Multi-Agent AI System for Automated Software Development

Group Members

Aden Javed (01-131222-006)

Zohaib Shafqat (01-131222-052)

Supervisor: Dr. Awais Majeed

A Final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelor of Software Engineering

May 2026

FYP COMPLETION CERTIFICATE

Student Name: Aden Javed Enrolment No: 01-131222-006

Student Name: Zohaib Shafqat Enrolment No: 01-131222-052

Programme of Study: Bachelor of Software Engineering

Project Title: Multi-Agent AI System for Automated Software Development

It is to certify that the above students' project has been completed to my satisfaction and to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at _____ that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: _____ Name: Dr. Awais Majeed

CERTIFICATE OF ORIGINALITY

This is certify that the intellectual contents of the project Multi-Agent AI System for Automated Software Development are the product of my/our own work except, as cited properly and accurately in the acknowledgements and references, the material taken from such sources as research journals, books, internet, etc. solely to support, elaborate, compare, extend and/or implement the earlier work. Further, this work has not been submitted by me/us previously for any degree, nor it shall be submitted by me/us in the future for obtaining any degree from this University, or any other university or institution. The incorrectness of this information, if proved at any stage, shall authorities the University to cancel my/our degree.

Name of the Student: Aden Javed

Signature: _____ Date: _____

Name of the Student: Zohaib Shafqat

Signature: _____ Date: _____

PROJECT TITLE (MULTI AGENT AI SYSTEM FOR AUTOMATED SOFTWARE DEVELOPMENT)

Sustainable Development Goals

(Please tick the relevant SDG(s) linked with FYDP)

SDG No	Description of SDG	SDG No	Description of SDG
SDG 1	No Poverty	SDG 9	Industry, Innovation, and Infrastructure
SDG 2	Zero Hunger	SDG 10	Reduced Inequalities
SDG 3	Good Health and Well Being	SDG 11	Sustainable Cities and Communities
SDG 4	Quality Education	SDG 12	Responsible Consumption and Production
SDG 5	Gender Equality	SDG 13	Climate Change
SDG 6	Clean Water and Sanitation	SDG 14	Life Below Water
SDG 7	Affordable and Clean Energy	SDG 15	Life on Land
SDG 8	Decent Work and Economic Growth	SDG 16	Peace, Justice and Strong Institutions
		SDG 17	Partnerships for the Goals

Range of Complex Problem Solving			
	Attribute	**Complex Problem**	
1	Range of conflicting requirements	Involve wide-ranging or conflicting technical, engineering and other issues.	
2	Depth of analysis required	Have no obvious solution and require abstract thinking, originality in analysis to formulate suitable models.	
3	Depth of knowledge required	Requires research-based knowledge much of which is at, or informed by, the forefront of the professional discipline and which allows a fundamentals-based, first principles analytical approach.	
4	Familiarity of issues	Involve infrequently encountered issues	
5	Extent of applicable codes	Are outside problems encompassed by standards and codes of practice for professional engineering.	
6	Extent of stakeholder involvement and level of conflicting requirements	Involve diverse groups of stakeholders with widely varying needs.	
7	Consequences	Have significant consequences in a range of contexts.	
8	Interdependence	Are high level problems including many component parts or sub-problems	
Range of Complex Problem Activities			
	Attribute	**Complex Activities**	
1	Range of resources	Involve the use of diverse resources (and for this purpose, resources include people, money, equipment, materials, information and technologies).	
2	Level of interaction	Require resolution of significant problems arising from interactions between wide ranging and conflicting technical, engineering or other issues.	
3	Innovation	Involve creative use of engineering principles and research-based knowledge in novel ways.	
4	Consequences to society and the environment	Have significant consequences in a range of contexts, characterized by difficulty of prediction and mitigation.	
5	Familiarity	Can extend beyond previous experiences by applying principles-based approaches.	

Abstract

In this project, AIMARES (M.A.I.S. for Automating Software Development) is proposed. This unique system is responsible for automated creation of user stories that have a very high level of detail, specific regulations, API designs, and so on. Such requirements are demanded from engineers in order to start software development. Usually, there are initial problem statements when new projects arise. However, engineers need software specifications which often leads to miscommunication and waste of time spent on translation of general ideas into specific software needs.

In order to solve this issue, AIMARES was created with the help of another approach than the standard AI prompt one. The work of the tool is based on the deterministic multi-agent pipeline. This mechanism receives a generic specification of a project and transforms it into highly structured software specifications. In this regard, Next.js serves as an interface while FastAPI back-end executes the algorithm of this transformation process. Moreover, AIMARES relies on DDA and RAG (FAISS) mechanisms to make prompts more specific.

The process comprises various steps, including the assessment of the project's environment, business logic modeling, describing its UI/UX, planning the development cycle, validation of results, and producing a final package. In general, all the agents generate a complete set of outputs, which includes user stories, a complete Software Requirements Specification (SRS) document, non-functional requirements, APIs, traceability matrices, and quality reports. The final product is condensed into one downloadable package.

The architecture incorporates sophisticated features, which ensure the stability and efficiency of the operation, such as the ability to save the current state, playbacks, resume operations, debugging with rich logs, and manifest-driven data processing. We evaluated the performance of our solution through testing it on a hospital management software platform. The task included processing complex issues, such as patient registration, emergency operations, diagnosis, pharmacy, and billing procedures.

To summarize, the output demonstrates that artificial intelligence performs significantly better in acquiring software requirements in a structured and traceable workflow rather than being a simple text generator.

Dedication

This project is hereby dedicated to our parents who have shown infinite love and belief in us throughout their lives. They have been our constant companions despite whatever life brings, and without their support, we could never have accomplished anything that we have done before now.

Acknowledgments

Our most heartfelt gratitude goes out to our supervisor Dr. Awais Majeed, who through his guidance and criticism helped us develop AIMARES successfully. Guidance from our supervisor made us successful and contributed greatly to the quality of our efforts.

Our gratitude goes out to Bahria University and the Software Engineering Department for helping us by providing us with the required resources and an enabling environment to undertake this effort. Knowledge gained from education has served as a great resource for this purpose.

Friends and colleagues have provided us with constant support and discussion of issues related to this effort, for which we are grateful to all, especially to our friend Ramish.

Most importantly of all, We are thankful to our family members for their constant support and encouragement for this effort.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Any piece of software begins from a general idea, while the development requires a number of details. Prior to any design, any mock-up of the interface, and the writing of the first code line, the developer must understand the requirements, user roles, workflows, constraints, and the quality criteria of the product. However, even the initial gathering phase is not an effective process, as clients give too generalized descriptions of the project that contain many important elements, thus making the work for developers take lots of time. Therefore, developers need to analyze the project's description and produce the technical one

This problem can be solved by implementing a Multi-Agent AI System called AIMARES. The idea of this system lies in transforming a general description of the project into the technical documentation of the product. The technical documentation will include the user stories with the standard template structure, the documentation similar to that of SRS, NFRs, API output, traceability data, and quality documentation. Rather than utilizing a single large AI prompt to create everything, AIMARES employs a deterministic multi-agent workflow when individual agents carry out particular steps of the process.

1.1. Motivation

There is a universal issue that faces all teams working on any kind of development. Initially, you receive very general information regarding your future product. For instance, you might hear that "you need to create a Hospital Management System". From the outside, it may seem quite satisfying. However, it completely ignores a lot of aspects, including who needs this product, what data flow should take place inside it, what constrains apply, and finally, what should the output be like. Due to this lack of understanding of details, software engineers and business analysts waste numerous days creating comprehensive documentation based on tiny bits of information.

Another evident trend nowadays is the increasing role of artificial intelligence in all industries. Even though the former is capable of writing texts incredibly fast, they are frequently irrelevant, not credible, or even out of the topic in question. Problems

become particularly evident when dealing with something more complicated than plain writing. Creating a product in the sphere of healthcare, for example, means that it should be useful to a number of different specialists, such as patients, doctors, nurses, administrators, pharmacists, and accountants. Omitting even one aspect of functioning, like managing hospitalization or paying for certain services, will result in the entire project being doomed to fail.

Therefore, we chose to develop a tool that goes beyond a simple text generator. Firstly, it was required to become a well-established pipeline, performing particular operations precisely. Among them are defining an accurate initial prompt, identifying the industry in which the process takes place, retrieving groundings, composing multiple technical documents gradually, checking oneself and tracking what is currently happening. This is precisely the case of our project called AIMARES.

1.2. Objectives

The primary purpose of the present research is to develop a system which will enable to move from the basic concept of software engineering towards the comprehensive set of project requirements.

To achieve the stated purpose, it will be necessary to focus on achieving the following goals:

1. Creation of an intelligent and interactive system, which will enable extracting the basic information about the project and providing a full-fledged description thereof.

2. Development of a highly accurate, thoroughly designed multi-agent network which will enable the automated creation of all required technical documentation including user stories, standard SRS, non-functional specifications, APIs, and loggings.

3. Improvement of the efficiency of the developed AI documentation by means of applying relevant techniques for grounding and logic, quality assurance, progress saving, replay and fault tolerance mechanisms.

1.3. Main contributions

- Building up a complete requirements engineering system with the use of AIMARES but at the same time avoiding the trap of creating just one more primitive chatbot style prompt generator.
- Developing an ecosystem of different agents where every step in the workflow is accountable. Thus, there will be researching the industry, acquiring data, working out requirements and business requirements, formulating the description of the interface, coding architecture, testing for faults, and putting the whole process together.
- Creating a strictly deterministic pipeline. That will guarantee the fact that irrespective of how many times the specifications are generated, they will always be trackable, repeatable, and, if need be, recoverable in case of any mistakes.
- Giving the possibility to the software to produce tangible output documents. In seconds, it will create user stories, standard SRS documentation, API documentation, logs, QA documentation, and much other necessary material.
- Including powerful developer tools in the toolset directly. They will cover saving the current state, redoing previous steps, recovering after failures, tracking sources of the data, and using special debugging channels.
- Testing the whole system against the complex challenge of medical administration management. This case study is designed to show that the platform of requirements engineering works well in complex fields with multiple user personas and tasks.

This project becomes distinctive owing to the rigorous approach towards building a chain of steps in requirements engineering with the use of artificial intelligence. Thus, the final result will feature vastly improved specifications of the project?

1.4. Report organisation

The structure of this report is as follows:

Chapter 2 presents the background and conceptual basis of the project, including AI-assisted requirements engineering, multi-agent orchestration, retrieval-based grounding, deterministic workflow design, and validation concepts.

Chapter 3 focuses on the system requirements of AIMARES, including functional, interface, database, and non-functional requirements

Chapter 4 explains the system design, covering architecture, workflow, logical structure, and data flow design.

Chapter 5 presents the implementation details, including frontend, backend, orchestration, retrieval subsystem, and agent modules.

Chapter 6 discusses testing and evaluation, including benchmark-based validation and system behavior.

Chapter 7 concludes the report and highlights future work and possible improvements.

Chapter 2

Background Study/Literature Review

A literature review presents a survey of previous works related to AIMARES. Since AIMARES makes use of artificial intelligence to automate software specification generation, it becomes crucial to discuss the way this document usually looks like in the conventional software development practice. In addition, it is essential to analyze the position of new AI-powered solutions in the context of software development. This section will outline limitations of traditional approaches and discuss the identified problem that has driven the creation of our product.

Identification of the tasks which a particular software application is required to perform is called requirement engineering. It is an important phase in software development that takes place much earlier than designing, programming, and testing. In order to perform this task, developers require quite much effort, as well as sufficient knowledge about the program on the part of a client. Often, the initial proposition made by the client would be too ambiguous to understand or too sophisticated to implement. The task of gathering and processing the information about the system to be created can turn out to be particularly challenging in complicated domain areas with several user roles, overlapping responsibilities, and many regulations to comply with.

It is true that significant progress has been made in AI technologies lately. Many advances were achieved especially within the framework of LLMs. As a result, researchers conduct plenty of experiments aimed at the optimization of processes associated with preparation of software requirements specification. Recently, one such study was conducted by Binkhonain and Alfayez and its findings appeared in the paper titled "Are prompts all you need? Evaluating prompt-based Large Language Models (LLMs) for software requirements classification." The findings of this study laid the basis for the method used in AIMARES.

2.1. Key Concepts

2.1.1 Requirements Engineering

Requirements Engineering (RE) is a core activity in software development. It includes elicitation, analysis, prioritization, specification, and verification of requirements. Its purpose is to ensure that the final system reflects stakeholder needs in a clear and

usable form. The literature shows that good requirements improve communication, reduce development risks, and support traceability and quality assurance.

A common problem in RE is that stakeholders usually express their ideas informally. For example, a statement such as “build a hospital management system” does not define user roles, workflows, constraints, or validation conditions. This means developers and analysts must spend extra effort converting vague ideas into structured documents. Requirements classification and organization therefore become important, because they improve understanding, prioritization, and management of requirements throughout the lifecycle.

2.1.2 AI-Based Requirements Generation

With the widespread use of LLMs, professionals have started considering whether these applications can complete certain tasks like drafting, organizing, and detailing the needs of projects. First, it is necessary to admit that the use of LLMs undoubtedly speeds up the process of working out an initial draft. At the same time, research shows that the sole use of LLMs means omitting some data, having contradictions, or even hallucinations. These issues usually occur when the machine requires formatting and knowledge of the subject matter.

To sum up, artificial intelligence is definitely a helpful tool; however, it is not able to complete all the tasks. Prior to applying artificial intelligence to practice, one should validate the data obtained.

2.1.3 Multi-Agent Systems

Another key concept needed in this type of technology is multi-agent architecture. In this case, it refers to the strategy through which an extremely huge task can be divided into stages with each stage delegated to one of the bots. Through this arrangement, the whole process stays highly organized, ensuring that everything focuses on only one thing.

Some of the phases included in the AIMARES process include; requirements appraisal, business logic diagrams, generation of design interfaces, development plan formulation, checking of errors, and packaging. Such a format is what makes AIMARES able to generate highly readable documents from a highly managed system.

Research conducted in this industry reveals that having more than one bot leads to high modularity and flexibility. However, when the processes performed by such bots are not structured, the result of their activities may prove to be unpredictable. It is for this reason that AIMARES uses a highly structured approach in addition to using bots.

2.1.4 Retrieval-Augmented Generation (RAG)

The facts related to the particular domain will be fed into the AI system via Retrieval Augmented Generation as soon as the process of output generation starts. In addition to just knowing all aspects of the particular topic, it is crucial to actively seek the information necessary for the ultimate outcome.

The use of this method proves particularly helpful in the case of our AIMARES project, primarily owing to the specificity of its orientation towards very narrow domains, such as hospital management. The lack of information external to the AI system would make the specifications of the project seem too vague.

2.2. Discussion of Existing Work

2.2.1 Prompt-Based LLMs for Requirements Classification

The most important paper used for this assignment is "Are prompts all you need? Evaluating prompt-based Large Language Models (LLMs) for software requirements classification" authored by Manal Binkhonain and Reem Alfayez. In this article, the authors investigate whether it is possible to reduce dependency on labeled datasets through the use of prompt-based LLMs regarding the problem of requirements classification.

Specifically, the authors consider five LLMs utilizing four kinds of prompts – zero shot, few-shot, persona-based prompts, and chain-of-thought prompting. These methods were assessed within three requirements classification tasks. The major f inding of the authors is that using prompt-based LLMs and, particularly, a few-shot approach is enough to achieve comparable results to transformer-based LLMs, such as BERT. Furthermore, in certain cases, better results could be obtained by applying a persona or persona and chain-of-thought prompt.

This paper is related to the aim of AIMARES since it confirms that using prompt based LLMs can perform a reasonable task of requirements analysis at minimal 22 training cost. Still, it is more focused on classification and not on generation of requirements packages.

2.2.2 LLM-Based Agents for Requirements Specification

One more key paper we considered in our work was "A design science research approach to Large Language Model-Based Agents for Requirements Specification (LLMBA4RS) in low-code applications" by Cristian Rotar and Qingyu Zhang. It is important for us since the authors went further than just making an attempt to communicate with AI with a standard prompt. In other words, the researchers created an agent-based system with the help of RAG and CrewAI technologies..

It is noteworthy that the authors focused on one of the most essential challenges for agile and low-code software engineering. Namely, there is an extremely high degree of dependence on exact project specifications, which create problems for developers since they are always ambiguous, incomplete, and inconsistent. The authors offered a solution by designing a set of specialized bots that would generate full-fledged user stories and make connections between different tasks. In other words, as it is explained by the

authors, the design of the proposed solution includes three major stages: interpretation of user inputs, bot management, and acquisition of additional information by means of the RAG layer. As shown in the picture included in the paper, each of these processes is independent of others.

2.3. Research Gap

Previous research highlights important advancements; however, there are certain drawbacks.

Firstly, previous machine learning models relied on vast amounts of manually tagged data, making it expensive, time-consuming, and hard to scale across industries. The use of Large Language Models with prompt support completely obviates this need, leading to an important milestone achieved by engineers.

Secondly, while prompt-driven classification has its merits, it generally deals with specific tasks like classifying functional and non-functional requirements. None of these approaches actually tackle the problem of turning a plain proposal into technical documents

Also, systems that incorporate several agents, such as LLMBA4RS, are definitely much closer to real-world processes in industries. Yet, even such approaches continue to face challenges such as the repetition of the text, lack of learning from the users' responses, and wrong associations with the narrative of the users.

Finally, most studies focus on low-code systems. AIMARES is a complex system that does not fit into such systems at all. Given the fact that the system aims at developing various kinds of documentation related to user stories, quality assurance, API description, interfaces, and a project as a whole, it seems reasonable to create the process of its performance together with traceability.

2.4. Chapter Summary

The basic principles of software requirements acquisition using AI have been described in this chapter. The significance of effective project management, the importance of prompt-based language model, and the advantages of having intelligent agents network have all been described in this chapter. The necessity of using true information, a consistent process flow, error testing, and documentation of every change made has been described at length. Besides, the shortcomings of current practices, which led to the necessity of developing a framework like AIMARES. The issues discussed above will set the stage for the technical aspects of AIMARES framework.

Chapter 3

System Requirements

This is a set of technical specifications for AIMARES, considering its practical implementation. Due to the fact that the software is a Multi-Agent AI System for Automated Software Development, it cannot be directly compared to any ordinary text generator in regard to the scope of functions performed. The software is supposed to get the structure of the project, analyze the surrounding environment and gather necessary data, and then be thoroughly processed with the help of a complex network of specialized agents, and the results will be checked and presented in the form of engineering output.

Accordingly, considering the complexity of these procedures, the main specifications for the platform may be separated into distinct categories.

3.1. System Level Use Case Diagram

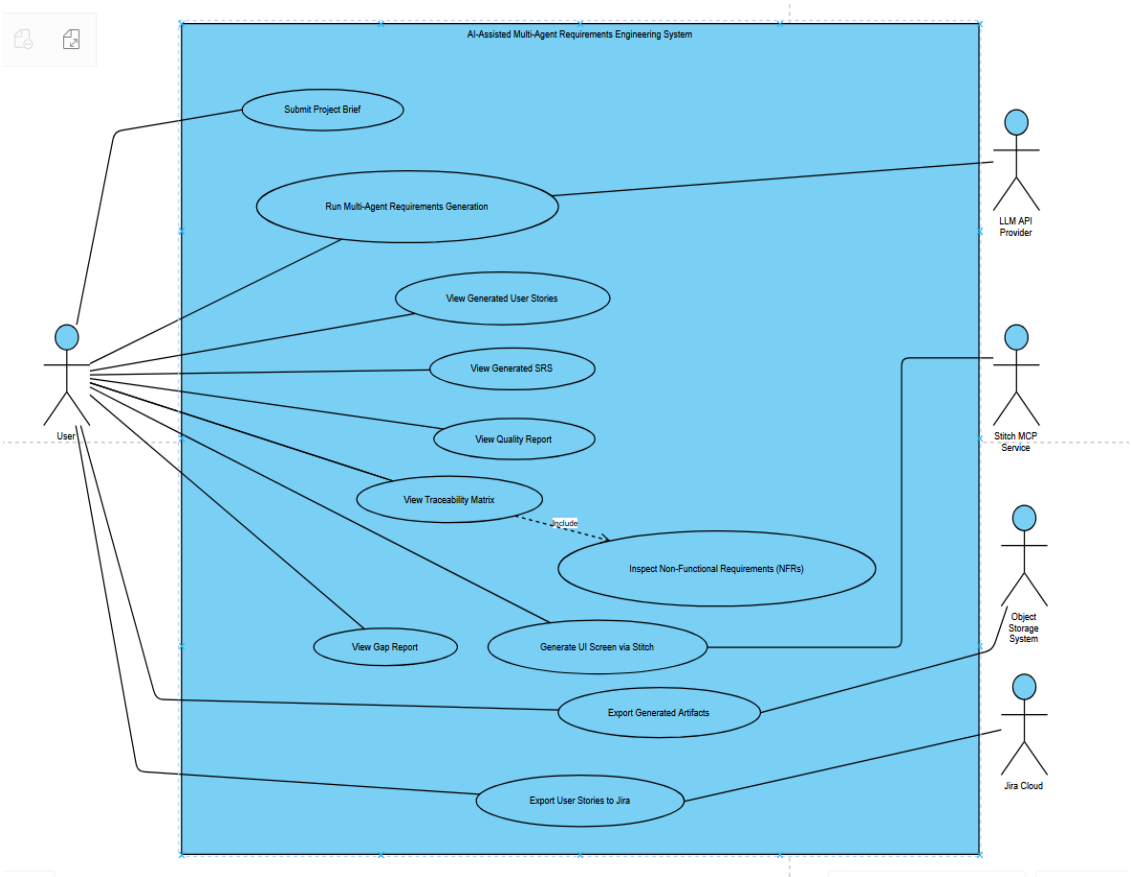


Figure 1 Use Case Diagram

3.2. Functional Requirements presented as Use Case Descriptions

3.2.1 FR-01: Brief Intake

Use Case ID:	UC-01	
Use Case Name:	Submit Project Brief	
Actor(s):	Student/User	
Pre-Conditions:	User is authenticated and on the Brief Submission screen.	
Priority:	High	
Basic Flow:	User submits the Brief/input prompt	
Actor Actions		**System Response**
1.	User enters project title, domain hint, problem statement, and constraints.	2. System performs client-side validation (required fields, max length, etc.).
3.	User clicks the Save / Submit Brief button.	4. System creates or updates a Session record and stores the brief in the database.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	User leaves required fields empty or enters invalid data.	2.a System highlights invalid fields and displays inline error messages (e.g., "Title is required").
1.b	Network/API error occurs while saving the brief.	2.b System displays an error banner and does not mark the brief as saved; the user can retry submission.

Table 1 Brief Intake Use Case

3.2.2 FR-02: Multi-Agent Workflow Execution

Use Case ID:	**UC-02**	
Use Case Name:	Run Multi-Agent Requirements Generation	
Actor(s):	Student/User	
Pre-Conditions:	User is authenticated and on the Brief Submission screen.	
Priority:	Very High	
Basic Flow:	system runs the multi-agent workflow to generate requirements and artifacts.	
Actor Actions		**System Response**
1	User clicks “Run Session.”	2. System opens an SSE connection for live trace streaming.
3.		4. System executes the multi-agent workflow
4.		6. RAG subsystem retrieves relevant domain knowledge via FAISS top-k search.
5.		7. Requirements are validated (INVEST/ISO 29148).
6.		8. System generates SRS, User Stories Booklet, Traceability Matrix, Quality Report, and Gap Report.
7.		9. System marks session as COMPLETED and shows the Results screen.
Alternative Course of Action (if any)		
Actor Action		**System Response**

1.a	User submits incomplete brief.	2.a System displays validation errors and blocks session creation.
1.b	LLM or embedding API fails.	2.b System displays error state, marks session as FAILED, and stops execution.
1.c	Workflow exceeds max turns.	2.c System returns partial outputs and ends the session.

Table 2 Run Multi-Agent Requirements Generation Use Case

3.2.3 FR-03: View Generated User Stories

Use Case ID:	UC-03	
Use Case Name:	View Generated User Stories	
Actor(s):	Student/User	
Priority:	High	
Basic Flow:	User views the Generated User Stories	
Actor Actions		**System Response**
1.	User clicks “View User Stories”.	2. System retrieves Stories data from storage and displays it in a readable format.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	Stories data is unavailable	2.a System displays: “Stories data not found. Please run the workflow first.”

Table 3 View Generated User Stories Use Case

3.2.4 FR-04: View Generated SRS

Use Case ID:	UC-04	
Use Case Name:	View Generated SRS	
Actor(s):	Student/User	
Priority:	High	
Basic Flow:	User views the Generated User Stories	
Actor Actions		**System Response**
1.	User clicks “View Generated SRS”.	2. System retrieves SRS data from storage and displays it in a readable format.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	SRS data is unavailable	2.a System displays: “SRS data not found. Please run the workflow first.”

Table 4 View Generated SRS Use Case

3.2.5 FR-05: View Quality Report

Use Case ID:	UC-05	
Use Case Name:	View Quality Report	
Actor(s):	Student/User	
Priority:	High	
Basic Flow:	User views the Generated User Stories	
Actor Actions		**System Response**
1.	User clicks “View Quality Report”.	2. System retrieves Quality Report data from storage and displays it in a readable format.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	Quality Report data is unavailable	2.a System displays: “Quality Report data not found. Please run the workflow first.”

Table 5 View Quality Report

3.2.6 FR-06 View Traceability Matrix

Use Case ID:	UC-06	
Use Case Name:	View Traceability Matrix	
Actor(s):	Student/User	
Pre-Conditions:	Run Multi-Agent Requirement Generation must have completed.	
Priority:	High	
Basic Flow:	User views the traceability mapping of stories to requirements and artifacts	
Actor Actions		**System Response**
1.	User clicks "View Traceability Matrix".	2. System retrieves traceability data from storage and displays it in a readable format.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	Traceability data is unavailable	2.a System displays: "Traceability data not found. Please run the workflow first."

Table 6 View Traceability Matrix Use Case

3.2.7 FR-07: Generate UI Screen Via Stich

Use Case ID:	UC-07	
Use Case Name:	Generate UI Screen Via Stich	
Actor(s):	Student/User	
Pre-Conditions:	Run Multi-Agent Requirement Generation must have completed.	
Priority:	High	
Basic Flow:	User triggers screen generation and the system returns a generated UI preview	
Actor Actions		**System Response**
1.	User clicks Generate UI Screen via Stitch	2.System sends generation request to Stitch endpoint and returns a screen resource reference with in-progress status.
3.	User waits on the preview panel	4. System polls Stitch status and updates progress until complete.
5.	User opens generated preview	6.System fetches proxy image/HTML and renders the generated UI output
Alternative Course of Action (if any)		
Actor Action		**System Response**
1. a	Stitch API key is missing or invalid	2.a System shows: Stitch integration is not configured. Please contact administrator.

