Faculty of Informatics
Engineering Department of
Software Engineering &
Information System

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

كلية هندسة المعلوماتية
قسم هندسة البرمجيات و نظم
المعلومات

AutoTest & DocGen Manager – Intelligent System for Automated Testing and Documentation

Prepared by

Areej nooraldeen **Kamar aldiab** **Ola najibeh** **Wiaam alouni**

Supervised by

Dr. Riad Sonbol **Eng.Raghad Alhossny**

2025-2026

SUPERVISION CERTIFICATION

I certify that the preparation of this project entitled ……………………………………………………………,prepared by ……………………………………………
……………………………………………,was made under my supervision at Department of Software Engineering & Information System / Faculty of Informatics Engineering in partial fulfillment of the Requirements for the Degree of Bachelors of Software Engineering & Information System / Faculty of Informatics Engineering

Name: …………… Signature: …………….. Date:

إقرار المشرف

أشهد بأن هذا المشروع الموسوم...
....
والمعد من قبل الطلاب...
....
قد تم تحت إشرافي في قسم هندسة البرمجيات ونظم المعلومات / كلية هندسة المعلوماتية، وهو جزء من متطلبات نيل شهادة الإجازة في هندسة البرمجيات ونظم المعلومات.

الاسم: …………… التوقيع: …………… التاريخ:

ABSTRACT

Software development forms a cornerstone of the modern technological era, requiring high precision and efficiency to meet growing demands. With rapid advancements, the reliance on intelligent systems to enhance code quality and streamline testing and documentation processes has become indispensable.

The AutoTest & DocGen Manager is an intelligent system designed to enhance the software development process by integrating automated testing and comprehensive documentation. Leveraging static analysis, AI-driven reasoning through Large Language Models (LLMs), and code-to-text generation, the system automatically processes uploaded codebases.
It generates a wide range of test cases, including unit, integration, and edge tests, executes them, and provides detailed reports with code coverage metrics. Additionally, it produces precise technical documentation covering API descriptions, UML class and sequence diagrams, and natural-language explanations of classes and functions, while identifying and updating outdated documentation.
The system also offers AI-driven code summarization for complex modules and features an interactive dashboard for developers to upload projects, review generated tests, and analyze results. Furthermore, it integrates with continuous integration tools like GitHub and GitLab, enabling automatic testing and documentation with each commit, thereby improving software quality, accelerating development cycles, and boosting team efficiency.

ملخص

تطوير البرمجيات يشكل ركيزة أساسية في عصر التكنولوجيا الحديث، حيث يتطلب دقة وكفاءة عالية لتلبية الاحتياجات المتزايدة. مع التطور السريع، أصبح الاعتماد على أنظمة ذكية لتحسين جودة الكود وتسهيل عمليات الاختبار والتوثيق ضرورة لا غنى عنها.

يُعدّ AutoTest & DocGen Manager نظامًا ذكيًا مصممًا لتحسين عملية تطوير البرمجيات من خلال الجمع بين الاختبار الآلي والتوثيق الشامل. يعتمد النظام على تحليل الكود الثابت (static analysis) والاستدلال المدفوع بالذكاء الاصطناعي باستخدام نماذج اللغة الكبيرة (LLMs) وتقنية تحويل الكود إلى نصوص (code-to-text generation) لمعالجة قواعد الكود المرفوعة تلقائيًا.

يقوم النظام بإنشاء مجموعة واسعة من حالات الاختبار تشمل اختبارات الوحدة، والتكامل، والحالات الحدية، مع تنفيذها وتقديم تقارير مفصلة حول النتائج ونسبة تغطية الكود. بالإضافة إلى ذلك، يولد وثائقًا فنية دقيقة تشمل وصف واجهات البرمجة (APIs)، ومخططات UML (فئة وتسلسل)، وشروحات طبيعية للفئات والدوال، مع اكتشاف الوثائق المتجاوزة وتحديثها عند حدوث تغييرات في الكود. يدعم النظام أيضًا الملخصات الذكية للكود (AI-driven code summarization) لتوضيح الوحدات المعقدة، ويوفر لوحة تحكم تفاعلية تتيح للمطورين رفع المشاريع، استعراض الاختبارات، وتحليل النتائج.

كما يتكامل مع أنظمة التكامل المستمر مثل GitHub وGitLab لضمان تشغيل الاختبارات والتوثيق تلقائيًا بعد كل التزام (commit)، مما يساهم في تعزيز جودة البرمجيات، تسريع التطوير، ودعم الفرق التنموية في تحقيق كفاءة أعلى

TABLE OF CONTENT

List of Tables

List of Figures

Chapter 1 Introduction

1. Introduction:

This chapter introduces the AutoTest & DocGen Manager project. We will outline the core challenges and motivations driving the development of this intelligent system, define its key objectives and goals, and provide a structural overview of both the report organization and the system's architecture.

2. Problem Definition:

Documentation in software is essential for understanding and maintaining code. Good documentation helps new developers understand the project quickly and enables the team to work more efficiently.

However, in reality, developers face many problems:
- Outdated Documentation: When code is changed, developers often forget to update the documentation, so the documentation no longer reflects the actual code
- Incomplete Documentation: Many projects lack complete documentation for APIs or important functions
- Missing Visual Diagrams: Absence of UML diagrams that help understand the project structure
- Time Waste: Developers spend a long time writing documentation manually instead of focusing on writing code.

These problems lead to reduced project quality, increased costs, and difficulty onboarding new developers.

Therefore, we need an intelligent documentation system that automatically analyzes code, generates documentation, detects outdated documentation, and keeps everything updated with changes.

3. Problem Objectives:

The main objective of this project is to build an intelligent documentation system that helps developers generate and update documentation automatically instead of writing it manually.

1. Automatic Code Analysis
- Read code and understand its complete structure
- Identify all classes, functions, and variables present
- Support multiple programming languages such as Python, Java, and JavaScript

2. Documentation Generation
- Write clear explanations for each class and function in the project
- Create API documentation with complete details (Swagger/OpenAPI)
- Draw UML class diagrams showing code relationships

3. Detecting Outdated Documentation
- Compare current code with existing documentation
- Find parts that have changed but their documentation hasn't been updated
- Alert the developer to sections that need updating

4. Intelligent Updates
- Use artificial intelligence to suggest documentation updates
- Write easy-to-understand explanations for complex code
- Send notifications to developers about required updates

5. User-Friendly Dashboard
- Upload and manage projects easily
- Display quick statistics about the project
- Download generated reports and documentation
- Maintain a record of all previous versions

6. Security and Authentication
- Secure user login
- Support Single Sign-On (SSO)
- Protect uploaded code and documentation.

4. Report Organization:

The report will be organized in the following chapters:

- Chapter 1: introduction.
- Chapter 2: literature review.
- Chapter 3: project management.
- Chapter 4: System analysis.
- Chapter 5: System Design
- Chapter 6: practical implementation.
- Chapter : Conclusion.

Chapter 2 literature review

Literature Review

Similar System Comparison

Comparison of Similar Systems

We aim to benefit from these systems to extract requirements and expand them with additional features to create a more powerful and effective system.

DocuWriter.ai – AI Code Documentation Tool System: (System 01)

DocuWriter.ai - #1 AI Code documentation tools

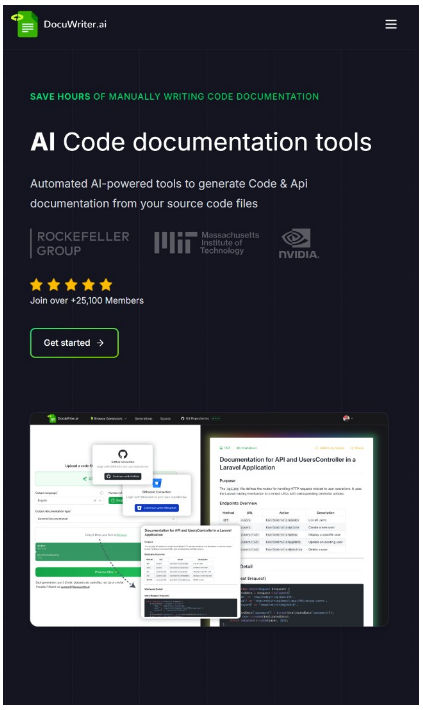
Key Feature Points (Feature Description)

Table 1: DocuWriter.ai

Feature	Description
Automated Code Documentation Generation	Uses AI to produce accurate,consistent documintation from source code files that refreshes continuously to avoid becoming outdated.
Swagger API Documentation	Generates Swagger-compliant JSON documentation from source code, compatible with tools like Postman for direct import.
AI-Powered Code Tests Suite Generation	Automates the creation of comprehensive test suites to ensure code reliability, catch bugs faster, and reduce manual test writing.
Intelligent Code Refactoring	AI-driven optimization to simplify code, improve readability, maintainability, and adherence to best practices (e.g., better variable names and comments).
Code Language Converter	Translates code between languages or frameworks quickly, such as Java to Dart or Bootstrap to TailwindCSS.
Code Comments & DocBlock Generator	Automatically adds comments and DocBlocks to codebases for better clarity.
UML Diagram Generator	Automates the creation of UML diagrams from code to visualize structure.
n8n Workflow Integration	Enables automated documentation generation triggered by events like Git pushes, with team notifications and scheduled updates.
Knowledge Base Management	Organizes documentation in customizable spaces with multi-language support, Markdown & PDF export, and unlimited history.

Additional Integrations	Supports Zapier, MCP for AI assistants (e.g., Cursor, Claude, ChatGPT), Git repository uploads, and chat with Git repos in higher plans.
Plans and Subscriptions	Offers tiered plans (Starter, Professional, Enterprise, Unlimited) with credits for generations; educational discounts available. Services focus on automation without free tiers mentioned for core features.

System Interfaces:

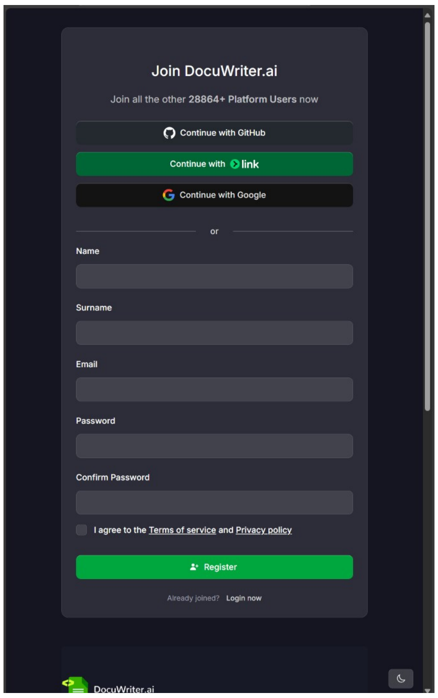

SAVE HOURS OF MANUALLY WRITING CODE DOCUMENTATION

AI Code documentation tools

Automated AI-powered tools to generate Code & Api documentation from your source code files

Join over +25,100 Members

DocuWriter.ai
ROCKEFELLER GROUP
Massachusetts Institute of Technology
NVIDIA.
Get started
Documentation for API and UsersController in a Laravel Application


Join DocuWriter.ai

Join all the other 28864+ Platform Users now

Name

Surname

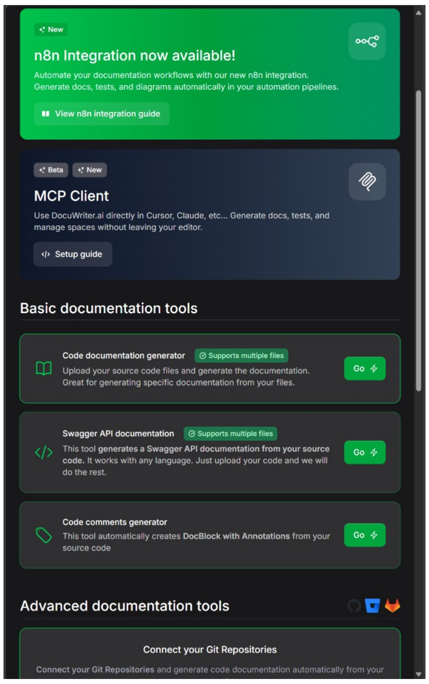
Email

Password

Confirm Password

I agree to the Terms of service and Privacy policy

Already joined? Login now


n8n Integration now available!

Automate your documentation workflows with our new n8n integration. Generate docs, tests, and diagrams automatically in your automation pipelines.

MCP Client

Use DocuWriter.ai directly in Cursor, Claude, etc... Generate docs, tests, and manage spaces without leaving your editor.

Basic documentation tools

Code documentation generator — Supports multiple files

Upload your source code files and generate the documentation. Great for generating specific documentation from your files.

Swagger API documentation — Supports multiple files

This tool generates a Swagger API documentation from your source code. It works with any language. Just upload your code and we will do the rest.

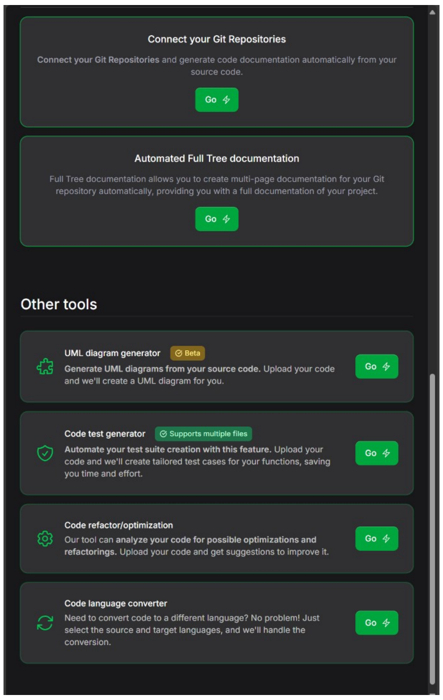
Code comments generator

This tool automatically creates DocBlock with Annotations from your source code

Advanced documentation tools

Connect your Git Repositories

Connect your Git Repositories and generate code documentation automatically from your


Connect your Git Repositories

Connect your Git Repositories and generate code documentation automatically from your source code.

Automated Full Tree documentation

Full Tree documentation allows you to create multi-page documentation for your Git repository automatically, providing you with a full documentation of your project.

Other tools

UML diagram generator — Beta

Generate UML diagrams from your source code. Upload your code and we'll create a UML diagram for you.

Code test generator — Supports multiple files

Automate your test suite creation with this feature. Upload your code and we'll create tailored test cases for your functions, saving you time and effort.

Code refactor/optimization

Our tool can analyze your code for possible optimizations and refactorings. Upload your code and get suggestions to improve it.

Code language converter

Need to convert code to a different language? No problem! Just select the source and target languages, and we'll handle the conversion.

Qodo.ai (formerly Codium) – Agentic Code Integrity Platform System: (System 02)

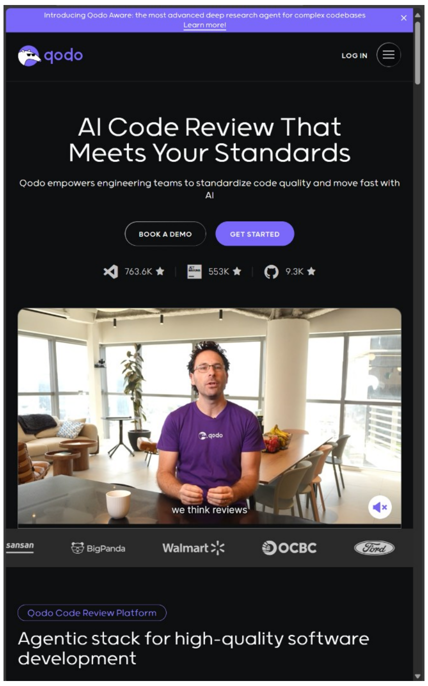
https://www.qodo.ai/

Table 2: Qodo.ai

Feature	Description
Qodo Gen – IDE Plugin	Intelligent agents for code generation, test workflows, and AI chat directly within VS Code and JetBrains IDEs. Supports all major programming languages with context-aware code generation.
Advanced Test Generation	Generates comprehensive test suites including happy paths, edge cases, and rare scenarios. Analyzes code behavior and dependencies to create meaningful tests aligned with project style and frameworks.
Run and Auto-Fix Tests	Executes tests directly within the IDE and automatically fixes failing tests. Supports multiple frameworks (Jest, Mocha, Vitest, Pytest, etc.) with customizable test configurations.
Qodo Aware (RAG Technology)	Retrieval-Augmented Generation provides deep codebase awareness, capturing naming conventions, architecture, and past implementations for context-aware suggestions with filtering for high-quality, relevant context.
Agentic Coding	AI agents that make decisions, ask questions, use tools, and carry out complex tasks autonomously. Supports custom tools and data sources for enhanced functionality.
Intelligent Code Refactoring	Safely refactors code across large files and modules while following team coding standards and best practices. Analyzes dependencies to maintain code integrity.
Qodo Merge – Git Agent	Automates PR reviews, generates comprehensive descriptions, and provides thorough walkthroughs. Includes PR Chat for AI-powered Q&A within GitHub environment.
Multi-Model Support	Switch between multiple AI models depending on task complexity. Maintains threaded conversations with context preservation.
Continuous Integration	Integrates with GitHub, GitLab, and Bitbucket. Automatically generates tests as part of code review workflows based on code diffs and commits.
Documentation Generation	AI-powered documentation for code with thorough explanations of classes, functions, and APIs. Generates inline comments and comprehensive technical documentation.
Bug Detection and Analysis	Finds and fixes bugs using agents that understand logic, behavior, and flow. Identifies vulnerabilities and provides security assessments.
Customization and Best Practices	Highly customizable with ability to index best practices, create custom agents, and tailor testing frameworks, mocks, and preferences to team standards.
Security and Compliance	SOC2 certified with SSL encryption. Analyzes only necessary code for context. Custom compliance checks available for enterprise users.

