UNIVERSIDADE DE AVEIRO

DETI Bot

Inês Ferreira 104415, Pedro Matos 102993, Diogo Gaitas 73259, Simão Antunes 104092, Guilherme Lopes 103896

Licenciatura em Engenharia Informática

Supervisor: Mário Antunes

June 6, 2024

Abstract

The DETI Bot project aims to enhance the accessibility of information related to the Department of Electronics, Telecommunications, and Informatics (DETI) at the University of Aveiro. This chatbot leverages advanced technologies, including Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG), to provide accurate and contextually relevant responses to user inquiries.

Developed by a team of five students as part of their Bachelor's Degree in Informatics Engineering, the DETI Bot monitors various resources such as webpages, forums, and mailing lists, and offers a natural language interface for interaction. The project addresses the challenge of scattered information, making it easier for students, professors, and visitors to access comprehensive details about courses, events, and departmental activities.

The architecture of DETI Bot includes a presentation layer for user interaction, a business layer that processes data and queries, and a database layer utilizing Qdrant for vector storage and MySQL for relational data. The bot ensures high performance, scalability, and reliability by converting text into high-dimensional vectors for efficient storage and retrieval.

Key features of the DETI Bot include context-aware responses, a user-friendly interface, and the ability to manage and update knowledge sources effectively. The project highlights the integration of modern AI technologies to solve real-world problems, providing a valuable tool for the DETI community.

This report details the design, implementation, and testing of the DETI Bot, demonstrating its potential to streamline information access and support the educational environment at the University of Aveiro, specially in DETI.

Acknowledgements.

Special thanks to our supervisor: Professor Mário Antunes.

Contents

List of Figures

List of Tables

Glossary

API
: Application Programming Interface is a set of routines and standards established by a software application to provide its functionalities to other applications without getting involved in the software's implementation details.

DETI
: Department of Electronics, Technology, and Informatics.

Glossary
: A glossary is a kind of small dictionary specific to little-known words and expressions present in a text, whether they are of a technical nature, regional, or from another language.

LLM
: Large Language Model is an advanced artificial intelligence model trained on vast amounts of text data to understand, generate, and manipulate human language.

MySQL
: MySQL is an open-source relational database management system that uses Structured Query Language (SQL) for accessing and managing data.

Qdrant
: Qdrant is a high-performance, open-source vector search engine and database for storing, searching, and managing embeddings from machine learning models.

RAG
: Retrieval-Augmented Generation (Patrick Lewis(2020)) is a technique in artificial intelligence where a language model is supported by an information retrieval system that indexes relevant textual sources to provide more accurate and contextually appropriate responses.

WWW
: The World Wide Web is a system of interlinked documents accessible through the Internet using a web browser.

UI
: UI stands for User Interface, which refers to the space where interactions between humans and machines occur. The goal of a UI is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

Chapter 1

Introduction

1.1 Context

DETI (uni(2024)) is a department housing multiple student groups and research groups, which organize several different activities throughout the semester. Additionally, the department offers a wide range of educational opportunities. However, given the breadth of these activities and resources, it can be challenging to stay informed about everything happening within the department at any given time.

This project proposes the development of DETI Bot, a chatbot that monitors relevant resources (webpages, forums, Discords, and mailing lists) and provides a natural language interface for the DETI community to interact with. Large Language Models (LLMs) enable the construction of chatbots that can interact with users through natural language. However, a major limitation is that fine-tuning LLMs is quite expensive, considering computational costs, dataset acquisition, and other factors. One method to overcome this limitation is to use Retrieval-Augmented Generation (RAG), a process where the LLM functions as a simple language model and is supported by an information retrieval system that indexes relevant textual sources.

This project was developed by a team of five members as a component of the Project in Informatics course within the Bachelor's Degree program in Informatics Engineering at the University of Aveiro. Throughout the second semester, we were tasked with developing a software solution addressing a self-selected problem. We elected to pursue a challenge proposed by Professor Mário Antunes, entitled DETI Bot. This project involves the creation of an AI-powered chat bot designed to respond comprehensively to all inquiries related to DETI.

1.2 Motivation

As of today, information pertaining to DETI and its associated organizations is scattered across various platforms like discord servers, slack chats, DETI's organizations websites(like NEI's website), and there is information about deti and its related courses which u could only obtain by sending an email to DETI's office, thus complicating the process and consuming additional time for individuals seeking insights into events, courses, or general knowledge about DETI,

specially new students, which tent do struggle to find these websites or even be aware of them. Therefore, our goal is to streamline the accessibility of this information, ensuring that users can retrieve it more promptly and conveniently through the DETI Bot. This tool allows users a significantly facilitating experience while on their information-gathering process.

It will bring together information from the most relevant DETI related websites as it will involve a platform where DETI's staff will input important information that may not even be available anywhere else. With the technologies available today DETI would also benefit from such a platform, since with today's technologies the DETI bot could be created and implemented as a way to attract new students and let them know more about the department.

1.3 Objectives

This project aims to develop a DETI Bot agent that leverages both Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) to monitor and index all relevant data sources pertaining to departmental activities. The bot will offer diverse interfaces for the community to seamlessly interact with this information. In points we present the main stepping stones to reach our main goal of creating the DETI Bot:

- An user-friendly interface to chat with the bot
- Context-aware and up to date bot responses
- It should answer to questions relevant to all DETI staff, students and professors.

Chapter 2

State of the Art

2.1 Introduction

State of the Art is the current highest level of development, achievement, or advancement in a particular field. Staying updated with the state of the art of that field is crucial as it drives innovation, improves model performance, and guides research.

2.2 Related Work

Retrieval-augmented generation (RAG) (Patrick Lewis(2020)) is the current state of the art technique for enhancing the accuracy and reliability of generative AI models with facts fetched from external sources. The objective of this project is to gather all the information available about DETI and it's events and apply this technique to power a chat bot with it.

Large Language Models (LLM) are cutting-edge AI models designed to understand and generate human-like text. By training on vast amounts of text data, LLMs can learn complex patterns and structures of language, enabling them to generate coherent responses in natural language. Leveraging the power of LLMs, this project aims to enhance the capabilities of the chatbot by providing it with the ability to understand and respond to user queries about DETI and its events.