Table 7 Generate UI Screen Via Stich Use Case

3.2.8 FR-08: View Gap Report

Use Case ID:	UC-08	
Use Case Name:	View Gap Report	
Actor(s):	Student/User	
Pre-Conditions:	Run Multi-Agent Requirement Generation must have completed.	
Priority:	High	
Basic Flow:	User views the Gap report	
Actor Actions		**System Response**
1.	User clicks “View Gap Report”.	2. System retrieves View Gap Report data from storage and displays it in a readable format.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	Gap Report data is unavailable	2.a System displays: “Gap Report data not found. Please run the workflow first.”

Table 8 View Gap Report Use Case

3.2.9 FR-09: Export Generated Artifacts

Use Case ID:	UC-09	
Use Case Name:	Export Generated Artifacts	
Actor(s):	Student/User	
Pre-Conditions:	Session must be successfully completed.	
Priority:	High	
Basic Flow:	User Exports generated artifacts after session completion	
Actor Actions		**System Response**
1.	1. User navigates to the **Results** screen.	2. System displays all downloadable artifacts.
3.	2. User clicks on a file link (Stories, Matrix, etc.).	3. System initiates file download.
3.	3. User clicks **"**Download ZIP Bundle**."**	4. System downloads the full artifact bundle with manifest.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1. a	1.a User tries to download before session completion.	2.a System displays: "Artifacts not available until session completes."

Table 9 Export Generated Artifacts Use Case

3.2.10 FR-10: Export User Stories to Jira

Use Case ID:	UC-10	
Use Case Name:	Export User Stories to Jira	
Actor(s):	Student/User	
Pre-Conditions:	Run Multi-Agent Requirements Generation should be completed. User stories must exist for the selected run. Jira integration should be configured	
Priority:	High	
Basic Flow:	User exports generated stories to Jira	
Actor Actions		**System Response**
1.	User clicks Export User Stories to Jira	2. System optionally runs dry-run preview and shows mapped stories/issues
3.	User confirms export	4. System sends export request and creates/updates Jira issues.
3.	User checks sync results	4. System returns created/updated/failed counts and issue-key mappings.
Alternative Course of Action (if any)		
Actor Action		**System Response**
1.a	Jira authentication/configuration is invalid.	2.a System shows: Jira connection failed. Verify integration settings.

Table 10 Export User Stories to Jira Use Case

3.3. Interface Requirements

3.3.1 User Interface Requirements

ID	Requirement
UIR-01	The system shall provide a web-based user interface accessible from standard modern browsers on desktop and laptop devices.
UIR-02	The system shall provide an intake interface for entering project brief details including title, scope, domain context, roles, capabilities, and quality constraints.
UIR-03	The intake interface shall validate mandatory fields before submission and display actionable validation messages.
UIR-04	The UI shall allow users to start a generation run from the intake flow.
UIR-05	The UI shall display live run progress including node status transitions pending, running, completed, and failed.
UIR-06	The UI shall display generated outputs including user stories, SRS-oriented content, quality findings, and traceability-related views.
UIR-07	The UI shall provide Stitch screen generation controls with status polling and preview rendering support.

UIR-08	The UI shall allow artifact export and bundle download actions after successful run completion.

Table 11 User Interface Requirements

3.3.2 Software Interface Requirements

ID	Requirement
SIR-01	The frontend shall communicate with backend APIs using HTTP-based JSON interfaces.
SIR-02	The run start interface shall support prompt submission through the MAF start endpoint.
SIR-03	The system shall expose run metadata interfaces for status, artifact references, and trace retrieval.
SIR-04	The system shall expose server-sent event interfaces for live run trace streaming.
SIR-05	The system shall expose artifact download interfaces for run-level and session-level artifact retrieval.
SIR-06	The Stitch integration interface shall support screen generation, status polling, and proxy rendering endpoints.

Table 12 Software Interface Requirements

3.3.3 Hardware Interface Requirements

ID	Requirement
HIR-01	The client interface shall run on commodity end-user hardware without specialized devices.
HIR-02	The backend shall require network access to database and object storage services.
HIR-03	The backend shall require network access to configured external providers for LLM, Jira, and Stitch features.

Table 13 Hardware Interface Requirements

3.4. Database Requirements

3.4.1 Database Platform Requirements

ID	Requirement
DBR-01	The system shall use a PostgreSQL-compatible relational database as the primary transactional store.
DBR-02	Database configuration shall be environment-driven to support local, staging, and production deployments.
DBR-03	The database layer shall be accessed through managed connection pooling from the API service.
DBR-04	All timestamps shall be stored in UTC with timezone-aware columns.
DBR-05	The database shall support JSON/JSONB fields for semi-structured pipeline payloads.

Table 14 Database Platform Requirements

3.4.2 Core Schema Requirements

ID	Requirement
DCSR-01	The system shall persist session metadata in a sessions table.
DCSR-02	The system shall persist artifact metadata in an artifacts table (including storage key, type, and checksum fields where available).
DCSR-03	The system shall persist normalized story outputs in a stories table.
DCSR-04	The system shall persist normalized inferred APIs in an apis table.
DCSR-05	The system shall persist normalized inferred APIs in an apis table.

Table 15 Core Schema Requirements

3.4.3 Query and Indexing Requirements

ID	Requirement
DCSR-06	The database shall index run retrieval by session and creation time to support run history views.
DCSR-07	The database shall index trace lookup by run_id and idx for ordered replay retrieval.
DCSR-08	The database shall index trace lookup by session and timestamp for session-scoped diagnostics.
DCSR-09	The database shall index artifacts by run_id, session_id, and kind for fast artifact listing and download preparation.
DCSR-10	The database shall index normalized output tables by run_id/session_id for report and export operations.

Table 16 Query and Indexing Requirements

3.4.4 Relationship and Integrity Requirements

ID	Requirement
DCSR-11	Each run shall reference a valid session via foreign key constraints.
DCSR-12	Each trace event shall reference valid run and session records via foreign keys.
DCSR-13	NFR records shall enforce uniqueness per run using run_id and nfr_id.
DCSR-14	API records shall enforce uniqueness per run and endpoint signature using run_id, method, and path.

Table 17 Relationship and Integrity Requirements

3.5. Non-Functional Requirements

ID	Category	Requirement
NFR-01	Determinism	System shall produce consistent outputs for the same input and configuration
NFR-02	Performance	System shall execute workflows within acceptable time limits
NFR-03	Maintainability	System shall be modular and easy to extend
NFR-04	Usability	System shall provide an intuitive and user-friendly interface
NFR-05	Scalability	System shall support multiple runs and increasing workload
NFR-06	Traceability	System shall track all outputs and their generation process
NFR-07	Security	System shall protect API keys and sensitive data

Table 18 Non-Functional Requirements

3.6. Project Feasibility

First, let's talk about the practical use of AIMARES. For this purpose, AIMARES will be assessed through four different approaches – technological approach, usability, affordability, and ethical considerations. This means that, instead of relying on the theory alone, the conclusion has been made after analyzing the actual product.

3.6.1 Technical Feasibility

The AIMARES could certainly be designed from a technological standpoint. After all, it has been successfully built using cutting-edge software technology tools which can serve us in both educational and practical industry scenarios.

As far as the frontend side is concerned, we leveraged the capabilities of Next.js, React, and TypeScript. As an extra feature, we employed Tailwind CSS as our language of styling, together with Framer Motion and Zustand. For our backend development, we utilized Python, along with FastAPI which powers the management of APIs, workflow, and business logic.

In order to perform tasks execution, a well-controlled workflow engine referred to as Multi-Agent Flow (MAF) is adopted by the platform. This is attributed to the fact that the engine plays a crucial role in facilitating the adoption of a very structured approach in performing tasks. As for the data storage component, PostgreSQL is employed as the primary relational database. In addition, Supabase database connectivity services and file storage for projects are integrated in the platform, together with FAISS-based RAG, where embeddings from sentence transformer embeddings are used for domain-specific data.

The main principle behind the development of AI revolves around the Gemini framework. Conversely, the parameters within the system provide the flexibility of employing other OpenAI models effortlessly. Also, we integrated some additional tools with the backend of the system. Out of these, Jira helps in identifying bugs within the project while Stitch generates mockups for the user interface.

Overall, the system proves to be technically feasible since all of its key elements work properly and integrate successfully. Nevertheless, several issues should be taken into account. In terms of the first issue, relying on the external services provided by the LLM, Jira, and Stitch may cause the slowdowns or even breakdowns in case of their downtime. From another perspective, the app stores the status of the Stitch job in the memory buffer only, implying that any server restart would reset this information. In addition, taking into account the possible discrepancy between AI responses, schema validation should be performed extensively. Finally, because the current infrastructure design assumes the operation in a single-node mode, certain scalability improvements should be introduced for handling massive volumes of traffic.

3.6.2 Operational Feasibility

The AIMARES is very practical and useful for real-life situations and can easily be implemented in university labs and startup companies.

AIMARES is designed to cater to only selected few individuals such as:

- Senior university students developing software
- Businessmen designing business logic and architectures
- Frontline programmers
- Developers

The biggest challenge addressed by the proposed platform is the challenge associated with translating the vague ideas into technical documents. The proposed platform addresses this challenge by offering automatic creation and tracking of essential deliverables in various project tasks. Some of the deliverables include user stories, software requirement specifications (SRS), APIs documentation, and quality checking reports.

The ease of use that this instrument provides can be attributed to the following features:

- Interactive intake wizard to organize your initial data in an appropriate manner.
- Real -time visualization of the entire chain of the process while running.
- Friendly user dashboards including your project results3.6.3 Legal & Ethical Feasibility

This was kept in mind during the development of AIMARES, where the project was developed by taking all the legal and ethical aspects into account relating to AI-based systems.

While it is true that currently the system makes use of only standardized test data along with artificial data and thus there is no actual live data being fed into it, once it becomes operational, the data used to feed into it can well be confidential information belonging to some corporate/personal entity.

Given the possibility of this risk, it becomes necessary for us to consider handling some important aspects:

- Maintaining the privacy of individual information during interactions between our application and external AI engines.
- Avoiding information leaks during API calls.
- Proper handling of sector-specific information, especially within medical applications.

While considering strict environments like health care domain, we will require extra security measures to be considered, including the following:

- Hiding of personal identity within the database
- Tight access control based on user roles.
- Regular audit log
- Encrypted file during storage and transmission through the network.

From an ethical point of view, it is important to remember that machine-driven products can be faultless; however, it can have invisible hallucinations or fallacies. To avoid such scenarios, the design is such that human involvement in the loop is ensured, and therefore the output will always get the human approval.