Plans and Pricing	Free tier for individual developers with core features. Teams tier with collaboration tools. Enterprise tier with multi-repo awareness, self-hosting, SSO, and priority support.

System Interfaces:

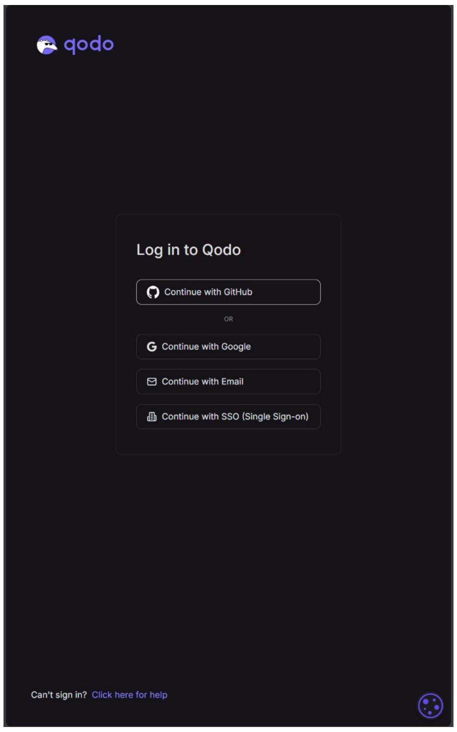


Introducing Qodo Aware: the most advanced deep research agent for complex codebases Learn more!

AI Code Review That Meets Your Standards

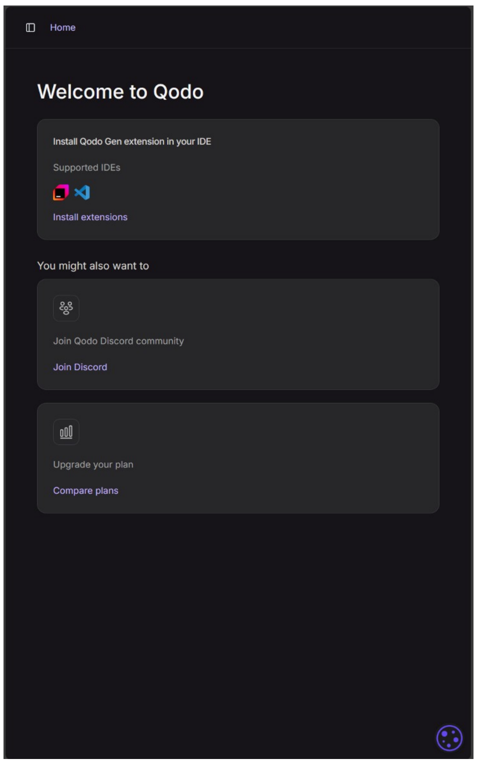
Qodo empowers engineering teams to standardize code quality and move fast with AI

Agentic stack for high-quality software development



Log in to Qodo

Can't sign in? Click here for help



Welcome to Qodo

Install Qodo Gen extension in your IDE

Supported IDEs

You might also want to

Join Qodo Discord community

Upgrade your plan

AutoTest & DocGen Manager – Intelligent System for Automated Testing and Documentation : (System 03)

Table 3: AutoTest & DocGen Manager

Feature	**Description**
Codebase Analysis	A web-based platform for software engineers to upload codebases, extract structure, dependencies, and core components automatically.
Automated Test Generation	Generates unit, integration, and edge-case tests using AI and code parsing techniques.
Test Execution and Reporting	Executes generated tests with detailed reports on results and overall code coverage percentage.
Automatic Documentation Generation	Produces documentation explaining classes, functions, and APIs in clear, understandable technical language.
UML Diagram Generation	Creates UML class and sequence diagrams directly from the analyzed code structure.
Outdated Documentation Detection	Detects outdated documentation and suggests updates when code changes occur.
AI-Driven Code Summarization	Generates natural-language explanations for complex functions or modules.
Interactive Dashboard	Allows developers to upload projects, view generated tests, run analyses, and download reports interactively.
Continuous Integration Support	Integrates with systems like GitHub or GitLab for automatic testing and documentation after each commit.

Non-Functional Requirements:

Table 4: Non-Functional Requirements

Category	**Requirements**
Performance	• Response time not exceeding two seconds for test generation and execution • Ability to process at least 100 simultaneous codebase uploads • Documentation generation time not exceeding 5 seconds per module
Usability	• Simple interface suitable for developers of various experience levels • Completion of analysis and generation in less than 5 steps • Support for Arabic and English languages at minimum • Providing guidance and tips for new users
Security	• Encryption of uploaded codebases • Two-factor authentication for user accounts • Logging of all analysis and generation operations • Multi-level permissions for team access
Scalability	• Support for a 50% annual increase in users without impacting performance • Ability to easily add new features or integrations • Customization of dashboards and reports for different projects

Conclusion: (Main Output)

Table 5: Conclusion

Feature/Type	DocuWriter.ai	Qodo.ai	AutoTest & DocGen Manager	Recommendations for Improvement
Web-based System	✓ Web-based	× (IDE-focused)	✓ Web-based	Maintain web-based accessibility for broader reach
Specialized in Software Products	✗ General code focus	✗ General code focus	✓ Product-focused	Competitive advantage - maintain product specialization
AI-Powered Codebase Analysis	✓ Comprehensive	✓ RAG Technology	✓ Advanced extraction	Implement RAG technology for deep context awareness
Automated Test Generation	✓ Advanced	✓ Behavior-driven (happy paths, edge, rare scenarios)	✓ Comprehensive (unit, integration, edge cases)	Adopt behavior-based generation with comprehensive coverage
Test Execution and Reporting/Coverage	✗ Not present	✓ Auto-fix capabilities	✓ Detailed (with coverage reports)	Maintain execution with detailed coverage reporting
Technical and API Documentation Generation	✓ Comprehensive (continuous refresh, Swagger API)	✓ AI-powered (inline comments)	✓ Basic (Classes, Functions, APIs)	Combine continuous refresh with AI-powered context awareness
UML/Workflow Diagram Generation	✓ Present	✗ Limited	✓ Class & Sequence diagrams	Provide comprehensive diagram types including class, sequence, and activity
Outdated Documentation Detection	✓ Continuous refresh	✗ Via RAG only	✓ Present (detection + suggestions)	Competitive advantage - proactive detection with automatic update suggestions
Natural Language Code Summarization	✗ Not mentioned	✓ Advanced	✓ Present	Enhance with multi-language support (Arabic/English)
Multi-Language	✗ Limited multi-	✗ EN primary	✓ Arabic and	**Competitive**

Support (Arabic/English)	language		English	**advantage** - maintain and expand language support
Code Refactoring/Optimization	✓ Advanced	✓ Team standards adherence	✗ Not present	Consider adding refactoring that adheres to team-specific coding standards
Language/Framework Converter	✓ Present (Java↔Dart, Bootstrap↔Tailwind)	✗ Not present	✗ Not present	Evaluate feasibility as future enhancement
RAG Technology for Context Awareness	✗ Not present	✓ Qodo Aware	✗ Not present	Consider implementing for enhanced context understanding
Pull Request Automation	✗ Not present	✓ Qodo Merge (PR Chat)	✗ Not present	Evaluate for future CI/CD enhancement
IDE Integration	✗ Web-based	✓ Native plugins (VS Code, JetBrains)	✗ Web-based	Consider plugin development while maintaining web platform
Notifications Automation	✓ Team notifications (n8n/Zapier)	✓ PR Chat	✓ Test results, failures, coverage alerts	Maintain comprehensive notification system
CI/CD Integration	✓ n8n, Zapier, Git pushes	✓ GitHub, GitLab, Bitbucket	✓ GitHub, GitLab, Jenkins	Expand integration options to cover major platforms
Agentic AI Capabilities	✗ Limited	✓ Full agents	✗ Not present	Evaluate autonomous AI agents for future development
System Security	✓ High (code deletion post-processing)	✓ SOC2 certified	✓ Required (encryption, 2FA, logging)	Ensure SOC2 certification and custom compliance checks for enterprise
Programming Language Support	✓ Multiple	✓ All major	✓ Multiple	Support all major programming languages with framework-specific features

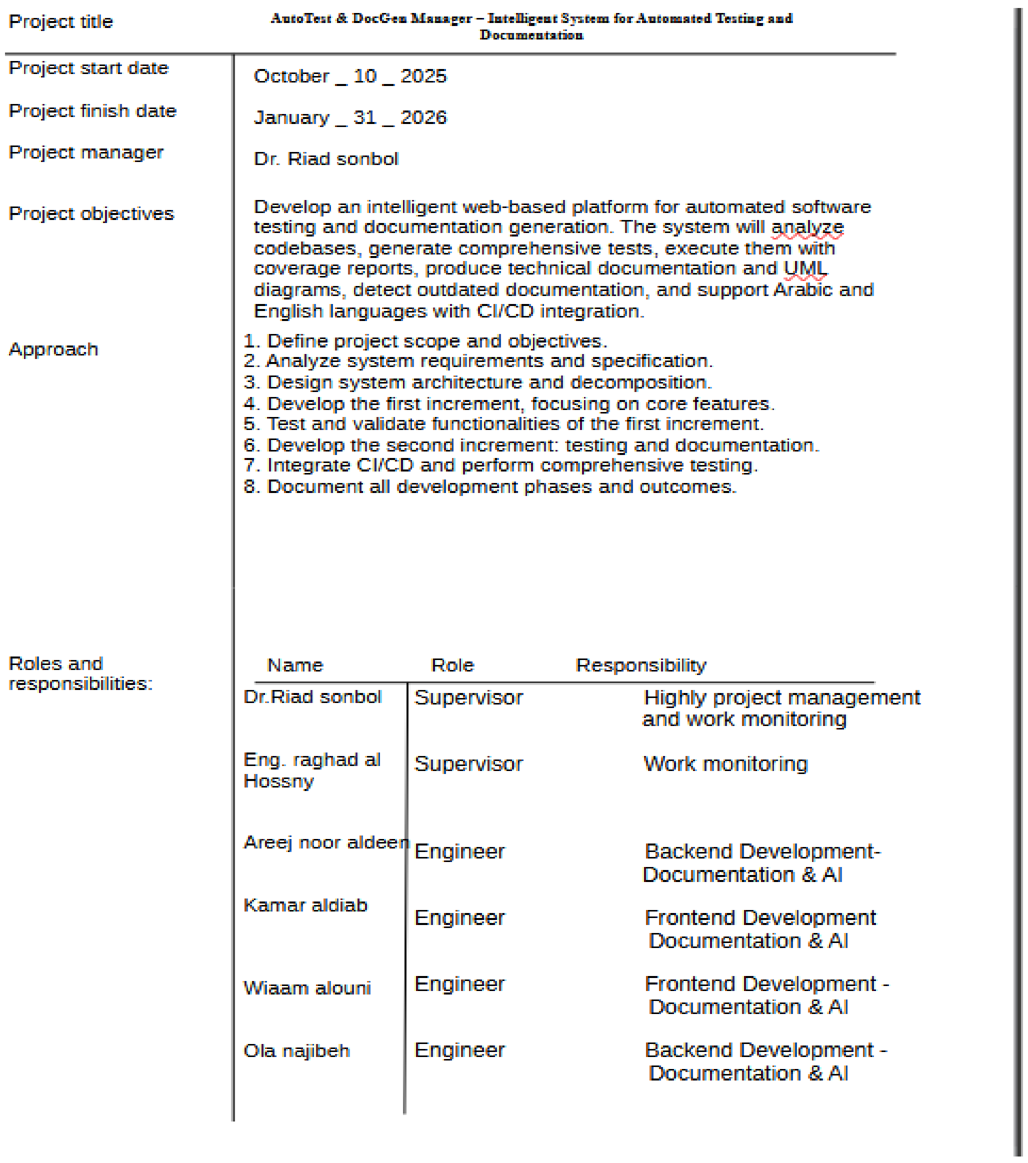
Chapter 3 Project Management

1. Introduction:

In this chapter, we will dive into the management phase of the AutoTest & DocGen Manager project, which is a critical aspect of ensuring the project's success. We will examine the project charter, project plan, Statement of Work (SOW) document, stakeholder analysis,.This intelligent system aims to revolutionize software development workflows by automating testing and documentation processes, ultimately improving software quality and maintainability.

2. Project Charter:

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision-making and project governance.



Project title	AutoTest & DocGen Manager – Intelligent System for Automated Testing and Documentation
Project start date	October _ 10 _ 2025
Project finish date	January _ 31 _ 2026
Project manager	Dr. Riad sonbol
Project objectives	Develop an intelligent web-based platform for automated software testing and documentation generation. The system will analyze codebases, generate comprehensive tests, execute them with coverage reports, produce technical documentation and UML diagrams, detect outdated documentation, and support Arabic and English languages with CI/CD integration.
Approach	1. Define project scope and objectives. 2. Analyze system requirements and specification. 3. Design system architecture and decomposition. 4. Develop the first increment, focusing on core features. 5. Test and validate functionalities of the first increment. 6. Develop the second increment: testing and documentation. 7. Integrate CI/CD and perform comprehensive testing. 8. Document all development phases and outcomes.
Roles and responsibilities:	

Name	Role	Responsibility
Dr.Riad sonbol	Supervisor	Highly project management and work monitoring
Eng. raghad al Hossny	Supervisor	Work monitoring
Areej noor aldeen	Engineer	Backend Development- Documentation & AI
Kamar aldiab	Engineer	Frontend Development Documentation & AI
Wiaam alouni	Engineer	Frontend Development - Documentation & AI
Ola najibeh	Engineer	Backend Development - Documentation & AI

3. The SOW document

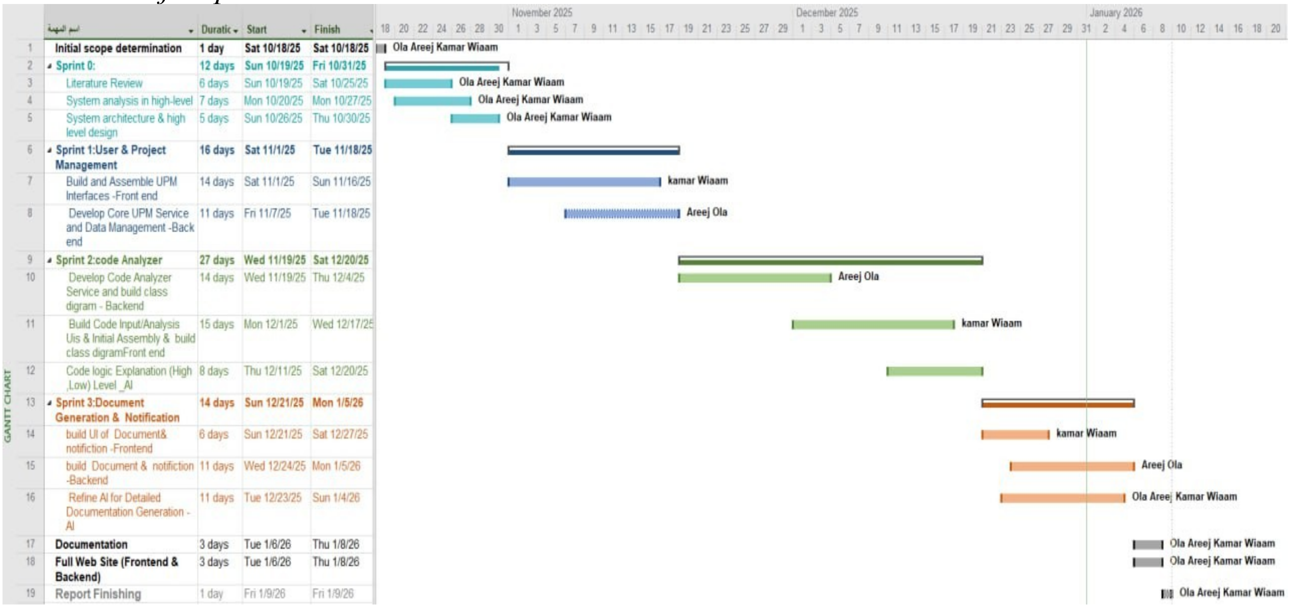
A Statement of Work (SOW) is a comprehensive document that defines the scope of work for a project. It outlines specific tasks, deliverables, timelines, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project objectives, and the success criteria.

Project Title:	**AutoTest & DocGen Manager – Intelligent System for Automated Testing and Documentation**
Project Description and Objectives:	The project aims to develop an intelligent web platform based on artificial intelligence that automatically analyzes codebases, and produces updated technical documentation. The system utilizes static analysis and AI-powered reasoning along with code-to-text conversion to enhance software quality and ensure documentation consistency with the codebase.
Project goals:	• Automatically analyze uploaded codebases and extract structure, dependencies, and core components. • Produce automatic documentation that explains classes, functions, and APIs. • Create UML class diagrams from the analyzed code. • Detect outdated documentation and suggest updates. • Provide AI-supported summaries for complex code units. • Offer an interactive dashboard for project uploads and report downloads.
Project Deliverables:	• Project plan and SRS documents. • Functional diagrams for system analysis. • Back-end and front-end system components with full functionality. • Automatic technical documentation with API documents and UML diagrams. • Final project report.
Technology and Tools:	Programming Languages: Python, JavaScript – html - CSS. Frameworks: Django for backend, React for frontend
Human Resources:	Dr. Raid Sonbol - Project Manager Eng.Raghad al Hossny – Work monitoring Kamar aldiab - SE Developer Ola najibeh - SE Developer Areej nooraldeen - SE Developer Wiaam alouni - SE Developer
Schedule:	Project Start Date: October 10, 2025 First second project Seminar: November 15, 2025 First second project Seminar: November 27, 2025 Project Finish Date: January 31, 2026.

4. Project plan – Gantt chart:

A document outlining tasks, deadlines, and resources needed to achieve project objectives, serving as a roadmap for successful project execution.

