technology stack with icons

3.5 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 4

Study and Implementation of Sprint 0: Infrastructure Setup

4.1 Introduction

Sprint 0 establishes a robust, scalable infrastructure foundation using Docker containerization. This sprint creates the development environment and core services supporting the entire application ecosystem, ensuring consistency across environments and facilitating deployment.

4.2 Sprint Objectives

1. Database Setup (PostgreSQL with persistent storage)

2. Object Storage (MinIO S3-compatible service)

3. Diagram Service (PlantUML server)

4. Web Application (Next.js with TypeScript/Tailwind CSS)

5. ORM Configuration (Prisma with PostgreSQL)

6. Environment Configuration and Integration Testing

4.3 Technology Stack Implementation

4.3.1 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 15	Full-stack React framework with TypeScript support
ORM	Prisma	Type-safe database access layer
Containerization	Docker Compose	Service orchestration and environment consistency

Table 4.1: Infrastructure services and technologies

4.4 Infrastructure Deliverables

4.4.1 Docker Compose Configuration

The complete infrastructure is orchestrated through Docker Compose:

Listing 4.1: 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
```

4.4.2   Application Container Configuration

Listing 4.2: 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"]
```

4.4.3 Environment Configuration

Essential environment variables for secure operation:

Listing 4.3: 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=*********************
```

4.5 Sprint Retrospective

4.5.1 Key Achievements

- Complete containerized infrastructure with service integration

- Database schema design and Prisma ORM integration

- Secure environment configuration and networking

4.5.2 Challenges Resolved