3.7. Analysis Models

3.7.1 Sequence Diagram Multi Agent Generation

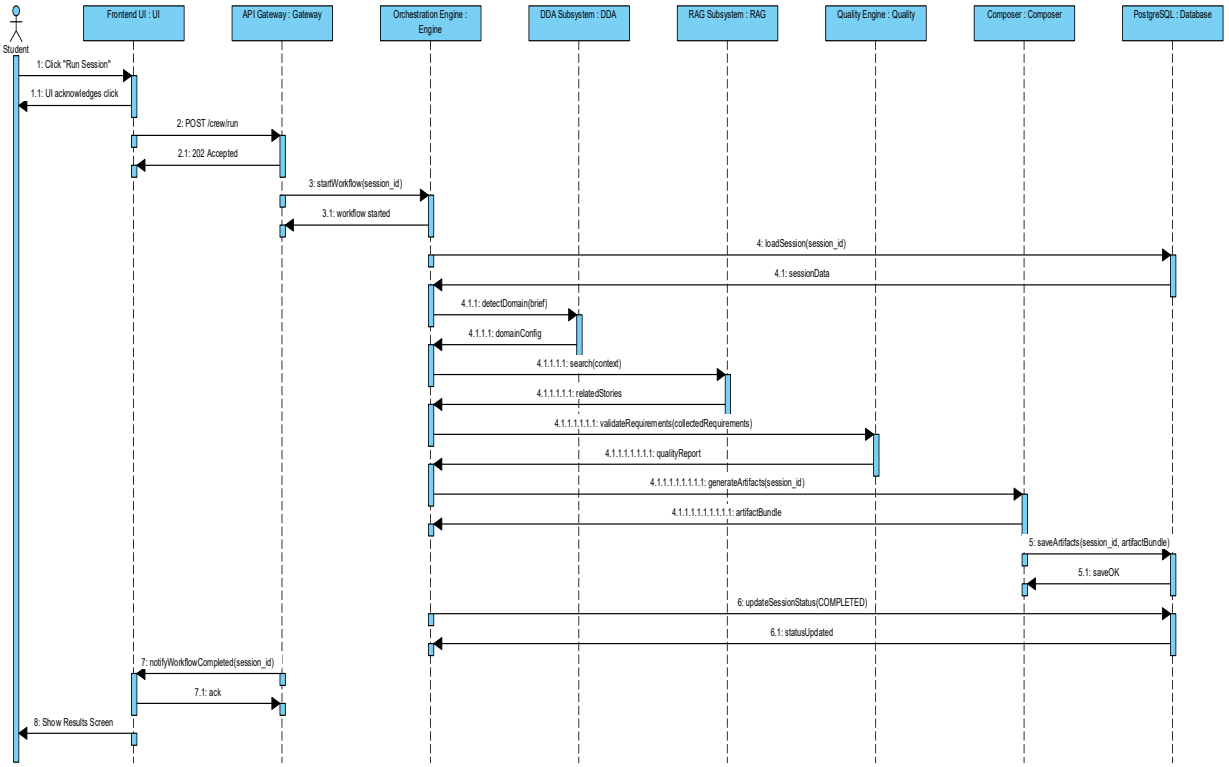


Figure 2 Sequence Diagram Multi Agent Generation

3.7.2 Sequence Diagram Stich Screen Generation

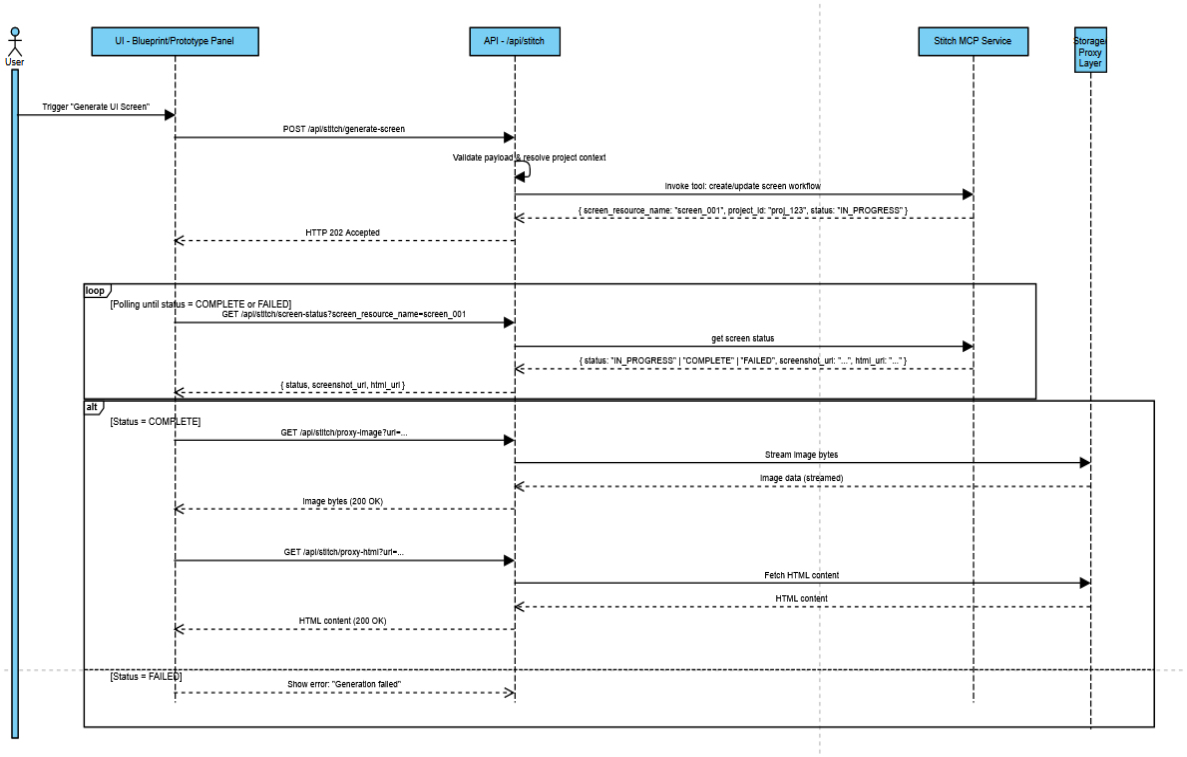


Figure 3 Sequence Diagram Stich Screen Generation

3.7.3 Jira Export Sequence Diagram

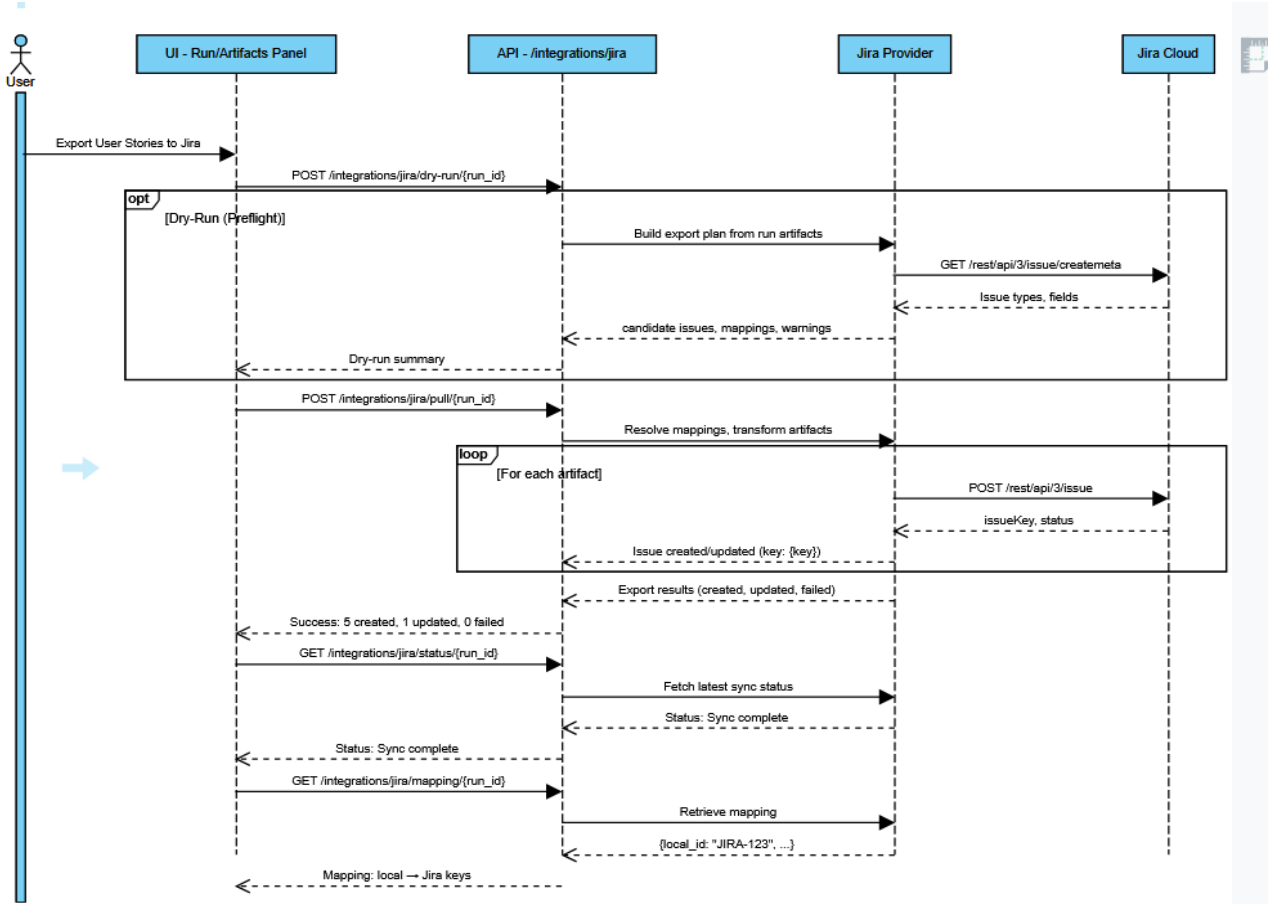


Figure 4 Jira Export Sequence Diagram

3.7.4 End to End Activity Diagram

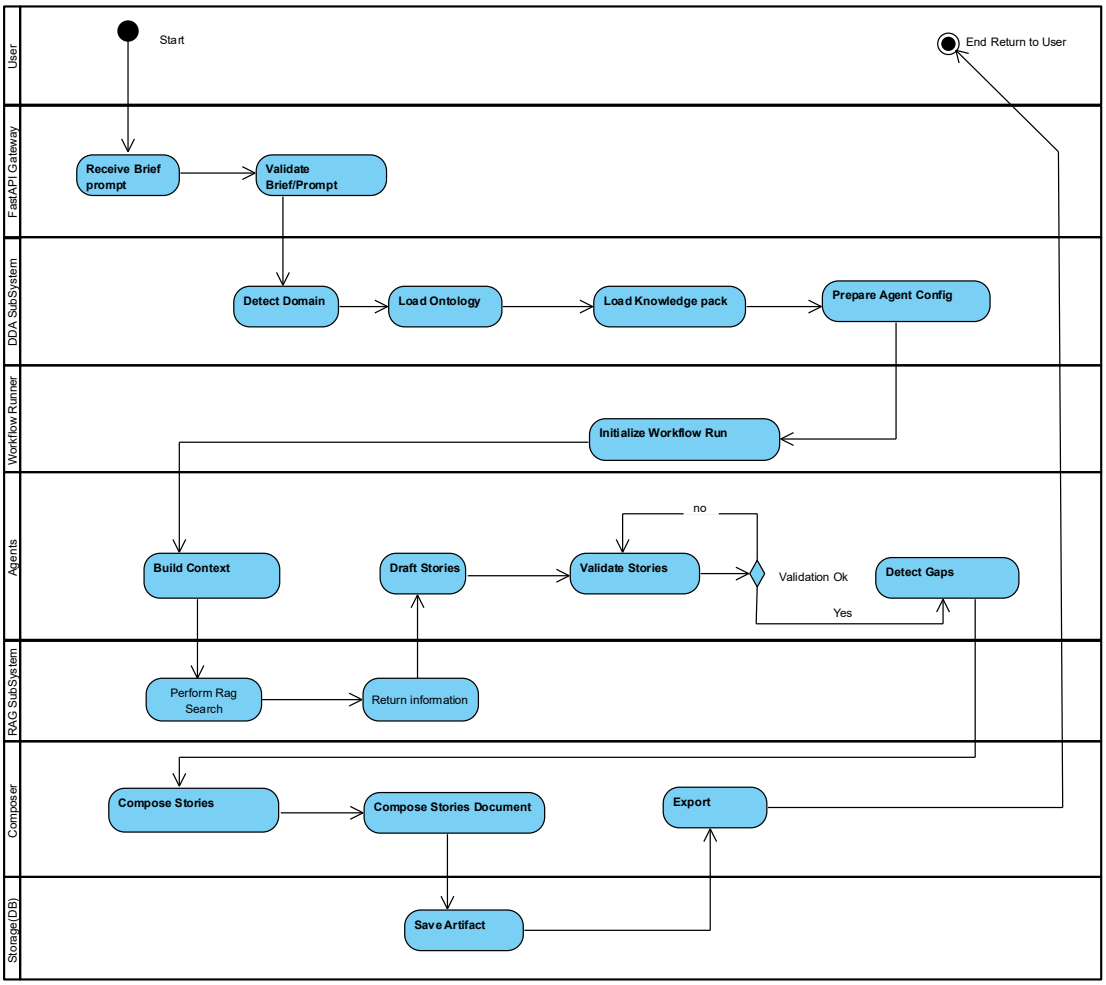


Figure 5 End to End Activity Diagram

3.7.5 Activity Diagram Stich UI Generation

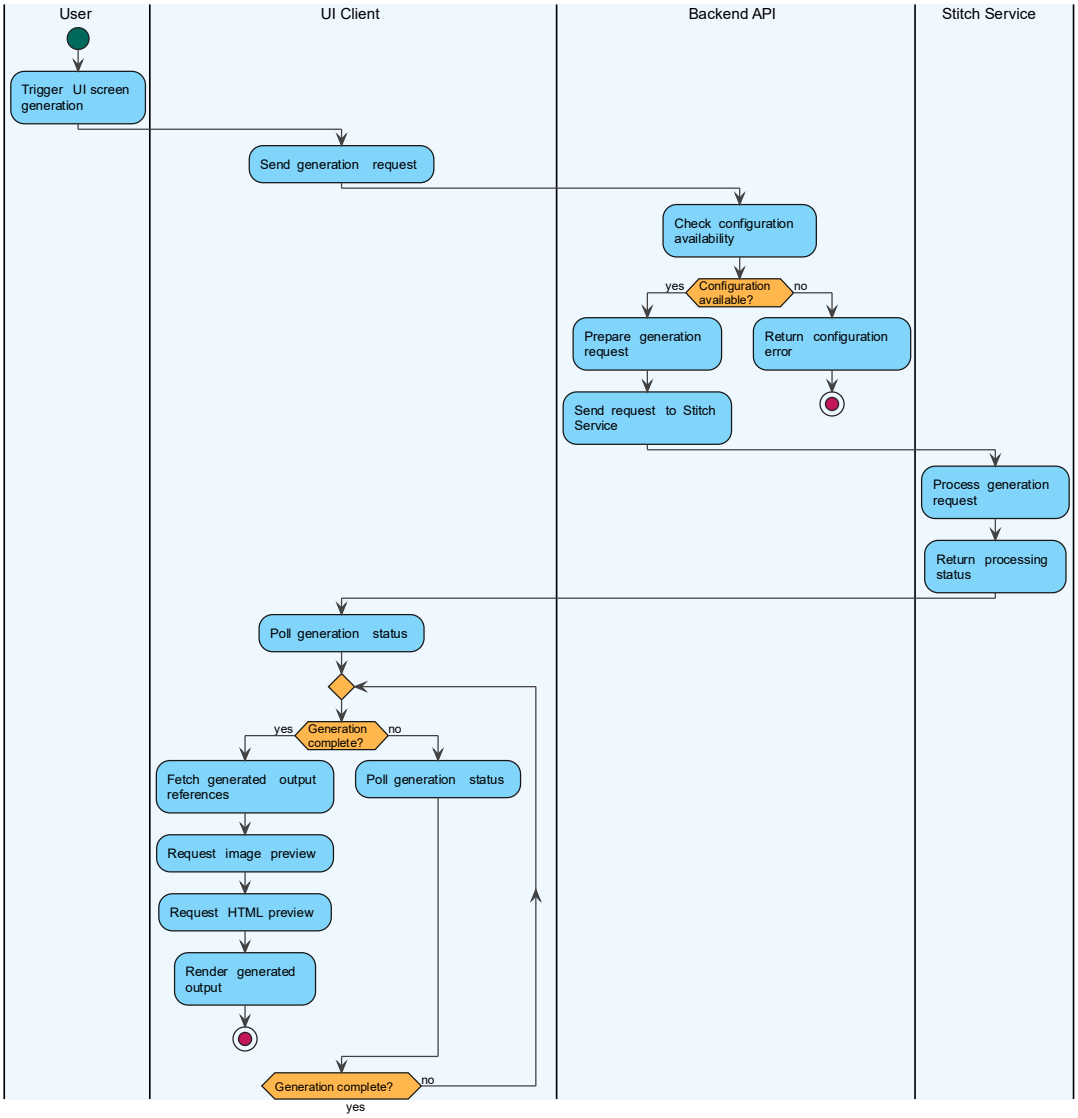


Figure 6 Activity Diagram Stich UI Generation

3.7.6 Activity Diagram Jira Export

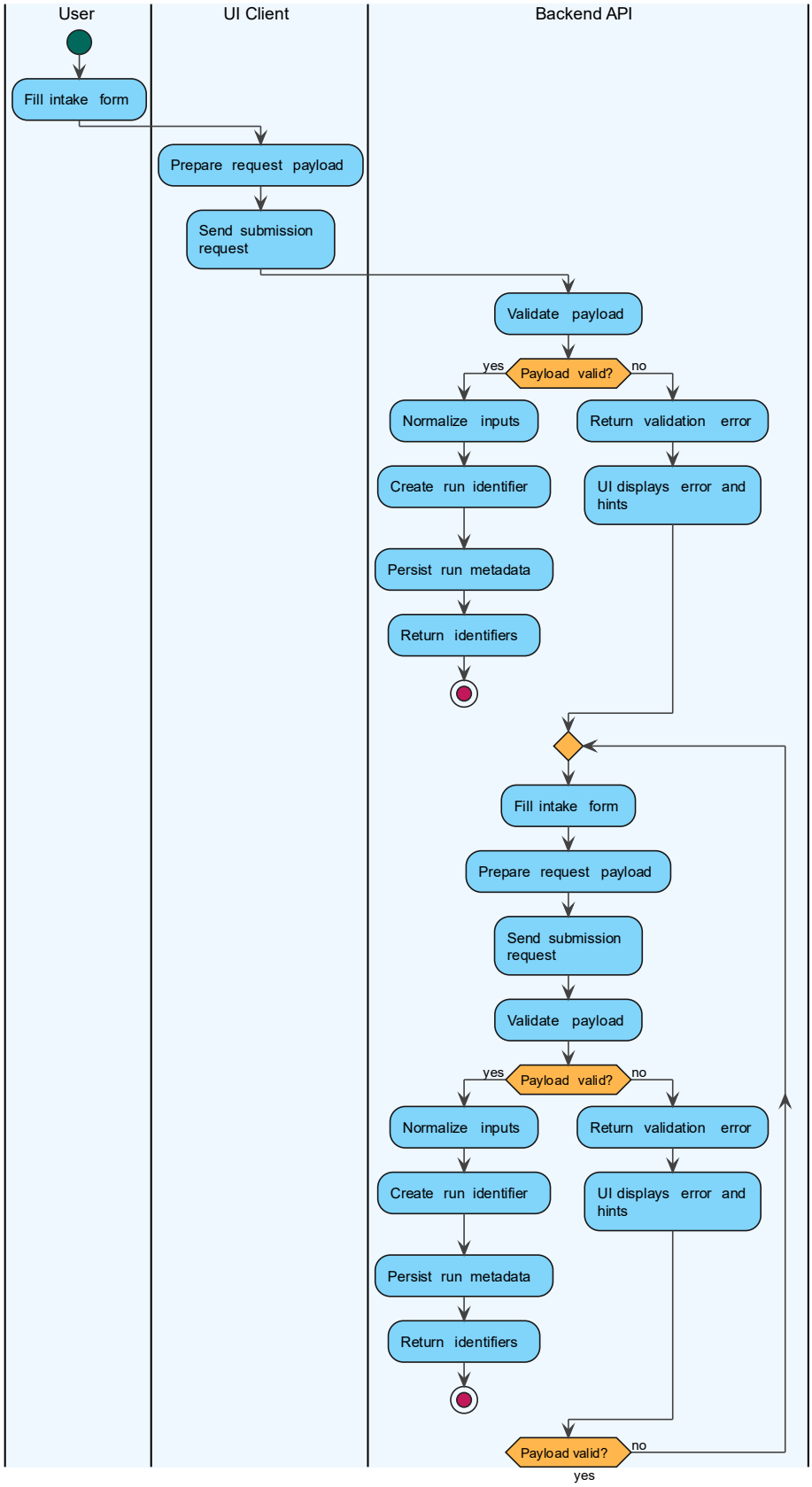


Figure 7 Activity Diagram Jira Export

3.7.7 Data Flow Diagram Level 0

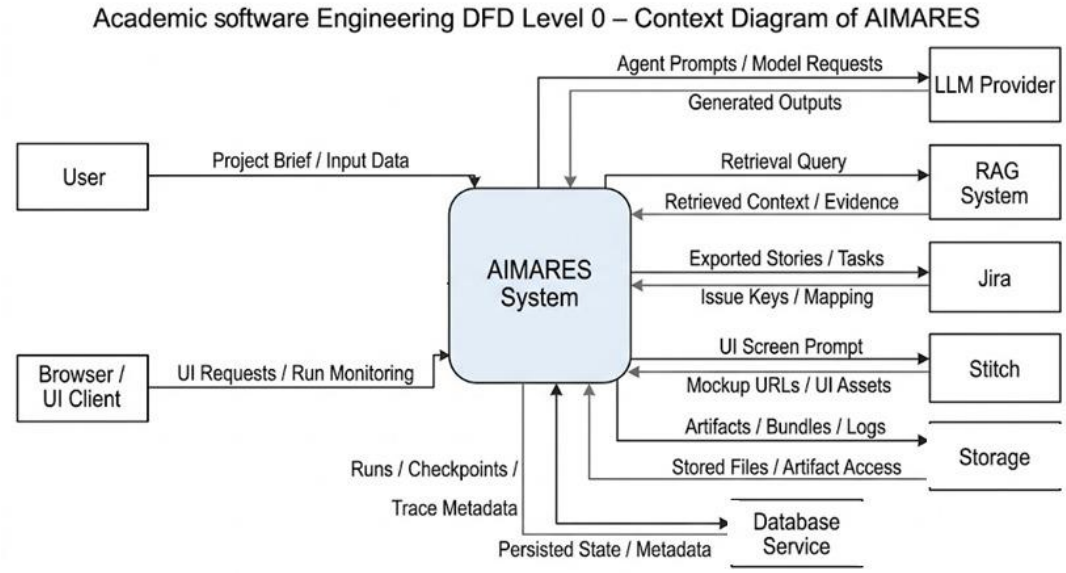


Figure 8 Data Flow Diagram Level 0

3.7.8 Data Flow Diagram Level 1

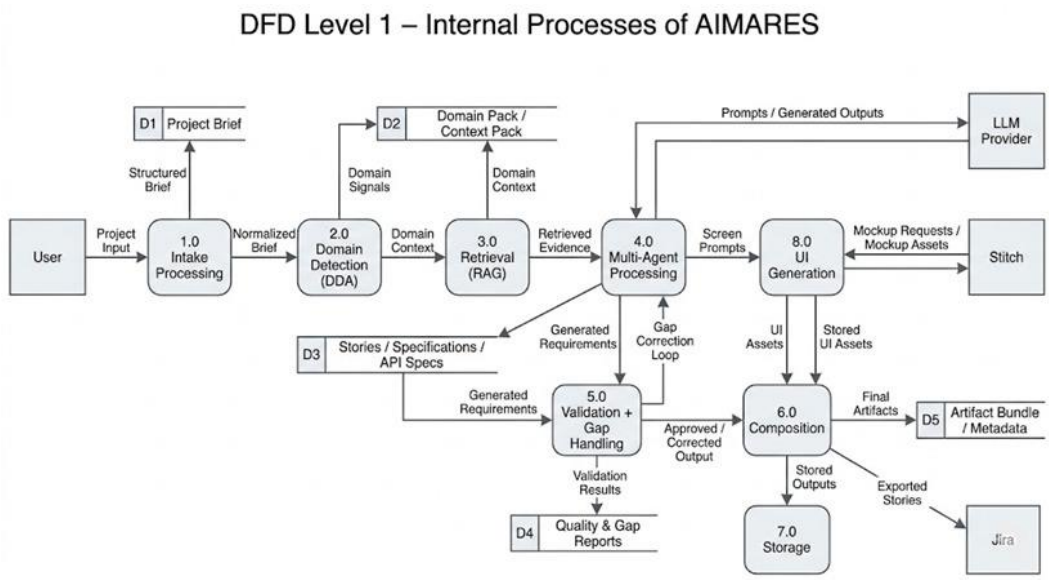


Figure 9 Data Flow Diagram Level 1

3.7.9 Data Flow

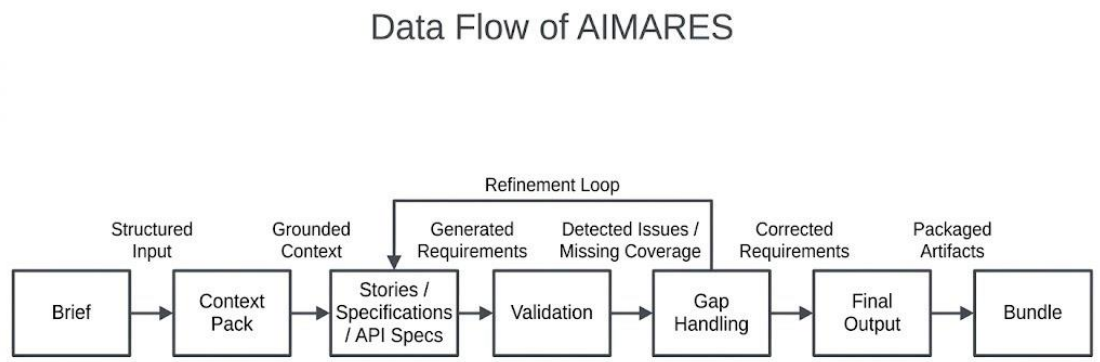


Figure 10 Data Flow

3.8. Conclusion

In this part, the basic requirements of AIMARES are introduced. Specific use cases are presented alongside the requirements both functional and non-functional, interface design, database design, feasibility study, and analytical model. Additionally, we provided an overview of the whole process from start to the very end when the desired output is generated by the system. To be more specific, we pointed out how AIMARES takes in the initial business ideas, generates specifications, verifies their correctness, and finally generates engineering artifacts.

Additionally, we showed that AIMARES is organized into a well-designed system, which involves extensive logging, automated error detection and correction, and interaction with third-party applications such as Jira and Stitch. These basic requirements define the capabilities of our software, and thus provide the ground for designing the software architecture, which will be discussed in the next section.

Chapter 4
System Design

Structure and communication within each module of AIMARES, which is used to generate software requirements automatically, will be considered thoroughly here. Rather than relying on a single software program that does everything, the proposed platform consists of an AI-based network made up of many agents. In this way, an army of highly efficient robots works together through an intelligent pipeline process.

4.1. Design Approach

AIMARES' architecture has been implemented based on a deterministic pipeline with the help of numerous bots. Rather than using one giant prompt generated by an AI, the software splits up the whole process of acquiring requirements into a few stages along a predetermined path.

The method used for accomplishing this feat is known as multi-bot orchestration. In this technique, each bot is assigned a specific function, like creating context, writing a user story, designing the interface, planning the development, detecting errors, and bridging gaps, ensuring that the code remains well-organized and allowing for easy updates in the future.

Deterministic behavior is at the core of the design of this software. The software guarantees deterministic behavior with the help of the following method:

- Deterministic flow and node graph.
- Deterministic AI configuration, including temperature and seed.
- Deterministic configurations with full data logging.
- Save points for pausing, rewinding, and resuming the process.

The development process consisted of rapid iterations and sprints to implement a modular design. One of the major focuses in our work was constant testing and regression testing to make sure that implementation of new features would not ruin existing features.

When talking about the overall architecture of the product, it became evident to us that one large language model should be avoided. It makes control and prediction of the system unmanageable. Using agents with specific skills ensures that every step is productive and generates organized data, which can be easily verified by the system and optimized.

Finally, we implemented RAG into the architecture of the product to make sure that all the generated information was based on industrial facts. This way, the number of fake information sources becomes minimal.

4.2. Design Constraints

The design of AIMARES is influenced by several technical, system level, and project related constraints.

4.2.1 Technical Constraints

While building our technology, we use plenty of API links that provide us with access to our language model and the tools for creating the interface. As always, working with API links brings certain difficulties to the process:

- Restrictive usage and restrictive requests
- Inconsistency in network latency and response time
- Long execution times of some operations, such as creating mockups of the interface, that could potentially take a few minutes

Besides all these difficulties, working with vectors and embeddings is quite computationally heavy. This becomes especially noticeable when our application operates exclusively on the CPU.

4.2.2 System Constraints

Currently, the system architecture is using the single node architecture design. This means that the entire task execution process would be performed through one API service. The use of distributed queues and workers is not implemented in the current architecture design; therefore, the architecture design cannot support very high loads of parallel tasks.

Furthermore, some parts of the system architecture keep track of their own states and store the state information in the system's memory. Should anything happen to the API service, such as crashing, all the state information will be lost. As a consequence, all background services will live for the same lifetime period as the main API service.

4.2.3 Data Constraint

The efficiency of the RAG system relies heavily on the data that is stored in the index. If the amount of data available within each dataset is not enough or the depth of information regarding the certain industry is inadequate, then the system will fail to provide accurate responses based on reliable facts. Also, the selection of industries that should be considered when running the system can be controlled by the settings on our platform. Using the platform without support for a certain industry will result in low accuracy.

4.3. System Architecture

Primarily, the system operates on a layered platform. This design works well in linking the user interface to the process of workflow on the server, while simultaneously integrating components like data collection, networking within artificial intelligence, thorough error checking, and database management.

4.3.1 High-Level Architecture (L0)

Overall, the system is built up of the following basic elements:

- Frontend (Next.js): Responsible for controlling the user interface and gathering all necessary information about the project.
- Backend (FastAPI): Ensures all API connections and manages all workflows
- Workflow Engine (MAF): Manages the structured multi-agent workflow
- RAG Subsystem: Gathers specific industry information that can help in creating context for AI algorithms.
- LLM Provider: Creates all documents and outputs.
- Storage System: Stores all information related to the execution process and generated documents.
- Integrations: Supports external services; for example, Jira for exporting tasks and Stitch for managing interface prototyping.

Besides those basic elements, the system architecture comprises:

- Observability and trace element: Helps collect all necessary information related to system behavior and background executions.

4.3.2 Layered Architecture (L1–L6)

The internal architecture is organized into layered components:

Layer	Description
L0	UI and API Gateway
L1	Domain Detection (DDA)
L2	Retrieval and Context Grounding (RAG)
L3	Orchestrator and Agent Runtime (MAF)
L4	Quality and Governance (Validation + Gap Loop)
L5	Composition Layer (Artifact Generation)
L6	Storage and Operations (DB, artifacts, checkpoints)

Table 19 Layered Architecture

This layered design ensures separation of concerns and allows independent evolution of system components.

4.3.3 Workflow Execution Architecture

The system executes a deterministic sequence of nodes:

Run Initialization→ Initial State Preparation→ DDA → System_Analyze → Business_Analyze → UIUX_Design → LeadDev_Plan → Validate ↔ Gap_Apply → Compose

The validation and gap stages form a loop, ensuring that outputs meet quality criteria before final composition.

RAG is applied at multiple stages (per-step grounding), providing context to improve output quality.

4.3.4 Agent Design

Here are some ways through which the AI agents help in managing the workload of the platform, as a result of the different roles assigned to each one of them:

- System Analyzer: Obtains the primary context of the project by identifying the key objective, the end-user, and some basic assumptions on the system.
- Business Analyst: Creates detailed user stories, alongside acceptance criteria.
- UI/UX Designer Agent: Identifies layouts for the visual interface and establishes design rules.
- Lead Developer Agent: Creates layouts for the APIs and creates a technical procedure for implementing the APIs.
- Validation Agent: Validates the material created for quality assurance purposes.
- Gap Filler Agent: Fixes errors in the created material or fills in missing parts of it.

Having clear roles for every AI agent helps make the whole system modular, all actions trackable, and debugging easy.

4.3.5 Data Flow in Architecture

The system processes data in a structured sequence:

- Brief → Context Pack
- Context Pack → User Stories
- User Stories → Validation
- Validation → Gap Correction
- Gap → Composition
- Composition → Final Artifact Bundle

This structured flow ensures that outputs are validated and refined before final delivery.

4.3.6 Integration Architecture

To facilitate further development of its features, it is connected to several third-party systems, among which are:

• Jira Integration: It allows you to manually transfer the information about your requirements to the Jira platform. To prevent any error from occurring, the system enables you to conduct a dry run of the process and see the level of data mapping, creating a hierarchy of tasks and issues.

• Stitch Integration: It is intended for creating designs of the user interface. Its work is automatic (asynchronous). It polls rendering progress and delivers the final visual data using the secure proxy endpoint.

• RAG Integration: The RAG Integration: Right away, it uses the logic of artificial intelligence that is based on real facts from the field at each step of the process. However, it is very flexible and customizable since you can modify its settings according to your needs.