Embedded vectors, also known as embeddings, play a crucial role in representing words or phrases in a numerical format. These representations capture semantic relationships between words, allowing models to understand context and meaning. By incorporating embedded vectors into the chatbot's architecture, we can improve its ability to process and generate text that is contextually relevant to DETI. The Llama3 8B model has demonstrated remarkable performance across various benchmarks assessing language comprehension and reasoning abilities. It outperformed reference models such as Mistral 7B and Gemma 7B in benchmarks like MMLU,

	MMLU	AGIEval	ARC	DROP	BIG
Llama3	**66.6**	**45.9**	78.6	**58.4**	**61.1**
Mistral	63.9	44.0	78.7	54.4	56.0
Gemma	64.4	44.9	**79.1**	56.3	56.3

Table 2.1: LLM

	Metadata Filtering	Hybrid Search	Delete	Store Documents	Async
Qdrant	Yes	Yes	Yes	Yes	Yes
Elasticsearch	Yes	Yes	Yes	Yes	Yes
Chroma	Yes	No	Yes	Yes	No
Redis	Yes	No	Yes	Yes	No

Table 2.2: rags

which test factual and common-sense knowledge understanding. Additionally, in AGIEval, measuring tasks like question-answering and sentiment analysis, Llama3 8B showcased versatility by surpassing Mistral 8x22B and Gemma 7B. In the ARC benchmark, focusing on skill acquisition, Llama3 8B excelled, surpassing Mistral 8x22B. Lastly, in DROP, emphasizing logical and numerical reasoning, Llama3 8B demonstrated superiority over Gemma 7B.

2.3 Technologies

2.3.1 Frontend - React

React is a popular JavaScript library created by Facebook used for developing user interfaces and related components. React has a component-based architecture and its efficient rendering makes it an ideal choice for building dynamic and interactive user interfaces. React allows developers to build large web applications that can alter data without requiring a page reload. The fundamental goal of React is to be quick, scalable, and simple.

2.3.2 Backend - FastAPI/LangChain/Ollama

FastAPI

(Ramírez(2018)) FastAPI is a high-performance, web framework for building APIs with Python based on standard Python type hints. We chose this framework for it's, support for asynchronous code by using async/await python keywords; for it's high-performance; for it's simplicity in learning how to use it and using it and finally because of the nature of our project that uses LangChain as the power house of our application, being only necessary a web framework that handles the API requests.

LangChain (lan(2024))

LangChain is an open source framework that lets software developers working with artificial intelligence (AI) and its machine learning subset combine large language models with other external components to develop LLM-powered applications. The goal of LangChain is to link powerful LLMs, such as OpenAI's GPT-3.5 and GPT-4, to an array of external data sources to create and reap the benefits of natural language processing (NLP) applications. We chose it because of it's open-source building blocks, components and third-party integrations that are available as well as it's extensive documentation and support guides. With LangChain we were able to use the libraries provided by it, to load the content of the knowledge sources and index

that information to the vector database, (Qdrant), as well as retrieve that information from the vector store and feed it to a LLM, provided by ,Ollama.

Ollama

(oll(2024)) Ollama is an open-source project that serves as a powerful and user-friendly platform for running LLMs on our local machine. It acts as a bridge between the complexities of LLM technology and the desire for an accessible and customizable AI experience. At its core, Ollama simplifies the process of downloading, installing, and interacting with a wide range of LLMs, empowering us to explore their capabilities without the need for extensive technical expertise or reliance on cloud-based platforms.

2.3.3 Database - Qdrant/MySQL

(qdr(2024)) (Widenius et al.(2002)Widenius, Axmark, and Larsson)

QDrant

Qdrant is a vector similarity search engine that provides a production-ready service with a convenient API to store, search, and manage points (i.e. vectors) with an additional payload. It offers several advantages when dealing with large amounts of text information. Some of the key benefits include high performance and scalability, advanced indexing techniques, fault tolerance and reliability.

MySQL

MySQL is a Database Management System (DBMS) that uses SQL Language as its interface to access and manipulate data. MySQL is a free and open-source software and it is ideal for both small and large applications. MySQL is a relational database, that is, a database where data is stored in tables and tables are related to each other by keys. MySQL is one of the most popular databases.

2.3.4 Containerisation - Docker

Docker is a software framework that helps developers build, share, manage and run applications in containers. Docker's containerisation technology provides a lightweight and portable way to package software with all of its dependencies and run it consistently across different environments that makes it easy to deploy and scale applications in different environments. Compared to virtualisation, containerisation is more lightweight, requires less effort to deploy, run and manage which makes it a great choice for continuous integration and continuous delivery (CI/CD) pipelines and to improve productivity and efficiency.

Chapter 3

System Requirements and Architecture

In this chapter, we aim to delve deeper into the purposes of our product, detailing its diverse capabilities and describing how our team of developers successfully implemented them.

3.1 System Requirements

3.1.1 Elicited Requirements

Throughout the development of the DETI Bot, we, along with our project advisor, carefully defined and refined the various capabilities our chat bot should offer. Presenting them:

- Select language between portuguese or english
- Select trending queries
- Ask questions by text
- Receive answers by text
- View chat history
- Use the bot at anytime
- Obtain information about DETI
- Mantain conversational context
- Get user feedback on bot responses

.

3.2 Non-functional Requirements

The system's non-functional requirements are critical to ensuring its overall quality and performance.

Availability is addressed through robust error handling and fault tolerance mechanisms, ensuring the system remains operational and reliable even in the face of unexpected issues.

Performance focuses on scalability and response time, aiming to provide a responsive experience under varying loads and enabling the system to grow seamlessly with increased demand.

Usability is emphasized by designing a user-friendly UI and ensuring accessibility, making the system intuitive and easy to use for a wide range of users, including those with disabilities.

Maintainability is achieved through modularity and comprehensive documentation, facilitating easier updates, troubleshooting, and enhancements, thereby extending the system's lifespan and reducing maintenance costs.

3.3 Actors

Actors

- **Visitor**: An external user who interacts with DETIBOT.
- **DETI Professor**: An academic staff member who engages with the system.
- **DETI Student**: A student from DETI who uses the bot.
- **Admin**: An administrator responsible for managing the knowledge base of DETIBOT.

Interactions

The primary interaction involves asking various questions about DETI, which is central to the system.

Types of Questions and Interactions

- **Ask for the lowest grade to enter a course**: Visitors can inquire about admission requirements.
- **Ask about DETI courses and their subjects**: All user types can request information about courses and subjects offered by DETI.
- **Rate answer accuracy**: Professors and students have the option to rate the accuracy of the responses provided by DETIBOT.
- **Ask who's the professor in charge of a subject**: Users can ask for the responsible professor of a specific subject.
- **Information on important dates**: Students can inquire about key dates and deadlines.

Knowledge Management

- **Insert new knowledge**: Admins have the capability to add new information to the system.
- **Update existing knowledge (Automated)**: The system is designed to automatically update its existing knowledge base.