- **Security Management**: Implemented secure environment variable handling

- **Schema Synchronization**: Coordinated Prisma migrations in containers

4.5.3 Future Enhancements

- Health checks implementation for all services

- Logging and monitoring solutions integration

- Automated backup systems for database

- CI/CD pipeline preparation

4.6 Conclusion

Sprint 0 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.

Chapter 5

Study and Implementation of Sprint 1: Authentication & Landing Page

5.1 Introduction

Sprint 1 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.

5.2 Sprint Planning

5.2.1 Objectives

- Implement secure OAuth authentication (Google, GitHub)

- Develop cross-device session management

- Create responsive landing page

5.2.2 Backlog Items

ID	Feature	Sub-ID	User Story	Priority
1	Authentication	1.1	As a user; I want to authenticate using my Google account.	M
1	Authentication	1.2	As a user; I want to authenticate using my GitHub account.	M
1	Authentication	1.3	As a user; I want to stay authenticated across multiple devices.	S
1	Authentication	1.4	As a user; I want to log out from my account.	M
2	Explore Landing Page	2.1	As a user; I want to explore the landing page so that I can understand the platform's features and benefits.	M

Table 5.1: User Stories Requirements Table

5.3 System Analysis

5.3.1 Use Case Overview
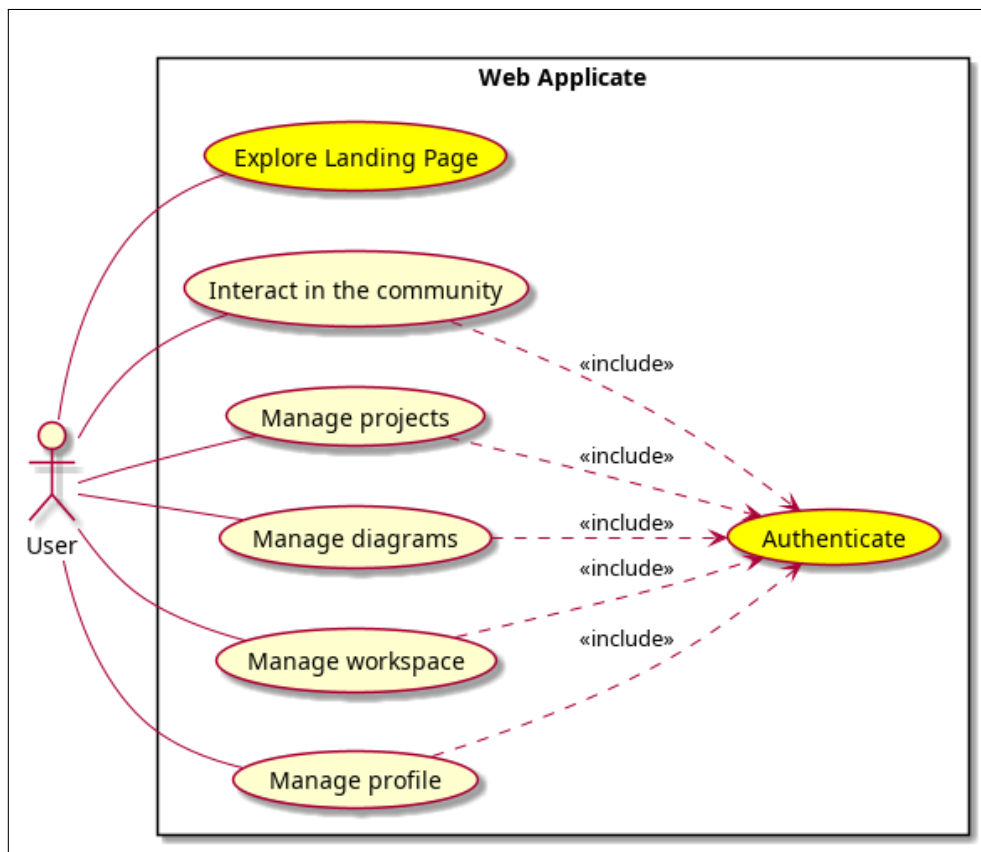
Figure 5.1: Sprint 1 Use Case Diagram

5.3.2   Authentication Use Cases
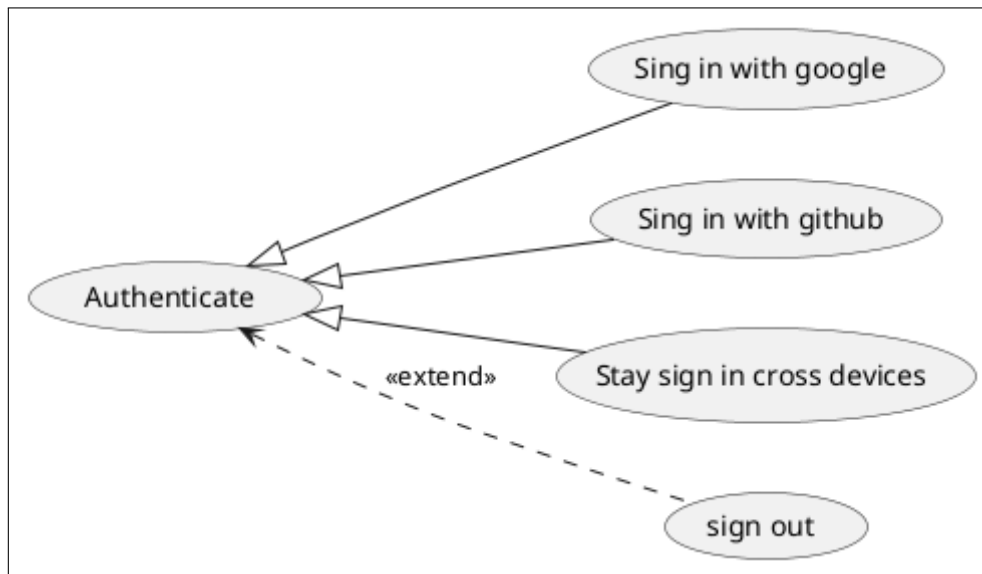
Figure 5.2: Refined Authentication Use Case

5.3.2.1   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.

5.4   Sequence Diagrams

Figure 5.3: OAuth Authentication Sequence

5.5 Implementation Results

5.5.1 Landing Page
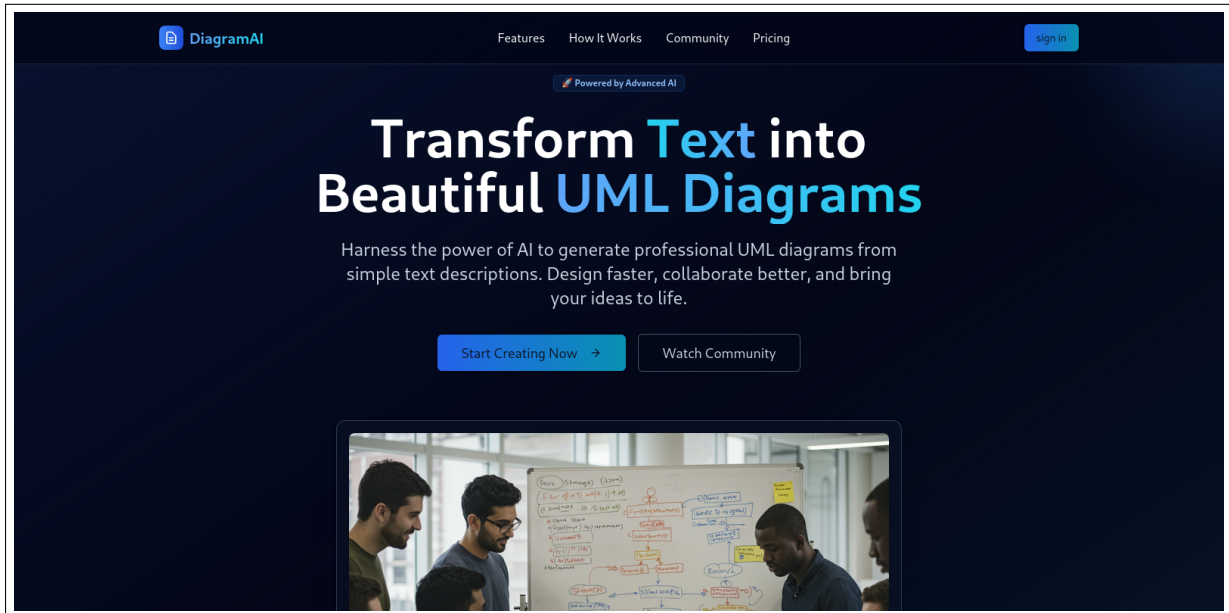
Figure 5.4: Responsive Landing Page

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