4.4. Logical Design

Class Diagram

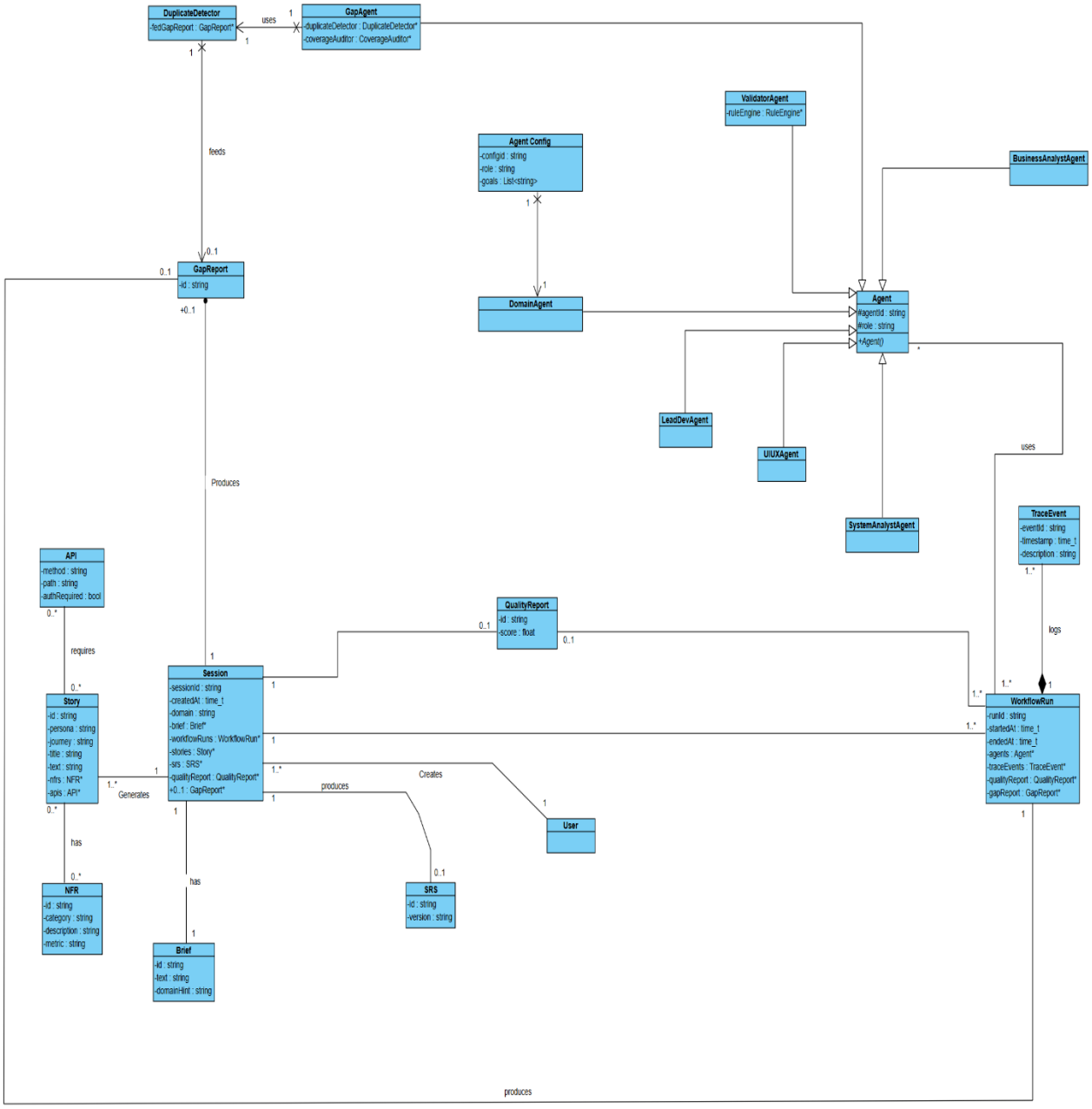


Figure 11 Class Diagram

4.5. Dynamic View

4.5.1 Sequence Diagrams

4.5.1.1. Sequence Diagram Submit Project Brief

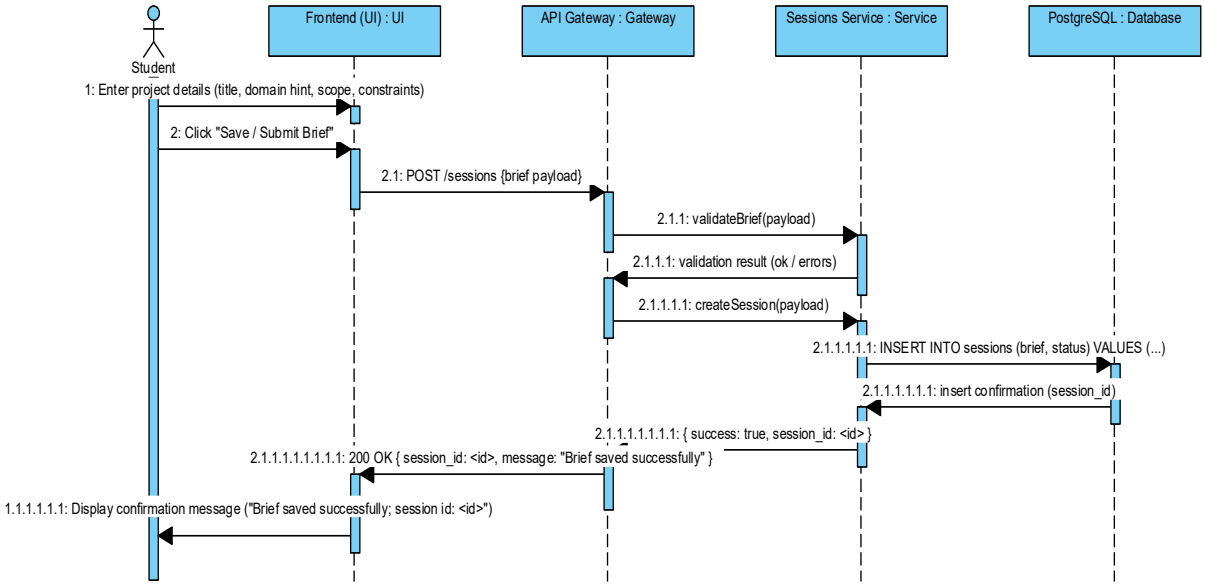


Figure 12 Sequence Diagram Submit Project Brief

4.5.1.2. Sequence Diagram Run Multi Agent Generation

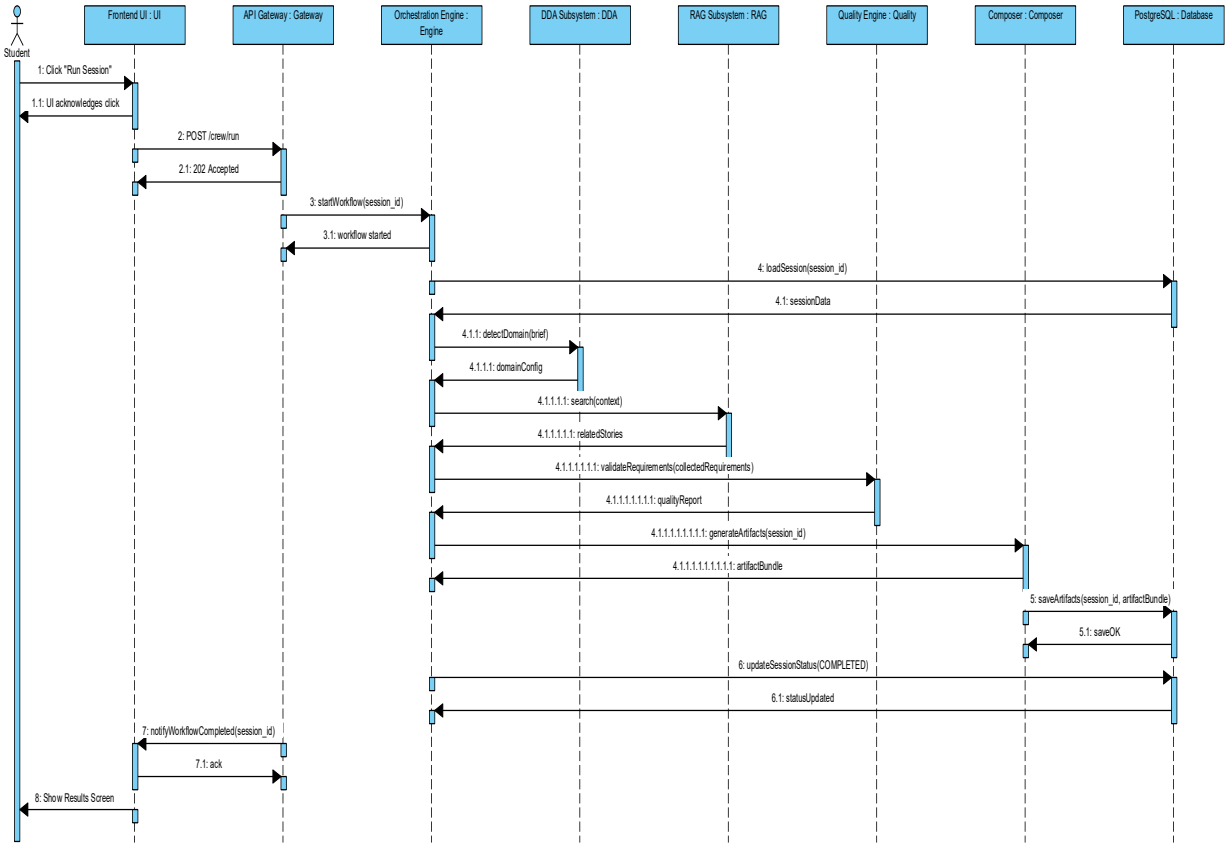


Figure 13 Sequence Diagram Run Multi Agent Generation

4.5.1.3. Sequence Diagram Workflow Progress

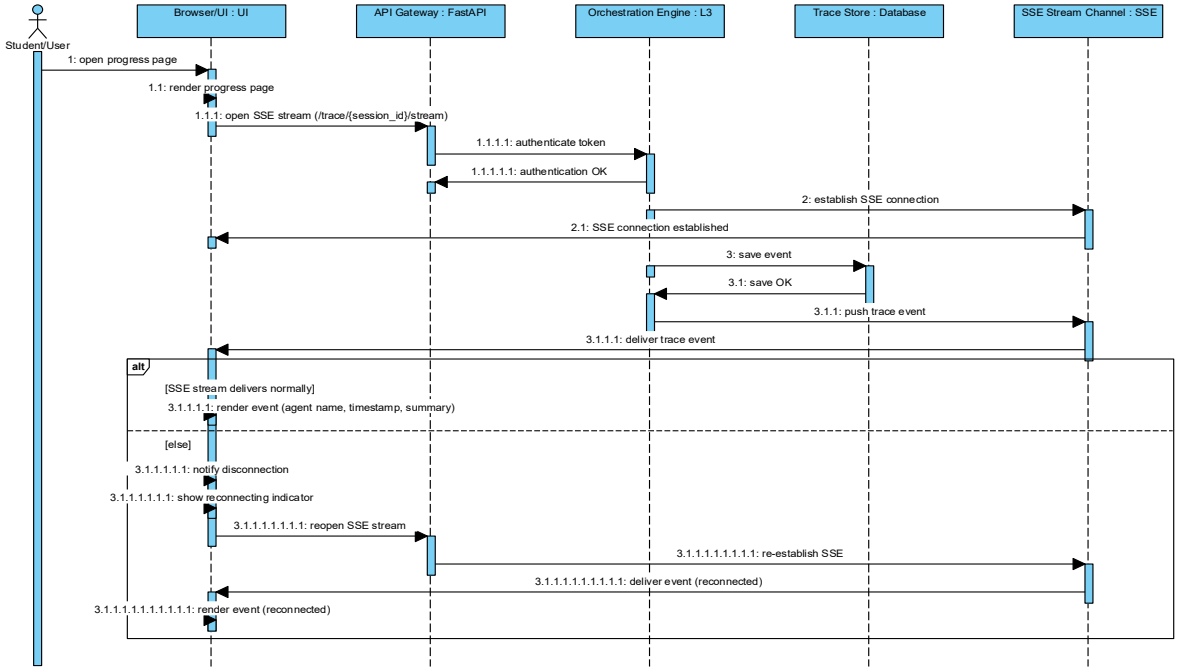


Figure 14 Sequence Diagram Workflow Progress

4.5.1.4. Sequence Diagram Jira Export

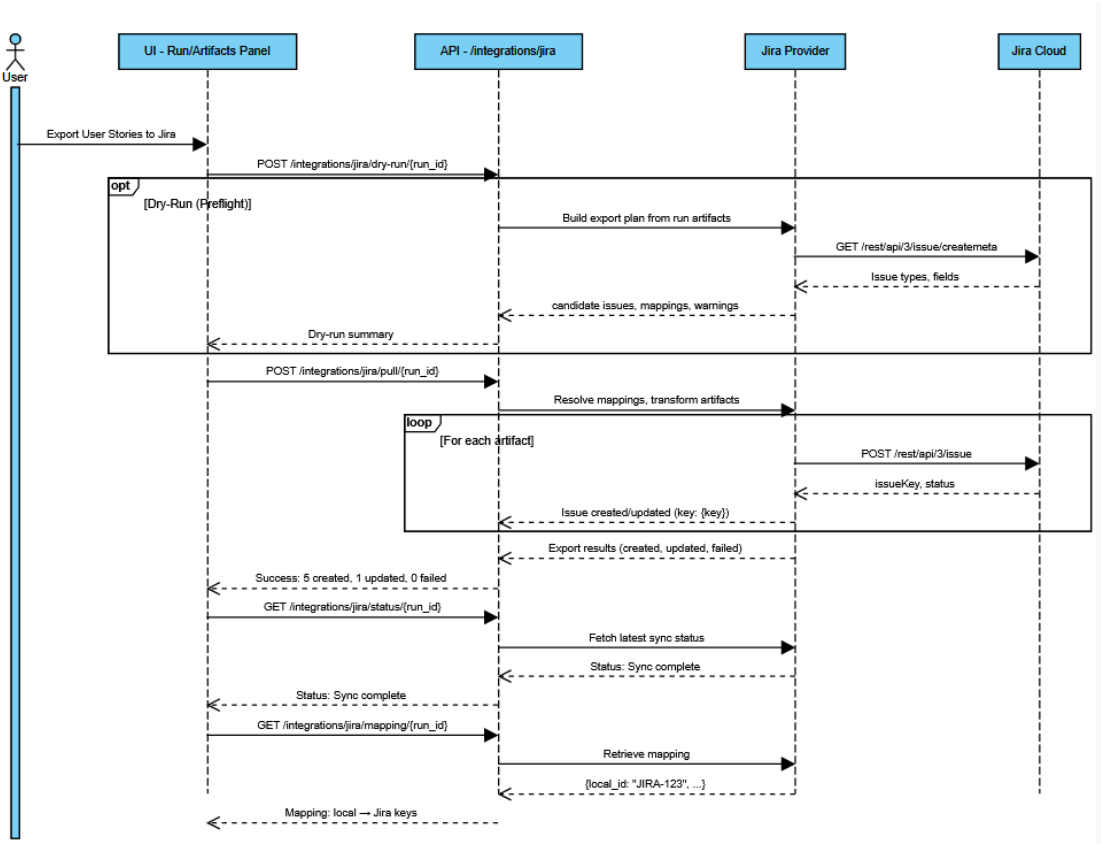


Figure 15 Sequence Diagram Jira Export

4.5.1.5. Sequence Diagram Stich UI Screen

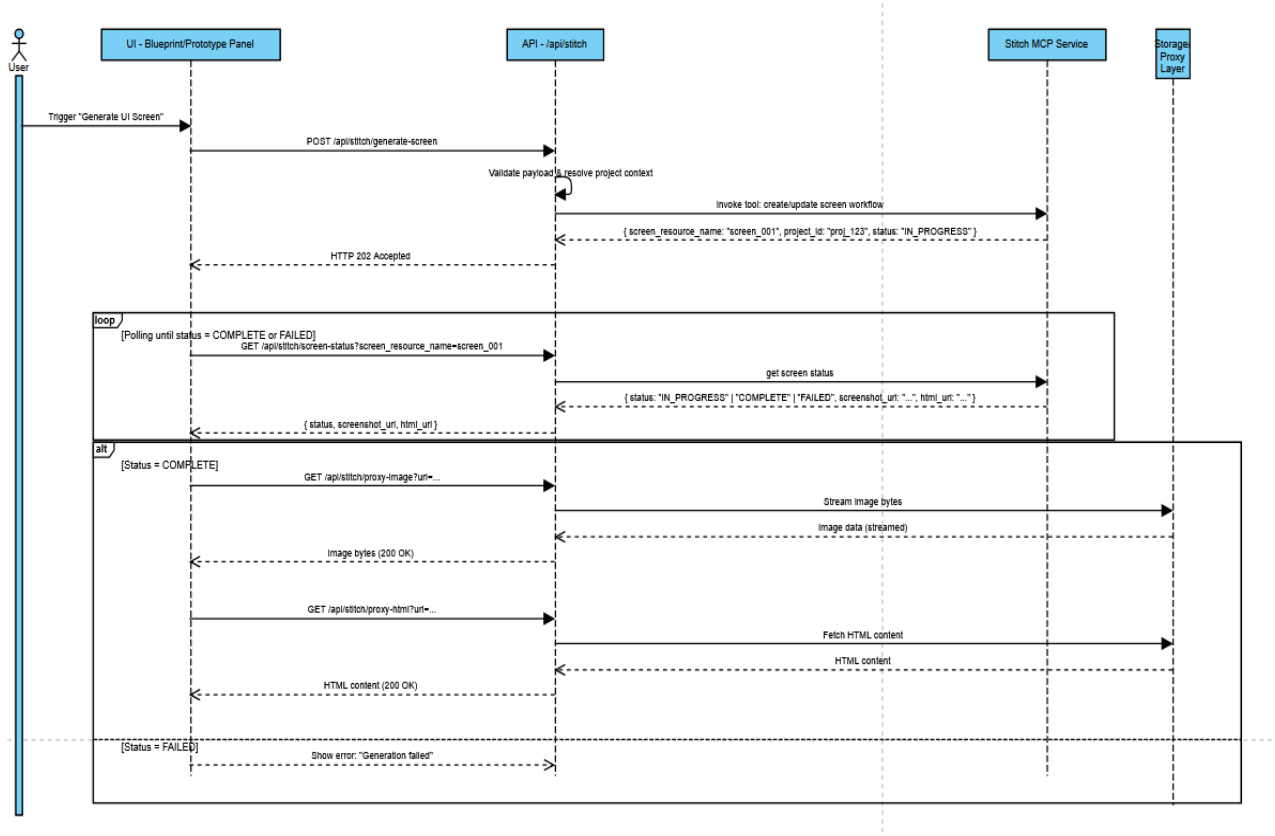


Figure 16 Sequence Diagram Stich UI Screen

4.5.2 Activity Diagrams

4.5.1.6. Activity Diagram Submit Brief

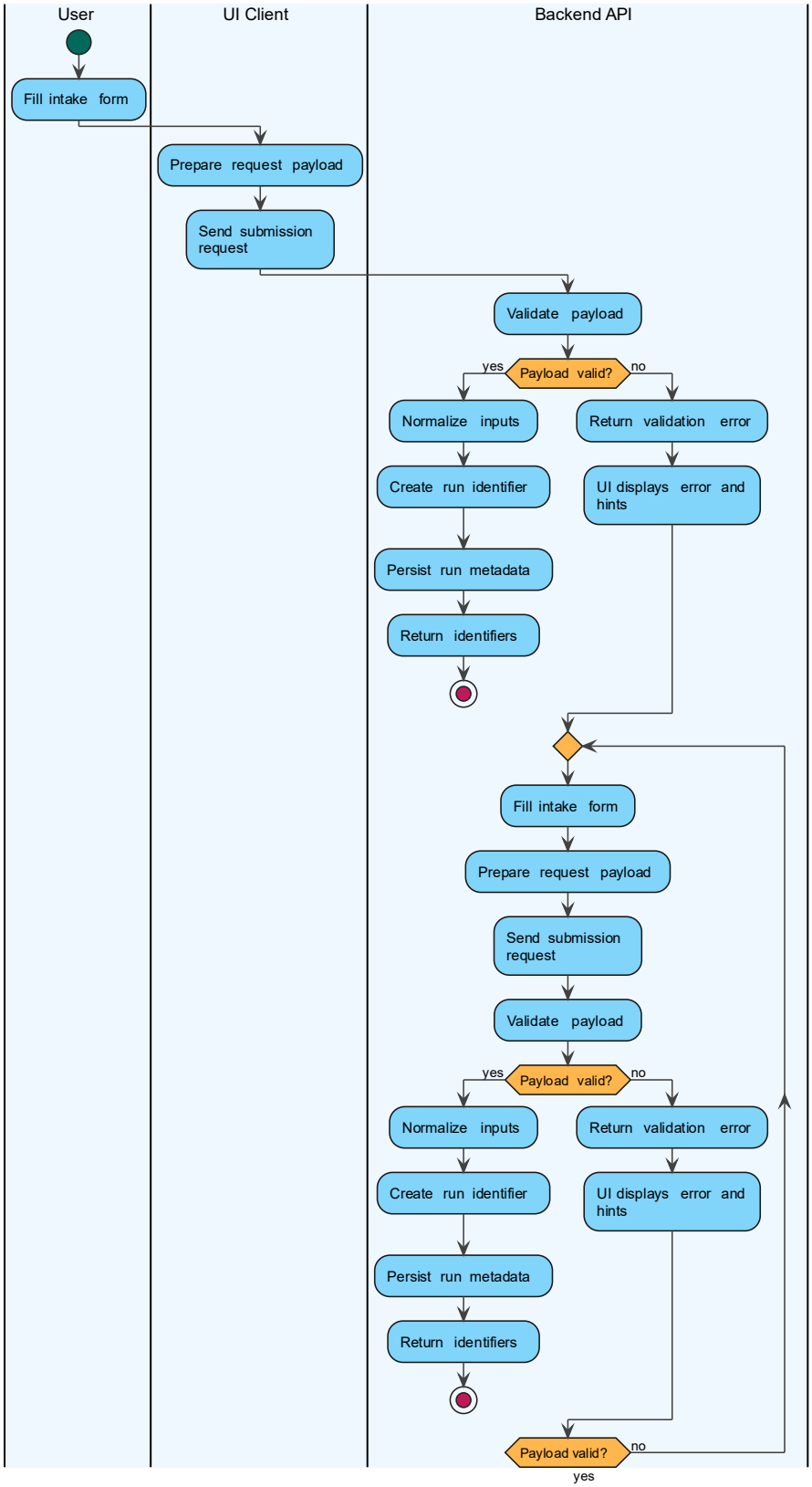


Figure 17 Activity Diagram Submit Brief

4.5.1.7. Activity Diagram Artifact Export

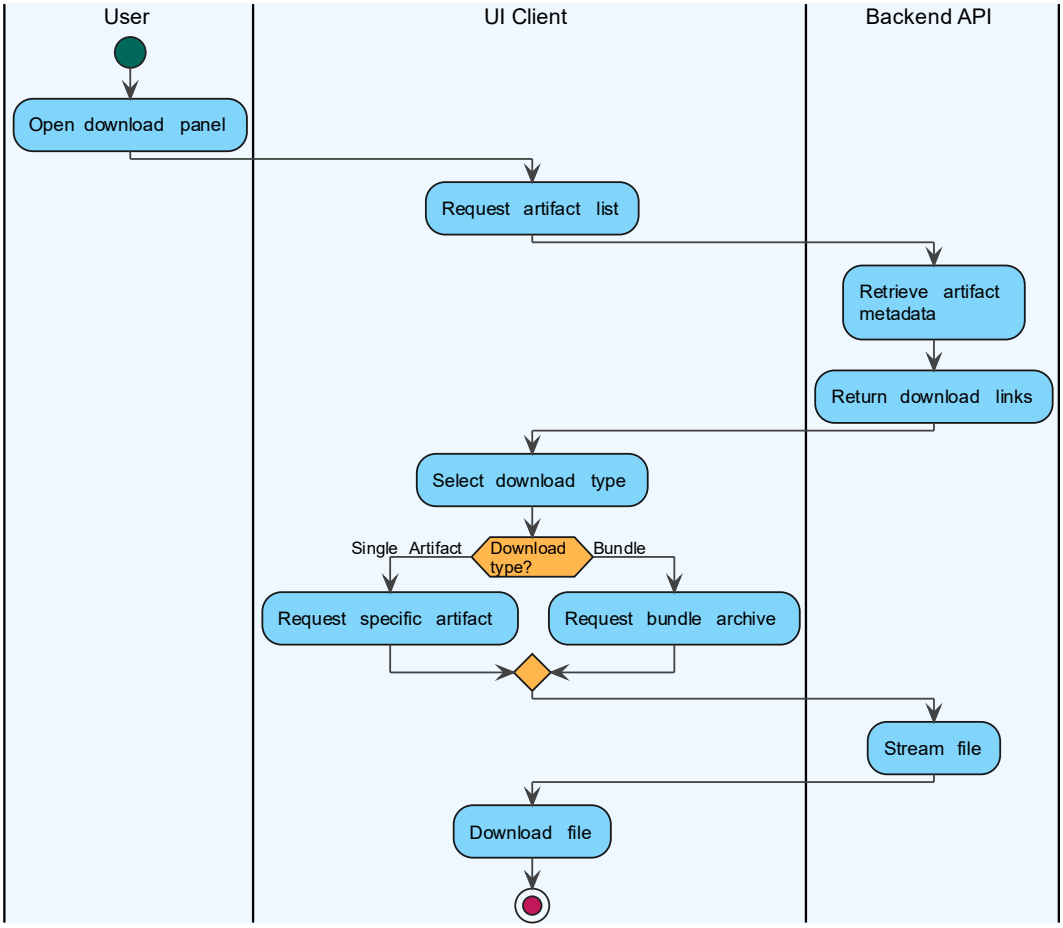


Figure 18 Activity Diagram Artifact Export

4.5.1.8. Activity Diagram Requirements Generation Workflow

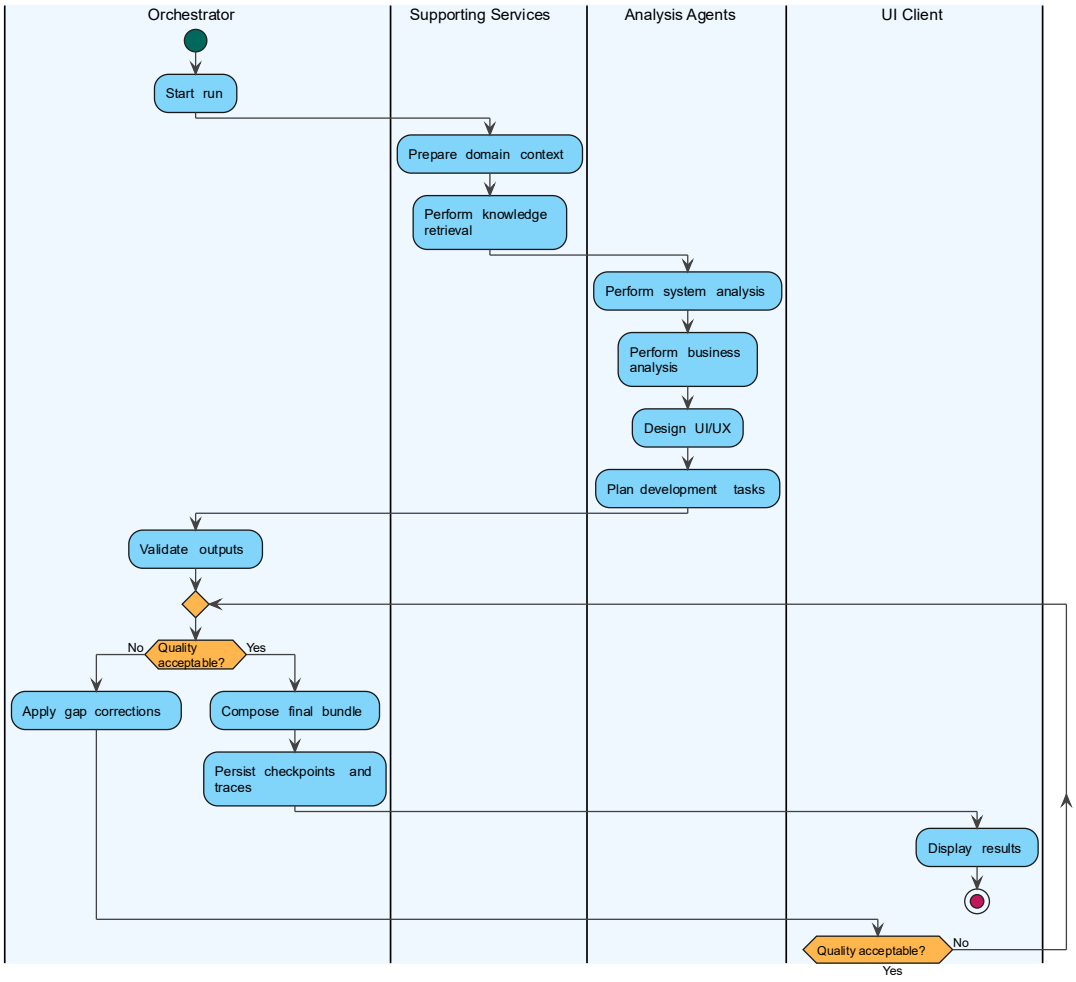


Figure 19 Activity Diagram Requirements Generation Workflow

4.5.1.9. Activity Diagram Export to Jira

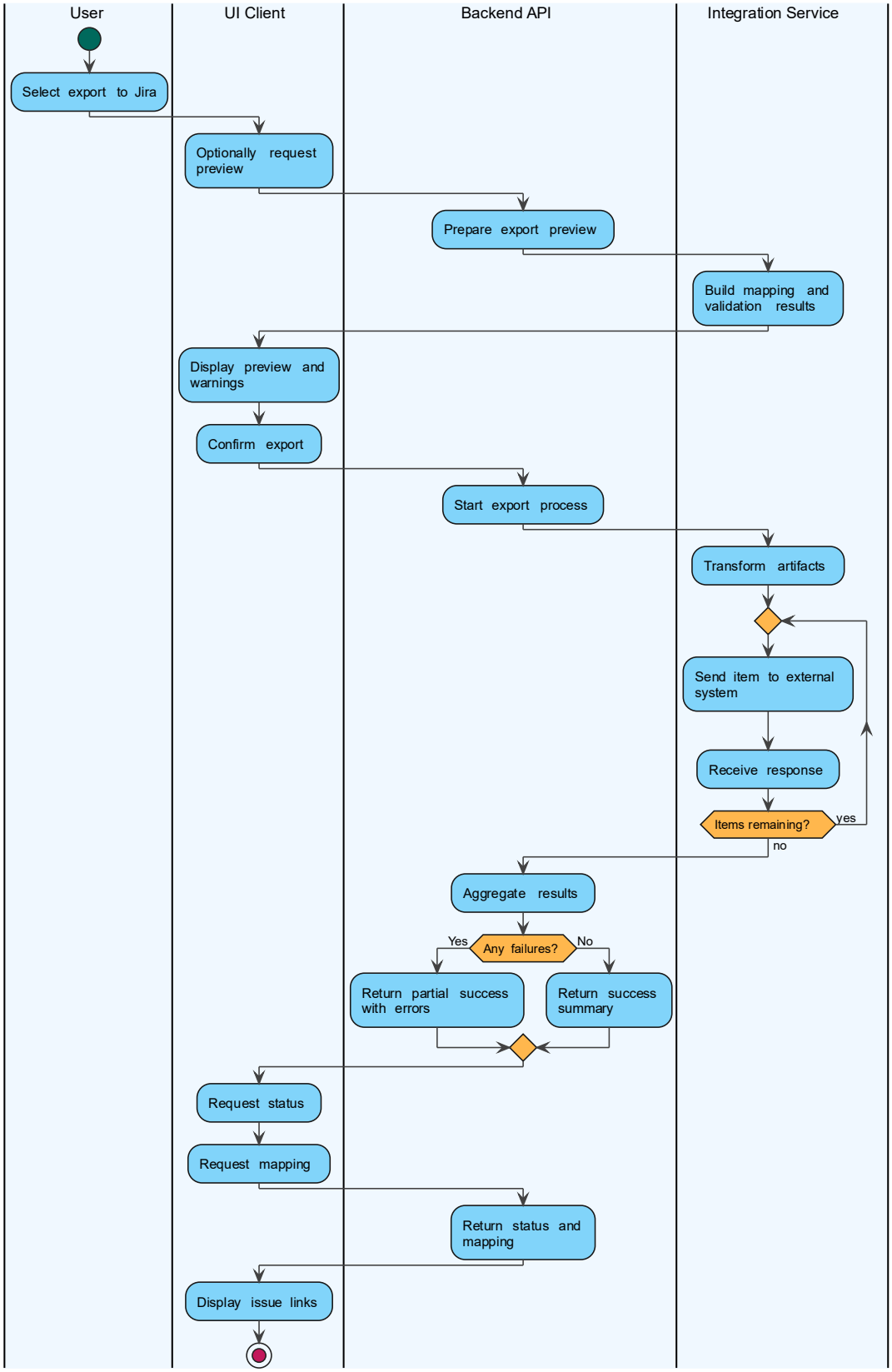


Figure 20 Activity Diagram Export to Jira

4.5.1.10. Activity Diagram Stich Screen UI

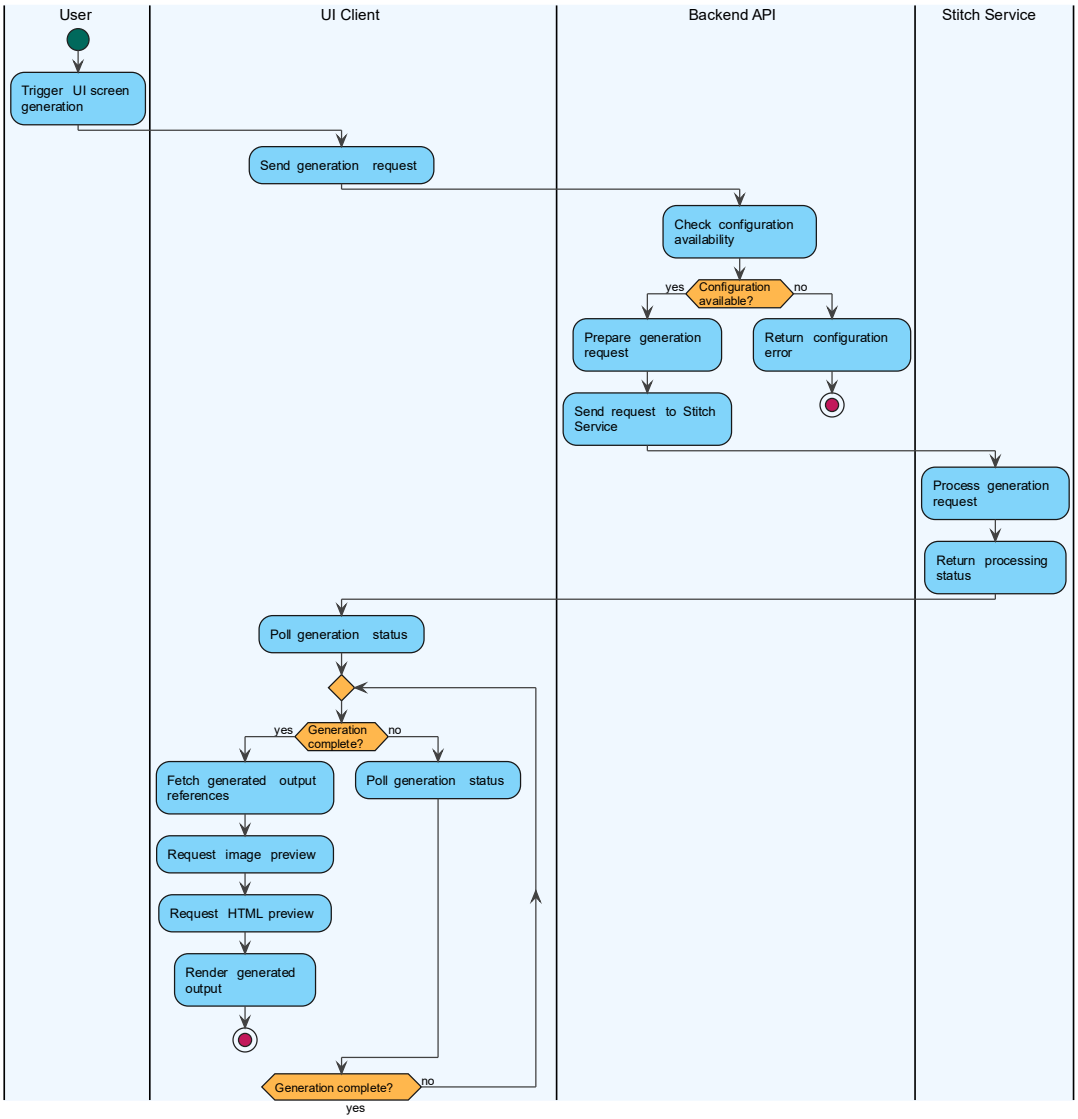


Figure 21 Activity Diagram Stich Screen UI

4.6. Component Design

4.6.1 Deployment Diagram

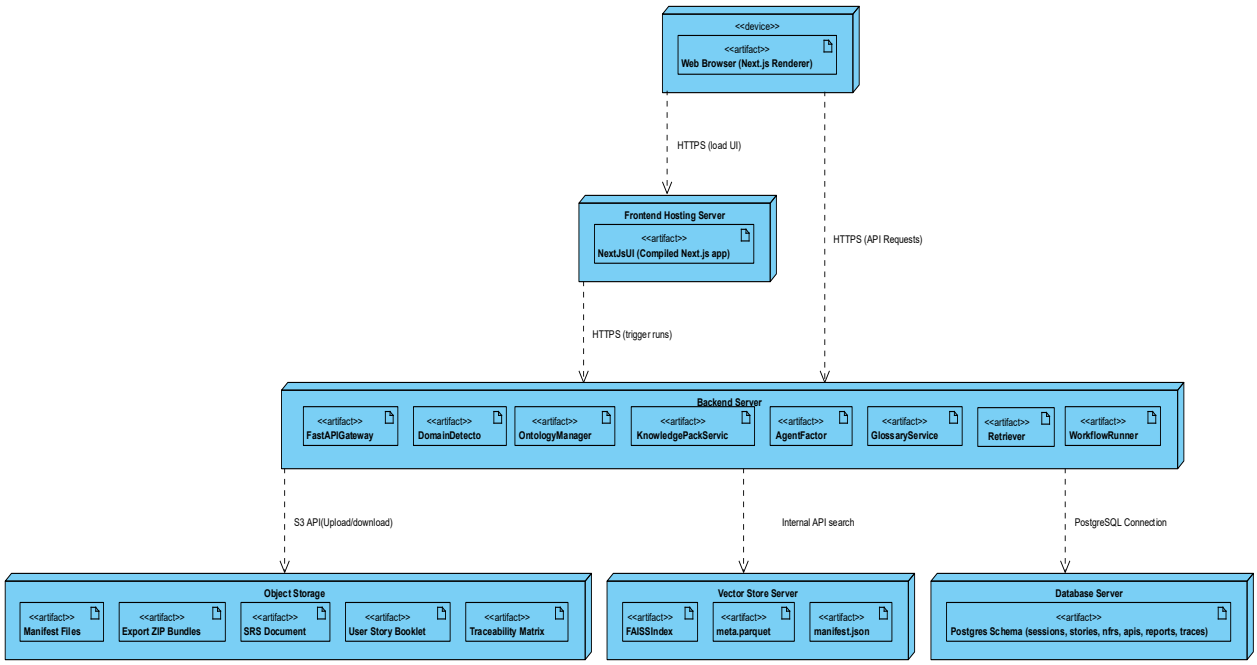


Figure 22 Deployment Diagram

4.6.2 Component Diagram

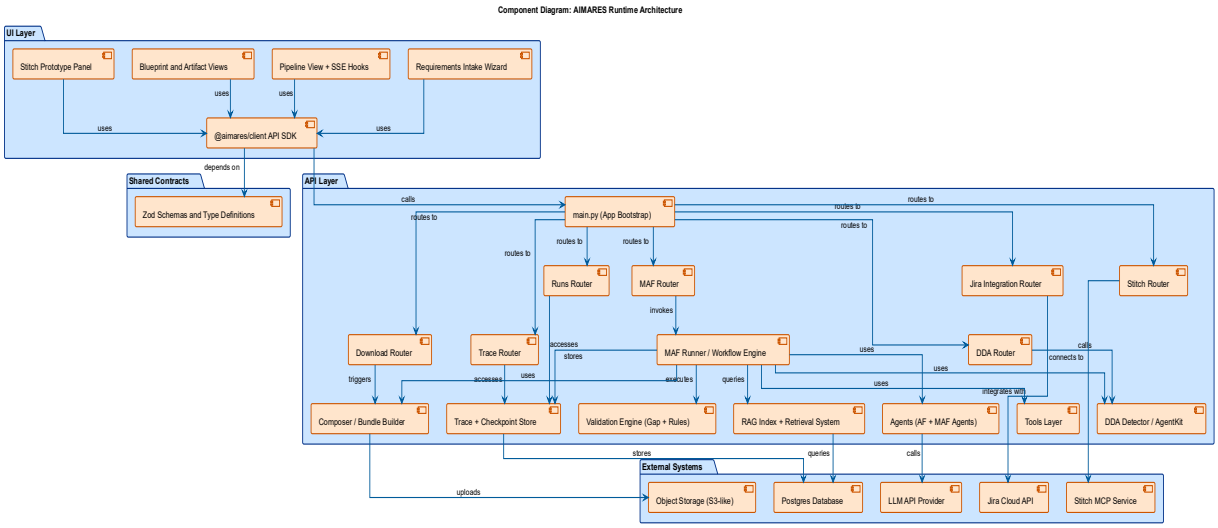


Figure 23 Component Diagram

4.6.3 Package Diagram

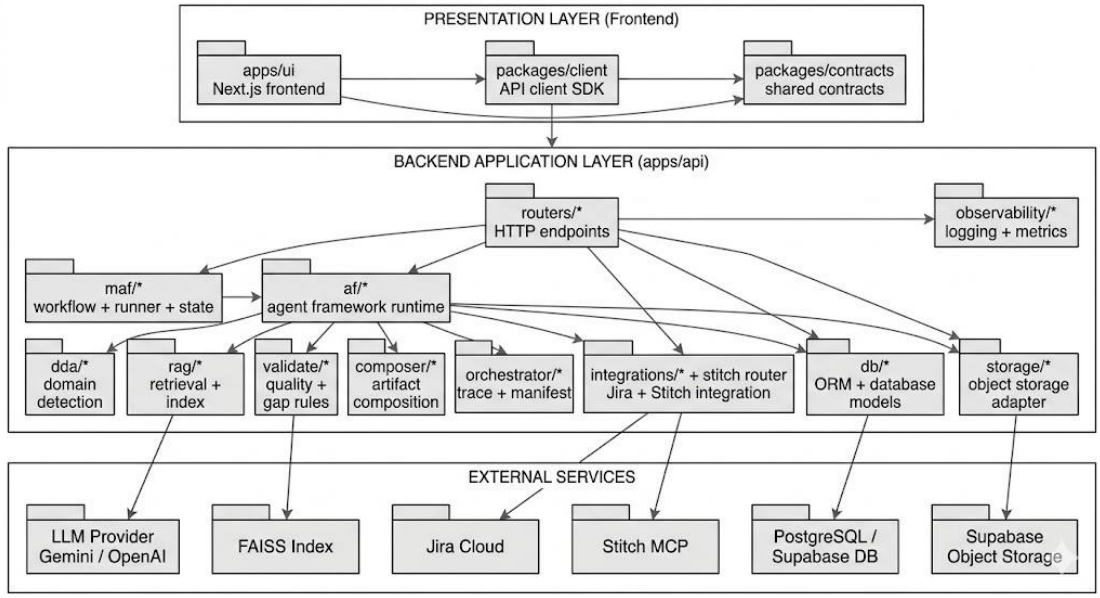


Figure 24 Package Diagram

4.7. Data Models

4.7.1 ER Diagram

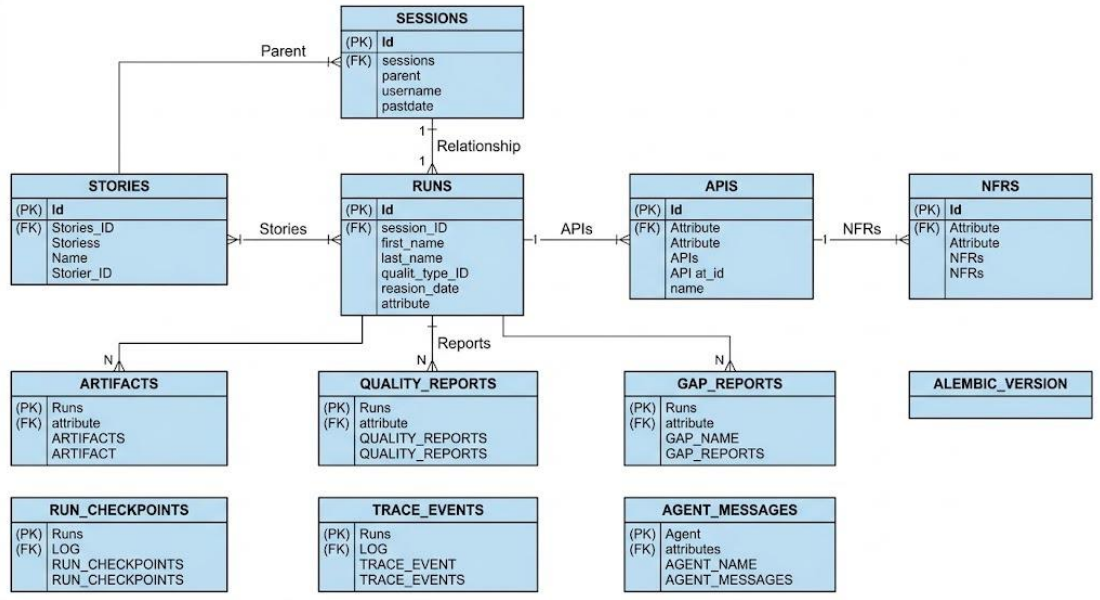


Figure 25. ER Diagram

4.8. User Interface Design

4.8.1 Main Project Intake Window

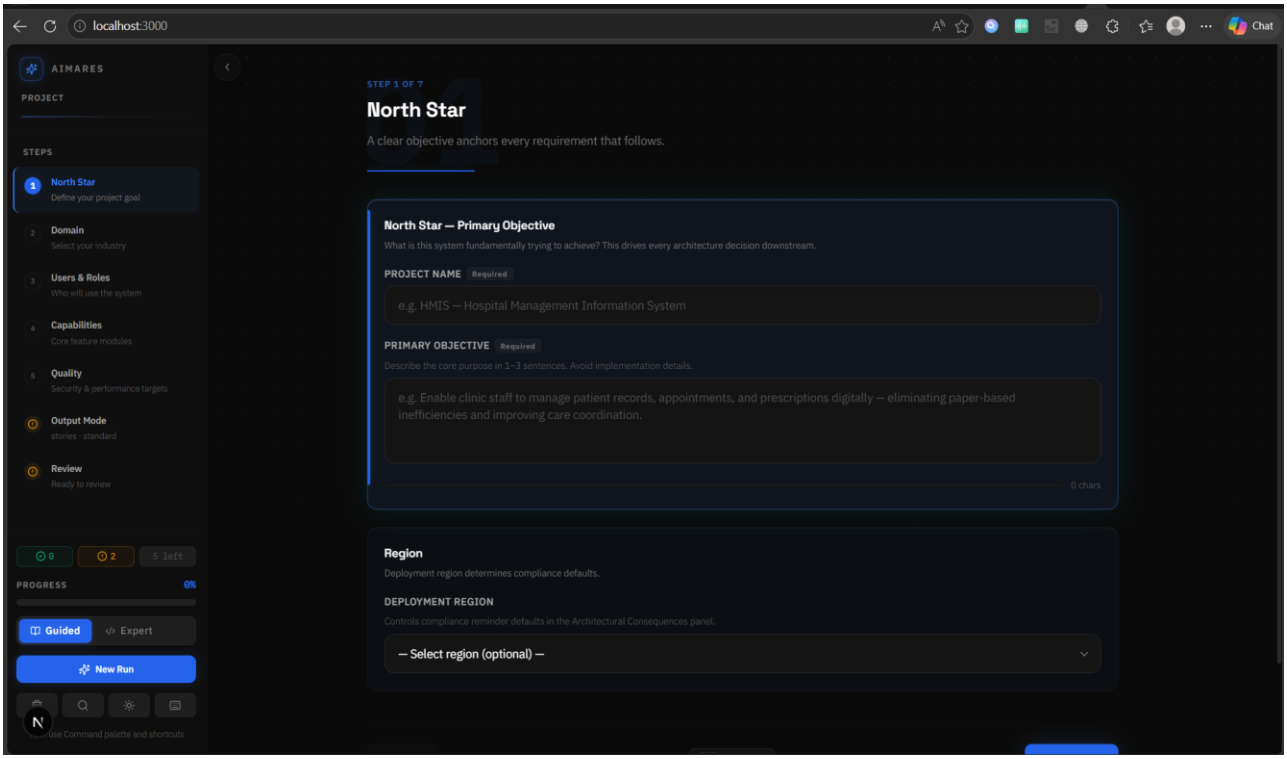


Figure 26 Project Intake

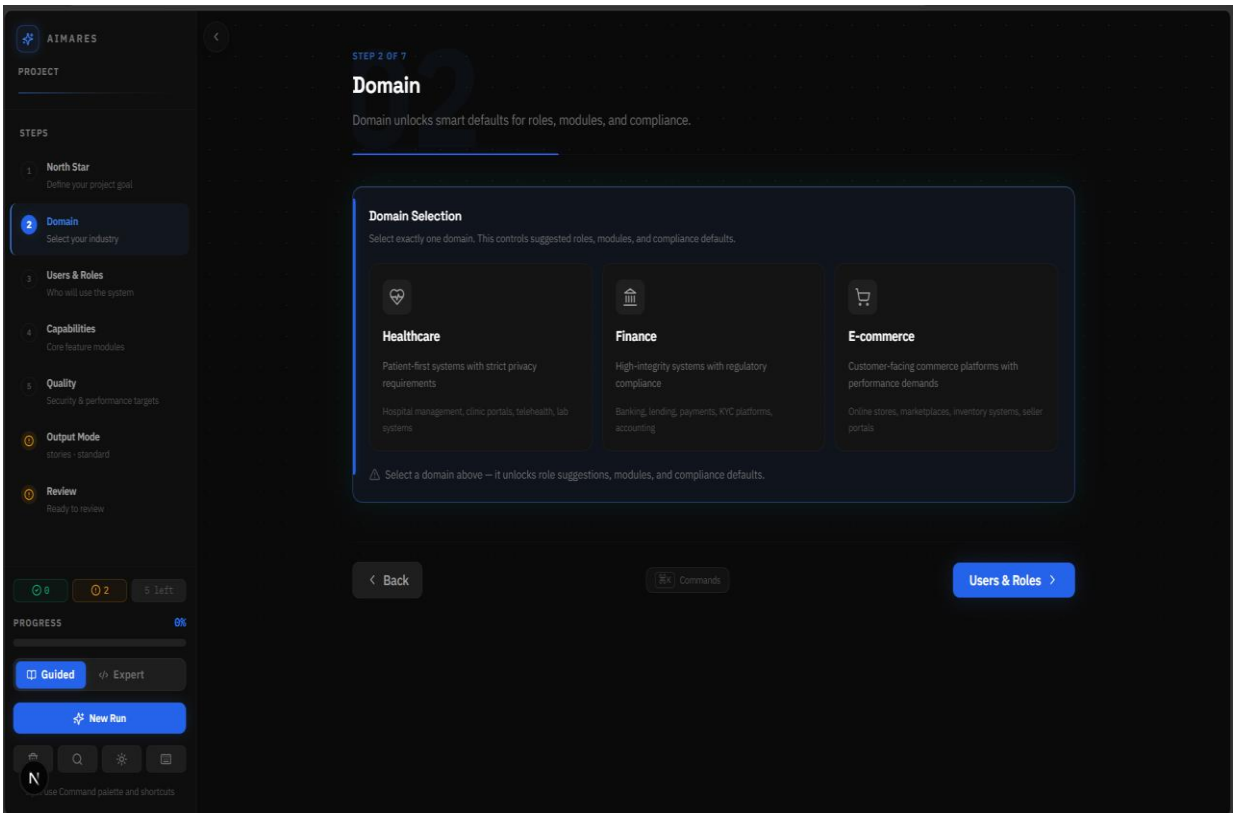


Figure 27 Project Domain

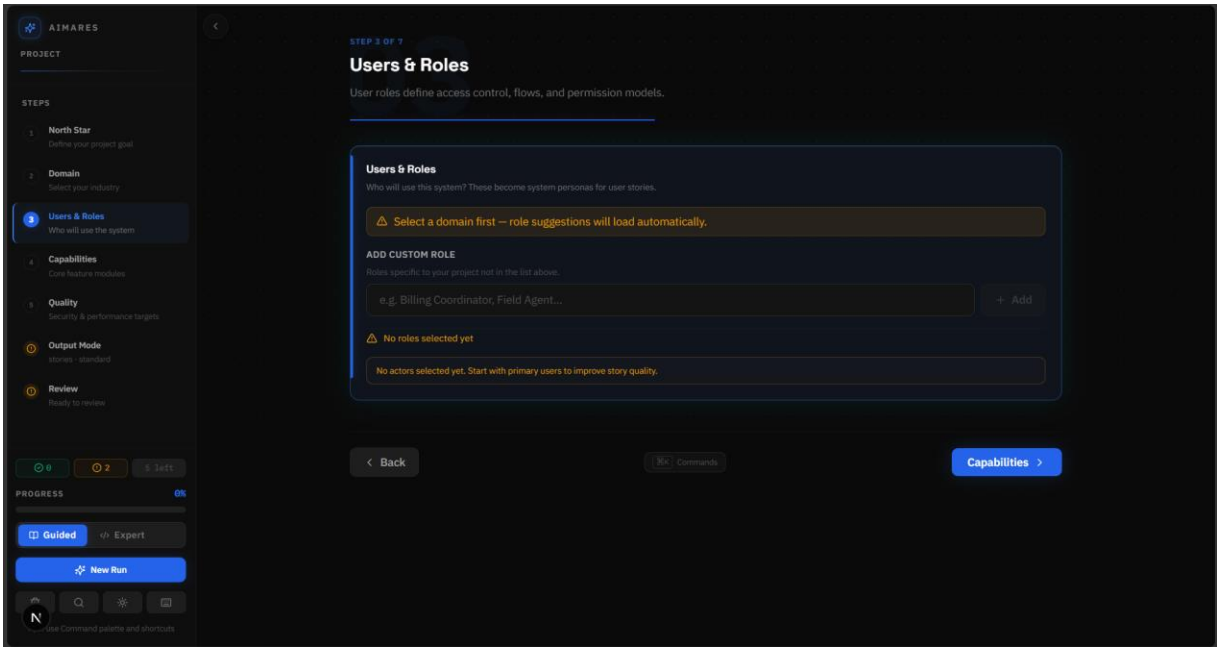


Figure 28 User Roles

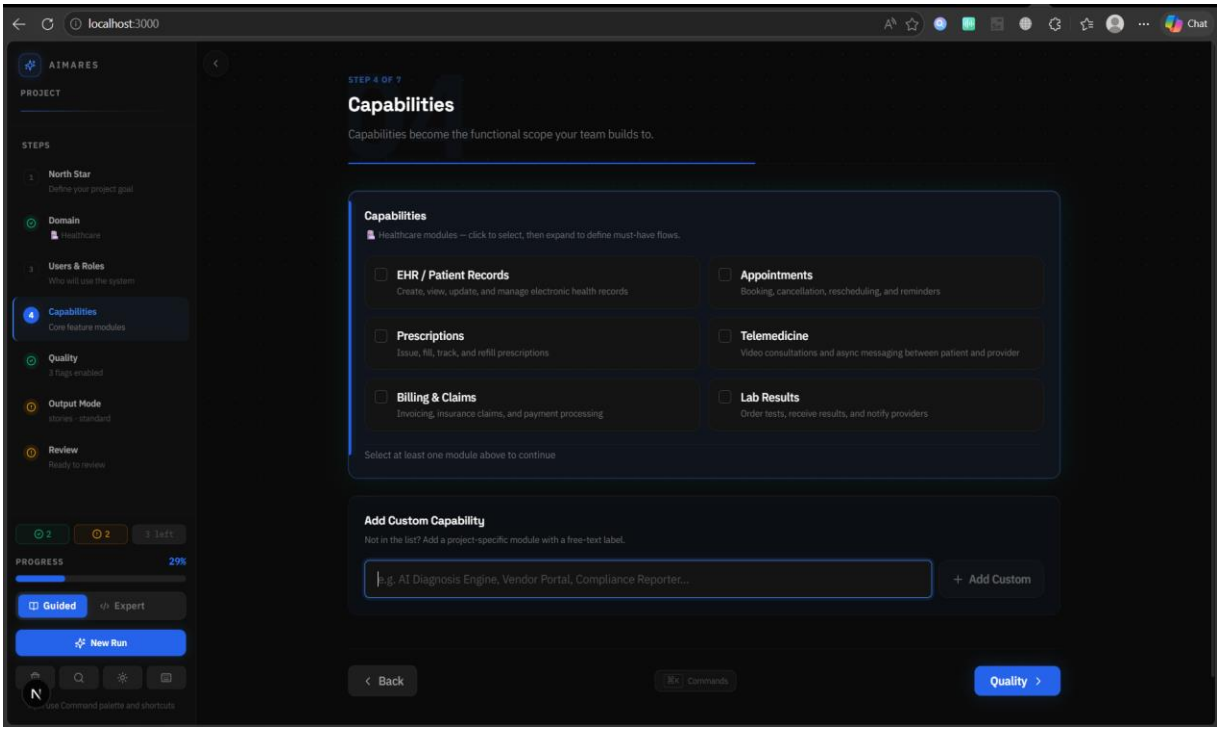


Figure 29 Requirement Modules

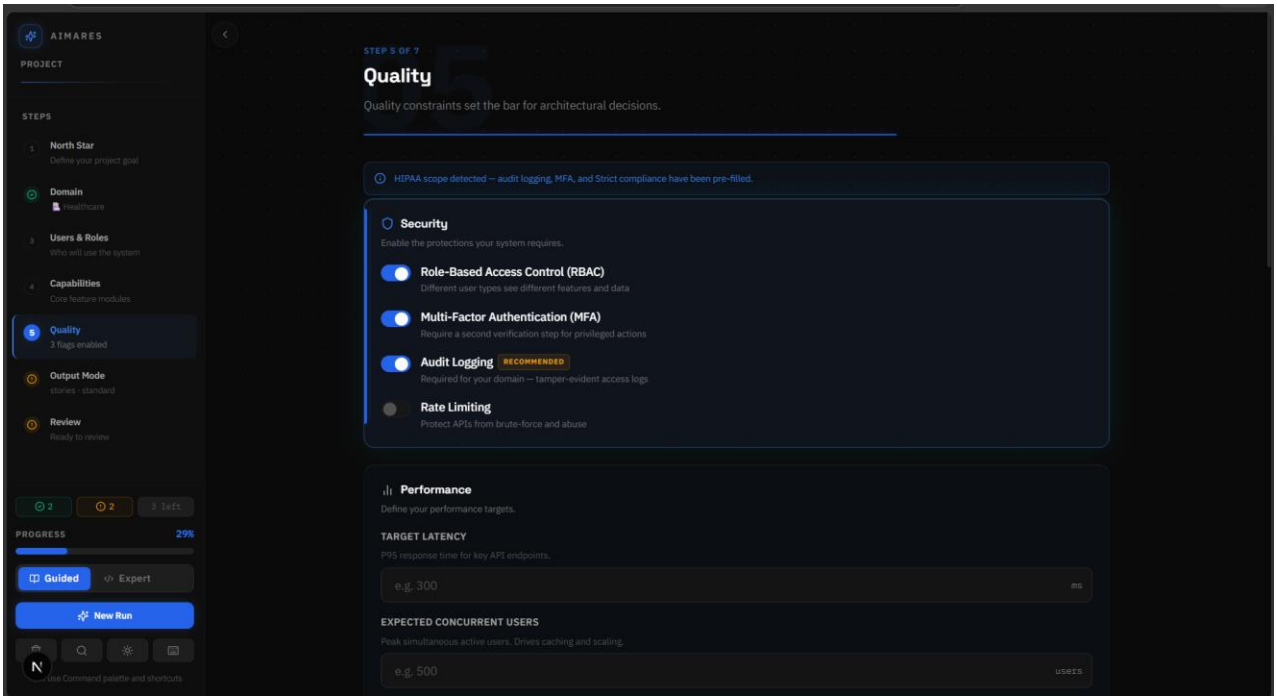


Figure 30 Quality Window

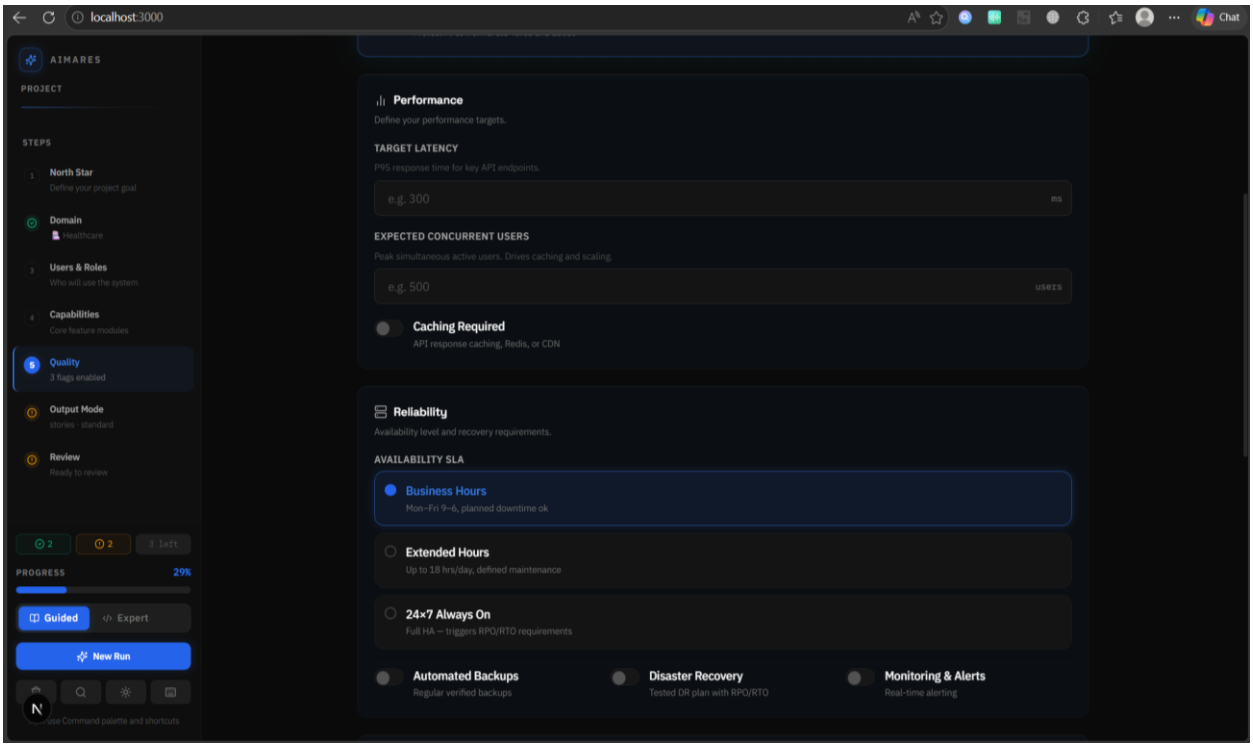


Figure 31 Quality Window

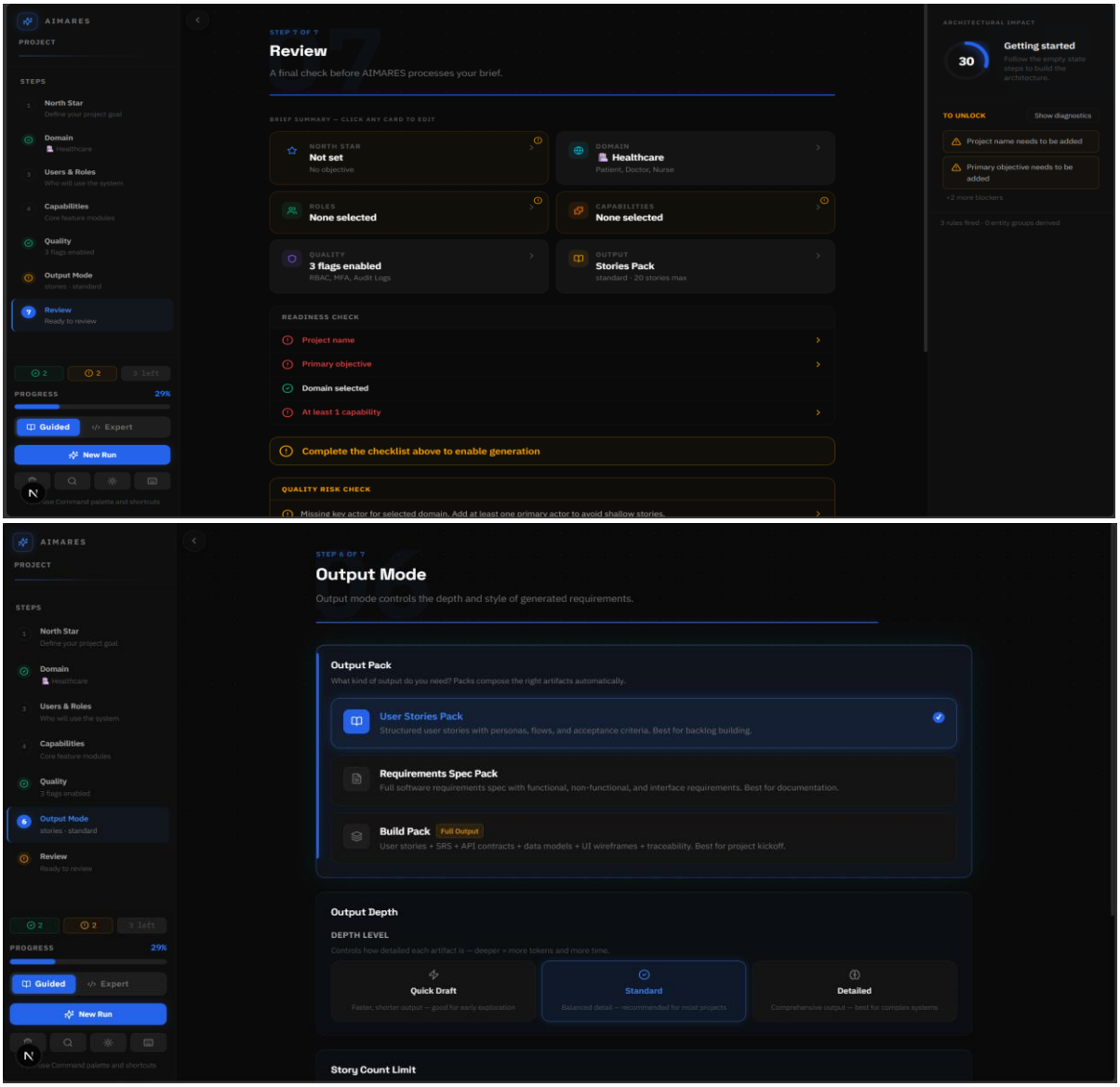


Figure 32 Final Prompt Submission

4.8.2 View Pipeline:

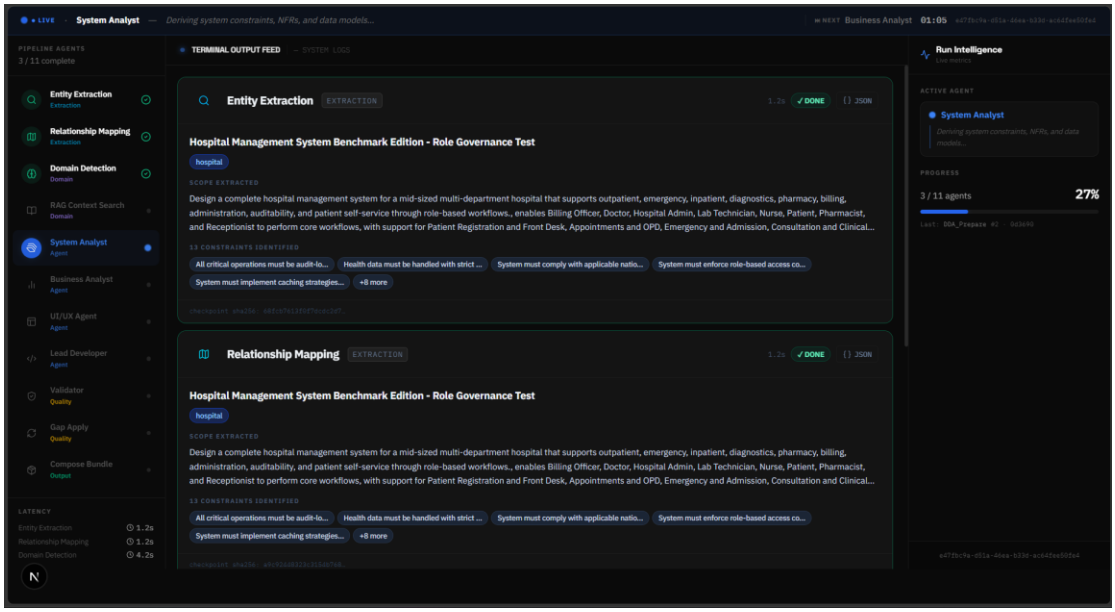


Figure 33 Pipeline

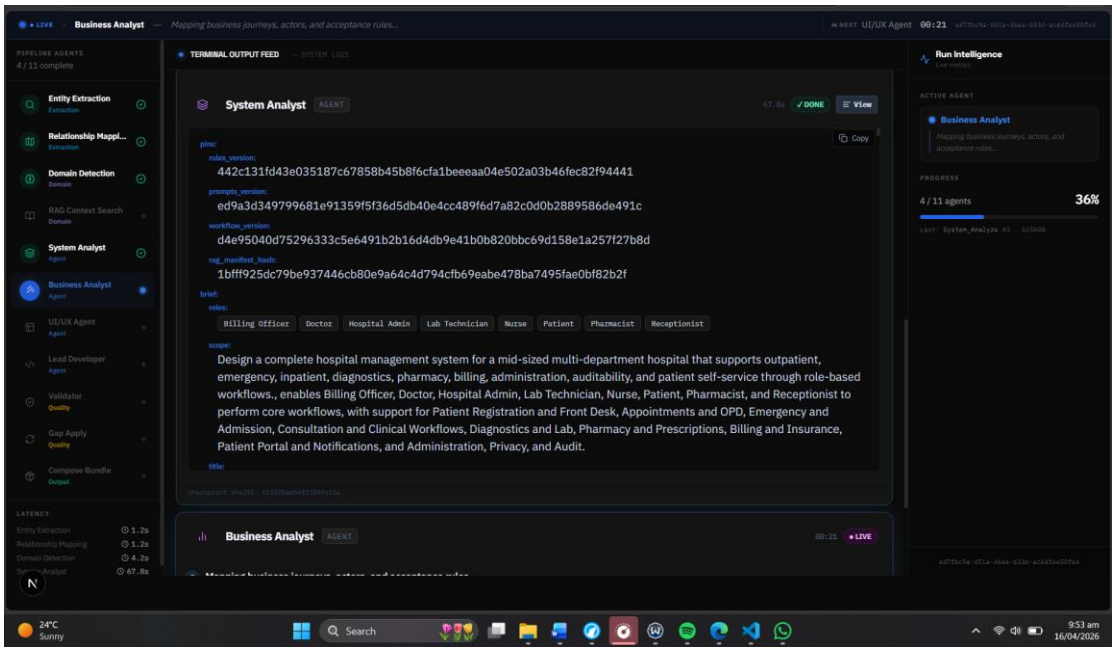


Figure 34 Pipeline System Analyst

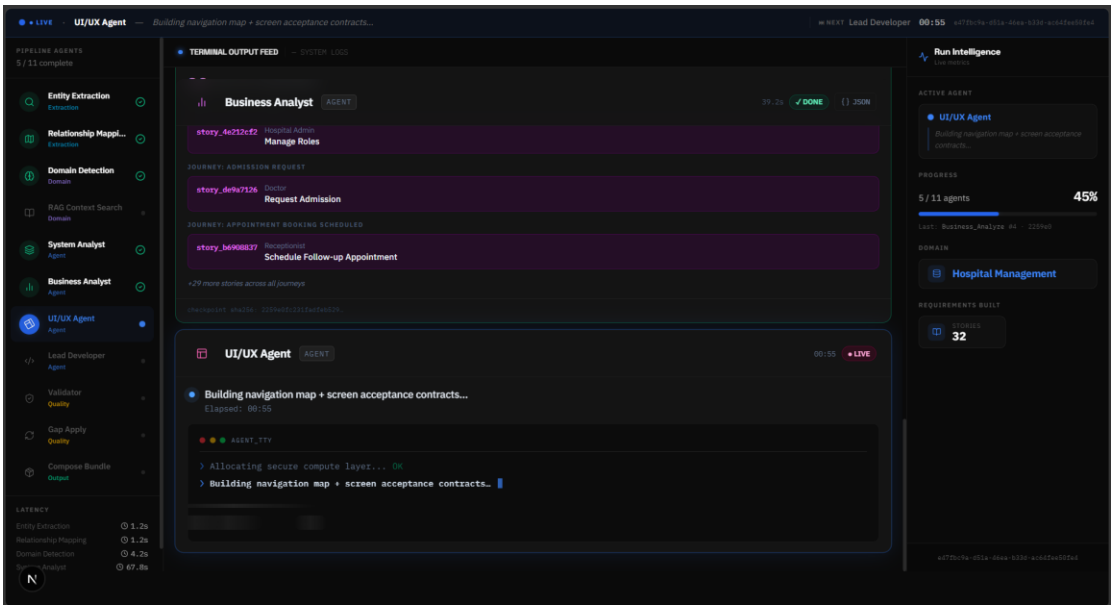


Figure 35 Pipeline Business Analyst

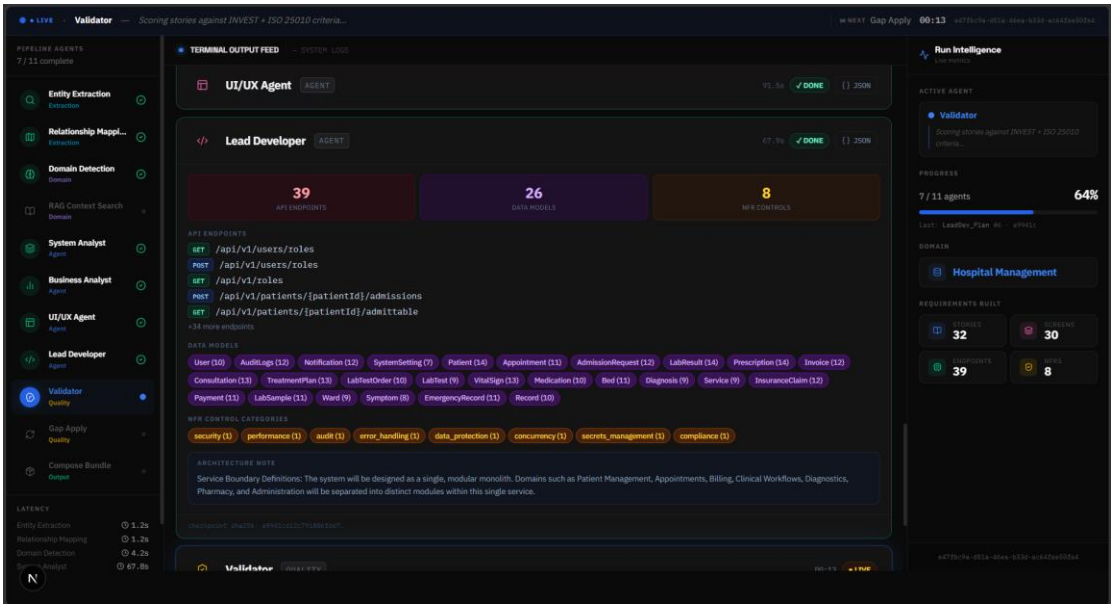


Figure 36 Lead Developer

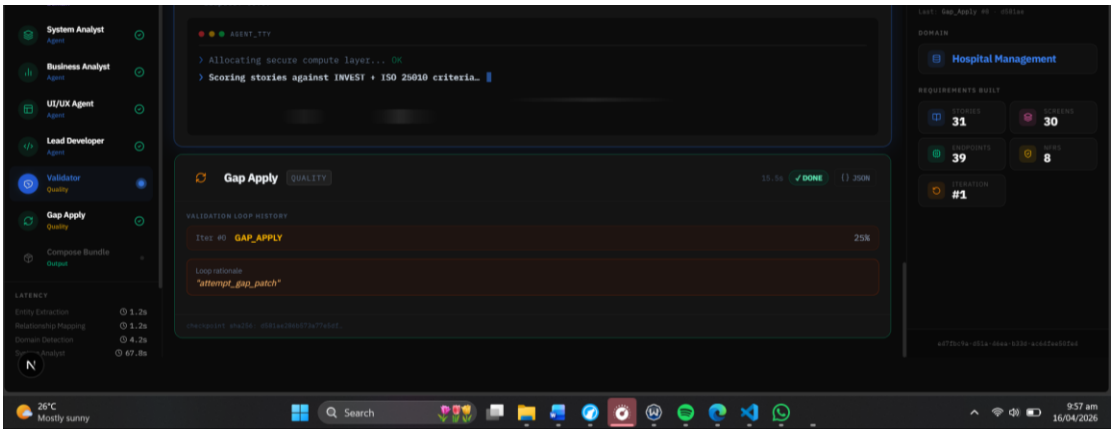


Figure 37 Gap Agent

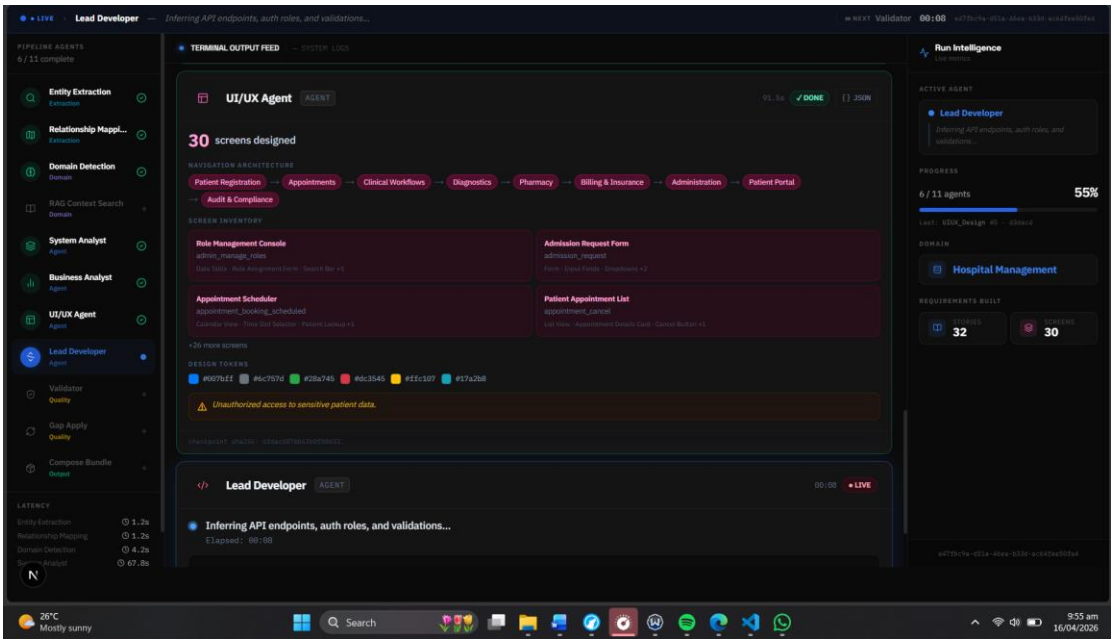


Figure 38 UI/UX Agent

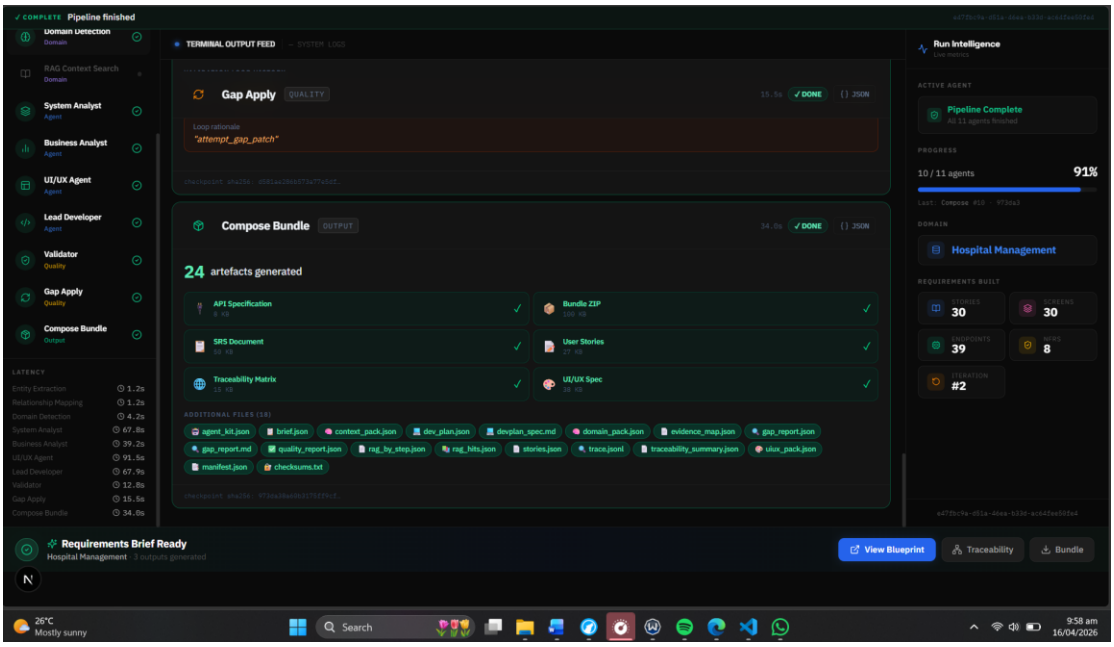


Figure 39 Pipeline Complete

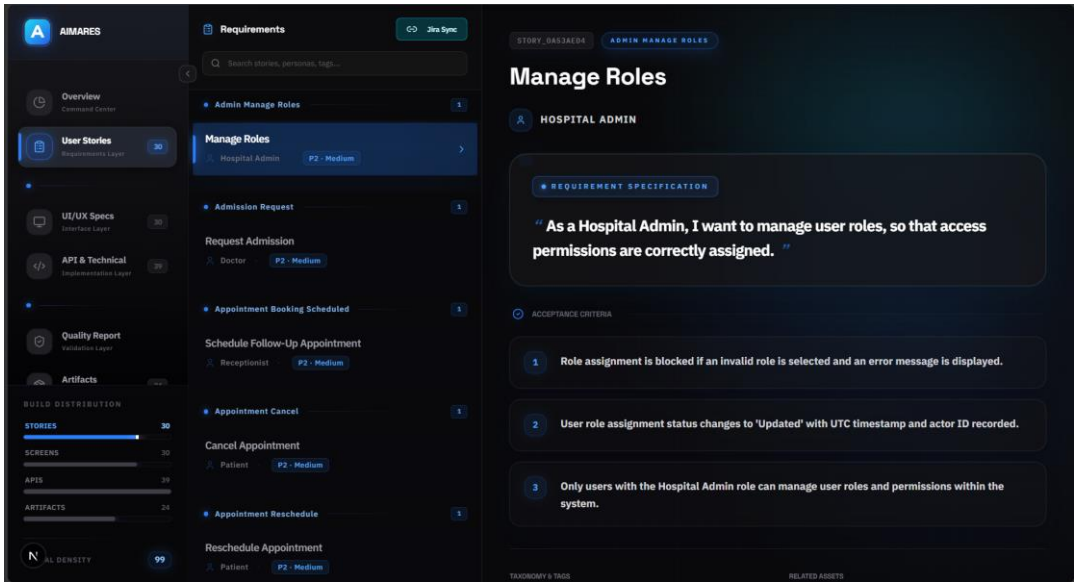


Figure 40 User Stories

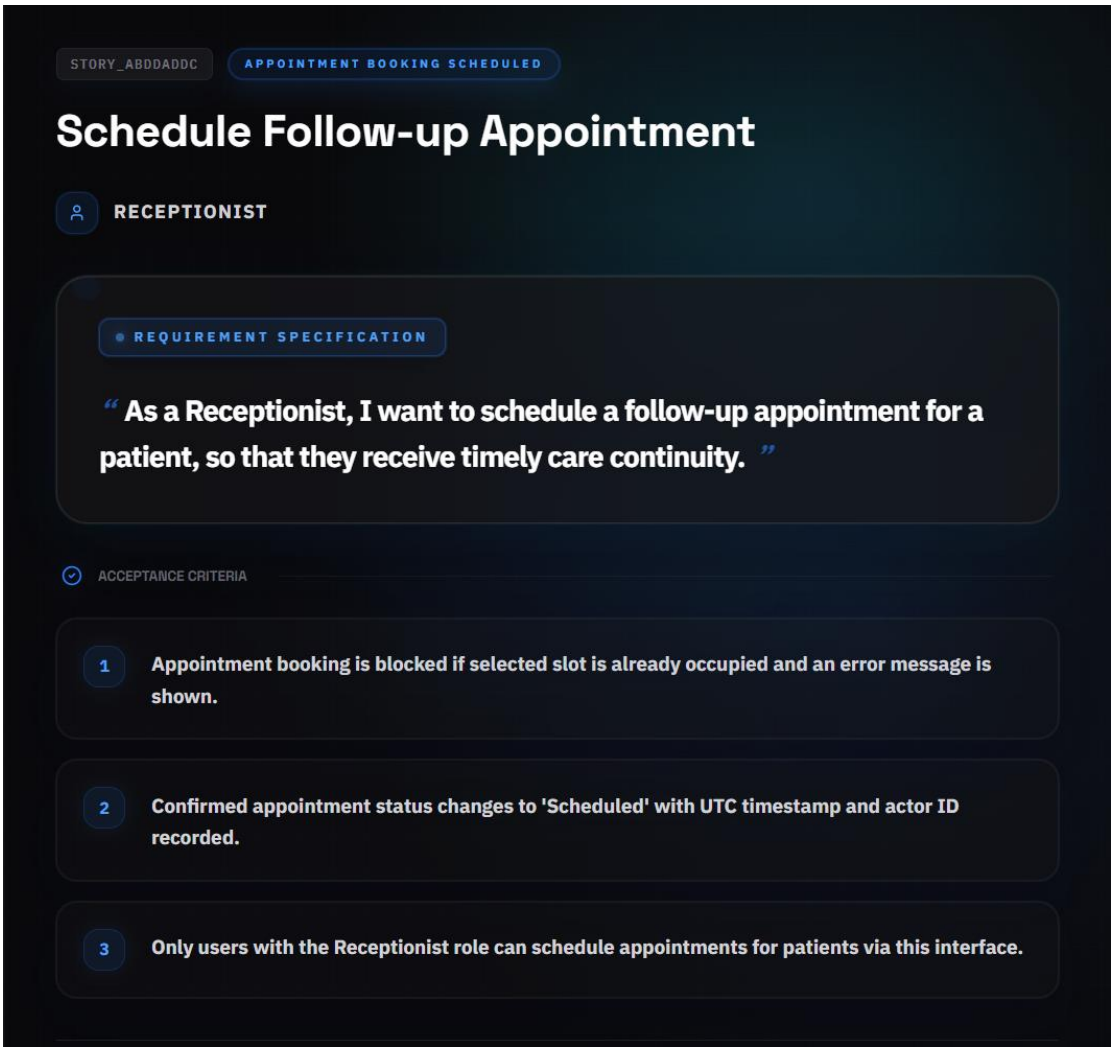


Figure 41 User Stories

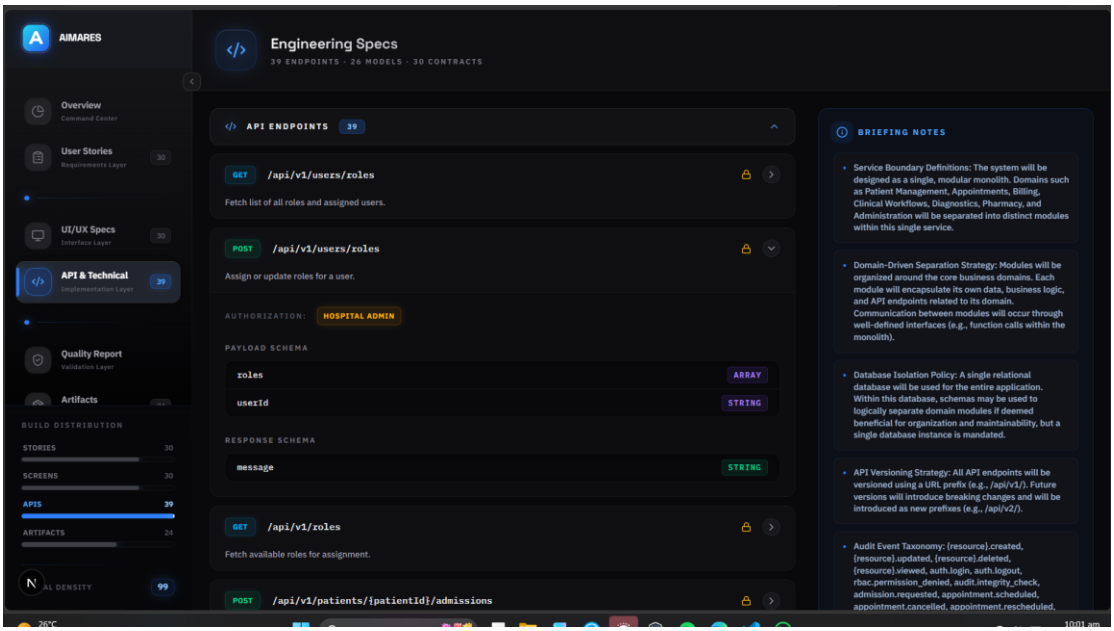


Figure 42 Api Endpoints

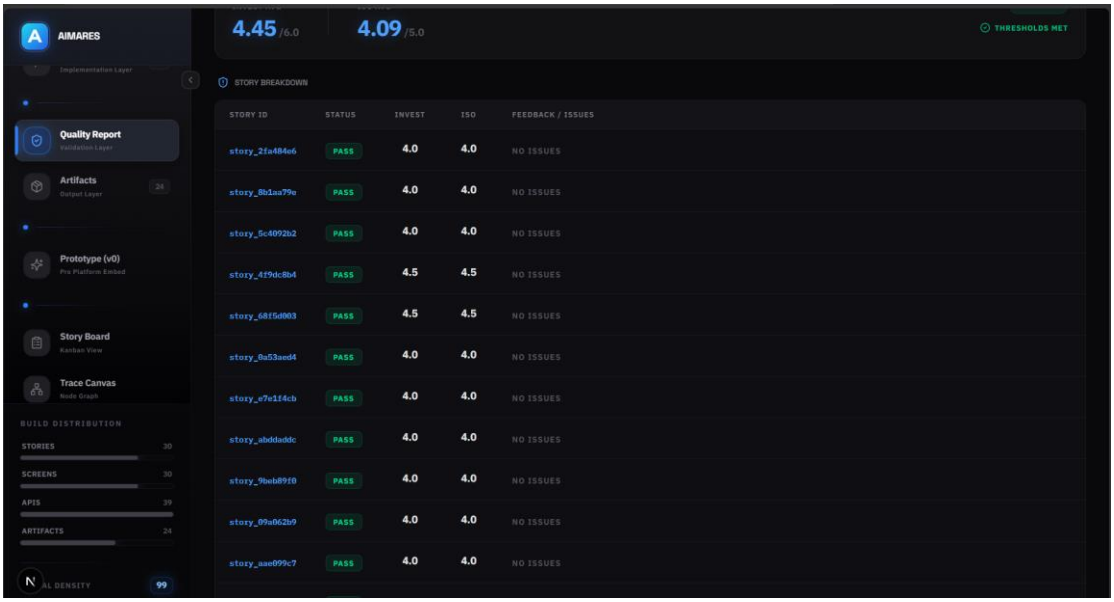


Figure 43 Quality Report