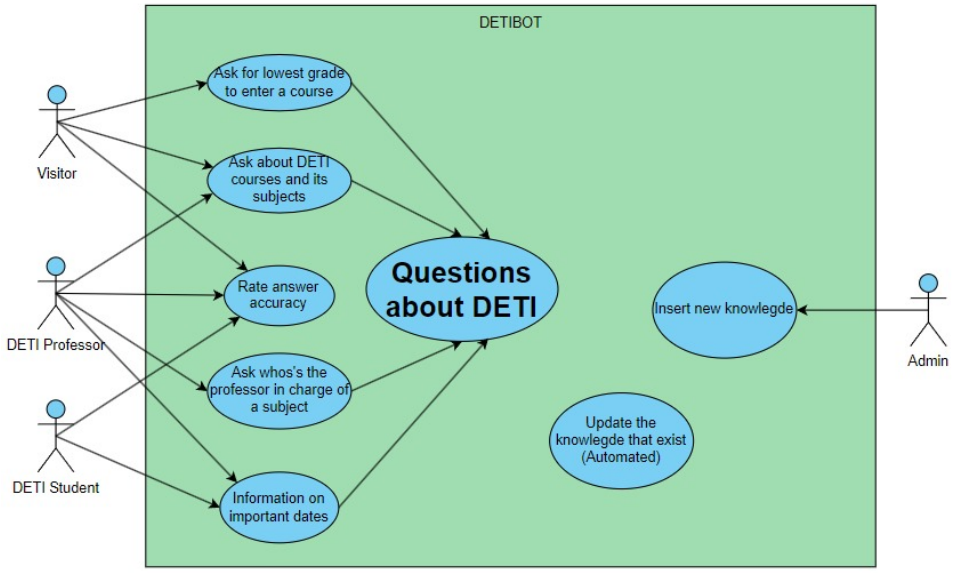

Figure 3.1: Interaction Diagram of DETIBOT

3.4 Use Cases

3.4.1 1st Use Case: Joana Matos

Name	Joana Matos
Age	22
Occupation	Student at DETI
Location	Aveiro

Table 3.1: Joana Matos

Table 3.2: Joana Matos

Joana is a 22-year-old student at DETI who wants to know when her Easter holidays will start. To find out, she accesses the DETI Bot and asks, "When do my Easter holidays start?" The DETI

Bot then responds with the dates for that specific year. For example, in the year 2024, it would answer: "Easter holidays at the University of Aveiro will take place between April 25 and April 31."

3.4.2 2nd Scenario: Francisco Vale

Name	Francisco Vale
Age	18
Occupation	High school Student
Location	Porto

Table 3.3: Francisco Vale

Table 3.4: Francisco Vale

Francisco Vale wants to know what his first subjects are in the first year of the Informatics course at the University of Aveiro. To find out, he accesses the DETI Bot and asks, "What are my first subjects in the first year of the Informatics course at DETI?" The DETI Bot then responds with the list of subjects for that specific year. For example, it would answer: "In the first year of the Informatics Engineering course at the University of Aveiro, your subjects include Fundamentals of Programming, Introduction to Web Technologies, Systems Modeling and Analysis, Linear Algebra and Analytical Geometry ...."

3.4.3 3rd Use Case: Ana Martins

Name	Ana Martins
Age	36
Occupation	Teacher at DETI
Location	Aveiro

Table 3.5: Ana Martins

Ana Martins, a 36-year-old professor at the Department of Electronics, Telecommunications, and Informatics (DETI) at the University of Aveiro, needs information about the history of DETI since she is a new employee.

She logs into the DETI Bot web app and opens the Deti Bot interface. Ana types her query: "Can you tell me about the history of the DETI?"

The Deti Bot responds with a detailed history of the DETI department. It includes information about the department's founding year, major achievements, research contributions, and key figures.

3.4.4 4th Scenario: Joana Martins

The same Joana Martins from the 1st scenario is still in the same chat with the DETI Bot, but now after asking when do her easter holidays start she then asks "When do they end?" to which the DETI Bot contextualizes as asking about her easter holidays and answers with the respective date in which her easter holidays do end.

3.5 Architecture and technologies

3.5.1 Domain Model

To efficiently handle large amounts of text, we transform text into vectors and store them in a Qdrant vector database. This approach offers high performance, scalability, advanced indexing, fault tolerance, and reliability.

Key components of the domain model:

- **Deep Neural Network**: Converts text inputs (e.g., "striped blue shirt made from cotton") into high-dimensional vectors for efficient storage and retrieval.- Qdrant

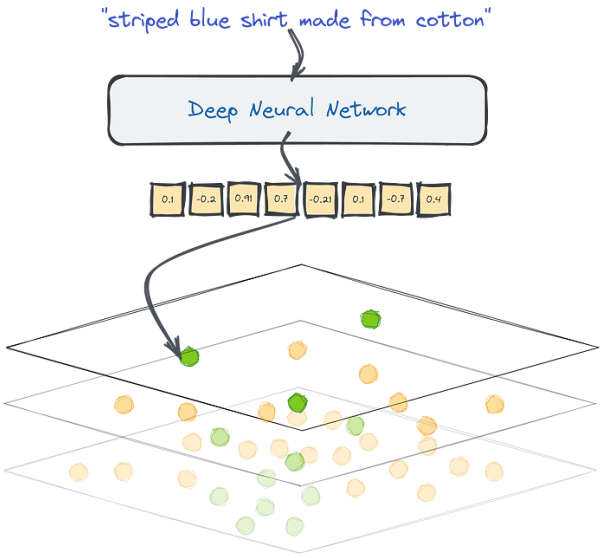

Figure 3.2: Vector search

- **Database Schema**:
  - **url_source**: Stores source URLs with fields like `id`, `url_link`, `parts`, `descript`, and `update_period`.
  - **url_child_source**: Manages child URLs linked to parent URLs with fields `id`, `url_link`, and `parent_id`.
  - **update_time**: Tracks update timings with fields `id`, `period_date`, and `update_period`.
  - **faq_source**: Contains FAQs and answers with fields `id`, `question`, and `answer`.
  - **file_source**: Stores file information with fields `id`, `file_name`, `file_path`, `loader_type`, and `descript`.