5.5.2 Authentication Interface
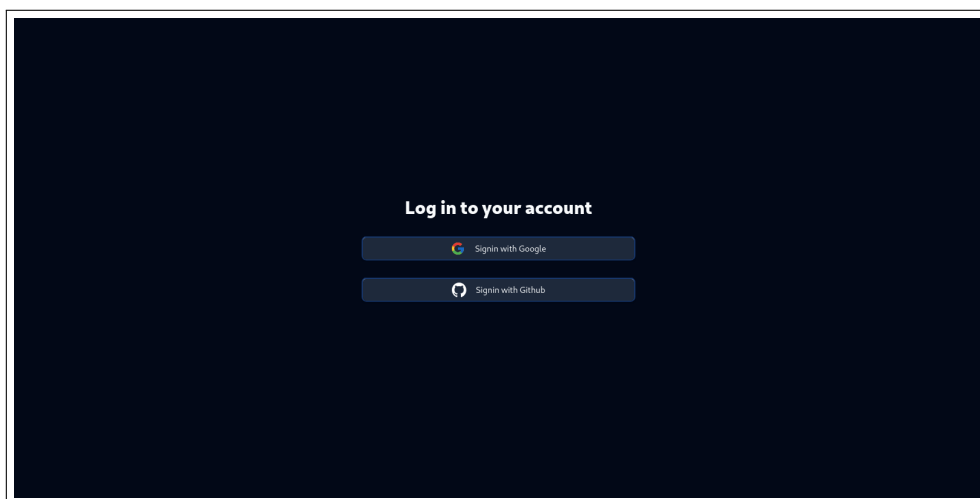
Figure 5.5: OAuth Sign-In Interface

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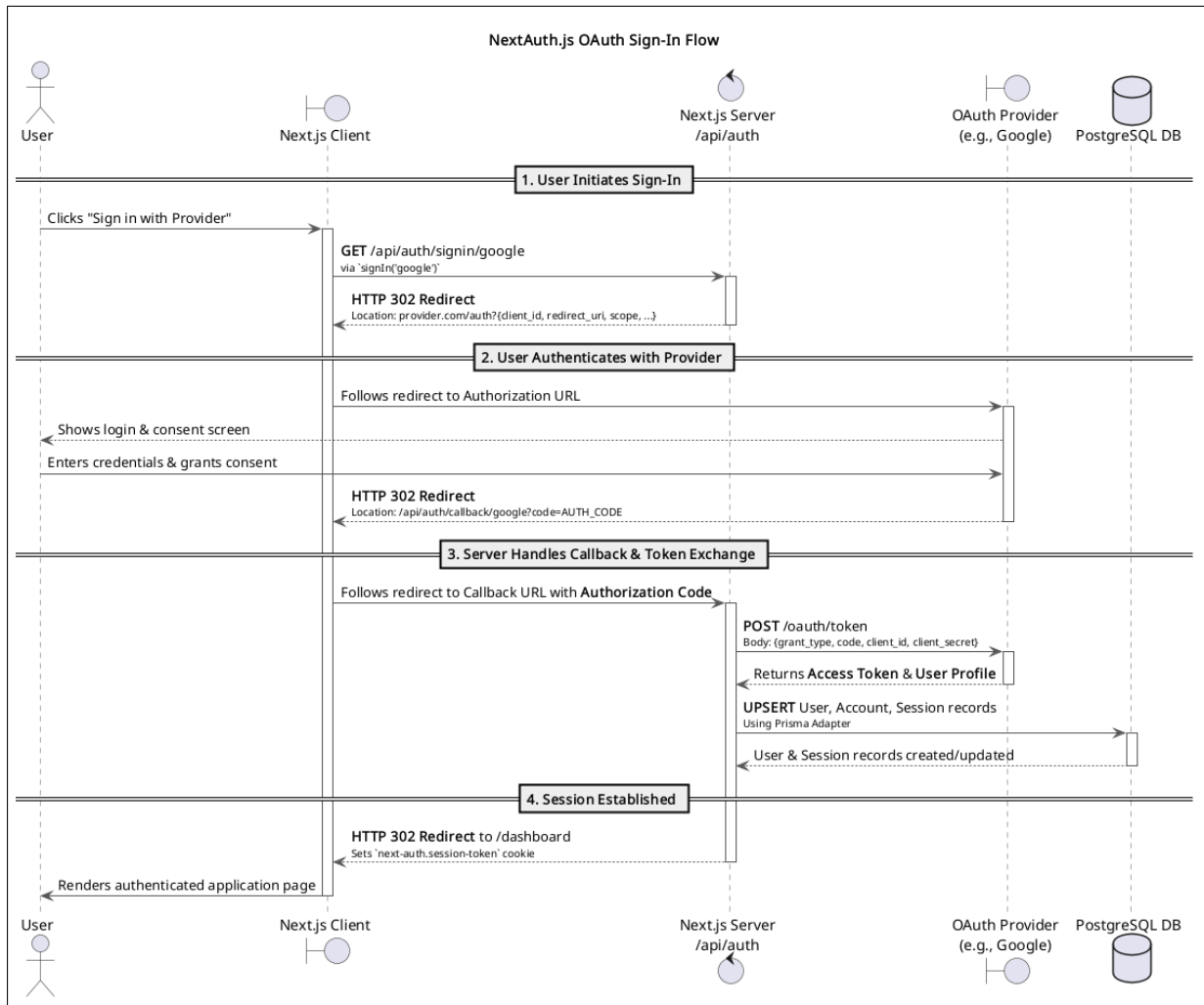
5.6 Sprint Retrospective

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

5.7 Conclusion

Sprint 1 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 6

Study and Implementation of Sprint 2: Project Management

6.1 Introduction

Sprint 2 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.

6.2 Sprint Planning

6.2.1 Objectives

Sprint 2 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

6.2.2 Sprint Backlog

Table 6.1: Manage Projects User Stories Requirements
Table

ID	Feature	Sub-ID	User Story	Priority
3	Manage Projects	3.1	As a user; I want to create a new project so that I can organize my diagrams.	M
		3.2	As a user; I want to view my project details.	M
		3.3	As a user; I want to update project details.	M
		3.4	As a user; I want to delete a project.	M
		3.5	As a user; I want to download project diagrams as images in a compressed ZIP file.	S
		3.6	As a user; I want to share my project with others.	S

6.3 Analysis and Design
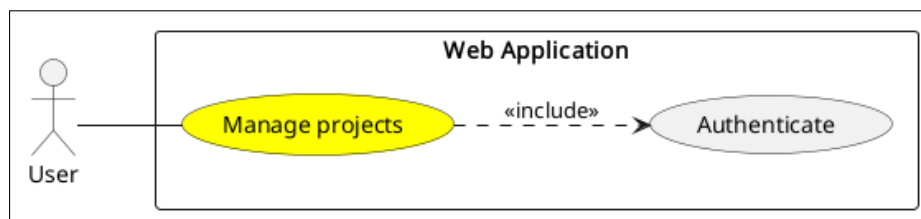
6.3.1 Use Case Analysis
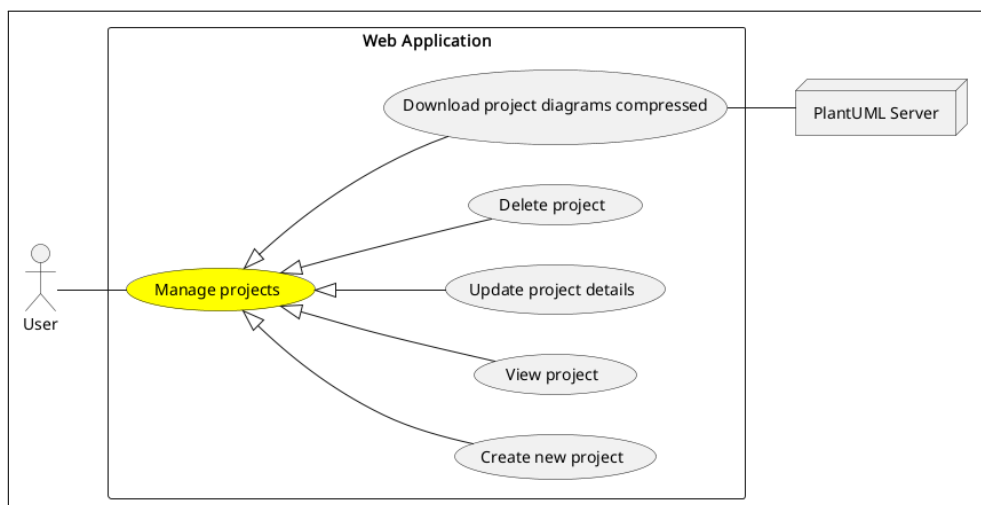

Figure 6.1: Sprint 2 Use Case Diagram

Figure 6.2: Refined Project Management Use Cases

6.3.2 Key Use Case Specifications

6.3.2.1 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.

6.3.2.2 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.

6.3.2.3 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.

6.3.2.4 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.

6.3.2.5 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.

6.4 Sequence Diagrams

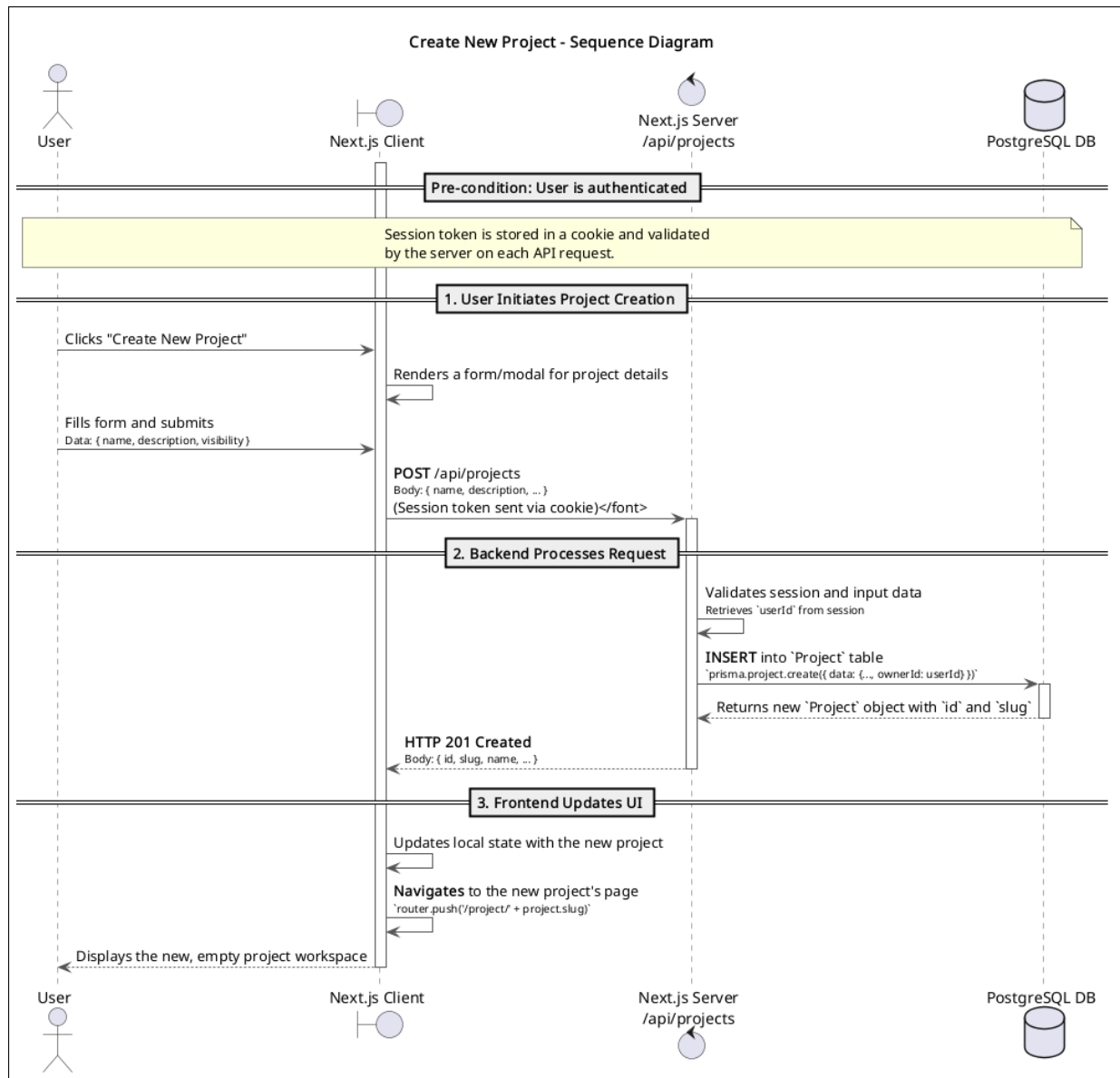The following diagrams illustrate system component interactions:

Figure 6.3: Create Project Sequence
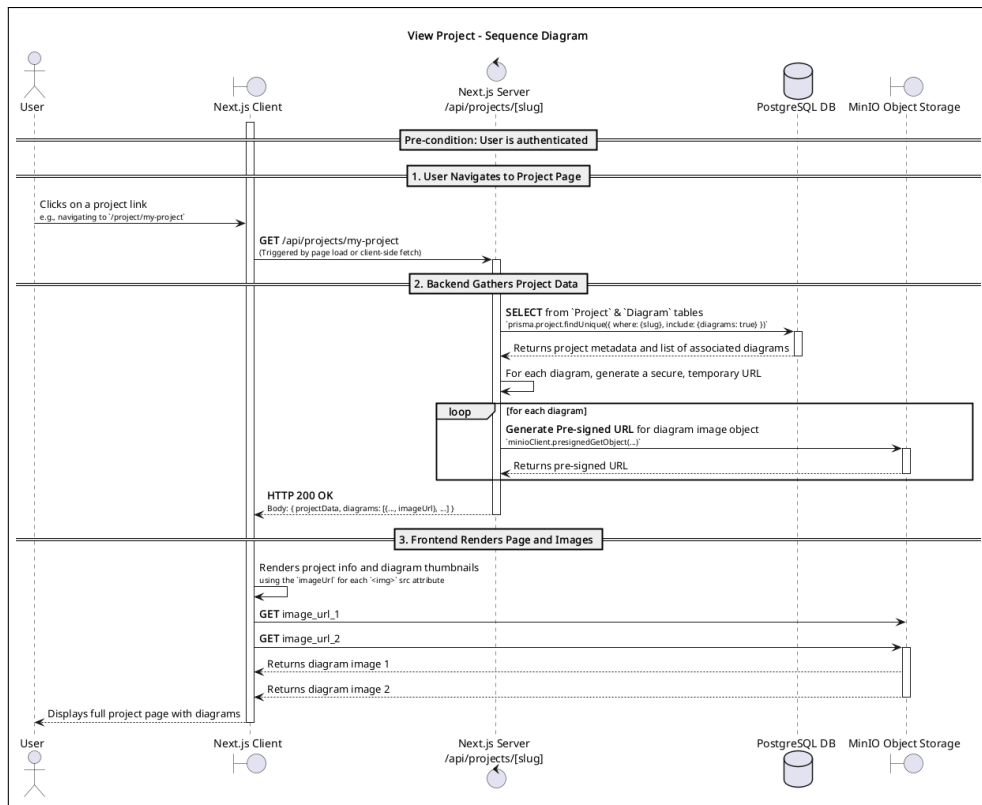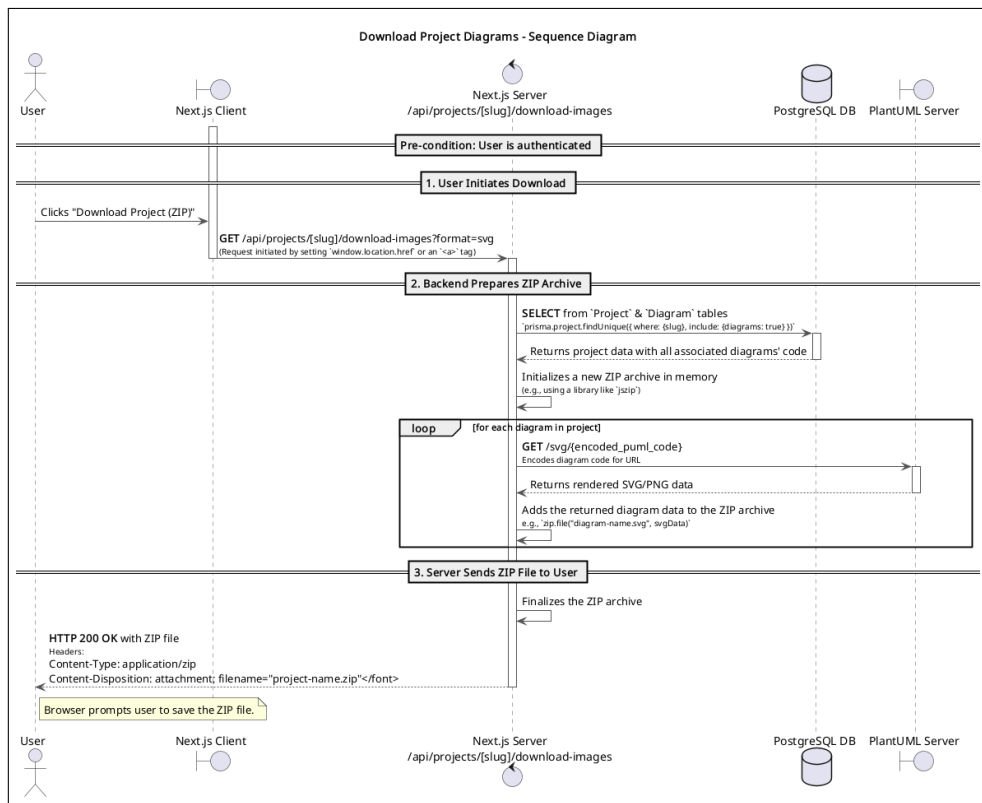
Figure 6.4: View Project Sequence

Figure 6.5: Download Project Diagrams Sequence

6.5   Implementation Results

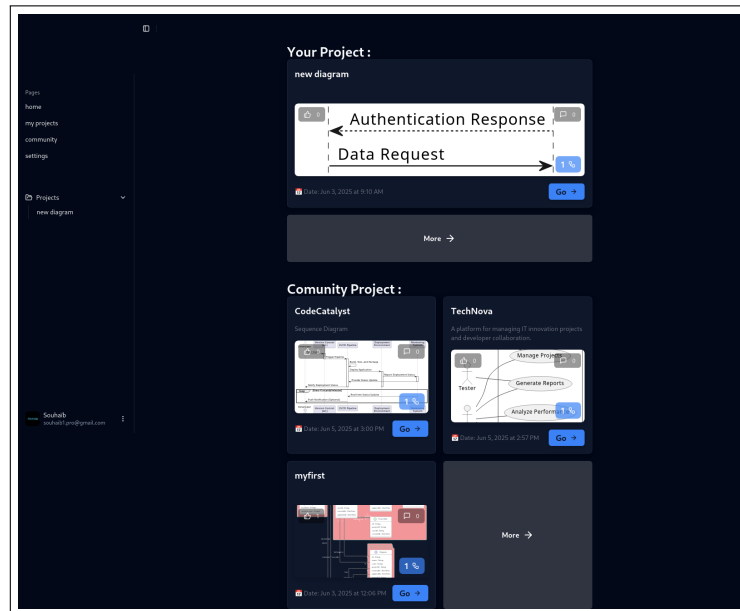### 6.5.1   User Interface Screenshots

Figure 6.6: Home page with project overview
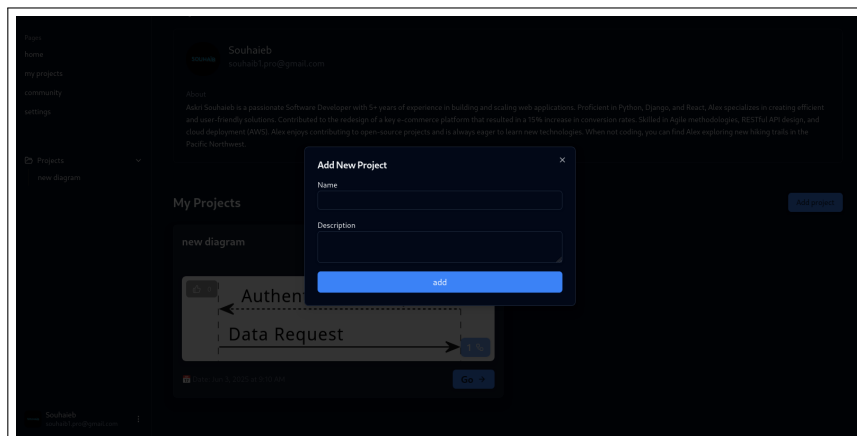
Figure 6.7: Project creation interface
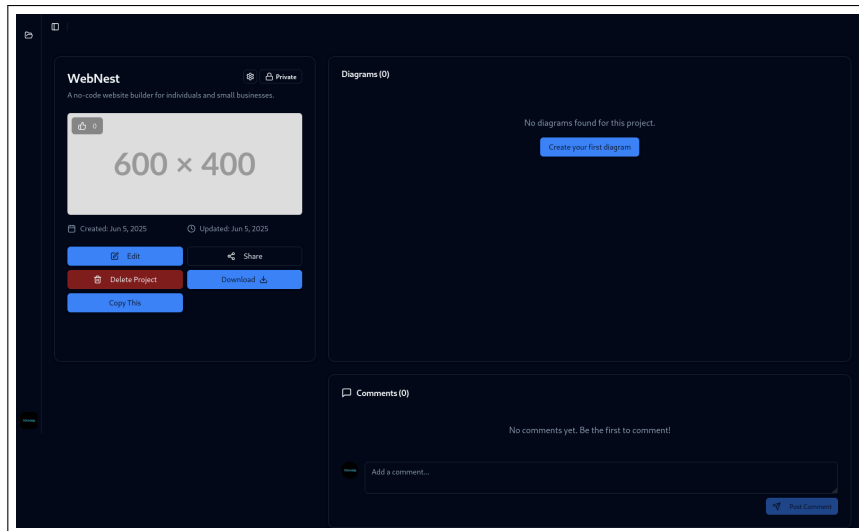
Figure 6.8: Project details and management

6.6 Sprint Retrospective

6.6.1 Achievements

- Complete CRUD operations implementation

- Robust export functionality with multiple formats

- Intuitive user interfaces with validation

- Strong technical execution and requirement analysis

6.6.2 Areas for Improvement

- Enhanced error handling mechanisms

- Performance optimization for large projects

- Extended sharing capabilities

6.7 Conclusion

Sprint 2 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 7

Study and Implementation of Sprint 3: Diagram & Workspace Management

7.1 Introduction

Sprint 3 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.

7.2 Sprint Planning

7.2.1 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.

ID	Feature	Sub-ID	User Story	Priority
4	Manage Diagrams	4.1	As a user; I want to create a new diagram.	M
		4.2	As a user; I want to view my diagram.	M
		4.3	As a user; I want to update diagram details.	M
		4.4	As a user; I want to delete a diagram.	M
5	Manage Workspace	5.1	As a user; I want to edit diagram code in an interactive editor.	M
		5.2	As a user; I want to chat with an AI model to edit diagram code.	C
		5.3	As an AI system; I need to respond to user requests and help edit diagram code.	C
		5.4	As a PlantUML Server; I need to render diagram code into diagram images.	M

Table 7.1: Manage Diagrams and Workspace User Stories Requirements Table