Table 6: Project plan-Gant chart



	اسم المهمة	Duratic	Start	Finish
1	Initial scope determination	1 day	Sat 10/18/25	Sat 10/18/25
2	Sprint 0:	12 days	Sun 10/19/25	Fri 10/31/25
3	Literature Review	6 days	Sun 10/19/25	Sat 10/25/25
4	System analysis in high-level	7 days	Mon 10/20/25	Mon 10/27/25
5	System architecture & high level design	5 days	Sun 10/26/25	Thu 10/30/25
6	Sprint 1:User & Project Management	16 days	Sat 11/1/25	Tue 11/18/25
7	Build and Assemble UPM Interfaces -Front end	14 days	Sat 11/1/25	Sun 11/16/25
8	Develop Core UPM Service and Data Management -Back end	11 days	Fri 11/7/25	Tue 11/18/25
9	Sprint 2:code Analyzer	27 days	Wed 11/19/25	Sat 12/20/25
10	Develop Code Analyzer Service and build class digram - Backend	14 days	Wed 11/19/25	Thu 12/4/25
11	Build Code Input/Analysis Uis & Initial Assembly & build class digramFront end	15 days	Mon 12/1/25	Wed 12/17/25
12	Code logic Explanation (High ,Low) Level _AI	8 days	Thu 12/11/25	Sat 12/20/25
13	Sprint 3:Document Generation & Notification	14 days	Sun 12/21/25	Mon 1/5/26
14	build UI of Document& notifiction -Frontend	6 days	Sun 12/21/25	Sat 12/27/25
15	build Document & notifiction -Backend	11 days	Wed 12/24/25	Mon 1/5/26
16	Refine AI for Detailed Documentation Generation -AI	11 days	Tue 12/23/25	Sun 1/4/26
17	Documentation	3 days	Tue 1/6/26	Thu 1/8/26
18	Full Web Site (Frontend & Backend)	3 days	Tue 1/6/26	Thu 1/8/26
19	Report Finishing	1 day	Fri 1/9/26	Fri 1/9/26

5. Risk Management:

Table 7: Risk management

Risk_ID	Risk Category	Risk Title	Risk Description	Impact	Mitigation Plan
RK-01	Technical (AI)	Context Window Constraints	The risk of uploaded code exceeding the maximum token limit that the AI model can process at once.	High	Implementing "Intelligent Code Chunking" to process the code in logical, manageable segments.
RK-02	Management	Task Dependency	Potential delays in Backend development hindering the progress of related Frontend features.	Very High	Utilizing API Mocking to allow Frontend development to proceed independently of backend logic completion.

RK-03	Technical	Logic Accuracy & Hallucination	The possibility of the AI providing inaccurate explanations for complex, non-standard, or highly nested code structures.	High	Conducting manual validation of AI outputs and continuously refining System Prompts for better precision.
RK-04	Management	Scope Creep	Adding new requirements or features beyond the agreed SOW, which threatens the project delivery timeline.	Medium	Adhering strictly to the "Project Charter" and deferring secondary features until core functionalities are finalized.
RK-05	Security	Data Privacy & Security	The risk of sensitive source code being exposed or leaked during the upload and analysis process.	Very High	Implementing data encryption during transit and ensuring files are deleted from the server immediately after documentation is generated.
RK-06	Technical	Export & Rendering Quality	Potential formatting errors when converting complex UML diagrams or documentation into PDF or Markdown formats.	Medium	Using robust export libraries (e.g., React-PDF) and conducting cross-browser compatibility testing for all output formats.

6. Summary:

In conclusion, good project management is vital for successful software system development. It provides structure, ensuring projects are completed within scope and schedule. Overall, strong project management enhances the delivery of high quality software systems that meet the goals of the project.

Chapter 4 System Analysis

1. Introduction:

In this chapter we will introduce the analytical study of the system using the needed UML diagrams for system requirements modeling.

2. Requirements Elecitiation:

Table 8: Requirements Elecitiation

Req_ID	Requirment Title	Actor	Priority	UseCase
RE-FR-01	The system shall provide user login.	Developer	High	Uc-01
RE-FR-02	The system **shall** support single-sign-on (SSO).	Developer	High	Uc-01
RE-FR-03	The system **shall** provide account creation.	Visitor	High	Uc-02
RE-FR-04	The system shall compare the current codebase against existing documentation to detect and flag outdated content.	Developer	Medium	Uc-03
RE-FR-05	The system shall alow the user to upload the code	Developer	High	Uc-04
RE-FR-06	The system shall perform comprehensive Code Analysis, extracting structure, dependencies, and elements.	Developer	High	Uc-04
RE-FR-07	The system **should** track execution flow and analyze conditions/loops.	Developer	High	Uc-04
RE-FR-08	The system **should** support multiple programming languages (e.g., Python, Java, JavaScript).	Developer	High	Uc-04
RE-FR-09	The system **should** support multiple output formats (Markdown, PDF) and allow exporting diagrams as images or editable files.	Developer	High	Uc-05
RE-FR-10	The system **shall** generate UML (class) diagrams .	Developer	High	Uc-05
RE-FR-11	The system **shall** generate comprehensive API documentation, automatically generating Swagger/OpenAPI specifications, including endpoints and parameters.	Developer	High	Uc-05
RE-FR-12	The system **should** explain classes and functions using clear, technical language.	Developer	High	Uc-06
RE-FR-13	The system **should** explain complex algorithms and data structures.	Developer	High	Uc-06
RE-FR-14	The system **should** utilize AI-Driven Code Summarization to generate natural language explanations for functions and	Developer	High	Uc-06

	modules based on code context.			
RE-FR-15	The system **shall** allow running analyses and downloading reports.	Developer	High	Uc-07
RE-FR-16	The system **should** offer a customizable user interface and report templates.	Developer	High	Uc-07
RE-FR-17	The system **should** compare results with previous versions.	Developer	High	Uc-07
RE-FR-18	The system **shall** display a Main Dashboard with project overview and quick statistics.	Developer	High	Uc-07
RE-FR-19	The user **shall** be able to manage project settings and configurations.	Developer	High	Uc-07
RE-FR-20	The user **shall** be able to create, delete, and archive projects.	Developer	High	Uc-07
RE-FR-21	The system **shall** allow the user to upload a new digital document file	Developer	High	Uc-08

3. SRS Document:

1. Introduction:

i. Purpose:

This Software Requirements Specification (SRS) defines the key functional and non-functional requirements of the system. It serves as a clear reference to align stakeholders and developers throughout the project.

ii. Project Scope:

An intelligent assistant for developers that automatically analyzes any codebase including API docs, class diagrams, and detailed explanations of functions and components. The system relies on static analysis and AI-driven reasoning (LLMs) with code-to-text generation to enhance software quality and maintainability.

The System seeks to provide a smooth experience. Objectives includes:

- Analyze uploaded codebases to extract structure, dependencies, and key components.
- Produce comprehensive automated documentation, including function descriptions, classes, APIs, and UML diagrams.
- Discover and update outdated documentation with natural, AI-powered summaries.
- Provide an interactive dashboard for project management and results review.

iii. Document overview:

The software requirements specification SRS document will be structured:
1. Introduction.
2. Overall description and main features.
3. Non_Functional Requirements.

2. Overall Description and main features:

i. Main features:

The following functionalities are extracted from the system's use cases:

- **Sign In (UC-01) :** Allows registered developers to access the system by providing their login credentials.
- **Sign Up (UC-02) :** Enables new users (Developers) to create and register a new account within the system.
- **Document / Code Compatibility Check (UC-03) :** Verifies the consistency and compatibility between existing documentation and the uploaded code, or checks the internal harmony of the code itself.
- **Code Analysis & Structure Extraction (UC-04) :** Analyzes the uploaded code to extract its structural blueprint and underlying programming logic.
- **Generate Document (UC-05) :** Automatically creates and generates technical documentation for the project based on the analyzed source code.
- **Code Logic Explanation (UC-06) :** Offers a simplified and logical explanation of how specific parts of the code function or operate.
- **Project Management (UC-07) :** Provides the developer with tools to manage and organize their software projects within the system.
- **Upload Document (UC-08) :** This use case allows the user to submit external document file to the system. It secures the file, validates its format, and links it to the appropriate project (UC-07) for subsequent processing and analysis.

ii. Main actors and their related functionalities:

1. Developer

The Developer is the primary user who interacts with the system. They are responsible for initiating all use cases related to managing, analyzing, and documenting their code and projects.

2. Visitor

The Visitor is an unregistered user in the system. Thier primary role is to create a new account or sign in to access the system's functionalities

3. Admin

The Admin is the primary user with permissions in the system. They are responsible for managing projects. They can fully manage projects.

3. Non-Functional requirements:

1. Performance:

- Analysis Response Time: Code analysis and documentation must be completed in less than 10 seconds for projects under 10MB.
- Concurrent Processing: The system must be capable of efficiently handling at least 50 simultaneous analysis.
- Dashboard Loading: The Main Dashboard loading time should not exceed 3 seconds to display instantaneous statistics.

2. Usability:

- Interaction Ease: The interface must support drag-and-drop for project uploads and feature a simple, developer-friendly design.
- Language Support: The system must support multiple natural languages (e.g., English and Arabic) for code explanations and interface text.
- Comprehensive Dashboard: A single main dashboard must provide a quick project overview, key statistics (e.g., code coverage, execution time), and actionable insights.
- The interface should be responsive and compatible with both desktop and tablet screens.

3. Security:

- Authentication: The system must support Single Sign-On (SSO) in addition to standard user login and account creation.
- Integration Security: Secure authorization mechanisms (e.g., OAuth) must be used and all access tokens must be protected.
- All user passwords must be encrypted using industry-standard hashing algorithms (e.g., BCrypt).

4. Maintainability:

- Project History: The system must maintain a complete record of project versions and history, including all completed analysis, documentation, and test results.
- Self-Documentation: The system should automatically contribute to the documentation of its own code and structure through its outputs (API docs, UML diagrams).

5. Availability :

- The system should maintain an uptime of 99.5% to ensure developers can access their documentation anytime.

4. Requirements Modelling:

Use case diagram: use-case diagrams model the behaviour of a system and help to capture the requirements of the system.

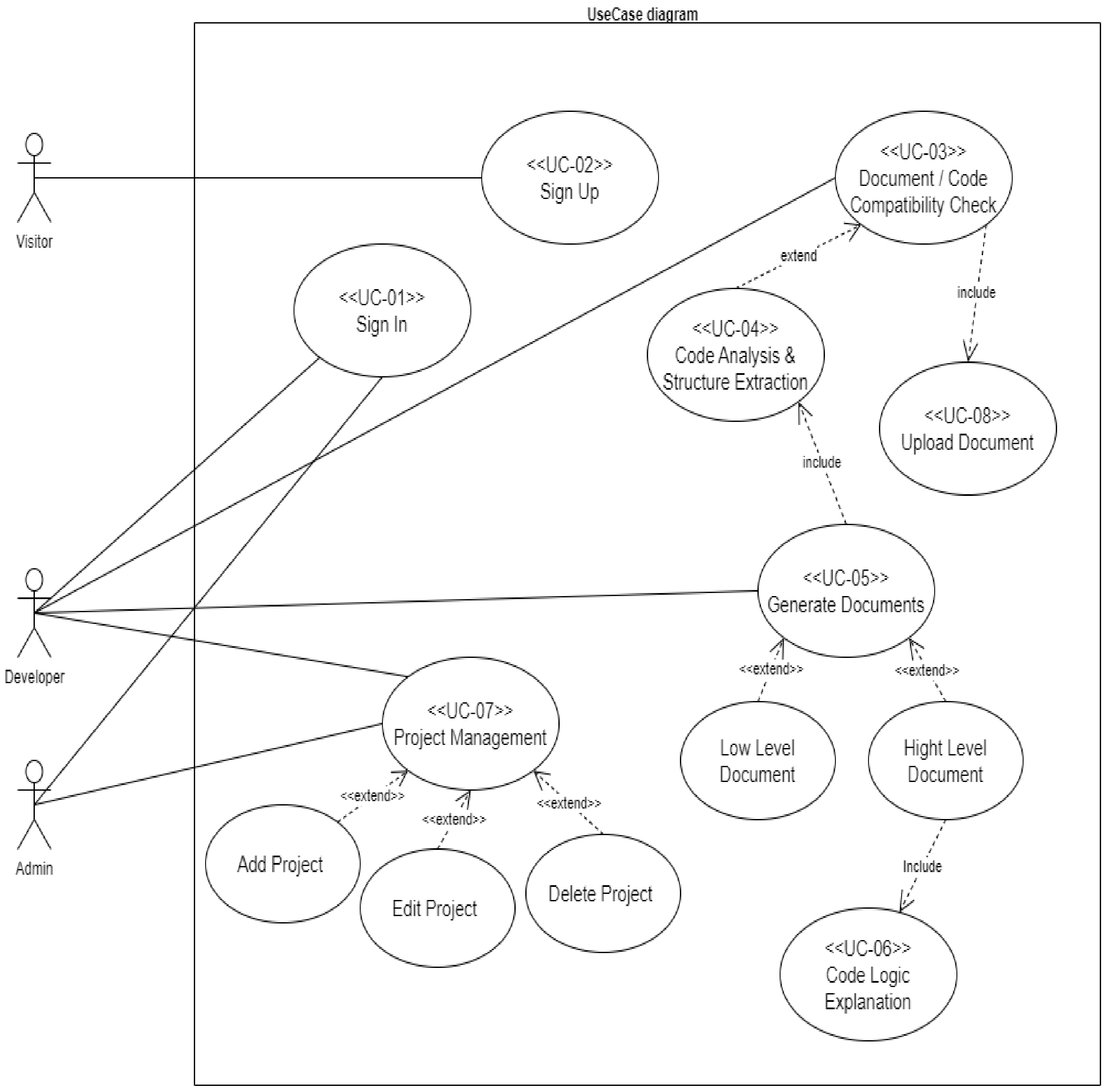


Figure 1: Usecase Diagram

System features – use cases:

- Sign In (UC-01) :

Table 9: sign-In use case specification

Use case name:	Sign In
Participating Actors:	Developer
Main Flow:	1. The system displays the login screen, requesting the username/email and password. 2. The Developer enters the credentials and clicks "Sign In". 3. The system sends the credentials to the Authentication Service. 4. The Authentication Service verifies the data (matching the name and the hashed password in the database). 5. If the data is correct, the system creates an access token/session for the Developer. 6. The system directs the Developer to the main dashboard.
Alternative Flows	First alternative flow A1: First Time Login • After step 5 (Session Creation), the system checks if this is the user's first login. • If so, the system directs the user to a profile setup page or a guided tour instead of the dashboard directly.
Exception Flows	Exception 1: Incorrect Credentials • Failure to match the username/email or the hashed password. • The process is stopped without indicating which part of the data is wrong (for security), and a failed attempt is logged. • "Incorrect username or password. Please try again." Exception 2: Failure to Connect to User Database • The system is unable to retrieve the user record for comparison. • Halt the flow, and log a system error internally. • "Sorry, an unexpected technical error occurred. Please try again later."
Preconditions	• The user must have previously signed up for the system (UC-02).
Postconditions	• The user is directed to the main dashboard, and a working session is created for them.

Sequence diagram:

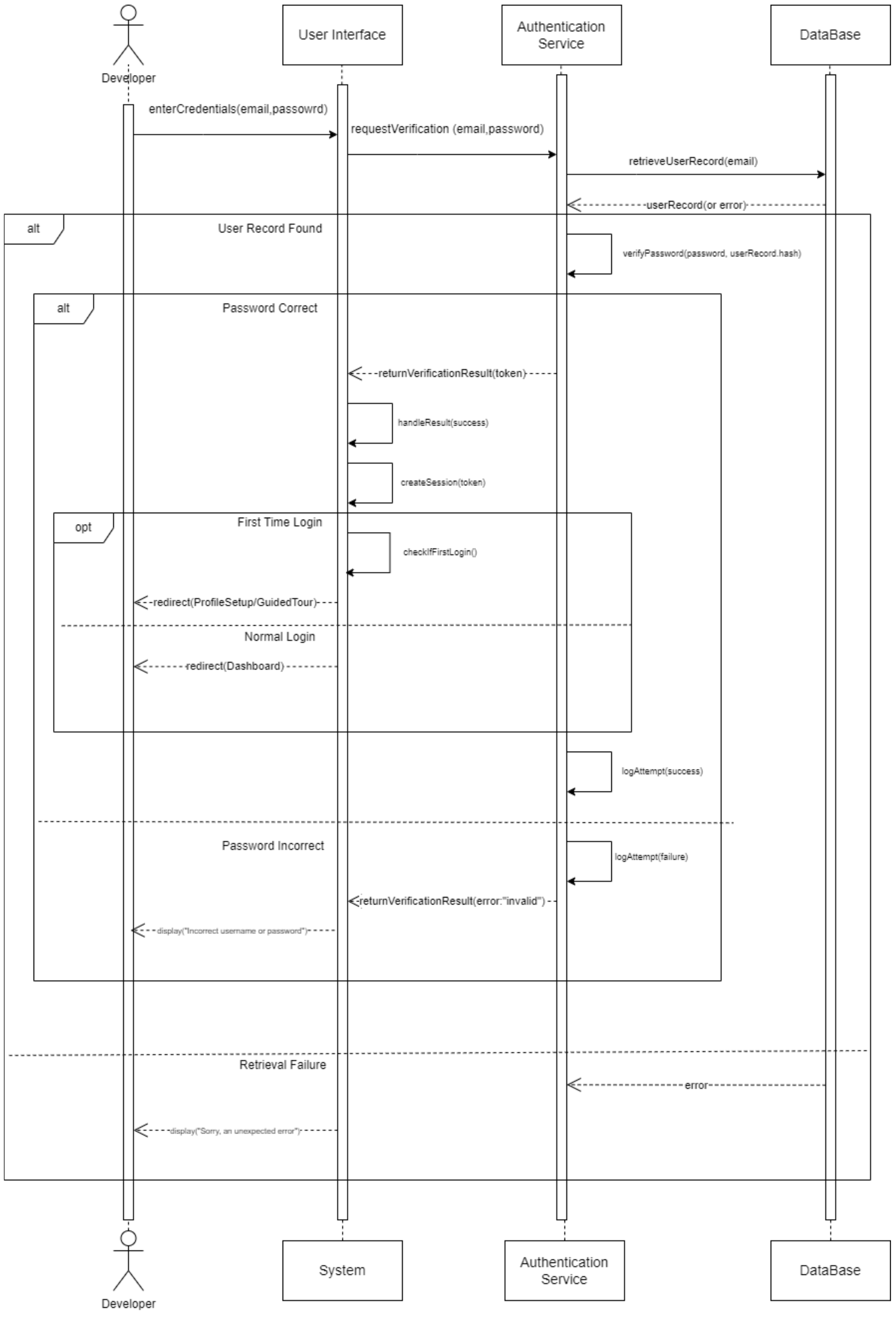


Figure 2: Sign-In sequence diagram

- Sign Up (UC-02) :

Table 10: sign-up use case specification

Use case name:	Sign IUp
Participating Actors:	Visitor
Main Flow:	1. The system displays the registration screen, requesting (Name, Email, Password). 2. The user enters the data and agrees to the terms and conditions. 3. The system verifies that the email address is not already in use. 4. The system hashes the password. 5. The system creates a new user record in the database. 6. The system sends a welcome or email verification message (Optional). 7. The system directs the user to the login page or the dashboard directly.
Alternative Flows	First alternative flow A1: Sign Up with Additional Verification Step • . After step 5 (Record Saving), the account is not immediately activated. • The system sets the account status to "Pending/Verification Required". • A verification link is sent via email (successfully). • Upon clicking the link, the account status is updated to "Active", and the Basic Flow is considered complete. Second alternative flow A2: Omission of Optional Data • In step 2 the user decides not to enter optional fields (e.g., phone number). • The system proceeds to step (3.0), leaving the optional fields empty in the database.
Exception Flows	Exception 1: Email Already Exists • **Details:** Database check shows the entered email is already in use. • **Action:** The record creation process is halted. • **Response:** "Sorry, this email address is already registered. Please log in or use the forgot password option." Exception 2: Data Validation Failure • **Details:** Failure to validate email format or mismatched password and confirmation. • **Action:** Submission is halted, and the field containing the error is highlighted.