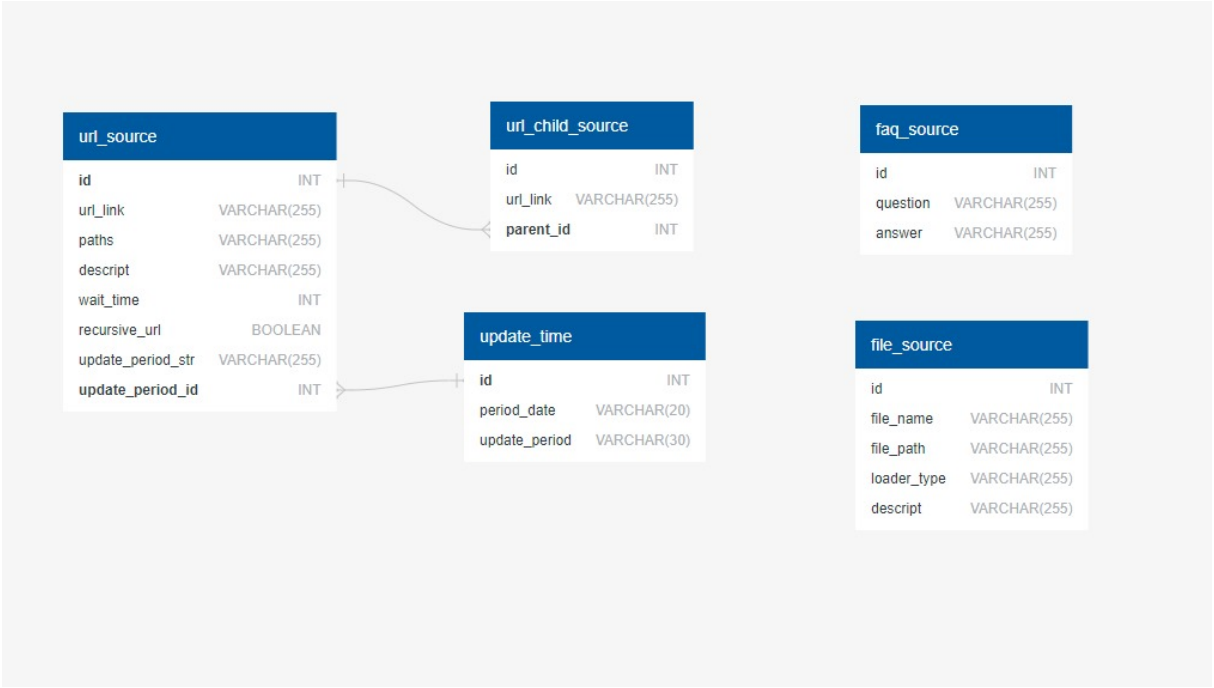

Figure 3.3: Database diagram

3.5.2 Architecture Design

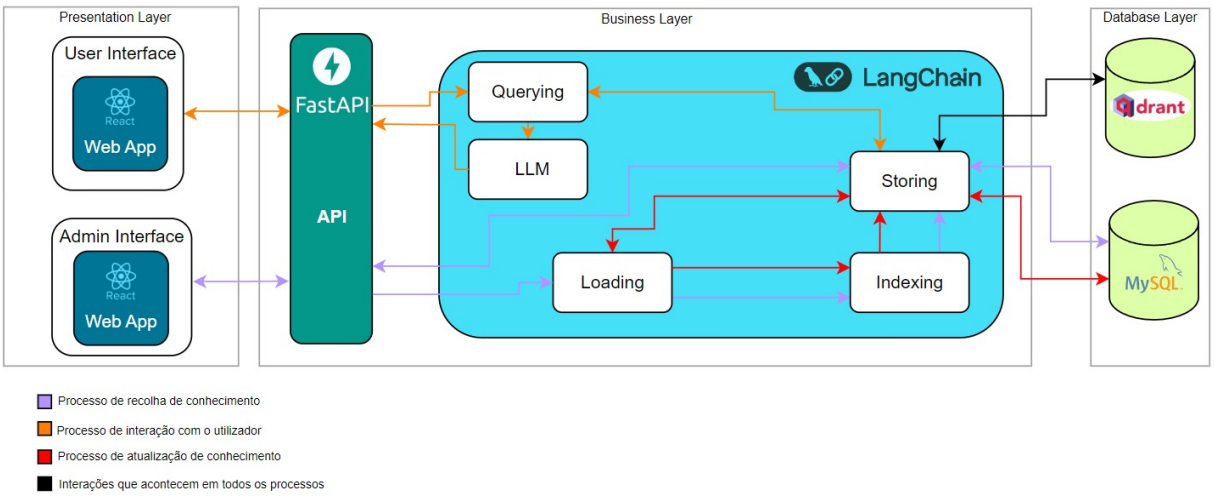

Figure 3.4: Architecture

For our application to work as intended, we developed the architecture shown above . This architecture is composed of three layers:

Presentation Layer

The Presentation Layer consists of two primary components: the User Interface and the Admin Interface. Both components are implemented as web applications using React(Walke(2013)). This layer is responsible for facilitating interaction between the user and the chatbot, as well as enabling administrators to manage the chatbot's knowledge.

Business Layer

The Business Layer is the core of the chatbot's functionality. It includes several key components:

- **API (FastAPI (Ramírez(2018)))**: This acts as the gateway for communication between the Presentation Layer and the business logic. It handles requests from the user interface and forwards them to the appropriate modules within the Business Layer.
- **Loading**: This component manages the process of loading data into the system. It retrieves the data from the knowledge sources.
- **Indexing**: This module is responsible for the indexing of the data retrieved in the loading component and save it into the vector store.
- **Querying**: This module handles the logic for retrieving relevant information from the vector store to then with the help of an LLM give an answer to the user.
- **LLM (Large Language Model)**: The LLM generates an answer to the user, according to the given information extracted by the querying module from the vector store and the question asked by the user.
- **Storing**: This class provides methods for the components above to help them interact with the databases used in this project.

Database Layer

The Database Layer includes two primary storage solutions:

- **Qdrant**: It's the vector store, enabling efficient handling of high-dimensional data such as as the knowledge of the system.
- **MySQL**: It's the relational data storage, that stores the metadata of the knowledge sources.

Interaction Processes

The architecture diagram highlights various processes involved in the chatbot's operation:

- **Knowledge Collection and Managing Process (Purple)**: This involves the gathering and managing of the knowledge sources that make the knowledge base.
- **User Interaction Process (Orange)**: This outlines the flow of interaction between the user and the chatbot, from input to response generation.
- **Knowledge Update Process (Red)**: This process involves updating the stored knowledge of the Urls, based on a predefined frequency.
- **Interactions Occurring in All Processes (Black)**: This indicates interactions and data flows that are integral to all the processes, ensuring seamless operation across the system.