7.2.2 Sprint Backlog

7.3 System Analysis

7.3.1 Use Case Overview
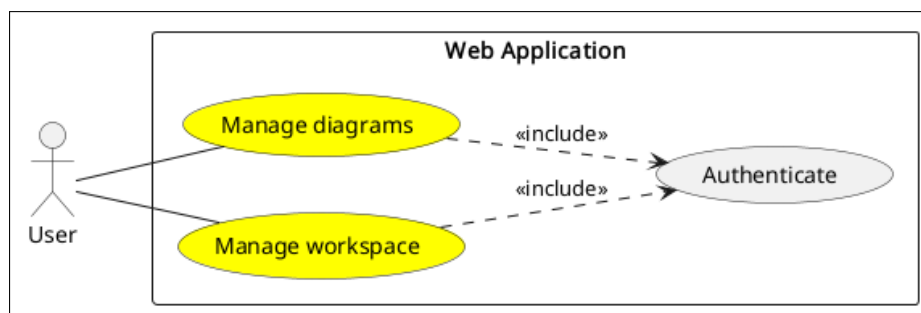
Figure 7.1: Sprint 3 Use Case Diagram

7.3.2 Core Features

7.3.2.1 Diagram Management
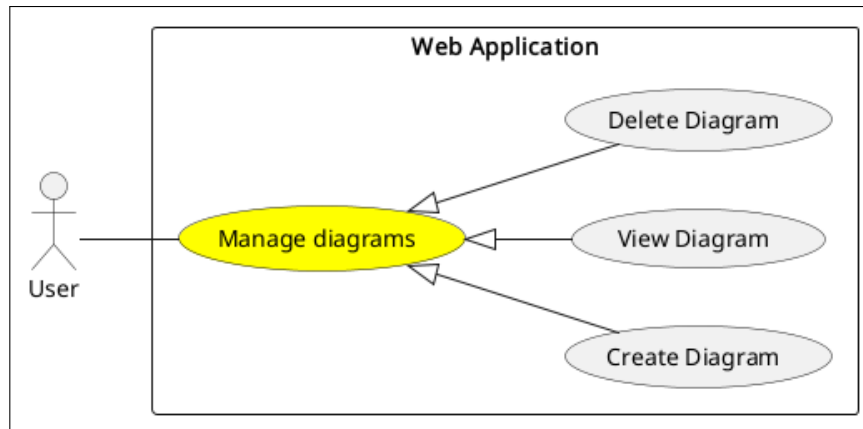
Figure 7.2: Diagram Management Use Cases

Key operations include:

- **Create Diagram**: User initiates new diagram creation with name and type selection

- **View Diagram**: Interactive editor with real-time preview and validation

- **Delete Diagram**: Secure deletion with confirmation dialog

7.3.2.2 Workspace Management
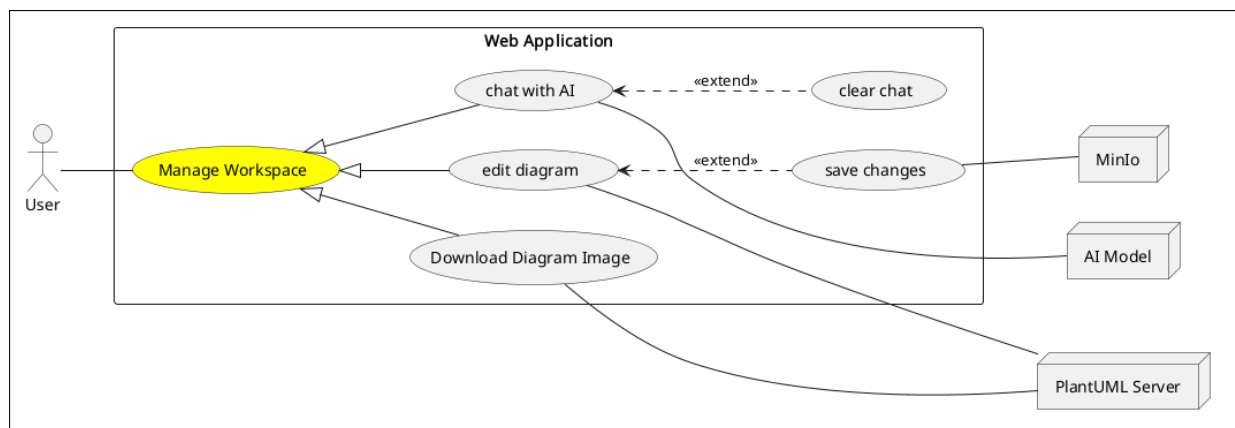
Figure 7.3: Workspace Management Use Cases

Core workspace features:

- **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

7.4   Sequence Diagrams

7.4.1   Diagram Creation Process
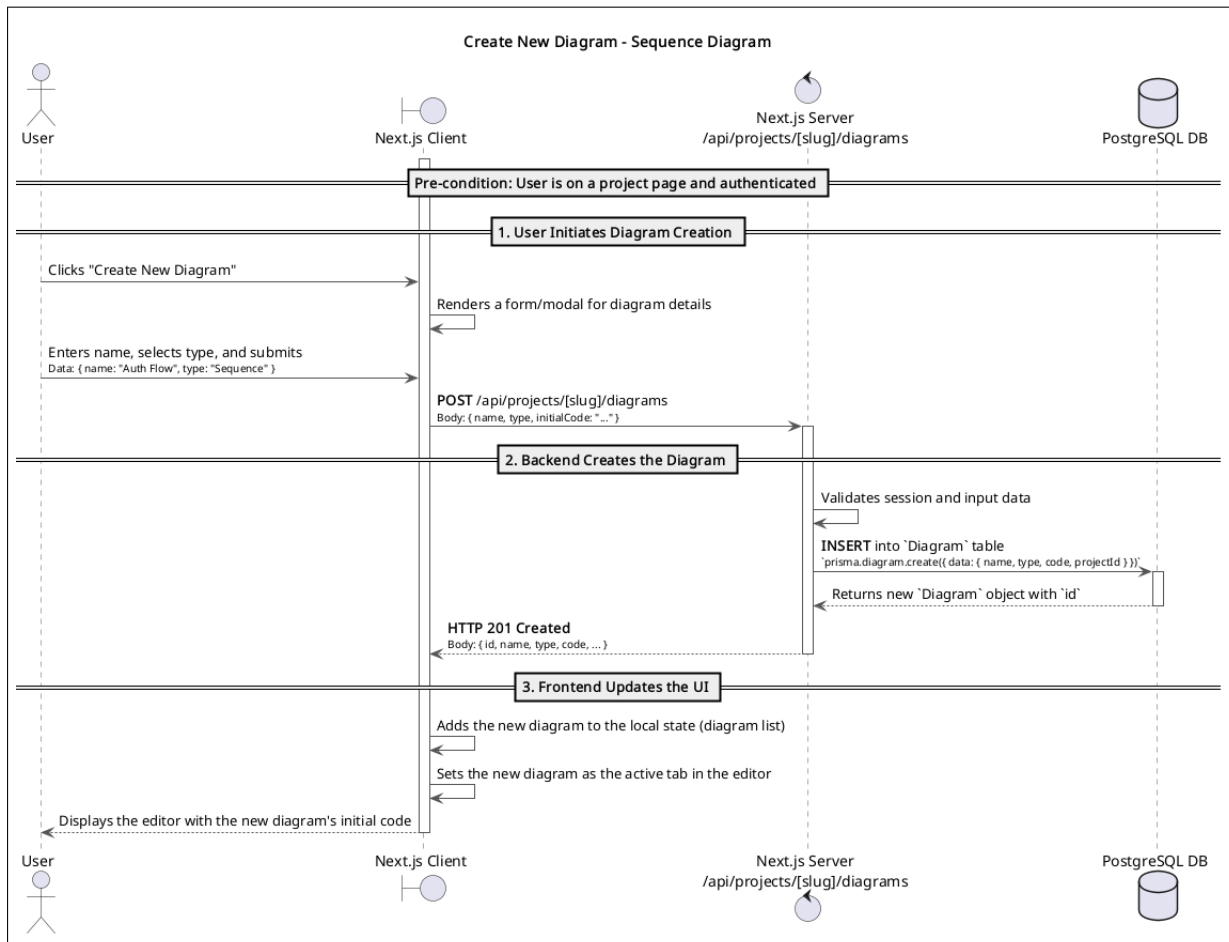
Figure 7.4: Create New Diagram Sequence

7.4.2   AI-Assisted Editing
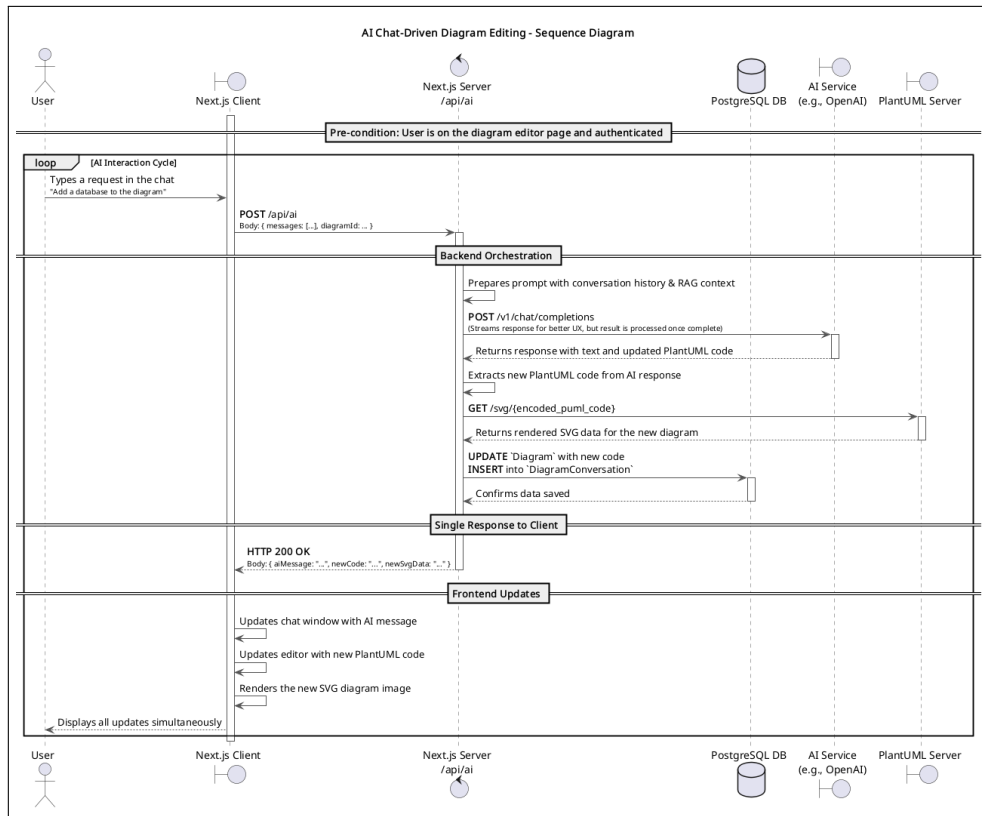
Figure 7.5: AI Chat Integration Sequence

7.5   Implementation Results

7.5.1   Core Interfaces
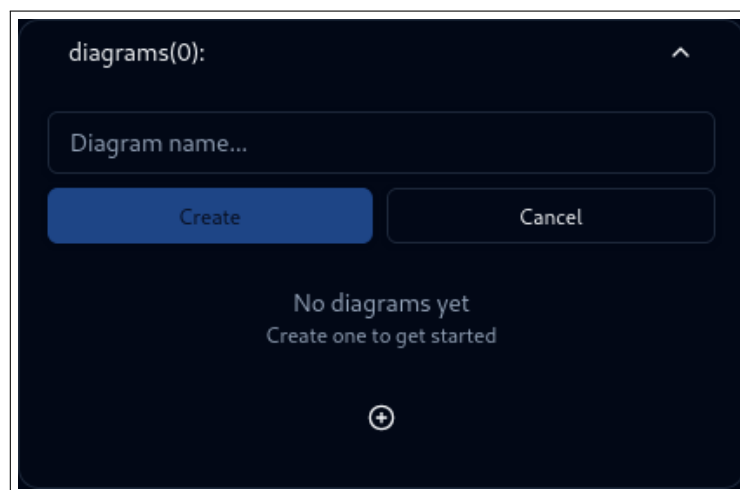
Figure 7.6: Diagram Creation Interface

The diagram creation interface provides intuitive name specification, type selection, and project initialization capabilities.
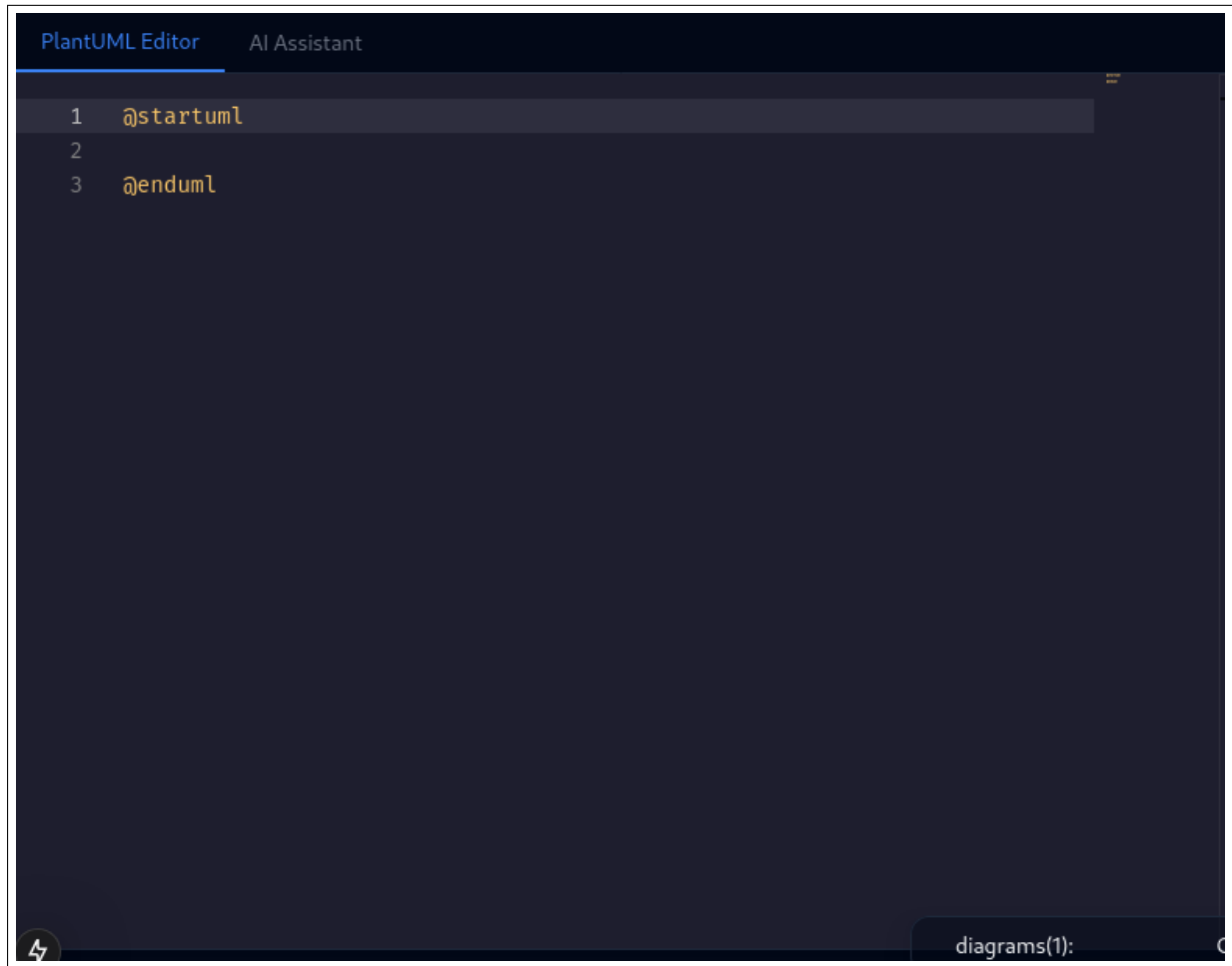
Figure 7.7: Interactive Code Editor with Real-time Preview

The interactive workspace features syntax highlighting, real-time validation, and seamless preview integration for enhanced productivity.
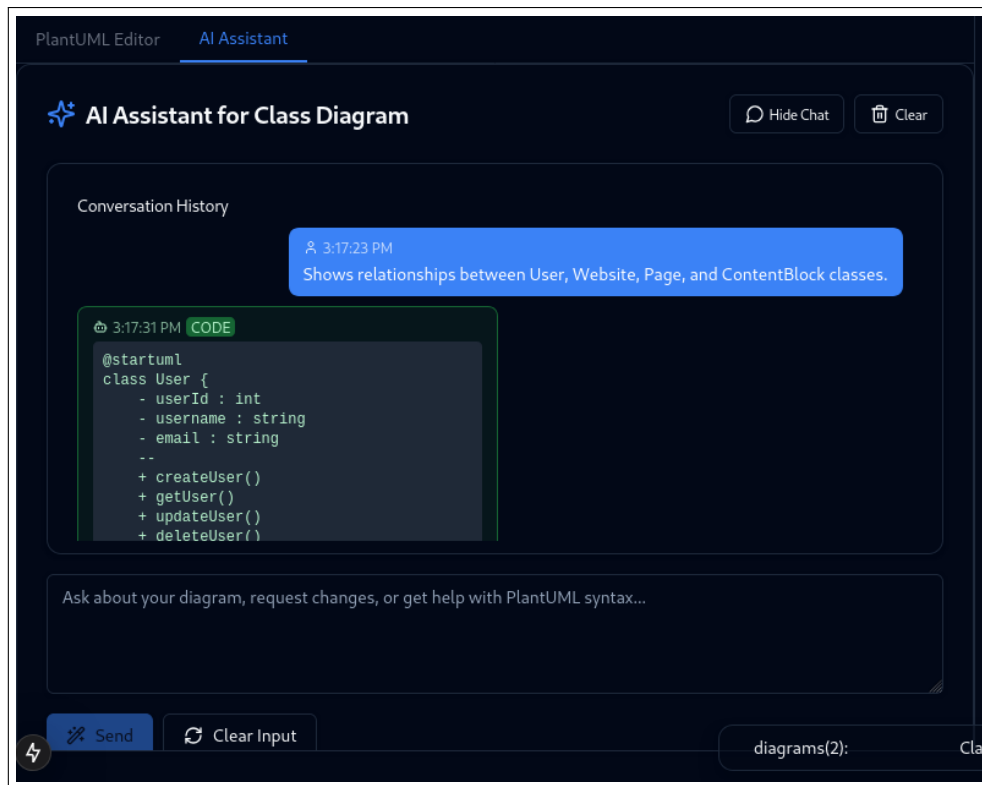
Figure 7.8: AI Assistant Integration

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

7.5.2   Workspace Environment
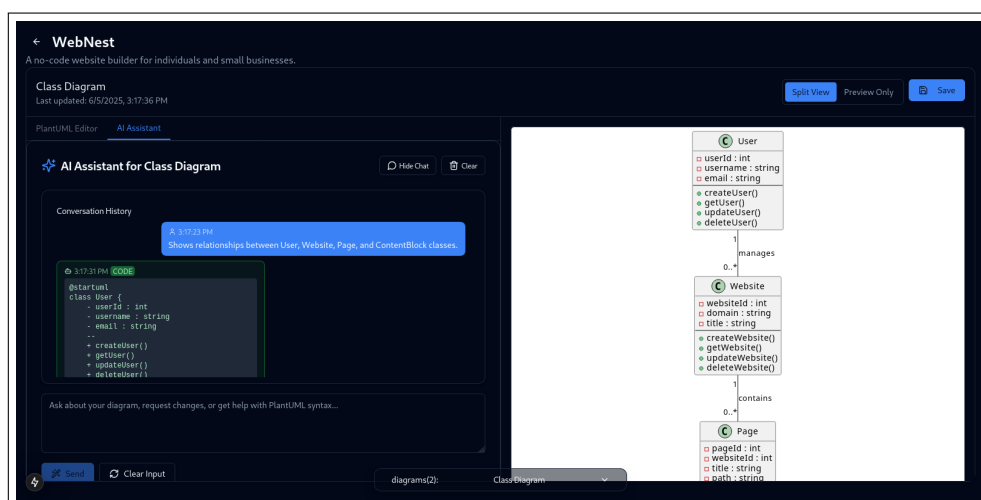
Figure 7.9: Split-View Workspace - Secondary View

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

7.6 Sprint Retrospective

7.6.1 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