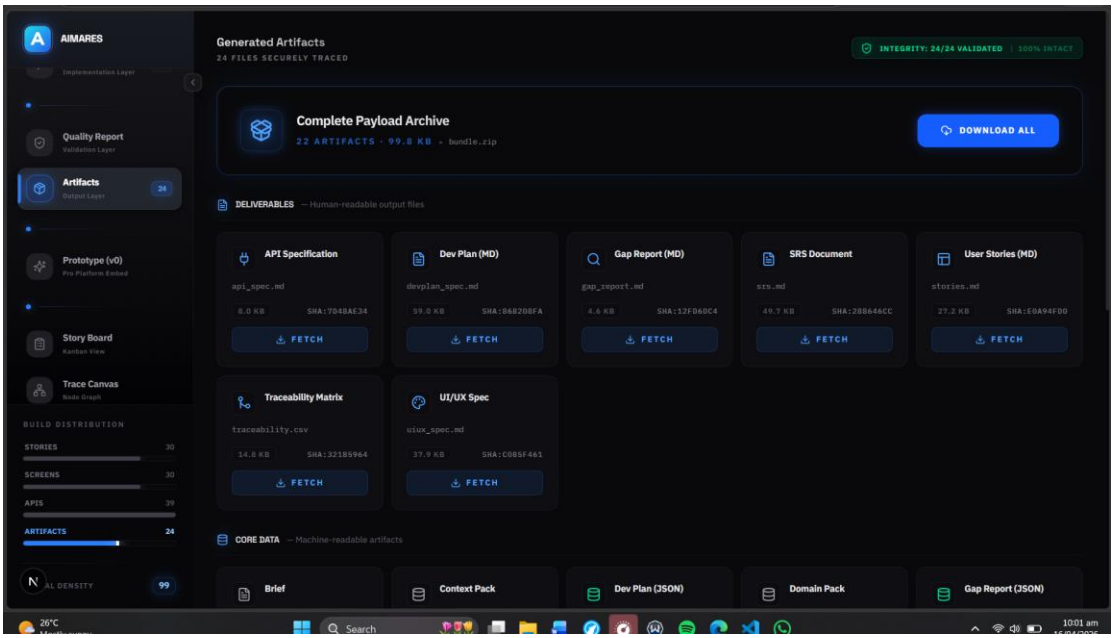


Figure 44 Artifacts

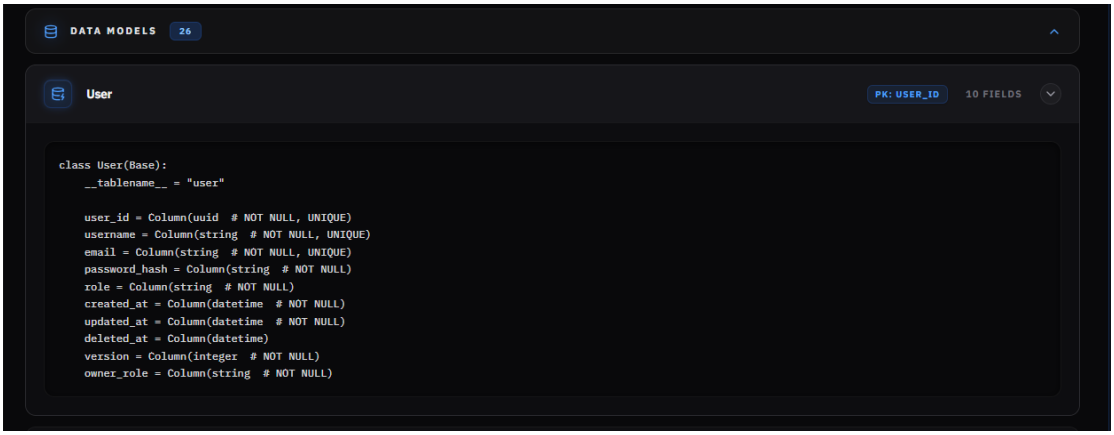


Figure 45 Data Model

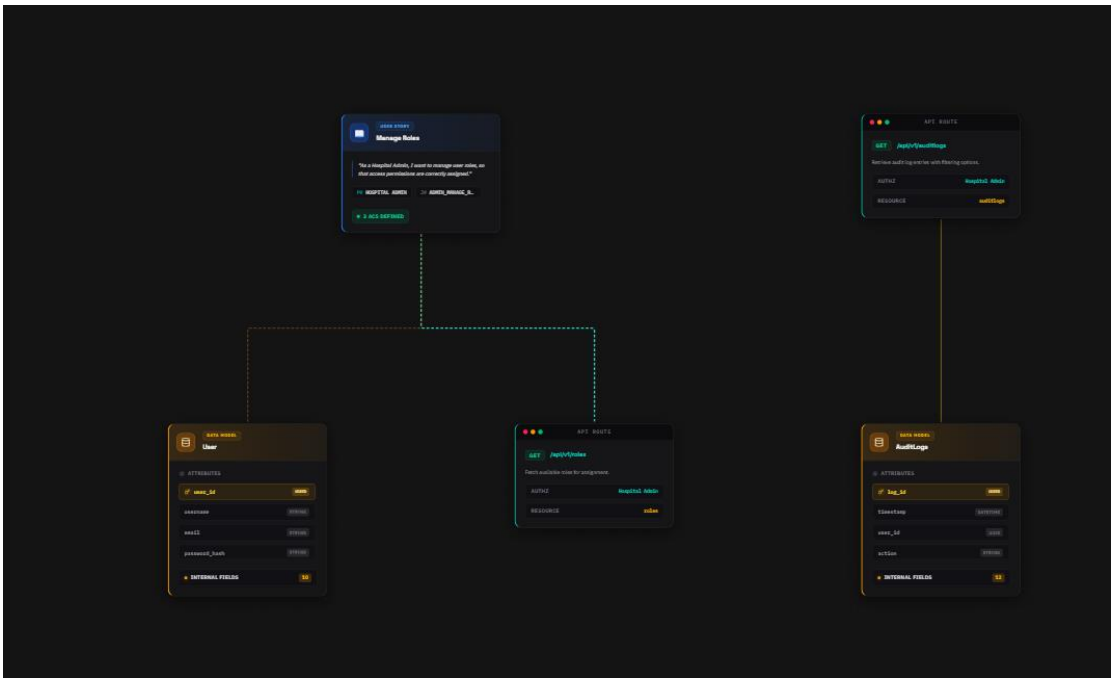


Figure 46 Traceability

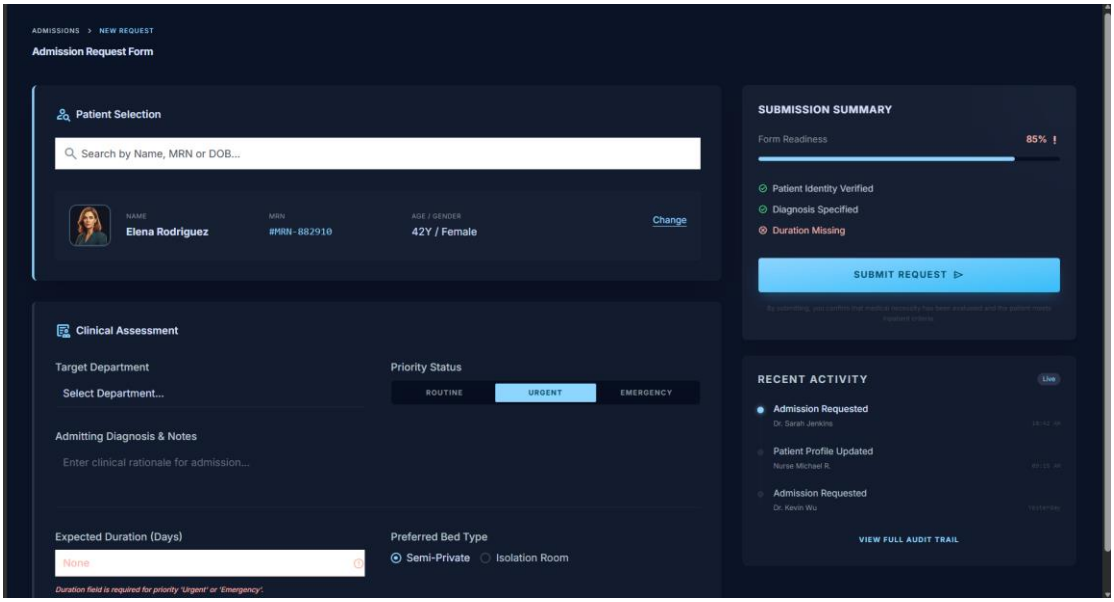


Figure 47 Generated UI Screen

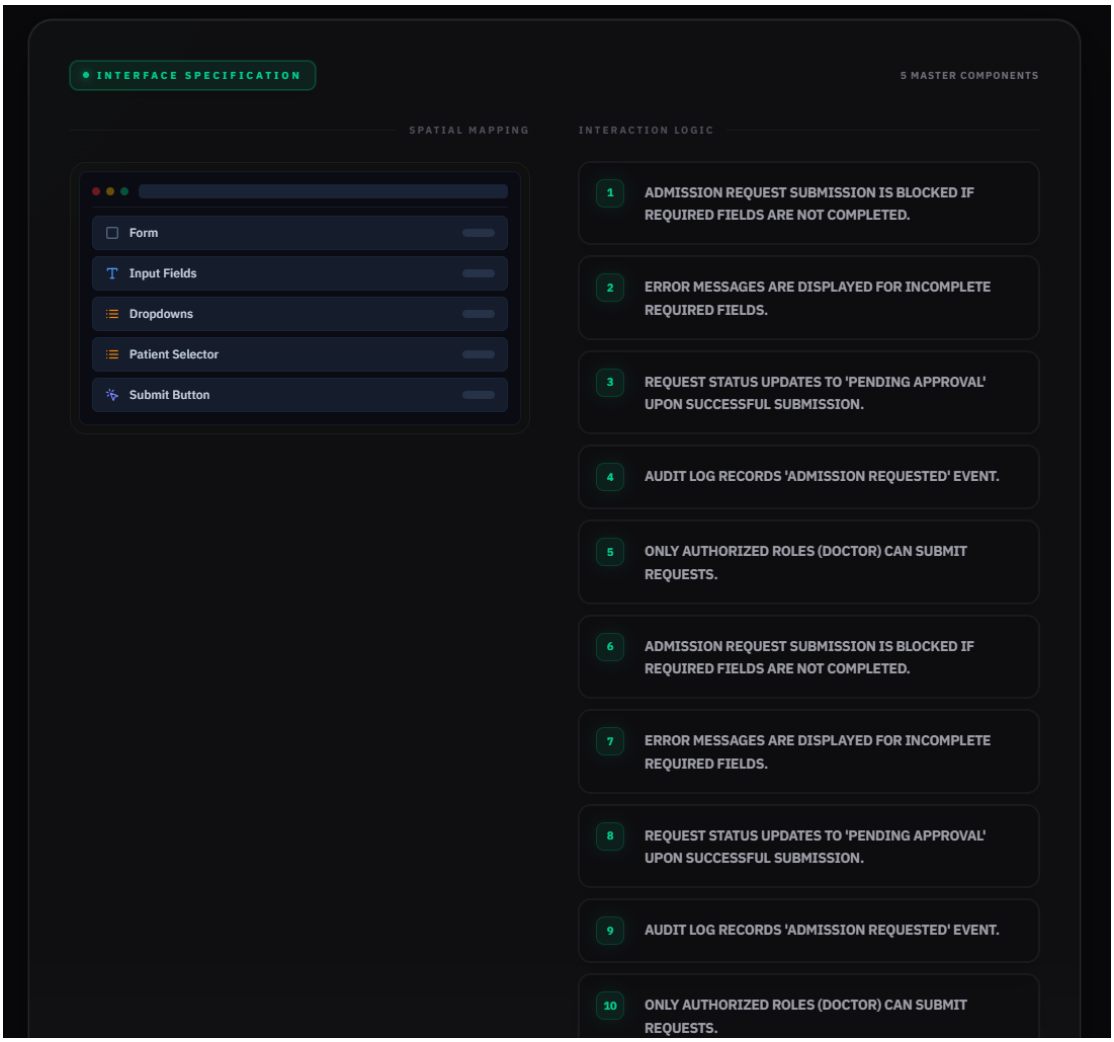


Figure 48 UI Screen Guidelines

4.9. System Prototype

A prototype of the AIMARES project was developed using a high fidelity approach, which mimics the end product quite accurately. In the process of developing such a prototype, it was essential to visualize the interaction of the user with the interface as well as the workflow for multi-agents in general.

Initially, we relied on low fidelity prototypes to sketch the basic framework of the solution by paying particular attention to the initial intake of the project phase and further stages of the whole pipeline. Thanks to these steps, it became possible to define precisely what inputs were needed from the process itself. Soon enough, we transitioned to higher levels of fidelity in prototype building.

The sophisticated prototype shows important screens in the interface, including an intake dashboard screen, which is the place where all initial briefs of the projects are being sent, as well as the pipeline tracker screen showing the work of the agents involved. There are also screens that show outputs and where users can see their stories and interface designs.

In general, this type of prototype development made the solution incredibly intuitive and interactive.

4.10. Conclusion

This section, therefore, has looked at all aspects of AIMARES architectural design ranging from its basic design approach to the structural approach, data flow, and modeling of the system. We have been able to achieve modularity, control, and traceability by designing the system as a rigid multi-agents pipeline approach, an approach that would not be possible for single prompts.

By designing the multi-layers of the system with the work process in mind, we are certain that everything that will happen when the system is running will focus on one particular task. Furthermore, with the examination of our prototype model, we show that such complex back end activities can easily be combined into an efficient user friendly interface.

Indeed, these designs form a strong foundation upon which our actual system is going to be implemented in the subsequent section.

Chapter 5

System Implementation

In this chapter, the development process of the AIMARES system will be described. In particular, the software development tools that were employed, the program code architecture, and the techniques used for the development of each of the system modules will be described in detail. The system was designed as a deterministic pipeline with multiple agents that transformed the initial idea of the system into requirements.

5.1. Technology Stack

The technology stack was developed using a variety of technologies from the frontend, backend, database, and AI sectors. This approach was adopted with the aim of achieving optimal performance, scalability, and ease of development.

Component	Technology	Purpose
Backend Framework	FastAPI	Handles API requests and orchestrates workflow
Programming Language	Python	Core backend logic
