Tunisian Republic

Ministry of Higher Education

and Scientific Research

TEK-UP Private College of Engineering and Technology

TEK-UP

End-of-year internship report

Major : Information Systems And Software Engineering

By

Zenned Yassine

Intelligent Marketplace MarketAI

Professional supervisor: **Mr Aymen BelHaj**

Project conceived and elaborated within Evergates

Academic Year 2023 - 2024

Contents

List of Figures

List of Tables

General introduction

Since the advent of Artificial Intelligence (AI), it has revolutionized industries by introducing smart systems capable of performing tasks that once required human intervention. The growing integration of AI into everyday applications, from virtual assistants to automated systems, has significantly enhanced productivity, efficiency, and user experience in both personal and professional environments.

Among the many use cases, AI-powered chatbots and intelligent assistants have gained immense popularity. These systems offer instant responses, can manage large sets of data, and even make decisions based on user interactions. With the aid of Natural Language Processing (NLP) and machine learning techniques, chatbots are now becoming more context-aware and capable of handling complex conversations.

The combination of AI models like OpenAI's GPT and modern development frameworks like Spring Boot presents a promising platform for building scalable and intelligent solutions. These technologies enable not only effective user communication but also assist in various business applications such as product recommendation, personalized responses, and image-based analysis.

Our end-of-study project, completed within the framework of AI-driven software development, focuses on building an advanced assistant chatbot using Spring Boot and OpenAI. The chatbot integrates Retrieval-Augmented Generation (RAG) to fetch and provide relevant information while maintaining context history, offering a more interactive and coherent user experience.

Additionally, the project implements a product description generation feature that allows users to upload images, and the system automatically generates descriptions, highlights key features, and estimates the price, enhancing the product management process.

chapter 1 : "General Context" presents an overview of the project background, problem statement, objectives, and the proposed AI-based solution.

chapter 2 : "Analysis and Requirements Specification" outlines the project actors, the functional and non-functional requirements, and details the development sprints.

chapter 3 : "Sprint 0" covers the technical choices, tools, and frameworks, along with the creation of use case and class diagrams, laying the foundation for the development phase.

chapter 4 : "Sprint 1" focuses on the development of the assistant chatbot, detailing the implementation of Retrieval-Augmented Generation (RAG) and context history management.

chapter 5 : "Sprint 2" includes the implementation of the product description generation feature, enabling users to upload images and receive descriptions, key features, and price estimates. It also details testing and refinement efforts.

chapter 6 : "Sprint 3" involves fixing bugs from previous sprints, optimizing performance, and improving the user experience. Further enhancements are made to both the chatbot and the product description generation system, ensuring the robustness of the final deliverable.

Chapter 1

General presentation and overview

Introduction

The purpose of this introductory chapter is to put the work in its general context. We begin with a presentation of the host company. Then we describe the subject around which our graduation project is baesd. Finally, we study the concepts, technologies, solutions and principles used in our project

1.1 Study of the Existing Solutions

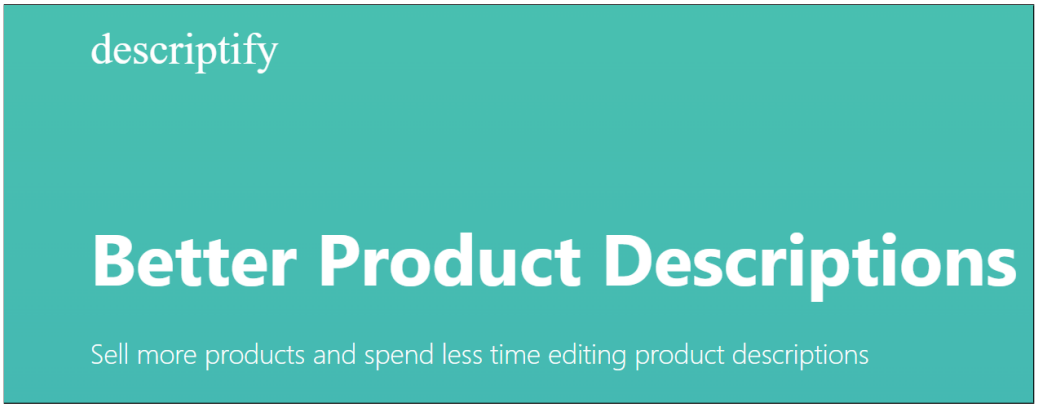

Figure 1.1: Descriptify

Descriptify is an AI tool designed specifically for generating product descriptions for e-commerce. It enables sellers to input basic product details and generates SEO-friendly descriptions based on the provided information. This tool improves the speed and consistency of description generation, helping sellers create more professional and discoverable listings. However, Descriptify lacks a chatbot or support feature to handle live queries or engage in complex conversations with users. This limitation highlights an area where additional functionality could enhance the overall user experience, especially for sellers who need both product description generation and real-time support.

Cons :

- **Design**: Limited flexibility in customization and user flow for different marketplaces.
- **Performance**: Existing AI tools lack integration for image-based product description generation, making the listing process slower and more reliant on manual data entry.
- **Features**: Many current tools provide basic description generation but lack an intelligent chatbot for real-time seller support or context-aware assistance in handling buyer queries and issues.
- **Support**: Sellers face challenges obtaining quick support from platforms, resulting in slow issue resolution for inquiries related to product listings, descriptions, and transactions.

1.2 Problematic

In the fast-growing world of online marketplaces, sellers frequently face challenges when it comes to providing clear, detailed, and accurate descriptions for the products they are listing. Many sellers either lack the time, expertise, or resources to craft well-structured descriptions that effectively highlight key features, benefits, and pricing. As a result, potential buyers are often left with incomplete or unclear information, leading to confusion, mistrust, and a lack of confidence in making purchasing decisions.

This lack of quality product descriptions directly impacts sales conversion rates and customer satisfaction. Moreover, poorly written descriptions can harm search engine optimization (SEO), making products harder to discover in search results, both on the marketplace itself and through external search engines. For smaller businesses or individuals selling items, the challenge of creating consistent, high-quality descriptions across large inventories further exacerbates the problem, often resulting in missed opportunities and reduced competitiveness in a crowded market.

In addition to these issues, many sellers face difficulties when trying to obtain support or resolve issues through the marketplace platforms themselves. Whether it is gathering more detailed information about buyer queries or dealing with problems like returns, payment delays, or delivery concerns, the lack of responsive, real-time support often leaves sellers without timely solutions. This further strains seller-buyer interactions and negatively affects both user experience and overall business performance.

Therefore, the problem lies in the need for a solution that not only automatically generates clear, accurate, and appealing product descriptions but also provides real-time support for sellers to address inquiries and resolve issues quickly, improving the quality of listings, customer trust, and overall marketplace efficiency.

1.3 Suggested solution

To address the challenges sellers face with unclear or incomplete product descriptions in online marketplaces, our AI-powered solution offers a seamless approach to automate the generation of high-quality product descriptions. By leveraging a combination of OpenAI's natural language processing capabilities and image recognition technology, our application allows sellers to upload an image of their product, which is then analyzed to generate a detailed, accurate description. The system highlights key features, suggests pricing, and ensures that descriptions are clear, professional, and optimized for search engines (SEO). Additionally, the assistant chatbot, equipped with Retrieval-Augmented Generation (RAG) and context history, enhances user interaction by answering questions, managing product details, and maintaining coherent conversations, providing a smooth, efficient experience for sellers and buyers

alike. This solution not only saves time and effort for sellers but also improves the overall quality of marketplace listings, leading to increased sales and customer trust. .

