Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Bachelor Thesis 2023

Towards Large Language Models for robotic nutritional coach

Source : https://luxai.com/wp-content/uploads/2022/03/QTrobots-for-research--700x696.png

Student : Ismael Jaggi

Professor : Michael Schumacher

Submitted on : 24. November 2023

HES-SO Valais-Wallis • rte de la Plaine 2 • CP 80 • 3960 Sierre
+41 58 606 93 11 • info.informatique@hevs.ch • www.hevs.ch/ig

Abstract

The objective of this bachelor thesis is to integrate a Large Language Model (LLM) into a QTrobot to create a nutritional coach. In this endeavour, the unique capabilities of the robot, such as facial expressions, gestures, and speech, are combined with the generated responses of the Large Language Model to ensure efficient and user-friendly coaching.

The initial phase involves providing an overview of LLMs and robots already employing LLMs. Subsequently, various open-source LLMs are introduced and compared to identify the most suitable one for the project. The selection process led to the adoption of Llama 2 from MetaAI due to multiple factors.

Given the computational limitations of the robot, preventing the local installation of LLM, alternative solutions were explored. DeepInfra with the Llama 2 70B API emerged as the optimal choice owing to its ease of use and cost-effectiveness.

The implementation on the robot commenced with the creation of a user profile and a login. Subsequently, the two main functions, nutritional coaching, and meal suggestions, were implemented using the DeepInfra API. To enhance user interaction, gestures, facial expressions, and emotions were integrated using the Google Cloud Sentiment API and the functionalities of the robot. Since the program is intended to function in a terminal without a robot, a Platform Abstraction Layer was incorporated. Additionally, in the terminal version, Google Cloud Text-to-Speech (TTS) and Speech-to-Text (STT) APIs were utilized to improve the user experience.

Throughout the implementation phase, test scenarios were devised and conducted three times: twice on the robot and once for the terminal version. To gain an external perspective on the outcome, a user test was conducted at the conclusion of the project.

Keywords: Large Language Model, Speech-to-Text, Text-to-Speech, Robot, Nutrition, QTrobot

Foreword

This bachelor's thesis was conducted as part of the Business Informatics degree program at Hes-so Valais-Wallis. The work was carried out between September 19, 2023, and November 24, 2023. The topic was proposed by Professor Michael Schumacher, who also supervised the execution.

The subject of this bachelor's thesis is "Towards Large Language Models for robotic nutritional coach." Within this topic, two future technologies are encompassed—both robotics and Large Language Models are poised to reshape the field of computer science in the coming years. Additionally, during the course of this work, I gained my initial experiences with Text-to-Speech and Speech-to-Text technologies. It was a significant enhancement of my skills to delve into these topics through this bachelor's thesis.

I would like to express my gratitude to Professor Michael Schumacher and my coach Davide Calvaresi, who were always accessible to provide guidance and support whenever I had questions. Furthermore, I extend my thanks to Gaetano Manzo, who, in the context of a meeting, offered further insights into working with Large Language Models and provided his assistance. Additionally, I would like to thank Nicolas Bellwald for proofreading my thesis.

Table of content

List of Tables

List of Figures

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ARC	AI2's Reasoning Challenge
CPU	Central Processing Unit
GPU	Graphics Processing Unit
LLM	Large Language Model
MMLU	Massive Multitask Language Understanding
NLP	Natural Language Processing
TTS	Text To Speech
VM	Virtual Machine

1. Introduction

In recent years, technological advancements in robotics and Large Language Models (LLMs) have exerted a significant impact on various aspects of our lives. The continuous improvement of robots and the introduction of powerful LLMs have elevated human-like interactions and artificial intelligence to new heights. The objective of this bachelor's thesis is precisely to merge the vast knowledge and capabilities of an open-source LLM with the functions and possibilities of a humanoid robot to achieve the best possible and user-friendly nutritional coaching.

The thesis is structured into different sections. Initially, LLM and Prompt Engineering are theoretically explained to establish a knowledge base for subsequent chapters. This is followed by an overview of the current state of robots that already utilize LLMs. Subsequently, the currently most significant open-source LLMs are listed and examined. To identify the suitable LLM for this work, various characteristics of the LLMs and their performance in various benchmark tests are compared. After the decision, the installation of the LLM is considered, and alternatives for local installation are listed and compared. Then, a description of the architecture used is provided, along with a description of the implementation. Finally, test results from manual tests and external user tests are presented and analysed.

