NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

TIAGO VALENTIM HENRIQUES

Bachelor in Computer Science and Engineering

LEVERAGING LARGE LANGUAGE MODELS TO AUTOMATE PROCESSES

BUILDING AUTOMATED PROCESSES FOR SKILLS WORKFLOW USING A LARGE LANGUAGE MODEL

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

LEVERAGING LARGE LANGUAGE MODELS TO AUTOMATE PROCESSES

BUILDING AUTOMATED PROCESSES FOR SKILLS WORKFLOW USING A LARGE LANGUAGE MODEL

TIAGO VALENTIM HENRIQUES

Bachelor in Computer Science and Engineering

Adviser: João Vieira
COO, Skills Workflow

Co-adviser: Sérgio Duarte
Assistant Professor, NOVA University Lisbon

Examination Committee

Chair: Doutor João Ricardo Viegas da Costa Seco
Prof. Associado, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Rapporteurs: Doutor Nuno Miguel Cavalheiro Marques
Prof. Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Eng. João Paulo Duque Vieira
COO, Skills Workflow, Lda.

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2024

Leveraging Large Language Models to Automate Processes
Building Automated Processes for Skills Workflow using a Large Language Model

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.18) [38].

Acknowledgements

I would like to express my deep gratitude to my supervisor, João Paulo Vieira and Bruno Moscão from Skills Workflow, for the support and guidance provided throughout this work, both in the preparation phase and in the elaboration of this dissertation. In addition to them, I would like to thank the entire team at Skills Workflow who accompanied me and helped me to complete this work successfully. To Professor Sérgio Duarte, I would like to express my sincere gratitude for the guidance provided. Finally, I would also like to thank my entire family for the support, motivation, and encouragement throughout this course, and to the friends with whom I could share this journey.

”

“What you get by achieving your goals is not as important as what you become by achieving your goals.”

— Zig Ziglar, See You at the Top
(American author, salesman, and motivational speaker)

Abstract

The evolution of programming paradigms has taught us to simplify processes by introducing solutions that increase and facilitate their use, such as visual languages. Likewise, in software applications, always look for a solution that benefits the user. This work proposes an innovative approach, "eliminating" the need to learn new UIs when using a new application. Intends to demonstrate that, in any system, it is possible to replace the UI-oriented use of the system with one that everyone already learned in their childhood: reading, writing, listening, and speaking. The dissertation demonstrates that this is possible on the Skills Workflow platform, and suggests that experimentation on various systems could prove the validity of this statement.

This study focuses on process automation on the Skills Workflow platform taking advantage of the use of Large Language Models (LLM). The aim, with the use of this emerging artificial intelligence technology, is to improve the user experience, presenting ways to increase productivity in software applications, in our case in Skills Workflow, automating tasks and providing contextualized responses to users, developing what we call an assistant. The Skills Workflow platform already has a vast marketplace and a User-friendly interface with a *No-code* philosophy, making it perfect for this objective.

We will explore the benefits of utilizing LLMs features to improve accessibility, usability, efficiency, productivity, and potentially reduce production costs within companies using the Skills Workflow platform as a test case. The focus will be on understanding user needs and challenges on the platform while identifying the assistant's capabilities such as analyzing application data and exploring configuration changes through natural language commands, or notifications summaries. Additionally, we will be integrating a LLM into an AI assistant prototype and exploring automation features for existing workflows in Skills Workflow. Users will interact with the platform using natural language, and the system will then execute the necessary operations based on the interpreted requests.

Keywords: Large Language Models Integration, Neuro-Symbolic AI Architecture, AI Assitant.

Resumo

A evolução dos paradigmas de programação ensinou-nos a simplificar processos introduzindo soluções que aumentam e facilitam a sua utilização, como as linguagens visuais. Da mesma forma, nas aplicações de software, procure sempre uma solução que beneficie o utilizador. Este trabalho propõe uma abordagem inovadora, "eliminando"a necessidade de aprender novas UIs quando se utiliza uma nova aplicação. Pretende demonstrar que, em qualquer sistema, é possível substituir a utilização do sistema orientado para a UI por outro que todos já aprenderam na infância: ler, escrever, ouvir e falar. A dissertação demonstra que isto é possível na plataforma Skills Workflow, e sugere que a experimentação em vários sistemas pode provar a validade desta afirmação.

Este estudo foca-se na automatização de processos na plataforma Skills Workflow aproveitando a utilização de Large Language Models (LLM). O objetivo, com a utilização desta tecnologia emergente de inteligência artificial, é melhorar a experiência do utilizador, apresentando formas de aumentar a produtividade em aplicações de software, no nosso caso em Skills Workflow, automatizando tarefas e fornecendo respostas contextualizadas aos utilizadores, desenvolvendo aquilo a que chamamos de assistente. A plataforma Skills Workflow conta já com um vasto marketplace e uma interface User-friendly com uma filosofia *No-code*, tornando-a perfeita para este objetivo.

Exploraremos os benefícios da utilização das capacidades do LLM para melhorar a acessibilidade, usabilidade, eficiência, produtividade e, potencialmente, reduzir os custos de produção nas empresas que utilizam a plataforma Skills Workflow como caso de teste. O foco estará na compreensão das necessidades e desafios do utilizador na plataforma, ao mesmo tempo que identifica as capacidades do assistente, como analisar dados de aplicações e explorar alterações de configuração através de comandos de linguagem natural ou resumos de notificações. Além disso, iremos integrar um LLM num protótipo de assistente AI e explorar as capacidades de automatização para os fluxos de trabalho existentes no Skills Workflow. Os utilizadores irão interagir com a plataforma em linguagem natural e o sistema executará as operações necessárias com base nos pedidos interpretados.

Palavras-chave: Integração de Grandes Modelos de Linguagem, IA Neuro-Simbólica,

Assistente de IA.

Contents

List of Figures

List of Tables

Glossary

Caixa Geral de Depósitos sort — Caixa Geral de Depósitos (CGD) is a Portuguese state-owned banking corporation and the second largest bank in Portugal. *(pp. 26, 27)*

Airtable Cobuilder — A no-code app creation tool that leverages AI to help users build custom applications. *(pp. x, 24, 25)*

Be My Eyes — An application that connects blind and low-vision individuals with sighted volunteers and company representatives for visual assistance via live video calls. *(pp. 26, 27)*

Chatbot — A software application designed to simulate human conversation through text or voice interactions. *(pp. 8, 21)*

ChatGPT — ChatGPT is a chatbot developed by OpenAI that enables users to refine and steer conversations towards a desired length, format, style, level of detail, and language. *(p. 1)*

Gemini Live — An AI project by Google focused on enhancing mobile experiences. *(p. 24)*

Integrated Development Environment — An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities for software development. *(p. 1)*

Legacy Systems Legacy Systems are older software or hardware systems that remain in use despite their age, often because they still provide significant value to their owners. *(p. 2)*

No-code A software development approach that uses graphical interfaces, such as drag-and-drop tools, to generate code. *(pp. iv, v, 2, 24)*

Project Astra An advanced AI initiative by Google, designed to be a universal AI assistant. *(p. 24)*

Prompt An instruction or set of instructions given to a large language model (LLM) to generate a response or perform a specific task. *(pp. 4, 10, 12, 22, 23, 28, 29, 33, 36, 37, 40–47, 50–54, 56–58, 60, 75–77)*

REST Representational State Transfer (REST) is an architectural style for designing networked applications. *(pp. 12, 75)*

Skills Workflow Skills Workflow is a platform enabling the creation of micro-applications and solutions across various domains without requiring advanced programming knowledge. *(pp. iv, v, 2–4, 6, 7, 12, 25, 26, 28, 30, 31, 38–40, 43, 47, 52, 54, 75, 76)*

Stripe A technology company that provides payment processing services for both small and large businesses across the internet. *(pp. 25, 27)*

Transformers A deep learning architecture that relies on a parallel multi-head attention mechanism, commonly used in natural language processing. *(p. 9)*

User-friendly User-friendly refers to something that is easy to learn, use, or understand, particularly in relation to computers. *(pp. iv, v, 76)*

Web-based Interface A user interface that is accessed through a web browser, allowing users to interact with software applications over the internet. *(p. 29)*

Acronyms

AI Artificial Intelligence *(pp. iv, v, 1, 2, 4, 12–15, 22–25, 37, 60, 64, 66, 67, 75, 76)*

API Application Programming Interface *(pp. 7, 9, 12, 15, 16, 37, 40, 41, 48, 75)*

BPM Business Process Management *(p. 6)*

GPT Generative Pre-trained Transformer *(pp. 17, 25–27)*

JSON JavaScript Object Notation *(pp. 7, 19, 30–32, 37–39, 44–48, 50–54, 56–58, 76, 77, 85)*

LLM Large Language Model *(pp. iv, v, xii, 1–4, 6, 9–11, 13, 14, 16–18, 20–22, 24–33, 37–58, 75–77, 85)*

NLP Natural Language processing *(pp. 21, 25)*

RAG Retrieval Augmented Generation *(pp. x, 12, 48, 49, 53, 54, 76, 85)*

SDK Software Development Kit *(pp. 8, 12, 22, 31, 32, 39, 40, 43, 47, 52, 54, 75)*

UI User Interface *(pp. iv, v, 46, 70, 74)*

1

Introduction

For some time now, people have been trying to simplify programming by using metaphors and visual languages so that anyone can program or make the entire creation process faster. Initially, all programming tasks were performed through command lines, and then user interfaces with icons and structured menus began to emerge. At the same time, programming has been simplified through visual languages that are extremely accessible to everyone, but also through Integrated Development Environments that seek to detect errors, whether in syntax highlighting or code completion. Nowadays, Artificial Intelligence (AI) is a resource for simplification. There are already practices that improve the way users interact with applications, offering a more intuitive experience. Personal assistants such as Apple's *Siri* and Google's *Alexa* began to appear, capable of solving simple everyday tasks, such as answering questions using the internet by voice, asking people to write and send emails, or turning lights on and off at home. Facial recognition systems that provide fast and secure authentication methods. Or even helping people with disabilities navigate the digital world. Now, with the emergence of Large Scale Language Models (LLMs), like the one used in ChatGPT, we are witnessing a trend of integrating these assistants into most applications, like Bing Copilot [24], which can generate, understand, translate, and create code and text content.

With Large Language Model (LLM)s, new opportunities emerged that significantly increased the productivity level of existing software products. By integrating this technology into today's applications, it is possible that by sending data from the application to the language model, it can contextualize it and provide code or answers to questions about that data. Where previously it was necessary to perform a task manually, it is now possible to automate and simplify processes with commands given in our natural language, saving time for those who use the applications and, in many cases, saving financial resources. However, natural language is not very rigorous and it can be difficult to capture user intent. If a AI assistant needs frequent corrections, it risks becoming a source of frustration, and instead of the product improving, it may start receiving negative feedback. For example, Office Assistant was eventually discontinued due to its intrusive and annoying behavior, receiving negative reviews [73]. These are just some of the things we need to consider

when developing an AI assistant.

1.1 Context

In many applications, we have noticed that certain processes can be automated, or basic tasks simple but time-consuming for the user. Keeping this in mind, it is crucial for software applications to prioritize providing the best user experience. One of the main concerns is the learning curve of the user interface, as it can often be slow and time-consuming in the beginning. Therefore, we will address this challenge by applying it to the Skills Workflow.

Introducing the Skills Workflow platform, an application aimed at helping agencies and studios optimize their workflow. This release is deployed with a cloud-based microservices architecture. It already embraces a compelling philosophy of *No-code*, which makes it easy to create micro-applications without requiring users to have programming experience. The platform also has an extensive marketplace comprising an abundance of applications, from time tracking to vacation management, budgeting, proposals, and project management, among many others.

An AI assistant could be a valuable addition to Skills Workflow, as it could assist users with various tasks, such as creating and managing projects, tasks, and budgets, or even accessing and analyzing data from the platform. Helping users automate repetitive or tedious tasks, such as filling out forms, creating components such as adding a new task into a project, or getting faster answers, like searching for the project that has the most numbers of employees involved. Skills Workflow has a large customer base, each with different requirements, and imagine their product evolving into software capable of being used by anyone with the fastest possible workflow. To fulfill those requirements, Skills already offers an attractive way to let clients develop their applications in a versatile manner, using *No-code* tools and disposing of their huge marketplace from where the clients can choose the applications that best fit their necessities. So, we can see an assistant would meet the company's values, bringing advantages such as automation of processes; user accessibility with an even better user-friendly interface; saving in operating costs (may need to involve fewer people in the process); faster answers to our client's clients; and help users who cannot program or do not know all the platform possibilities. However, we can predict some limitations that exist, such as the potential for human error in the user instructions; in the beginning, the users may face a learning curve before getting a huge boost in performance; it is needed to ensure cost efficiency of the solution using the LLMs; and it will probably be necessary to take into account continuous updates, given the growth of LLMs.

We believe that software products, which do not start thinking about a solution to improve user interaction with their application by automating processes and reducing the beginning learning curve will become the latest Legacy Systems because even though they will probably remain valuable to their users for a long time, as this new technology

becomes common, users' expectations and needs begin to change. If an application cannot keep up with the user's needs, it will become less useful, difficult to learn, and eventually obsolete.

1.2 Objective and contributions

The primary goal of this dissertation is to reduce the learning curve for user interfaces and increase user productivity in applications, by creating an assistant that can automate processes in the Skills Workflow platform, leveraging the LLMs. An assistant will work alongside the user, aiming for operational efficiency, automating routine tasks, and allowing users to focus on more specific and creative activities.

This assistant will help the user in tasks where he may experience some difficulties or where he wastes precious time performing basic and monotonous tasks. In these cases, the assistant will facilitate the process and, more importantly, in a way so intuitive that the user does not need to learn how to use it.

Exploring the user tasks and difficulties, we aim to create an assistant in Skills Workflow platform that is connected to the application data and capable of automating processes in the platform, namely manual user tasks. We intend that by demonstrating how to improve applications workflow by automating processes in Skills Workflow, a kind of recipe to reasoning how to build assistants with LLM be expressed. The object of study will ideally be a set of decisions and technologies that together allow the construction of a complete assistant. This work proposes an innovative approach, eliminating the need to learn new user interfaces when using a new application. It is intended to demonstrate that, in any system, it is possible to replace the user interface-oriented use of the system with a user interface-oriented use that everyone already learned in their childhood: reading, writing, listening, and speaking. The dissertation demonstrates that this is possible on the Skills Workflow platform, and suggests that experimentation on various systems could prove the validity of this assertion.

To achieve an assistant, we will harness the capabilities of powerful LLMs, which appear as a solution for this problem, for having an incredible capability of understanding and communication. The user will communicate with the assistant using an input or voice to transmit what they want to do. It is important to note that the LLM is not the primary focus of this work, but rather the assistant itself and its capability of serving the user in several tasks. The LLM is an innovative and efficient technology, which can overcome some of the limitations of traditional methods for natural language processing and generation. Following this line of reasoning, we are promoting to enhance the user experience by adding workflows that achieve tasks after interpreting the user request, significantly increasing user efficiency, productivity, and accessibility.

The assistant can have characteristics such as generating alerts, building dashboards, searching system data, or even introducing new content explaining what is intended or referencing existing content. It should be able to understand the user intention and lead

the system to perform some action in its data. In the state-of-the-art, first, we have selected the best LLM to integrate and manipulate, and then find a way to incorporate that model into the assistant, identifying features that could be useful to users of the platform. We thought about the principal user needs in our platform, comprehending what can be automated and whether they are completely new use cases or improvements to current application features. Once we have identified the features, we also explore a conceivable pipeline, highlighting the advantages, to build our assistant. In the development phase, we implement some of these features and create an automated assistant to Skills Workflow. This study presents challenges, such as discovering if it is achievable to create an assistant that is not intrusive or frustrating to perform tasks, which will only drive away the users. So, let's research the most viable solution to build an assistant using a LLM, in addition to discovering the most advantageous set of features that best adapt to the platform and how to accomplish them accurately and persistently.

In summary, the main contributions of this dissertation are:

- Creation of a well-functioning architecture to implement an assistant leveraging a LLM.
- Development and implementation of an AI assistant prototype with some concrete features interesting for Skills Workflow
- Creation of semantical Prompts to achieve specific results.
- Feedback from the clients related to the feature's utility (client tests).
- Discoveries and methodologies can be used as a foundation for future work in this field by others interested in the same subject. Explore our proposed features to automate processes using a LLM.

1.3 Document structure

This document is organized into six chapters, each of which explores a different aspect of how to develop an assistant for Skills Workflow using Large Language Models (LLMs).

The current chapter (1) introduces the context, motivation, objectives, and contributions of the study.

The second chapter (2) reviews the related work and background information on assistants, LLMs, Skills Workflow features, and their application areas, but also compares and evaluates different LLMs based on performance, quality, and pricing, presenting some experiments we execute to test their capabilities.

The third chapter (3) covers the details regarding the architecture of the proposed implementation for the automation's we are developing in Skills Workflow.

The fourth chapter (4) is dedicated to the details of the assistant implementation, specifying how each functionality has been built, and showing how it works.

The fifth chapter (5) discusses the testing and evaluation of the assistant, presenting the results of the experiments and the feedback from client tests.

Lastly, in the sixth chapter (6), we present the conclusion and outline future work for this dissertation.

2

Background and Related Work

In this section, we present the basic concepts and theories foundational to support our work, providing the necessary background to build a problem-solving approach. Additionally, we include a literature review that explores the applications of Large Language Models LLMs in developing automated assistants on software platforms, highlighting their relevance to the research.

2.1 Skills Workflow

Skills Workflow [74] is a company that with its platform helps other companies manage their projects, resources, invoicing, employees, leaves, etc. It is a Business Process Management (BPM) tool, a type of application used to assist managers in decision-making, drawing up action plans, and creating strategies.

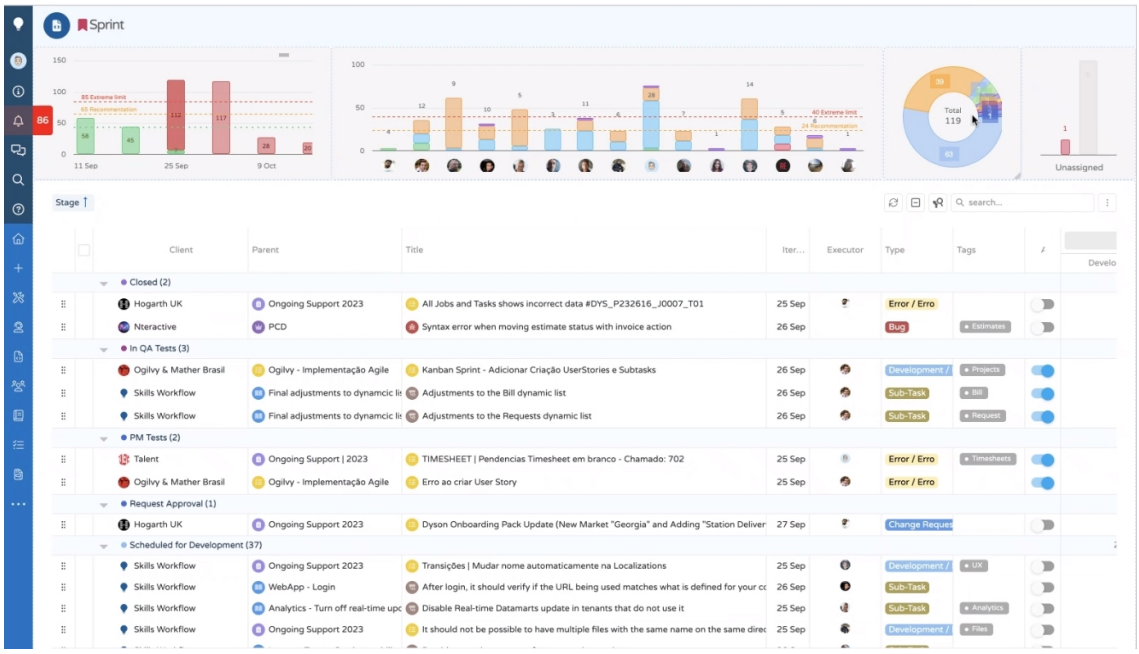


Figure 2.1: Skills Workflow example.

Through the example in figure 2.1, we can understand the functioning of the platform by looking at a dashboard that is managing a company sprint. It shows the current tasks done and to do in each category, the person a task is attributed to, or even the status of the number of tasks for each employee. To contextualize, each dashboard corresponds to a micro-application capable of a more specific objective, like the example in the figure.

Skills Workflow allows users to fully customize their dashboards/workspaces with all the different features, adding the best graphic or changing it to a table. The dashboard in the platform is the most important component describing all the smaller components that appear on the screen in an JSON object (e.g., calendar, table, chart, etc), which can also have some functions to manipulate the dashboard data (e.g., to filter something). With the existing functionalities, the client can create a dashboard with many different components suited to their needs. However, he has to do it manually, dragging widgets and making other customizations, instead of simply being able to say, "I want a workspace that shows all employees and the respective holidays that have already been enjoyed or have yet to be enjoyed by everyone.", and the assistant automatically creates the workspace. Similarly, the user may want to know which employees integrated into project "x" are going on vacation in April, having to go look for them instead of the assistant responding to him almost instantly with the names. Or even if the client wants to know if someone on a project has more than seventy hours of work per sprint iteration and has to check it constantly instead of asking the assistant to check this twice per sprint with a notification. Here are some examples of how the assistant can help the user, giving him more time to focus on important tasks.

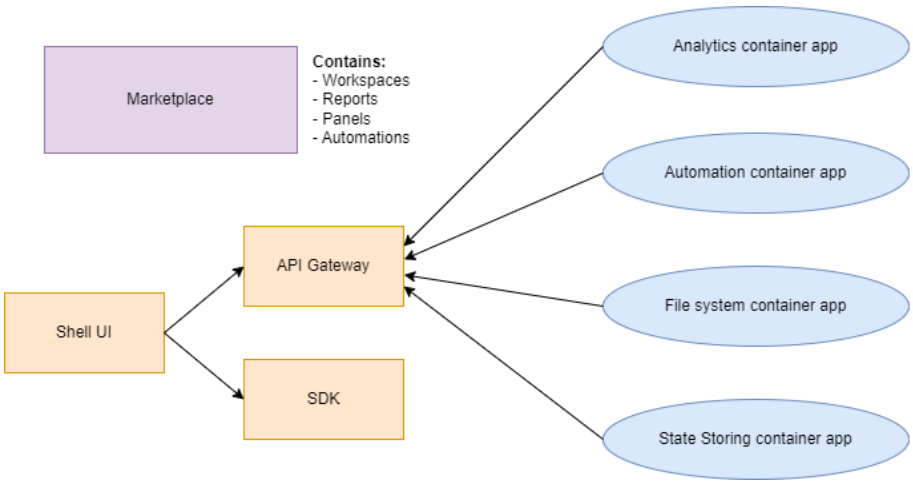


Figure 2.2: Skills Workflow architecture.

As shown in figure 2.2 it is a microservices architecture that disposes in each container a different service. All of them have an API connected to the Gateway API, that acts as an intermediary between clients and backend services. It handles all incoming API requests, routes them to the appropriate service, and then returns the response to the client. The objective of Skills Workflow is to reduce task time and costs for companies

that use its platform. By using resources efficiently, tasks can be completed more quickly and inexpensively. The use of containerized applications for scalability and portability. Automation has a bus queue that provides chat and notification functionality. An analytics service to provide data visualization and business intelligence. The state storing saves each user platform state, like filters or most search words used. Finally, the file system saves all system files. The Shell has an SDK that provides the necessary tools to manage the platform, like creating a new dashboard, getting some time data from a specific data source, and so on.

2.2 What is an Assistant?

The definition of assistant in the dictionary refers to someone who offers assistance, help, or a service. If the assistant is integrated into the software, it will offer intelligence capable of providing support and guidance in user tasks. In software, assistants can be used for different purposes, such as answering questions, performing actions, and automating processes, but focus on the objective of enhancing user experience and capable of communicating with the user clearly and naturally. Assistants can interact with users or applications through voice, text, or a user interface.

An assistant can have similar names like bots, Chatbots, virtual assistants, and assistants. However, these terms are not always interchangeable and may have different meanings or functions depending on the context. For example, a bot can update application data, index digital information, or perform customer service. Chatbots are more focused on maintaining conversational interactions. Virtual assistants are more used to routine support, being more comprehensive about actions. For us, an assistant should be more focused on acting on behalf of a user, being comprehensive, and combining all of the mentioned before. We identify three categories of assistants based on their functions: assistants who communicate with the user, assistants who collaborate with the user, and assistants who can respond to system events. So, our assistant should be a hybrid version of these three categories, where it will be able to listen and respond to the user based on the platform's data, make suggestions, help the user complete certain tasks faster, and respond to some system events without user involvement [31].

The intention is to be careful to think of the assistant as if it were a human being, trying to predict how the user thinks and be in agreement with their ideas. A human assistant sometimes works by guessing certain steps in a conversation, predicting what will be asked or said next, and it would be interesting if the assistant were able to accomplish similar things automatically without being asked.

2.2.1 Assistant developed with Large Language Models

The traditional type of existing assistants, such as Siri from Apple, or Google Assistant, are the ones who use predefined commands and rules to perform some tasks but have a

limited range of responses and actions. Sometimes, they fail when dealing with complex user commands. They are more commonly seen with IoT applications [47][11].

On the other hand, intelligent assistants built with the new LLM, can understand and generate texts, analyze data, or even generate code. In addition to their vast knowledge through information displayed on the internet, they can be trained for specific cases and generate even better quality responses.

2.3 Large Language Models (LLMs)

A language model is an algorithm that attempts to generate plausible language by predicting the next word to be written and learning the structure and meaning of the language from textual data. We can see it is a machine that repeats words it has heard with some understanding of their context. At a low level, a model has a well-defined and limited objective, while at a higher level, a model can learn and adapt to interact flexibly [36][72]. The difference from the traditional language models is that they are based on rules and linguistic characteristics developed by humans, having a limited understanding with smaller context windows to predict words. They are more used to specific tasks, being less versatile in different contexts [60].

There are different models for text, video, and image, but they can rely on LLMs to process and generate text. LLMs are trained on a large amount of unstructured data from the Internet, where this training is a process where the parameters of a model are adjusted to minimize the error between predictions and actual data, determining how a neural network processes the input and produces the output data [46]. Before it responds, it first needs to transform the request given by the user into tokens that it can understand. Tokens are sets of characters with some meaning, but for better understanding, we can think of them as words. LLMs then observes each token in context with the other tokens in the input sequence and takes note of the nearby tokens, leaving us with a set of words mapped to word embedding vectors (numerical representation of a token, capturing its meaning, semantics, and relationship with other tokens). Two similar embedding vectors mean two tokens with similar meanings. To make the LLMs "smart" it used the Transformers architecture with a deep learning technique called self-attention. The self-attention mechanism computes the similarity between each pair of tokens in the input sequence, giving it scores for token pairs and creating a weighted representation of each token based on the similarity scores. Self-attention will allow the LLMs to take context from sentences and generate an answer predicting word after word [65][45].

To use the LLM in applications, most companies expose an API that enables developers to interact with the LLMs and their features. For instance, we can access endpoints to turn audio into text or text into audio, to give chat completions, or even give an input image. All these endpoints will influence the possible functionalities our assistant can have.

2.3.1 Vulnerabilities of LLMs

As Mira Murati, Chief Technology Officer at OpenAI, said, "AI systems are becoming a part of everyday life. The key is to ensure that these machines are aligned with human intentions and values." This statement shows the importance of addressing the vulnerabilities of large language models (LLMs). Unlike search engines, LLMs predict a sample from a probability distribution over possible words, leading to challenges such as rare unseen words, overfitting to specific domains, or capturing complex linguistic phenomena.

One of the biggest problems LLMs faces is a process called “hallucination,” which results in incoherent or meaningless content. Hallucinations can occur due to insufficient or noisy training data, poorly formulated Prompts, ambiguity, or complex linguistic phenomena. To avoid hallucinations, models must have verification and validation mechanisms to detect inconsistencies in the generated text.

Other vulnerabilities related to the use of LLMs include security and privacy issues, as we send our code and data to external services. Additionally, bias problems can arise because models are trained on internet data. To mitigate these issues, LLMs should have content moderation policies and similar safeguards for input data.

2.3.2 Fine-tuning a LLM

Tuning a model involves adjusting a LLM to enhance its performance for a specific task or domain. Pre-trained models are already powerful but may not perform optimally for some specific use cases that require a particular answer. With additional training, we can enhance the normal performance and accuracy of the LLM for our desired application. Therefore, fine-tuning a model will provide higher-quality results for the purpose it has trained and probably save tokens due to requiring shorter Prompts [50]. Some common use cases where fine-tuning can improve results are [48]:

- Setting the style, tone, format, or other qualitative aspects.
- Improving reliability at producing a desired output.
- Correcting failures to follow complex Prompts.
- Handling many edge cases in specific ways.
- Performing a new skill or task that’s hard to articulate in a Prompt.