Core backend logic	Next.js (React AND TypeScript)	User interface and interaction
Database	PostgreSQL (Supabase)	Persistent storage
Retrieval System	FAISS AND Sentence Transformers	Context retrieval (RAG)
ORM	SQLAlchemy	Database interaction

AI Model	Gemini	Requirement generation
Agent Framework	MAF	Multi-agent orchestration
Storage	Supabase Storage	Artifact storage
Testing	Pytest	Backend testing

Table 20 Technology Stack

5.2. System Architecture Implementation

The AIMARES model is structured as a logical and deterministic flow of steps. The whole process begins when the project brief is sent to the model from the user via the front end. This brief is processed by the back end API, which routes the request to the pipeline within the orchestration process. With the help of a systematic series of processes, the output from each process is systematically organized to be used in the next step.

Main Workflow:

1. User specifies project;
2. Identification of system context using Domain Detection Approach (DDA);
3. Retrieval of useful data through RAG approach;
4. Creation of output results based on agents' pipeline;
5. Validation process to ensure output results are verified;
6. Improvement phase;
7. Compilation and storage of output.

One of the main characteristics of the proposed solution is the validation process which guarantees that output results are verified and corrected several times.

5.2.1 RAG Implementation

The reason is to make sure that the output data still remain relevant to the right context. The vector similarity search process is done via the embedding method all-MiniLM-L6-v2, while the database indexing technology used is FAISS. We keep and index certain kinds of data, such as requirements already made by ourselves. When there is an input query, the query is converted into embeddings and then searched inside the vector database. Following the searching process, the output is collected and fed to the agent pipeline. Thus, the output data will be influenced not only by the trained model's processes.

The retrieval process consists of the following steps:

1. Top-K Similarity Search: Searching for the most appropriate data samples in the vector database.

2. Domain-based Filtering: Filtering out the results on the basis of the specific field of the project.

3. Context Injection: Injecting the obtained context into the agent's pipeline during each crucial stage.

This systematic retrieval technique greatly enhances both the quality and consistency of generated requirements

.5.2.2 Agent Implementation

Multi-agent architecture is used in the design of this system, whereby each agent assigned specific responsibilities in the process pipeline. In such a case, the whole system will be easily understandable because each process in the system will have different responsibilities based on the goals that they need to achieve.

In such a scenario, tasks overlap does not occur.

Agent	Input	Output	Purpose
System Analyst	Brief + RAG context	Context Pack	Defines system understanding
Business Analyst	Context Pack	User Stories	Generates requirements
UIUX Agent	Stories	UI Specifications	Designs interface
LeadDev Agent	Stories + UI	Development Plan	Defines technical implementation

Validator Agent	Outputs	Quality Report	Evaluates quality