To adhere to a schedule, a new sprint was initiated every Tuesday. These will be explained in more detail towards the end of the thesis. The short sprints allowed for quick responses to emerging issues, increased adaptability, and enhanced communication among all stakeholders. Additionally, a product backlog was created to keep track of pending tasks and gain a good overview of the current work status. In addition to individual sprint planning and the breakdown of user stories into tasks, a journal was maintained to document the work each day. Thanks to the journal, it was possible to monitor progress daily and gain a better understanding of the actual duration of each task.

Within the scope of this bachelor's thesis, ChatGPT was utilized to transform previously written text into improved English without altering the content of the text. The prompt used for this purpose was:

Write this text in good, scientific English, but without changing the content: (text)

2. State of the Art

2.1 What is a Large Language Model?

Large Language Models, often referred to as LLMs, represent a recent advancement within the realms of artificial intelligence (AI) and natural language processing (NLP). These models possess a remarkable capability to not only comprehend human language but also to generate it. A prominent illustration of LLM application is exemplified by ChatGPT, a chatbot harnessed by OpenAI, which leverages the prowess of these language models. Nevertheless, it's important to note that the utility of LLMs extends well beyond chatbots, encompassing a wide array of applications. These versatile language models find utility in tasks ranging from text generation and translation to the creation of chatbots and even automated customer support systems. As we will explore further in this chapter, many enterprises have begun integrating LLMs into robotic systems, enabling them to undertake tasks that don't necessitate pre-programming and instead adapt dynamically to new scenarios. This dynamic nature of LLM deployment reflects the transformative potential of these models across various domains. (Elasticsearch Inc., n.d.)

LLMs are rooted in the realm of artificial neural networks, a fundamental concept derived from the inspiration of the human brain within the domain of machine learning. Neural networks are structured as a series of layers, with each layer comprising artificial neurons that serve specific roles. Commencing with the input layer, this initial layer records and processes raw data, with each neuron corresponding to specific characteristics present in the data. Subsequently, multiple hidden layers follow, designed to discern intricate correlations and patterns embedded within the data. Throughout the training of the neural network, connections are forged between these neurons, and each connection is endowed with a weight. These weights signify the significance of one neuron's output concerning its influence on other neurons. The higher the weight, the greater the relevance of the neuron's output in the overall process. This mechanism of weighting empowers the neural network to prioritize pertinent information, determining its effectiveness in data processing. The last layer is the output layer, tasked with generating the ultimate results. The nature of this output layer can vary contingent upon the data upon which the neural network was initially trained. (Wuttke, 2023)

Many of the newer large language models are based on the Transformer architecture, which was first introduced in 2017. This architectural design has brought a significant shift in the way we approach machine learning for natural language processing. In the following explanation, the architecture will be explained as simply as possible. All the technical details will not be delved into, as they go beyond the scope and are not crucial for understanding. One of the key features of the Transformer architecture is the Self-Attention mechanism. This mechanism allows the network to pay special attention to specific parts of input text and build connections between words. This helps the model understand context, even across multiple sentences. Essentially, it calculates a weight for each word, showing how it relates to other words in a sentence. Additionally, the architecture uses positional encodings and input embeddings. To work with text data, it needs to be converted into a numerical form. Text is divided into smaller units called tokens, and each token is associated with a numeric vector called an embedding. (Klofat, n.d.) These embeddings are like numerical representations of words and their relationships. (Kaikramer Consulting, n.d.) During training, these embeddings are fine-tuned within a matrix that contains embeddings for each word or token. To provide the model with information about the position and order of words in the input text, positional encodings are added to these embeddings.