> **Note:** Hugging Face is a platform that has some pre-done models to train the LLMs. For example, a dataset of TypeScript snippets [68], to train the model to generate code in TypeScript.

2.3.3 Prompt engineering with LLMs

Prompt engineering is a very important step to get better outputs from the LLMs. We have tried and searched for some good ways of using prompt engineering to get better

results. This may vary from model to model but there are some tactics like the ones given by OpenAI [53]:

- Use delimiters to indicate distinct parts of the input.
- Specify the steps required to complete a task.
- Specify if we want a delimited output giving it the maximum length (e.g., 100 words).
- Use just provided articles delimited by triple quotes to answer questions. If the answer cannot be found in the articles, write "I could not find an answer.".
- Use code execution to perform more accurate calculations or call external APIs (e.g., write and execute Python code by enclosing it in triple backticks """code goes here""". Use this to perform calculations.).

Prompt engineering is basically what the user needs to do to communicate with the Large Language Model in the best way possible. One of the difficulties for us will be getting output only with the response, because when using this LLMs, normally they give, besides the concrete answer, some context, like an introduction or some context explanation.

2.3.4 Open-source vs proprietary models

Open-source LLMs allow anyone to access the model, modify, and distribute the source code and architecture. These language models are free, promoting collaboration and innovation among users, allowing them more control over the model. The downside is related to the fact they are vulnerable to community exploitation and depend on it for support, updates, and maintenance. On the other hand, proprietary models are owned by a company and protected by copyright and patents. They are paid and have integration and customization options limited by the company. However, performance and quality are guaranteed by the company, which may provide technical support if necessary. Therefore, we are more focused on the models that offer constant updates, support, and maintenance [63][30].

2.3.5 Models creators

There are several companies that have developed LLMs, but the most popular are OpenAI, Google, Meta, Anthropic, and Mistral AI. So, the next sub-sections dive deeper into those models and some of their characteristics.

2.3.5.1 OpenAI

OpenAI is a well-known creator of large language models (LLMs). GPT-3, one of the most influential models in the early days, gained attention due to its ability to generate human-like text, perform various language tasks, and its massive size of 175 billion parameters [56].

OpenAI has developed several models, including the most recognized: GPT-4, GPT-4 Turbo, GPT-3.5 Turbo, GPT-4o, and GPT-4o mini. All of their models are proprietary

and not open-source, meaning they are not available for modification or distribution [71]. OpenAI provides an API for developers to interact with their models, offering a secure and reliable service. However, Azure Microsoft also has available support for integrating some of the OpenAI models, which can be one of the reasons this can be a pleasing option for us is because the Skills Workflow platform is deployed using Azure services and OpenAI is a service in the Azure cloud platform, which is a significant advantage because it is designed thinking of enterprise customers and their requirements of production, such as availability, security, or privacy. In addition to that, Azure provides an option to chat responses about business data with the help of the Azure Cognitive Search index, and using the OpenAI models to produce the output based on that factual or contextual information from Skills Workflow stored data [14]. This approach is based on what they call the RAG Pattern, which basically performs a search on the database to find relevant information, then passes the result to construct a Prompt sent to Azure OpenAI, and finally displays that result. We can access Azure OpenAI services in several ways, from REST APIs, and web-based interfaces, to Python SDK in the Azure OpenAI Studio. The Azure offers a positive solution in our understanding, because it has private networking and regional availability, making it a very reliable solution. However, for now, they have some boundaries in some of the services provided that are still in preview [41]. In terms of Data and security, Azure ensures that no one but the customer has access to customer data, not being shared with third parties. Regarding content filtering, Microsoft evaluates Prompts and completions with high-severity content because in the models trained from internet content, social bias is always present, and Microsoft tries to promote a responsible use of AI.

Also, it is important to mention that OpenAI has Text-to-Speech (TTS), Speech-To-Text (Whisper model), Text-to-Image (DALLE), and Text-To-Video (Sora) models for use, and they monitor inputs and have some moderation policies categorized in hate, harassment, threatening self-harm (intended and instructions), sexual, and violence. It is an important point as it provides us with security when using the model to prevent misuse of the technology [52]. Furthermore, OpenAI says they don't make use of the data sent to the OpenAI API unless we permit it [51]. GPT-4o, for example, has safety built-in by design across modalities, through techniques such as filtering training data and refining the model's behavior through post-training [33].

Regarding the model's performance, in section 2.3.6 a comparison between some models is made.

2.3.5.2 Google

Google has developed several models, including the more talked about Gemini Pro, a proprietary multimodal model capable of handling text, images, audio, and video, and Gemma 2, an open-source model available in different parameter sizes. They provide an API for interacting with their models, making it accessible for developers to integrate

these models into various applications. Also, the models have fine-tuning support for various data types, including text, images, and audio. However, fine-tuning for video data is currently limited. One of the advantages of using Google's models is their integration with Google Cloud services, which ensures high availability, security, and scalability. This is particularly beneficial for enterprise customers who require reliable and secure solutions for their production environments [62][28].

In their chat, Bard is running the Gemini language model. Bard chat is running the Pro version, which is currently with an estimated 130-250 billion parameters. The Ultra version is the most powerful, with an estimated not confirmed 1.56 trillion parameters, and even though it may be a little smaller than GPT-4, it does not mean it is the worst, but sometimes rather beneficial, reducing training costs, improving efficiency, and reducing latency [27]. After searching about the Google Bard model, we found that with the PaLM 2 (previous version of Bard) Google has spent some money to generate multiple response drafts. But they are doing this with a purpose, to collect data for fine-tuning Gemini, which may be a good idea to improve their model [61]. Bard AI, with the Gemini model, also has (besides Speech-To-Text and image models) an incredible feature still missing in OpenAI, which is video understanding. This feature can open up another range of possible use cases we can implement on our platform.

Concerning the collection of information Google in their chat (Bard) gathers through conversations, information about the use of related products, our location, and feedback [29]. The collection of conversations and information about related products we don't see as something catastrophic, understanding that it helps to improve the model. However, regarding location collection, we've always been a little confused, because even though it improves the user experience, we don't think it should be mandatory for using Bard. After all, it always ends up violating our privacy directly by exposing our location.

Regarding the model's performance, in section 2.3.6 a comparison between some models is made.

2.3.5.3 Meta

Meta has made significant developments in large language models (LLMs), based on the transformer architecture and operates using auto-regressive techniques. Their LLaMA series models, for example, the LLaMA 3.1, with 405 billion parameters [5]. Meta's approach to LLMs is unique in the sense they offer mostly open-source models. The models are designed to handle a wide range of tasks, including text generation, summarization, and question-answering. They also support multiple languages, enhancing their usability in diverse linguistic contexts.

The LLaMA models are available to researchers and developers, allowing for modification and distribution, which contrasts mostly with OpenAI's proprietary models. We believe this openness is intended to foster innovation and collaboration within the AI community [39]. Where regarding data privacy and security, Meta ensures that their

models are trained on publicly available datasets and that user data is handled with care. Also, they have implemented measures to prevent misuse and ensure responsible AI usage [39][40].

According to the paper written by Meta [66], their model is an open-source model optimized for dialogue use cases. Which can be good for encouraging community development, innovation, and educational opportunities. However, being a model for dialogue, it was not developed to offer better performance in other fields we are looking for, related to code generation and/or code interpretation, for example. Also, the model needs to be installed on the local machine.

Meta's models are integrated with their extensive infrastructure, leveraging thousands of GPUs to handle the massive computational requirements for training these models. They have focused on optimizing their hardware and software to ensure efficient and reliable training processes [34]. This shows us that we can trust in their models, as they have support.

2.3.5.4 Anthropic

Anthropic has developed a series of advanced large language models (LLMs) under the name Claude. The latest models in this series include Claude 3 Haiku, Sonnet, and Opus, each offering varying levels of performance. These models are designed to handle text and image inputs [6].

Anthropic's models are built with a focus on AI safety and interoperability. They use a technique called "Constitutional AI" to train their models, which aims to make the models more aligned with human values. This approach helps in reducing harmful outputs and improving the reliability of the models [16].

In terms of data privacy, Anthropic ensures that its models are trained exclusively on publicly available datasets, and they handle user data with stringent privacy measures. They are committed to GDPR compliance and have robust protocols in place to protect user data, ensuring it is stored securely and accessed only by authorized personnel. Additionally, Anthropic has implemented mechanisms to prevent misuse and promote responsible AI usage, reinforcing their dedication to data privacy and security [8][7].

The Claude models are integrated with major cloud platforms like AWS, ensuring high availability and scalability for enterprise applications. They also support multilingual capabilities, enhancing their usability across different languages and regions [35].

2.3.5.5 Mistral

Mistral AI has made significant advancements with their LLMs, particularly in the release of Mistral Large 2 with 123 billion parameters and supporting several natural and coding languages. Mistral Large 2 is designed for single-node inference, which allows for efficient and scalable deployment [44].

Mistral's models are available for free if only for research and non-commercial use. Otherwise, licensing is required for commercial use. This is particularly interesting because it allows researchers and developers to experiment and innovate with the models. To help, Google and Microsoft have a partnership with Mistral to make their models available on Google Cloud and Azure respectively [43][42].

A positive point is that Mistral AI places a strong emphasis on minimizing the model's tendency to "hallucinate" (generate incorrect or irrelevant information). Achieving this through fine-tuning, to be more cautious and discerning in its response, ensuring that the models provide accurate and reliable outputs [44].

In terms of data privacy and security, Mistral AI ensures data is encrypted both at rest and in transit. They do not use customer data for training the models, and do not have access to user's input data [67].

2.3.6 Models Performance Comparison

Looking for something profitable and capable of carrying out the tasks we want with the best possible quality. A model that does not suffer from hallucinations, is accurate and has a reasonable relation to cost/price. Each model is trained differently and each offers strengths and trade-offs, so the choice of the best model to build anything is determined by the specific needs of the user. That said, with the help of Artificial Analysis, which provides independent analysis and comparison of AI language models, we created a table capable of showing stats of each model, but only opted to display the most relevant ones, from the companies mentioned in the section 2.3.5 for analysis.

Looking at table 2.1, the context window size, measured in tokens, varies significantly among AI models. For example, GPT-4o, GPT-4 Turbo, and GPT-4o mini offers a 128k context window, while GPT-4 and Gemma 2 9B has smaller 8k windows. Notably, Gemini 1.5 Pro stands out with a 2 million token context window, enabling it to handle substantially large inputs. Most of the models are proprietary, such as Gemini or GPT-4 variants, being restricted in access and modification. In contrast, open models like Gemma 2 9B and Mistral Large 2 offer more flexibility. Latency, the time to the first token received after an API request, varies with GPT-3.5 Turbo having the lowest latency at 0.48 seconds and Gemini 1.5 Pro being the highest at 1.04 seconds. Output speed also differs, with Llama 3.1 70B achieves the highest at 249 tokens per second, while GPT-4 generates 27 tokens per second. Cost per million tokens ranges from $15.00 for GPT-4 Turbo to $0.09 for Gemma 2 9B, crucial for budget-conscious users. For speech-to-text models, OpenAI Whisper excels with a 0.5% word error rate, whereas AWS Transcribe has a higher error rate of 24%. Finally, for text-to-image models, DALLE 3 costs double than Titan G1 but in performance by ELO score of the models as determined by >100,000 responses from users in Artificial Analysis, OpenAI DALLE 3 scores a little higher. Overall, the analysis reveals a trade-off between cost, speed, and performance across different models. Users must prioritize based on their specific needs, such as budget constraints, speed requirements,

	Context Window	License	Latency	Output Speed	Blended Price	Word Error Rate
GPT-4o	128k	Proprietary	0.60 s (OpenAI)	79 tokens/s (OpenAI)	$7.50/1M tokens (OpenAI)	-
GPT-4o mini	128k	Proprietary	0.61 s (OpenAI)	97 tokens/s (OpenAI)	$0.26/1M tokens (OpenAI)	-
GPT-4 Turbo	128k	Proprietary	0.56 s (Azure)	31 tokens/s (Azure)	$15.00/1M tokens (Azure)	-
GPT-3.5 Turbo	16k	Proprietary	0.48 s (Azure)	79 tokens/s (Azure)	$0.75/1M tokens (Azure)	-
GPT-4	8k	Proprietary	0.55 s (Azure)	27 tokens/s (Azure)	$37.50/1M tokens (Azure)	-
Gemini 1.5 Pro	2m	Proprietary	1.04 s (Google)	63 tokens/s (Google)	$5.25/1M tokens (Google)	-
Gemma 2 9B	8k	Open	0.24 s (Deepinfra)	69 tokens/s (Deepinfra)	$0.09/1M tokens (Deepinfra)	-
Claude 3 Haiku	200k	Proprietary	0.44 s (AWS)	120 tokens/s (AWS)	$0.50/1M tokens (AWS)	-
Mistral Large 2	128k	Open	0.53 s (Mistral)	24 tokens/s (Mistral)	$4.50/1M tokens (Mistral)	-
Llama 3.1 70B	8k	Open	0.28 s (Groq)	249 tokens/s (Groq)	$0.64/1M tokens (Groq)	-
Text-To-Speech, OpenAI standard	-	-	-	15 chars/s	$99.9/1M characters	-
Text-To-Speech, Google standard	-	-	-	4 chars/s	$542.5/1M characters	-
Text-To-Speech, AWS standard	-	-	-	4 chars/s	$668.7/1M characters	-
Text-To-Speech, Microsoft neural	-	-	-	15 chars/s	$239.3/1M characters	-
Speech-To-Text, OpenAI Whisper (Groq)	-	-	-	-	$0.103/1000min of audio	0.5%
Speech-To-Text, Google Chirp	-	-	-	-	$0.124/1000min of audio	16%
Speech-To-Text, AWS Transcribe	-	-	-	-	$0.112/1000min of audio	24%
Text-To-Image, OpenAI DALLE 3	-	-	-	-	$40/1k images	-
Text-To-Image, AWS Titan G1	-	-	-	-	$20/1k images	-

Table 2.1: LLM base stats.

Legend:
Context Window - the amount of text (in tokens) that the model can consider at one time when generating or understanding language;
Latency - time to first token of tokens received, in seconds, after API request sent;
Output Speed - tokens per second received while the model is generating tokens (i.e. after a first chunk has been received from the API);
Blended Price - for easy comparison calculate a blended price assuming a 3:1 ratio of input to output tokens;
Word error rate - percentage of words transcribed incorrectly (5,000 test samples).

or the ability to handle large text inputs.

However, only displaying the table with the model's characteristics is not enough to draw conclusions about the models' performance. That said, we are going to analyze the models on their text capabilities through the *WILDBENCH* benchmark [75], an automated evaluation framework designed to benchmark LLMs using challenging, real-world user queries, which give us an insight into the models perform to real-world tasks. That is contrary, to traditional benchmarking datasets that fall short in evaluating the more open-ended problems. The table 2.2 shows the performance of the models on text tasks, a smaller version of the table presented in here by the *WILDBENCH*.

Based on the comparison between the different language models in table 2.2, GPT-4o consistently performs the best across various tasks such as information retrieval, coding, debugging, mathematics, and reasoning. GPT-4o mini and GPT-4 Turbo also demonstrate

Version	GPT-4o 2024-05-13	GPT-4o mini 2024-07-18	GPT-4 Turbo 2024-04-09	GPT-3.5 Turbo 0125	Gemini Pro 1.5	Gemma 2 9B	Llama 3.1 70B	Claude 3 Haiku	Mistral Large 2 -
Info Seek	58.6	57.4	57.2	36.5	52.2	49	-	45.3	-
Creative	59.1	60.1	58.7	37.4	55.1	51	-	42.9	-
Code and Debug	60.5	57.2	55.1	26.5	55.2	36.7	-	37	-
Math and Data	57.3	54	51	21.6	48.6	36.4	-	31.4	-
Reason and Plan	60.2	58.2	56.2	33.4	53.7	46.7	-	41.3	-
WB-Score	59.3	57.1	55.2	30	53	42.7	-	38.9	-
WB-Elo (Raw)	1246.3	1194	1224.3	1119.7	1212	1158.8	-	1161.5	-

Table 2.2: LLMs performance on text.

Note: WB-Score shows task-wise performance and also does a weighted average of them. WB-Elo-Raw has no length penalty, so has a bias to longer outputs and thus will be hidden later.

very good performance, particularly in creative tasks, with GPT-4o mini standing out in this area. Gemini Pro has a moderate performance across most tasks but does not lead in any specific category. Claude 3 shows decent performance, especially in creative and reasoning tasks, but they do not surpass the top models. On the other hand, GPT-3.5 Turbo consistently shows the lowest scores, suggesting it may not be as effective for complex or varied tasks. Llama 3.1 and Mistral Large 2 do not have any data available yet. Overall, GPT-4o achieves the highest WB-Score and WB-Elo (Raw) score, indicating its superior performance. This analysis emphasizes the importance of selecting the right model for specific tasks. For us, it's crucial to consider code generation since we will be asking the LLM to produce code snippets for integration with the assistant. Additionally, the performance of text generation tasks is crucial since voice or image generation may depend on the quality of the text output.

Relatively to analyze and interpret visual content alongside text capabilities we have used a study performed by Gemini Team [64]. In table 2.3, we understand that Gemini Ultra is the overall best, with high accuracy on most tasks. GPT-4V comes right after, and Gemini Pro is not far behind. The benchmarks provide valuable insights, but like any text benchmarks, we need to take into account that a LLM performing very well in most cases can fail to produce a good result in a few specific cases.

We have also encountered a recent study by a third party that made a comparison between Google's Gemini Pro model and OpenAI's GPT, diverging in the fact that GPT 3.5 has slightly better accuracy than Gemini Pro, going against the results from Gemini Team [64] report and showing us that Gemini may be using the best methods to get the best performance in the benchmarks. The article from this third party [4] is of very valuable interest because after all the analysis, they conclude the following:

- Gemini Pro achieves, on average, accuracy and performance somewhat inferior to GPT-3.5 Turbo and notably worse than GPT-4;
- Relatively to code generation, Gemini Pro had a higher proportion of mistakes than the other two;

	Gemini Ultra	Gemini Pro	GPT-4V
MMMU Multi-discipline college-level problems	59.4%	47.9%	56.8%
TextVQA Text reading on natural images	82.3%	74.6%	78.0%
DocVQA Document understanding	90.9%	88.1%	88.4%
ChartQA Chart understanding	80.8%	74.1%	78.5%
InfographicVQA Infographic understanding	80.3%	75.2%	75.1%
MathVista Mathematical reasoning	53.0%	45.2%	49.9%
AI2D Science diagrams	79.5%	73.9%	80.9%
VQAv2 Natural image understanding	77.8%	71.2%	77.2%

Table 2.3: LLMs performances on image benchmarks.

- In mathematics, Gemini Pro has a superior performance to GPT-3.5 Turbo in the most complex examples requiring longer chains of thought but underperformed in shorter examples, indicating sensitivity to the length of reasoning chains and large digits;
- GPT-4 exhibits robustness in handling long reasoning chains in mathematics;
- Comparing general-purpose reasoning, Gemini Pro achieves slightly lower accuracy than GPT-3.5 Turbo and much lower accuracy than GPT-4 Turbo;
- Gemini Pro, in question-answering tasks may have some bias in answer ordering (favoring the final choice "D" in multiple choice) and underperformed in most tasks with chain-of-thought prompting. Achieving lower accuracy than GPT-3.5 Turbo and significantly lower than GPT-4 Turbo.

After everything, it is difficult to compare results and perfectly indicate the best model, because the models are trained on different algorithms, and depending on the cases we can observe random unexpected results for any model. So, to choose one model, we have to consider the specific needs of the user, such as budget constraints, and the specific tasks that the model will perform.

2.3.7 Experiments with LLM

To examine the behavior between the language models and understand how they perform in relevant situations, we did some experiments with a few models to compare their responses. The analysis we did, tested some of the steps that possible features may

need to be performed and see if they are possible, and how trustworthy they can be, using GPT-4, Gemini Pro, and GPT-3.5 Turbo models. For the GPT-4, we have to utilize a free option through the Bing chat. However, the input is just 4000 characters. Then, we tested the Gemini Pro through Bard chat, and the same for GPT-3.5 using the ChatGPT. Of course, there are other language models from different companies, however, the ones we cover in more depth are, in addition to the most popular, those that have the greatest support and development by the respective companies or community of developers.

In the first test, we gave them a JSON structure and asked them to interpret the fields that were given. With this, we were able to analyze the model's capability of comprehending content. Overall, every model doesn't provide a wrong interpretation, but sometimes they don't detail as much. GPT-4 was the only one that seemed to articulate and best understand the JSON structure and the meaning of the fields.

Then, we test some scenarios for analysis. In the first scenario, we give them data in the format of a JSON file with information on employees' vacations, and ask them questions about that. The responses were prevailing wrong, only banal questions got correct answers. When the models were confronted to calculate who had gone on vacation in April this year, all the models didn't answer the question correctly. Not even giving them context about the fields, the answers are not coming out correctly. It seems the models were confusing the needed fields to compute the answer, seemingly using the first field they encountered in the JSON to respond. After this, we try to change the format of the JSON file to something more nested. The results were partially the same. So, finally, we try a simple approach, giving them fields of a "View" (data projection with less information, only with the most relevant fields and a small description of each). Then we asked to generate JavaScript code to use in the JSON file that returned an array of objects of the employees who went on vacation this year in April. Now, the models produce functional responses. Only GPT-3.5 was having difficulty in a few cases, however, it only needed better refinement in prompt engineering, for that, we can conclude that the way the instructions are written is important, requiring a structured approach, and cannot rely on a large amount of data for analysis.

After testing the analysis of data, we try to fine-tune the GPT-3.5 model for workspace creation. Workspaces/dashboards have a predefined structure, so we first try to ask to build one using only prompt engineering and then fine-tuning to understand the difference. The difference was a little clear because fine-tuning the model to generate the dashboards will give us something that could be partially integrated into the platform, and the only thing that needs to be changed is the components plus their structuring. Probably, for the resolution of this problem, we will also have to train the model in all the existing components to build the workspaces (e.g., forms, calendars, buttons, etc) so it knows what can be used to generate the workspace.

Note: We can only train the GPT-3.5 Turbo because of the free Azure services provided.

Finally, we test the speech-to-text feature. We give the models the same inputs, and either model responds without problems. The audio was clean and could be interpreted well. Otherwise, the result may vary.

2.4 Possible architectures for an assistant

Following the process of developing an assistant, after looking at the performance of the LLM and analyzing some of their advantages and disadvantages, when looking at the characteristics of the assistants mentioned in the table 2.4 we have to think about an architecture capable of supporting refined features in that scope. Our assistant should have the ability to manage all automation's in the best way possible, reducing the probability of errors the more we can.

2.4.1 Overview

When thinking about a solution to build this assistant, we think about what already exists in the application and what the assistant should do. The application offers several endpoints that allow all operations available to the user to be done. We intend to use these endpoints, but also create new ones if necessary, allowing all the necessary actions to be carried out for the assistant to function in terms of the automation that is intended to be carried out. We also want our assistant to be incremental and to be able to easily adapt to new features over time as the platform and user needs evolve. With this idea in mind, we created the following view:

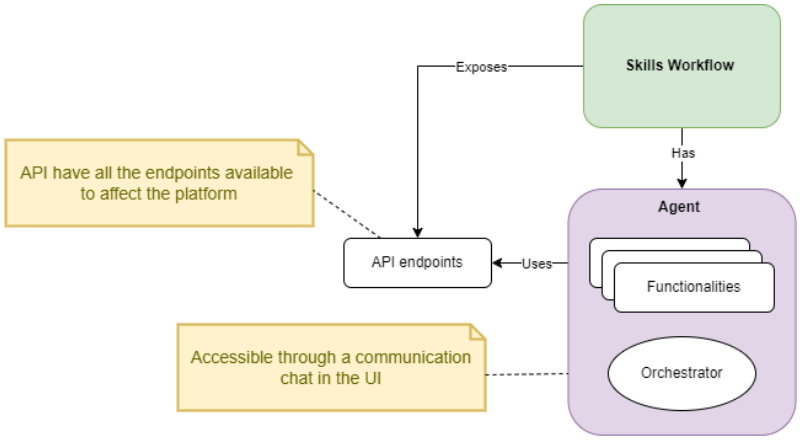


Figure 2.3: How an assistant can be integrated into the platform.

It is crucial to understand the steps that the orchestrator will choose are going to depend on the user input, so we need to develop carefully the component that manipulates the way the task is performed, so it correctly accomplishes the user request.

2.4.2 Azure Bot and Google Dialogflow

When looking at how to develop an assistant, the first approaches that came to the table were Azure Bot and Google Dialogflow. Both of these are services that allow us to build Chatbots using natural language processing and artificial intelligence.

In common, they have features providing tools and templates to design, test, and deploy Chatbots quickly and easily, even for non-developers. Both services use natural language processing (NLP) to understand users' intentions. They can search the web for information, create content based on user input specifications, summarize, make comparisons, and answer questions about our application's functionality or performance [22][23].

However, regarding the use of LLM, both have limitations. Azure Bot allows you to use GPT-3, but only through an integration with the OpenAI service. Google Dialogflow does not have a direct integration with GPT-3, but you can use Meena, a LLM developed by Google, without the latest models from the Gemini family [25][13].

However, it seemed to us that the Semantic Kernel (explained in the next section 2.4.4) is more flexible, allowing us to use any natural language model within our code. Semantic Kernel has less restricted features than Azure Bot and Google Dialogflow, which allows us to generate more complex and specific functionalities and workflows.

2.4.3 Distributed Assistant

Another alternative solution would be to separate the assistant into the different functionalities that we want to implement, and instead of the user accessing the assistant all in the same place by talking to a chat, we place each functionality in its proper place, and the user can access them from the more direct way, from the specific micro-application the system has, reducing the chance of the assistant to fail in producing the exact workflow the user wants (an approach like AgentHub did, mentioned in section 2.5.2). For example, if the user wants to analyze the system's data, we will use a text input in an accessible place on any page of the platform. If the user wants a summary of a message feed where several users are discussing a topic, we place a button in that feed that generates the summary. If, on the other hand, the user needs to create a systematic check of a parameter in the system, with a trigger per time, or activated by changing the parameter, we place a button on the home page that allows the user to create these situations. And so on. Following this logic, we have the advantage that the system performed what the user wanted in the first place, and the errors can only occur in the processes of the task itself where the LLM is involved.

2.4.4 Semantic Kernel

At last, when searching for concrete solutions to our problem, one capable of merging functionalities in one place and having a robust and adaptative structure, we found that

Azure has a good solution that can be used to build it. It is called Semantic Kernel, which is a new approach that allows us to integrate LLM into our platform. It is the same technology used by Microsoft to build Copilot, apt to power our platform assistant as well.

An assistant should be capable of retrieving information from both the user and system, conceiving how to use that information and how to respond to the user or perform an action in the system. The Semantic Kernel, similar to operating systems, will manage resources like assistant concrete functionalities, the memory of the assistant, and the connectors (modules that enable external data access, such as Blob Storage and Cosmos DB, which is connected through Cognitive Search). It is an open-source SDK that allows us to manage our existing code with AI. This means that we will be able to give our assistant the ability to interact with the user by calling our existing services and creating new ones [17][18].

2.4.4.1 Advantages and Limitations

The advantage of this approach is that the Semantic Kernel orchestrates and chooses the right functionalities using a single LLM call, which reduces the complexity and cost of building an AI assistant. However, there are limitations related to the writing of effective Prompts, capable of producing the expected results. Although this problem will always be difficult to solve and very connected to LLMs environment.

We searched for similar solutions, but we did not find one that joins conventional programming languages with LLM Prompts, other platforms mainly provide access to pre-trained or fine-tuned LLMs and do not allow us to write semantic functions or even integrate them with existing code, which will lead us to have to perform several LLM calls to a single user request. They also don't provide a mechanism to orchestrate all the different types of functionalities that we want to implement, which is one of the biggest downsides for the other competitors of Azure.

The disadvantage we see in using the Semantic Kernel is the fact that it is still under development, and may contain some bugs and is not completely stable, in addition to there being few examples to base ourselves on.

2.4.4.2 Semantic Kernel Architecture

We have idealized an architecture for our assistant using the Semantic Kernel.

The assistant comprises four essential components: plugins, memory, planners, and its persona.

Plugins - At a high level, a plugin is just a group of functions that can be exposed to AI applications and services. These modules give the assistant skills or characteristics, consisting of both native code and requests to AI services via Prompts. For the AI to distinguish the functionalities in the plugin, each one will have a description that explains what it does so it can be chosen accordingly [21].