1.4 Working Methodology: Agile Scrum

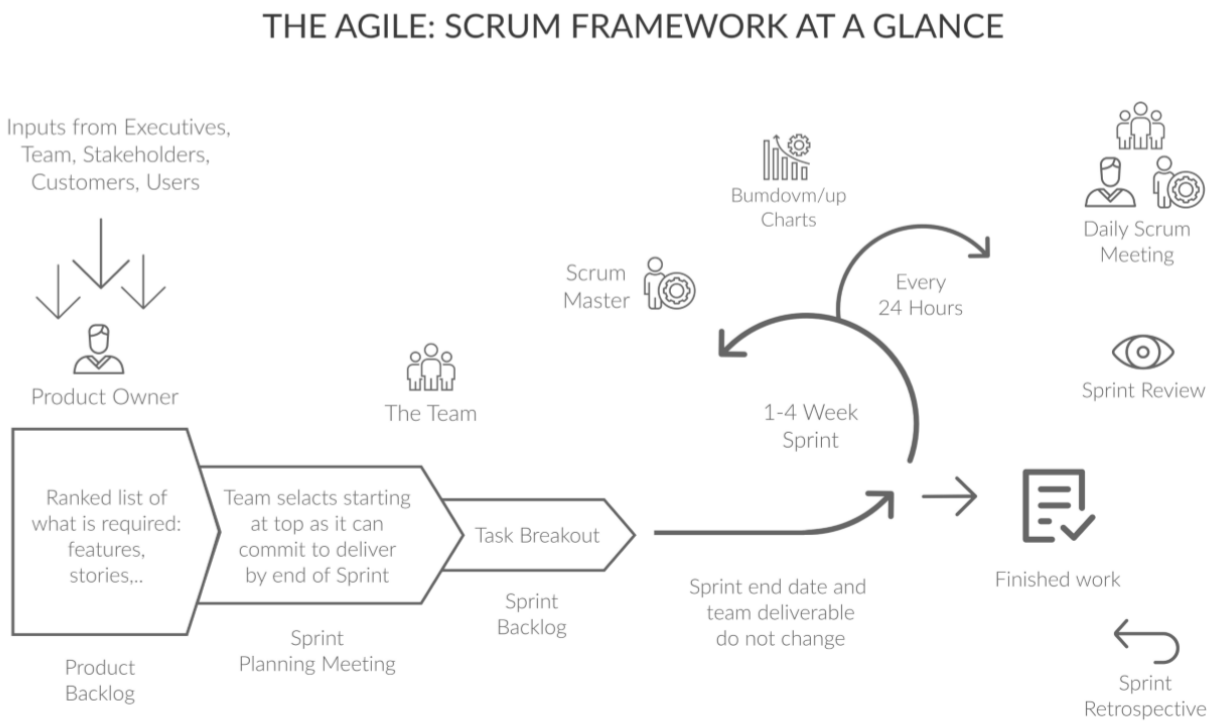

Figure 1.2: Global view of software development process Scrum

For years, most software developments have been based on methodologies called agile. This banner combines several methods based on iterative development and incremental, in which the search for solutions to problems is based on collaboration involves the client from the beginning to the end of the project. It considers that the need cannot be frozen and proposes to adapt to the latter's changes. Scrum is considered as a project management framework. This framework consists of a definition of roles, meetings and artifacts. Scrum defines 3 roles:

- **The Product Owner:** he is the customer and the user's representative, he defines the needs, the product specifications and the order in which the functions will be developed.
- **The Scrum Master:** It is the intermediary between the Product Owner and the team. It ensures the proper application of the Scrum methodology.
- **The development team:** It is the team that produces the product while respecting the pre-determined deadlines.

The life of a Scrum project is made up of a set of clearly defined meetings that are strictly limited in time.

- **Sprint Planning:** This meeting will highlight the priority elements of the list of functional requirements of the product.
- **Sprint Review:** It is held at the end of each pen- sprint. in which the developers show the Product Owner the developed functionality and collect its feedback.
- **Daily Scrum:** It allows the development team to synchronize their work. This meeting lasts no more than 15 minutes, allowing everyone to determine what they have achieved since the last Daily Scrum meeting, which is why they have been working on the project. That he has to finish before the next Daily Scrum and identify the obstacles that could block him.
- **Sprint Retrospective:** held at the end of each Sprint, allowing everyone to improve (productivity, quality, efficiency, working conditions, ..)

1.5 Architecture of the application

we are going to present the architecture of our application to better understand its functionality.

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application.

- **Model:**

Models are data classes that you create according to your needs and with the help of the controller, you can manipulate the model's data.

- **Vue:**

Views are all the Widgets and Pages within the Flutter Application. These views may contain a "view controller" themselves, but that is still considered part of the view application tier.

- **Controller**:

The controller layer is high-level functions that perform a specific type of task. They typically fetch from services and inject them into models and that way they control and update app state.

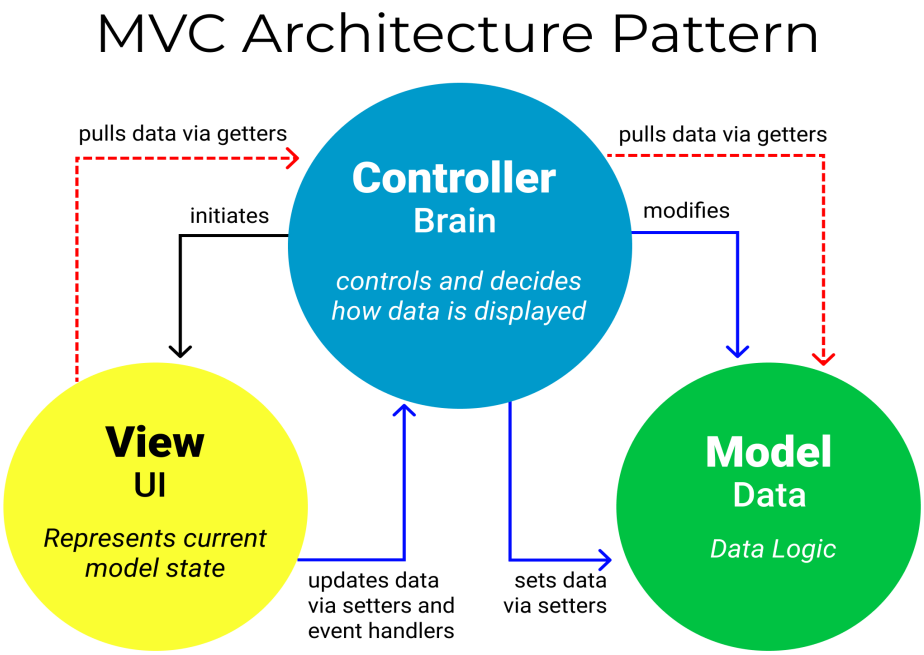

Figure 1.3: MVC

Conclusion

In conclusion, the challenges faced by sellers on online marketplaces, such as providing clear product descriptions and receiving adequate support from platforms, can significantly impact their ability to succeed in a competitive environment. These issues lead to decreased visibility, lower customer satisfaction, and missed business opportunities. The proposed AI-driven solution addresses these problems by automating the generation of high-quality product descriptions and offering real-time support for sellers. This approach not only enhances the overall efficiency of the selling process but also boosts customer trust and increases sales potential.

Chapter 2

Analyze and Requirements Specification

Introduction

The requirements specification is a crucial step in the realization of any IT project. In this chapter, we will present this concept in detail. We will start by defining the roles of each user and their primary functionalities. Then, we will identify the functional and non-functional requirements necessary to create the product backlog and initiate the first sprint planning.

2.1 Requirements Specification

In this section, we define the actors involved in the system, their roles, and the functional and non-functional requirements identified during periodic meetings with the Scrum Master and the supervisors. The purpose of this preliminary study is to clarify the functionalities and constraints of the application..

2.2 Identification of Actors

In our system, we can identify two main actors:

- **The User**: interacts with the system to generate product descriptions, receive price recommendations, check product details, and access support through the chatbot.
- **The AI**: responsible for analyzing uploaded images, generating product descriptions, recommending prices based on market trends, and assisting users with product-related queries via the chatbot.

2.3 Functional Requirements

Our main objective is to design and implement the two applications that cover all the needs for each user. So we present the system requirements classified by actor:

- **The User**:
  - Upload product images.
  - Edit existing product descriptions.
  - Request and generate product descriptions.
  - Use the chatbot for inquiries.
- **The AI**:
  - Process uploaded images for features.
  - Generate product descriptions based on images.

- Implement Retrieval-Augmented Generation (RAG) for enhanced responses.
- Maintain context history in the chatbot for better user interaction.

2.4 No Functional Requirements

Non-functional requirements correspond to the technical requirements and constraints at which our system must respond to. Our system must necessarily meet these needs:

- **Usability**: The application must be easy for the users to use.

- **Ergonomics**: The interface of the application must be ergonomic, user-friendly and simple.

- **Security**:The user must authenticate in a secure way to access his part of the application, no one but authenticated users can access the applications and the sensitive data like users passwords should be encrypted.

- **Scalability**:The system must be able to grow as needed in the future.

- **Performance**:The applications should be always available and ensure a short response time despite the number of users online.

- **Technical documentation**: The application code must be documented in order to ensure readability and comprehensibility. The documentation includes the definition and explanation of APIs, data structures and algorithms.

2.5 Project management with Scrum

In this section we describe the progress of our project throughout the period of internship.

2.5.1 Scrum tools: ScrumDesk

Project management tools with meaningful practices. Manage projects from goals to subtasks in a very transparent and instant way. Ideal for teams that prefer agile practices. Scrum or Kanban. Objectives and key results. Backlogs. Roadmaps. Kanban boards. Let your teams improve with

retrospectives and root cause analysis. Build by agile coaches. Since 2007. Trusted by 40k companies, 400 universities.

2.5.2 Product Backlog

Scrum's approach proposes to start by listing the customer's requirements to produce the product backlog as a user Stories list. This list contains everything that could be required in the product and is the only source of requirements for all modifications to bring to the product. Each backlog item is estimated in terms of development effort by the Scrum team. In our project.

The table 2.1 shows the product backlog designated for the project enumerating these attributes:

- **ID:** the unique and auto-incremented number of each story.
- **User story**: The description of the task to be realized.
- **Estimation**: the number of days allocated to each task(User story).
- **Priority**: Once The backlog is defined, the choice of the order of prioritization varies according to the project, the available resources and the objectives. We opted for the Moscow method:

  1 M - Must have : Non-negotiable product needs that are mandatory for the team.

  2 S - Should have : Important initiatives that are not vital, but add significant value.

  3 C - Could have : Nice to have initiatives that will have a small impact if left out.

  4 W - Will not have : Initiatives that are not priority for this specific time frame.

Tableau 2.1: Product's backlog.