The Transformer architecture comprises two integral components: the encoder and the decoder, organized in layers of neural networks. The initial stage involves input data presented as a sequence of tokens, which is subsequently received by both the encoder and decoder. The encoder's primary task involves the creation of input embeddings and the incorporation of positional encodings. Following this, the Self Attention mechanism is employed to imbue these embeddings with weighted values. Subsequently, a Feedforward Neural Network processes the output. Feedforward Neural Networks convert input data into numerical values and relay them to subsequent layers. Typically, multiple such layers are cascaded, with the output of the final layer serving as input to the next. The encoder's output furnishes a comprehensive representation of the input sequence, encompassing semantic meaning and inter-token relationships. This output is relayed to the decoder, which is tasked with generating the target sequence. Within the decoder, similar layers are utilized, commencing with embeddings and positional encoding layers. This is succeeded by a layer featuring the masked self-attention mechanism, which serves to enhance computational efficiency by selectively masking portions of the input sequence. Subsequent to this, the output is transformed into a multidimensional vector and normalized through the application of a SoftMax function. Finally, the target sequence is meticulously generated token by token. The example provided here illustrates the application of the Transformer architecture, which can be adapted in various ways. There are instances in which solely the encoder is employed, and configurations where the Masked Self-Attention mechanism is integrated within the encoder and beyond. (Klofat, n.d.)

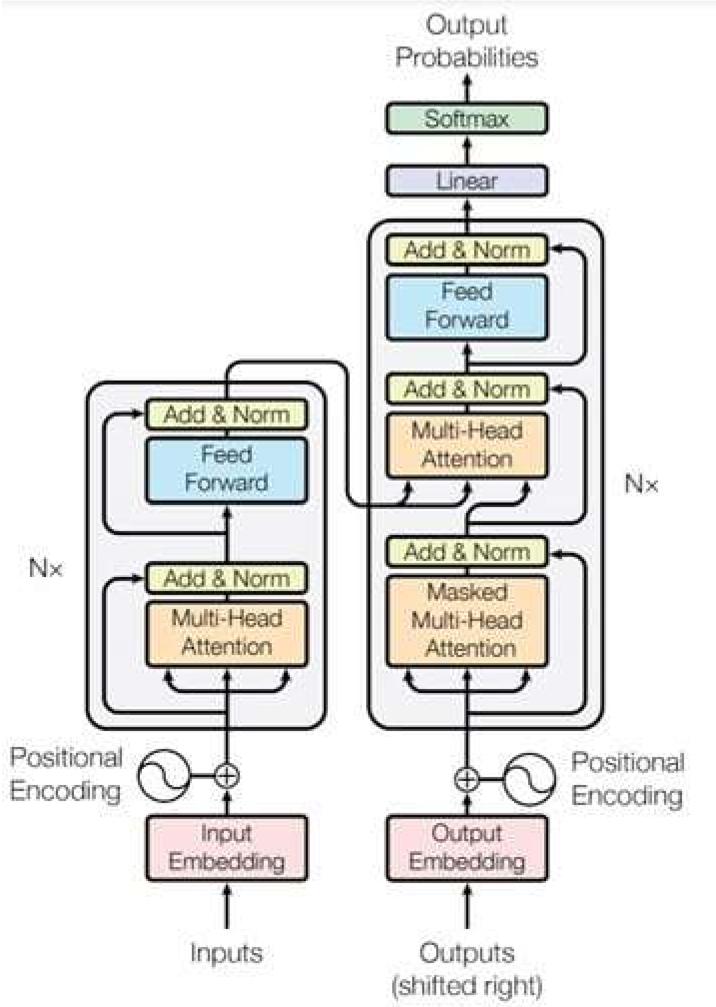


Figure 1: Transformer Architecture
Source: https://www.alexanderthamm.com/wp-content/uploads/Architecture -Overview-of-the-Transformer-.jpg

Before these language models can be employed, a crucial preliminary step is training. This process entails the utilization of extensive collections of text documents. Employing unsupervised learning techniques, LLM acquires proficiency in understanding words, discerning their semantics, and grasping their contextual usage. Furthermore, it establishes neural connections and assigns weights to these connections. The extent of training data and the abundance of model parameters, including neuron connections and their corresponding weights, directly impact the model's ability to accumulate knowledge and identify intricate patterns. Upon the culmination of the training phase, the LLM can be subject to fine-tuning for specific applications. This fine-tuning operation relies on supervised learning. During this phase, the LLM confronts various tasks along with their solutions, leading to the refinement of specific model parameters. It is essential to emphasize that the training process demands significant resources, encompassing substantial time investments, extensive memory capacity, and formidable computational power. (Luber, 2023a)

In summary, Large Language Models (LLMs) operate on the foundation of neural networks. These neural networks undergo training using big amounts of text data and can be further tailored for specific tasks through Fine-Tuning if required. Their proficiency in comprehending and generating text is rooted in the acquisition of language patterns and relationships. Numerous recent models are constructed based on the Transformer architecture, wherein the self-attention mechanism stands out as a pivotal feature, enabling the capabilities of LLMs in their present state. LLMs are already making significant inroads across diverse applications, and their utilization is anticipated to expand even more extensively in the future.