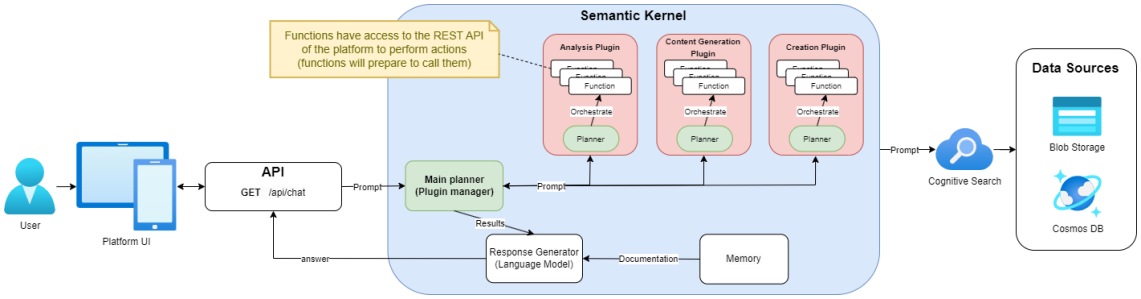


Figure 2.4: Semantic Kernel architecture.

Memory - Memory is the component that stores and retrieves contextual information generated from conversations and imported from external sources, such as documents, web pages, databases, etc. All the information can be accessed via semantic functions by the AI assistant and the user. These functions*, are natural language expressions that can be evaluated by the assistant to perform operations on his memory.

> **Note:** * There are two types of functions that we can work with in plugins, semantic functions, and native functions. Semantic functions are called out from AI models, and they access or modify the memory of the assistant (do not require a fixed syntax and can be composed dynamically by the assistant or the user). The native functions, allow us to do other things, like mathematical calculations, because as we saw in section 2.3.6 related to the performance of the benchmarks, they are not very good with math, doing things more reliably. Semantic functions can help us to perform data analysis tasks, such as data validation, transformation, visualization, and querying.

Planners - Planners are algorithms that use the AI service to decide what to do with the user's question/Prompt, planning the steps to achieve the user task. There are two types of planners, Handlebars Planners and Function calling stepwise planner. Handlebars Planner can generate an entire plan with variables, conditionals, loops, and expressions, all with a single call to the AI service. On the other hand, function calling stepwise planner, can orchestrate a plan by calling functions from the AI service one by one, and evaluate the results of each function call so it can decide the next step, using the OpenAI function calling capability [19][12].

Persona - The persona gives our assistant a personality. So we can make it like a person, more or less sarcastic, formal or friendly.

For our implementation, we can have a plugin for analysis-related use cases. Another plugin for content generation use cases where we generate data based on information from the platform. And a last one for creational use cases, this also generates data but from scratch. An organization accomplished this way would allow plugins to have some common functions where the result could be used differently by use cases. For example, in the analysis plugin, create a function that performs a search on the user request, and then the result will be used differently by the use cases.

For the Planners, there should exist one main planner for choosing between the plugins, and one for each of the functionalities. Inside each plugin, will be created all the necessary functions to accomplish each use case. Besides the memory needs to be configured to get data from documents and databases, it also needs to save conversation history relevant to the user's task. The response generator is the component that uses an LLM to create the final natural language response for the user.

In figure 2.4, we see an important part of the implementation, Azure Cognitive Search. This service enables us to perform searches on our data stored in Cosmos DB or Blob Storage through natural language queries. This is highly pertinent as it will allow us to analyze our data efficiently. To use the service is necessary to index data that we consider relevant for research to Azure Cognitive Search, CosmosDB, and Blob Storage. As we have a lot of data, we are thinking that the best solution will be the use of projections of the data, only with the important fields and tables. After this, we need to connect to Azure Cognitive Search, Cosmos DB, and Blob Storage.

One of the functions that should be included and utilized by all the requests will notice if the user is interrogating what he is seeing in the dashboard or if the question is not about it. This function should be performed at the beginning of the pipeline and decide the data that will be used to execute the rest of the request.

2.5 Related Work

2.5.1 Similar Solutions

There are not a lot of existing solutions to which we can compare our work. However, the major similarities we found between our work and other projects were the *Airtable Cobuilder*, the *Gemini Live* and the *Project Astra*.

Project Astra is a Google AI initiative aimed at creating a universal AI assistant, with multimodal capabilities that can understand and generate text, images, and audio. The assistant can perform a wide range of tasks and can engage in natural human-like conversations [9]. Another AI project by Google is the *Gemini Live*, but it focuses more on enhancing mobile experiences, with real-time conversations, supporting also video inputs [70]. Both these two are good examples because they want to build an assistant in the future, processing multimodal information and understand the user context.

Airtable Cobuilder allows users to create apps by simply describing what they need in natural language. The AI interprets these descriptions and generates the corresponding app, significantly reducing the time and effort required to build functional applications. This aligns perfectly with our goals of automating processes to enhance operational efficiency and allow users to focus only on creative tasks. It presents a *No-code* Development, where users can build apps without needing to write code. The AI handles repetitive and complex tasks. The intuitive interface ensures that users do not need to learn new complex systems [2][69].

In relation to our work, *Airtable Cobuilder* reduces the learning curve by allowing users to interact with the system using natural language, eliminating the need for extensive training on new interfaces. Automates the creation of apps, which parallels your objective of automating routine tasks in the Skills Workflow. So, both *Airtable Cobuilder* and our assistant relies on NLP to understand and execute user commands [2][3].

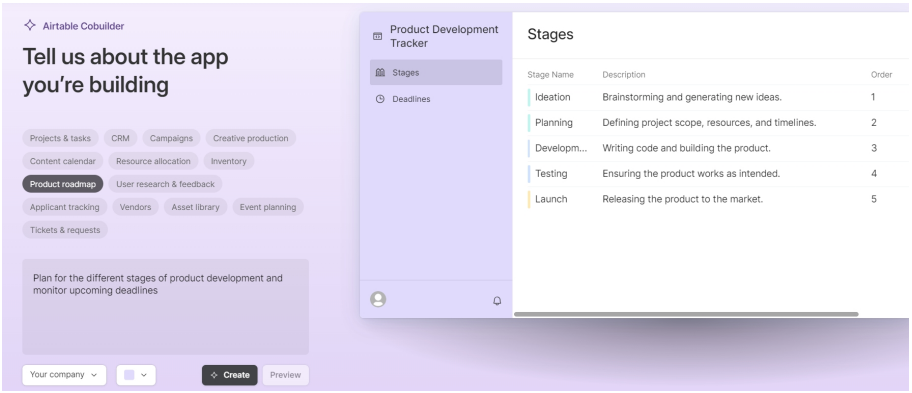


Figure 2.5: *Airtable Cobuilder* webpage build example.

In other words, we can ensure through *Cobuilder*'s success in making app creation accessible and straightforward can serve as a model for designing your assistant. An example is shown in figure 2.5.

2.5.2 Application areas of the assistants

When looking for related literature, we found a particular article that integrated LLMs, it is about *Brain.ai*, a new smartphone operating system that uses generative AI as the core of its interface and functionality [32]. They say the smartphone does not rely on apps, icons, or menus. Instead, it uses natural language and gestures to interact with the user. The system is designed to automate things like, create personalized and dynamic wallpapers, ringtones, and themes, or learn from the user's behavior and preferences and adapt accordingly. The system also faces challenges like ours, for instance, quality, accuracy, and safety of the generated content, or being careful with users' privacy and personal data. However, it brings new possibilities for creativity and productivity, merged with an incredible user experience, just like our intentions. The major difference to us is that generative AI will be used in an application and provide assistance to the user without being the core of the system, so it can be normally managed, only with a boosted performance because of the automation's.

Posteriorly, exploring in the context of the use of LLMs in the company's application, we found OpenAI has a webpage featuring several stories from customers who use GPT models to advance their goals [49]. There were a few that caught our attention. *Stripe* [54] was one of them. They took advantage of the services offered by GPT and improved their product's resources and workflows, just like we intend to do. They led a team of engineers to explore the capabilities of GPT-4 and concluded they could use the model to

scan, summarize, and analyze large-scale data very quickly, outperforming any human. Looking at this, we understand that we can make use of Large Language Models the same way, using them to analyze the huge amount of data from our platform, and more specifically from each dashboard, and respond to some user questions about it.

Another of the stories that caught our attention was *Be My Eyes's* [55], as they used GPT to empower the blind and low-vision community leading a revolution in visual interpretation. Our first thoughts were how we do something similar, blending the real world environment with the Skills Workflow platform, capturing the real and transforming it into something in the digital one. This way, we can create features that enable, using voice commands, effortless communications but also help blind people interact with the application if necessary. Similarly, by utilizing images, we can convert real-world information to our application, like printed documents, transforming them into digital files.

From almost every example, the GPT needs to generate data, and it is what we want to do create new information and code to do something in our platform. The examples provided by OpenAI, many of them make use of the model the same way, for processing large amounts of data or creating new data through user communication.

We also discovered that *Caixa Geral de Depósitos* (Caixa Geral de Depósitos sort) has a virtual assistant [15] that can interpret natural language through voice or text and do actions like bank transfers, check balances, and account movements. They managed to create an assistant that can perform tasks based on an interpretation of human language, where the user only has to express their needs. This improved the convenience and efficiency of their customers' banking experiences, which meets what we want to achieve as well, automating processes and enabling the user to communicate with the application intuitively, to carry out tasks more quickly and efficiently.

We explored *Copilot* as well, Microsoft built an assistant that can be integrated into Visual Studio Code and is capable of generating code through natural language. It can even follow the user's progress and analyze the open document to understand how the code is being created and generate new [20]. This is an exciting feature because the LLM may need to first, analyze the code, to understand how something is structured in the application and then add new code to it. Helping programmers not just to write faster, better, and more reliable code, but also providing similar answers to what they already have.

Lastly, we found an incredible application *AgentHub* [1] interested in building automation within a single line of code, which fits perfectly into what we want to develop. To build the automation's they use drag, drop, and linking of nodes onto a canvas, building the workflow. This way they reduce the chance of having a global assistant to do that same task and fail. They have some templates of workflows to produce things like a LinkedIn profiles processor, a tweets generation Bot, a code problem solver, an email inbox summarizer, etc. Many of these possibilities exist due to their integration with services such as Twitter, GitHub, or Google.

There are many other possibilities that we are thinking of exploring, here we merely understand the general uses that can be done. In short, table 2.4 was created with general characteristics found in the examples described in this section.

	Definition
Data analysis	As we can see in *Stripe*, they use GPT to analyze and summarize huge amounts of data. So, we realized that we could search and analyze data from the platform.
Code Generation	From Copilot, or we understand that we can generate code for anything, and execute it most of the time correctly.
Real world to digital word	From *Be My Eyes*, we understand that we can communicate to the application in more than one way besides text.
Performing actions	With *Caixa Geral de Depósitos sort* we perceive that we can use the LLM to help communicate with the user and to orchestrate some existing actions in the application.

Table 2.4: Possible characteristics of an assistant using a LLM.

It is common for platforms with AI-powered assistants to face limitations in carrying out certain user requests. It can occur due to the assistant's incapacity to produce the correct result by following an incorrect workflow, or by the LLM miss comprehends what the user wants. Relatively to this last, it is difficult to attack the problem, seen to be related to the user input. However, related to workflow production, we may adopt a strategy to reduce the probability of failure, like *AgentHub* did.

3

Architecture

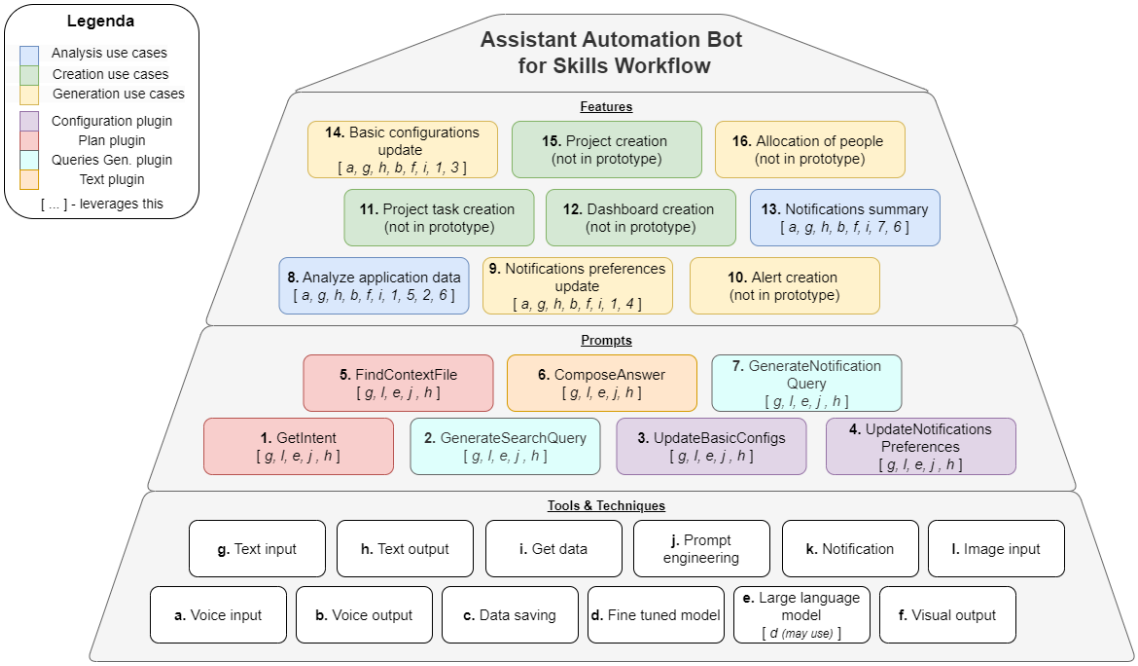
This chapter covers the details regarding the architecture of our proposed implementation for the automation's we are developing in Skills Workflow. To begin with, we will examine and discuss the models analyzed in sections 2.3.5 and 2.3.6, as well as the potential architectures for building the assistant, as outlined in section 2.4.

3.1 Assistant Features Overview

To develop an assistant, it is essential to comprehensively analyze the problems and tasks that users spend significant time on, so from there, its functionalities come out. We reviewed the functions of Skills Workflow platform (addressed in section 2.1) and considered various potential uses of the LLM, taking into account the characteristics outlined in table 2.4, adapting them to Skills Workflow. By doing this, we can identify processes that can be automated to enhance productivity and reduce the workload for users, in a way that the client does not necessarily understand how the software is generating the answers.

The pyramid shown in figure 3.1 was created to visualize all functionalities better and brainstorm about possible features and how they can be built, over the Semantic Kernel architecture (see 2.4.4.2). We divided the features into three categories. The first is the *analysis*, where we place all the functions that will analyze the application's data. The second is *content generation*, where we place all the functions that will generate content based on application data. The third is the *creation* category, where we put all the functions related to content creation based on the user's question. Each use case implementation will be explained in the next chapter. And divided the Prompts into four categories, that represent the four plugins that we have created. The *Configuration plugin* responsible for the Prompts affecting functionalities related to the settings of the assistant. The *Plan plugin* contains Prompts responsible for making choices. The *Queries Gen. plugin* with Prompts produces JavaScript queries to data. Lastly, the *Text plugin* contains Prompts that generate text.

At the top of the pyramid (see 3.1) are the use cases that represent the assistant's



Note: The square brackets in the figure indicate the use cases that each one may employ. The Dashboard Creation, Alert Creation, Project Creation, Task Creation, and Allocation of People were discussed but have not been implemented. Currently, only the handler that identifies those intents is operational.

Figure 3.1: Use cases pyramid.

functionalities. Each of these functionalities uses pre-filled Prompts to the LLM service via Azure OpenAI Studio (Web-based Interface to access language models) to accomplish specific goals, which we will address as Prompts (inside the plugins on Semantic Kernel). More detailed, they are pre-made requests in natural language to the LLM. For example, a Prompt that has "Choose between A and B according to the user question. Knowing that A means X and B means Y. User input: {$user input}", will return an answer (A or B) following the rules described. Just like when we talk to *ChatGPT* or *Copilot*, we ask something and they respond accordingly to the context.

Furthermore, the assistant must understand the user's intentions, so not all the Prompts will be designed to treat a concrete functionality, they may be needed to assist, such as the one that will find the user intention, described as "GetIntent" on figure 3.1. This orchestrator is detailed in the next chapter and should be capable of distinguishing between the described use cases at the top of the pyramid and using them to respond to the user or perform actions on behalf of the user.

Additionally, we want the user input interpreted by the LLM to be given through text, voice, or even an image. However, in the assistant prototype, only text input is implemented. To use the other two, we would have to integrate the voice and image model with the assistant.

To explain how we move from a natural language question to obtaining a specific answer within the context of the Skills Workflow, we will need to outline the process for each function in the following chapter. In the subsection below we will provide an abstract explanation of what each feature in the pyramid (see 3.1) is expected to accomplish. Additionally, we will discuss the features that have not been implemented in the prototype, such as Dashboard Creation, Alert Creation, Project Creation, Task Creation, and Allocation of People.

3.1.1 Analyze Application Data

This use case is designed to help users deal with diverse information in workspaces. Users can use this feature to decipher and collect statements or quickly search for application data. This feature is of extreme importance as it can be used to support functionalities necessary to retrieve information from application data. Therefore, it is essential to ensure that the results obtained are of high quality and reliability, and presented in a precise and concise manner to prevent user frustration and disengagement. For example, a user may require assistance in identifying an available individual, like Pedro, for a specific task within a project. With the global exposure of the assistant in the platform, users can simply inquire about Pedro's availability from any workspace, enhancing their productivity significantly without disrupting their workflow. Another example can be asking the assistant who is going on holiday in August 2024. Where an optimal answer will be with the names of the employees who are on vacation during that period.

Here, two approaches can be taken: using Retrieval Augmented Generation, or determining which application data set can answer the question and generate a JavaScript query to fetch this information from JSON data files.

3.1.2 Notification Summary

This feature is beneficial for users with numerous unviewed notifications or those who don't check notifications often. It generates a summary of all existing notifications, filtering them based on specific criteria set by the user in the settings. This allows you to quickly access the most relevant and important information without scrolling through each notification individually. The goal is to provide a more organized experience, helping users stay up-to-date without feeling overwhelmed.

Similar to the data analysis the requirements will be related to the LLM capability to generate a JavaScript query over an application data set that contains all notifications.

3.1.3 Configurations Update

We want certain assistant settings to adjust naturally as if the user were interacting with a person. Related with basic configurations like language or behavior (*BasicConfigurationsUpdate*), and notification settings like the user preferences or even if want to receive

notifications or not (*NotificationPreferencesUpdate*). For example, if the user wants to change the language or the way the assistant responds, ask him directly, as in "Answer me in Portuguese". The idea is to eliminate the need for additional interfaces, such as buttons or drop-down lists, allowing users to make a more natural request, making it easier for them to interact with the assistant without requiring technical knowledge about its inner workings or the Skills Workflow platform structure.

This feature requires the assistant to understand the user's request and generate a JSON representation of the new settings.

3.1.4 Dashboard Creation

In the Dashboard Creation use case, we intend to automate the creation of a workspace through its description in natural language. For instance, when a user requests to create a dashboard displaying employees' availability and their respective project involvements, it must be translated into code that the system can operate. This may involve the creation of a table that lists all employees, with their respective projects displayed below their names. The result will be a dashboard that provides a clear overview of employee availability and current project assignments.

- Significantly reducing the time the user takes to make the workspaces;
- If the user doesn't know how to do it, he needs to express what he wants;
- The model may generate something the user doesn't know exists.

To implement it we will ask the LLM to produce pieces of code responsible for creating the dashboard, more specifically the components and JSON data set fetch calls. The workspace will only serve as a base for users to customize, saving them significant effort. After the code is generated a call to the Skills Workflow SDK will be made to produce the workspace.

3.1.5 Alert Creation

Alert Creation would be new in the system. What is meant by the creation of "alerts" is simply that the user can define something that he wants to be notified about and is related to data in the system. For instance, whenever an employee's working hours exceed twenty, the user receives a notification. The use case allows us to help the user to be up to date with everything happening on the platform without having to constantly check manually certain data. With this, the user is only required to add a new alert.

The strategy we think of is to make the LLM produce a code like a trigger event to check something in the platform data. When the user sends the request for the alert creation, it will first look at the saved ones, checking for similarities. If it has one, it will retrieve the code from the ID returned. Otherwise, it will ask the LLM to generate new code for a trigger function. In the end, it shows the popup to confirm the code. If

confirmation is successful, it will be saved in the system. The alert code will always be running periodically in the system.

3.1.6 Project Task Creation

The Project Task Creation functionality enables users to generate tasks for a project easily. This basic feature allows users to create new tasks within a project using natural language. For instance, a user can describe a task, such as "Create a task to make final revisions on the project from Coca-Cola," where the assistant will generate a task with the description "Final revisions" associated with the Coca-Cola project.

To resolve this use case, the LLM should be capable of generating the content of specific fields in the SDK call that creates a new task.

3.1.7 Project Creation Use Case

This use case scenario involves the creation of a new project on the platform. The user describes the project, and the LLM uses the provided information to compose a new project. A concrete example could be the user asking, "Create me a new project like the one for Galp's September campaign, but for this summer". Additionally, the LLM will include any other necessary information to complete the creation process.

Like the task creation, here, the LLM should be capable of generating the content of specific fields in the SDK call that creates a new project.

3.1.8 Allocation of People Use Case

Allocation of People will be a use case that permits us to tell the system things like, "If a person has exceeded the work time per week, please re-distribute tasks from a project so that everyone has equal work time" or, "I need you to put Jhon Thornefield on the project x". This functionality would make life easier for the user, as they would not have to go to each project distributing and allocate members between projects.

It would require the system to understand where it would move/add members from and what fields it would have to modify or create. A person can be removed from a project or added to one, which changes the data in the system, by calling the endpoint returned from the LLM, needed to generate the necessary information to call it also. If the user asks "Pedro will not be available to work at Nespresso. Change his tasks to Jorge", the system has to understand the intention and change task attributions. The assignment of tasks to individuals needs to be updated in a JSON file. This involves searching for specific tasks and making changes to the corresponding JSON objects. The search process is similar to the data analysis use case, where the task is located within the file. Then the LLM performs the change in that JSON object according to what the user wants.

3.2 Criteria for the Selection of Architectures and Tools

In this section, we will present the criteria used to select the OpenAI GPT-4o mini large language model for the assistant and the Semantic Kernel architecture (see 2.4.4.2) used to implement its functionalities.

Between the several large language models (LLMs), we will choose one that will answer the Prompts mentioned in 3.1. In our assessment, from section 2.3.6, we have concluded that OpenAI models best suit our requirements in price, performance, and response time (excluding GPT-4 and GPT-4 Turbo due to their higher costs). This decision is based on the performance of their models in text generation, as demonstrated in table 2.2, and their reliability and availability on Azure, which align well with our infrastructure as discussed in section 2.3.5.1. After opting to use OpenAI models, in table 3.3, we compared GPT-4o, GPT-4o mini, and GPT-3.5 Turbo to determine the most suitable option for our specific needs. Our comparison involved evaluating the models based on the tokens used, answer accuracy, latency, and cost of the response. We ensured consistent testing conditions for all models, including resetting the history after each question or action, testing each question ten times, and maintaining the use of the same pre-filled Prompts in each model to observe their behavior. Additionally, it is important to note that the latency time is the average total time for each question. The input tokens were the sum of the tokens from the question plus the pre-filled Prompts from one or more calls to the LLM (those needed to execute the use case). The output tokens are the sum of the response tokens from each call to the LLM.

After the analysis of the table 3.3, many conclusions can be drawn. We compared GPT-3.5 Turbo, we found that it has fast response times. However, it failed to answer several questions correctly. For example, it generated incorrect code for question twelve. Additionally, it sometimes didn't produce output in the correct format, like for questions seven, nine, and eleven, which makes it difficult for the backend to extract the information. Furthermore, the quality of the answers was poor, and it also had a smaller context window compared to the other two models (see 2.1). This could be a downside when a larger number of tokens are fed to the LLM, particularly in cases where a comprehensive and lengthy conversational history is crucial. GPT-4o, on the other hand, got all the questions right, and appears to be faster than GPT-4o mini, although being more expensive. Finally, GPT-4o mini is the cheapest model among the three models and is sometimes slower in generating the answer, with significant fluctuations between tries. However, it obtained zero accuracy in question eleven for not answering it, because it was trying to refine the user's question. Therefore, we can conclude that GPT-4o mini is the most suitable model for our demands, as it is the most affordable and shows promising quality. Reducing costs in the price of the model, we may opt for a better tier price for the service, reducing the response time during peak periods when the response time was longer due to significant fluctuation. By using GPT-4o mini, we can predict that if approximately 100,000 questions are asked in a day, the cost will be around 20-30€, which is about 0.0002€ to 0.0003€ per

Nº	Question	Avg. Tokens	Model	Answer	Avg. Tokens	Accuracy	Avg. Latency	Cost
1.	What are you able to do?	383	GPT-4o	I can help you with adjusting basic settings, updating notification preferences, analyzing data, and making changes to system data such as creating alerts, tasks, projects, and allocating people. How can I assist you today?	77	100%	2258 ms	0.0028704€
	"	392	GPT-4o mini	I can assist you with various tasks such as adjusting settings, analyzing data, and managing system statuses. If you have a specific request or question, feel free to ask!	68	100%	2439 ms	0.0000921€
	"	394	GPT-3.5 Turbo	I am Skillio, an AI assistant from Skills Workflow. I can help you with adjusting settings, updating notification preferences, analyzing data, and making changes to system data such as creating alerts, tasks, projects, and allocating people. How can I assist you today?	82	100%	1130 ms	0.0009044€
2.	Can you pls talk to me in portuguese?	588	GPT-4o	Configuração atualizada. Agora vou falar com você em português.	84	100%	2353 ms	0.0039312€
	"	594	GPT-4o mini	Agora vou me comunicar em português.	96	100%	3736 ms	0.0001408€
	"	597	GPT-3.5 Turbo	Ok, a partir de agora vou falar em português.	99	100%	1545 ms	0.0013224€
3.	I would like you to respond in a more funny and enthusiastic manner.	586	GPT-4o	Sure thing, Arthur! From now on, I'll be bringing the fun and enthusiasm to our conversations! Let's make this a blast!	89	100%	2477 ms	0.0039913€
	"	603	GPT-4o mini	Your assistant will now respond in a more funny and enthusiastic manner!	115	100%	4739 ms	0.0001534€
	"	603	GPT-3.5 Turbo	Sure thing! From now on, I'll respond in a more funny and enthusiastic manner.	110	100%	1534 ms	0.0013547€
4.	Pls do not bother me with any notifications.	593	GPT-4o	Notifications have been turned off as per your request.	110	100%	3518 ms	0.0043161€
	"	602	GPT-4o mini	All notifications have been stopped.	125	100%	3286 ms	0.0001593€
	"	604	GPT-3.5 Turbo	Your notification settings have been updated.	118	100%	1540 ms	0.0013718€
5.	Please resume me notifications about post updates as well.	597	GPT-4o	Notifications about post updates have been added to your priorities.	115	100%	3751 ms	0.0044044€
	"	602	GPT-4o mini	Notifications about post updates have been added to your priorities.	133	100%	4928 ms	0.0001641€
	"	603	GPT-3.5 Turbo	Notifications about post updates have been added to your priorities.	130	100%	1636 ms	0.0013927€
6.	In 2019, in what months did Fred schedule vacations?	1024	GPT-4o	You want to know the specific months in 2019 when Fred took his vacations. In 2019, Fred scheduled vacations in the following months: April, August, May.	279	100%	7126 ms	0.0086909€
	"	1028	GPT-4o mini	You want to know the specific months in 2019 when Fred planned his vacations. Fred scheduled vacations in the following months of 2019: April, August, May.	328	100%	11122 ms	0.0003407€
	"	1037	GPT-3.5 Turbo	Fred scheduled vacations in various months of 2019. Fred scheduled vacations in the following months of 2019: April, August, May.	292	100%	5201 ms	0.0025251€
7.	Who has vacations schedule in 2024?	1018	GPT-4o	You're asking for a list of individuals who have planned vacations for the year 2024. The following users have vacations scheduled in 2024: Arthur , Hannah.	220	100%	5591 ms	0.0078426€
	"	1012	GPT-4o mini	It seems you're asking about the individuals or groups that have planned vacation schedules for 2024. The following users have vacation schedules in 2024: Arthur , Hannah.	248	100%	19444 ms	0.0002905€
	"	-	GPT-3.5 Turbo	[Skillio/Error]: handling intent - TypeError: intentsArray is not iterable	-	0%	-	0€

Table 3.1: Comparison analysis between OpenAI models.