ID	Feature	User Story	Priority	Estimation	Complexity
1	Generate Product Descriptions	As a seller, I want the AI to generate product descriptions based on uploaded images and key details.	M	10 days	High
2	Product Description Editor	As a seller, I want to edit AI-generated descriptions to customize them further if needed.	M	4 days	Medium

– Next Page

Table 2.1 – Product's backlog

ID	Feature	User Story	Priority	Estimation	Complexity
3	Chatbot Support	As a seller, I want access to a chatbot for assistance with product listing queries and buyer inquiries.	M	7 days	High
4	Image Analysis for Features	As a user, I want the app to analyze uploaded images and extract relevant product features for description.	M	8 days	High
5	Price Recommendation	As a seller, I want the app to provide pricing suggestions based on product type and market trends.	S	5 days	Medium
6	User Interface Design	As a user, I want an intuitive interface to easily navigate product listing and chatbot support features.	M	7 days	Medium
7	Implementing RAG (Retrieval-Augmented Generation)	As a seller, I want the AI to access relevant product information for more accurate and contextualized descriptions.	S	8 days	High
8	Context History	As a user, I want the chatbot to retain context in conversations for more personalized support and efficient interactions.	S	6 days	High
9	Testing and Debugging	As a developer, I want thorough testing of the description and support features to ensure accuracy and performance.	M	5 days	High
10	Deployment	As a team, we want to deploy the application to a server, ensuring secure access and easy integration for users.	M	3 days	Medium

Based on the product backlog we produced the release backlog witch contain the user story ordered by the priority that has been set by the product owner. The following table present the

planned sprints:

Sprint 0	**Sprint 1**	**Sprint 2**	**Sprint 3**
from 01/07 to 15/07	from 16/07 to 30/07	from 01/08 to 15/08	from 16/08 to 31/08

2.6 Diagram of use cases

In this section, we present the global use case diagrams to better understand and organize the requirements of our system. We will give more details about this diagram later on the realization phases in our sprints. In a use case diagram, each actor represents a specific role and each use case represent a functionality performed by the system.

The Figure 2.1 represents the use case diagram

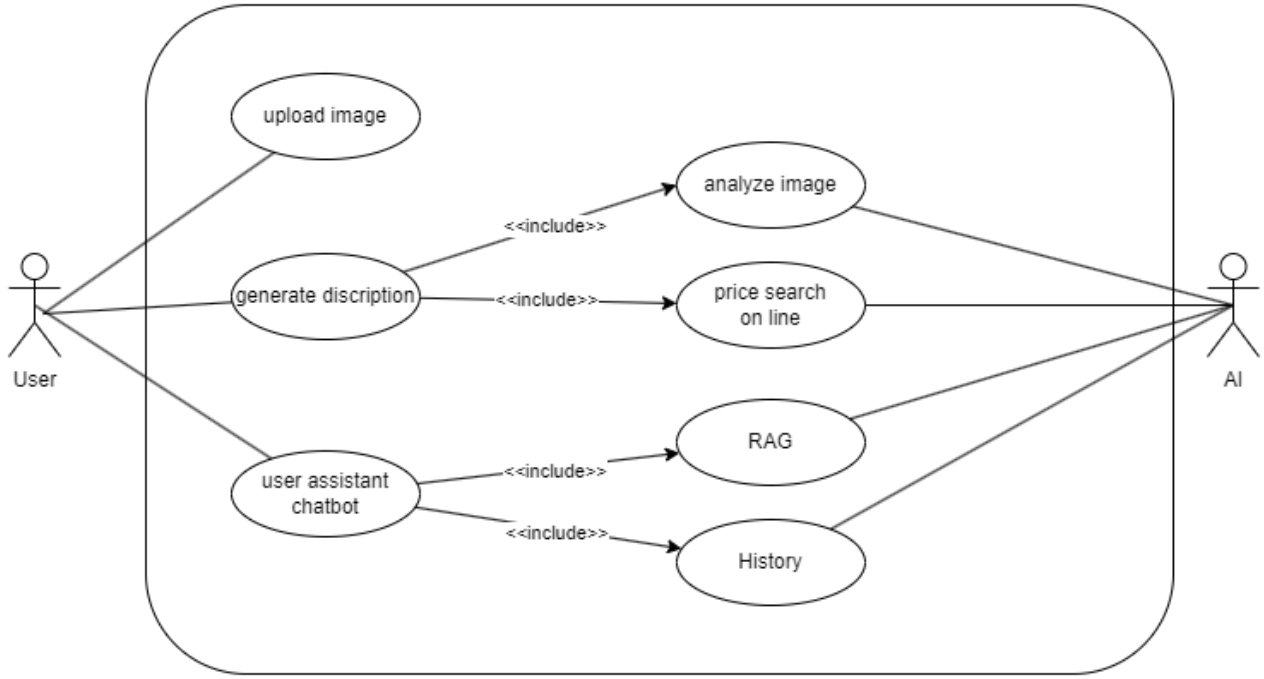

Figure 2.1: Global use case diagram

Conclusion

Throughout this chapter, we prepared our work plan, in which we identified the project team. We built our product backlog based on functional and non-functional requirements. Finally, we mentioned the division of our realization into sprints. In the next chapter, we will present our first sprint.

Chapter 3

Sprint 0: Setting up the application environment

Introduction

In this chapter, we present the sprint zero, during which we introduce the architectural design of the solution as well as the technologies and programming languages adopted for the implementation of our solution.

3.1 Unified Modeling Language

For the design of our application, we used the UML a modeling language, a graphical language that is used to specify, visualize, modify and build the necessary documents for a proper development of an object-oriented software. Thanks to its graphic formalism and the notion of diagrams, it allows us to express the needs of our system and analyze them. UML is independent of the application domain and application languages. We used "Lucidchart" as it allows to model complex systems in a simplified and standardized graphical and textual format.

The diagrams are hierarchically dependent and complete each other, so that a project can be modelled throughout its life cycle. It is a uniform language supported by the Object Management Group that oversees the definition and maintenance of UML specifications.

3.2 Software environment

3.2.1 Development tools

IntelliJ IDEA

IntelliJ IDEA is a powerful and versatile integrated development environment (IDE) that supports a wide range of programming languages and frameworks. It is widely used for Java development and offers various features such as code completion, debugging, and version control, making it a popular choice among developers.

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase. Flutter provides a fast development environment and a wide range of widgets for building responsive and visually appealing applications.

Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. Android studio provides a unified environment where you can build apps for Android phones, tablets, Android Wear, Android TV and Android Auto. Structured code modules allow you to divide your project into units of functionality that you can independently build, test, and debug[1].

Postman

Postman is an application used for API testing. It is an HTTP client that tests HTTP requests, utilizing a graphical user interface, through which we obtain different types of responses that need to be subsequently validated[2].

Docker

Docker is a platform for creating, deploying, and managing containerized applications. It allows developers to package applications and their dependencies into lightweight containers, ensuring consistency across development, testing, and production environments. Docker simplifies the setup of isolated environments, making it easier to manage microservices and deploy applications with minimal configuration.

n8n

n8n is an open-source workflow automation tool that enables users to create automated workflows without extensive programming knowledge. It provides a visual interface to connect and automate tasks across various applications and services. In this project, n8n can be used to automate processes, such as triggering API calls or data processing tasks, streamlining operations and improving efficiency.

3.2.2 Technological Choices

For the backend, we used Spring Boot combined with Spring AI to integrate OpenAI APIs for natural language processing capabilities, enabling features such as AI-powered product description generation and chatbot support. Additionally, we incorporated Tavily as a search engine for large language models (LLMs) to provide contextual data retrieval, enhancing the relevance and accuracy of AI responses.

In terms of mobile app development, cross-platform frameworks were considered to reduce development time and costs while maintaining high performance.

Flutter: Developed by Google, Flutter is a powerful SDK that supports high-performance cross-platform apps with a single codebase. Its library of widgets enables rapid development of engaging and functional user interfaces. Flutter has gained significant popularity and is considered an excellent choice for applications requiring robust performance and a polished UI.

Considering our application's needs for performance, user interface quality, and cross-platform functionality, Flutter was selected as the preferred framework.

3.3 Conclusion of Technological Choices

The selected stack, including Spring Boot, OpenAI API integration, Tavily, and Flutter, provides a comprehensive solution that balances performance, cross-platform support, and AI capabilities. This setup is well-suited to meet the project's requirements for fast, responsive AI-powered features while offering a seamless user experience across both iOS and Android. SQLite was chosen as the database for its simplicity and ability to handle transactional data.

Conclusion

In conclusion, the challenges faced by sellers on online marketplaces, such as providing clear product descriptions and receiving adequate support from platforms, can significantly impact their ability to succeed in a competitive environment. These issues lead to decreased visibility, lower customer satisfaction, and missed business opportunities. The proposed AI-driven solution addresses these problems by automating the generation of high-quality product descriptions and offering real-time support for sellers. This approach not only enhances the overall efficiency of the selling process but also boosts customer trust and increases sales potential

Chapter 4

STUDY AND REALISATION OF SPRINT 1

Introduction

In this chapter, we will focus on the first release of the project, Sprint 1. we will present the work done during this sprint and the different obstacles we have encountered and how we have overcome them. The sprint goal was to deliver a first version of the application with key features: image analysis for extracting product features, price recommendations based on market trends, AI-generated product descriptions from uploaded images, and an editor for customizing these descriptions.