3.6 Deployment

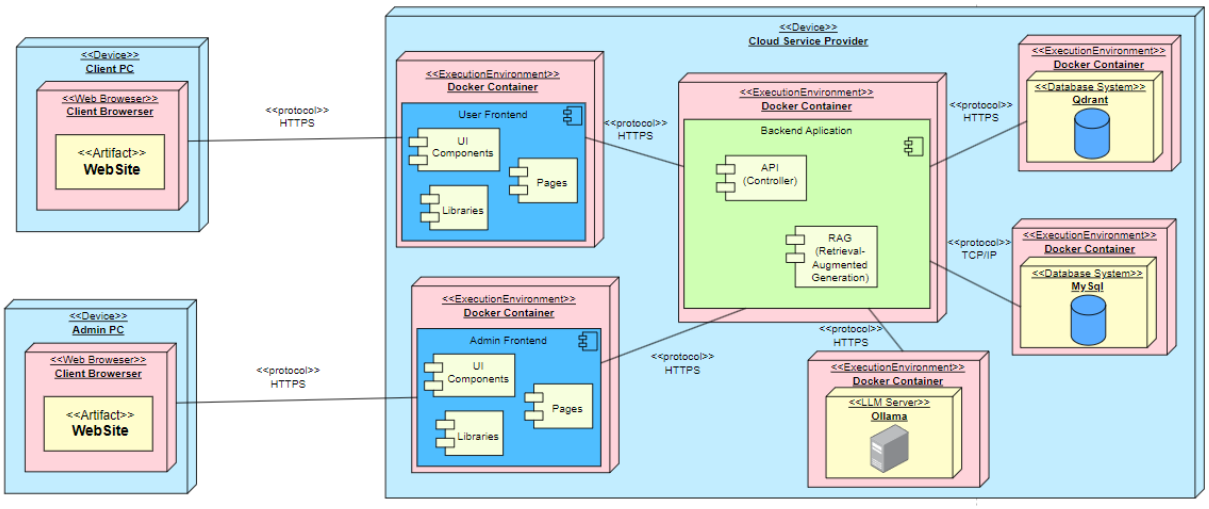

Figure 3.5: Deployment diagram

Deployment is a critical aspect of software engineering, it is the boundary between development code that is in a stable phase and a release product that is ready to show to the client.

Do to the lack of availability of a virtual machine to deploy our project, we could only deploy it locally in our personal computers through docker (Hykes(2013)). Do to this we encountered various limitations in the processing power as well as the storage capabilities during the development.

In spite of this we still develop the necessary code to implement the deployment diagram that is above.

The diagram is composed of the client PC and Admin PC where the administrators and users interact with our application, and composed by the cloud service provider where it has all the components necessary for the workings of our project.

Chapter 4

Implementation

4.1 Presentation layer:

This layer includes the user interface and the admin interface. The user interface allows the final user to interact with the DETI Bot through a web app that consists of a simple chat with the DETI Bot. The admin interface enables administrators (those responsible for maintaining the data accessible by the DETI Bot) to introduce new knowledge into the system by filling out a form, as well as to modify knowledge sources to ensure the information remains updated and relevant.

For the presentation layer we used React on both interfaces. React has a component-based architecture and its efficient rendering makes it an ideal choice for building dynamic and interactive user interfaces. React allows developers to build large web applications that can alter data without requiring a page reload. The fundamental goal of React is to be quick, scalable, and simple.

4.2 Database Layer:

This layer comprises two databases: a vector store, Qdrant, and a relational database, MySQL, each serving different purposes. The Qdrant database is used to store the knowledge of the system. The MySQL database is used to store the metadata about the knowledge sources, allowing the business layer to update the data stored in the vector store automatically and providing admins with the information they need to manage the DETI Bot's knowledge. The database has five tables: *update_time*, *file_source*, *url_source*,*faq_source* and *url_child_source*. The *url_source* table contains metadata about all knowledge sources that are URLs, which are updated automatically due to their nature.

The *file_source* table stores all the metadata of knowledge sources that are files, these require manual updates by the admins.

The *faq_source* table stores a type of knowledge source that is composed of a question and it's corresponding answer, it allows for a more precise and predefined answer to the respective

question, this knowledge source is created by the admins or through the positive feedback of the user, these also require manual updates by the admins.

The *update_time* table stores four different rows, each representing the frequency and date of updates, used to help the system determine which rows in the *url_source* need to be updated at any given time.

Finally we have the *url_child_source* which is a table created to support the operations regarding the *url_source* table because there are a few knowledge sources of the url type that are recursive and have a path which means that through the metadata of that url other urls linked to it are also loaded.

4.3 Business Layer:

This is the heart of the project, containing all necessary methods to interact with the other layers and ensure the project functions correctly. It includes an API powered by FastAPI and a set of services powered by LangChain. We also use Ollama to provide an LLM to generate answers, along with a Python script that handles the MySQL storage process. The LangChain-powered services include loading, indexing, and querying. The loading service extracts content and metadata from knowledge sources based on their type (URL or file) and passes it to the indexing service. The indexing service stores this information in the vector store for later retrieval by the querying service. The querying service retrieves the most similar vectors from the vector store based on a given prompt and feeds them, along with the prompt, to the LLM to generate an accurate answer. The API handles requests from user and admin interfaces, calling the appropriate services based on the request.

1. **Gathering and managing knowledge sources:** This process starts with the admin interface and can occur in two ways: inserting a new knowledge source or editing an existing one.

    - **Inserting a new knowledge source:** This process begins in the admin interface when an administrator inserts a new knowledge source. After filling out the form with the correct information and submitting it, a POST request is received by the API. The storing script is called to store the information in the database, and the loading service loads the information and calls the indexing service, which indexes the content and metadata in the vector store, finalizing the process.

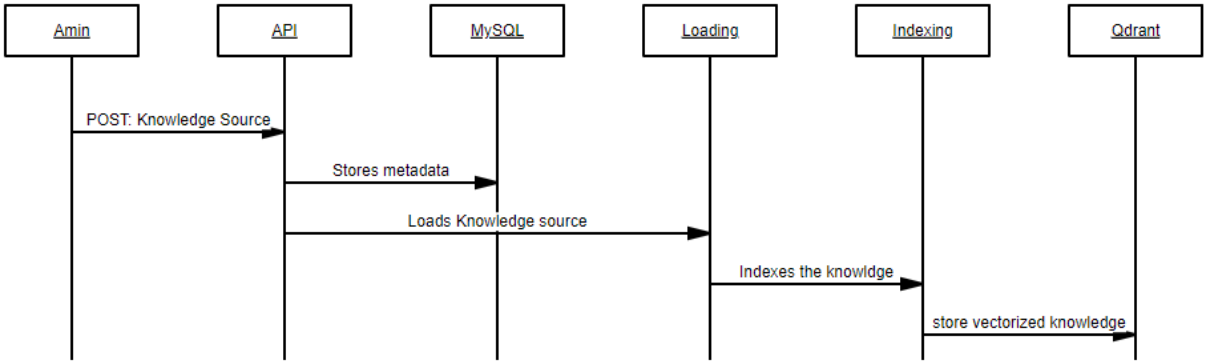

Figure 4.1: Inserts new Knowledge into the system

- **Editing an existing knowledge source:** In the admin interface, there is a page showing the metadata stored in the MySQL database. The admin can delete or modify this information. Deleting a knowledge source will remove it from both the MySQL database and the Qdrant vector store. Modifying a knowledge source updates it in MySQL, deletes the current vectors stored in Qdrant associated with that source and calls the loading service, which processes the modified knowledge source and updates it in the vector store as described above.
-