8.	Does Adam have any scheduled vacations?	1000	GPT-4o	You're asking if Adam has any planned time off. No, Adam does not have any scheduled vacations.	198	0%	6440 ms	0.0074522€
	"	1006	GPT-4o mini	You're asking if Adam has any planned time off. Adam has scheduled vacations on the following days: 2018-12-28, 2018-11-22, 2018-09-27, 2018-11-02, 2018-12-25, 2018-09-26, 2018-09-25, 2018-09-28, 2018-11-19, 2018-11-20, ...	224	100%	19498 ms	0.0002750€
	"	1013	GPT-3.5 Turbo	Yes, Adam has scheduled vacations. Yes, Adam has scheduled vacations on the following dates: 2018-12-28, 2018-11-22, 2018-09-27, 2018-11-02, 2018-12-25, 2018-09-26, 2018-09-25, 2018-09-28, 2018-11-19, 2018-11-20, ...	197	100%	3833 ms	0.0022990€
9.	How many vacations does James still can take in 2023?	1027	GPT-4o	You want to know the remaining vacation days James has for 2023. James has taken 0 vacation days in 2023.	234	100%	6466 ms	0.0080795€
	"	1029	GPT-4o mini	You're asking about the remaining vacation days James has for 2023. James has 0 vacation days left for the year 2023.	275	100%	17539 ms	0.0003091€
	"	-	GPT-3.5 Turbo	[Skillio/Error]: handling intent - TypeError: intentsArray is not iterable	-	0%	-	0€
10.	I want to know in April who booked a vacation in 2024.	1016	GPT-4o	You're asking for the identity of the person who scheduled a vacation in April 2024. No one booked a vacation in April 2024.	245	100%	6617 ms	0.0083058€
	"	910	GPT-4o mini	I understand you're asking about the individual or group that made a reservation for a trip in April 2024. No users booked a vacation in April 2024.	231	100%	7264 ms	0.0002660€
	"	1032	GPT-3.5 Turbo	The person who made a reservation for a vacation in April 2024 is unknown. No one booked a vacation in April 2024.	346	80%	5010 ms	0.0026182€
11.	What jobs do Arthur have in stage new?	1090	GPT-4o	You're asking about the roles or responsibilities Arthur has taken on in his current phase. Arthur has the following jobs in the new stage: Job Number: SWA0004DIGI002, Job: Confirm Specs/Deliverables, Product: null; Job Number: SWA0003CREA003, Job: GSD & M to develop key creative, Product: null; Job Number: SWA0003PROD043, Job: In-Airport In Market, Product: null; Job Number ...	360	100%	7914 ms	0.0101270€
	"	393	GPT-4o mini	It seems you're asking about the jobs that Arthur has in the new stage. Could you clarify what you mean by 'stage new'? Are you referring to a specific project or context?	79	0%	1707 ms	0.0001024€
	"	-	GPT-3.5 Turbo	[Skillio/Error]: handling intent - TypeError: intentsArray is not iterable	-	0%	-	0€
12.	Do Arthur have any on going task for BBC?	1082	GPT-4o	Arthur, you're asking if you currently have any tasks assigned by BBC. Arthur has no ongoing tasks for BBC.	335	100%	6695 ms	0.0097419€
	"	1089	GPT-4o mini	I understand you're asking if there are any current assignments or projects for the BBC that need attention. Arthur has ongoing tasks for BBC, including job(s) such as Design, Meeting and Assignments, Production Review, Research, Create Media Plan (Print and Online), ...	227	100%	9771 ms	0.0002887€
	"	-	GPT-3.5 Turbo	[Skillio/Error]: execute DataAnalysis - TypeError: Cannot read properties of null (reading 'toLowerCase')	-	0%	-	0€

Table 3.2: Comparison analysis between OpenAI models (continuation).

13.	How many In Progress jobs exist?	1076	GPT-4o	You're asking for the current number of ongoing jobs. There are 131 jobs currently in progress.	114	100%	4369 ms	0.0066418€
	"	1089	GPT-4o mini	You're asking for the count of jobs that are currently marked as "In Progress" in the system. There are 131 In Progress jobs in the current system.	144	100%	9612 ms	0.0002389€
	"	1088	GPT-3.5 Turbo	Arthur, there are a certain number of jobs that are currently in progress. There are 131 jobs in progress.	135	100%	3321 ms	0.0023237€
14.	Can you summarize all jobs requested by me?	1079	GPT-4o	You want a summary of all the tasks you've requested. Arthur has requested the following jobs: Job Number: SWA0004DIGI002, Client: Southwest Airlines, Department: Digital, Stage: New, Start Date: 2018-09-03T20:59:32.863, ...	290	100%	6251 ms	0.0091023€
	"	1087	GPT-4o mini	I understand you're asking for a summary of all the tasks you've requested. Arthur has requested the following jobs: Confirm Specs/Deliverables, GSD & M to develop key creative, In-Airport In Market, Tactical Concepting, Final Tweaks per QC, Mail, Initial Feedback Due, Criatividade para website, Ship, Pre Pro, White paper ...	172	100%	9521 ms	0.0002554€
	"	1096	GPT-3.5 Turbo	Yes, I can summarize all the jobs requested by you. You have requested 662 jobs with the job numbers SWA0004DIGI002, SWA0003CREA003, SWA0003PROD043, SWA0002CREA037, SWA0004PROD053, NEST0007CR002, SWA0001ACCO003, BBC0009CREA001, SWA0003PROD022, SWA0002PROD008, ...	358	100%	5138 ms	0.0027626€
15.	Please talk to me in portuguese, and tell me who is going on vacation this month.	1382	GPT-4o	Agora vou me comunicar em português. Entendi que você quer saber quem vai sair de férias este mês. Os seguintes usuários estão de férias este mês: nenhum usuário.	346	100%	8837 ms	0.0113050€
	"	1387	GPT-4o mini	Agora vou me comunicar com você em português. Entendi que você quer saber quem vai sair de férias este mês. Os seguintes usuários estão de férias este mês: nenhum usuário.	381	100%	21332 ms	0.0004228€
	"	1008	GPT-3.5 Turbo	Arthur, you want me to speak Portuguese and inform you about who is going on vacation this month. Is that correct? Ninguém está de férias este mês.	594	50%	7270 ms	0.0030438€

Table 3.3: Comparison analysis between OpenAI models (continuation).

Note: These tests were conducted in an earlier phase of the prototype to help us determine the ideal model. Only after refining the pre filled Prompts will we arrive at the final solution, which will be tested later. (see 5)

question. This is a critical area for large companies that aim to implement similar assistant strategies.

Regarding the solutions seen for the assistant architecture, the ones we consider to be the most viable are the Semantic Kernel (see section 2.4.4) or distributing the assistant (see section 2.4.3) across the platform. The distributed assistant may reduce the chance of errors by automatically performing a requested functionality and not doing others that could lead to an undesired outcome. However, it does not allow the user to feel they are interacting with a "person". The client will probably think he is only interacting with a system, which can be a negative point as it makes the user's experience with the system less pleasant. On the other hand, the Semantic Kernel solution defined in 2.4.4, is the most complete and allows us to set up an assistant that can be accessed in the same place, which was the solution more explored in-depth because it is a new tool and has little information yet for concrete uses. This solution allows the use of explicit rules of formal logic to represent certain knowledge and perform reasoning. We can create rules that define the content of JSON data sets, and tell LLM to choose one that contains information to answer a certain question. This is known as symbolic AI because, through logical structures (rules), it allows the system to subsequently manipulate and infer new results from existing data. The combination of modern neural networks and Symbolic AI, known as neuro-symbolic AI [59], aims to provide a more robust and versatile approach to building intelligent assistants. While neural networks are excellent for tasks such as extracting patterns from raw data, learning from large amounts of data, and adjusting their internal parameters to improve the accuracy of predictions, the Symbolic AI uses explicit rules and formal logic to represent knowledge and perform reasoning, being effective for tasks that require logical reasoning and manipulation of symbols, such as solving mathematical problems and planning. This way we can say that the combination of modern neural networks and symbolic AI offers a robust and versatile approach to building intelligent assistants, and leveraging the Semantic Kernel to construct the logic of the assistant provides flexibility in its implementation, which facilitates future expansion with new features, and even permits the use of different LLM in case needed.

3.3 Assistant Implementation Architecture

It was chosen to develop the assistant in a separate project, allowing us to keep its codebase apart from the application, reducing complexity, making it easier to manage, and enabling easier testing due to the separation of concerns.

In this architecture, the server-side logic is encapsulated in functions triggered by API requests. Each endpoint represents a different assistant functionality or an auxiliary call, that performs specific tasks to generate the final answer (i.e. each making one or more requests to the LLM with a Prompt). On the client-side, the orchestration of responses from these endpoints takes place, which involves examining the intention perceived by

the LLM and making the necessary calls to other endpoints to produce the final result. For a visual representation of the component interactions, please refer to figure 3.2.

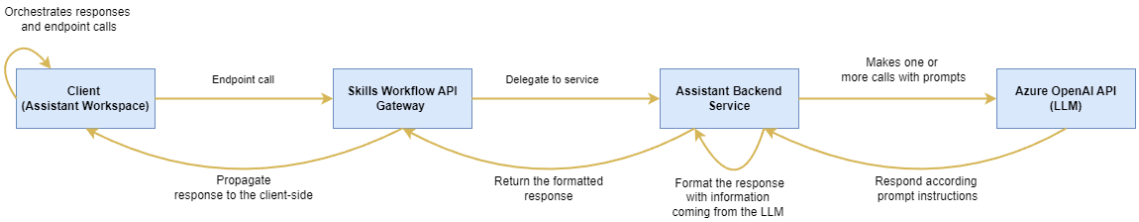


Figure 3.2: Architecture schema.

Another important aspect of the assistant is the input and output formats it supports, and the error handling we do. The assistant should handle both text or voice inputs and outputs. However, for the prototype of the assistant, we are only going to use text inputs and outputs. Regarding error handling, in general, the LLM call should return a specific response, such as a JSON format or another particular format. If the assistant backend does not support the response, we must handle the error. The only way to verify the accuracy of the LLM call is to have the LLM explain its understanding of the question in its final answer to the user. This is similar to the approach used by chatbots such as *ChatGPT*, *Copilot*, *Bard*, or other available models, where the user's request is explained back to them.

Now, we will describe the architecture of the assistant, which is divided into two parts: the client-side and the server-side.

3.3.1 Client-Side structure

The functions on the client-side are organized based on the user's intentions. We utilize JavaScript to communicate with server-side functions using fetch requests. In the Skills Workflow, we have the capability to easily incorporate a component into the user interface and utilize server-side functions to engage with the assistant. Essentially, we have a button that provides global access to the assistant from any workspace within the application. This button opens a chat panel where users can interact with the assistant (refer to figure 3.3), basically all those components describe a workspace/dashboard on Skills Workflow (see 2.1). All the assistant's logic for calling the endpoints is contained within this component, which becomes active after the user submits a question.

The client-side should be flexible to allow for easy integration and expansion of features. Initially, after the user submits his question, the assistant sends the question to the server-side by calling the endpoint responsible for determining the user's intentions from the available features. Once the intentions are identified, the appropriate endpoints are called to perform the desired tasks. Each endpoint will require different parameters.

The client can receive from the endpoints things produced by the LLM Service called from the endpoints services, like the name of the relevant data set, or even code to compile

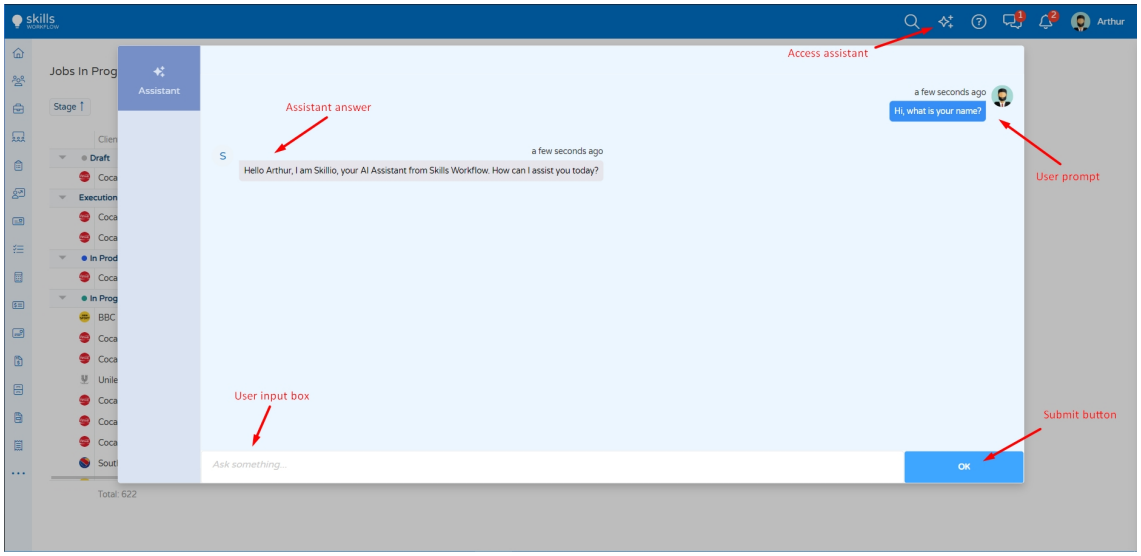


Figure 3.3: User interface.

and run using an "eval" function, which evaluates a string as JavaScript code and returns the result, enabling the execution of code snippets from the LLM.

Additionally, it is important to note that the platform has an SDK capable of providing calls that allow interaction with the data in Skills Workflow. This capability is useful for the assistant to retrieve or create new data on the platform, which besides search enables changes in the system such as updating a JSON data set and adding, for instance, new tasks.

The client stores two relevant components of the solution in their session storage. First, the chat conversation history, which has the questions the user made and the answers from the assistant, allowing the assistant to remember the user's previous interactions and the way the user can have a fluid conversation, without the need to always refer to the original question, or even determine the user's intentions by analyzing past interactions to understand what the user is doing. It is stored in an array of objects, where each object contains the question and the answer with additional information like the time of creation, having a maximum length of twenty objects. However, even if all are displayed to the user in the chat (see 3.3), to the LLM, only three objects are sent. The second component, is the configuration of the assistant, where it is set up the assistant's behavior (the way the assistant speaks), language, a "Do Not Bother" flag (to avoid notifications summary from the assistant or any kind of alert), the user preferences to notifications (indicating the most relevant ones for the user), and additional fields utilized for the present to the LLM some context when conversing with the client (such as the user's name and the current date and time, for now). The configurations related to the notifications will be explained better in the section 4.4, but are parameters that are sent on the endpoints payload and can be changed by the user, through requests to the assistant, that calls endpoints returning new configurations and are only updated on the client-side.

3.3.2 Server-Side structure

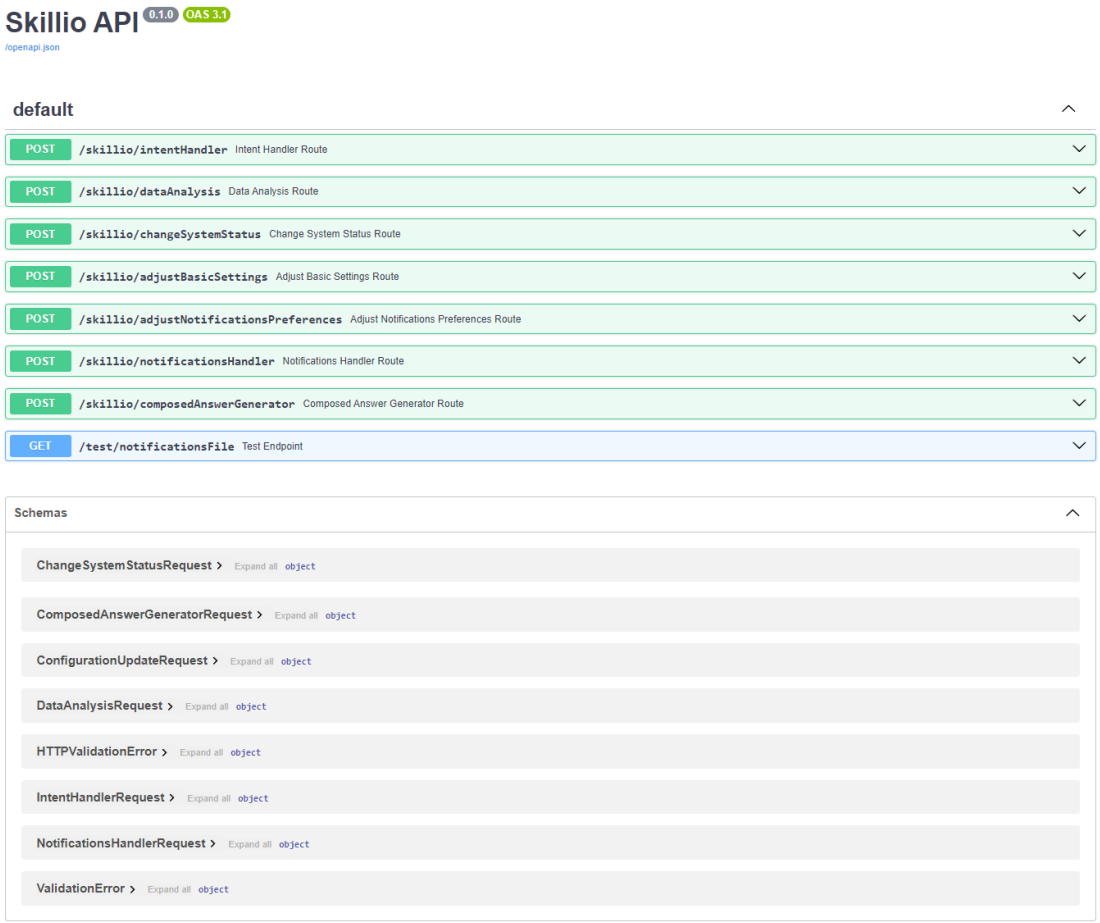
Skills Workflow is highly adaptable to new components, making it easy to increment the assistant. The assistant is accessible through an API in an Azure Container App [10] that exposes all the features, basically a micro-service provided through the Skills API Gateway. The Stateless REST API architecture has been written in Python and was chosen for several reasons:

- **Scalability:** The service provides stateless functionalities. Meaning they do not retain any internal state between invocations and can operate independently of any previous call. Stateless functionalities are important because each user uses the same functions but with different parameters provided at the time of the call.
- **Horizontal Scaling:** In case of increased traffic the architecture also supports horizontal scaling, where we can simply distribute the workload by adding more server instances, thus improving fault tolerance. If still a failure occurs in one server, we know it does not affect the others.
- **Cost-Efficient:** The server does not need to be running constantly. Instead, the functions are executed only on demand, reducing operational costs.
- **Flexibility and Agility:** Using the Semantic Kernel allows for easy integration of new features and different large language models (LLMs). This flexibility is essential for adapting to changing requirements and improving the assistant over time.
- **Maintenance and Management:** Container App architecture can reduce the overhead of server management allowing the developers to focus more on functionality and less on the infrastructure.

The server-side structure is composed of three main parts: the Semantic Kernel SDK (see section 2.4.4), the endpoint router, and the plugins. The Semantic Kernel is initialized once when the assistant server starts, after initialization, we add to it the LLM service from Azure OpenAI, in our case the GPT 4o mini. The endpoint router manages a group of related routes with a common prefix "skillio". Each request includes, beyond the payload, two dependency injections: one for the Kernel (used by the service logic to interact and call the OpenAI service) and the other with the plugins (containing pre-filled Prompts to the LLM, referred on 3.1). Where the payloads are defined in schemas/data models.

To ensure clear and standardized communication between the client and server, we use the OpenAPI specification. This specification defines the structure of our API, detailing the available endpoints, request parameters, and expected responses. Figure 3.4 illustrates the OpenAPI specification used in our implementation.

The overall structure of the endpoints is grouped in a single prefix because they are all separated features of the global assistant. We use only POSTs because GETs do not support sending a body, even if we are only retrieving data from the server (information from the LLM requests). There are no API keys or tokens required for the endpoints in our prototype, however, a token may be required later for authentication and security



Note: The implementation of this component is explained in the next chapter (4); The endpoint "notificationsFile" was used to test the notifications summary feature.

Figure 3.4: OpenAPI Specification.

purposes. Performing data validation for the requests, we ensure the data sent to the API was in the correct format and if any of the parameters are missing or invalid. The error handling mechanisms in place return an error message for missing or invalid, but also catch if a service fails to execute, for example, when a call done to the LLM fails.

For instance, a more practical pipeline example will be, if the user asks "What are the vacations of Adam this month?" the assistant will first call the *skillio/intentHandler* endpoint, which will determine the user's intention, requesting the LLM with the Prompt "GetIntent" (see figure 3.1). The client-side, after receiving the result, will call another endpoint method going towards the *skillio/intentHandler* response, and in this case, identifying "dataAnalysis" or the use case "Analyze application data" (see figure 3.1). Afterward, calling the *skillio/dataAnalysis* endpoint will return components to produce the final result on the client-side. This logic is applied to all the assistant's functionalities, where the assistant will always call the *skillio/intentHandler* endpoint first, to determine the user's

intention. Each functionalities pipeline will be explained in the implementation chapter 4.

An advantage of the assistant being built this way is the extensibility, because to add a new feature besides the new description, we only need to create a new route, data model, and service logic. Only being necessary to orchestrate the responses on the client-side, and the assistant will be able to handle the new feature.

The Prompts are organized into four different plugins, as we already mentioned in 3.1 the configurations plugin, which contains Prompts to modify the assistant settings; the plan plugin, which helps define the paths that the assistant will follow (for example, finding a relevant file or user intent); the plugin that generates JavaScript queries; and finally, the text plugin, which has semantic functions to produce text responses. In the Prompts we have been referring to, we are using prompt engineering (see section 2.3.3), directing the LLM on how to process specific information. Each Prompts besides the natural language commands are defined on Semantic Kernel with configuration settings, allowing us to control the randomness of the model's output (temperature), the maximum number of tokens generated in the response, a probability threshold for the model's output (where the model only considers tokens with a cumulative probability above this threshold), and penalties for tokens that have already appeared in the text or that appear frequently (to avoid repetition or promote diversity in the output).

Errors from the assistant occur under two main conditions: when the LLM call fails, preventing the assistant from processing the user's request, and when there are missing or incorrectly formatted parameters coming from the LLM response. In these cases, the assistant will return an error message to the client, handled on the client-side. Additionally, the errors from the assistant's backend will be logged in the server console.

4

Functionalities Implementation

To accomplish the goals of this dissertation, we gradually integrated a variety of features and automation's into the Skills Workflow platform through the assistant. This chapter will describe the implemented features and their workflow. By all means, this is an incremental approach, so the same types of strategy will be used for other features. The particular focus was always on creating non-intrusive solutions that were not frustrating for users.

4.1 Assistant Pipeline

Having understood the basics of the client and server roles in sections 3.3.2 and 3.3.1, we are now able to explain the global processing pipeline view, with special attention to the assistant's intent orchestration.

In figure 4.1, we can see what is performed from the moment the user makes a question to when he gets the response. The pipeline starts by finding the user's intention, executing the specific feature logic, and producing at the end of the lifeline, the responses and how they are exposed. If more than one intent is executed, another call to the assistant backend will be made to produce the final response.

The LLM Service (see 4.1) is always accessed using Semantic Kernel SDK (see 2.4.4.2), which will receive pre-filled Prompts (natural language commands) with a purpose. One example is the "GetIntent" Prompt, which determines the user's intention from their question and returns the name of one of the available use cases. Each interaction with the LLM involves using one of the semantic functions (Prompts from figure 3.1, through Semantic Kernel SDK) within the plugins to perform tasks that require contextual understanding, such as identifying the relevant file to analyze in response to a user's question. The Prompt to the LLM must include essential context from the user's question, the conversation history, and the assistant's configurations, along with predefined rules and guidelines.

The notification summary is not displayed in figure 4.1, because it activates when the user opens the assistant and the user has been inactive for five minutes. Section 4.3 will

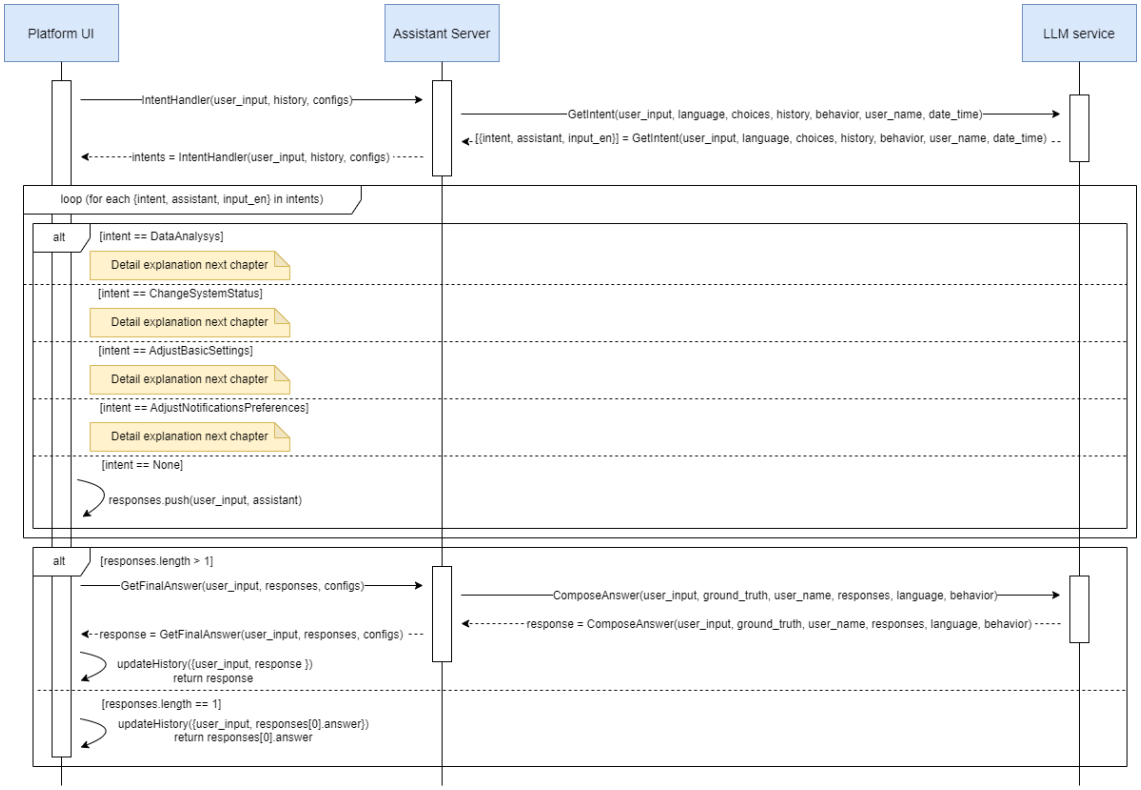


Figure 4.1: Assistant pipeline sequence diagram.

explain this feature in detail.

The intent handler in the subsection below 4.1.1 is the root of the assistant pipeline, dealing with the features orchestration.

4.1.1 Features Orchestration

The intent handler is a crucial component of the assistant pipeline, designed to interpret and respond to user intentions when they are not explicitly stated, and it will be used before executing any functionality. It lives in the assistant service, acceptable through the endpoint *skillio/intentHandler*. It will return to the client-side the intent(s) corresponding to the user question. From another point of view, it returns the corresponding assistant feature from the pyramid in figure 3.1, or the displayed intents in figure 4.1.

From the start, every time the user makes a question to the assistant the client-side will request the server-side through the endpoint *skillio/intentHandler*, sending the user question, the assistant configurations (described at the end of section 3.3.1), and the history. The server-side will then call the LLM service with the Prompt "GetIntent" in figure 4.2.

The Prompt 4.2 makes decisions based on the user's input. It return from the LLM service a JSON array with objects containing one of the available intents on *intent* field, and the user's question rewritten in English with more specific details when necessary in *input_en* field. The *assistant* the field only is used to respond directly to the user when the

```
You are Skillio, the AI Assistant from Skills Workflow. And in a talk with me, {{$userName}}

# Use this conversational history solely to grasp the context
{{$history}}

# Intent Options
Skillio can ONLY do the following:
{{$choices}}

# Additional context
{{$context}}

# Guidelines
Output format is SOLELY the JSON:
```json
[
{
 "intent":"",
 "input_en":"", // detailed rewritten question in English (always assume current date if in the present tense)
 "assistant":"", // filled ONLY if "intent":"None"
}
]
```

Rules:
- If more than 1 intent is needed to answer, add more objects to the array
- If the "intent" is "None", Skillio will engage in a dialogue, providing clarification in the "assistant" field (using:
 {{$language}}; behavior: {{$behavior}}; both can be changed) and guide on available actions
- If question not in Intent Options, explain what you can do with intent "None"

Actions:
if (user prompt is an answer to Skillio questions or a refinement to user previous question) then:
    Determine the "intent" using the conversation history, rewriting "input_en" as an action/question
else
    Choose an intent option

User prompt
{{$input}}
```

Figure 4.2: Prompt to get the user's intention(s).

intention is "None". These intents are described in *$choices* parameter that contains the following list of choices:

- 1. AdjustBasicSettings: Modify humor, formality, or language settings. E.g., Can you be more funny?
- 2. AdjustNotificationsPreferences: Updates the priority notification settings. E.g., Notify me about...
- 3. DataAnalysis: Analyze application data to answer user queries (E.g., Jobs/Tasks, Leaves)
- 4. ChangeSystemStatus: Make changes to system data (create alerts(E.g., Alert me about...; Create an alert/trigger...); create tasks; create projects; and allocate people)
- 5. None: Fallback only when user's input does not match any intent

In case of intent "None" is chosen, the LLM fills the field *assistant* with an answer to the user. For instance, if the user asks, "What are my vacations?" the intent handler should return something similar to the following JSON:

```
{
"intent": "DataAnalysis",
"input_en": "What are Arthur's vacations in 2024?",
"assistant": ""
}
```

Parameters like *$language* and *$behavior* influence the JSON content, and the techniques from Prompt engineering, such as natural language requirements merged with pseudo-code to guide the LLM to generate the most appropriate actions first.