7.6.2 Challenges

- AI service integration complexity

- Performance optimization for large diagrams

7.7 Conclusion

Sprint 3 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 8

Study and Implementation of Sprint 4: Community Interaction & Profile Management

8.1 Introduction

Sprint 4 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.

8.2 Sprint Planning

8.2.1 Objectives and Backlog

Sprint 4 objectives include developing community exploration, project interaction features, comment management, project copying functionality, and comprehensive profile management.

ID	Feature	Sub-ID	User Story	Priority
6	Community Interaction	6.1	As a user; I want to explore the community.	S
		6.2	As a user; I want to comment on projects.	C
		6.3	As a user; I want to like/unlike projects.	C
		6.4	As a user; I want to share projects.	C
		6.5	As a user; I want to update my comments.	C
		6.6	As a user; I want to delete my comments.	C
		6.7	As a user; I want to like/unlike comments.	C
		6.8	As a user; I want to copy community projects to my workspace.	S
7	Profile Management	7.1	As a user; I want to edit my profile.	S
		7.2	As a user; I want to view public projects on profiles.	S

Table 8.1: Community Interaction and Profile Management User Stories Requirements Table

8.3   System Analysis

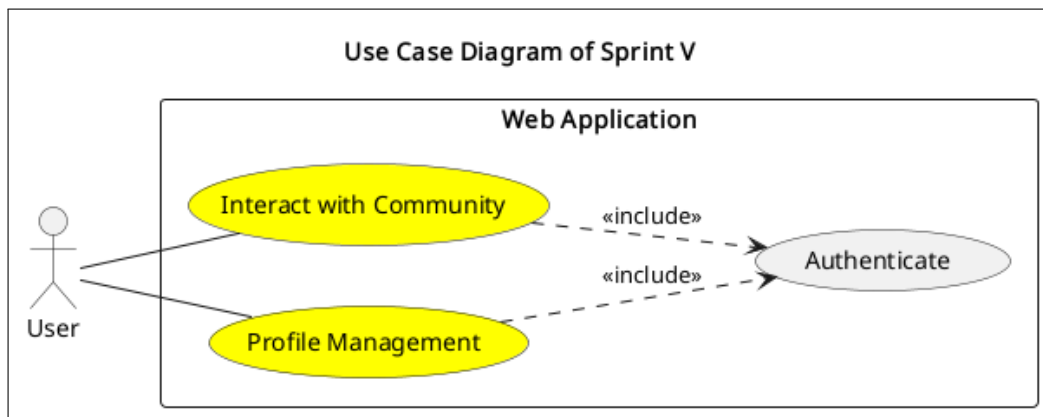### 8.3.1   Use Case Overview

Figure 8.1: Use Case Diagram for Sprint V
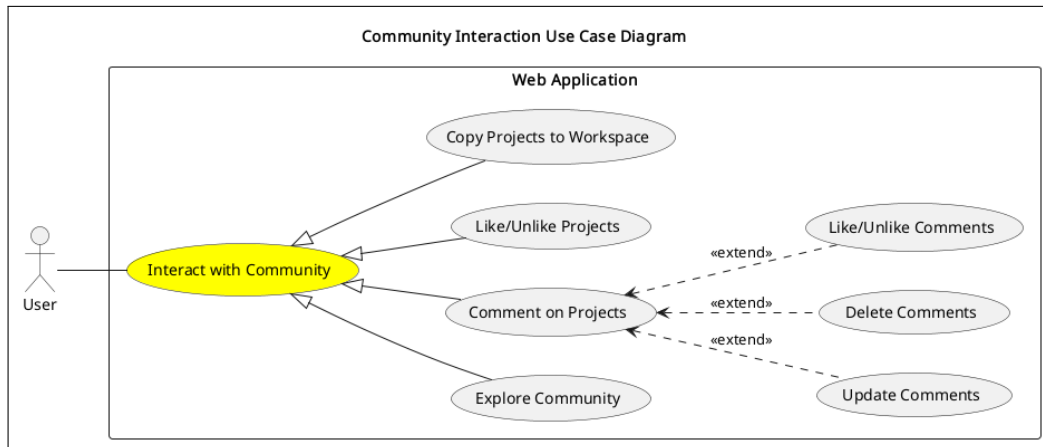
8.3.2 Community Interaction Features

Figure 8.2: Community Interaction Use Cases

8.3.2.1 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.

8.3.3 Profile Management Features
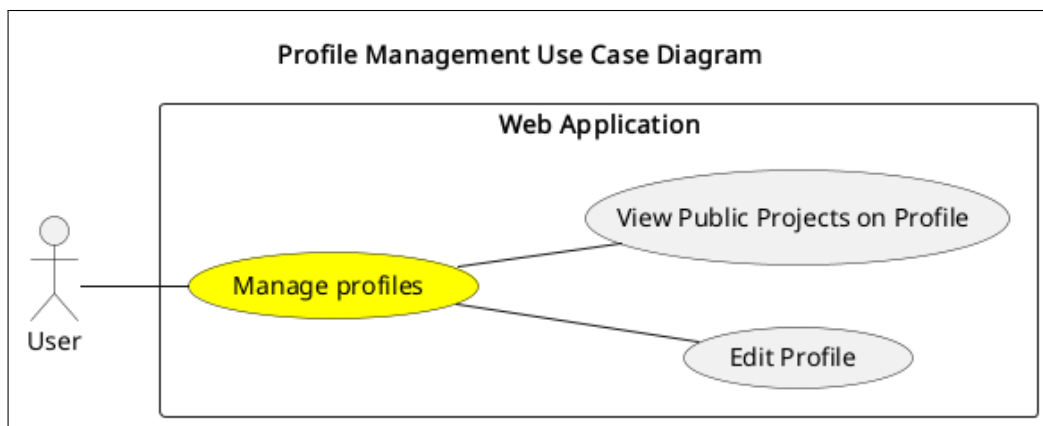
Figure 8.3: Profile Management Use Cases

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.

8.4 Sequence Diagrams

Figure 8.4: Community Exploration Flow