	• **Response:** "Passwords must match" or "Invalid email format.
Preconditions	• The user must have previously signed up for the system (UC-02).
Postconditions	**On Success:** • The user is directed to the main dashboard, and a working session is created for them. **On Failure:** • Failure to send the welcome/verification email.

<u>Sequence diagram:</u>

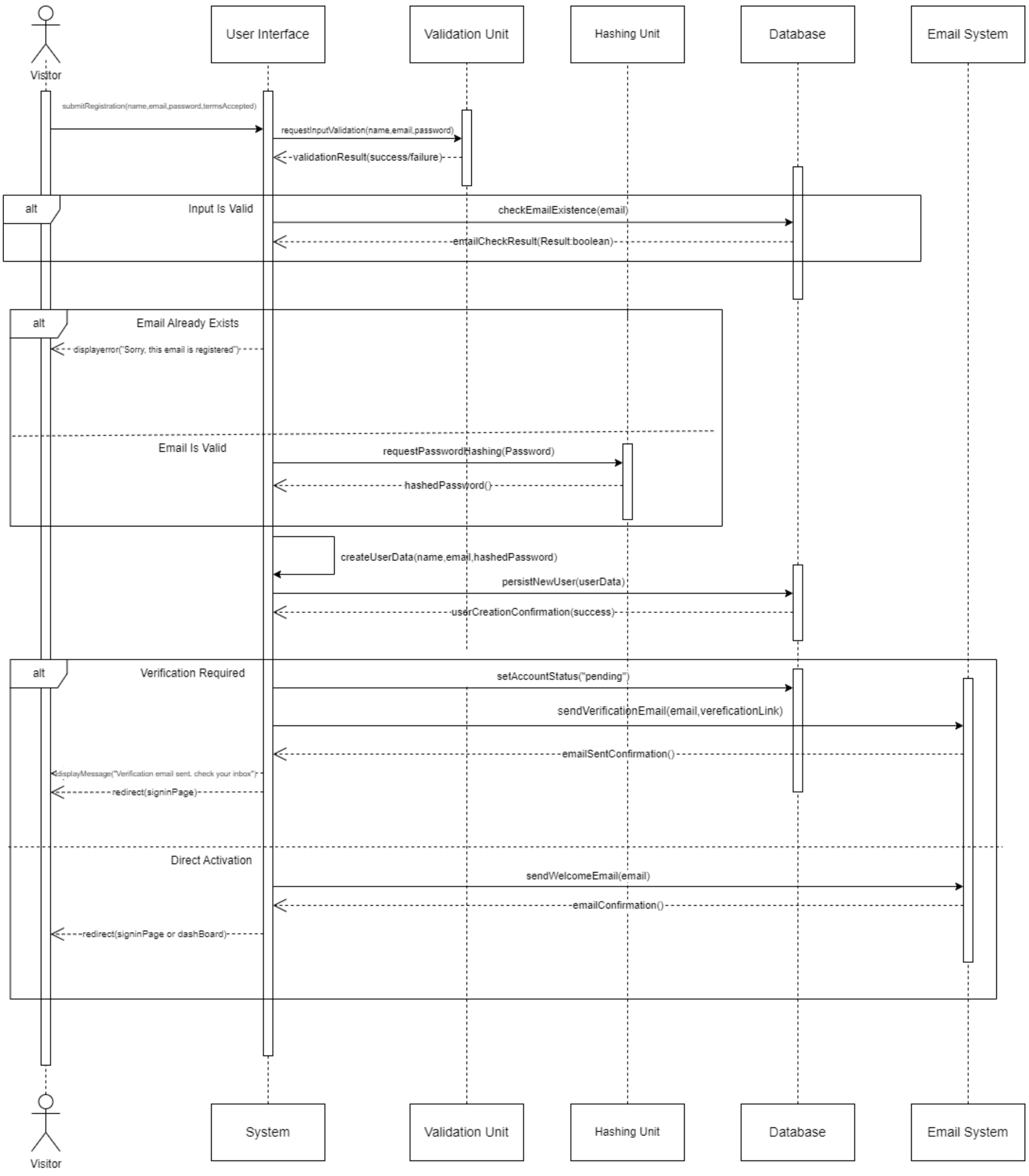

Visitor
User Interface
Validation Unit
Hashing Unit
Database
Email System
submitRegistration(name,email,password,termsAccepted)
requestInputValidation(name,email,password)
validationResult(success/failure)
alt
Input Is Valid
checkEmailExistence(email)
emailCheckResult(Result:boolean)
alt
Email Already Exists
displayerror("Sorry, this email is registered")
Email Is Valid
requestPasswordHashing(Password)
hashedPassword()
createUserData(name,email,hashedPassword)
persistNewUser(userData)
userCreationConfirmation(success)
alt
Verification Required
setAccountStatus("pending")
sendVerificationEmail(email,vereficationLink)
emailSentConfirmation()
displayMessage("Verification email sent. check your inbox")
redirect(signinPage)
Direct Activation
sendWelcomeEmail(email)
emailConfirmation()
redirect(signinPage or dashBoard)
Visitor
System
Validation Unit
Hashing Unit
Database
Email System

Figure 3: SignUp sequence diagram

- Document / Code Compatibility Check (UC-03):

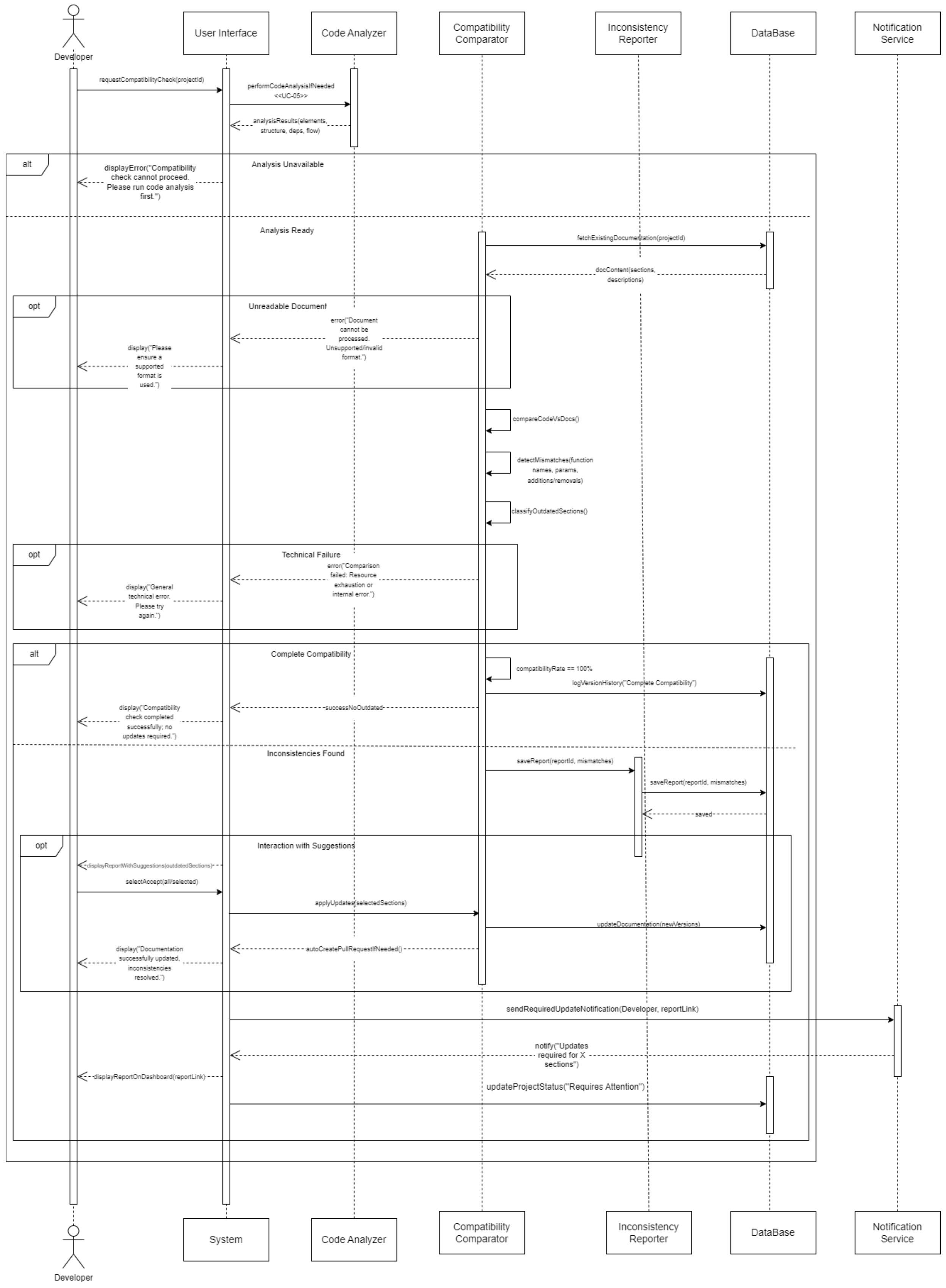
Table 11: Document/code compatibility check use case specification

Use case name:	Document / Code Compatibility Check
Participating Actors:	Developer
Main Flow:	1. The Developer requests a compatibility check (or it runs automatically). 2. The system extends **<<UC-04>>** to perform the necessary code analysis. 3. The system compares the code analysis results with the content of the existing documentation in the database. 4. The system checks for structural changes (e.g., function name change, addition/removal of parameters) and compares them with the textual explanations. 5. The system classifies non-matching documentation as "Outdated". 6. The system creates and sends a notification to the Developer about the required updates.
Alternative Flows	First alternative flow A1: Complete Compatibility State • The **System** finishes comparing all code and documentation elements • The **System** determines that the compatibility rate is 100% and no documentation sections are classified as "Outdated." • The **System** logs the "Complete Compatibility" result in the Version History. • A brief message is displayed on the dashboard stating, "Compatibility check completed successfully; no documentation updates are required." Second alternative flow A2:Interaction with Suggestions • The **System** generates the Inconsistency Report, as in the Main Flow. • The developer reviews the suggestions and selects either "Accept All" or "Accept Selected." • The **System** directly modifies the outdated documentation files within the system (or automatically creates a `Pull Request`), and saves the new, updated version. • A notification is displayed: "Documentation successfully updated, and inconsistencies resolved."
Exception Flows	Exception 1: Analysis Data Unavailable • The system detects that code analysis results (from UC-05) are missing or corrupt. • The system refuses to start the comparison process. • The system displays a message to the Developer: "Compatibility check cannot proceed. Please run code analysis first." Exception 2: Unreadable Document/Unsupported Format • The system attempts to read the stored document for comparison. • The system detects that the document format (e.g., a corrupted Word file or unanalyzable PDF) is unsupported or invalid. • The system halts the comparison. • The system logs a file processing error and displays a message to the Developer: "Document cannot be processed. Please ensure a supported format is used."

	Exception 3: Technical Comparison Failure/Resource Exhaustion • During the comparison step between code structures and text, an internal software error occurs, or the system fails to allocate sufficient resources for the complex comparison process. • The system immediately stops the check. • The system logs the failure and displays a general technical error message to the Developer.
Preconditions	• A project containing new/modified code and previous documentation has been uploaded. • <<UC-04>> Code Analysis & Structure Extraction has been completed successfully.
Postconditions	**On Success:** • The **Compatibility Status** field in the project record is updated to "Completed" or "Requires Attention". • The parts incompatible with the current code are identified and linked to the generated report. • Notifications are sent to the Developer regarding sections that require updating. **On Failure:** • The Compatibility Status field in the project record remains "Analysis Required" or "Failed Check". • An Error Log is created detailing the reason for the check failure (e.g., Missing Analysis Data). • A notification is sent to the Developer explaining the failure and suggesting the next step (rerunning UC-05).

Sequence diagram:

Figure 4: Document / Code Compatibility Check sequence diagram



Developer
User Interface
Code Analyzer
Compatibility Comparator
Inconsistency Reporter
DataBase
Notification Service
requestCompatibilityCheck(projectId)
performCodeAnalysisIfNeeded <<UC-05>>
analysisResults(elements, structure, deps, flow)
alt
Analysis Unavailable
displayError("Compatibility check cannot proceed. Please run code analysis first.")
Analysis Ready
fetchExistingDocumentation(projectId)
docContent(sections, descriptions)
opt
Unreadable Document
error("Document cannot be processed. Unsupported/invalid format.")
display("Please ensure a supported format is used.")
compareCodeVsDocs()
detectMismatches(function names, params, additions/removals)
classifyOutdatedSections()
opt
Technical Failure
error("Comparison failed: Resource exhaustion or internal error.")
display("General technical error. Please try again.")
alt
Complete Compatibility
compatibilityRate == 100%
logVersionHistory("Complete Compatibility")
successNoOutdated
display("Compatibility check completed successfully; no updates required.")
Inconsistencies Found
saveReport(reportId, mismatches)
saveReport(reportId, mismatches)
saved
opt
Interaction with Suggestions
displayReportWithSuggestions(outdatedSections)
selectAccept(all/selected)
applyUpdates(selectedSections)
updateDocumentation(newVersions)
autoCreatePullRequestIfNeeded()
display("Documentation successfully updated, inconsistencies resolved.")
sendRequiredUpdateNotification(Developer, reportLink)
notify("Updates required for X sections")
displayReportOnDashboard(reportLink)
updateProjectStatus("Requires Attention")
Developer
System
Code Analyzer
Compatibility Comparator
Inconsistency Reporter
DataBase
Notification Service

- Code Analysis & Structure Extraction (UC-04) :

Table 12: Code Analysis & Structure Extraction usecase specification

Use case name:	Code Analysis & Structure Extraction
Participating Actors:	Developer
Main Flow:	**1.** The system validates the uploaded files and checks the file formats. **2.** The system identifies the programming language(s) used in the project. **3.** The system performs static code analysis: • Analyzing source code files. • Extracting classes, functions, methods, and their parameters 4. The system analyzes the project structure: • Identifying modules and packages. • Mapping file organization and hierarchy 5. The system extracts dependencies: • Identifying external libraries and frameworks. • Mapping internal dependencies between modules. 6. The system tracks execution flow: • Analyzing conditional statements (if/else, switch). • Identifying loops (for, while, etc.). • Mapping function call chains 7. The system generates a comprehensive analysis report. 8. The system saves the analysis results to the database. 9. The system displays a summary of the analysis to the user on the dashboard. 10. The user reviews the extracted structure and the analysis results.
Alternative Flows	<u>First alternative flow A1: Detecting Multiple Programming Languages</u> • Step 2: If the system detects multiple programming languages: • The system analyzes each language separately • The system generates separate analysis reports for each language • The system merges the results into a unified project view • Continue to Step 3 <u>Second alternative flow A2: Incremental Analysis</u> • In Step 3: If the project has been analyzed previously: • The system detects only the changed files • The system performs analysis on the modified files • The system updates the existing analysis results • Continue to Step 7
Exception Flows	<u>Exception 1: Invalid File Format</u> • In Step 1: If the uploaded files are not valid code files: • The system displays an error message: "Invalid file format detected" • The system displays a list of supported file types • Use Case ends <u>Exception 2: Unsupported Programming Language</u> • In Step 3: If the programming language is not supported: • The system notifies the user: "The language is currently not supported" • The system provides a list of supported languages • The system allows the user to request language support • Use Case ends <u>Exception 3: Syntax Errors in the Code</u>