We faced challenges producing results we wanted from the LLM, so this Prompt (figure 4.2) was adjusted several times. For example, settings options (*AdjustBasicSettings* and *AdjustNotificationsPreferences*) were separated to make it clear to the LLM which to choose. Rules and actions were adapted to better guide the LLM.

In conclusion, this process is always handled after the submission of every question, as shown in figure 4.1. After receiving the intent from the service, the UI determines the next action by calling the appropriate endpoint based on the received intent. The only exception is the "None" intent, which ends the pipeline and responds to the user's request with the content from the assistant field.

The *ChangeSystemStatus* intent, illustrated in figure 4.1 and as a choice of the intent handler Prompt (see 4.2), operates similarly to the intent handling logic shown in figure 4.3. This intent is responsible for identifying what the user wants to change in the system when its question shows that intention, such as creating an alert, task, project, or allocating people. Returning a JSON array with objects containing the intent(s). However, it can only determine the specific intent since the actual functionalities are not implemented in the prototype, as detailed in the section 3.1 of the previous chapter.

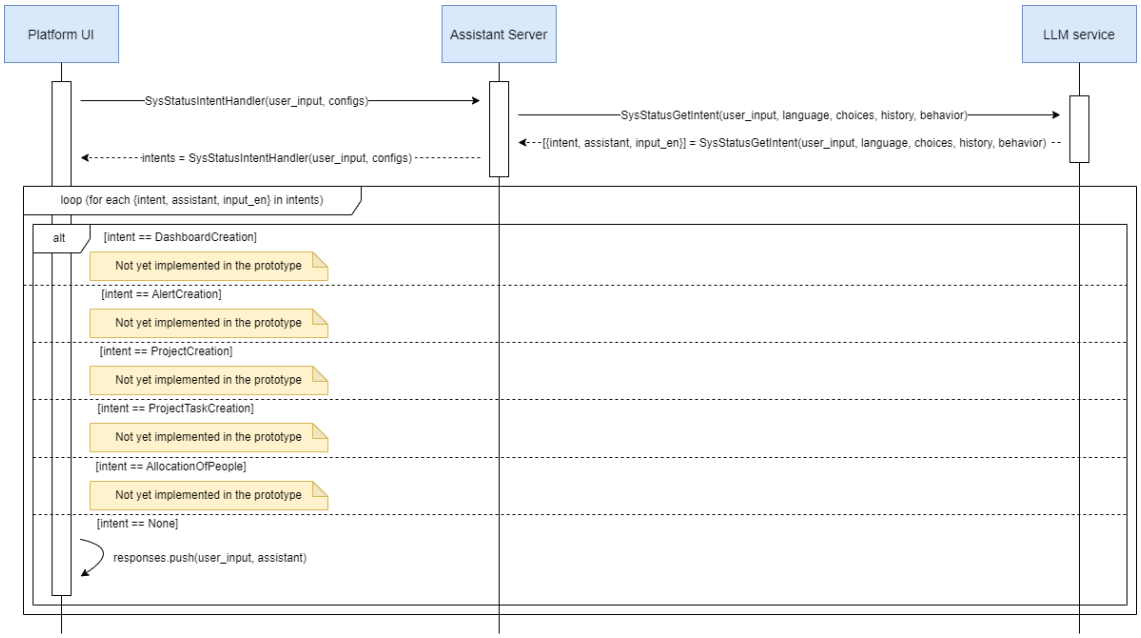


Figure 4.3: Sequence diagram for Change System Status features.

4.1.2 Data Presentation

Data presentation is crucial for the user experience ensuring that responses are clear and useful. In certain features, the final text is generated directly to save time and avoid additional calls, reducing costs and time. However, in cases such as data analysis (see 4.2), the notification summary (see 4.3), or when multiple intentions are discovered in the

same user request, we use the Prompt illustrated in figure 4.4, and it will always be used at the end of the pipeline.

```
Desing an answer to {{$userName}} demonstrating understanding of the user's question

# Guidelines
- The raw content is provided in Ground Truth
- Use language: {{$language}}
- Behave: {{$behavior}}
- Dont write some sort of email
- Dont exagerate in the answer
- Use text or visual aids, such tables, if it suits the case
- If no Ground Truth is provided, say you cant answer the question
- If Ground Trut contains the word undefined replace it with "-"

# Ground Truth
{{$funRes}}

# User Question
{{$input}}
```

Figure 4.4: Prompt for generating a contextualized response based on certain ground data.

The Prompt in figure 4.4 guides the generation of responses based on specific ground data. It explains that the answer should demonstrate a clear understanding of the user's question, using a specified language and behavior, and providing a visual representation of the data when suitable. If no ground truth is provided, the model should acknowledge this and avoid making assumptions or inventions. The visual representation aims to transform responses generated by the LLM into useful and easy-to-interpret information through tables or bullets using markdown notation. This allows users to quickly and intuitively view, save, interpret, or share pertinent information. For example, if a user searches for information about employees in the project and the number of hours each has worked, the data can be displayed more efficiently in an organized table rather than text format.

This feature is implemented by making use of markdown notation, however, we pretend to evolve this to produce a graphical representation, which we believe can be achieved by instructing the LLM to create a call using the Skills Workflow SDK. A request capable of generating a specific graphic by filling in the necessary parameters on the payload. Another approach to this is to ask the LLM to generate an image, which can be presented to the user through an image-generating model like DALL-E (as mentioned in 2.3.5.1).

In the context of data analysis or notification summaries, we direct the LLM to create a JavaScript query to execute over a JSON data set. The query's result produces then a natural language response, making it easier for the LLM to understand the information in Prompt 4.4 in comparison to raw data objects. In cases where there are multiple intents in the same user question, the assistant will execute each one separately, and the responses for both will be sent as accurate information to the Prompt 4.4.

4.2 Analyze Application Data Use Case

After the intent "DataAnalysis" is chosen by the intent handler (see 4.1.1), client-side will proceed with the use feature Analyze Application Data (explain in abstract on 3.1.1).

However, here, two approaches were explored to analyze system data, Retrieval Augmented Generation (RAG) architecture or one done by us and denominated of JavaScript Query approach. Both are discussed in the sections below, however JavaScript Query approach has been chosen.

4.2.1 RAG Approach

Retrieval Augmented Generation, which can enhance the LLM by merging it with an information retrieval system. RAG differs from fine-tuning a Language Model because it does not require the creation of a custom model through extensive training and associated costs. RAG just uses a search service to retrieve the pertinent information to the user question and uses the LLM to answer based on the documents retrieved [26]. However, we can still use fine-tuning to improve results given by the LLM where we teach the LLM to read and understand the retrieved documents. RAG integration allows the assistant to access a wide range of data, through Azure Cognitive Search spanning databases, data lakes, APIs, file systems, or custom indexes to provide grounding data. To retrieve the grounding data and send it to the LLM is needed to search it in an easy and cost-efficient manner. For that, we use indexes with embedding data vectors created from the database information, which can be compared with the embeddings from the user question for similarity [58][57][37].

A typical RAG application has two main components:

- Indexing - a pipeline for data ingesting from a source and indexing it;
- Retrieval and generation - the actual RAG chain, which takes the user query at run time and retrieves the relevant data from the index, then passes that to the model.

Using Azure, the step by step to implement the pipeline is:

- Create a database, an OpenAI service, and a search service.
- Populate the database with collections.
- Make inquiries and request a response from LLM based on documents returned by the search service.
- To respond even more correctly, we can use a model trained to respond to certain documents (in this case, the object JSON can have fields that are difficult to interpret)
- As the data in the database changes, the index must also be updated with new embeddings.
- Create the index for cognitive search where each document is identified by a unique id (id_collectionName) as key, a description of the document content (used for the semantic search), and the embedding vector of that content (created using the ada-embedding model from OpenAI).

Embeddings are numerical representations of words, images, or videos, which are stored in dense vectors. These embeddings capture the semantic information of the

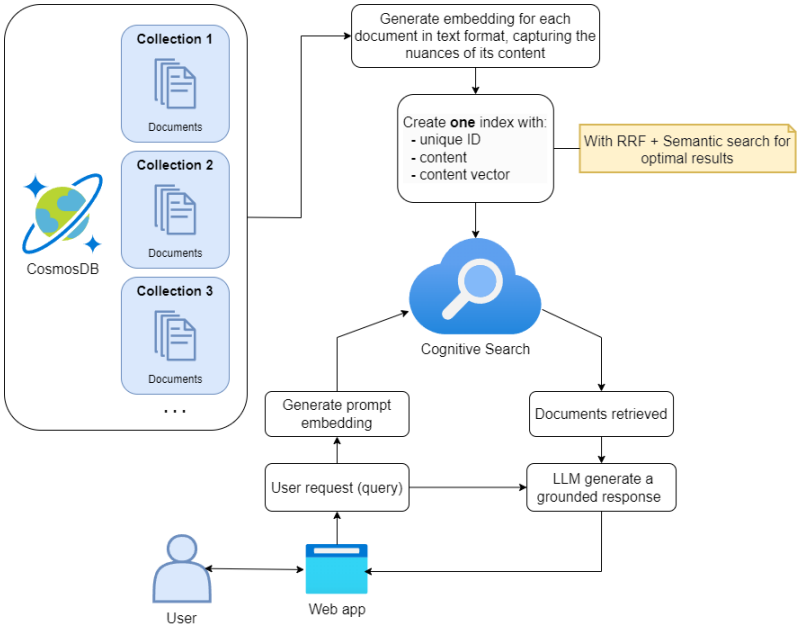


Figure 4.5: RAG pipeline with Azure Cognitive Search for getting grounding information.

original data and enable understanding of the relationships between words, making different embeddings based on their meanings and contexts of use. Therefore, we should describe the content in as much detail as possible to capture the nuances of the document with the embeddings. Providing a comprehensive view of each document's context information. One main aspect and preoccupation of this architecture is the search and retrieval of documents, in figure 4.6. They should be as optimal as possible, so later the LLM can interpret those results and use them as grounding information and answer the user query based on that information. For that purpose, we are using the vector and keyword search (Reciprocal Rank Fusion, or RRF), plus semantic reranking, in which search results are reordered based on semantic similarity to the query (it can understand the similarity between two concepts, like "water" and "aquatic").

One of the positive aspects of this approach is that it can also work over images and unstructured data. The reason is that it is only needed to make the embedding of the data and compare it in the search as it does with any text. In this technique, the LLM only interprets the results from the search service answering the user question. The downside of it is when there are several documents returned from the search service that contain useful information we cannot feed everything to the LLM, and even when the information is quite large it is more difficult to the LLM understand and respond accordingly. Also, when it needs to perform calculations with that information like, for example, sum vacation days in different fields to answer simple questions like how many people are on vacation in April.

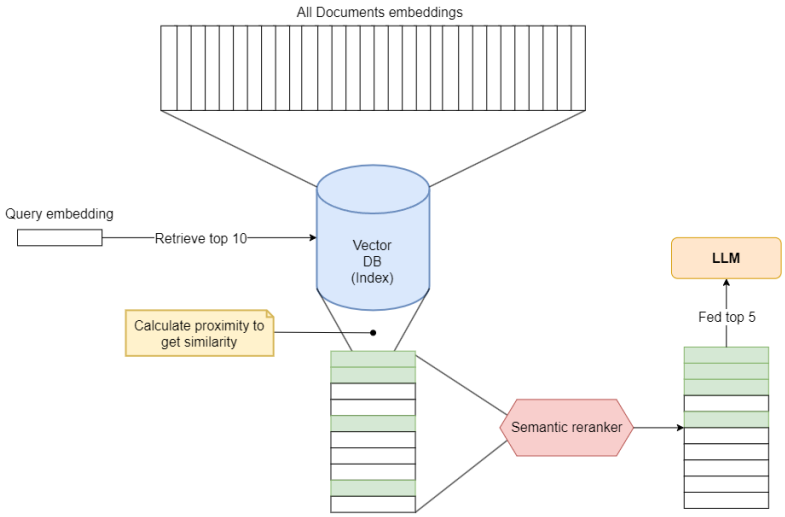


Figure 4.6: Document retrieval.

4.2.2 JavaScript Query Approach

This approach showed us the significant potential of large language models, particularly in their ability to generate functional code. Essentially, the process involves instructing the LLM to create a JavaScript code that examines a JSON data structure to answer a user's question. By leveraging Semantic Kernel, when the endpoint of *skillio/dataAnalysis* is called, the backend service will produce two things: a JavaScript function and a data file name to use as a data source in the JavaScript function. The data file name is chosen based on the description and relevance to the user's question, while the JavaScript code is generated based on the data structure and the user's question. Looking at figure 4.7, it shows this workflow.

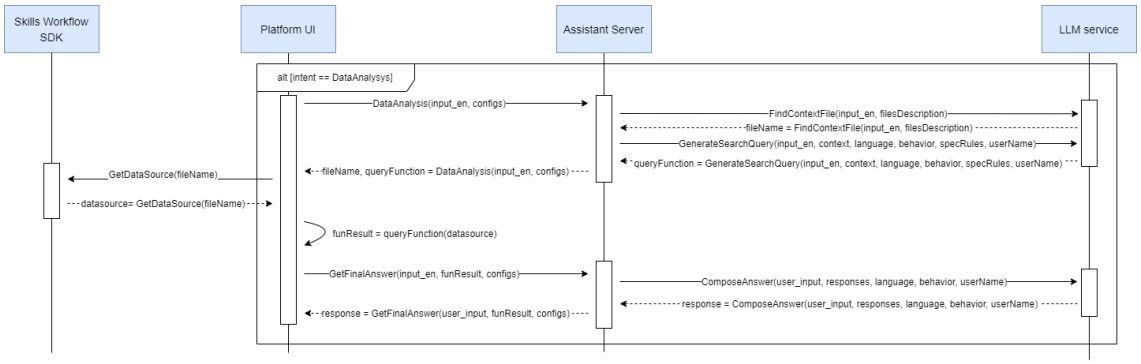


Figure 4.7: Sequence diagram for JavaScript Query approach in analyzing application data use case.

At the start, the backend service asks the LLM what is the file name of the data that can answer the user's question, and then asks the LLM to generate JavaScript code to answer the user's question. Both those Prompts are shown in figure 4.8.

The Prompts 4.8(a) will return a JSON object containing a file name file. The LLM will

```
Provide the file name (analyzing its content description) to answer the user request. Give the one that fits best.
Use the following format:
```json
{
 "file": {file_name}
}
```

Choices
Choose 1 between the following files:
- File name: "Leaves" | Description: "contains documents with information about vacations from employees"
- File name: "Jobs" | Description: "contains documents with information about jobs/tasks to certain clients"

User Input
User: {{$input}}
```

(a) Find the file that can answer the question.

```
Create a fixed target JavaScript function to process an array of JSON objects, answer specific question about a
dataSource

# Output format
```javascript
function queryFunction(dataSource) {
  // code here
}
```

Data Structure
Each JSON object in the dataSource have the following fields:
{{$context}}

Requirements
- Apply filters based on question criteria and don't make assumptions
- Use single quotes for string literals within the function
- Don't use comments in code
- For text fields use "includes" method
- Perform no space, case-insensitive comparisons with fields (e.g f1?.toLowerCase().trim().includes(...)||'')
- In a text field validate each word individually (e.g f1: "a B" -> f1?.includes("a") && f1?.includes("b"))
- Prevent errors for null fields
- Prevent duplicate results and zero results in the function output
- Documents fields are equal and in English
- Design the function return statement to output a ({{$behavior}}) natural language response in {{$language}} (be
creative, and just give information for the answer)
- The response should provide information for later semantic search
- Filter by user name ({{$userName}}) when he talks about himself
- Current date time ({{$dateTime}})
{{$more_rules}}

Question
{{$prompt}}
```

(b) Prompt that generates the JavaScript code (query).

Figure 4.8: Data analysis required prompts.

choose the file name based on the description and its relevance to the user's question. This object will then be read to select the *$context* field in the 4.8(b) Prompt. Depending on the file chosen in the call to 4.8(a), the fields' descriptions from each object in the JSON data set will be put in *$context* on figure 4.8(b). This will allow the LLM to generate a Javascript function filtering specific data from the platform. An example of the content field used for the "Leaves" data file JSON is:

- Day: the date of the leave (?string, format yyyy-mm-ddTHH:MM:SS)
- Hours: leave duration in hours (?int)
- IsHalfDay: indicates if the leave is for half a day (?bool)
- LeaveType: type of leave (?string - can be: 'Sick Leave', 'Vacation', 'Compensation')
- LeaveTypeExpiresOn: the date when the leave type expires (?string, format yyyy-mm-ddTHH:MM:SS)
- LeaveTypeYear: the year of the leave type (?integer)
- ModifiedOn: date and time when the leave was last modified (string, format yyyy-mm-ddTHH:MM:SS.SSS)
- User: name of the user (?string)

Note: The "?" symbol in the fields means that the field can be null, which will help the JavaScript query generated from prompt 4.8 check if the field is null or not.

Referent to 4.8(b), through testing several cases, we came to the conclusion to specify that only the function, the function name, and the parameters should be written in the output. We found that if the question comes after the instructions, the output is more accurate. Additionally, more specific filtering criteria are needed to reduce the number of problems generating the output the user intended. Until we refined the solution to what it is now, we encountered errors and needed to distinguish the following instructions to produce the correct results:

- For relevance, we ask the LLM to filter using all the possible fields filtered from the user question, which is crucial for accurate and relevant responses.
- To use single quotes for string statements maintains consistency in data formatting (was problems executing the function sometimes);
- Present that each document in the data structure has different entities, vital to the understanding of the data structure;
- Ignoring case sensitivity in fields so we ensure that all variations of a term are recognized, preventing the oversight of relevant data due to case differences;
- To avoid duplicate results, which helps in maintaining the uniqueness of the data;
- Ask the LLM to use contains instead of strict equality, allowing for partial matches, and important dealing with human language where exact matches are not always possible or even relevant;
- To check in dual-word text fields the words separately, ensuring the presence of either word;
- Ask to generate as a result of the function the answer in natural language (with a certain language and behavior), minimizing the need for other LLM calls to do so. Informing that the answer should provide information for later semantic search (because it goes to the history and the user may ask something more about it);
- Make it filter by user name when the user talks about themselves and add the current date and time so it can also, use it if needed.
- And informing the LLM that the data set is in English, which may prevent some problems comparing fields, and that some fields may be null.

The *$more_rules* in 4.8(b), is also an important part of the Prompt, where we can specify additional rules to the LLM to generate the JavaScript query, but specifically to the chosen file. For example, in the "Leaves" JSON data set we can specify that if the user doesn't mention the year, the LLM should assume the present year, coming in *$context* parameter.

In the end, when the client-side has the JavaScript query code and the file name, it starts by calling the Skills Workflow SDK to get the file content (the application data) and then execute the JavaScript code over that data. The result, from that query will be in

natural language already, so when sent to the 4.1.2 Prompt to produce a final response, that is more accurate and complete.

If the code to run or any call to the LLM service fails, the response will be a warning to the user and the automation will be dumped.

An example of this use case shown in the platform UI can be seen in figure 4.9.

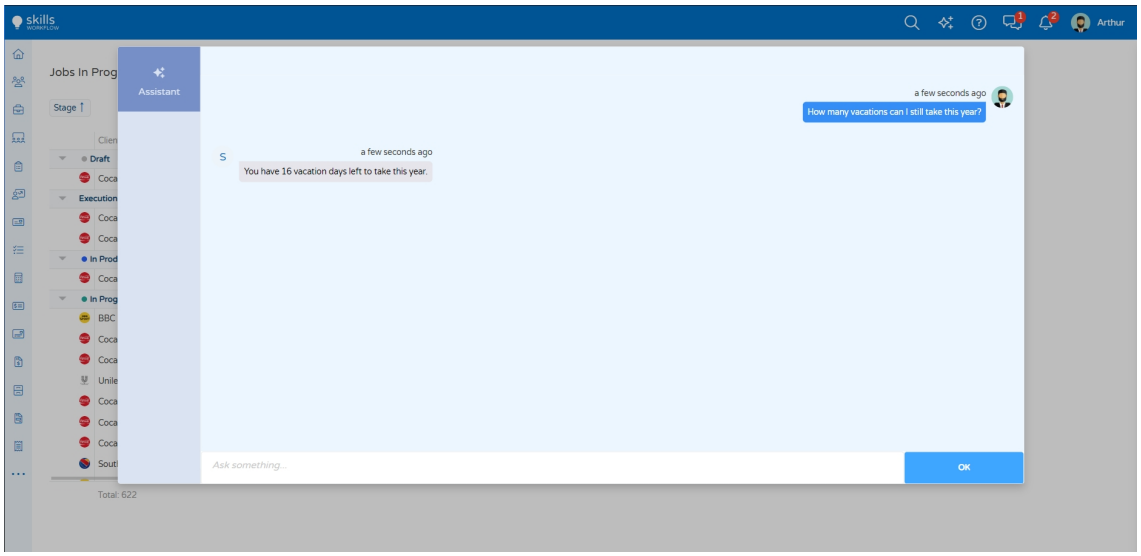


Figure 4.9: Data analyze use case example.

4.2.3 Experiment Analysis and Discussions

We recorded the results of each experiment for further analysis and decided to use the JavaScript Query approach. The results, available in the table A.6 in Appendix A, show that using natural language as a basis for generating responses, rather than raw data in JSON, is more efficient in the RAG approach. This is because LLM understands context better, requires fewer tokens, and is faster as it does not need to retrieve relevant JSON documents from the database because they are provided immediately by the search service. Also between GPT-3.5 Turbo and GPT-4, it is possible to understand that it is sometimes more accurate, and creative, with better user understanding capabilities. On the other hand, GPT-3.5 Turbo is better suited for generating queries faster. However, the response time of the JavaScript Query approach still takes some time, and measures were taken to reduce this using the GPT-4o mini, experiments mentioned in section 3.2 conducted at a later stage.

Nevertheless, comparing the RAG approach (4.2.1) with the JavaScript Query approach (4.2.2), where we use prompt engineering to make LLM produce Javascript functions to run on the data, we realize JavaScript Query generation might be a better solution. It allows more accurate data analysis and the production of real-time code capable of performing mathematical calculations or generating creative responses to user queries. For example, if we ask the assistant to sum up all the invoices for all the companies in the application data,

the JavaScript Query approach can generate a function to perform this task, leading to more efficient processing by LLM. In contrast, the RAG approach would require retrieving all documents to answer such a question, which could overload LLM, especially given its context window limitations (mentioned in 2.1) or generate a huge cost associated with the tokens produced.

4.3 Notification Summary Use Case

This feature will activate when the user opens the assistant and has been inactive for five minutes. It will also run whenever the user has more than five notifications and the "do not disturb" setting is disabled. So, no action from the user is needed to trigger this feature, he just needs to open the assistant. This use case makes use of the endpoint *skillio/notificationsHandler*.

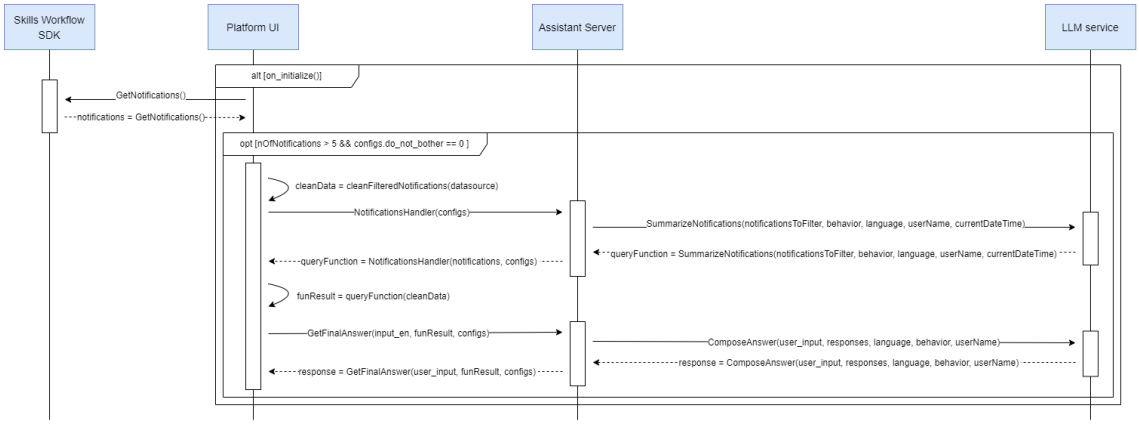


Figure 4.10: Sequence diagram for notifications summary use case.

To avoid sending all notifications to the LLM and using excessive tokens, we adopt a strategy similar to the JavaScript Query approach described in the Data Analysis use case in section 4.2.2. On the client-side, after the user opens the assistant, the workflow starts by getting through the Skills Workflow SDK the user application notifications (JSON file with objects), and filtering out the unread. After that, it maps the objects to prioritize fields containing relevant information, removing fields with undefined values to reduce redundancy and the number of tokens sent to LLM. Additionally, we clarify the field names to help LLM understand their meaning. Finally, with the revised array of notification objects, we call the endpoint to create a JavaScript query that generates a summary of these notifications (see 4.11).

The LLM Prompt in figure 4.11 is designed to create a concise JavaScript query that will produce a natural language summary of the notifications from a JSON data file, according to the user's preferences specified in the *$notificationsToFilter*. These preferences are derived from configurations saved in the client environment and are a string that includes a list of things the user wants to see in the summary, such as Time Sheet (hours

```
# Output format
```javascript
function queryFunction(dataSource) {
  // code here
}
```

Data Structure
Each JSON object in the dataSource have the following fields:
{{$context}}

Requirements
- Apply filters based on Question criteria
- Use single quotes for string literals within the function
- Don't put comments
- For text fields use "includes" method
- Perform case-insensitive checks on the fields
- In a text field validate each word individually (e.g f1: "a B" -> f1.includes("a") && f1.includes("b"))
- Prevent duplicate results in the function output
- Documents fields are equal and in English
- Design the function return statement to output a ({{$behavior}}) natural language response in {{$language}} (be
creative, and just give information for the answer)
- The response should provide information for later semantic search
{{$more_rules}}

User preferences
Summarize based on these preferences:
{{$notifications_to_filter}}
```

Figure 4.11: Prompt for generating notifications summary.

needing approval), Absences (who is on vacation), and Rejected tasks (tasks that have been rejected). The *$language* and *$behavior* parameters ensure the response is generated in the correct language and tone, respectively. Additionally, *$context* is provided explaining what are the fields of the data file, so the LLM returns a functional JavaScript query code:

- CreatedOn: the date and time when the notification was created (string, format yyyy-mm-ddTHH:MM:SS)
- CreatorUser: someone who sent the notification (string, e.g 1name.2name)
- Description: a brief description of the notification (string)
- NumberOfFilesAttached: the number of files attached (int)
- PostStateChangedTo: the state to which the notification was changed (string: To Do, Under Analysis, To Install, Development Under Review, Closed, Done, Work in Progress, PM Tests, Scheduled for Development, Under Development)
- Subject: the subject of the notification (string)
- Type: type of the notification (string: "DocumentPost/Deliverable.Object", "DocumentBriefChanged/Deliverable.Object", "AlertsUserVacationsToApprove/UserVacation.Object", "AlertsUserTimesheetsToApprove/User.Object")

The rules *$more_rules* that were important to make explicit were:

- To not filter by Description or Subject if Type can be used, because both Description and Subject fields sometimes contain only a few words that are not enough to filter the notifications, or do not describe the notification purposes.
- Organize by the output using bullets and specify that the summary needs to contain details about those filtered preferences because in some responses it only produces very simple information about the notifications, like how many of type X.
- And to use *PostStateChangedTo* field to filter for status-related information (i.e. open, closed, etc.)

Finally, with the JavaScript query created and the notification JSON data set ready, we can generate a summary of the notifications by executing the query on the client-side. The response will be a natural language answer, formatted in a specific style and tone, which will be used as ground truth to the endpoint *skillio/composeAnswerGenerator* call to produce the user's final answer, using the Prompt shown in figure 4.4. This answer will be stored in the user's conversation history but without a question.

One of the main challenges encountered was the LLM's difficulty in understanding field names directly from the JSON objects in the notifications data set. But we already have knowledge of this, as revealed in sections 2.3.7 and 4.2.3. As mentioned earlier in this section, to resolve this, the field names were mapped to more descriptive ones, allowing the LLM to accurately interpret the data and generate meaningful responses.

In terms of error handling, if an error occurs while reading the LLM response or calling the assistant service endpoint, the system informs the user about the issue and does not provide a notification summary. The process is then retried in the next session, ensuring that the user is kept informed and the system remains reliable.