4.1 Sprint Backlog

Tableau 4.1: Product's backlog.

Id	Feature	User Story	Priority
1	Image Analysis for Features	As a user, I want the app to analyze uploaded images and extract relevant product features for description	High
2	Price Recommendation	As a seller, I want the app to provide pricing suggestions based on product type and market trends.	Medium
3	Generate Product Descriptions	As a seller, I want the AI to generate product descriptions based on uploaded images and key details.	High
4	Product Description Editor	a seller, I want to edit AI-generated descriptions to customize them further if needed.	High

4.2 Sprint Functional specifications

4.2.1 Sprint1's use case diagram

- **Sprint1's use case diagram**

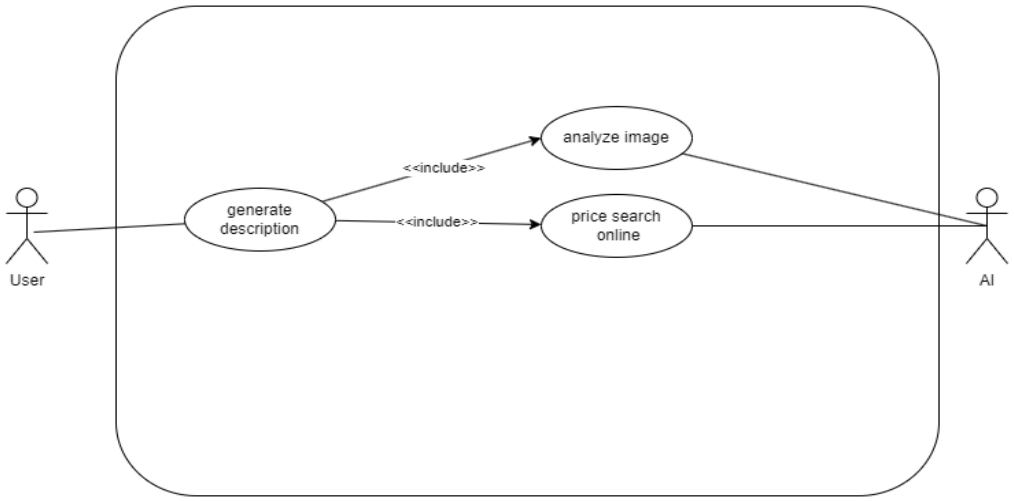

Figure 4.1: Sprint1's use case diagram

Tableau 4.2: textual representation of use case "Generate Product Description"

Title	Generate Product Description
Actor	Seller
Pre-condition	The item image is available for upload
Post-condition	Product description generated and editable
Nominal Scenario	1 Seller uploads the item image. 2 The interface displays a "Generate" button. 3 Seller clicks on "Generate". 4 The system analyzes the image to extract features and performs an online search to recommend a price. 5 The interface displays the generated description, including features and price. 6 Seller can edit the description if needed.
Exception Scenario	If no relevant features are detected, the system displays an error message and suggests retrying or entering information manually.

- **Sequence diagram of Generate Product Description**

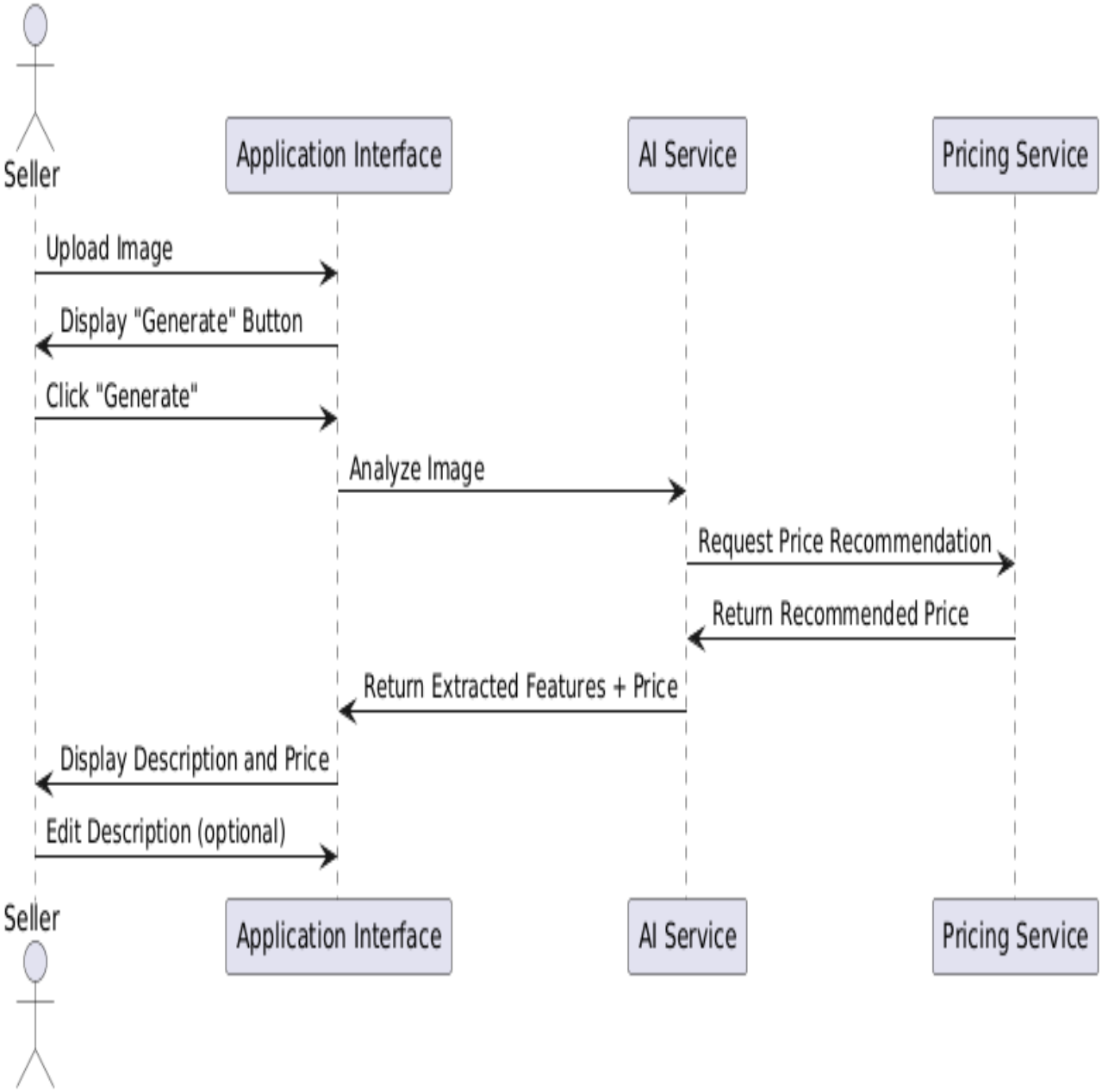

Figure 4.2: Sequence diagram of Generate Product Description

4.3 Realization phase

Our goal in the first sprint was to deliver to our client a release that included the main features of the system: image analysis for extracting product features, price recommendations based on market trends, and AI-generated product descriptions based on uploaded images. There is a specific order we need to follow. We first implemented image analysis to extract relevant features, allowing us to generate accurate descriptions and determine price recommendations effectively. Finally, we included an editor for users to customize AI-generated descriptions.

We encountered issues with price accuracy in the price search process, which took longer than expected to resolve. Initial searches often returned inconsistent or outdated prices due to overly broad search criteria. To address this, we refined our query approach and added filters, resulting in more reliable price recommendations. Further adjustments may be needed, but this has improved overall accuracy.