	• In Step 3: If the code contains critical syntax errors: • The system attempts to proceed with the analysis while handling the errors. • The system logs the syntax errors with line numbers. • The system generates a partial analysis report. • The system notifies the user of syntax issues. • Continue to Step 7 with partial results.
Preconditions	• The user must be logged into the system. • A valid project/code must be uploaded to the system. • The uploaded code must be in a supported programming language (Python, Java, JavaScript, etc.). • Project files must be accessible and readable.
Postconditions	**On Success:** • The complete structure of the project has been extracted and saved in the system • All code elements (classes, functions, variables) have been identified and indexed • A map of dependencies and relationships between components has been drawn • The execution flow have been analyzed • The analysis results are available for other use cases (documentation generation) **On Failure:** • An error log with failure details has been generated • The user has been notified of the analysis failure • Partial results (if any) have been saved for debugging purposes

Sequence diagram:

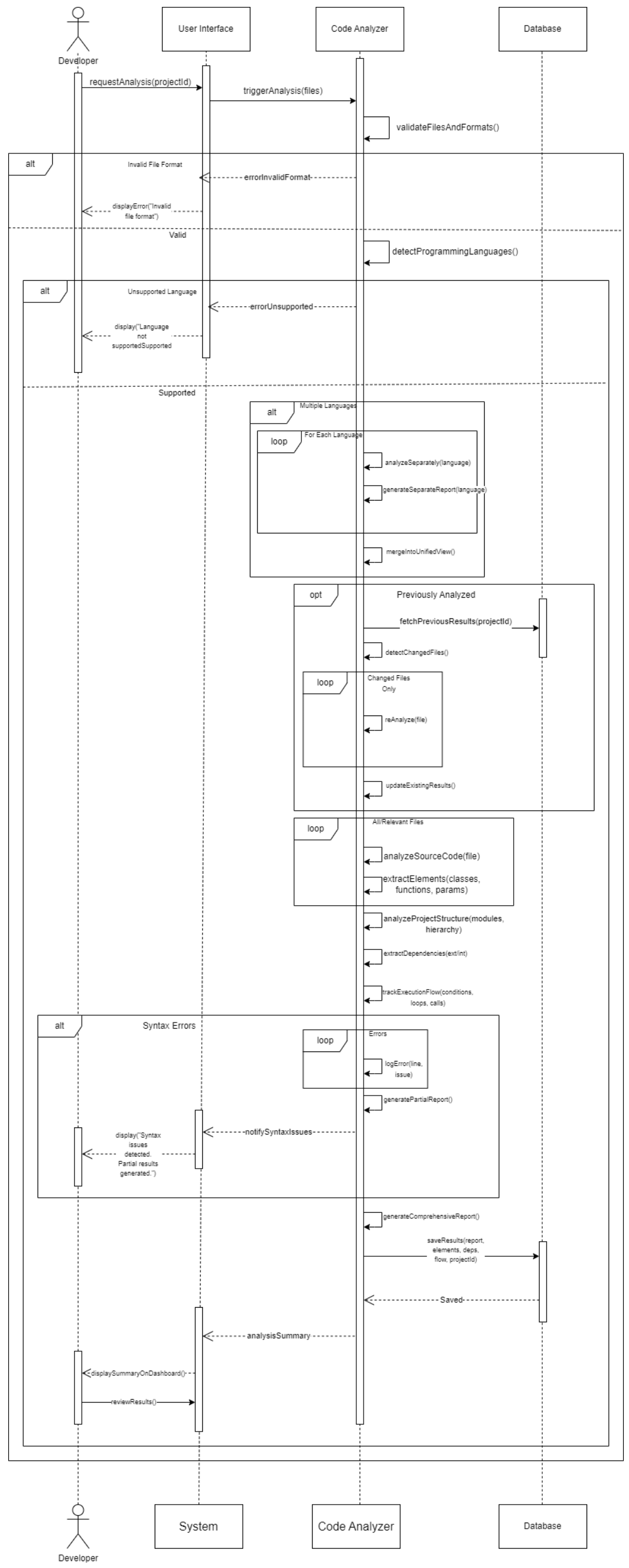

Developer
User Interface
Code Analyzer
Database
requestAnalysis(projectId)
triggerAnalysis(files)
validateFilesAndFormats()
alt
Invalid File Format
errorInvalidFormat
displayError("Invalid file format")
Valid
detectProgrammingLanguages()
alt
Unsupported Language
errorUnsupported
display("Language not supportedSupported
Supported
alt
Multiple Languages
loop
For Each Language
analyzeSeparately(language)
generateSeparateReport(language)
mergeIntoUnifiedView()
opt
Previously Analyzed
fetchPreviousResults(projectId)
detectChangedFiles()
loop
Changed Files Only
reAnalyze(file)
updateExistingResults()
loop
All/Relevant Files
analyzeSourceCode(file)
extractElements(classes, functions, params)
analyzeProjectStructure(modules, hierarchy)
extractDependencies(ext/int)
trackExecutionFlow(conditions, loops, calls)
alt
Syntax Errors
loop
Errors
logError(line, issue)
generatePartialReport()
notifySyntaxIssues
display("Syntax issues detected. Partial results generated.")
generateComprehensiveReport()
saveResults(report, elements, deps, flow, projectId)
Saved
analysisSummary
displaySummaryOnDashboard()
reviewResults()
Developer
System
Code Analyzer
Database

Figure 5: Code Analysis & Structure Extraction sequence diagram

- Project Management (UC-07):

Table 13: Project Management usecase specification

Use case name:	Project Management
Participating Actors:	Developer Admin
Main Flow:	1. The Developer selects the "Project Management" option from the dashboard. 2. The system displays a list of the Developer's current projects. 3. The Developer selects one of the following main actions: a. **Create New Project**: The Add Project sub-use case is triggered b. **Modify Existing Project**: The Edit Project sub-use case is triggered. c. **Delete Project**: The Delete Project sub-use case is triggered. d. **Review Project Status**: The Developer selects a project to review compatibility reports (from UC-03) and generation logs (from UC-06). 4. The system displays a confirmation message for the successful action (add/edit/delete).
Alternative Flows	First alternative flow A1: Modify Settings Without Code Change • The Developer selects the project, then chooses "Edit Project." • The Developer modifies only the project settings (e.g., project name, description, or default generation language). • The system updates the data. • The system displays a success confirmation message and returns to the project list. (No need to run UC-03). Second alternative flow A2:No Prior Projects Exist • The Developer selects "Project Management." • The system finds the project list empty. • The system displays a message: "No current projects" and suggests a link or button for "Create New Project" (which executes Add Project). • The **Developer** proceeds to create a new project.
Exception Flows	Exception 1: Deletion Rule Violation • The Developer chooses "Delete Project" for a project linked to a final generated document. • The system rejects the deletion request. • The system displays a clear error message: "**Deletion Failed:** You must unpublish/delete the associated generated documents first." • The project remains in the database (Postcondition: On Failure). Exception 2:Database Connection Failure • The Developer attempts to add, modify, or delete a project. • The system fails to establish a connection to the database to execute the change. • The system displays a technical error message: "Failed to save changes. Please try again later." • The project status remains unchanged (Postcondition: On Failure).
Preconditions	• The **Developer** must be successfully signed in.
Postconditions	**On Success:** • The project database is updated to include the new project status (added, modified, or deleted). • An Audit Log is created recording the operation type, date, and the

	Developer who performed it. • The User Interface (UI) is updated to reflect the new list of projects. **On Failure:** • The project status remains unchanged in the database. • A clear error message is displayed, preventing the operation and explaining the reason. • Navigation to other pages is prevented to ensure the error is addressed first.

<u>Sequence diagram:</u>

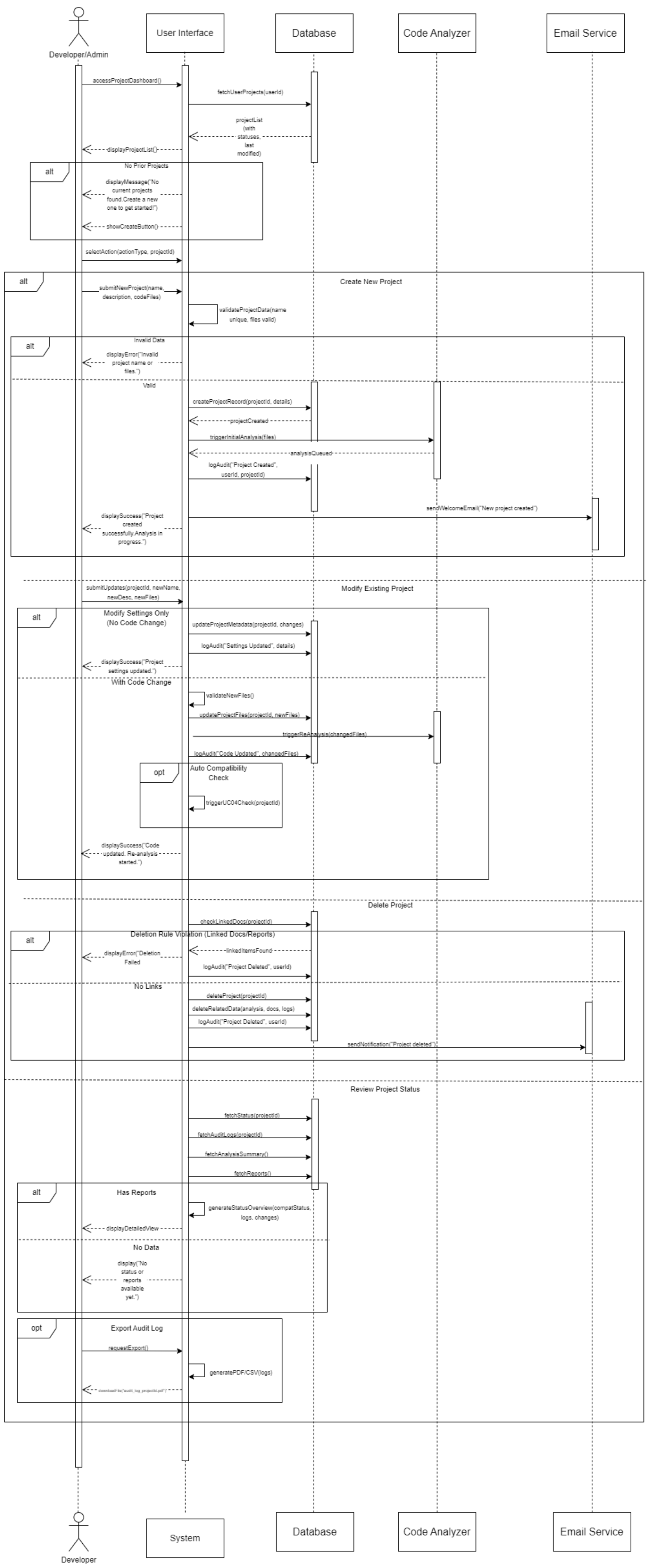
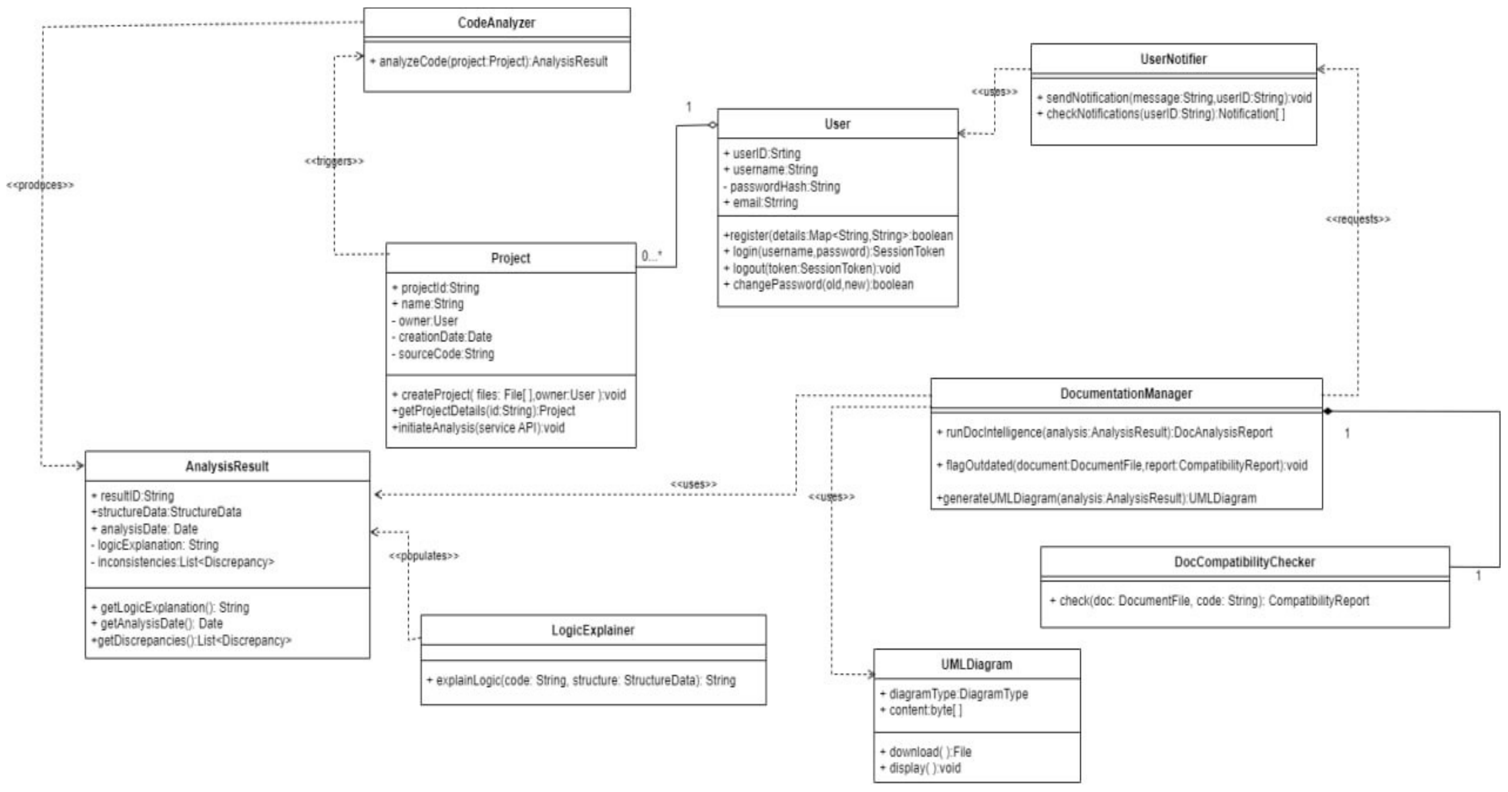

Developer/Admin
User Interface
Database
Code Analyzer
Email Service
accessProjectDashboard()
fetchUserProjects(userId)
projectList (with statuses, last modified)
displayProjectList()
alt
No Prior Projects
displayMessage("No current projects found.Create a new one to get started!")
showCreateButton()
selectAction(actionType, projectId)
alt
Create New Project
submitNewProject(name, description, codeFiles)
validateProjectData(name unique, files valid)
alt
Invalid Data
displayError("Invalid project name or files.")
Valid
createProjectRecord(projectId, details)
projectCreated
triggerInitialAnalysis(files)
analysisQueued
logAudit("Project Created", userId, projectId)
sendWelcomeEmail("New project created")
displaySuccess("Project created successfully.Analysis in progress.")
Modify Existing Project
submitUpdates(projectId, newName, newDesc, newFiles)
alt
Modify Settings Only (No Code Change)
updateProjectMetadata(projectId, changes)
logAudit("Settings Updated", details)
displaySuccess("Project settings updated.")
With Code Change
validateNewFiles()
updateProjectFiles(projectId, newFiles)
triggerReAnalysis(changedFiles)
logAudit("Code Updated", changedFiles)
opt
Auto Compatibility Check
triggerUC04Check(projectId)
displaySuccess("Code updated. Re-analysis started.")
Delete Project
checkLinkedDocs(projectId)
alt
Deletion Rule Violation (Linked Docs/Reports)
linkedItemsFound
displayError("Deletion Failed
logAudit("Project Deleted", userId)
No Links
deleteProject(projectId)
deleteRelatedData(analysis, docs, logs)
logAudit("Project Deleted", userId)
sendNotification("Project deleted")
Review Project Status
fetchStatus(projectId)
fetchAuditLogs(projectId)
fetchAnalysisSummary()
fetchReports()
alt
Has Reports
generateStatusOverview(compatStatus, logs, changes)
displayDetailedView
No Data
display("No status or reports available yet.")
opt
Export Audit Log
requestExport()
generatePDF/CSV(logs)
downloadFile("audit_log_projectId.pdf")
Developer
System
Database
Code Analyzer
Email Service

Figure 6: Project Management sequence diagram

Analysis class diagram:


CodeAnalyzer
+ analyzeCode(project:Project):AnalysisResult
<<produces>>
<<triggers>>
1
0...*
User
+ userID:Srting
+ username:String
- passwordHash:String
+ email:Strring
+register(details:Map<String,String>:boolean
+ login(username,password):SessionToken
+ logout(token:SessionToken):void
+ changePassword(old,new):boolean
<<uses>>
UserNotifier
+ sendNotification(message:String,userID:String):void
+ checkNotifications(userID:String):Notification[ ]
<<requests>>
Project
+ projectId:String
+ name:String
- owner:User
- creationDate:Date
- sourceCode:String
+ createProject(files: File[ ],owner:User):void
+getProjectDetails(id:String):Project
+initiateAnalysis(service API):void
DocumentationManager
+ runDocIntelligence(analysis:AnalysisResult):DocAnalysisReport
+ flagOutdated(document:DocumentFile,report:CompatibilityReport):void
+generateUMLDiagram(analysis:AnalysisResult):UMLDiagram
1
AnalysisResult
+ resultID:String
+structureData:StructureData
+ analysisDate: Date
- logicExplanation: String
- inconsistencies:List<Discrepancy>
+ getLogicExplanation(): String
+ getAnalysisDate(): Date
+getDiscrepancies():List<Discrepancy>
<<uses>>
<<uses>>
<<populates>>
DocCompatibilityChecker
+ check(doc: DocumentFile, code: String): CompatibilityReport
1
LogicExplainer
+ explainLogic(code: String, structure: StructureData): String
UMLDiagram
+ diagramType:DiagramType
+ content:byte[ ]
+ download( ):File
+ display( ):void