2.2 Prompt Engineering

An important key concept in the utilization of large language models is "Prompt Engineering." A prompt typically refers to a text-based input or instruction given to a generative AI system. In prompt engineering, one examines the requirements for obtaining the best results. When using an LLM, even minor changes to the prompt can lead to significant variations in outcomes. Therefore, the quality of the results heavily relies on the quality of the task description. Prompt engineering finds applications in various generative AI systems, including chatbots, language models, and text-to-image models. The increasing use of such generative AI systems has given rise to the profession of a Prompt Engineer, whose sole responsibility is the design and optimization of prompts. Once a prompt is optimized for a model, it is stored in prompt libraries, making optimized prompts readily available and eliminating the need to optimize prompts independently.

Various techniques are employed in prompt engineering. Inputs can be unstructured or structured, with structured inputs often breaking down the task into multiple sentences to assist the AI in better understanding the task, leading to improved responses. Another technique is "Role Prompting," where the AI is instructed on how to behave and respond in a specific style. This allows generating responses with varying levels of complexity. (Luber, 2023b)

In the context of new language models, it's possible to use zero-shot prompts, where tasks are presented to the AI without examples. For example:

Classify the following statement as positive, negative, or neutral.

I love apples.

Such prompts are straightforward and can be correctly answered without issues. In more complex scenarios, a few-shot prompt can be employed, which includes examples. For example:

I like cherries // positive

I hate bananas // negative

I enjoy eating pears // positive

I despise pineapples //

Another technique, especially useful for complex problems, is "Chain-of-Thought." Here, the AI is directed to break down problems into individual steps and solve the problem step by step. This can be achieved through zero-shot prompting by instructing the AI to solve the problem step by step, or by using a few-shot prompting to provide related problem-solving steps to be followed step by step. (Prompting Guide, n.d.)

As prompt engineering becomes more widespread, it also opens the door to malicious applications. The goal is to compel the AI to generate responses that the developer has prohibited. Several techniques are used for these purposes, including "Adversarial Prompting," "Prompt Injection," "Token Smuggling," "Jailbreaking," and "Prompt Leaking." The utilization of prompt engineering offers numerous advantages. It leads to more specific AI responses and enables the resolution of complex tasks. Moreover, by applying the presented techniques, the best possible responses can be generated. (Luber, 2023b)

2.3 LLM Powered Robots

The integration of Large Language Models (LLMs) into robots represents a transformative advancement, enabling them to comprehend non-predefined statements and respond adeptly to diverse scenarios. Robots, while already employed in numerous domains, hold the potential for even broader applications and the assumption of increasingly complicated tasks, thanks to the great progress achieved in the domain of large language models. Although widespread deployment of LLM-equipped robots has not been fully realized as of now, a multitude of projects are surfacing worldwide, actively engaged in the development of increasingly intelligent robotic systems. This chapter's primary objective was to delve into an array of application areas and methodological approaches while providing an overarching perspective on some of the most intriguing projects. Our exploration commences with the advancement of humanoid pilot robots in South Korea and progresses to the evolution of Google's language models. The incorporation of LLMs into industrial robots to enhance worker support is also examined. Subsequently, we scrutinize a publication from the University of Princeton, wherein a robot endowed with an LLM is tasked with executing room cleaning instructions through simple prompts. The chapter further introduces Furhat, a social robot capable of engaging in meaningful conversations. To conclude, light is shed on the robot utilized in the context of this bachelor thesis, the QTrobot.

2.3.1 Pibot

Pibot, short for "Humanoid Pilot Robot," is a humanoid robot specially designed for the task of piloting airplanes. This robot possesses the capability to effectively manage and operate the multitude of buttons and levers typically found in an aircraft's cockpit. Furthermore, it can establish a direct connection with the aircraft's control systems. Pibot was meticulously developed by a dedicated team hailing from the Korea Advanced Institute of Science & Technology (KAIST), undertaking this significant project on behalf of South Korea's governmental agency responsible for advancing defense technology..