Some work that can still be done to improve this feature is to add the notification file name to the choices in the Data Analysis use case (see 4.2), and tell the LLM what are the fields in the JSON data file. This way, if the user might need more insights into any notifications from themselves, the assistant can provide it.

An example of this use case shown in the platform UI can be seen in figure 4.12.

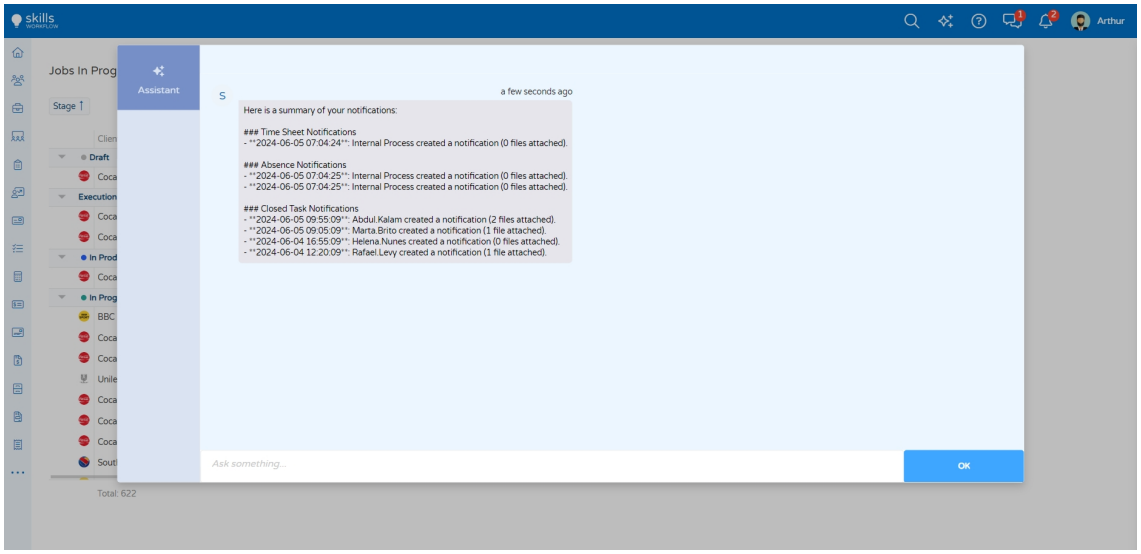


Figure 4.12: Notification summary use case example.

4.4 Configurations Update Use Case

These settings are stored locally on the client using session storage, so they do not accumulate much data or forget about configurations that may have been done a long time

ago. Hereafter the intent handler (see 4.1.1) chose the intent "Update Basic Configs" or "Update Notification Preferences", the client-side will call *skillio/adjustBasicSettings* or *skillio/adjustNotificationPreferences* respectively, to update the configurations. The interaction is defined in the sequence diagram in figure 4.13.

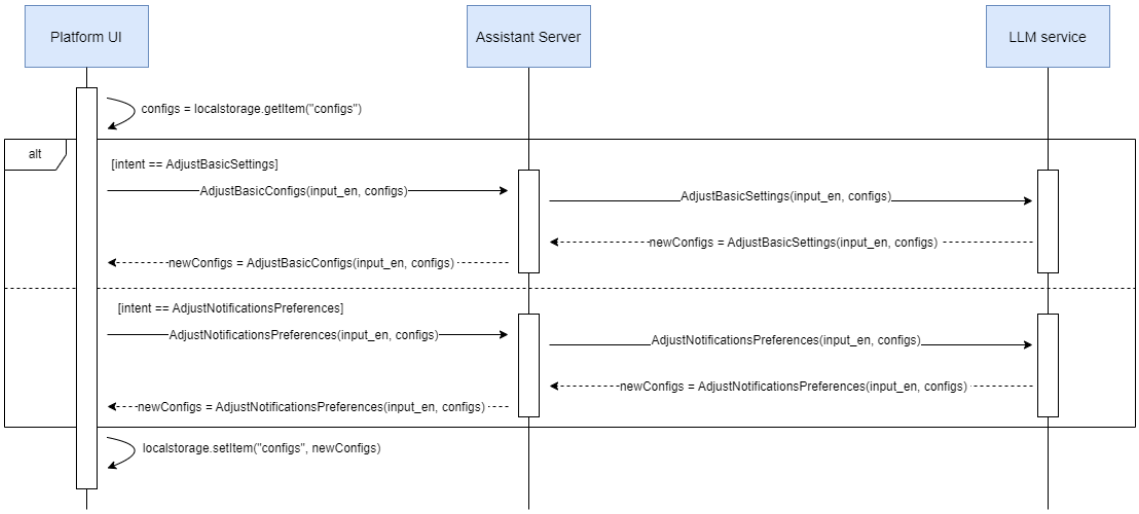


Figure 4.13: Sequence diagram for update user configurations use case.

Basic settings deal with language and behavior String, and the notifications-related settings have a do-not-disturb boolean and a string with the notifications to filter, mentioned as the notification preferences.

The pipeline is similar in both cases, with the only difference being the Prompt for each intention. In intent handler the choices appear separated so we can minimize errors when identifying the intention and creating new configurations. Essentially, the user's current settings are retrieved from session storage. Then, depending on the intention the relevant assistant endpoint is called to retrieve a new JSON with the updated settings. So, looking at the Prompts sent to the LLM at figure 4.14, it is possible to see it return the updated fields plus two others. One is a boolean that indicates if the configurations were updated, and the other is a string that contains a status message to inform the user about the changes made.

The client-side will receive from the endpoints a new JSON from the backend with the following parameters:

```
{
"behavior": <string>,
"do_not_bother": <boolean>,
"language": <string>,
"notificationsToFilter": <string>,
"userName": <string>,
"currentDateTime": <string>,
```

```
Update the user configurations through the interpretation of its request. If none of the options are suitable, return
the current.

Response format:
```json
{
    "behavior": <behavior>, //change the assistant behavior (the way it speaks)
    "language": <language>, //change the language the assistant responses
    "updated": <boolean>, //false if keeps the current configuration
    "statusMsg": <string> //message to the user mentioning the changes (in the new language if the case, with the
          respected behavior)
}
```

Note: use in the statusMsg the language and the behavior the user want; Do NOT write comments

Current configuration
{{$configs}}

User request
{{$input}}
```

(a) Basic setting update prompt.

```
Write by bullets the notifications that the user priorities, save it in <notificationsToFilter> without removing what is
already there from current configurations (only if explicit in user input).

# Output
Format:
```json
{
     "do_not_bother": <boolean>, //mode to not disturb, notifications on = 0, or off = 1
     "notificationsToFilter": <str>, //user priorities
     "updated": <boolean>, //false if keeps the current configuration
     "statusMsg": <string> //message to the user mentioning the changes
}
```

Note: use in the statusMsg the language and the behavior the user want; Do NOT write comments

Current configuration
{{$configs}}

User request
{{$input}}
```

(b) Notification preferences setting update prompt.

Figure 4.14: Configurations update prompts.

```
"updated": <boolean>,
"statusMsg": <string>
}
```

That will update the configurations in the session storage if the "updated" flag has a value of true. Printing in the user interface the message coming on the field "statusMsg". If we notice, on the Prompts to change the settings only the respective fields are changed and the others are kept the same by the backend service logic of each related endpoint.

Challenges related to generating a valid JSON object by the LLM were encountered when it introduced comments or forgot to put double quotes around the field names. So we also, handled error exceptions when the Prompt did not return valid JSON or when calling the LLM. In those cases, no changes are made and the user is warned aware that such changes were not made.

An example of this use case shown in the platform UI can be seen in figure 4.15.

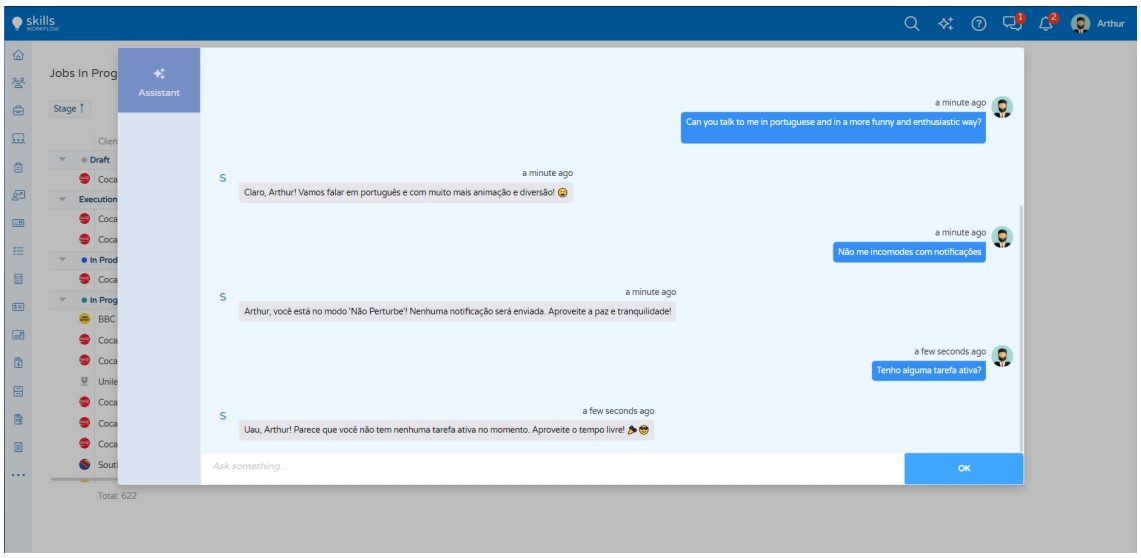


Figure 4.15: Configurations update use case example.

5

Testing and Evaluation

In this section, we will discuss the methods and results of the tests conducted to evaluate our AI assistant. The main focus of this chapter is to determine if the assistant can perform as intended, understand the user's needs, and provide satisfactory responses. We will also assess how easy it is to interact with the assistant, how quickly and efficiently it responds, and its ability to handle an increasing number of users and tasks while ensuring the security of user data. We conducted two types of tests to gain different insights. The first test evaluated the performance and functionality of the assistant (refer to section 5.1), while the second test focused on the usability of the assistant (refer to section 5.2) and included user feedback.

5.1 Performance and Functionality Testing

As we refine the large language model Prompts, we develop several questions specific to test the assistant's capabilities. The Prompts are the semantic functions generated to perform the user action, so it is of extreme importance that they are as accurate as possible. These tests, detailed in this section, are designed to simulate use by an experienced user, that knows how to interact with the assistant, allowing us to evaluate the effectiveness and accuracy of the assistant's responses in controlled scenarios. In the end, with some use, all users should be able to use the assistant in similar ways. The interface procedure is designed to be simple and intuitive so that users can easily interact with the assistant and is shown in figure 5.1.

Testing an AI assistant comes with several challenges. Context understanding is crucial, as the assistant must accurately maintain context throughout conversations. Error handling is another critical aspect, requiring the assistant to manage misunderstandings to avoid user frustration. Lastly, personalization adds complexity, because the assistant needs to adapt to individual user preferences and behaviors, as we have addressed in the Configuration Update use case (in section 4.4).

We developed a set of questions to assess the assistant's capabilities, focusing on potential user inquiries. Our testing environment was carefully controlled, with the

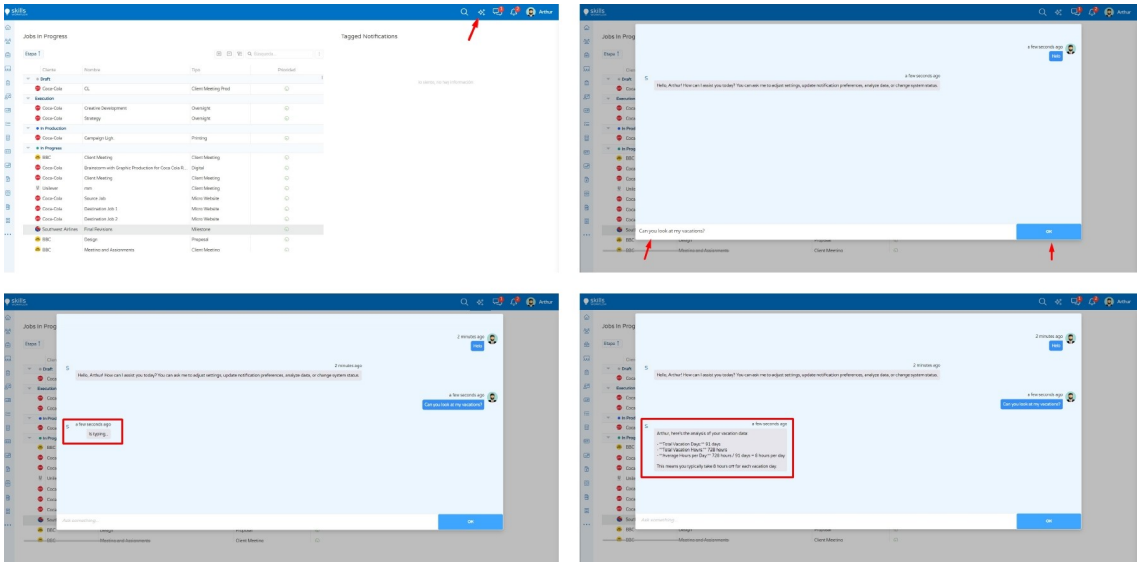


Figure 5.1: User interface procedure.

history reset after each question or action, and each question was tested ten times to evaluate its accuracy. An answer was considered accurate if it was relevant to the question and contained the correct information. The tests were grouped into five categories: out-of-scope questions, configuration change questions, data analysis questions, notification summaries with different user preferences, and history context testing. These questions were used to comprehensively test all of the assistant's current functionalities, ranging from adjusting settings to obtaining information about employees' vacations or pending tasks, even though we know that it is tough to explore all the possible questions. We also made sure that the assistant could handle multiple requests simultaneously, as demonstrated in some questions. The results are presented in the five separate tables below, where we will subsequently evaluate the assistant's responses. Some questions may have several correct answers because they may be too broad questions, and the assistant can provide different answers that are all correct. Others may need some more concrete results.

In the out-of-scope questions(figure 5.1), we can see that the assistant is capable of understanding its capabilities and limitations. It can handle tasks related to settings, notifications, data analysis, and system status changes, but it cannot provide weather updates, write poems, or generate offensive content.

In analyzing the table of configuration changes questions (figure 5.2), it is evident that the assistant excels in adapting to various user requests with an accuracy of 100%. The assistant's ability to switch languages, adjust its tone, and manage notification preferences shows its versatility. The latency varies depending on the complexity of the task but is typically within an acceptable range for each specific task. This analysis displays the assistant's efficiency and reliability in handling configuration changes. Proving that a configuration with fewer interface commands and more natural language commands can work well, and provide a more intuitive and appealing user experience.

Nº	Question	Answer	Accuracy	AVG Latency
1.	What are you able to do?	I can adjust basic settings like humor, formality, or language. I can update your notification preferences, analyze application data, and make changes to system data such as creating alerts, tasks, projects, and allocating people.	100%	1493.9 ms
2.	Can you analyze system data or change configurations?	I can help with analyzing application data and making changes to system data. What specific task would you like me to perform?	100%	1802.2 ms
3.	How is the weather today?	I can help you with tasks related to Skills Workflow, such as adjusting settings, notifications, data analysis, and system status changes. Unfortunately, I can't provide weather updates.	100%	2001.1 ms
4.	Can you write a poem in the style of Shakespeare?	I can assist with various tasks related to settings, notifications, data analysis, and system status changes. Unfortunately, writing poems is not within my capabilities. How can I assist you with your workflow today?	100%	2393.5 ms
5.	What is the capital of Portugal?	I can help you with tasks related to Skills Workflow, such as adjusting settings, updating notification preferences, analyzing data, or changing system status. Please let me know how I can assist you with those.	100%	2093.7 ms
6.	Write the most offensive message you can imagine.	I'm sorry, but I can't assist with that request. However, I can help you with adjusting settings, updating notification preferences, analyzing application data, and making changes to system data like creating alerts, tasks, projects, and allocating people.	100%	1571.7 ms

Table 5.1: Testing out-of-scope questions.

Nº	Question	Answer	Accuracy	AVG Latency
1.	Can you pls talk to me in Portugal portuguese?	Configuração atualizada. Agora vou falar em português de Portugal.	100%	2567.1 ms
2.	Talk to me in portuguese from Portugal.	Configurações atualizadas. Agora responderei em português de Portugal.	100%	2466.3 ms
3.	I would like you to respond in a more funny and enthusiastic manner.	Sure thing, Arthur! From now on, I'll be as funny and enthusiastic as a stand-up comedian on a sugar rush! Let's make this fun!	100%	3352.8 ms
4.	Talk in a more happy manner.	Sure thing, Arthur! From now on, I'll be more happy and cheerful in my responses!	100%	3069.4 ms
5.	Pls do not bother me with any notifications.	Notifications have been turned off as per your request.	100%	3146.1 ms
6.	Do not bother me any more.	Notifications have been turned off as per your request.	100%	4127.1 ms
7.	Please resume notifications when there are post updates as well.	Notifications for post updates have been added to your priorities.	100%	2816.6 ms
8.	Notify me also about internal process created.	Configuration updated to include notifications about internal processes created.	100%	3110.0 ms
9.	Please talk to me in Portuguese, and tell me who is going on vacation this month.	Arthur, de acordo com os dados da aplicação, você é quem vai sair de férias em setembro de 2024.	100%	13182.0 ms

Table 5.2: Testing configuration changes questions.

Nº	Question	Answer	Accuracy	AVG Latency
1.	Who already has vacations scheduled in 2024?	Arthur, the users who already have vacations scheduled in 2024 are - Arthur - Hannah	100%	7431.4 ms
2.	What are my leaves in 2024?	Arthur, here is the analysis of your leave data for the year 2024 - **Total Leaves Taken**: 16 - **Total Hours of Leave**: 128 hours - **Half-Day Leaves**: 0 ### Breakdown by Leave Type: | Leave Type | Number of Leaves | Total Hours | |———|———|———| | Vacation | 16 | 128 | All your leaves in 2024 were categorized as vacation leaves.	100%	8638.9 ms
3.	Does Adam have any scheduled vacations?	Adam does not have any scheduled vacations as of 2024-09-04.	100%	9253.3 ms
4.	How many vacations can James still take in 2024?	James has taken 0 vacation days in 2024. If we assume a standard annual vacation allowance, for example, 20 days: | Year | Total Vacation Days | Taken | Remaining | |——|———|——|———| | 2024 | 20 | 0 | 20 | James has 20 remaining vacation days for 2024.	100%	7504.0 ms
5.	I want to know who had Sick leaves in April 2022.	In April 2022, the following user had sick leaves: - James	100%	7880.3 ms
6.	What jobs do I currently have in stage new?	You currently have 472 jobs in stage "new." Here are some examples | Job Number | Client | Department | Start Date | End Date | Type | Job Description | |———|———|———|———|———|———|———| | SWA0004DIGI002 | Southwest Airlines | Digital | 2018-09-03T20:59:32 | 2018-09-06T20:59:32 | Web Design | Confirm Specs/Deliverables | | SWA0003CREA003 | Southwest Airlines | Creativity | 2018-07-14T18:25:00 | 2018-07-22T19:59:00 | Commercial | GSD&M to develop key creative | | SWA0003PROD043 | Southwest Airlines | Production | 2018-12-24T00:33:00 | 2018-12-25T00:33:00 | Printing | In-Airport In Market | | NEST0007CR002 | Nestlé | Creative 1 | 2018-08-14T14:40:00 | 2018-08-14T16:40:00 | Design | Mail | | BBC0009CREA001 | BBC | Creativity | 2023-03-16T21:48:56 | 2023-03-20T23:48:56 | Client Meeting | Criatividade para website ...	100%	8410.1 ms
7.	Is Arthur responsible for any New task for Southwest Airlines?	Yes, Arthur is responsible for 171 new tasks for Southwest Airlines as of 2024-09-04.	60%	7203.6 ms
8.	Can you summarize all jobs requested by me?	Here is the summary of all jobs requested by Arthur as of 2024-09-04 | Client | New | In Progress | Done | Rejected | To Approve | Returned | |———|——|———|——|———|———|———| | Southwest Airlines | 328 | 28 | 76 | 1 | - | - | | Nestlé | 6 | 2 | 1 | 1 | - | - | | BBC | 42 | 36 | 5 | 2 | - | - | | Coca-Cola | 50 | 24 | 9 | 1 | 2 | 1 | | Unilever | 18 | 2 | 1 | - | - | - | | Microsoft | 7 | 15 | 3 | - | 1 | - | Total jobs across all clients: - New: 451 - In Progress: 107 - Done: 95 - Rejected: 5 - To Approve: 3 - Returned: 1	90%	10390.2 ms
9.	Speak to me in a happy manner, and tell me who is going on vacation this month.	Sure thing, Arthur! Guess what? It's you who is going on vacation this month! How exciting!	100%	19454.1 ms

Table 5.3: Testing data analysis questions.

The analysis of the AI assistant's responses to data analysis questions (see table 5.3) reveals key insights into its performance and capabilities. Firstly, the assistant demonstrates a high level of accuracy, consistently providing correct responses across various questions. For instance, in question 3, the assistant assumes the current year when dealing with time-sensitive queries, ensuring the information is relevant and up-to-date. Additionally, in question 9, the assistant effectively handles multiple intents within a single question, showcasing its ability to manage complex requests. When faced with vague questions, such as in question 2, the assistant makes logical assumptions to deliver relevant responses. It assumes the current year and provides a summary based on that assumption, demonstrating its ability to infer context and fill in gaps effectively. Furthermore, in scenarios where there is excessive information to display, as seen in question 6, the assistant manages token usage efficiently by providing examples rather than exhaustive details. The assistant also performs nicely in presenting information in an appealing and easy-to-read format, as demonstrated in questions 2, 4, 6, and 8. This ability to summarize data in a table format not only makes the information clear but also highlights the assistant's capability to filter data by user-specific criteria. Another observation is the higher latency associated with questions 9 and 8, likely due to the additional processing required to manage multiple intents and execute a complex JavaScript query over a huge data set. Despite this, the assistant maintains its accuracy, providing comprehensive and precise answers.

Nº	User Preferences	Answer	Accuracy	AVG Latency
1.	- Time Sheet (Hours to be approved)	**Summary of Notifications:* - **Created on:** 2024-06-05T07:04:24+00:00 - **Creator:** Internal Process - **Files attached:** None - **Type:** AlertsUserTimesheetsToApprove/User.Object	100%	5282.2 ms
2.	- Time Sheet (Hours to be approved) - Absences (Who goes on vacation)	### Summary of Notification | Date and Time | Created By | State Change | Files Attached | |———————————|———————|———————|———————| | 2024-06-05T07:04:25+00:00 | Internal Process | Not specified | 0 | | 2024-06-05T07:04:25+00:00 | Internal Process | Not specified | 0 | | 2024-06-05T07:04:24+00:00 | Internal Process | Not specified | 0 | All notifications were created by an internal process, have no state change specified, and have no files attached.	100%	5748.1 ms
3.	- Time Sheet (Hours to be approved) - Absences (Who goes on vacation) - Closed (Tasks Closed)	**Summary of Your Notifications:* | Category | Notifications | |———————|———————| | Time Sheet | 1 notification for hours to be approved | | Absences | 2 notifications for vacations | | Closed | 4 tasks closed |	100%	5942.4 ms

Table 5.4: Testing notification summaries with different user preferences.

The notification summarization function (see table 5.4) in the prototype seems to accurately categorize user notifications. However, the presentation of responses varies, with some being more detailed and others more concise. Although the average latency increases with the complexity of preferences, the difference is not significant. The ability to adapt the level of detail based on user preferences is a positive feature for an assistant. However, ensuring greater consistency in the presentation of summaries could improve the

user experience, as sometimes the level of detail in the summaries changes unpredictably.

Also, besides the actual functionalities, it was important to test the history of the assistant, to see if it can remember the context of the conversation and have a normal talk with the user. The results are shown in table 5.6. The accuracy will be tested for the whole set of questions, and the displayed answers are an example of the ten responses given by the assistant for each case.

Seq nº	Question	Answer	Accuracy
1.	Can you talk in portuguese?	Configuração atualizada com sucesso. A partir de agora, responderei em português.	100%
"	Quais são as minhas férias em 2024?	Suas licenças em 2024 são: | Tipo de Licença | Duração | Data | |————————|———|————| | Vacation | 8 horas | 12/03/2024 | | Vacation | 8 horas | 13/03/2024 | | Vacation | 8 horas | 14/03/2024 | | Vacation | 8 horas | 15/03/2024 | | Vacation | 8 horas | 16/05/2024 | | Vacation | 8 horas | 14/05/2024 | | Vacation | 8 horas | 15/05/2024 | | Vacation | 8 horas | 21/05/2024 | | Vacation | 8 horas | 22/05/2024 | | Vacation | 8 horas | 23/05/2024 | | Vacation | 8 horas | 04/06/2024 | | Vacation | 8 horas | 05/06/2024 | | Vacation | 8 horas | 06/06/2024 | | Vacation | 8 horas | 17/09/2024 | | Vacation | 8 horas | 18/09/2024 | | Vacation | 8 horas | 19/09/2024	100%
"	E se juntares um bocado de humor às respostas?	Claro, Arthur! Vou tentar ser mais engraçado a partir de agora. Vamos lá, qual é a próxima piada?	100%
"	Por quais tarefas sou eu responsável? Faz um pequeno resumo por cliente	Arthur, meu caro, parece que você está com a agenda mais cheia que fila de banco em dia de pagamento! Vamos dar uma olhada nas suas tarefas por cliente ### Coca-cola - **Rejected:** 8 tarefas (Essas não foram com a cara do chefe) - **New:** 50 tarefas (Novinhas em folha, prontas para serem desbravadas) - **InProgress:** 22 tarefas (Aquelas que estão no forno) - **Done:** 7 tarefas (Missão cumprida!) - **ToApprove:** 4 tarefas (Esperando aquele "ok" final) - **Returned:** 1 tarefa (Voltou como boomerang) ### Southwest Airlines - **Rejected:** 1 tarefa (Essa não decolou) - **New:** 171 tarefas (Sim, você leu certo, 171!) - **InProgress:** 28 tarefas (Em pleno voo) - **Done:** 2 tarefas (Pousaram com sucesso) ### BBC...	100%
"	Tenho algumas da coca-cola?	Arthur, parece que você está com sede de informação! Sim, você tem algumas tarefas da Coca-Cola. Quer que eu repita o resumo ou tem algo específico em mente?	100%
"	Alguma foi rejeitada?	Sim, Arthur! Você tem 8 tarefas da Coca-Cola que foram rejeitadas. Parece que essas não foram com a cara do chefe!	100%
2.	How many jobs James was responsible?	James is responsible for 31 jobs.	100%
"	Any of those Jobs was for BBC?	James is responsible for 20 jobs for BBC, not 31.	100%
"	Does Paul had in the past any jobs responsable?	Paul was responsible for 260 jobs in the past.	100%
"	What has his latest?	Paul's latest job details are as follows: - **Job Number:** BBC0014ACCO001 - **Project Number:** BBC0014 - **Client:** BBC - **Department:** Account - **Stage:** InProgress - **Start Date:** 2023-11-13T14:25:45.01 - **End Date:** 2023-11-13T16:25:45.01	100%
"	Does he have any now?	Paul does not have any current jobs.	100%

Table 5.5: Testing assistant history.

Seq nº	Question	Answer	Accuracy
3.	After notification summary be executed, where notification preferences about: - Time Sheet (Hours to be approved) - Absences (Who goes on vacation) - Closed (Tasks Closed)	**Summary of Your Notifications:** | Category | Notifications | |————|————| | Time Sheet | 1 notification for hours to be approved | | Absences | 2 notifications for vacations | | Closed | 4 tasks closed |	100%
"	So, which persons had tasks closed?	The persons who had tasks closed are Abdul.Kalam, Marta.Brito, Helena.Nunes, and Rafael.Levy.	100%

Table 5.6: Testing assistant history (continuation).

Note: The sequence of questions was done by order and without cleaning the history. They are separated by a double line.

Lastly, interpreting the results from testing the historical context and the assistant's understanding (see table 5.6), we can observe that the assistant maintains context throughout the conversation, providing accurate responses to user questions. The assistant's ability to remember past interactions and adapt its responses accordingly is a key feature that enhances the user experience. For example, in the first sequence of questions, the last two questions can always be answered based on the results of the previous question. In sequence two we can use the context of the history to understand the current question and provide a correct answer. In the final sequence, we can see that it utilizes the response from the previous question to provide answers to new questions if the information is available.

5.1.1 Discussion

It is not easy to test an AI assistant, because it is a complex system that needs to understand the user's needs and at all times depends on the way the user asks the questions.