1

Figure 7: Class diagram

Entity Relationship Diagram ERD:

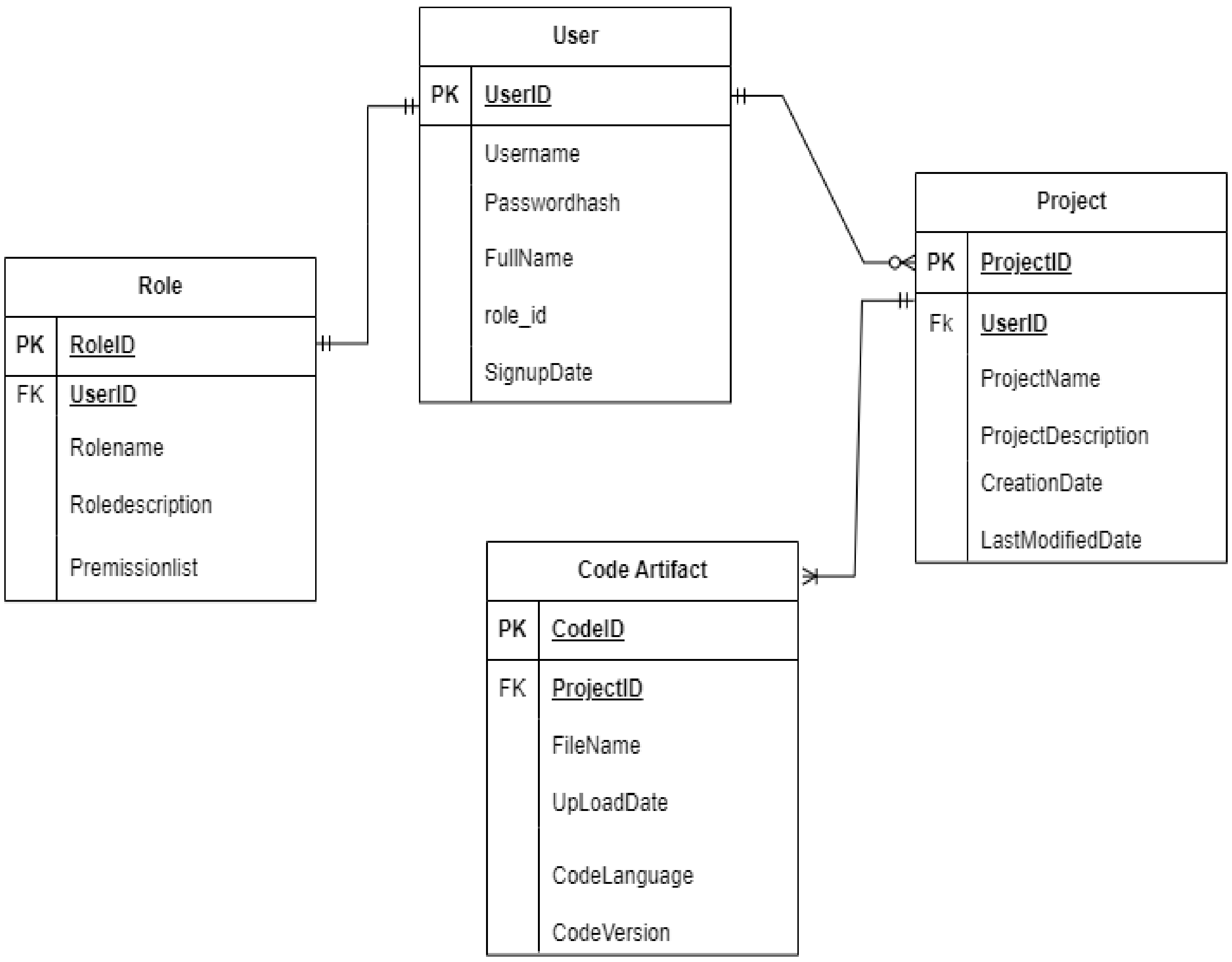


Figure 8: Entity Relationship Diagram ERD

5. Initial Test Cases:

Table 14: Initial Test Cases

Test Case Senario:		**Sce-01:Authentication & Authorization Test Cases**	
Test case id	Test case title	Test steps	Expected result
Tc-01	Successful Login Verification	1. Enter correct credentials. 2. Click "Login".	User logs in successfully and is redirected to the Main Dashboard.
Tc-02	Login Failure with Incorrect Password	1. Enter a correct username/email and an incorrect password. 2. Click "Login".	The system displays a general error message: "Username or password is incorrect".
Tc-03	New User Account Creation Verification	1. Complete the "Sign Up" form with all required, unique fields. 2. Click "Register".	The account is created successfully, and the user is either logged in or redirected to an email confirmation step.
Tc-04	Password Strength	Try to register with a 3-character password.	System rejects the password and requires a minimum of 8 characters.

Test Case Senario:		**Sce-02: Code Analysis & Structure Extraction Test Cases**	
Test case id	Test case title	Test steps	Expected result
Tc-05	Successful Analysis of a Supported Language (Python)	1. Upload a valid, small Python project file. 2. Click "Generate Class Diagram".	Analysis completes successfully. All Classes, Functions, and Dependencies are extracted and stored and class diagram appered.
Tc-06	Analysis of an Unsupported Language	1. Upload a project written in a language not listed in the supported list. 2. Click "Generate Class Diagram".	The system displays a clear error message: "Programming language is currently unsupported".
Tc-07	Analysis with Syntax Errors in Code	1. Upload a project containing a deliberate, critical syntax error. 2. Start Generating Class Diagram	The system logs the syntax errors and provides a partial class diagram with a warning to the user.
Tc-08	Code Upload Failure (Invalid File Type/Size)	1. Attempt to upload a file exceeding the maximum size limit OR a non-code file. 2. Observe the upload process.	The system immediately rejects the file upload, displaying a specific error message about size or type restrictions.

Test Case Senario:		Sce-03: Documentation Intelligence & Reporting Test Cases	
Test case id	Test case title	Test steps	Expected result
Tc-09	Detection of Outdated Documentation.	1. Ensure documentation exists for a function (X) whose name was recently changed to (Y) in the code. 2. Run the Documentation Intelligence function.	The documentation for function (X) is flagged as "Outdated" and the system suggests updating the name to (Y).
Tc-10	Notification of Required Documentation Updates.	1. Run a Documentation Intelligence analysis that detects a discrepancy. 2. Check the user notifications.	The user receives an in-system notification that documentation sections require updates.
Tc-11	UML Diagram Generation Verification.	1. Analysis of the project structure is complete. 2. Request the generation of a Class Diagram.	A correct UML Class Diagram representing the extracted code structure is displayed or downloaded.
Tc-12	Incompatible Document/Code Version Check.	1. Upload code from version A and attempt to generate documentation based o outdated standards from version B. 2. Request the generation of documents (UC-06).	The system issues a warning or blocks the documentation process, noting the incompatibility between code/standard versions (UC-04).
Tc-13	Code Logic Explanation Verification.	1.Select a complex function/module within a project. 2. Request “Code Logic Explanation” (UC-08). 3. The user choose between hight level or low level exeplination.	The system generates a clear, accurate, and step-by-step natural language explanation of the selected code’s purpose.

Test Case Senario:		Sce-04: Project Management Test Cases	
Test case id	Test case title	Test steps	Expected result
Tc-14	New Project Creation via Upload.	1. Click "Create New Project". 2. Add project name and description. 3. Click “Create Project”.	The project is created successfully, a unique ID is assigned, and the system logs the successful creation in its history
Tc-15	Code logic exeplination Report Download.	1. Ensure the exeplination is complete. 2. Navigate to the Reports page and click "Download".	The code logic exeplination report file is downloaded in a supported format.
Tc-16	Project Summary Display on Dashboard.	1. Log in. 2. View the Main Dashboard.	The dashboard displays a list of recent projects, their status, and quick statistics.

Tc-17	Successful Projet Edit (Rename/Update settings).	1. Navigate to an existing project's settings page. 2. Modify the project name. 3. Save the changes.	The project name is updated successfully across all views (Dashboard, project list), and the change is looged.
Tc-18	Successful Project Deletion	1. Navigate to the settings page for a test project. 2. Click "Delete Project" and confirm the action	The project is permanently removed from the system and no longer appears on the Dashboard or in the Project list.

6. Initial RTM Requirements Trackability Matrix:

Table 15: Initial RTM Requirements Trackability Matrix

req-id	Use cases	analysis	System design	Detailed design	coding	Test cases
RE-FR-01	Uc-01	SI-AN				Tc-01 Tc-02
RE-FR-02	Uc-01	SI-AI				Tc-01 Tc-02
RE-FR-03	Uc-02	SU-AN				Tc-03 Tc-04
RE-FR-04	Uc-03	DC-AN				Tc-16
RE-FR-05	Uc-04	CA-AN				Tc-05 Tc-06 Tc-07
RE-FR-06	Uc-04	CA-AN				Tc-05 Tc-06 Tc-07
RE-FR-07	Uc-04	CA-AN				Tc-05 Tc-06 Tc-07
RE-FR-08	Uc-05	Generate documents description				Tc-08 Tc-09 Tc-10
RE-FR-0 9	Uc-05	Generate documents description				Tc-08 Tc-09 Tc-10
RE-FR-10	Uc-05	Generate documents description				Tc-08 Tc-09 Tc-10
RE-FR-11	Uc-06	Code logic explanation description				Tc-17

RE-FR-12	Uc-06	Code logic explanation description	Tc-17
RE-FR-13	Uc-06	Code logic explanation description	Tc-17
RE-FR-14	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-15	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-16	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-17	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-18	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-19	Uc-07	PM-AN	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-20	Uc-08	Upload document description	Tc-15

Chapter 5 System Design

1. Introduction:

This chapter provides an in-depth look at the essential design stages, starting with establishing the **System Architecture (Architectural Design)** to define the overall structure and ensure scalability and performance. It proceeds through the **Detailed Design** of components, leading up to **Modeling** using **UML** diagrams.

These steps are vital for building a robust and maintainable system, capable of achieving its core goal: automatic code analysis, generating diverse Software Tests (Test Cases), and efficiently producing up-to-date Technical Documentation (API Docs and UML Diagrams). This process ultimately enhances software quality and streamlines continuous development.

2. System Architecture:

In this section, we will discuss **System Design**, which involves defining the overall structure and architecture of the software, ensuring that all components work seamlessly and harmoniously together. This phase establishes the fundamental blueprint for efficient data flow, interaction, and integration within the system.

The first constraints to take into consideration when building an architecture for a software system are the **Non-Functional Requirements (NFRs)** (or quality attributes). Therefore, we will start first by defining these requirements for the **AutoTest & DocGen Manager** system:

1. Performance:

- Analysis Response Time: Code analysis and documentation must be completed in less than 10 seconds for projects under 10MB.
- Concurrent Processing: The system must be capable of efficiently handling at least 50 simultaneous analysis.
- Dashboard Loading: The Main Dashboard loading time should not exceed 3 seconds to display instantaneous statistics.

2. Usability:

- Interaction Ease: The interface must support drag-and-drop for project uploads and feature a simple, developer-friendly design.
- Language Support: The system must support multiple natural languages (e.g., English and Arabic) for code explanations and interface text.
- Comprehensive Dashboard: A single main dashboard must provide a quick project overview, key statistics (e.g., code coverage, execution time), and actionable insights.
- The interface should be responsive and compatible with both desktop and tablet screens.

3. Security:

- Authentication: The system must support Single Sign-On (SSO) in addition to standard user login and account creation.
- Integration Security: Secure authorization mechanisms (e.g., OAuth) must be used and all access tokens must be protected.
- All user passwords must be encrypted using industry-standard hashing algorithms (e.g., BCrypt).

4. Maintainability:

- Project History: The system must maintain a complete record of project versions and history, including all completed analysis, documentation, and test results.
- Self-Documentation: The system should automatically contribute to the documentation of its own code and structure through its outputs (API docs, UML diagrams).

5. Availability:

- The system should maintain an uptime of 99.5% to ensure developers can access their documentation anytime.

<u>System decomposition - Component diagram:</u>

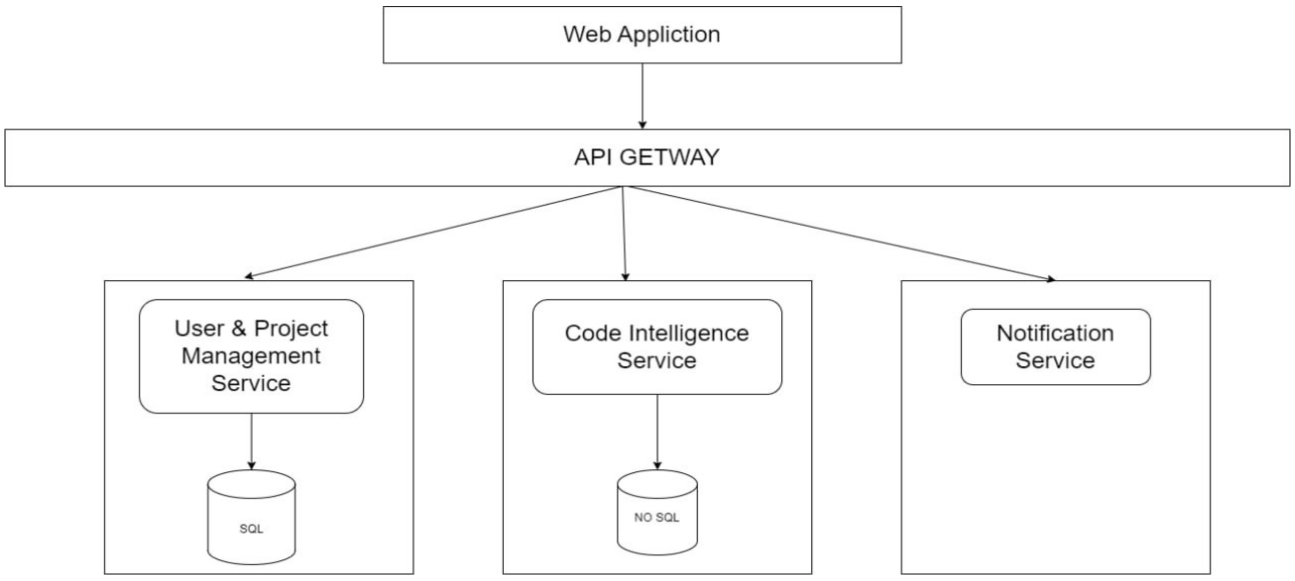


Figure 9: Component diagram

Components functionalities:

1. **API Gateway (API-PL-01):**Function: Serves as the single entry point for the entire system, responsible for routing all external web requests to the appropriate Microservice.
2. **User & Project Management Service (UPM-BLL-03)**:Function: A dedicated microservice specializing in managing user accounts, project settings, and project specifications.
3. **Code Intelligence Service (CI-BLL-04):**Function: A microservice dedicated to processing and analyzing complex data (like reviews or code) using AI algorithms.
4. **Notification Service (NS-BLL-05):** Function: A microservice responsible for processing and sending all types of alerts and notifications to users.
5. **SQL Database Access (SDA-DAL-06):**Responsible for communication with the SQL database of the system
6. **NoSQL Database Access (NSDA-DAL-07):**Responsible for communication with the NoSQL database of the system

System Architecture:

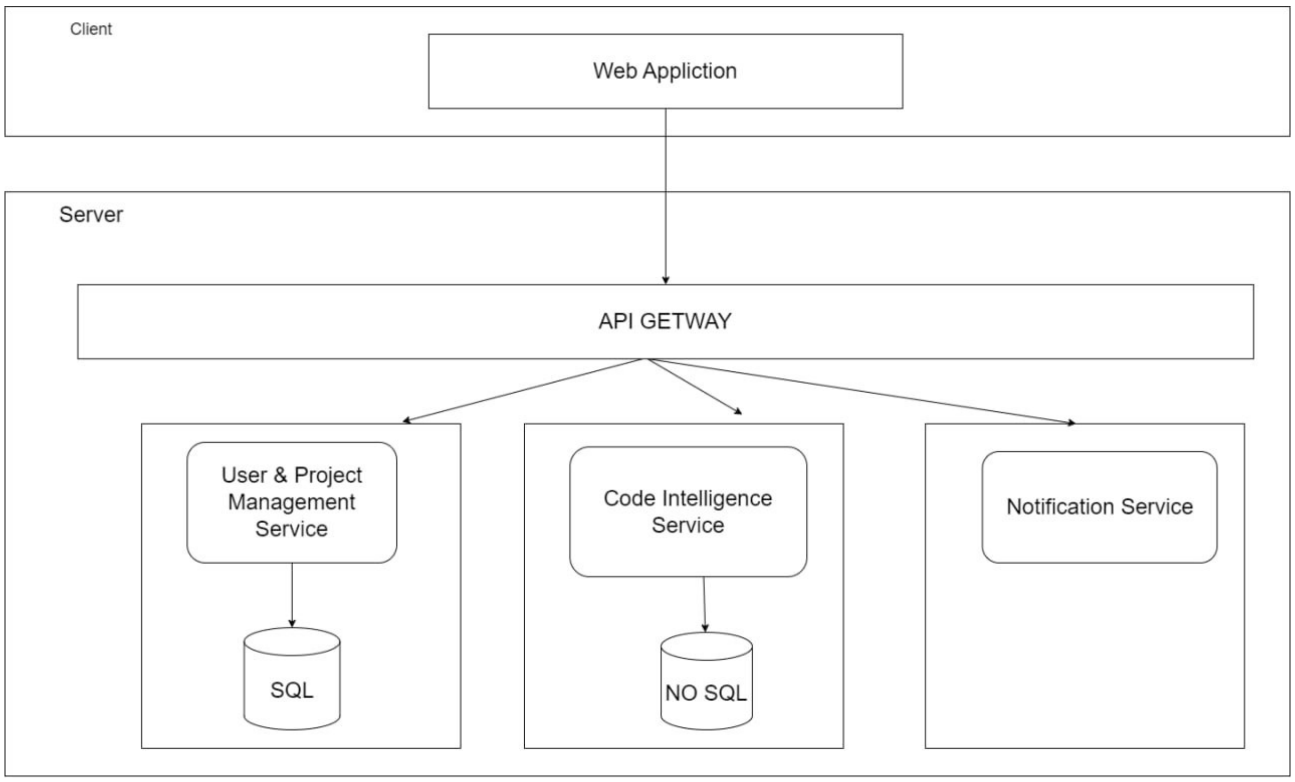


Figure 10: System Architecture

System Architecture Explanation:

Architecture Used:

- Microservices architecture with Client-Server

1. **Client Layer**

- Component: Web Application (Frontend user interface).
- Function: Provides the user interface for interacting with the system.
- Technology: (Retained: React Framework - Assuming, as it's common for web applications).