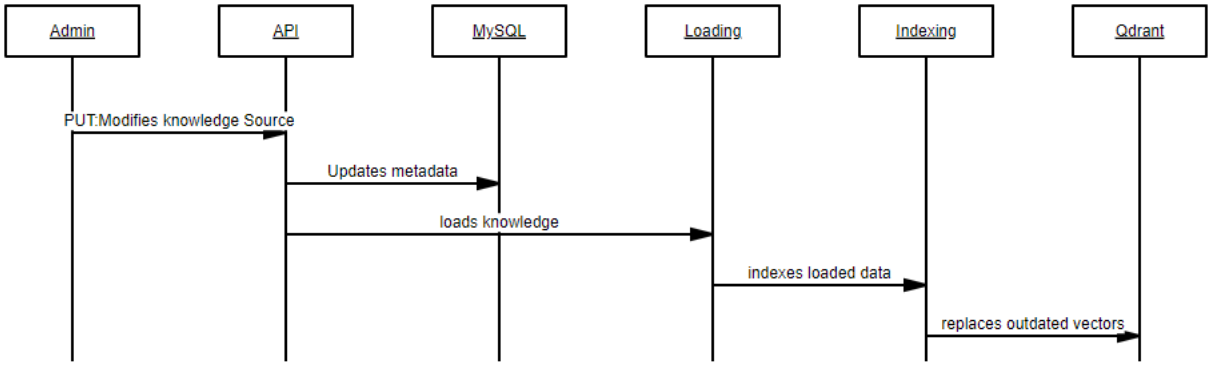

Figure 4.2: Modifies Knowledge Source

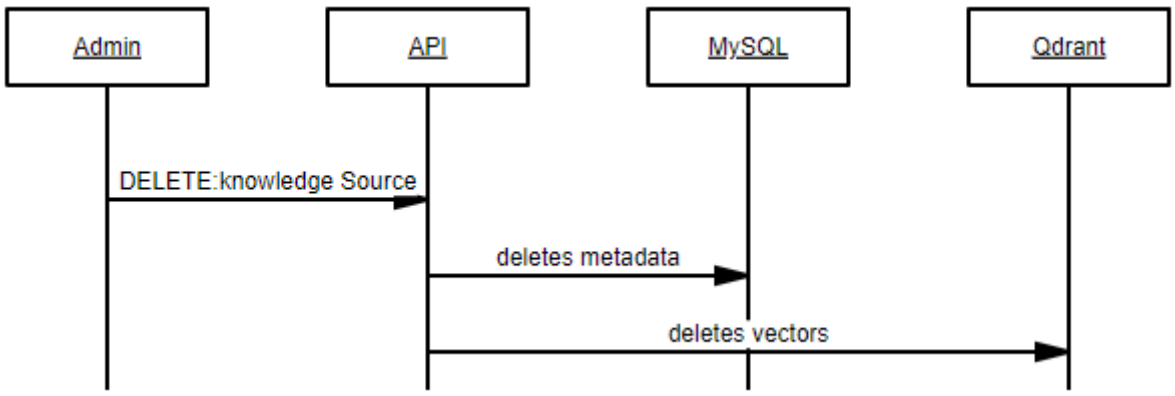

Figure 4.3: Deletes Knowledge Source

2. **Automated Update of URls:** This process involves knowledge sources in the *url_source* table, as explained in the Database Layer section. It occurs automatically at midnight daily, checking if any knowledge sources need updating based on the *update_time* table. If updates are needed, the vectors corresponding to that url are deleted as well as the child urls of that source, if they exist, and then the loading service processes the relevant knowledge sources, followed by the indexing service. After this is done through all the knowledge sources that needed updating, the *update_time* table is updates the rows that were involved in the process, completing the knowledge update.

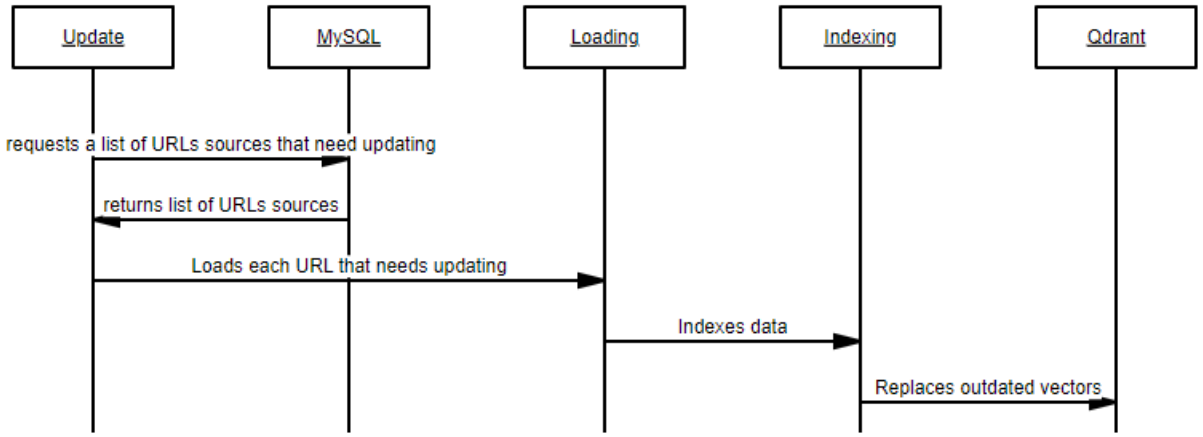

Figure 4.4: Automated update

3. **User interaction:** This process begins with the user interface when a user asks the bot a question. The question (or prompt) is sent to the API via a GET request. The querying service retrieves the four most similar vectors from the vector store and, with the help of a locally running Ollama server, generates an answer using the LLM. This answer is then returned to the user interface.