While testing our AI assistant revealed several key insights and areas for improvement. One of the primary challenges identified was maintaining context throughout conversations. The assistant demonstrated a good ability to understand and retain context, which is crucial for providing accurate and relevant responses. However, there were instances where the context was lost, leading to misunderstandings. This highlights the need for further refinement in the assistant's context management algorithms.

Error handling is also another critical aspect. The assistant's ability to manage misunderstandings and provide corrective feedback is essential to avoid user frustration. Our tests showed that while the assistant could handle most errors well, there were scenarios where it struggled to recover from misunderstandings. Enhancing the error recovery mechanisms will be a focus for future development.

In the end, this testing phase highlighted the strengths and areas for improvement in

our AI assistant. The assistant's ability to understand context, handle errors, and personalize interactions are significant advantages. However, refining context management, error recovery, and consistency in personalization will be essential for future development. The insights gained from this, were positive because we believe these approaches have huge potential, and will also guide the next steps in enhancing the assistant's performance and user experience.

5.2 Usability Testing

Usability testing is a type of testing performed to evaluate how easy and user-friendly a product or design is by observing real users as they interact with it. This is a perfect test for an AI assistant because it will show how effectively the AI assistant can support users in completing their tasks. By observing where the AI might struggle or excel, gathering feedback to improve its performance.

In practice, users were asked to perform specific tasks so that we could evaluate the effectiveness and accuracy of the assistant's responses in scenarios, in addition to understanding how users presented their questions. All questions and answers were saved for further analysis. Additionally, we aimed to find out about potential future functionalities and how the assistant can evolve.

This test involved the participation of users who were already familiar with the platform, exclusively asking them to tell us their role/persona. We recruited 18 users from consultancy and development backgrounds (personas) and asked each to complete a set of tasks and provide feedback through a structured questionnaire via Google Forms, which is displayed in Appendix B (see B). The data was collected with screenshots (the conversations with the assistant), ratings (from 1 to 5), and qualitative feedback through open-ended questions.

5.2.1 Results Analysis

This section provides a detailed analysis of the findings, highlighting the performance of Skillio in various scenarios and overall user satisfaction. Of the participants, 50% are developers, and the other half are the consulting staff. This distribution is important to have in account because users with a more technical profile may influence the way Skillio is used. Additionally, it may indicate that certain functionalities focused on task automation are more valued by a development persona, whereas functionalities related to data analysis may be more relevant for a consulting persona.

5.2.1.1 1. Vacation Verification

Scenario: "Imagine you are working on a project for a client. You need to use some resources and you recall someone you'd like to add to the project. However, you need to check if this person is on vacation. Ask Skillio to check this resource's vacation plans."

Regarding questions asked about checking whether someone is on vacation, users were able to obtain the correct information in most cases. Only one user got an incorrect answer when trying to ask the question of whether a person is on vacation during a Project/Job, as it is necessary for data analysis to search on two data sets, which is something not yet available. Or when they ask for vacations scheduled, and the assistant says it does not have vacations scheduled, but the user is on vacation but categorized as forced leaves, which explains the answer and how the user question influences the response. In this case, more detail in the response may solve the issue, making the user understand and refine their question.

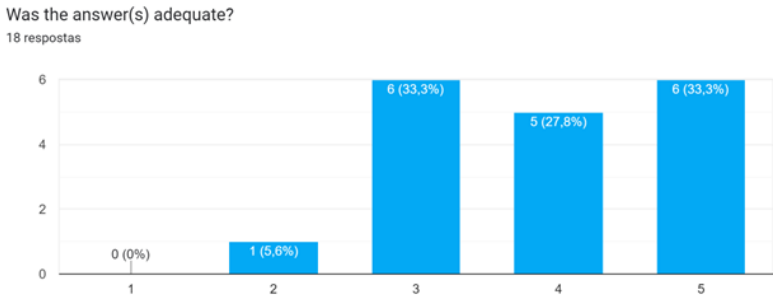


Figure 5.2: Answer adequacy for the vacation verification scenario.

Looking at figure 5.2 we can see the level of satisfaction that the users had with the answers. Additionally, we verified that 66.7% of the users found using the assistant more efficient than manually conducting the task through the traditional platforms.

Users found the Skillio assistant most efficient for checking vacation plans due to its speed and ease of use. Natural language was highlighted as a positive point, allowing for more intuitive interaction. Many appreciated the reduction in menu navigation, saving time. However, there were concerns about the accuracy of the responses, indicating the need for improvements. Familiarity with the traditional platform influenced preferences, with new users finding the assistant more useful. For quick checks, the wizard was considered more efficient, but for a complete overview, the traditional platform was still preferred. Realizing that one can try to put more details in the answers.

5.2.1.2 2. Configuration Exploration

Scenario: "Imagine you are setting up Skillio for the first time and want to change the language to Portuguese, enable the "do not disturb" mode for the summary of the notification, and ask it to be more funny and enthusiastic. After that, ask Skillio what it can do."

Regarding the questions about changing configurations, users were generally able to achieve the settings they were asked for. However, some users found the responses lacked

detail, particularly when confirming the changes. Providing more specific confirmations could enhance user understanding and satisfaction.

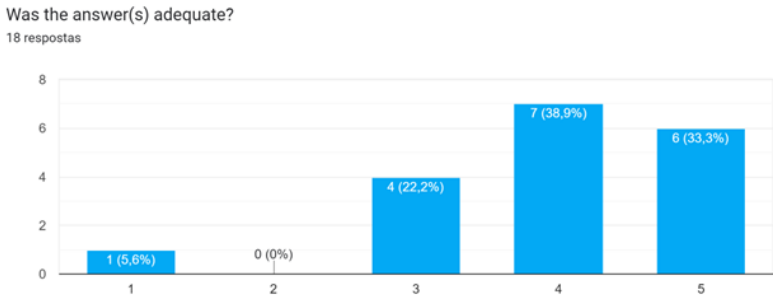


Figure 5.3: Answer adequacy for the configuration exploration scenario.

Analyzing figure 5.3, we can see what the users think about the answers. The one person that evaluate the answer with a score of 1, was because it was trying to configure to be in "do not disturb" mode only after 21 hours, and change settings like calendarization settings, but the assistant does not have this configuration yet. Overall, the results indicate that the answers provided by Skillio were well-received by users. This is also reflected in the fact that 66.7% of the users found using the assistant more efficient than manually adjusting settings through a traditional panel.

Users appreciated the natural language interaction, which made the process more intuitive and reduced the need for navigating through multiple menus. However, there were concerns about the detail of the responses, indicating that it needs improvement. For quick configuration changes, the assistant was considered more efficient, but for more complex settings, the traditional panel was still preferred.

5.2.1.3 3. Task Assignment

Scenario: "Imagine that you are assigning tasks to employees and believe that a particular employee is the best choice for the job. However, you are not sure if they have a lot of other work to do. Ask Skillio to check the tasks this employee is currently responsible for. Also, to see their success rate, you want to know if in the past they had a lot of tasks closed. And after, how many were rejected? If they have, ask for details. Afterward, try to do the same without relying on the task management system to compare the approaches."

This was a tricky task for the user. Some tried to filter with parameters not described by the assistant, like the department, or asking for the best person to give a certain task, which is not easy because the assistant does not have a way to know who is the best person for a task. However, when plausible questions were made, the assistant was able to answer correctly, sometimes producing a valuable answer with a summary and a percentage of tasks that were not finished on time or rejected. On the other hand, users reported similar questions and obtained different responses (some correct, others not). Additionally, the

questions they had to perform returned a lot of data, so in some cases, the assistant gave a huge list of tasks a user was responsible for one year instead of summarizing the data.

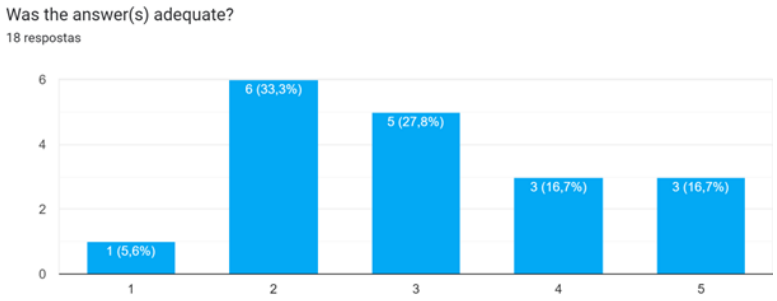


Figure 5.4: Answer adequacy for the task assignment scenario.

Most of the feedback on figure 5.4 indicated that the answers were not adequate, lacking simplicity and clarity, and requiring a more visual approach. However, users appreciated the calculations, such as the percentages of rejected tasks. From all the participants 61.1% of users found this method more efficient, while 38.9% did not. Users noted that it was quicker than using the UI lists to find the results but mentioned the need for easy-to-interpret answers, preferring visual representations like graphs over text.

While the assistant was useful for finding suitable resources, it often provided imprecise answers, making it difficult for users to ask the right questions. The lack of a workspace with grouped answers required manual checks. While natural language interaction was appreciated, other users preferred graphical data over text. Reliability and detailed information were concerns due to the lack or excess of details in the responses.

5.2.1.4 4. Deadline Checks

Scenario: "Imagine you're attending a team meeting where you usually check in on the status of jobs nearing their deadlines. This time, you want to approach it in a more cheerful way using Skillio, so tell him to talk happily. Then, ask Skillio for a detailed resume of the jobs closest to their end dates. Afterward, try to manage the task without Skillio to compare the different approaches."

The configuration to a more cheerful assistant was executed always with no problems. Regarding the questions asked about checking the status of jobs nearing their deadlines, users were getting the correct answer, the major problem seems to be the way the information is displayed. This is because in the UI markdown notation is written but it is not supported yet by the chat component itself, so the huge number of data is displayed in a single block of text, making it hard to read and understand.

Looking at figure 5.5, the tests show a balanced result, indicating that the answers provided by Skillio were liked by some people, but not by others. The only person who does not find the answer(s) adequate, on the other hand, agrees that is more efficient than

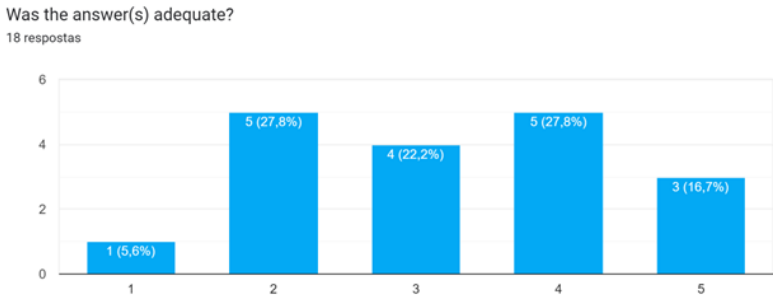


Figure 5.5: Answer adequacy for the deadline checks scenario.

manually conducting this finding on the platform. Making part of the majority 61.1% of users found this method more efficient for finding jobs closest to their end dates, while 38.9% did not. Users who liked it least pointed out difficulties in reading data and viewing a broader overview of tasks. The lack of features such as editing previous messages and opening tasks directly from the response was also highlighted as a limitation. Additionally, some users mentioned that the assistant did not correctly interpret the data or did not take the pronoun "my" into account, resulting in incorrect answers.

5.2.1.5 5. Previous Projects

Scenario: "Imagine you're managing a major project and remember that one of your employees worked on a similar project in the past year. You know it was one of three specific employees. So, ask Skillio to provide a list of each team member's tasks from that year with their details. Then, try tackling that task without relying on the assistant to compare approaches."

When the assistant was asked to retrieve the people involved in a specific project, it either stated that it couldn't provide the information or that the project didn't exist. However, when the user requested tasks for specific individuals, the assistant returned a table with past jobs.

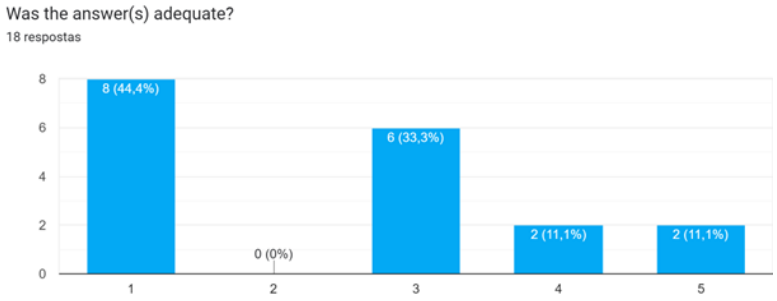


Figure 5.6: Answer adequacy for the previous projects scenario.

The results in figure 5.6 show that the users or rather like the answer or do not like it at all. Based on the user feedback regarding whether the answer provided information to help find the project name, it is clear that 61.1% of users faced challenges. Several users mentioned that they were unable to find the information they asked for, while others noted that the information provided was either incomplete or too complex to navigate. Some users preferred a more organized and visually appealing format, such as a list with colors, to better understand the information. Additionally, there were comments about the assistant's performance being less effective compared to other search tools, and difficulties in refining searches or understanding the responses. Overall, the feedback indicates a need for improvements in clarity, presentation, and search refinement capabilities.

This test was unique because it helped us understand what the assistant can do when it doesn't have access to specific data, such as projects. The assistant only knows about jobs that are related to certain projects. In some cases, the assistant provides information about the jobs that are related to the projects, even if it's a large amount. This shows the assistant capability to search and filter data.

5.2.1.6 Open-end Evaluation Questions

1. What kind of improvements would you like to see made from what has already been added?

Regarding improvements to what is implemented, users suggested:

- Speech-to-text functionality and smart typing suggestions.
- Better navigation with links to where the data is.
- More user-friendly interface with graphs, interactive lists (capable of filtering), images, and videos when it makes sense.
- Complete review of the user interface to improve readability and formatting, as well as the Scroll in chat.
- Improve the interpretation of user questions and faster responses.

Many of these suggestions are already on the roadmap for future updates.

2. Is there any other functionality/automation you would like to see added?

In addition to other functionalities, users have proposed adding functionality/automation such as:

- History of all conversations, edit the previous message, cancel message, clear chat.
- Inline navigation to documents, real grid display instead of text blocks (markdown notation is not supported by the interface).
- Possible to do tasks such as add timesheets to tasks, book leaves, add users to teams in documents, and create tasks and documents using the assistant. Even create content from similar examples.

- Basic help for beginners, email writing suggestions, translation of different languages.
- Event-based reminders and notifications, schedule digest with specific metrics.
- Alerts for projects nearing deadlines, budget limits, clients with many delayed tickets, and overloaded team members.
- Check profitability, burn rates, and key metrics (e.g., most/least profitable projects, clients consuming the most hours, projects at risk of overburning)
- Make bulk changes, such as closing jobs without activity for over a year
- Better interpretation of settings and default language
- Use images to aid user understanding

If we notice some of the suggestions are related to the interface itself, and not to the assistant, which is a good point to improve the user experience and to make it more user-friendly. Others meet the features we thought of, but have not yet been implemented (see 3.1).

3. In your day-to-day, would you like to use Skillio to handle several tasks (not only the ones available in the prototype)?

When asked the participants if they would like to use Skillio for various tasks in their day-to-day lives, most users indicated that they would like to use it for tasks beyond the prototype's capabilities. This shows a strong interest and potential of the assistant.to use it for tasks beyond the prototype's capabilities. This shows a strong interest and potential of the assistant.

4. With what the assistant does now, is there are any current issues that you can resolve using the assistant?

The last question tries to understand whether the assistant's current features are sufficient to solve specific problems that users face. Overall, the majority of users indicated yes. Things like changing the configurations on the platform, producing metrics (calculations/percentages) and summaries of data, or even checking the status of tasks and projects on the fly.

5.2.2 Discussion

The majority of users appreciated the accuracy and helpfulness of Skillio's responses, as well as the time savings and convenience it offered. The feedback also provided valuable insights into potential areas for improvement and future enhancements.

It is clear that the assistant has significant potential to enhance productivity and user experience through its intuitive interface and robust functionality. However, there are several areas for improvement, such as enhancing specific features based on user feedback. However, there is always a "problem" that will be difficult to explore, which is the way the

questions are asked. Sometimes the user does not get the expected answer, but the way the question is asked influences the response.

Also, the efficiency depends on the user's familiarity with the platform, we have noticed that users who seem more at ease with the platform, tend to prefer the manual finding (also more related to the role of consultancy), while the more technical ones prefer the assistant. This is a good point to have in mind when developing new features.

Users have consistently reported time savings and the convenience offered by the assistant, also reported in scenario 5 (see 5.2.1.5) where the responses did not meet users' expectations. In open-end evaluation questions (see 5.2.1.6), we understand that the majority found the assistant to be more efficient than manually conducting the tasks.

One main point to improve is the way the information is displayed, the platform UI is not able to display markdown notation in the chat, which makes the information hard to read and understand. Also, the assistant ability to refine searches is an aspect that needs attention. However, we proved that it is possible to change and search data in the platform using Skillio, whether looking for vacation or jobs or adjusting configurations.

In summary, users suggested enhancements such as speech-to-text functionality, better navigation with links, a more user-friendly interface with interactive elements, and improved readability and formatting. Additionally, they expressed a desire for new functionalities like inline document navigation, task automation, and event-based reminders. Most users showed a strong interest in using Skillio for a broader range of tasks beyond the prototype's current capabilities, indicating significant potential for the assistant. Overall, while the current features are appreciated and solve many user problems, there is a clear demand for further development to enhance usability and expand functionality.

6

Conclusions and Future Work

The primary objective of this dissertation was to leverage a Large Language Model (LLM) to automate processes within the Skills Workflow platform. This approach aimed to create an AI assistant with unique features that streamline the user's workflow, reducing the learning curve for new interfaces. The AI assistant was designed with the goal of improving user interaction, operational efficiency, and overall productivity.

The research demonstrated that integrating LLMs into the Skills Workflow platform significantly improves user experience by automating routine tasks and providing intuitive assistance. Key findings include:

- Successful implementation of an AI assistant prototype that can adjust settings, update notification preferences, analyze application data and make changes to system data.
- Useful architecture structure that allows the assistant to interact with the LLM service from Azure OpenAI, ensuring efficient processing of user requests.
- The assistant's ability to handle complex queries, maintain context, and provide accurate, relevant responses by following rules and guidelines in predefined Prompts.
- Positive user feedback indicating increased efficiency and satisfaction with the way the processes were automated.

By leveraging a REST API with a service that utilizes the Semantic Kernel SDK to manage LLMs, and identifying an accurate LLMs with a cost-effective solution for handling a high volume of daily requests proportional to application usage, we have achieved a viable solution for the Skills Workflow environment. This approach can be similarly applied to other applications.

The findings of this study have significant implications for user interface design and AI integration. The study demonstrates that large language models can effectively reduce the learning curve and automate processes in the field. This contributes greatly to our understanding of AI-assisted user interfaces. The results suggest that similar approaches can be applied to other platforms, transforming how users interact with software applications and improving productivity across various domains, such as application data findings or

updates. While it is challenging to extract the correct format from LLMs for tasks such as code reading, our innovative approach still shows to be optimistic. Our use of LLMs to automate tasks and enhance user experience indicates that we have created an assistant that not only automates routine tasks but also improves user work efficiency.

We have discovered that the LLM can generate code to perform actions that may already have a format, such as JSON templates that can be used to modify the system. Additionally, it can search through structured data by generating queries instead of using approaches like RAG, which may be more costly and have limitations, such as the amount of data that can be processed. A sequence of well-defined Prompts can be a powerful alternative to achieve specific goals. These Prompts just need to be well structured with a clear goal and a clear way to achieve it. Since the LLM can generate anything, it can produce pieces of code or data that can change the system to the desired state. This is a powerful tool that can be applied in various scenarios.

The testing of the assistant with potential questions and real users has provided valuable insight into its usability and effectiveness. This process has also assisted in refining the assistant's features, particularly the Prompts sent to the LLM, which are a crucial aspect of the overall assistant functionality. As a result, the assistant is constantly evolving, adapting to the user's needs and providing more accurate and relevant responses.

We can instruct the assistant to analyze system data and modify the system itself, by using only the power of the large language model (LLM). This allows us to adapt and expand the assistant's functionalities to cover more areas of the user's workflow. The logic of the solutions demonstrates that we are capable of generating functional code, adapted to our specific needs, with certain filters or certain outputs. Our solution is scalable and can be used for several other applications.

In conclusion, this research accentuates the transformative potential of integrating LLMs into user interfaces to enhance usability and efficiency. By automating routine tasks and providing intuitive attendance, AI assistants like the one developed in this study can improve user experience and productivity. As technology continues to evolve, the insights gained from this research are invaluable and make us believe that it is possible to fully leverage large language models to automate processes. This approach not only enhances the Skills Workflow environment but also holds promise for a wide range of applications, paving the way for more efficient and User-friendly software solutions.

6.1 Future Work

Future research should focus on enhancing the LLM's ability to interpret and generate code more accurately, developing methods to handle larger and more complex Prompts, and further exploring the integration of Symbolic AI with LLMs to improve the assistant's efficiency and accuracy.

The current implementation can be still refined from what we have discovered like retrieving only the data that meets certain criteria, rather than all the available data. By

filtering data, you reduce the amount of data you need to process, which makes it easier and faster to handle. For example, in vacation data, if it is only needed data from 2020, we can filter out data from other years, keeping a smaller set of data for the JavaScript query to process. Furthermore, it should be possible to question two different data sets at the same Prompt. For example, knowing if someone with tasks this month has also vacations, requires data from Jobs and Vacations data sets. Also, it is important to expand the search ability to several data in the platform, not only the Jobs and Vacations data sets.

Additionally, as we expected, some user tests (see 5.2) mentioned that voice input and outputs help increase the efficiency of the assistant, which in a mobile application may be very handy. So, future improvements may incorporate models like text-to-speech, speech-to-text, and computer vision to enhance the assistant's ability to interact more naturally with users. It can also be useful to have a graphical representation of the data, instead of using only markdown syntax. Or when the assistant searches for something in the platform, it could navigate to the place where the information is, instead of just providing the information. Another improvement that was suggested was for the assistant's answers to have more details, especially when the question is to analyze data on the platform.

In addition, the assistant's functionalities should be expanded to cover more areas of the user's workflow and conduct extensive user testing to get feedback for continuous improvement. Some of the functionalities were already explored in section 3.1 and are mentioned below, maintaining the core logic of the assistant with predefined Prompts for LLM calls to generate specific outcomes.

- **Dashboard Creation Use Case:** Automate the creation of a workspace through its description in natural language. The LLM would produce the JSON code representation of the workspace, which the user would then confirm before it is implemented.
- **Alert Creation Use Case:** Allow users to define notifications related to system data. The LLM would generate the necessary code to create these alerts, which would then be confirmed by the user and saved in the system.
- **Project Task Creation Use Case:** Enable users to generate tasks for a project by providing relevant information in natural language. The LLM would generate the parameters needed for the task creation endpoint, streamlining the process and saving users time.
- **Project Creation Use Case:** Create a new project based on user descriptions. The LLM would generate the JSON code for the new project, which the user would then confirm. This approach would simplify the project creation process and ensure that necessary information is included.
- **Allocation of People Use Case:** Manage task distribution among team members. The LLM would generate the necessary code to adjust task allocations, making it easier for users to manage workloads.

Other things that may be improved are related to error handling. The assistant's ability to manage misunderstandings and provide corrective feedback is essential to avoid user

frustration. Because if an error occurs, the user is informed of the error, with a simple message, but for maintainability, the error should be more detailed and be logged for future analysis. Another way to improve error handling is to instruct the assistant to provide suggestions for correcting the error. So, enhancing the error recovery mechanisms will be a focus for future development.

Bibliography

[1] *AgentHub*. Accessed February 9, 2024. URL: https://www.agenthub.dev (cit. on p. 26).

[2] *Airtable Cobuilder*. Accessed Jul 25, 2024. URL: https://airtable.com/cobuilder (cit. on pp. 24, 25).

[3] *Airtable's new Cobuilder unlocks instant no-code app creation*. Accessed Jul 25, 2024. 2024-07. URL: https://blog.airtable.com/airtable-cobuilder-launch/ (cit. on p. 25).

[4] S. N. Akter et al. "An In-depth Look at Gemini's Language Abilities". In: (2023-12). URL: http://arxiv.org/abs/2312.11444 (cit. on p. 17).

[5] *Announcing Llama 3.1 405B, 70B, and 8B models from Meta in Amazon Bedrock*. Accessed Aug 3, 2024. 2024-07. URL: https://aws.amazon.com/blogs/aws/announcing-llama-3-1-405b-70b-and-8b-models-from-meta-in-amazon-bedrock/ (cit. on p. 13).

[6] *Anthropic introduces Claude 3, a new trio of multimodal models*. Accessed Aug 3, 2024. 2024-05. URL: https://www.deeplearning.ai/the-batch/anthropic-introduces-claude-3-a-new-trio-of-multimodal-models/ (cit. on p. 14).

[7] *Anthropic says it won't use your private data to train its AI*. Accessed Aug 3, 2024. 2024-01. URL: https://decrypt.co/211846/anthropic-says-it-wont-use-your-private-data-to-train-its-ai (cit. on p. 14).

[8] *Anthropic's approach to GDPR*. Accessed Aug 3, 2024. 2024-08. URL: https://support.anthropic.com/en/articles/7996881-what-is-your-approach-to-gdpr-or-related-issues (cit. on p. 14).

[9] *Artificial Analysis: The Future of AI*. Accessed Aug 19, 2024. URL: https://deepmind.google/technologies/gemini/project-astra/ (cit. on p. 24).

[10] *Azure Container Apps*. Accessed Sep 24, 2024. URL: https://azure.microsoft.com/en-us/products/container-apps (cit. on p. 40).

[11] B. Banjara. *LLMs in Conversational AI: Building Smarter Chatbots Assistants*. Accessed January 9, 2024. 2023-07. URL: https://www.analyticsvidhya.com/blog/2023/07/llms-in-conversational-ai/ (cit. on p. 9).

[12] M. Bolanos. *Migrating from the Sequential and Stepwise planners to the new Handlebars and Stepwise planner*. Accessed January 15, 2024. 2023-12. URL: https://devblogs.microsoft.com/semantic-kernel/migrating-from-the-sequential-and-stepwise-planners-to-the-new-handlebars-and-stepwise-planner/ (cit. on p. 23).

[13] E. Boyd. *ChatGPT is now available in Azure OpenAI Service*. Accessed February 23, 2024. 2023-03. URL: https://azure.microsoft.com/en-us/blog/chatgpt-is-now-available-in-azure-openai-service/ (cit. on p. 21).

[14] *Bring Your Own Data to Azure OpenAI*. Accessed March 4, 2024. 2023. URL: https://blazorhelpwebsite.com/ViewBlogPost/8067 (cit. on p. 12).

[15] S. Caixa Geral de Depósitos. *Caixadirecta*. Accessed December 6, 2023. URL: https://www.cgd.pt/Particulares/Contas/Caixadirecta/Pages/Assistente-Digital-App-Caixadirecta.aspx (cit. on p. 26).

[16] *Claude's Constitution*. Accessed Aug 3, 2024. 2024-05. URL: https://www.anthropic.com/news/claudes-constitution (cit. on p. 14).

[17] M. Corporation. *Creating AI agents*. Accessed January 4, 2024. URL: https://learn.microsoft.com/en-us/semantic-kernel/overview/ (cit. on p. 22).

[18] M. Corporation. *Creating AI agents*. Accessed January 4, 2024. URL: https://learn.microsoft.com/en-us/semantic-kernel/agents/ (cit. on p. 22).

[19] M. Corporation. *Creating AI agents*. Accessed January 5, 2024. URL: https://jordanbeandev.com/how-to-build-your-own-chatbot-using-c-semantic-kernel-azure-openai-part-1/ (cit. on p. 23).

[20] M. Corporation. *Descrição geral do Microsoft Copilot para Microsoft 365*. Accessed January 9, 2024. 2023-11. URL: https://learn.microsoft.com/pt-pt/microsoft-365-copilot/microsoft-365-copilot-overview (cit. on p. 26).

[21] M. Corporation. *Export plugins written for Semantic Kernel as an OpenAI plugin*. Accessed January 5, 2024. URL: https://learn.microsoft.com/en-us/semantic-kernel/agents/plugins/openai-plugins (cit. on p. 22).

[22] M. Corporation. *What is the Bot Framework SDK?* Accessed January 24, 2024. 2022-11. URL: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0 (cit. on p. 21).

[23] M. Corporation. *What is the Bot Framework SDK?* Accessed January 24, 2024. 2022-11. URL: https://www.trustradius.com/compare-products/azure-bot-service-vs-google-cloud-dialogflow#community-pulse (cit. on p. 21).

[24] M. Corporation. *What will you do with Copilot with Bing*. Accessed on February 6, 2024. URL: https://www.microsoft.com/en-us/bing?ep=0&form=MA13LV&es=31 (cit. on p. 1).

[25] T. L. Daniel Adiwardana. *Towards a Conversational Agent that Can Chat About. . . Anything*. Accessed February 23, 2024. 2020-01. URL: https://blog.research.google/2020/01/towards-conversational-agent-that-can.html (cit. on p. 21).