2. **Server - Microservices Architecture**

- Technology: Django Framework

-API Gateway

- Function: Acts as the Single Entry Point for all requests from the Client Layer.
- Role: Responsible for Routing requests to the appropriate Microservice.
- Components: Serves as the interface for the 3 core services below.

Independent Microservices:

➢ **User & Project Management Service**

- Independent service for managing user Identity & Authorization, along with Project Management functionalities and requirements.
- Database: SQL

➢ **Code Intelligence Service**

- Independent service for Code Processing, Documentation, and sophisticated Artificial Intelligence analysis.

- Database: NO SQL

➢ **Notification Service**

- Independent service dedicated entirely to handling and sending all forms of notifications and alerts.

- Database: NoSQL

3. Databases

- SQL Databases: separate database (Dedicated to User & Project Management).

- NoSQL/MongoDB: 1 separate database (Dedicated to Code Intelligence).

3. Detailed design for the system components:

This section provides an in-depth exploration of the system's individual components, highlighting the architectural decisions, design strategies, and patterns employed to ensure the software is robust, maintainable, and aligned with its functional and non functional requirements.

Used Design patterns:

- Language Processor status:

The State Pattern is a design solution that allows an object to change its behavior when its internal state changes, making it appear as if it changes its class.The principle lies in separating the behavior associated with each state (such as the "In Progress" status of an order) into an independent class.This ensures a flexible and extensible system, where new states can be added in the future without modifying the core code of the object.

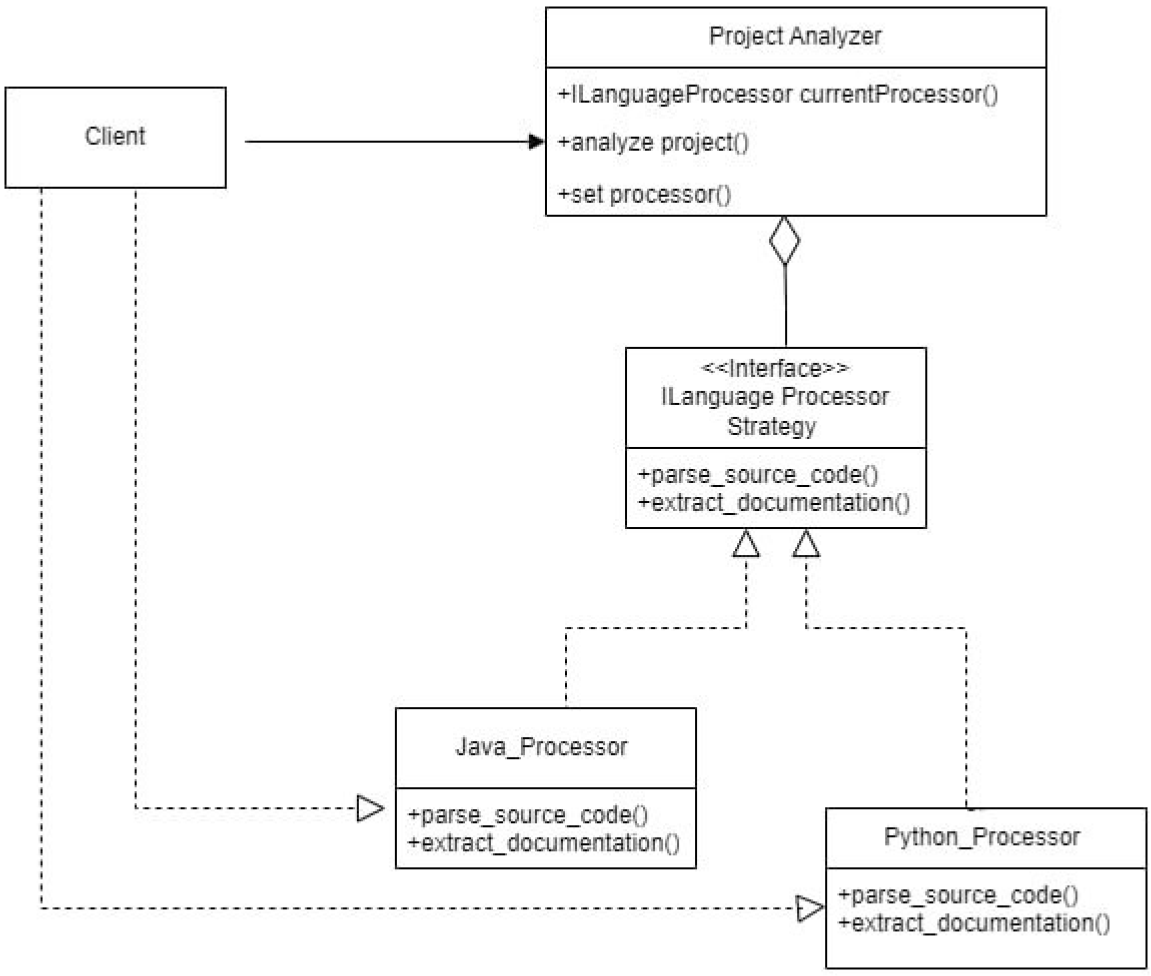


Figure 11: Language processor design pattern

➢ Report Output Status:

ThIS diagram implements the Template Method pattern, where the Report_Generator class provides a fixed sequence of steps for report creation. The final step, export_document(), is deferred to the subclasses MarkdownOutput and PDFOutput, allowing the output format to be changed without altering the core analysis logic.

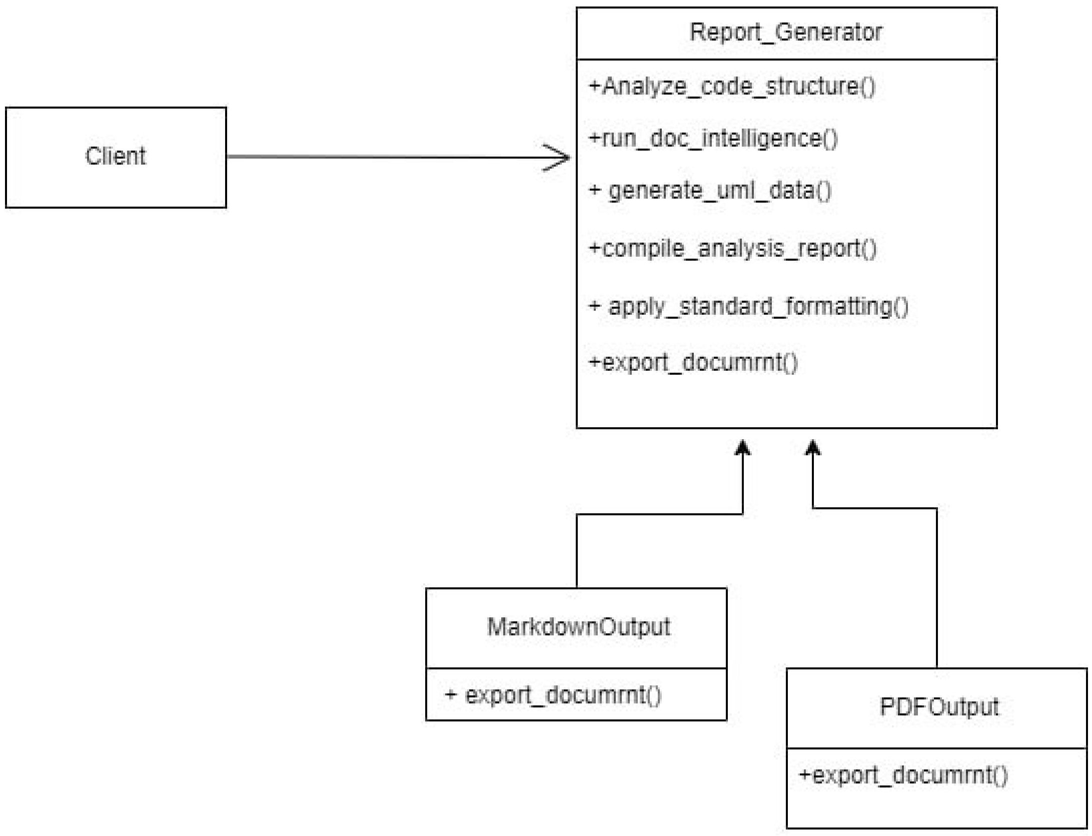
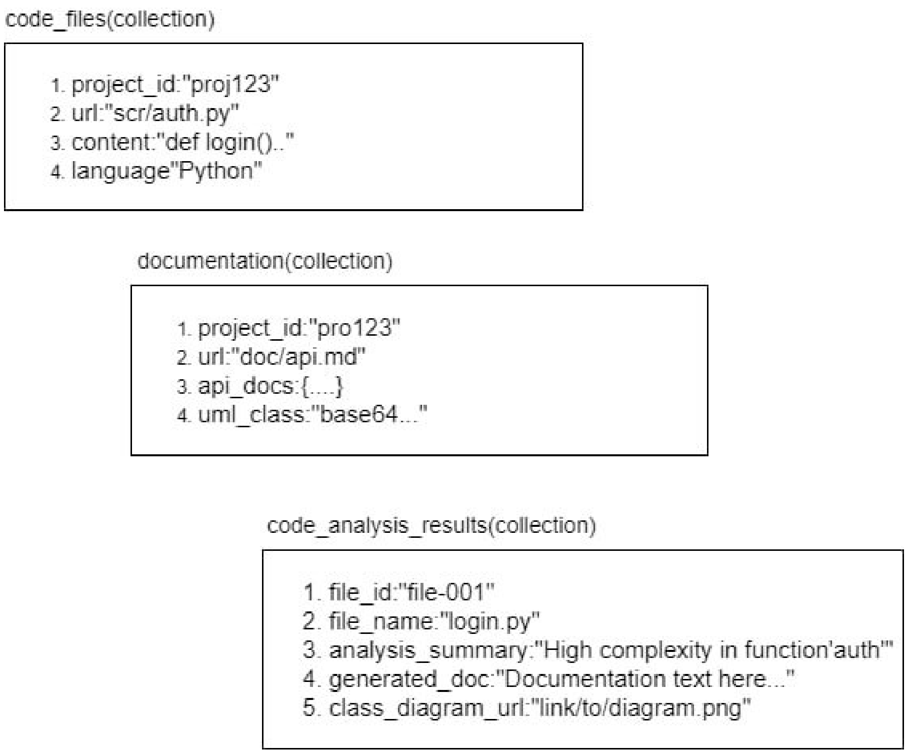


Figure 12: Report output design pattern

4. Database Design :

NoSQL Database – MongoDB: For data related to code-files, documentation and code_analysis_result , we employ MongoDB. This NoSQL database is well-suited for handling large volumes of unstructured data, providing the flexibility and scalability needed to manage diverse and dynamic datasets efficiently.



code_files(collection)

1. project_id:"proj123"
2. url:"scr/auth.py"
3. content:"def login().."
4. language"Python"

documentation(collection)

1. project_id:"pro123"
2. url:"doc/api.md"
3. api_docs:{....}
4. uml_class:"base64..."

code_analysis_results(collection)

1. file_id:"file-001"
2. file_name:"login.py"
3. analysis_summary:"High complexity in function'auth'"
4. generated_doc:"Documentation text here..."
5. class_diagram_url:"link/to/diagram.png"

Figure 13: NoSQL database

SQL Database – MySQL: In our system, MySQL is utilized for storing operational data related to accounts, projects, and requirements. This data is highly structured, making MySQL an ideal choice due to its efficiency in handling defined schemas and transactional integrity.

MySQL- database modeling:

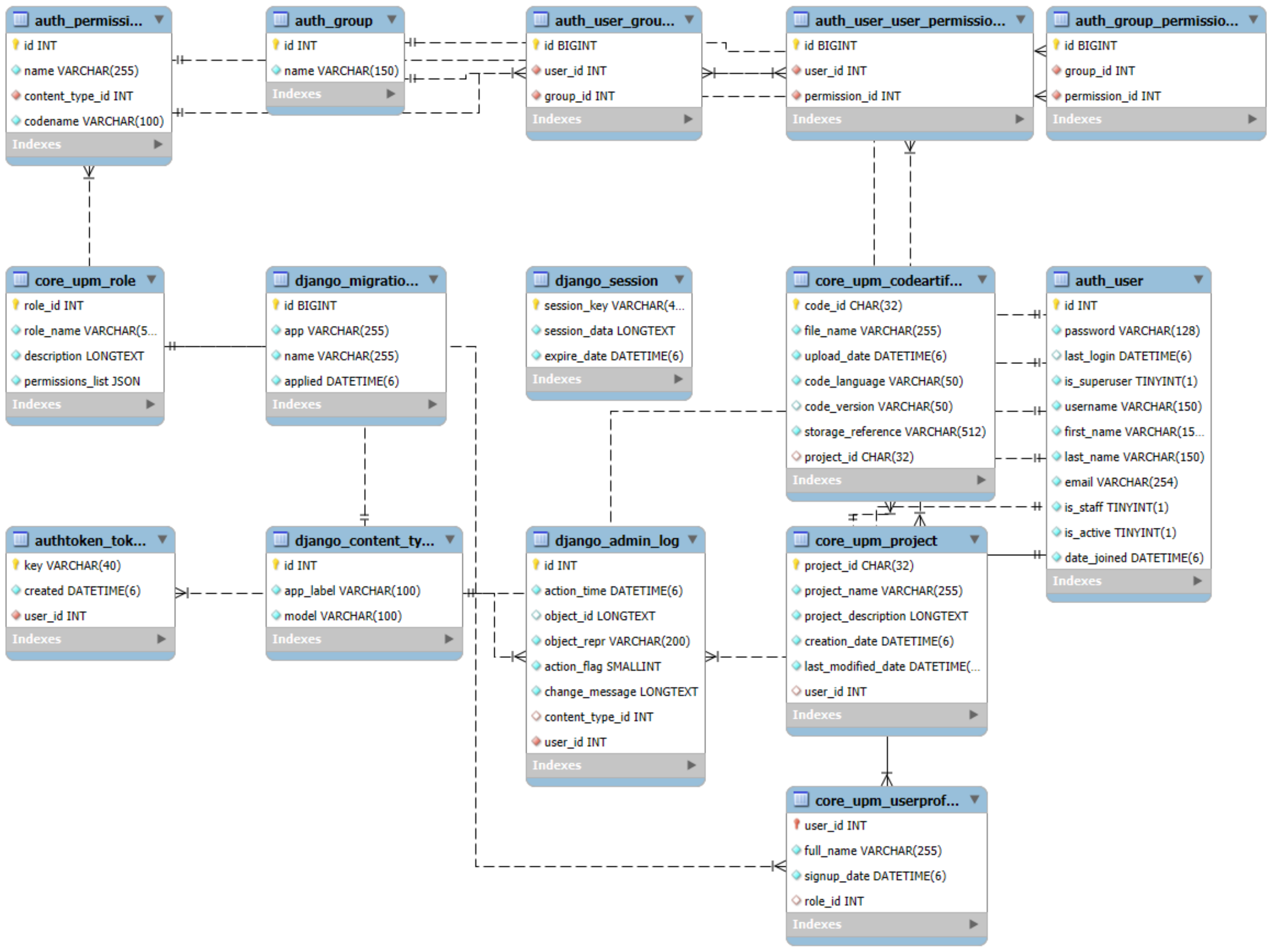


Figure 14: MySQL database modeling

5. Updating the RTM:

Table 16: Updating RTM

req-id	Use cases	analysis	System design	Detailed design	coding	Test cases
RE-FR-01	Uc-01	SI-AN	SI-BLL-03			Tc-01 Tc-02
RE-FR-02	Uc-01	SI-AI	SI-BLL-03			Tc-01 Tc-02
RE-FR-03	Uc-02	SU-AN	SU-BLL-03			Tc-03 Tc-04
RE-FR-04	Uc-03	DC-AN	DC-BLL-04			Tc-16
RE-FR-05	Uc-04	CA-AN	CA--BLL-04	DD-LPD		Tc-05 Tc-06 Tc-07
RE-FR-06	Uc-04	CA-AN	CA-BLL-04	DD-LPD		Tc-05 Tc-06 Tc-07
RE-FR-07	Uc-04	CA-AN	CA-BLL-04	DD-LPD		Tc-05 Tc-06 Tc-07
RE-FR-08	Uc-05	Generate documents description	GD-BLL-04	DD-ROD		Tc-08 Tc-09 Tc-10
RE-FR-0 9	Uc-05	Generate documents description	GD-BLL-04	DD-ROD		Tc-08 Tc-09 Tc-10
RE-FR-10	Uc-05	Generate documents description	GD-BLL-04	DD-ROD		Tc-08 Tc-09 Tc-10
RE-FR-11	Uc-06	Code logic explanation description	CLE-BLL-04			Tc-17
RE-FR-12	Uc-06	Code logic explanation description	CLE-BLL-04			Tc-17
RE-FR-13	Uc-06	Code logic explanation description	CLE-BLL-04			Tc-17
RE-FR-14	Uc-07	PM-AN	PM-BLL-03			Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-15	Uc-07	PM-AN	PM-BLL-03			Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-16	Uc-07	PM-AN	PM-BLL-03			Tc-12 Tc-13

				Tc-14 Tc-15
RE-FR-17	Uc-07	PM-AN	PM-BLL-03	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-18	Uc-07	PM-AN	PM-BLL-03	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-19	Uc-07	PM-AN	PM-BLL-03	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-20	Uc-08	Upload document description	UD-BLL-04	Tc-15

Chapter 6 Practical Implementation

1. Introduction:

In this chapter, we will explain how the system was built in practice, including the tools and technologies used. We will also show the system's interfaces and finish by running test cases to check that the system works correctly in different situations.