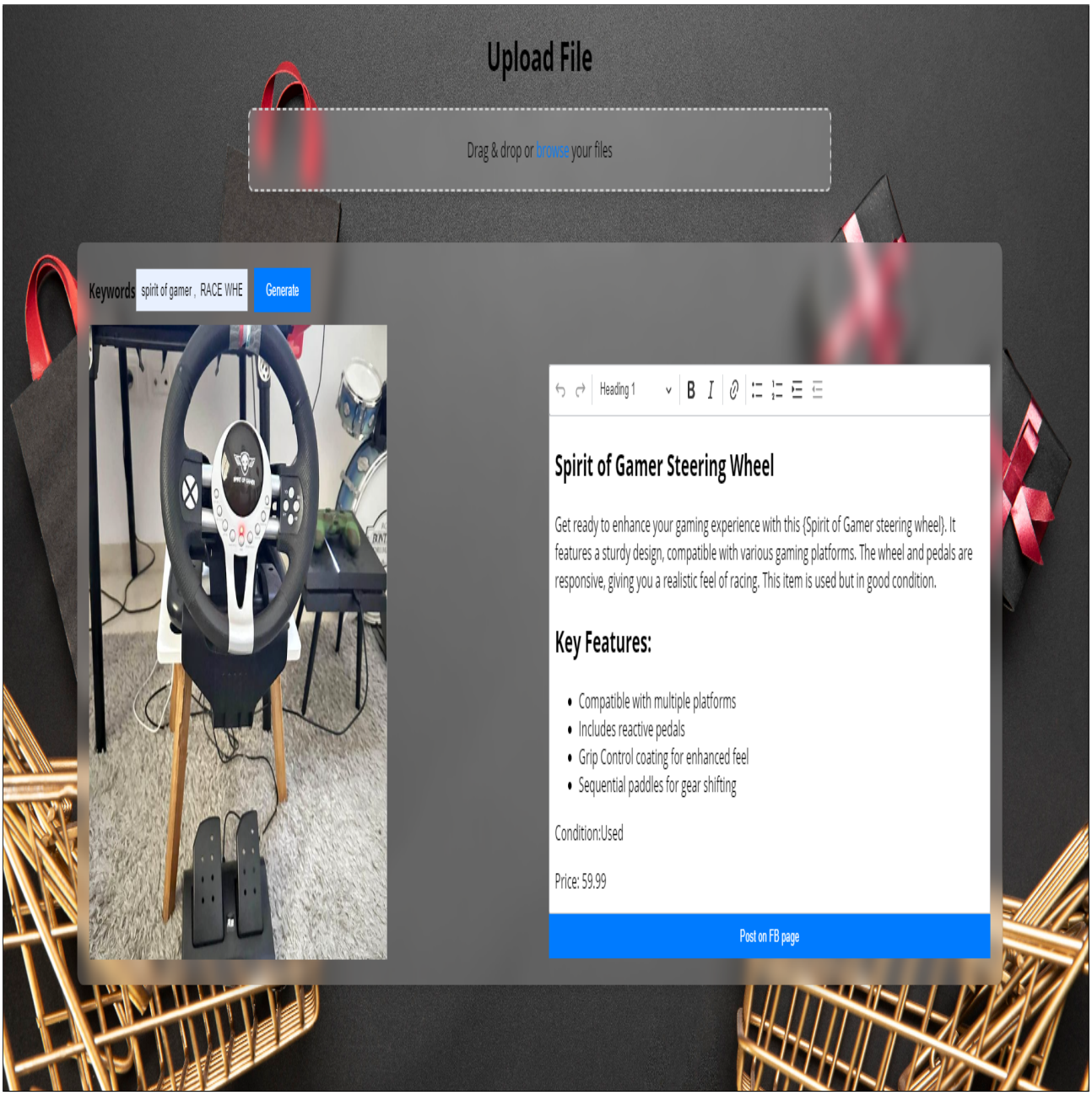

Figure 4.3: Generate Product Description

4.4 Sprint Retrospective

In this sprint, we have reached our goal to deliver the main functionally of the system. We did not make it to implement all the user story in our sprint backlog because we miss-estimate the velocity of some user story. but we will make sure not to repeat this mistake in the future.

Chapter 5

STUDY AND REALISATION OF SPRINT 2

Introduction

In this chapter, we will be focusing on the second release of the project Sprint 2. we will present the backlog of the second sprint, the functional specifications, conception and we will dive through the realization then we will end it with a conclusion.

5.1 Sprint Backlog

The goal of this sprint is to implement all the features related to the ChatBot .

Tableau 5.1: Product's backlog.

Id	Feature	User Story	Priority
1	Chatbot Support	As a seller, I want access to a chatbot for assistance with product listing queries and buyer inquiries.	High
2	Implementing RAG (Retrieval-Augmented Generation)	As a seller, I want the AI to access relevant product information for more accurate and contextualized descriptions.	High
3	Context History	As a user, I want the chatbot to retain context in conversations for more personalized support and efficient interactions.	High

5.2 Sprint Functional specifications

5.2.1 Use Case Diagram

The figure 5.1 present the use case diagram of sprint two.

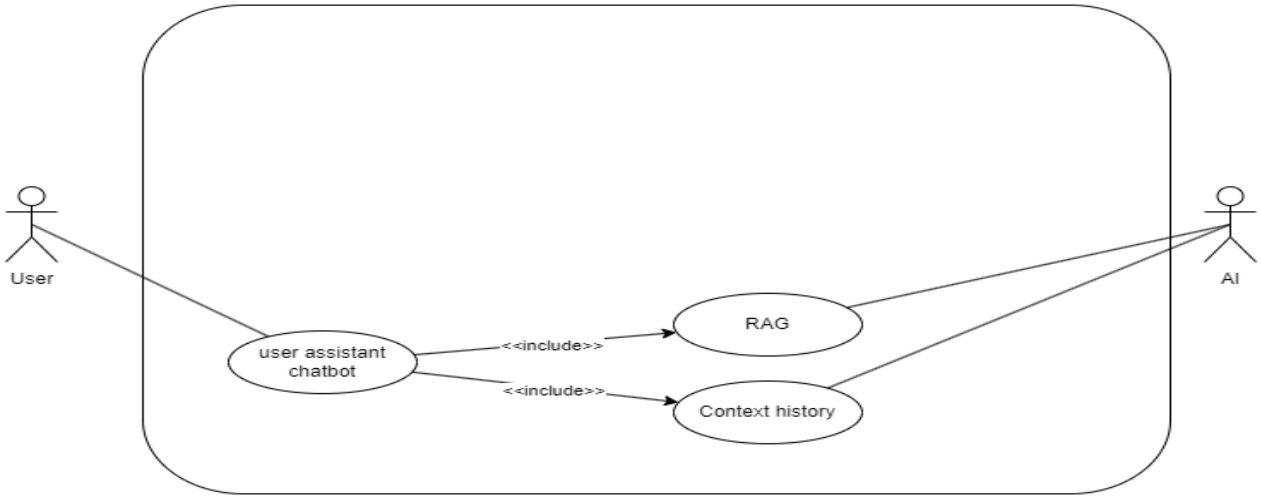

Figure 5.1: Sprint 2 use-case diagram

Tableau 5.2: textual representation of use case "Assistant Chatbot with RAG and Context History"

Title	Assistant Chatbot with RAG and Context History
Actor	Seller
Pre-condition	User is authenticated
Post-condition	The chatbot provides responses with context awareness, improving the seller's assistance experience.
Nominal Scenario	1 The seller initiates a query or request for product listing assistance. 2 The chatbot, using RAG, retrieves relevant information from a knowledge base to generate a response. 3 The chatbot maintains context from previous interactions to personalize responses for the seller. 4 The chatbot provides a detailed and contextually relevant answer to the seller.
Exception Scenario	E1: Connectivity Issue 5 - The system shows an error if there's a loss of connection or failure to retrieve relevant information. The scenario returns to point 2 once connection is reestablished. E2: Context Loss 6 - If context history is unavailable, the chatbot prompts the seller to rephrase or reinitiate the query to clarify. The scenario returns to point 1.

5.3 Design

5.3.1 Class diagram

This figure 5.2 represent the class diagram of sprint 2:

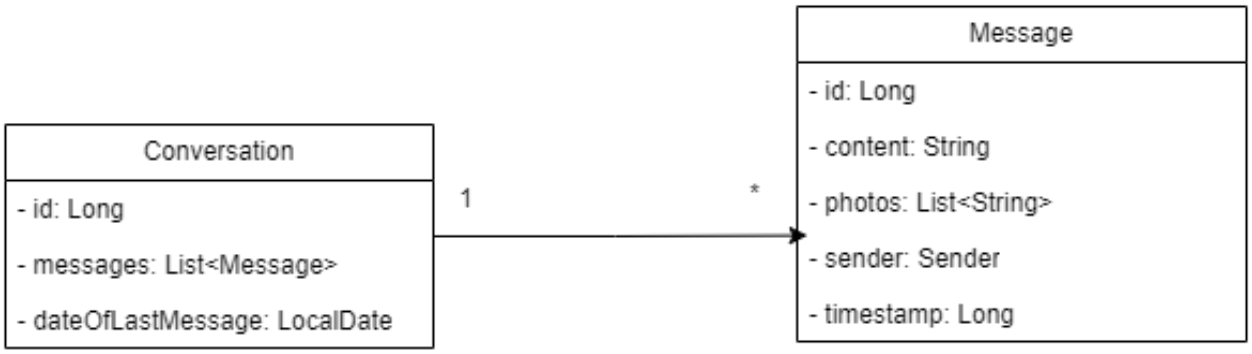

Figure 5.2: Class diagram of sprint 2

5.3.2 System sequence diagrams

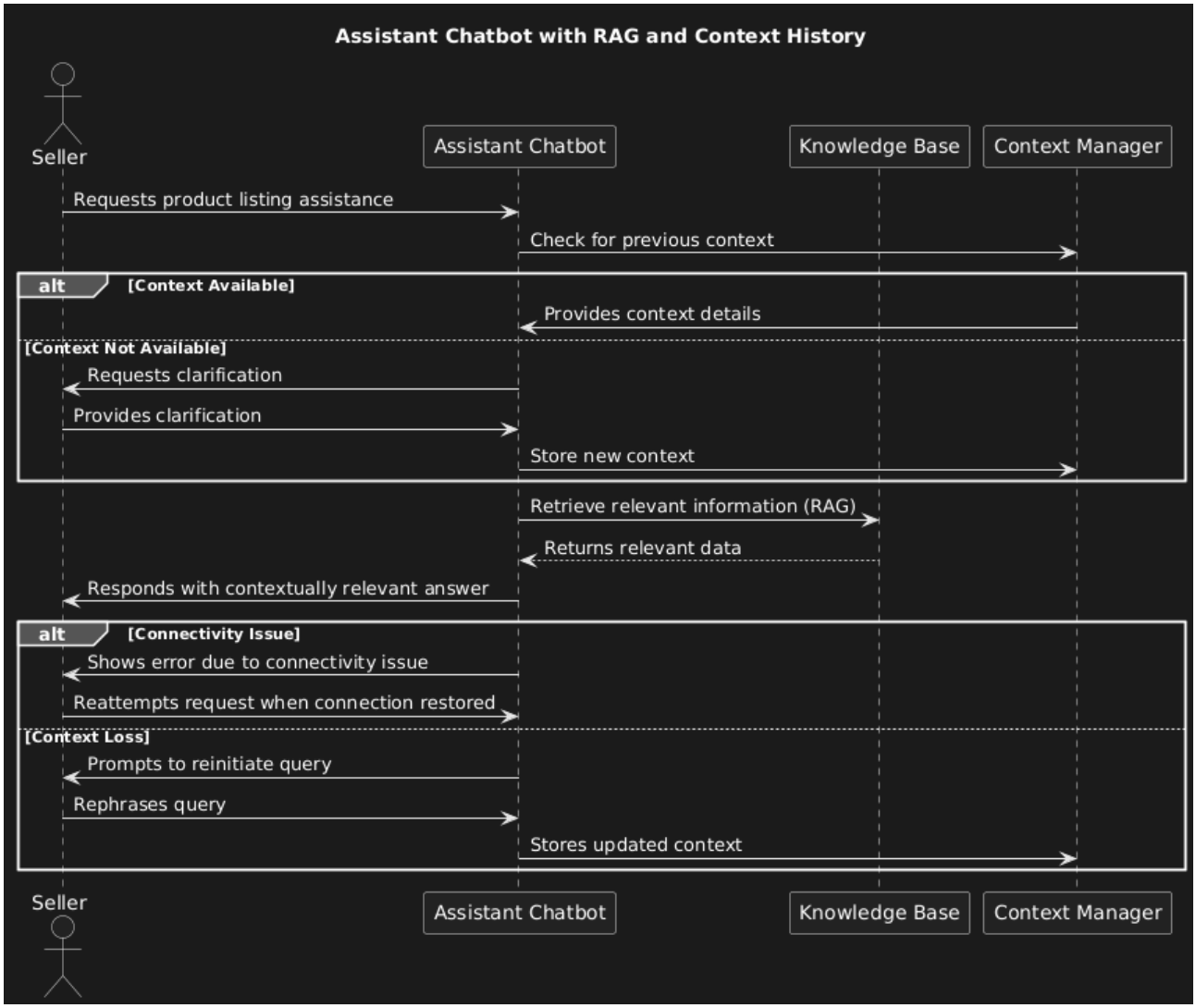

Figure 5.3: Sequence diagram of Assistant chatbot

5.4 Realization phase

In this section, we present the work done during this sprint through screenshots of some interfaces of our application.

In our project, we utilized various libraries and tools . Here's a list of key libraries and tools we used:

- **Spring AI OpenAI API**: This library integrates the OpenAI API within our backend, enabling natural language processing and AI-driven features.
- **Tavily Search Engine**: We leveraged Tavily as a search engine tailored for language models

(LLMs), allowing our application to retrieve and process relevant data from the internet as part of the product description generation.

- **AWS S3**: We used AWS to store images in an S3 bucket, ensuring secure and scalable storage for the images captured and saved by our application.

5.5 Testing

In this sprint, we focused on testing the Retrieval-Augmented Generation (RAG) and context history functionality of the chatbot. This approach allowed us to ensure that the chatbot can effectively retrieve relevant information and maintain context across interactions, improving its consistency and accuracy in responses.

Here is an example of a test:

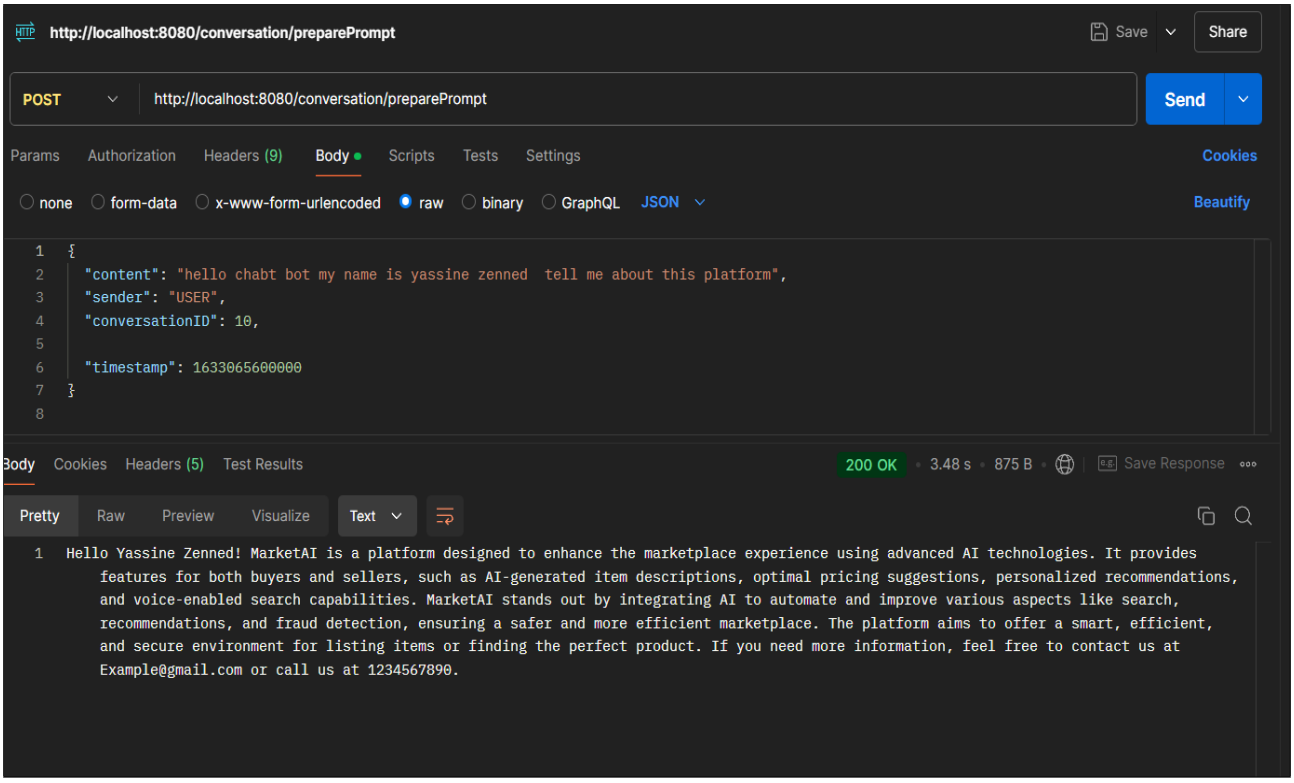

Figure 5.4: Testing the Context + RAG

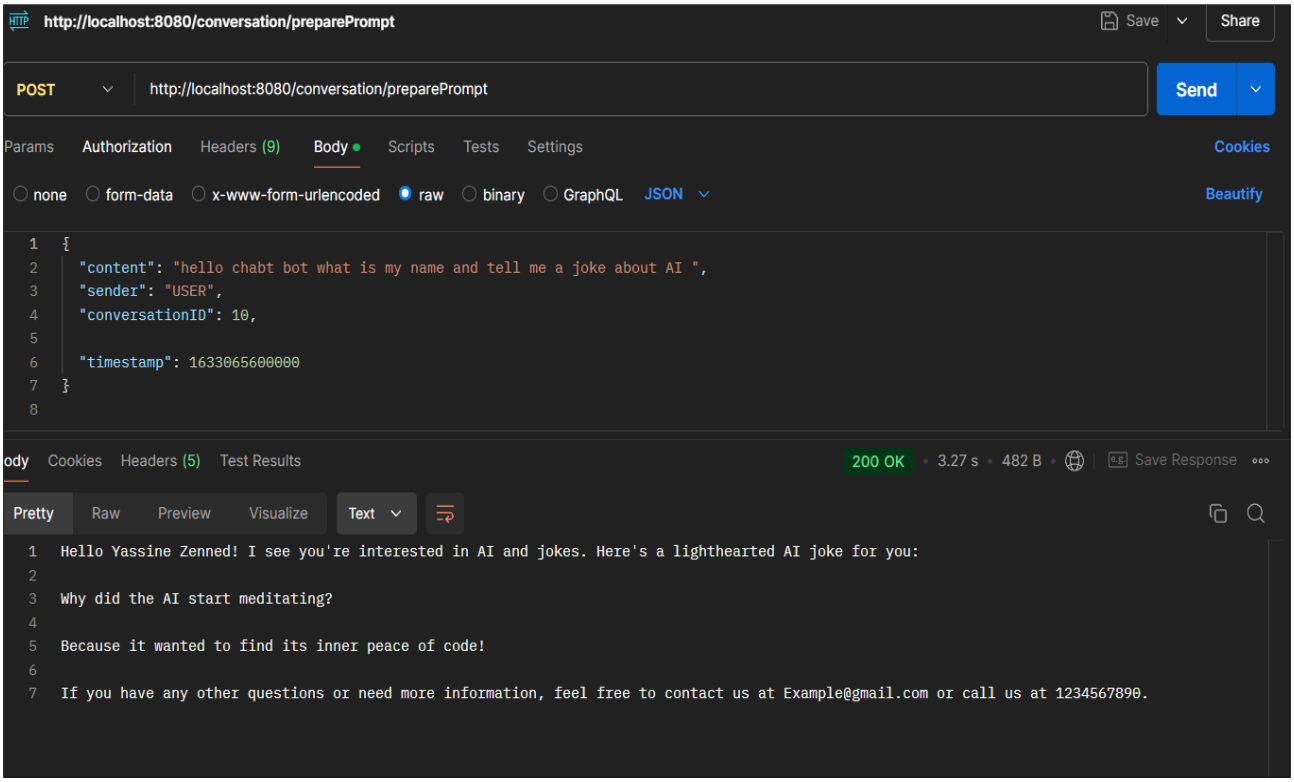

Figure 5.5: Testing the Context + RAG

5.6 Sprint Retrospective

In this sprint, we successfully completed all the items in our backlog, along with the uncompleted features from Sprint 1. However, we discovered some bugs within the user interface. While we addressed several of these issues, a few remain and will be prioritized for resolution in the next sprint.

Chapter 6

Sprint 3: User interface + Testing + Deployment

Introduction

In this chapter, we will focus on the third and final release of the project, Sprint 3. The goal of this sprint was to complete the application by focusing on building the UI, conducting thorough testing, and ensuring successful deployment.

6.1 Sprint 3 Backlog

Tableau 6.1: Product's backlog.

Id	Feature	User Story	Priority
1	User Interface Design	As a user, I want an intuitive interface to easily navigate product listing and chatbot support features.	Medium
2	Testing and Debugging	As a developer, I want thorough testing of the description and support features to ensure accuracy and performance.	High
3	Deployment	As a team, we want to deploy the application to a server, ensuring secure access and easy integration for users.	Medium

6.2 Realization phase

We began Sprint 3 by fixing bugs and deploying the backend. Once this was complete, we prioritized a new feature: implementing a workflow to post images and descriptions directly to a Facebook page. This section presents the work done during this sprint, including screenshots of relevant interfaces.

Screenshots form that the new Features:

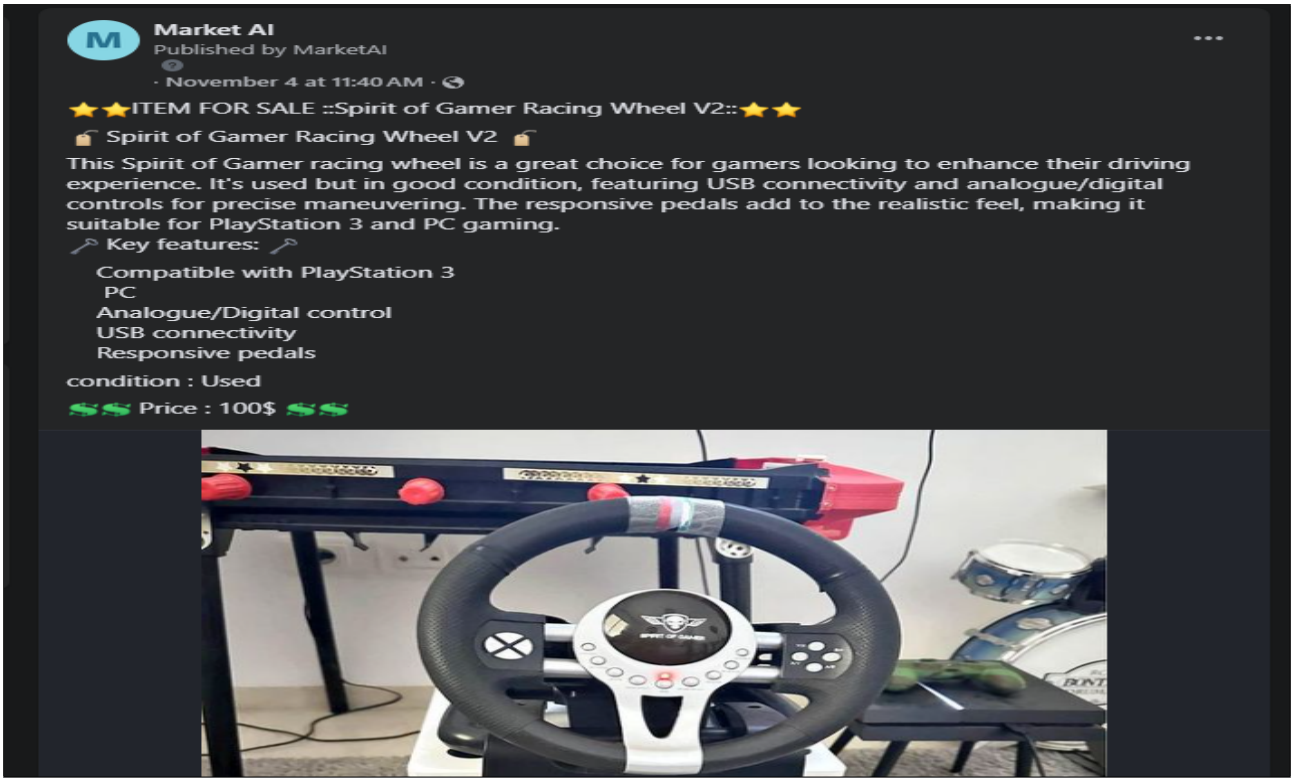

Figure 6.1: Facebook Post

6.3 Testing

In this sprint, we implemented testing using a combination of Testcontainers and an H2 in-memory database to create isolated environments for unit and integration tests. This approach allowed us to effectively simulate external dependencies and test database interactions, ensuring consistency and accuracy.

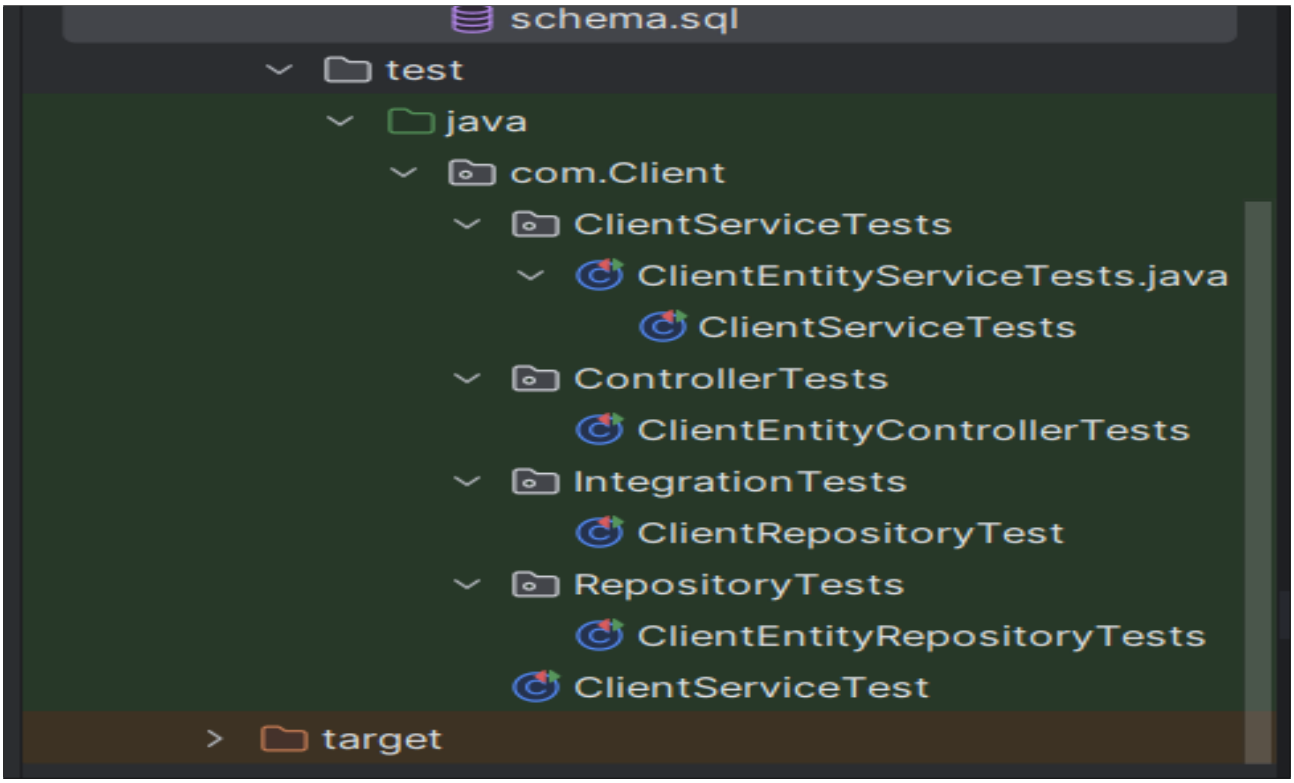

Figure 6.2: all the TESTs

Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. By utilizing the H2 in-memory database, we isolated each unit test, making them faster and preventing conflicts with other tests.