Figure 8.5: Project Commenting Flow

51

Figure 8.6: Profile Editing Flow

8.5 Implementation Results

8.5.1 Community Features

Figure 8.7: Comment System Implementation
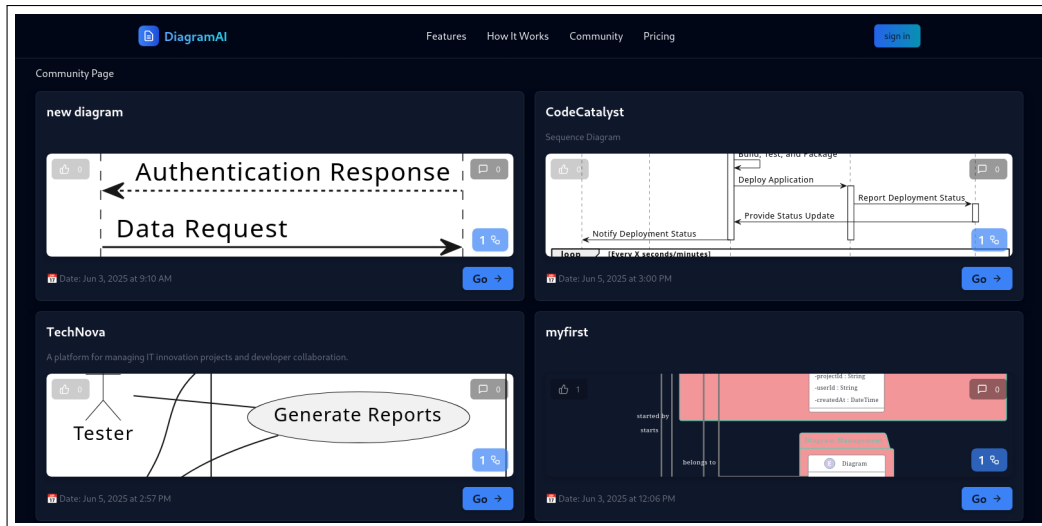
Figure 8.8: Community Exploration Interface

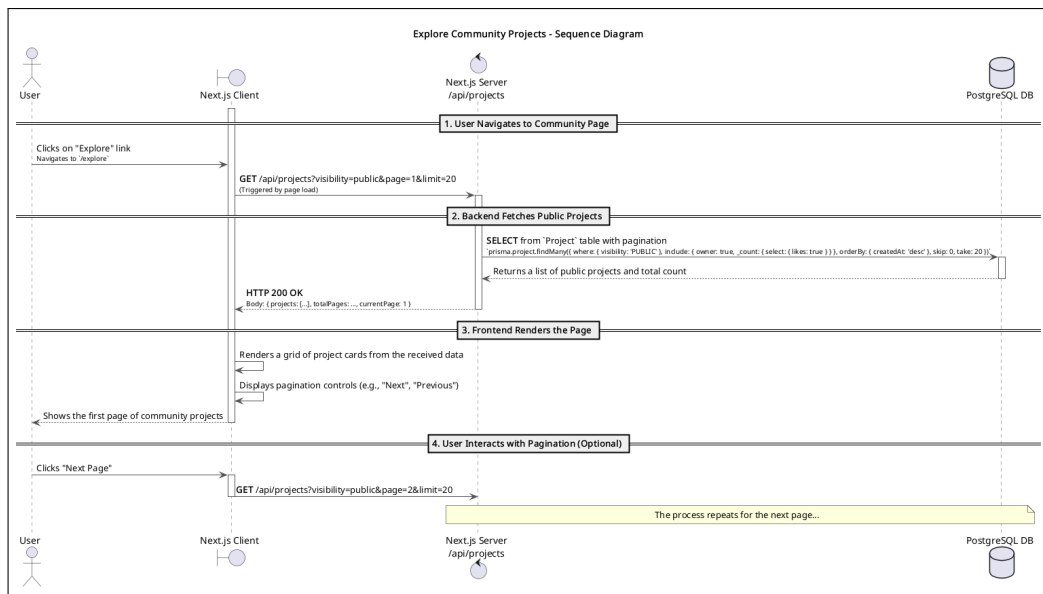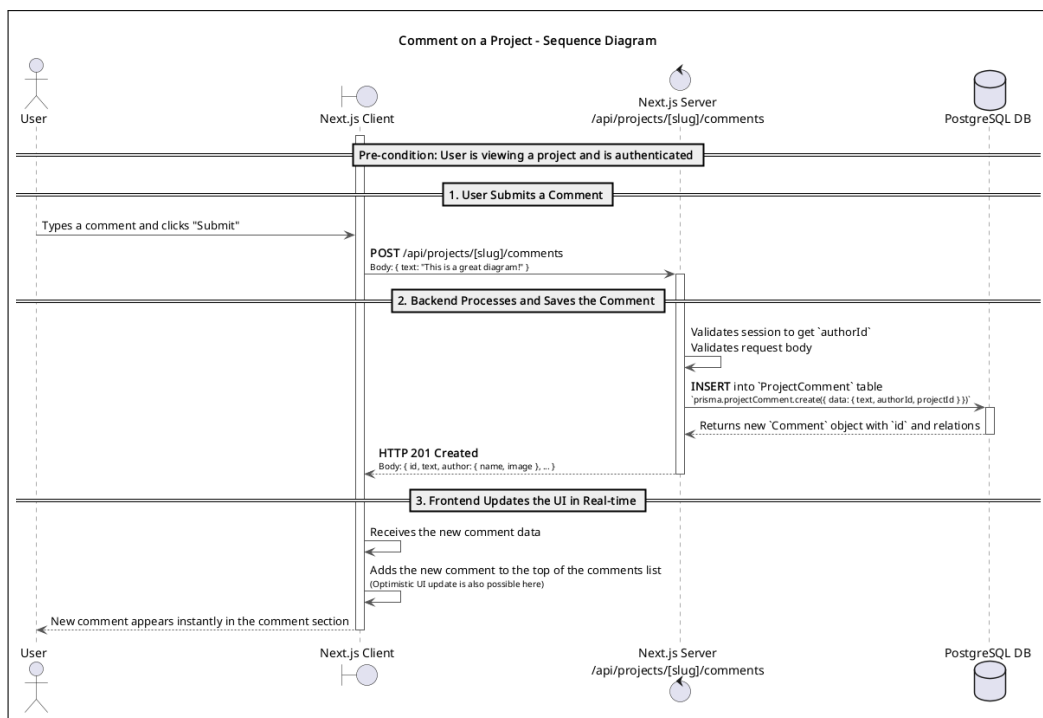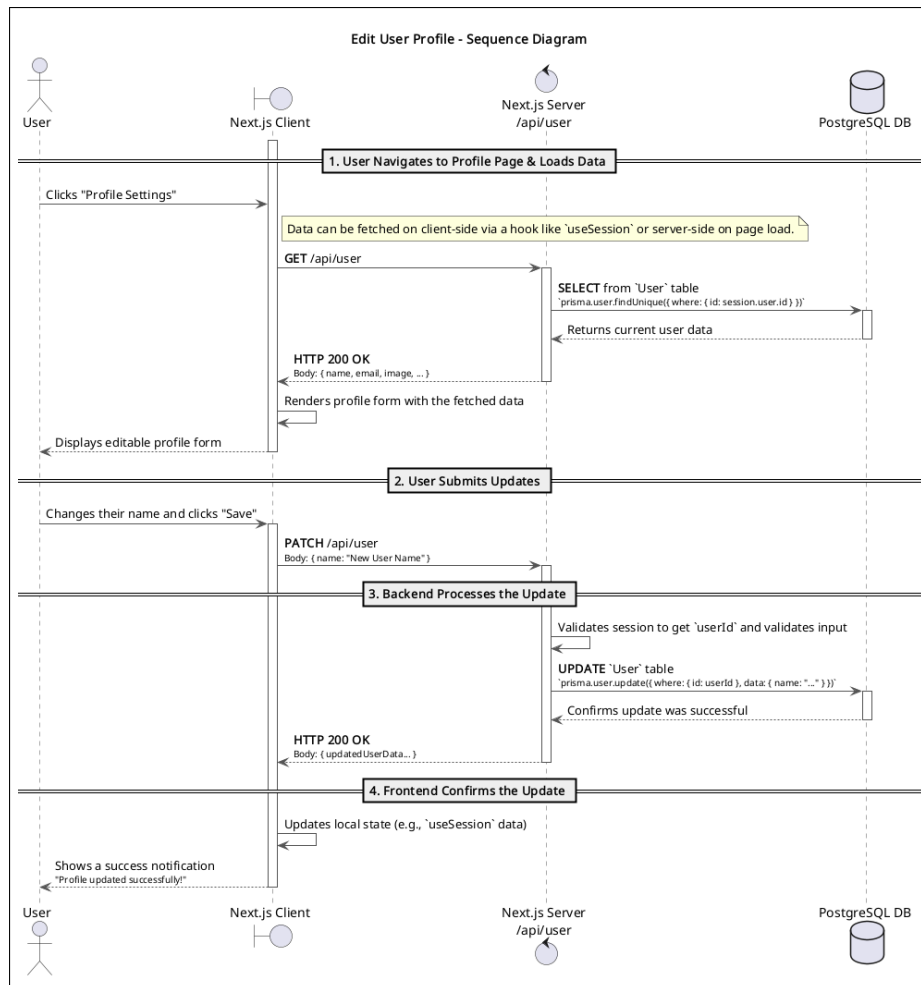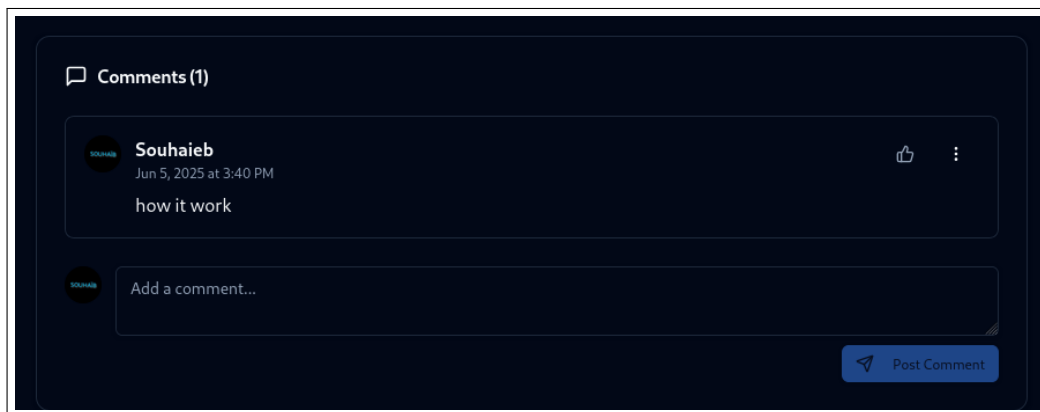The comment system demonstrates comprehensive management including creation, editing, deletion, and interaction features for meaningful project discussions.

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

8.6   Sprint Retrospective

8.6.1   Achievements

- Successfully implemented comprehensive community interaction features

- Smooth integration of profile management features

8.6.2   Future Actions

- Implement real-time notifications for community engagement

- Enhance mobile user experience across all features

8.7   Conclusion

Sprint 4 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.

Chapter 9

General Conclusion

9.1 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. 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.

9.2 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.

9.3   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] ChatUML. *ChatUML – AI-Powered UML Diagram Generation.* Retrieved April 3, 2025, from https://chatuml.com/

[2] Diagramming AI. *Diagramming AI – Intelligent Diagram Creation.* Retrieved April 5, 2025, from https://diagrammingai.com/

[3] LangChain. *LangChain – Build LLM-powered Applications.* Retrieved April 8, 2025, from https://www.langchain.com

[4] MinIO. *MinIO – High Performance Object Storage.* Retrieved April 10, 2025, from https://min.io

[5] PlantUML. *PlantUML – Open-Source UML Tool.* Retrieved April 12, 2025, from https://plantuml.com