Gap Agent	Reports	Updated Output	Refines results

Table 21 Agents

Agents operate sequentially and produce structured outputs, which are passed to subsequent stages.

5.2.3 Key Feature Implementation

Unique Features of the AIMARES System

The AIMARES system has some unique features that distinguish it from other systems powered by artificial intelligence:

1. Pipeline with Agents: Instead of performing all tasks using only one process, various tasks are allocated to specific agents. Each element of the workflow receives sufficient attention in this system to ensure high-quality performance.

2. Deterministic Order: The order of the entire process is deterministic in nature. It means that identical requests will yield identical results and do not require the randomness typical of AI chatbots.

3. Validation and Corrections: All products created using this system are checked for compliance with quality standards. When problems are detected, corrections

are performed automatically in a loop until the appropriate level of quality is reached.

4. Traceability: The process from initial request to final output can be traced at any level of detail. It makes the process suitable for professional purposes, since it is possible to easily detect errors and fix them.

5. Saving Artifacts: Artifacts produced during the process are saved to the database to facilitate easy exporting to development environments.

5.2.4 API Design

The system exposes several API endpoints to manage workflow execution and data retrieval.

Endpoint	Method	Purpose
/maf/hello	POST	Initiates workflow
/maf/checkpoints	GET	Retrieves progress
/maf/threads	GET	Fetches agent outputs
/maf/download	GET	Downloads results
/maf/replay	POST	Re-executes workflow

Table 22 API Design

5.3. Conclusion

The technical foundation of the AIMARES framework was presented in this chapter. It includes the technology stack, architecture, and work pipeline. The fusion of multi-agent orchestration, retrieval-augmented grounding, and automatic validation gives us the opportunity to develop the pipeline transforming abstract ideas into accurate requirements. As can be seen from the proof-of-concept, AI is actually an efficient means of requirement engineering if applied in a deterministic manner.

Chapter 6

System Testing & Evaluation

In the current chapter, the procedures we used to test AIMARES will be outlined. In order to make sure that everything works properly, we designed an appropriate testing plan. In addition to that, one of our goals was to guarantee quality of all output produced by the system. In this regard, we used different approaches, such as component testing, unit testing, integration testing, and finally system testing Test Strategy:

It is worth noting that the test strategy for the implementation of AIMARES has been multi-level. At the first level, each module underwent a separate test prior to being integrated for execution of the complete workflow process. This was essential since the structure of the system involved a number of interrelated elements that needed to be validated as a whole. These elements mainly comprised of domain identification, data acquisition, agents cooperation, validation and correction, as well as integration and storage with third parties.