2. Used Tools:

➢ Django:

is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. especially we use DRF: Django REST Framework is a widely used, full featured API framework designed for building RESTful APIs with Django. At its core, DRF integrates with Django's core features "models, views, and URLs" making it simple and seamless to create a RESTful API.

➢ React:

React is a popular JavaScript library for building user interfaces. It was created by Facebook and is widely used in web development. React allows developers to build reusable UI components that can efficiently update and render changes to the user interface when the underlying data changes. Reacts primary focus is on building user interfaces, and it excels in creating interactive and dynamic web applications.

➢ My SQL Database:

MySQL is an open-source relational database management system (RDBMS) that is widely used for storing, managing, and retrieving data. It 77 is one of the most popular and widely adopted databases in the world, known for its reliability, scalability, and ease of use.

➢ Visual Studio Code (VS Code):

Visual Studio Code combines the simplicity of a source code editor with powerful developer tooling, like IntelliSense code completion and debugging. We use it to develop the whole project (frontend, backend).

➢ HTML (HyperText Markup Language):

HTML is the fundamental markup language for creating web pages. It is used to organize and structure page content through elements and tags. HTML5 is the modern version that includes new semantic elements such as header, nav, section, and article, making the code clearer and more meaningful. It can be easily integrated with CSS and JavaScript to create interactive and responsive websites

➢ CSS (Cascading Style Sheets):

CSS is a styling language used to format and beautify HTML pages. It controls colors, fonts, layout, spacing, and all visual aspects of the website. CSS3 includes advanced features such as transitions, animations, Flexbox, and Grid Layout. It supports responsive design, ensuring the website appears perfectly on all devices from smartphones to desktop computers.

➢ Draw.io:

Draw.io (now known as diagrams.net) is a free and open-source web application for creating diagrams and flowcharts. It works directly from your browser without needing to download any software, and provides a comprehensive set of tools for creating various types of professional diagrams easily

➔ Technology Justification :

- **Frontend (React):** Chosen for its component-based architecture which allows for building a highly interactive and dynamic dashboard for real-time analysis updates.
- **Backend (Python/FastAPI):** Selected due to Python's superior ecosystem for AI and source code parsing libraries, and FastAPI's high performance in handling asynchronous tasks.
- **AI Engine:** Utilized to bridge the gap between complex technical logic and human-readable documentation, significantly reducing manual effort for developers.

3. AI Used Technologies and Tools:

This section outlines the diverse set of technologies, frameworks, and specialized tools utilized to develop the **AutoTest & DocGen Manager**. The architecture is built on a robust stack to ensure scalability, intelligence, and high performance.

1. Core Development Frameworks

- **Django (Python Framework):** Used as the primary backend framework for its security features, scalability, and seamless integration with the **Django Admin** for custom MongoDB management.

- **React.js:** The frontend library used to build a dynamic, responsive, and interactive user interface (UI), focusing on professional styling and user experience.

2. Artificial Intelligence & LLM Integration

- **Multi-Agent Architecture:** A sophisticated orchestration of three specialized AI agents:
    - **HighLevelAgent:** For executive summaries.
    - **LowLevelAgent:** For deep technical code explanation.
    - **VerifierAgent:** For quality assurance and output consistency.
- **Large Language Models (LLMs):** Utilized for natural language generation and code reasoning.
- **Exponential Backoff Algorithm:** Implemented in the AI client to handle Rate Limiting (Error 429) and ensure 99.9% API request stability.

3. Code Analysis & Parsing Tools

- **Tree-sitter (Java):** A concrete syntax tree parsing library used to extract precise structures from Java source code.
- **AST (Abstract Syntax Trees - Python):** Leveraged for deep structural analysis of Python code, identifying classes, methods, and complex dependencies.
- **Intelligent Code Chunking:** A custom-built algorithm to manage LLM context window constraints by splitting large files into logical units.

4. Database & Storage Solutions

- **MongoDB:** A NoSQL database used to store flexible code structures, metadata, and generated documentation.
- **Smart Caching System:** A high-performance caching layer designed to store generated explanations, reducing API costs and latency by up to 80%.

5. Documentation & Visualization Engines

- **Mermaid.js:** Integrated for dynamic generation of **UML Class Diagrams**, automatically detecting relationships like Inheritance and Composition.

- **ReportLab / PDF Engines:** Used for high-quality, professional PDF exporting with custom branding (themes, headers, and footers).
- **Markdown Parsers:** To support developer-friendly documentation formats.

6. Version Control & DevOps

- **Git & GitHub:** For version control, utilizing a strict **Pull Request (PR)** and code review workflow to maintain system integrity.
- **Branching Strategy:** Systematic use of feature branches (V1, V2, V3) and stable tagging for milestone releases.

4. System Interfaces:

5. Test Cases Execution:

6. RTM Last Version:

Table 17: RTM last version

req-id	Use cases	analysis	System design	Detailed design	coding	Test cases
RE-FR-01	Uc-01	SI-AN	SI-BLL-03		UPM_Project	Tc-01 Tc-02
RE-FR-02	Uc-01	SI-AI	SI-BLL-03		UPM_Project	Tc-01 Tc-02
RE-FR-03	Uc-02	SU-AN	SU-BLL-03		UPM_Project	Tc-03 Tc-04
RE-FR-04	Uc-03	DC-AN	DC-BLL-04			Tc-16
RE-FR-05	Uc-04	CA-AN	CA--BLL-04	DD-LPD	Ai_project	Tc-05 Tc-06 Tc-07
RE-FR-06	Uc-04	CA-AN	CA-BLL-04	DD-LPD	Ai_project	Tc-05 Tc-06 Tc-07
RE-FR-07	Uc-04	CA-AN	CA-BLL-04	DD-LPD	Ai_project	Tc-05 Tc-06 Tc-07
RE-FR-08	Uc-05	Generate documents description	GD-BLL-04	DD-ROD	Ai_project	Tc-08 Tc-09 Tc-10
RE-FR-0 9	Uc-05	Generate documents description	GD-BLL-04	DD-ROD	Ai_project	Tc-08 Tc-09 Tc-10
RE-FR-10	Uc-05	Generate documents description	GD-BLL-04	DD-ROD	Ai_project	Tc-08 Tc-09 Tc-10

RE-FR-11	Uc-06	Code logic explanation description	CLE-BLL-04	Ai_project	Tc-17
RE-FR-12	Uc-06	Code logic explanation description	CLE-BLL-04	Ai_project	Tc-17
RE-FR-13	Uc-06	Code logic explanation description	CLE-BLL-04	Ai_project	Tc-17
RE-FR-14	Uc-07	PM-AN	PM-BLL-03	Ai_project	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-15	Uc-07	PM-AN	PM-BLL-03	Ai_project	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-16	Uc-07	PM-AN	PM-BLL-03	Ai_project	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-17	Uc-07	PM-AN	PM-BLL-03		Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-18	Uc-07	PM-AN	PM-BLL-03	UPM_Project	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-19	Uc-07	PM-AN	PM-BLL-03	UPM_Project	Tc-12 Tc-13 Tc-14 Tc-15
RE-FR-20	Uc-08	Upload document description	UD-BLL-04	UPM_Project	Tc-15

Chapter 7 AI Section

1. Introduction:

Building the AutoTest & DocGen Manager

1) The Challenge of Software Quality and Developer Efficiency

At the core of modern software engineering lie two main objectives: **High Code Quality** and **Maximum Developer Efficiency**. Development teams face a dual challenge:

- **Comprehensive Testing:** Ensuring complete and accurate coverage of all system units and integrations (Unit and Integration Testing)—a process that is significantly time-consuming and resource-intensive when performed manually.
- **Updated Documentation:** Maintaining technical documentation (Technical Documentation) and code documentation (Code Documentation) as current and consistent with rapid code changes requires continuous effort. Negligence often leads to **Stale Documentation** that hinders system understanding and maintenance.

From this perspective, the **"AutoTest & DocGen Manager"** system was born. It is not designed merely as an auxiliary tool but as an automated partner intended to transform the development process by automating complex cognitive-heavy tasks. Its core importance lies in elevating the quality of outputs (code and documentation) and freeing developers to focus on innovation.

2) The Technical Dilemma and the Search for Reliable AI

When considering the use of Artificial Intelligence (AI) to achieve this goal, we encountered a fundamental technical dilemma. Standard Large Language Models (LLMs), despite their incredible text generation power, have inherent limitations that make them insufficient on their own for critical software engineering applications:

1. **Lack of Structural Awareness:** Tasks like generating a UML class diagram or integration tests require understanding the **code structure** and dependencies, which general LLMs cannot do efficiently.

2. **Inconsistency:** An LLM might generate a comment or description that does not factually align with the code's current logic, leading to **Code-Comment Inconsistency (CCI))**.
3. **Absence of Standardized Evaluation:** Teams need documentation that follows specific templates (like Javadoc) and tests that cover edge cases. This requires models capable of generation and evaluation based on strict engineering standards.

It was clear we needed an architectural approach that ensures **control and consistency** and forces the AI model to operate as a guided engineering expert, rather than a general text generator.

3) The Strategic Choice: Why "Topological Processing and Multi-Agents"?

This is where the architecture of a **Multi-Agent System**, combined with **Topological Code Processing**, emerges as the ideal strategic solution. This approach is a design philosophy that goes beyond mere "generation" to focus on "Augmented Understanding" and "Verification."

The importance of this approach lies in its multi-step mechanism that directly integrates with the cited research papers:

1. **Topological Analysis (Augmented Retrieval):** Before generating anything, the code is analyzed incrementally (**Incremental Context Building**) to understand structural relationships and dependencies. This ensures that the output (whether documentation or a test) is **grounded** in the actual code structure *(Linked to DocAgent)*.
2. **Specialization and Collaboration (Multi-Agent):** Tasks are divided among specialized agents (e.g., Reader Agent, Verifier Agent, Writer Agent). This ensures that every piece of documentation or test passes through specialized stages to guarantee its completeness and reliability.
3. **Self-Correction:** The system is equipped with verification mechanisms like **CCISOLVER**, which automatically detects and

repairs code-comment inconsistencies, significantly reducing documentation errors and ensuring the accuracy of technical outputs *(Linked to CCISOLVER)*.

4. **Standardized Generation:** The language model is guided to create documentation that doesn't just describe but adheres to **specific templates** (Template-based generation) like Javadoc, and generates visual representations such as **UML Class Diagrams** based on code reverse engineering *(Linked to Automated and Context-Aware and GENERATING UML Class Diagram)*.

In summary, we turned to the specialized agents architecture because it transforms the language model from a vast "black box" of knowledge into a **"precise and guided software engineer"** that we can trust for critical testing and documentation tasks.

2. For the research papers that were relied upon :

The project relied on a selection of recent research and studies focusing on Large Language Models (LLMs) and their applications in software engineering, specifically in the fields of code documentation, summarization, and bug detection:

1. A Comparative Analysis of Large Language Models for Code Documentation Generation

- **Summary:** This paper presents a comprehensive comparative analysis of LLMs such as GPT-3.5, GPT-4, Bard, Llama2, and Starchat for generating code documentation.

2. DocAgent: A Multi-Agent System for Automated Code Documentation Generation

- **Summary:** Introduces **DocAgent**, a novel multi-agent collaborative system that utilizes topological code processing for incremental context building, leading to the generation of high-quality documentation.

3. Automated and Context-Aware Code Documentation Leveraging Advanced LLMs

- **Summary:** Aims to address the gap in template-based documentation generation (e.g., Javadoc) using publicly available LLMs, by developing a tailored, context-aware dataset and methodology.

4. Source Code Summarization in the Era of Large Language Models

- **Summary:** Presents a systematic and comprehensive study on code summarization using LLMs, covering multiple aspects involved in the LLM-based summarization workflow.

5. AI Based Code Documentation Generation

- **Summary:** Provides a survey of the latest research on LLMs in documentation generation, addressing challenges like intelligent code chunking, dependency resolution, and multi-language support.

6. CCISOLVER: End-to-End Detection and Repair of Method-Level Code-Comment Inconsistency

- **Summary:** Presents **CCISOLVER**, an innovative LLM-based framework designed to improve code quality by identifying and rectifying Code-Comment Inconsistencies (CCI) at the method level.

7. GENERATING UML CLASS DIAGRAM FROM SOURCE CODES USING MULTI-THREADING TECHNIQUE

- **Summary:** Discusses the reverse engineering process to extract UML Class Diagrams from source code, which is one of the key visual outputs required in the report.

3. The Final Comparison: A Decision-Making Framework:

Table 18: The Final Comparison AI

Paper Title	Abstract / Main Goal	Technique / Methodology Used	Key Findings / Results	Strategic Benefit to our Companion System (Student Project)	Reference Info
DocAgent: A Multi-Agent System for Automated Code Documentation	To introduce a novel multi-agent collaborative system for high-quality code documentation, addressing incomplete/incorrect outputs from existing LLMs.	A Multi-Agent System (Reader, Searcher, Writer, Verifier) utilizing Topological Code Processing for accurate incremental context building.	The multi-agent approach significantly outperformed baselines, proving the effectiveness of structured processing for context accuracy.	Methodology: Implement a simplified multi-step processing pipeline (Agent-Based Design) over a single LLM call to ensure accurate context understanding and enhance output reliability, critical for a robust PoC.	2504.08725
A Comparative Analysis of Large Language Models for Code Documentation Generation	A comprehensive comparative analysis of LLMs (GPT-3.5, GPT-4, Bard, Llama2, StarChat) for code documentation generation across various parameters.	Evaluation using a checklist-based system on parameters like Accuracy, Completeness, Relevance, Readability, and Time Taken.	Closed-source models (GPT-3.5/4) showed superior performance. All models tested (except StarChat) outperformed original documentation.	Model Choice: For initial high-quality results, leverage the GPT-3.5 Turbo API (cost-effective) and focus on prompt design. For an open-source solution, intensive fine-tuning of a smaller model is necessary to bridge the performance gap.	A Comparative Analysis of Large Language Models for Code Documentation Generation
Automated and Context-Aware Code Documentation Leveraging Advanced LLMs	To assess LLMs' capabilities for context-aware, template-based documentation (like Javadoc) and address the lack of tailored datasets.	Developing a novel context-aware Javadoc dataset. Evaluation of open-source LLMs (LLaMA-3.1, Gemma-2, etc.) using Zero/Few-Shot techniques.	Highlights the necessity of model guidance (e.g., through fine-tuning) to ensure adherence to standard, structured documentation formats.	Methodology: Use Prompt Engineering or incorporate a post-processing step to strictly ensure that the generated documentation adheres to industry-standard templates (e.g., Python docstrings, Javadoc) for professional output.	Automated and Context-Aware Code Documentation Leveraging Advanced LLMs
Source Code Summarization in the Era of Large	A systematic study on code summarization in	Systematic study focusing on prevailing	LLMs have significantly boosted	Evaluation: Incorporate a Quantitative	Source Code Summarization in the Era of

Language Models	the era of LLMs, covering the workflow and evaluation methods.	automated evaluation metrics (e.g., ROUGE, BLEU) used for assessing LLM-generated summaries.	summarization performance. The selection of the appropriate automatic evaluation metric is crucial.	Evaluation section in the project using standard automated metrics (e.g., ROUGE, BLEU) to scientifically measure and report the quality of the summaries and documentation generated by Companion.	Large Language Models
CCISOLVER: End-to-End Detection and Repair of Method-Level Code-Comment Inconsistency	To introduce CCISOLVER, an innovative LLM-based framework to detect and fix Code-Comment Inconsistency (CCI) at the method level.	Introduction of a refined dataset (CCIBENCH) and use of the LLM-based framework CCISOLVER for detection and repair.	Established new state-of-the-art results for both CCI detection and fixing, confirming LLMs' capability to ensure code-comment consistency.	Feature/ Methodology: Integrate an Automatic Validation/Auditing step into the Companion system using the LLM to verify that the generated documentation is logically consistent with the surrounding code, maximizing output trustworthiness.	CCISolver: End-to-End Detection and Repair of Method-Level Code-Comment Inconsistency
AI Based Code Documentation Generation	A survey reviewing challenges in handling large codebases, dependency resolution, and suggesting a system using intelligent code chunking.	Survey paper proposing a novel system combining intelligent code chunking and dependency resolution to handle large, multi-language codebases.	The primary challenge for LLMs is managing the context window when dealing with large files and complex module dependencies.	Pre-processing: Implement Intelligent Code Chunking (e.g., function-level or class-level splitting) to efficiently manage context window constraints, reduce API costs, and improve the quality of documentation for larger code files.	IJIRT