[6] ShadCN UI. *ShadCN UI – Beautifully Designed UI Components.* Retrieved April 15, 2025, from https://ui.shadcn.com

[7] Docker. *Docker – Empowering App Development for Developers.* Retrieved April 15, 2025, from https://www.docker.com

[8] NextAuth.js. *NextAuth.js – Authentication for Next.js.* Retrieved April 15, 2025, from https://next-auth.js.org

[9] LaTeX Project. *LaTeX – A Document Preparation System.* Retrieved April 25, 2025, from https://www.latex-project.org

[10] Vercel. *Next.js – The React Framework.* Retrieved April 28, 2025, from https://nextjs.org

[11] Prisma. *Prisma – Next-generation ORM for Node.js.* Retrieved May 2, 2025, from https://www.prisma.io

[12] Tailwind CSS. *Tailwind CSS – Rapidly Build Modern Websites.* Retrieved May 5, 2025, from https://tailwindcss.com

BIBLIOGRAPHY

[13] Git SCM. *Git – Distributed Version Control System.* Retrieved May 8, 2025, from
https://git-scm.com

[14] Node.js. *Node.js – JavaScript Runtime.* Retrieved May 8, 2025, from https://nodejs.
org

[15] React. *React – A JavaScript library for building user interfaces.* Retrieved May 8,
2025, from https://reactjs.org

[16] TypeScript. *TypeScript – JavaScript With Syntax for Types.* Retrieved May 8, 2025,
from https://www.typescriptlang.org

[17] Mozilla Firefox. *Firefox – Fast, Private & Free Web Browser.* Retrieved May 18, 2025,
from https://www.mozilla.org/firefox

[18] VSCodium. *VSCodium – Free/Libre Open Source Software Binaries of VS Code.*
Retrieved May 20, 2025, from https://vscodium.com

[19] GitHub. *GitHub – Where the World Builds Software.* Retrieved May 22, 2025, from
https://github.com

[20] Linux Kernel Organization. *The Linux Kernel Archives.* Retrieved May 25, 2025, from
https://kernel.org

[21] PostgreSQL. *PostgreSQL – The World's Most Advanced Open Source Relational
Database.* Retrieved May 28, 2025, from https://www.postgresql.org