Here is an example of a unit test:

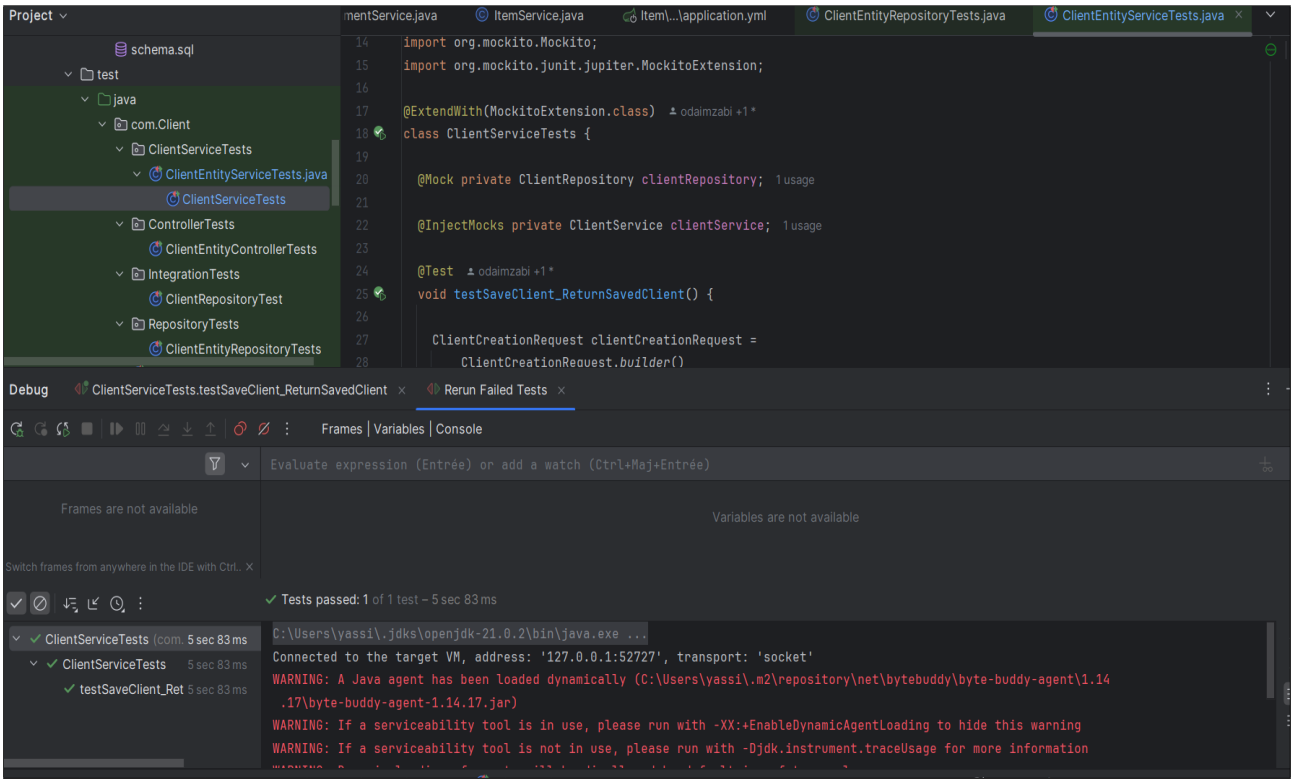

Figure 6.3: Unit test completed successfully

Integration Testing

Integration testing is the phase in software testing where individual software modules are combined and tested as a group. Using Testcontainers, we created a controlled, containerized environment to simulate the application's database and external dependencies. This setup allowed us to validate the interactions and compliance of components with specified functional requirements. Integration tests were conducted after unit tests and before system testing.

Here is an example of an integration test:

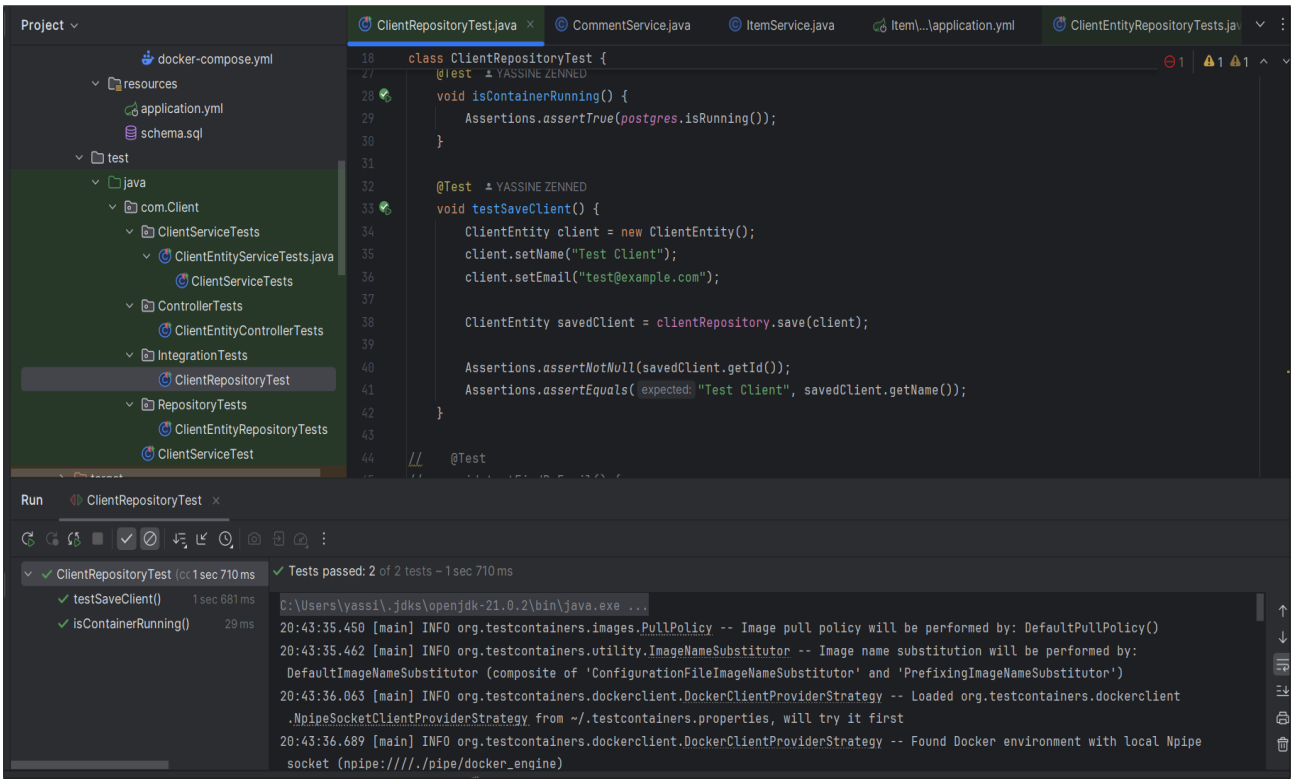

Figure 6.4: Integration test

6.4 Sprint Retrospective

In Sprint 3, we successfully met our delivery goal by implementing all the new features from our sprint backlog. This included key functionalities that were prioritized, contributing to the overall progress of the project. However, due to time constraints, we were unable to allocate sufficient time to complete the chatbot UI, which was initially planned for this sprint. Despite this, the new features were fully integrated and tested, making significant advancements in the application. The focus on these features allowed us to meet the main objectives of Sprint 3, even though the UI component for the chatbot was left unfinished.

General Conclusion

This project successfully developed and deployed an AI-powered application aimed at improving the product listing process for sellers. The features implemented include the generation of product descriptions based on images and key details, an editor for customizing AI-generated content, a chatbot for handling product listing and buyer queries, and advanced image analysis to extract relevant product features. Additionally, a price recommendation engine and the implementation of Retrieval-Augmented Generation (RAG) enabled more accurate and contextual descriptions, while context history in the chatbot allowed for personalized and efficient interactions.

The project followed the Scrum methodology, ensuring iterative progress through sprints. High-priority features such as product description generation, chatbot support, and RAG implementation were completed within the planned timelines, demonstrating effective teamwork and adherence to deadlines. We also focused on testing and debugging to ensure the app's accuracy and performance.

However, while the key features were successfully implemented, there is still room for improvement. Further enhancements could include refining the user interface for even more intuitive navigation, expanding the chatbot's capabilities to handle more diverse queries, and improving the AI's ability to provide dynamic price recommendations. Additionally, integrating feedback loops for continuous learning from user interactions and product trends could further optimize the platform.

In summary, the project met its objectives, delivering a functional and efficient application for product listing and support, but ongoing development and refinement are necessary to fully meet user expectations and market demands.