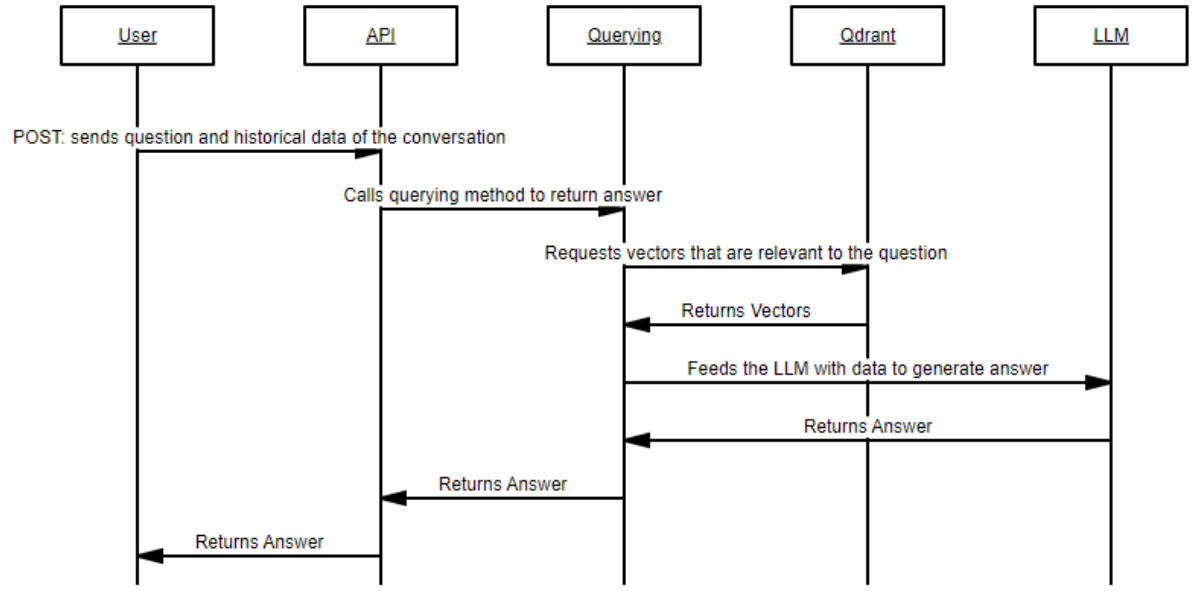

Figure 4.5: User Interaction

Chapter 5

Project management

Project management is crucial for software development teams as it ensures structured planning, efficient resource allocation, and adherence to timelines, ultimately leading to the successful delivery of high-quality software. It encompasses risk management, quality assurance, and effective communication, facilitating collaboration among team members and stakeholders. By managing scope, budget, and stakeholder expectations, project management helps prevent scope creep and financial overruns. It also promotes continuous improvement through lessons learned, ensuring projects are delivered on time, within budget, and to the desired standards, resulting in satisfied stakeholders and successful outcomes.

5.1 Team roles

In this project in These roles are essential for the successful development and delivery of high-

Inês Ferreira	Project Manager
Guilherme Lopes	Team Manager
Simão Antunes	Architect
Pedro Matos	DevOps
Diogo Gaitas	QA tester

Table 5.1: Team roles

quality software, each contributing their expertise to ensure the project runs smoothly and meets its objectives.

- **Project Manager:** The Project Manager plays a crucial role in overseeing the entire project from start to finish. They are responsible for setting clear project goals, creating detailed plans, allocating resources wisely, and establishing timelines. Effective communication is key, as they ensure that all stakeholders are aligned and informed throughout the project. The Project Manager monitors progress, manages risks, and addresses any issues that arise to keep the project on track and within budget.
- **Team Manager:** The Team Manager focuses on managing the development team, ensuring that team members are motivated, productive, and working well together towards

project objectives. They handle daily operations such as assigning tasks, evaluating performance, and resolving conflicts. Additionally, the Team Manager supports the professional development of team members, fostering a positive and efficient work environment.

- **Architect:** The Architect is responsible for designing the overall structure of the software system. They create high-level architectural plans that outline the technical framework and guide the development process. Ensuring the system meets standards for performance, scalability, and security is a key part of their role. The Architect collaborates with developers to solve complex technical problems, ensuring that the architectural vision is effectively implemented.
- **DevOps Engineer:** The DevOps Engineer bridges the gap between development and operations teams. They implement processes, tools, and methodologies to automate and streamline the software development lifecycle, including code integration, testing, deployment, and monitoring. DevOps aims to improve collaboration and efficiency, reduce deployment times, and ensure software reliability and scalability. Maintaining continuous integration and continuous delivery (CI/CD) pipelines is a critical aspect of their role.
- **QA Tester:** The Quality Assurance (QA) Tester ensures that the software meets quality standards before it is released. They design and execute test plans, create detailed test cases, and identify any defects. QA Testers perform various types of testing, including functional, performance, and security testing, to ensure the software works correctly and meets user requirements. They work closely with developers to reproduce and fix bugs, ensuring the final product is stable and user-friendly.

5.2 Project Calendar

For this calendar the team decided to use a Gantt Chart which was created predominantly in the begining of our project. The calendar is quite important as it provides a visual timeline that outlines key milestones, deadlines, and important events throughout the project's lifecycle. It helps team members and stakeholders stay organized and aligned by clearly communicating when tasks need to be completed and by whom. Additionally, a project calendar aids in identifying potential scheduling conflicts and ensures that resources are allocated efficiently, ultimately enhancing the project's overall coordination and productivity.

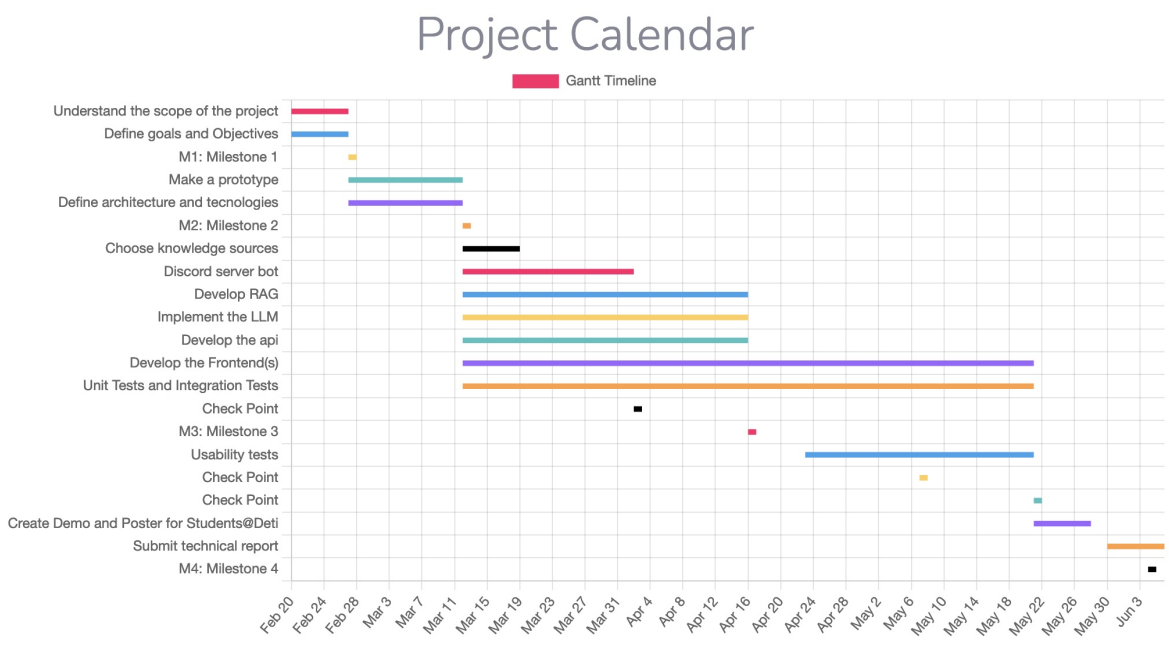

Figure 5.1: Project Calendar

The Project Calendar graph provides a detailed timeline and highlights the key milestones for our DETI bot development project, covering the period from February 20th to June 3rd. This timeline outlines the various stages and tasks necessary for the successful completion of our project.