Some of the key tests that were considered in our process were as follows:

- Unit The unit tests were conducted to check whether the particular function, constraints, and algorithms worked properly
- Component tests were conducted to help test all the various components such as DDA, RAG, validation, and composition
- Integration tests were conducted to check whether all the various stages of the pipeline work effectively.
- System tests were conducted to analyze the system through the entire process starting from the submission of the brief until the downloads of the output.
- Manual Manually, the tests were also conducted to test the user interface, where you find faults by trial and error

6.1. Component Testing

Component Before integrating the whole system and carrying out tests, separate testing of selected backend services was done. The objective was to ensure that each subsystem could correctly perform its functions and generate output in proper formats.

- Components tested during the process include:
- MAF orchestration services and run lifecycle endpoints.
- DDA and RAG grounding services.
- Agent services used by roles such as System Analyst, Business Analyst, UI/UX, LeadDev, Validator, and Gap Agent.

Component	Validation Focus	**Evidence Source**
DDA	Domain detection, journey/persona preparation	DDA detector, factory, and API tests
RAG	Index loading, pinned hash behavior, filtered retrieval	RAG provider and index tests
System Analyst	Context pack generation and schema stability	AF shadow/parity tests
UIUX	Screen specifications and UI contracts	AF shadow/parity tests
Business Analyst	Story generation shape and normalization	AF shadow/parity tests
LeadDev	API, data model, and task plan generation	AF shadow/parity tests

Validator	INVEST/ISO scoring, ambiguity, coverage, and gating	INVEST/ISO scoring, ambiguity, coverage, and gating
Gap Agent	Patch behavior and dedupe-related fixes	Gap patch and dedupe tests
Composer	Manifest, checksum, and artifact bundle generation	Composer and manifest tests

Table 23 Testing Scope

6.2. Unit Testing

Unit testing was done to ensure that individual functions and formulas were correct. Given the fact that AIMARES uses a complex logic system for performing activities like data extraction and calculating scores, unit testing was necessary to ensure that everything worked properly.

- The following factors were checked in particular:
- Filtering and ranking of information.
- Logic of identifying gaps in the requirements.
- Methods of detecting duplicates and clustering them.
- Quality scoring based on INVEST and ISO standards.

Category	Typical Assertions
Validation Rules	Correct pass, warn, and fail classification at threshold boundaries

Coverage Logic	Missing journeys, roles, and NFR categories detected correctly
Dedupe Logic	Duplicate candidates and merge decisions remain stable

Table 24 Unit test

6.3. Integrated Testing

Tests for integration were carried out to ascertain whether the software modules exchange information between themselves while accomplishing a certain task. The integration test is of great significance since the output from one module becomes the input for the next process.

Integration Path	**Validation Goal**
Intake to /maf/hello to run initialization	Verify run and session creation behavior
DDA to RAG to System Analyst	Verify grounded context creation
System Analyst to Business Analyst	Verify context-to-story contract compatibility
Business Analyst to Validate to Gap to Validate	Verify quality loop behavior
Validate to Compose to Storage	Verify artifact persistence and metadata
Run checkpoints to Replay/Resume	Verify crash recovery and deterministic replay

Story outputs to Jira pull	Verify issue creation flow and mapping
UI contract to Stitch async generation	Verify async polling and result retrieval

Table 25 Integrated Testing Paths

6.4. System Testing

System testing involved testing the entire process starting from when a user initiated posting of the summary up to when the project was fully implemented. For this process, the system was taken as one unit and not component-by-component.

Some of the most critical processes that took place during this phase were as follows:

- Initiation of the process by going through the route '/maf/hello'.
- Completion of the entire process through routes beginning from start up to finish on the output nodes.
- Downloads of outputs and packages done through the correct routes.
- Application of debugging methods used for analyzing various stages of the process as well as the work being done by agents.
- Resuming the process right from where it stopped after an interruption occurred during the process.
- Exporting of results to Jira to ensure that everything was in place before export. Testing the connection was equally important.
- Creating of user interfaces and analysis of performance using Stitch tool.

6.5. Test Cases

Below is a list of test cases that we have written in a structured manner. Just like the tabular method used in the sample report, each test case contains metadata, preconditions, actions, and anticipated outcomes. This makes the writing process

much easier for the testing section. It also proves that our testing procedure was more systematic than arbitrary.

6.6.1 Test Case#1 Run Initialization and Session Creation

Test Scenario ID	**TS-001**	**Test Case ID**	**TC-001**
Test Case Description	Verify run initialization and pipeline start via /maf/hello		
Test Priority	High		
Prerequisite	API running, USE_MAF enabled		
Post-Requisite	run_id and session_id available; run enters active state		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Submit valid brief	Rich brief JSON	Request accepted
2	Check response envelope	API response body	run_id and session_id returned

3	Verify run persistence	The run_id	Run record exists in database
4	Verify execution start	The run_id	Initial checkpoints and events begin appearing

Table 26 Test Case 1

6.6.2 Test Case#2 Checkpoint Listing and Order Validation

Test Scenario ID	**TS-002**	**Test Case ID**	**TC-002**
Test Case Description	Verify checkpoint listing and order correctness		
Test Priority	High		
Prerequisite	Existing run with completed or partial execution		
Post-Requisite	Ordered checkpoint history visible for audit and debugging		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Call checkpoint endpoint	The run_id	API returns checkpoint list
2	Validate ordering	idx sequence	Checkpoints sorted in ascending order

3	Validate metadata	node_id, state hash, timestamp	Required fields present
4	Validate count	result count	Count matches stored checkpoint rows

Table 27 Test Case 2

6.6.3 RAG Retrieval and Domain Filtering

Test Scenario ID	**TS-003**	**Test Case ID**	**TC-003**
Test Case Description	Verify RAG retrieval with domain-constrained filtering		
Test Priority	High		
Prerequisite	FAISS index built and available for selected domain		
Post-Requisite	Relevant hits returned with manifest identity preserved		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Trigger retrieval query	query + filters	Search executes without error
2	Validate domain filtering	domain filter	Returned hits match canonical domain

3	Validate hit structure	hit payload	IDs, title, score, and hash metadata available
4	Validate top-k	k value	Hit count limited by k

Table 28 Test Case 3

6.6.4 System Analyst Context Pack Generation

Test Scenario ID	**TS-004**	**Test Case ID**	**TC-004**
Test Case Description	Verify System Analyst context generation		
Test Priority	High		
Prerequisite	DDA and RAG stages completed		
Post-Requisite	Context pack generated with goals, personas, and journeys		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Execute System Analyst node	brief + domain + rag	Node starts successfully
2	Validate output contract	context pack	Required fields populated
3	Validate persona consistency	key personas	Personas align with canonical role set

4	Validate grounding	rag references	Evidence-linked context produced

Table 29 Test Case 4

6.6.5 Business Analyst Story Generation

Test Scenario ID	**TS-005**	**Test Case ID**	**TC-005**
Test Case Description	Verify story generation by Business Analyst		
Test Priority	High		
Prerequisite	Valid context pack available		
Post-Requisite	Story set generated with IDs, acceptance criteria, journey, and persona		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Execute Business Analyst node	context pack	Story generation starts
2	Validate story schema	story set	Required fields present

3	Validate acceptance criteria quality floor	acceptance criteria	AC list present and non-empty
4	Validate trace hints	story metadata	Trace links available for downstream use

Table 30 Test Case 5

6.6.6 A Validator Scoring and Routing Decision

Test Scenario ID	**TS-006**	**Test Case ID**	**TC-006**
Test Case Description	Verify validation scoring and gate decision		
Test Priority	High		
Prerequisite	Story set and supporting packs available		
Post-Requisite	Quality report and gap report generated with routing decision		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Execute Validate node	stories + context	Quality analysis completes

2	Check scoring	INVEST and ISO values	Global summary generated
3	Check coverage findings	gap report	Missing coverage identified if present
4	Check routing state	decision	Route set to compose or gap_apply

Table 31 Test Case 6

6.6.7 Gap Remediation and Re-Validation Loop

Test Scenario ID	**TS-007**	**Test Case ID**	**TC-007**
Test Case Description	Verify gap patch application and re-validation loop		
Test Priority	High		
Prerequisite	Validate node routes to gap_apply		
Post-Requisite	Stories patched and iteration counter updated		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Stories patched and iteration counter updated	gap report + stories	Patch actions applied
2	Validate story updates	updated stories	Gap-related fixes visible
3	Verify loop increment	iter_no	Iteration count increments

4	Verify re-entry	routing	routing

Table 32 Test Case 7

6.6.8 Compose and Artifact Bundle Generation

Test Scenario ID	**TS-008**	**Test Case ID**	**TC-008**
Test Case Description	Verify compose and artifact bundle generation		
Test Priority	High		
Prerequisite	Validation finished with compose decision		
Post-Requisite	Artifacts stored and bundle metadata retrievable		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Execute Compose node	final run state	Compose succeeds
2	Verify artifact list	artifact records	Core artifacts exist
3	Verify manifest and checksums	bundle metadata	Consistency validation passes

4	Verify download endpoint	run_id	Signed links returned

Table 33 Test Case 8

6.6.9 Jira Export Workflow

Test Scenario ID	**TS-009**	**Test Case ID**	**TC-009**
Test Case Description	Verify Jira export workflow including status, dry-run, and pull		
Test Priority	High		
Prerequisite	Export-eligible run and Jira configuration enabled		
Post-Requisite	Jira report and mapping artifacts generated		
S. No	**Action**	**Inputs**	**Expected Outcome**
1	Jira report and mapping artifacts generated	run_id	Eligibility and block reasons returned
2	Trigger dry-run	run_id	Create/skip plan returned

3	Trigger pull	run_id	Stories pushed and counts returned
4	Verify artifacts	report/mapping links	Jira artifacts available for traceability

Table 34 Test Case 9

6.6.10 Stitch Async Job Lifecycle

Test Scenario ID	**TS-010**	**Test Case ID**	**TC-010**
Test Case Description	Verify Stitch async lifecycle and polling behavior		
Test Priority	Medium		
Prerequisite	Stitch API key configured; generate-screen endpoint enabled		
Post-Requisite	Job reaches terminal COMPLETE or FAILED with retrievable status		
S. No	**Action**	**Inputs**	**Expected Outcome**

1	Trigger generate-screen endpoint	screen prompt payload	job_id returned with IN_PROGRESS
2	Poll screen-status endpoint	job_id	Status transitions observed
3	Validate terminal state	COMPLETE or FAILED	Terminal state eventually reached
4	Validate output fields	screenshot_url/html_url/status_message	Correct fields provided for terminal state

Table 35 Test Case 10

6.6. Results & Evaluation

The process of testing involved several steps: unit testing, component testing, integration testing, and systems testing with the application of benchmarks in order to prove the reproducibility of all results. There was only one key objective for the testing process, which was to check the correctness of functions, determinism of workflow, and integration processes. It turned out that everything works absolutely fine starting from the beginning of the program till bundling phase and recovery procedure. Testing on the backend proved the outstanding stability of the tested system. Apart from usual testing procedures, benchmarking tests were performed to check the correct start, order

of checkpoints, resumption, replay processes, and integrity of the bundles. The results showed that 57 tests were successfully passed, while three others were skipped because of some server-side requirements. However, there were still some constraints:

- Certain tests are avoided when working offline or under controlled circumstances.
- Tests of the front-end are less automated compared to those of the back-end.
- The reaction times of the external suppliers may at times affect the uniformity of the timing process

6.7. Conclusion

The process of validation and evaluation of the system will be considered within this chapter, where the methodology that is based on facts is utilized. That was done by means of analyzing the AIMARES system at different levels from the smallest logical parts up to the entire flow of its execution. It is evident that, as a result of the performed analysis, the system proves to be stable and to operate in accordance with the intended specifications. In addition, it has been shown that AIMARES successfully interacts with other tools such as Jira and Stitch.

Chapter 7
Conclusion

The multi-agent system AIMARES is designed in this research project. The multi-agent system utilizes AI techniques to convert an initial idea into a complete requirement specification. It is important to note that this research project solves a real-life problem within the field of software engineering. Project ideas are usually too concise for developers to start developing their projects.

Unlike other approaches that use just a simple AI prompt, there was a reliable procedure applied in the case of AIMARES. There were a number of phases involved in the process, and those were information gathering, domain identification, and grounding data to make sure that there are no errors. The tool is also capable of managing various specialized agents and conducting error checking. This allows AIMARES to generate outputs like user stories, UI/UX designs, and roadmaps.

In our work, it is shown that AI works much more effectively when used within a well-designed process, especially with quality assurance. Such elements as data grounding and restarts make the technology more reliable. In addition to the proposed prototype, this project provides a roadmap for building a reliable system.

7.1. Contributions

The contributions of this project can be understood in relation to the original objectives defined in Chapter 1.

1. First, To start with, the accomplishment within the project has been realized through the creation of the deterministic multi-agent requirements engineering pipeline.The process aligns with the overall aim of creating a system that not only performs text generation but systematically engineering processes based on process control processes. Firstly, the project was able to successfully create a deterministic multi-agent requirements engineering pipeline. The effort fulfills the general aim of creating a system that is not just capable of generating texts in one go but doing systematic engineering following a controlled process.of stages.
2. Secondly, there was the application of retrieval augmented generation (RAG), making the entire exercise more realistic.By being able to refer to real data

before applying artificial intelligence, the results were not very influenced by the fact that the model produced random outputs. In the end, the results were extremely reliable and consistent, which is crucial because it connects advanced artificial intelligence to engineering practices.

3. Thirdly: Specific Agent FunctionsThe project used specific agents like System Analyst, Business Analyst, UI/UX, Lead Dev, Validator, and Gap Agent among others. Such an arrangement improved the modularity of the design and made large engineering operations possible through breaking down into smaller sub-operations. Such an approach made it easier to test and extend the system.

4. Four: The Process Validation and Gap Fixation LoopApart from that, another innovation introduced in the project included the process validation and gap fixation loop. While use of results based on AI results involves no checks, this technique implies checking the results based on quality criteria and gaps. At the final stage, after necessary corrections have been made, the output is finalized and delivered to the consumer. One should note that this is a critical phase because it resembles practical engineering tasks.

5. Point Five: Traceability and Robustness In order to make the system more traceable and robust, more functions have been added to the system. There is now an array of functionalities in the system including checkpoints, replay, resume, log based on manifest and results package which you can download. This helps tremendously improve the overall system when applied academically and practically since it makes tracing the whole creation process of each output easy.

Finally, this work is a contribution to AI-assisted software engineering. As shown by this investigation, LLMs can achieve outstanding results by utilizing deterministic orchestration, evidence grounding, and validation. Moreover, this paper shows that it is critical to consider artifact-level traceability as well. Thanks to all these features, AIMARES is not just a prototype of requirements generation but an architecture for requirements generation as well.generation.

7.2. Reflections

The research managed to discover various interesting features, both from a technical point of view and in terms of science.

The main advantage of research like this one is the ability to treat requirement engineering as a process rather than something that only generates text in response to a prompt, which many of the existing approaches are capable of doing at best. However, as this study has proven, analyzing, generating, verifying, and modifying texts leads to much more coherent and logical output. This research approach proves extremely handy in the case of requirement engineering because of inconsistencies in requirements themselves.

In addition, through the project, we learned the importance of using certain agents in the project. The segregation of duties made sure that everyone knew what they were supposed to do. Moreover, testing became easy because everything could be done systematically. It was important that not only did each part work well on its own, but also matched the next step in the pipeline.

However, there are some downsides to be observed in this regard. For one thing, as was already pointed out earlier, the efficiency of the entire solution is largely based on the efficiency of external AI algorithms. While the efficiency of the process of workflow control in this particular case can be ensured, it does not mean that the results would be absolutely stable in their nature when using third-party LLM-based services. Moreover, as was also noted before, the performance of the algorithm itself is more effective when working with indexed data sources, and, therefore, its efficiency when working with unsupported domains might be somewhat lower.

Considering the total extent of the topic, the article demonstrates that artificial intelligence can be applied successfully when developing software products at their initial stages. The project offers valuable data for scholars and small groups of people who have to elaborate on structured requirements beginning from an idea. Hence, one can conclude that the study is practically significant in the field of software engineering and requirement analysis. Moreover, the research makes a valuable contribution to the scientific debate concerning software engineering based on large language models.

7.3. Future work

There are several directions in which this work can be extended.

7.3.1 Human in the loop review mechanism:

A good follow-up could be the inclusion of a more advanced human-in-the-loop reviewing stage.

While it is true that the current model already includes a validation and editing stage, future models could include granting individuals involved in the project such as managers and analysts the capability of reviewing and editing the output data that is produced by the machine.

7.3.2 Domain coverage and retrieval quality:

The second improvement which can be made is in.

Presently, the effectiveness of the process of grounding is contingent on having the indexed corpus available for a specific domain. In the coming time, it is possible to have the number of domains increased and the process of building the corpus improved too.

7.3.3 Reliability in AI-assisted requirements engineering:

With regards to future studies, it is essential to pay particular attention to the reliability of AI-based requirements engineering. In the first place, there is no doubt that the existence of such a methodology is required, which will allow assessing the quality of the resulting artifacts. Secondly, it is necessary to find a way to combine reproducibility with creativity. Lastly, it is vital to guarantee consistency in the application of AI in the process irrespective of the software

REFERENCES

- Are prompts all you need? Evaluating prompt-based Large Language Models (LLM)s for software requirements classification | Requirements Engineering | Springer Nature Link

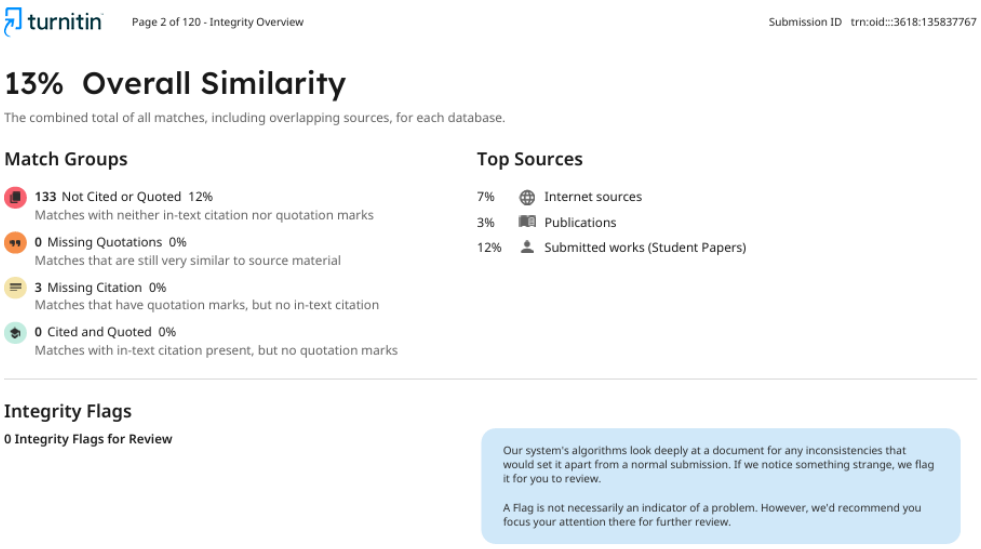
- A design science research approach to Large Language Model-Based Agents for Requirements Specification (LLMBA4RS) in low-code applications | Requirements Engineering | Springer Nature Link

- RM4ML: requirements model for machine learning-enabled software systems | Requirements Engineering | Springer Nature Link

- https://nodejs.org/en

- Next.js by Vercel - The React Framework

APPENDICES A



turnitin Page 2 of 120 - Integrity Overview Submission ID trn:oid:::3618:135837767

13% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- **133** Not Cited or Quoted 12%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **3** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

7% Internet sources
3% Publications
12% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Higher Education Commission Pakistan on 2025-05-14		1%
2	Internet	
prr.hec.gov.pk		<1%
3	Student papers	
RMIT University on 2024-09-21		<1%
4	Student papers	
Institute of Research & Postgraduate Studies, Universiti Kuala Lumpur on 2025-0...		<1%
5	Student papers	
SSBAS on 2026-04-11		<1%
6	Internet	
arxiv.org		<1%
7	Internet	
irigs.iiu.edu.pk:64447		<1%
8	Internet	
www.coursehero.com		<1%
9	Student papers	
University of Greenwich on 2016-05-19		<1%
10	Student papers	
Islington College,Nepal on 2025-12-30		<1%

11 Student papers
University of Southampton on 2011-09-26 <1%

12 Student papers
October University for Modern Sciences and Arts (MSA) on 2024-06-23 <1%

13 Internet
ir.unimas.my <1%

14 Student papers
Abo Akademi University on 2026-04-16 <1%

15 Student papers
Massey University on 2012-09-20 <1%

16 Student papers
Higher Education Commission Pakistan on 2020-12-20 <1%

17 Publication
Manal Binkhonain, Reem Alfayez. "Are prompts all you need? Evaluating prompt-... <1%

18 Student papers
University of Greenwich on 2013-02-27 <1%

19 Student papers
University of Westminster on 2025-04-07 <1%

20 Student papers
University of South Africa on 2026-01-15 <1%

21 Student papers
University of Westminster on 2025-11-13 <1%

22 Student papers
HELP UNIVERSITY on 2017-01-17 <1%

23 Student papers
Higher Education Commission Pakistan on 2015-05-21 <1%

24 Student papers
Higher Education Commission Pakistan on 2020-12-23 <1%

25	Internet	
discovery-pp.ucl.ac.uk		<1%
26	Student papers	
Asia Pacific Instutute of Information Technology on 2009-01-13		<1%
27	Student papers	
Universiti Malaysia Sabah on 2026-01-09		<1%
28	Student papers	
Zimbabwe Ezekiel Guti University on 2026-03-24		<1%
29	Student papers	
Singapore Institute of Technology on 2026-03-20		<1%
30	Student papers	
Swinburne University of Technology on 2025-09-16		<1%
31	Student papers	
University of East London on 2018-04-29		<1%
32	Internet	
eprints.utem.edu.my		<1%
33	Internet	
www.scribd.com		<1%
34	Student papers	
The American College of Greece-Learn SaaS on 2025-12-02		<1%
35	Publication	
Cristian Rotar, Qingyu Zhang. "A design science research approach to Large Lang...		<1%
36	Student papers	
Coventry University on 2008-09-12		<1%
37	Student papers	
Dublin Business School on 2025-08-28		<1%
38	Student papers	
Midlands State University on 2019-04-20		<1%

39 Student papers
Nottingham Trent University on 2022-09-02 <1%

40 Internet
www.diva-portal.se <1%

41 Student papers
Islington College,Nepal on 2025-12-29 <1%

42 Student papers
Manchester Metropolitan University on 2010-10-07 <1%

43 Student papers
University of Mauritius on 2026-03-27 <1%

44 Internet
www.mdpi.com <1%

45 Student papers
American University in Cairo on 2026-04-14 <1%

46 Student papers
CITY College, Affiliated Institute of the University of Sheffield on 2009-05-29 <1%

47 Student papers
Islington College,Nepal on 2025-12-30 <1%

48 Student papers
Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA) on 2026-01-11 <1%

49 Publication
Yuanzhe Wang, Danwei Wang. "Collaborative Fleet Maneuvering for Multiple Aut... <1%

50 Internet
ir.aiktclibrary.org:8080 <1%

51 Internet
uir.unisa.ac.za <1%

52 Publication
"The Impact of the 4th Industrial Revolution on Engineering Education", Springer ... <1%

53 Student papers
Asia Pacific Instutute of Information Technology on 2025-09-14 <1%

54 Student papers
Birzeit University Main Library on 2026-01-26 <1%

55 Student papers
Gulf College Oman on 2018-06-20 <1%

56 Student papers
Letterkenny Institute of Technology on 2018-05-04 <1%

57 Student papers
University of Greenwich on 2023-11-28 <1%

58 Internet
dspace.lib.buu.ac.th <1%

59 Student papers
Ain Shams University on 2020-09-05 <1%

60 Publication
Arvind Dagur, Sohit Agarwal, Dhirendra Kumar Shukla, Shabir Ali, Sandhya Sharm... <1%

61 Student papers
CSU, San Jose State University on 2014-04-18 <1%

62 Student papers
Higher Education Commission Pakistan on 2011-09-20 <1%

63 Student papers
Islington College,Nepal on 2026-04-11 <1%

64 Student papers
Liverpool John Moores University on 2026-01-16 <1%

65 Student papers
Midlands State University on 2014-10-13 <1%

66 Student papers
National University of Modern Languages on 2026-02-28 <1%

67 Student papers
Ton Duc Thang University on 2025-02-17 <1%

68 Student papers
University of Greenwich on 2010-12-31 <1%

69 Student papers
University of Greenwich on 2016-05-19 <1%

70 Internet
idoc.tips <1%

71 Student papers
FPT Academy International HoChiMinh City on 2020-05-06 <1%

72 Student papers
Higher Education Commission Pakistan on 2015-08-10 <1%

73 Student papers
KMD Institute (KMD002) on 2025-10-29 <1%

74 Student papers
Kentucky State University on 2026-01-26 <1%

75 Publication
Matias, Gonçalo Ramiro Valadao. "2D Phase Unwrapping Via Graph Cuts", Univers... <1%

76 Student papers
Middlesex University on 2014-04-04 <1%

77 Publication
Ramovha, Tshililo. "A Mobile Application for Supply Chain Coordination of Artemi... <1%

78 Student papers
The University of the West of Scotland on 2025-12-08 <1%

79 Student papers
University Tun Hussein Onn Malaysia on 2019-12-31 <1%

80 Student papers
University of Bradford on 2009-08-27 <1%

81 Student papers
University of Houston Clear Lake on 2025-10-21 <1%

82 Student papers
University of London External System on 2011-04-25 <1%

83 Student papers
University of Northampton on 2024-09-01 <1%

84 Student papers
University of Paisley on 2006-11-30 <1%

85 Student papers
University of West London on 2023-06-12 <1%

86 Student papers
University of Wolverhampton on 2021-05-23 <1%

87 Student papers
Victoria University on 2023-07-01 <1%

88 Internet
alessay.pages.dev <1%

89 Internet
home.ubalt.edu <1%

90 Internet
pdfcoffee.com <1%

91 Student papers
Higher Education Commission Pakistan on 2024-08-12 <1%

92 Student papers
Islington College,Nepal on 2025-12-30 <1%

93 Student papers
Islington College,Nepal on 2026-04-17 <1%

94 Student papers
Arab Open University on 2024-11-07 <1%

95 Student papers
Higher Education Commission Pakistan on 2018-02-26 <1%

96 Student papers
Higher Education Commission Pakistan on 2018-05-28 <1%

97 Student papers
Taibah University on 2020-04-24 <1%

98 Student papers
University of Limerick on 2020-04-30 <1%

APPENDIX B

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.