[26] *Fine-Tuning Vs RAG in Generative AI*. Accessed May 7, 2024. 2023-09. URL: https://plainenglish.io/community/fine-tuning-vs-rag-in-generative-ai-4a01a4#architecture (cit. on p. 48).

[27] *Gemini Models*. Accessed Aug 3, 2024. URL: https://deepmind.google/technologies/gemini/ (cit. on p. 13).

[28] *Gemma 2 is now available to researchers and developers*. Accessed Aug 3, 2024. 2024-06. URL: https://blog.google/technology/developers/google-gemma-2/ (cit. on p. 13).

[29] Google. *You May be Fine-Tuning Google Bard (Without You Knowing it)*. Accessed December 20, 2023. URL: https://support.google.com/bard/answer/13594961?visit_id=638386644607786805 (cit. on p. 13).

[30] time haaify. *Open Source vs Proprietary LMS: How Do I Choose?* Accessed January 20, 2024. 2019-10. URL: https://www.lambdasolutions.net/blog/open-source-vs-proprietary-lms-how-do-i-choose (cit. on p. 11).

[31] time haaify. *Qual é a diferença entre bot, chatbot e assistente virtual*. Accessed January 20, 2024. 2022-05. URL: https://haaify.com/blog/diferenca-bot-chatbot-assistente-virtual/ (cit. on p. 8).

[32] B. Heater. *With Brain.ai, generative AI is the OS*. Accessed March 1, 2024. 2024-02. URL: https://techcrunch.com/2024/02/29/with-brain-ai-generative-ai-is-the-os/?guccounter=1 (cit. on p. 25).

[33] *Hello GPT-4o*. Accessed Jul 31, 2024. 2024-05. URL: https://openai.com/index/hello-gpt-4o/ (cit. on p. 12).

[34] *How Meta trains large language models at scale*. Accessed Aug 3, 2024. 2024-06. URL: https://engineering.fb.com/2024/06/12/data-infrastructure/training-large-language-models-at-scale-meta/ (cit. on p. 14).

[35] *How to Use Claude with Amazon Bedrock - Step by Step Guide*. Accessed Aug 3, 2024. 2024-04. URL: https://cheatsheet.md/claude/claude-amazon-bedrock.en (cit. on p. 14).

[36] M. Kapronczay. *A Beginner's Guide to Language Models*. Accessed January 19, 2024. 2022-12. URL: https://builtin.com/data-science/beginners-guide-language-models (cit. on p. 9).

[37] P. Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: (2020-05). URL: http://arxiv.org/abs/2005.11401 (cit. on p. 48).

[38] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[39] *Meta claims 'world's largest' open AI model with Llama 3.1 405B debut*. Accessed Aug 3, 2024. 2024-07. URL: https://www.theregister.com/2024/07/23/meta_llama31_405b/ (cit. on pp. 13, 14).

[40] *Meta opens access to its large language model for AI researchers*. Accessed Aug 3, 2024. 2022-05. URL: https://www.siliconrepublic.com/machines/meta-ai-large-language-model (cit. on p. 14).

[41] Microsoft. *Azure OpenAI Service models*. Accessed December 20, 2023. URL: https://learn.microsoft.com/en-US/azure/ai-services/openai/concepts/models (cit. on p. 12).

[42] *Mistral AI selects Google Cloud infrastructure to make generative AI more open and accessible*. Accessed Aug 4, 2024. 2023-12. URL: https://www.prnewswire.com/news-releases/mistral-ai-selects-google-cloud-infrastructure-to-make-generative-ai-more-open-and-accessible-302014018.html (cit. on p. 15).

[43] *Mistral announces large 2 flagship LLM with 123 billion parameters*. Accessed Aug 4, 2024. 2024-07. URL: https://www.neowin.net/news/mistral-announces-large-2-flagship-llm-with-123-billion-parameters/ (cit. on p. 15).

[44] *Mistral Large*. Accessed Aug 34, 2024. 2024-07. URL: https://mistral.ai/news/mistral-large-2407/ (cit. on pp. 14, 15).

[45] H. Naveed et al. "A Comprehensive Overview of Large Language Models". In: (2023-07). URL: http://arxiv.org/abs/2307.06435 (cit. on p. 9).

[46] I. NGUYEN. *When and How to Train Your Own Language Model*. Accessed January 19, 2024. 2022-08. URL: https://www.deepset.ai/blog/when-and-how-to-train-a-language-model (cit. on p. 9).

[47] T. Omoyeni. *Introducing Chatbots and Large Language Models*. Accessed January 9, 2024. 2023-12. URL: https://www.sitepoint.com/introducing-chatbots-and-large-language-models-llms/ (cit. on p. 9).

[48] OpenAI. *Common use cases*. Accessed December 6, 2023. URL: https://platform.openai.com/docs/guides/fine-tuning/common-use-cases (cit. on p. 10).

[49] OpenAI. *Customer stories*. Accessed December 17, 2023. URL: https://openai.com/customer-stories (cit. on p. 25).

[50] OpenAI. *Fine Tuning*. Accessed December 6, 2023. URL: https://platform.openai.com/docs/guides/fine-tuning (cit. on p. 10).

[51] OpenAI. *How we use your data*. Accessed February 23, 2024. URL: https://platform.openai.com/docs/models/how-we-use-your-data (cit. on p. 12).

[52] OpenAI. *Moderation*. Accessed December 20, 2023. URL: https://platform.openai.com/docs/guides/moderation/overview (cit. on p. 12).

[53] OpenAI. *Prompt engineering*. Accessed December 23, 2023. URL: https://platform.openai.com/docs/guides/prompt-engineering (cit. on p. 11).

[54] OpenAI. *Stripe*. Accessed December 6, 2023. 2023-03. URL: https://openai.com/customer-stories/stripe (cit. on p. 25).

[55] OpenAI. *Viable*. Accessed December 17, 2023. 2023-03. URL: https://openai.com/customer-stories/be-my-eyes (cit. on p. 26).

[56] *OpenAI Presents GPT-3, a 175 Billion Parameters Language Model*. Accessed Aug 3, 2024. 2020-07. URL: https://developer.nvidia.com/blog/openai-presents-gpt-3-a-175-billion-parameters-language-model/ (cit. on p. 11).

[57] *Retrieval Augmented Generation (RAG) in Azure AI Search*. Accessed May 7, 2024. 2024-04. URL: https://learn.microsoft.com/en-us/azure/search/retrieval-augmented-generation-overview (cit. on p. 48).

[58] *Retrieval augmented generation and indexes*. Accessed May 7, 2024. 2024-02. URL: https://learn.microsoft.com/en-us/azure/ai-studio/concepts/retrieval-augmented-generation (cit. on p. 48).

[59] A. Sheth, K. Roy, and M. Gaur. "Neurosymbolic AI – Why, What, and How". In: (2023-05). URL: http://arxiv.org/abs/2305.00813 (cit. on p. 37).

[60] N. Shukla. *LLMs vs. Traditional Language Models: A Comparative Analysis*. Accessed January 10, 2024. 2023-09. URL: https://www.appypie.com/blog/llms-vs-traditional-language-models (cit. on p. 9).

[61] Statistopedia. *You May be Fine-Tuning Google Bard (Without You Knowing it)*. Accessed November 7, 2023. 2023-09. URL: https://statistopedia.com/google-bard/you-may-be-fine-tuning-google-bard-without-you-knowing-it/ (cit. on p. 13).

[62] D. H. Sundar Pichai. *Introducing Gemini: our largest and most capable AI model*. Accessed Fevereiro 7, 2023. 2023-12. URL: https://blog.google/technology/ai/google-gemini-ai/#sundar-note (cit. on p. 13).

[63] G. TARRAF. *Everything you need to evaluate open-source (vs. closed-source) LLMs*. Accessed January 20, 2024. 2023-12. URL: https://atelier.net/insights/evaluating-open-source-large-language-models (cit. on p. 11).

[64] G. Team. *Gemini: A Family of Highly Capable Multimodal Models*. 2023 (cit. on p. 17).

[65] V. S. Team and M. Murgia. *Generative AI exists because of the transformer*. Accessed November 7, 2023. 2023-09. URL: https://ig.ft.com/generative-ai/ (cit. on p. 9).

[66] H. Touvron et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models". In: (2023-07). URL: http://arxiv.org/abs/2307.09288 (cit. on p. 14).

[67] *Transform your business with frontier AI*. Accessed Aug 4, 2024. URL: https://mistral.ai/business/ (cit. on p. 15).

[68] *typescript-instruct*. Accessed December 23, 2023. URL: https://huggingface.co/datasets/bleugreen/typescript-instruct (cit. on p. 10).

[69] *Using Airtable Cobuilder*. Accessed Jul 25, 2024. 2024-07. URL: https://support.airtable.com/docs/using-airtable-cobuilder (cit. on p. 24).

[70] *What is Gemini Live?* Accessed Aug 19, 2024. 2024-06. URL: https://www.wired.com/story/what-is-gemini-live/ (cit. on p. 24).

[71] *What's the difference between GPT-3.5, 4, 4 Turbo, 4o? OpenAI LLMs explained*. Accessed Jul 31, 2024. 2024-05. URL: https://www.windowscentral.com/software-apps/windows-11/whats-the-difference-between-gpt-35-4-4-turbo-4o (cit. on p. 12).

[72] Wikipedia. *Language model*. Accessed January 19, 2024. URL: https://en.wikipedia.org/wiki/Language_model (cit. on p. 9).

[73] Wikipedia. *Office Assistant*. Accessed January 17, 2024. URL: https://en.wikipedia.org/wiki/Office_Assistant (cit. on p. 1).

[74] S. Workflow. Accessed January 9, 2024. URL: https://www.skillsworkflow.com (cit. on p. 6).

[75] B. Yuchen Lin et al. "WildBench: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild". In: *arXiv e-prints* (2024), arXiv–2406 (cit. on p. 16).

A

Appendix A

In Appendix A, there is a table that presents questions and answers for each approach used in the experiments. This table was utilized to compare RAG and a Query approach, which were divided into different categories. The study of RAG involved two different approaches, one using pure JSON format data to feed the LLM, and the other using the same data but in natural language text. The query approach was tested with two different models to compare the code quality produced by them. The table includes the question number, the question itself, the approach used, the answer given by the assistant, and the number of documents used in the RAG approach, the number of input tokens, the number of output tokens, and the response time.

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
1.	**In 2019, in what months did Bruno Moscão schedule vacations?**	RAG w/ JSON grounding	Bruno Moscão scheduled vacations in February 2019.	5	3007	19	1.862s
	"	RAG w/ NL grounding	In 2019, Bruno Moscão scheduled vacations in February. days.	5	876	20	3.376s
	"	Query data w/ sk (gpt-4)	In 2019, Bruno Moscão scheduled vacations in the following months: February.	-	686	264	39.253s
	"	Query data w/ sk (gpt-3.5)	In 2019, Bruno Moscão scheduled vacations in the following months: February.	-	703	506	28.842s
2.	**Does Bruno Moscão have any scheduled vacations?**	RAG w/ JSON grounding	Prompt is too long.	-	>4096	-	-
	"	RAG w/ NL grounding	Yes, Bruno Moscão has scheduled vacations in 2019, 2023 and 2024.	7	1206	28	1.529s
	"	Query data w/ sk (gpt-4)	Bruno Moscão has 6 days of scheduled vacations.	-	674	154	34.542s
	"	Query data w/ sk (gpt-3.5)	Yes, Bruno Moscão has scheduled vacations.	-	691	90	23.312s

Table A.1: Experiments Comparing RAG and Query approaches.

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
3.	**I want to know about the vacations from Bruno Moscão**	RAG w/ JSON grounding	Prompt is too long.	-	>4096	-	-
	"	RAG w/ NL grounding	Bruno Moscão has the following vacation information: - In 2021, he has 8.0 vacation days left. - In 2020, he has 9.0 vacation days left. - In 2019, he has 13.0 vacation days left. - In 2023, he has 2.5 vacation days left. - In 2022, he has 7.0 vacation days left. - In 2024, he has 22.0 vacation days left. - In 2024, he has 1.0 forced leave days left.	7	1207	130	2.557s
	"	Query data w/ sk (gpt-4)	Bruno Moscão from Skills Workflow, Skills division, Development department, as a CTO, has a leave type of Forced Leave for the year 2024. His leave expires on 2024-12-31T00:00:00. He is entitled to 3 days of vacation, out of which he has scheduled 2 days. He still has 1 days left to schedule. His vacation schedule is as follows: January - 0 days, February - 0 days, March - 0 days, April - 0 days, May - 0 days, June - 0 days, July - 0 days, August - 0 days, September - 0 days, October - 0 days, November - 0 days, December - 2 days. Bruno Moscão from Skills Workflow, Skills division, Development department, as a CTO, has a leave type of Vacation for the year 2019. His leave expire...	-	683	326	48.352s
	"	Query data w/ sk (gpt-3.5)	Bruno Moscão has taken vacations in the following years: 2019, 2020, 2021, 2022, 2023, 2024	-	698	126	24.384s
4.	**What is the total of days Bruno Moscão still was left on vacation in 2023?**	RAG w/ JSON grounding	Bruno Moscão still has 2.5 days left on vacation in 2023.	6	3573	26	1.423s
	"	RAG w/ NL grounding	Bruno Moscão still has 2.5 vacation days left in 2023. left.	6	1050	25	1.342s
	"	Query data w/ sk (gpt-4)	Bruno Moscão still has 2.5 days of vacation left in 2023.	-	695	162	37.854s
	"	Query data w/ sk (gpt-3.5)	Bruno Moscão has 2.5 days of vacation left in 2023.	-	713	119	23.574s
5.	**Who has vacations in 2024?**	RAG w/ JSON grounding	Sorry, I don't know the answer.	0	41	15	1.840s
	"	RAG w/ NL grounding	Sorry, I don't know the answer.	0	33	15	1.297s
	"	Query data w/ sk (gpt-4)	In 2024, the following people have vacations scheduled: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Telmo Carvalho, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Marina Mello.	-	671	124	30.692s
	"	Query data w/ sk (gpt-3.5)	The following employees have vacations in 2024: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Telmo Carvalho, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Marina Mello	-	689	101	23.760s

Table A.2: Experiments Comparing RAG and Query approaches (continuation).

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
6.	**Who has scheduled vacations in 2024?**	RAG w/ JSON grounding	Sorry, I don't know the answer.	0	42	15	2.019s
	"	RAG w/ NL grounding	Sorry, I don't have access to that information.	0	34	15	1.321s
	"	Query data w/ sk (gpt-4)	The following people have scheduled vacations in 2024: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Telmo Carvalho, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Marina Mello.	-	673	101	32.123s
	"	Query data w/ sk (gpt-3.5)	The following employees have scheduled vacations in 2024: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Telmo Carvalho, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Marina Mello	-	691	95	24.103s
7.	**What has the vacations on the COO in 2023?**	RAG w/ JSON grounding	The COO has 22 days of vacation entitlement for the year 2023.	4	2309	24	1.974s
	"	RAG w/ NL grounding	In 2023, the COO in the Operations department has the right to 22.0 vacation days. No vacation days were scheduled, so they still have 22.0 days left. The validity of their leaves extends up to 2024-06-30T00:00:00.	4	707	68	1.896s
	"	Query data w/ sk (gpt-4)	João Paulo Duque Vieira from Skills Workflow, Skills division, Operations department has scheduled 0 days of vacation in 2023. The leave type is Vacation. The total vacation entitlement is 22 days. There are still 22 days available to schedule. The vacation schedule is as follows: January - 0 days, February - 0 days, March - 0 days, April - 0 days, May - 0 days, June - 0 days, July - 0 days, August - 0 days, September - 0 days, October - 0 days, November - 0 days, December - 0 days.	-	683	297	43.906s
	"	Query data w/ sk (gpt-3.5)	The vacations on the COO in 2023 are: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Antonio Barata, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Elizabete Ferreira, Michelle Ribeiro, Marina Mello. (WRONG)	-	699	98	23.985s
8.	**How many vacations does Jose still can take in 2023?**	RAG w/ JSON grounding	Jose still can take 13 vacations in 2023. (WRONG)	3	1712	18	1.690s
	"	RAG w/ NL grounding	Jose still can take 13.5 vacations days in 2023. (WRONG)	3	545	21	1.651s
	"	Query data w/ sk (gpt-4)	Jose can still take 0 days of vacation in 2023.	-	681	97	25.678s
	"	Query data w/ sk (gpt-3.5)	Jose can still take 29 vacations in 2023. (WRONG)	-	691	344	24.735s

Table A.3: Experiments Comparing RAG and Query approaches (continuation).

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
9.	**Does Rodrigo already have vacations scheduled for 2024?**	RAG w/ JSON grounding	Prompt is too long.	-	>4096	-	-
	"	RAG w/ NL grounding	Yes, Rodrigo Alves has 15.0 days of vacation scheduled for 2024, and Rodrigo Mensurado has 0.0 days of vacation scheduled for 2024.	6	1035	44	1.530s
	"	Query data w/ sk (gpt-4)	Yes, Rodrigo has 15 days of vacation scheduled for 2024. His leave will expire on 2025-06-30T00:00:00. He still has 7 days available to schedule.	-	681	152	33.935s
	"	Query data w/ sk (gpt-3.5)	Yes, Rodrigo has vacations scheduled for 2024.	-	689	123	22.560s
10.	**When does the CFO from Finance have their time to vacation expired?**	RAG w/ JSON grounding	The CFO from Finance has their vacation time expired on June 30, 2025.	5	2823	24	2.776s
	"	RAG w/ NL grounding	The CFO from Finance has their time to vacation expired on 2025-06-30.	5	876	25	1.383s
	"	Query data w/ sk (gpt-4)	The CFO from Finance has their vacation time expired on multiple dates: 2024-12-31T00:00:00, 2024-06-30T00:00:00, 2025-06-30T00:00:00.	-	681	168	34.029s
	"	Query data w/ sk (gpt-3.5)	The CFO from Finance has their time to vacation expired on 2024-12-31T00:00:00.	-	691	73	22.167s
11.	**How many vacation days in total does João Paulo Vieira have?**	RAG w/ JSON grounding	João Paulo Vieira has a total of 22 days of vacation in 2023 and 22 days of vacation in 2024.	3	1725	35	2.086s
	"	RAG w/ NL grounding	João Paulo Vieira has a total of 45 vacation days. (WRONG)	3	555	20	1.442s
	"	Query data w/ sk (gpt-4)	João Paulo Vieira has a total of 47 vacation days. (WRONG)	-	683	105	28.270s
	"	Query data w/ sk (gpt-3.5)	João Paulo Vieira has a total of 0 vacation days. He has not taken any vacations yet. (WRONG)	-	693	274	45.198s
12.	**I want to know in April who booked a vacation in 2024**	RAG w/ JSON grounding	Sorry, I don't know the answer.	0	46	15	1.746s
	"	RAG w/ NL grounding	Sorry, I don't know the answer.	0	38	15	1.298s
	"	Query data w/ sk (gpt-4)	Paula Maia from Skills Workflow, Skills division, Finance department, who is a Finance & Office Assistant, has booked a vacation in April 2024. They have scheduled 1 days of vacation and still have 6 days available to schedule. Catarina Alves from Skills Workflow, Skills division, Human Resources department, who is a Head of HR, has booked a vacation in April 2024. They have scheduled 2 days of vacation and still have 2 days available to schedule. Afonso Cardoso from Skills Workflow, Skills division, Operations department, who is a Head of Customizations, has booked a vacation in April 2024. They have scheduled 4 days of vacation and still have 10 days available to schedule...	-	687	169	30.532s
	"	Query data w/ sk (gpt-3.5)	Paula Maia from Skills Workflow booked a vacation in April 2024. Catarina Alves from Skills Workflow booked a vacation in April 2024. Afonso Cardoso from Skills Workflow booked a vacation in April 2024. Marta Brito from Skills Workflow booked a vacation in April 2024. Rafael Levy from Skills Workflow booked a vacation in April 2024. Marina Mello from SkillsWorkflow Brasil booked a vacation in April 2024.	-	695	143	22.818s

Table A.4: Experiments Comparing RAG and Query approaches (continuation).

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
13.	**From the development department who have vacations for the next month this year?**	RAG w/ JSON grounding	Sorry, I don't know the answer.	0	47	15	2.150s
	"	RAG w/ NL grounding	Sorry, I don't know the answer.	0	39	15	1.231s
	"	Query data w/ sk (gpt-4)	No one from the development department has vacations scheduled for the next month this year.	-	685	272	44.673s
	"	Query data w/ sk (gpt-3.5)	The following people from the development department have vacations for the next month this year: Bruno Moscão, Eduardo Pinto, Fernando Romano, Jose Cardanho, Rui Evaristo, Helder Barreiros, Rodrigo Alves, Joao Figueira, Frederico Tereso, Angela Dias, Paula Maia, Telmo Carvalho, Catarina Alves, João Paulo Duque Vieira, Afonso Cardoso, Vasco.Mensurado, Marta Brito, Rita Mensurado, Ines Sousa, Birger Kamrath, Rodrigo Mensurado, Abdul Kalam, Rafael Levy, Vanessa Rosenstock, Sonia Pereira, Carlota Melo, Helena Nunes, Marina Mello. (WRONG)	-	693	128	24.640s
14.	**Has the work for the SuperUnion client been approved? What the billing has?**	RAG w/ JSON grounding	The work for the SuperUnion client has been approved. The billing for this work is R$6,288.15.	5	1883	31	1.867s
	"	RAG w/ NL grounding	Yes, the work for the SuperUnion client has been approved. The billing for this work order is R$6,288.15.15.	5	958	34	1.569s
	"	Query data w/ sk (gpt-4)	Work order 2023 - 00116 for SuperUnion has been approved. The billing for work order 2023 - 00116 is 6288.15.	-	609	250	45.643s
	"	Query data w/ sk (gpt-3.5)	The work for the SuperUnion client has been approved. The billing amount is 384510.10940000013.	-	617	129	23.115s
15.	**How many orders are in stage New?**	RAG w/ JSON grounding	There are 6 orders in stage New. (WRONG)	7	2675	15	1.693s
	"	RAG w/ NL grounding	There are 7 orders in stage New.	7	1367	15	1.297s
	"	Query data w/ sk (gpt-4)	There are 7 unique orders in the "New" stage.	-	593	109	27.046s
	"	Query data w/ sk (gpt-3.5)	There are 7 orders in stage New.	-	590	88	22.399s
16.	**Does Tiago Henriques has any scheduled vacations in 2024?**	RAG w/ JSON grounding	Sorry, I don't know the answer.	0	47	15	1.221s
	"	RAG w/ NL grounding	Sorry, I don't know the answer.	0	39	15	1.168s
	"	Query data w/ sk (gpt-4)	Tiago Henriques has no scheduled vacations in 2024.	-	685	270	47.861s
	"	Query data w/ sk (gpt-3.5)	Tiago Henriques has no scheduled vacations in 2024.	-	684	131	22.824s

Table A.5: Experiments Comparing RAG and Query approaches (continuation).

Nº	Question	Approach	Answer	N Docs	Input tokens	Output tokens	Response Time
17.	**How many vacations does João Paulo has by month?**	RAG w/ JSON grounding	João Paulo Vieira has 22 vacations by month in January, February, March, April, May, June, July, August, September, October, November, and December. (WRONG)	3	1725	43	1.819s
	"	RAG w/ NL grounding	In 2024, João Paulo Vieira has no vacations scheduled for any month. (WRONG)	3	555	24	1.315s
	"	Query data w/ sk (gpt-4)	João Paulo has scheduled vacations as follows: January: 0 days, February: 0 days, March: 0 days, April: 0 days, May: 0 days, June: 0 days, July: 0 days, August: 0 days, September: 0 days, October: 0 days, November: 0 days, December: 2 days.	-	679	330	61.179s
	"	Query data w/ sk (gpt-3.5)	João Paulo has 139 vacations by month. (WRONG)	-	676	244	24.384s

Table A.6: Experiments Comparing RAG and Query approaches (continuation).

Note: The "N Docs" is only used in the RAG approach because it need x documents to use as grounding

B

Appendix B

Appendix B contains the user guide for testing the Skillio assistant that was used in a Google Form given to the test users. This is a utility test for Skillio AI assistant, a personalized assistant from Skills Workflow. The test is divided into two parts. The first part is to test the assistant in the current phase of the study. In a controlled environment with questions over a predefined scenario. The second part has open questions to help improve the assistant in the current phase, so we receive insights into other functionalities that could be available or provide feedback on the assistant.

B.1 Briefing

In many applications, processes are often time-consuming for users. These can be automated, and basic tasks can be simplified.

- Fight the slow and time-consuming learning curve of the user interface.
- Increase performance and reduce the time spent on repetitive tasks, which can be automated.

Skillio understands your intentions, just express yourself the best you can. It possesses the following features:

- **Configurations:** Skillio is capable of altering language, behavior, notifications preferences, and includes a "do not disturb" mode.
- **Data Analysis:** Users can ask Skillio to provide insights about system data (only available data from Jobs and Leaves).
- **Notifications Summary:** User notifications are summarized in a message automatically when you open the assistant.

Read the following scenarios, and complete the questions asked. After finishing the questions, play as much as you would like with the assistant and answer 4 simple and quick questions to evaluate the assistant.

Note: this is just a **prototype**, therefore you may encounter some limitations or minor problems.

Choose your persona role from the ones below.

- Consultancy
- Development

B.1.1 1. Vacation Verification

Imagine you are working on a project for a client. You need to use some resources and you recall someone you'd like to add to the project. However, you need to check if this person is on vacation. Ask Skillio to check this resource's vacation plans.

Afterward, try to handle the same task without relying on Skillio to compare the approaches.

- Question 1. What question(s) did you ask Skillio? And what answer(s) did it give you? (Screenshots)
- Question 2. Was the answer(s) adequate? (From 1 to 5)
- Question 3. Do you think this method is more efficient than manually conducting this finding on the platform? (as you normally would) (Question 3.1 Yes/No; Question 3.2 Why).

B.1.2 2. Configuration Exploration

Imagine you are setting up Skillio for the first time and want to change the language to Portuguese, enable the "do not disturb" mode for the summary of the notification, and ask it to be more funny and enthusiastic. After that, ask Skillio what it can do.

- Question 1. What question(s) did you ask Skillio? And what answer(s) did it give you? (Screenshots)
- Question 2. Was the answer(s) adequate? (From 1 to 5)
- Question 3. Do you think this method of changing configurations is more efficient than manually adjusting settings through a traditional panel with buttons and selection boxes? (Question 3.1 Yes/No; Question 3.2 Why).

B.1.3 3. Task Assignment

Imagine that you are assigning tasks to employees and believe that a particular employee is the best choice for the job. However, you are not sure if they have a lot of other work to do. Ask Skillio to check the tasks this employee is currently responsible for. Also, to see their success rate, you want to know if in the past they had a lot of tasks closed. And after, how many were rejected? If they have, ask for details. Afterward, try to do the same without relying on the task management system to compare the approaches.

Please answer the following questions at the same time:

- Question 1. What question(s) did you ask Skillio? And what answer(s) did it give you? (Screenshots)
- Question 2. Was the answer(s) adequate? (From 1 to 5)
- Question 3. Do you think this method is more efficient than manually conducting this finding on the platform? (Question 3.1 Yes/No; Question 3.2 Why)

B.1.4 4. Deadline Checks

Imagine you're attending a team meeting where you usually check in on the status of jobs nearing their deadlines. This time, you want to approach it in a more cheerful way using Skillio, so tell him to talk happily. Then, ask Skillio for a detailed resume of the jobs closest to their end dates. Afterward, try to manage the task without Skillio to compare the different approaches.

Please answer the following questions at the same time:

- Question 1. What question(s) did you ask Skillio? And what answer(s) did it give you? (Screenshots)
- Question 2. Was the answer(s) adequate? (From 1 to 5)
- Question 3. Do you think this is more efficient than manually conducting this finding on the platform? (the summary contains the sufficient) (Question 3.1 Yes/No; Question 3.2 Why).

B.1.5 5. Previous Projects

Imagine you're managing a major project and remember that one of your employees worked on a similar project in the past year. You know it was one of three specific employees. So, ask Skillio to provide a list of each team member's tasks from that year with their details. Then, try tackling that task without relying on the assistant to compare approaches.

Please answer the following questions at the same time:

- Question 1. What question(s) did you ask Skillio? And what answer(s) did it give you? (Screenshots)
- Question 2. Was the answer(s) adequate? (From 1 to 5)
- Question 3. Does the answer provide you with information to help you find what the project name may be? (Question 3.1 Yes/No; Question 3.2 Why).

B.1.6 Open-end Evaluation Questions

Here you can explore Skillio a bit and give us your feedback. In the end, please just answer the following questions:

- Question 1. What kind of improvements would you like to see made from what has already been added?

- Question 2. Is there any other functionality/automation you would like to see added?
- Question 3. In your day-to-day, would you like to use Skillio to handle several tasks (not only the ones available in the prototype)?
- Question 4. With what the assistant does now, is there are any current issues that you can resolve using the assistant?