Key phases and milestones for our project include:

- **Understanding the Scope of the Project**: Our initial phase involved comprehending the project's requirements to establish a solid foundation.
- **Defining Goals and Objectives**: We set clear goals and objectives to effectively guide our project.
- **Milestone 1 (M1)**: This early milestone marked our initial progress. The goal of this presentation was to outline our initial plans and set the goals we aimed to accomplish throughout the development of the bot.
- **Defining Architecture and Project Requirements**: In this phase, we focused on detailing the architecture of our project and specifying the system requirements, ensuring that all technical aspects were thoroughly planned.
- **Milestone 2 (M2)**: This significant checkpoint focused on defining the architecture of our project and discussing the system requirements in greater detail.
- **Selecting Knowledge Sources and Its Handling**: We chose knowledge sources, such as the DETI website, and analyzed how to index the information effectively.
- **Core Component Development**: Our team concentrated on developing the Retrieval-Augmented Generation (RAG) system, implementing the Language Learning Model (LLM), and developing the API and front-end interfaces.

- **Milestone 3 (M3)**: The goal of this presentation was to demonstrate the working prototype and show overall progress, including consistent answers from our bot.
- **Testing Phases**: We conducted unit tests, integration tests, and usability tests to ensure our software functions correctly.
- **Final Preparations and Deliverables**: We created a demo and poster for Students@Deti, submitted our technical report, and achieved the final milestone (M4), where we showcased the finalized demo and discussed our achieved goals and future ideas for the project.

The graph uses color-coded bars to show the duration and dependencies of each task, making it easy to see what needs to be done and when. This careful planning helped our team stay on track and complete the project successfully.

5.3 Communication Plan

During the development of the DETI bot, the team ensured consistent and effective communication through various channels. For informal discussions and day-to-day collaboration, we primarily utilized WhatsApp and Discord, which fostered a dynamic and interactive environment. On a more formal note, we maintained a structured line of communication with our supervisor via Slack, facilitating clear and organized exchanges. Additionally, we held weekly meetings through Zoom to discuss progress, address concerns, and align on project objectives with our supervisor. For managing project updates and ensuring the seamless integration of code, we relied on a GitHub repository. This platform proved to be the most efficient means for continuously updating and managing our codebase, allowing for real-time collaboration and version control.

Chapter 6

Quality Assurance

Quality assurance (QA) is a critical process that ensures that software meets specified requirements and is reliable. It involves a systematic approach to evaluating and improving the software development process, from initial requirements through to deployment and maintenance. This chapter describes the QA methods that we used throughout the project development.

User Feedback

Incorporating user feedback into the QA process helps ensure the software meets user's expectations and performs well in real-world scenarios. While developing the project, the group created a Google form that allowed the DETI community to share questions that they would like DETI Bot to answer correctly. Based on those answers, we were able to identify what were the main sources of information that needed to be part of the knowledge sources of DETI Bot. Furthermore, the user can evaluate every answer given by the bot so that the answers can be fine-tuned over time.

Testing

Firstly, we focused on building something that could give us some responses. We started using the LangChain framework with some LLMs (llama2, mistral, etc) and started testing the data extraction with some recommended embeddings from HuggingFace. From the responses we had from our GoogleForms we started testing the bot with different LLMs and embeddings. We followed the recommendation from our supervisor and used Llama2. Then we improved to Llama3 using the Ollama framework.

Testing is a fundamental component of the QA process in software development. It involves various methodologies to ensure that the software performs correctly and meets all specified requirements. The key types of testing included on this project are:

- Integration Testing: this stage focuses on verifying that different parts of the system interact and function together as expected. It identifies issues that may arise from the integration of various modules or components.
- System Testing: in this phase, the complete system is tested to ensure it meets the defined requirements.

- Acceptance Testing: this final stage of testing assesses the software from an end-user perspective. It verifies that the software meets business needs and user expectations, ensuring it is ready for deployment.

Together, these testing types help deliver a high-quality product that meets both technical and business needs.

Planning and Documentation

QA involves detailed planning and documentation that ensure that all aspects of the software are thoroughly examined. Documentation served as a foundational element that ensured consistency, traceability and clarity so that the group could be synchronized. Daily communication helped the group establishing objectives, allocating resources and managing timelines and risks. All these factors ultimately led to the delivery of high-quality software.

The group was initially divided in 2/3 sectors: frontend, backend and extracting information (web-scraping).

Continuous Improvement

QA emphasizes continuous improvement in processes and products. Feedback from testing is used to make iterative improvements, leading to higher quality software. The main factors that contributed to continuous improvement were the group's desire to do more combined with the ideas given by professors throughout the semester. This allowed DETI Bot to have new functionalities and improve existing ones.

From the moment we had the LLM installed and API established, the first phase was already completed. Then we focused on adding new functionalities and improving data extraction efficiency.

Overall, quality assurance is integral to delivering high-quality software that meets or exceeds user expectations while being reliable, secure, and efficient.

Chapter 7

Conclusions

Our project demanded us to put into practice the knowledge we gained throughout our undergraduate studies, emphasizing the importance of iterative and incremental development methods and the adoption of best software engineering practices. We're confident that our software solution will be a valuable addition to the DETI community, offering benefits to students, professors, and staff alike. This journey not only showcased our academic prowess but also highlighted our ability to tackle real-world challenges with innovative software solutions.

Future work

Several functionalities and improvements still need to be implemented to complete and enhance the project.

Firstly,the ability to ask questions via audio and hear the responses should be added. This involves integrating voice recognition and speech synthesis APIs, enhancing accessibility and user experience.

Another important improvement is adding a section with common questions on the chatbot's initial screen. This will facilitate quick access to common information without the need for typing. Implementing a robust authentication system using FastAPI for accessing the administrative zone is crucial. This system should include credential verification and the use of JWT tokens to maintain secure sessions, ensuring that only authorized users can perform administrative operations.

It is also essential to prevent SQL Injection attacks. Measures such as validating and sanitizing data inputs, using parameterized queries, and adhering to best security practices in database management should be implemented to protect stored data.

References

Langchain: Framework for developing applications powered by language models, 2024. URL https://www.langchain.com/.

Ollama: Open-source project for running large language models locally, 2024. URL https://ollama.com/.

Qdrant: Vector similarity search engine, 2024. URL https://qdrant.tech/.

Universidade de aveiro, 2024. URL https://www.ua.pt/.

S. Hykes. Docker: Lightweight and portable software containerization technology, 2013. URL https://www.docker.com/.

R. S. Patrick Lewis, Barlas Oguz. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020. URL https://arxiv.org/abs/2005.11401.

S. Ramírez. Fastapi: A high-performance, web framework for building apis with python. 2018. URL https://fastapi.tiangolo.com/.

J. Walke. React: A javascript library for building user interfaces, 2013. URL https://reactjs.org/.

M. Widenius, D. Axmark, and A. Larsson. *MySQL Reference Manual*. O'Reilly Media, 2002. URL https://dev.mysql.com/doc/refman/8.0/en/.