Figure 2: Pibot Robot
Source: https://static.euronews.com/articles/stories/07/82/32/60/1440x810_cmsv2_632a80fa-90b5-5e62-b65d-8de530c9d42e-7823260.jpg

The initial iteration of this robot was finalized in 2016, and at that time, it did not incorporate a Large Language Model (LLM). However, with the significant advancements in LLM technology, this robot is now capable of comprehending and interpreting human-written manuals. As a result, it can swiftly learn to operate new types of aircraft and respond with remarkable speed and efficiency, particularly in high-pressure situations, outperforming its human counterparts. Moreover, Pibot can retain an extensive database of Jeppesen flight charts, enabling it to calculate the optimal flight path by leveraging aircraft-specific data. The integration of LLM technology also allows Pibot to engage in effective communication with humans and serve as a co-pilot to intervene in critical scenarios. Although the robot currently employs ChatGPT, which necessitates an Internet connection, the research team is actively developing a dedicated LLM tailored for flight operations. This local LLM will be stored within Pibot, reducing its dependency on an external connection. Pibot's expected completion date is set for 2026. (McFadden, 2023)

2.3.2 Google

Google is actively engaged in enhancing its large language model for integration into robotic systems. Google originally developed its proprietary large language model, PaLM, which has now evolved into its successor, PaLM-E. Unlike its predecessor, PaLM-E has transitioned into a broader category referred to as a "visual language model." In this updated version, training data is drawn not only from text but also from images and data sourced from robot sensors. Furthermore, the input methodology differs from conventional language models, as it accommodates a combination of text and image files. PaLM-E currently stands as the most extensive visual language model with an impressive 562 billion parameters. Throughout the training process, the system learns to fine-tune these variables to enhance its performance in subsequent prompts, such as sentence composition. For research and experimentation purposes, this model has been collaboratively developed with researchers from the Technical University of Berlin and successfully integrated into a robotic arm.

Figure 3: PaLM-E Robot
Source: ttps://cdn.arstechnica.net/wp-content/uploads/2023/03/palm-e-robot-800x450.png

For instance, consider the scenario where the task is to retrieve rice chips from a drawer. Thanks to the language model, the robot autonomously navigates the kitchen, opens the drawer, and neatly places the rice chips on the table. Notably, Google's PaLM-E model has attained the highest score in the OK-VQA test thus far. OK-VQA, or Open Knowledge Visual Question Answering, is a benchmark used to assess the accuracy of language models. The ultimate aim is to develop robots capable of adapting to novel environments and seamlessly executing everyday tasks. To excel in these tasks, an essential human trait is currently lacking: the ability to apply pre-existing knowledge to new challenges, which is commonly referred to as Positive Knowledge Transfer. (Szöke, 2023)

2.3.3 Fruitcore Robotics

In the industrial domain, there is a growing trend of integrating LLMs into robots to simplify their usability for humans. Fruitcore Robotics made a significant development in this area by introducing a new operating system with an integrated AI co-pilot for their Horst industrial robots at the Automatica 2023. These Digital Robots are pioneering the use of ChatGPT for setup and control, enabling users to engage with an intelligent assistant in real-time using natural language.

Figure 4: Horst Robot
Source: https://www.fruitcore-robotics.com/hs-fs/hubfs/Website/webP%20intern%20bearbeitet/01_Heros/fruitcore-robotics-horst600-header-truncated-1.webp?width=1800&height=1589&name=fruitcore-robotics-horst600-header-truncated-1.webp

The implementation of artificial intelligence in the Horst robots is expected to substantially enhance automation processes, making them more adaptable and user-friendly. With the robot's assistance, tasks related to control and integration can be executed more efficiently and swiftly, resulting in time and cost savings. This is particularly advantageous for both novices and experienced workers, as it simplifies robot operation and enhances communication between humans and robots. Users can seek real-time answers from the robot whenever questions or issues arise, reducing the need for external support requests. The AI co-pilot can assist with tasks such as transferring part positions to the robot, provide real-time guidance, generate functions, program modules, templates, optimize or correct existing programs, and even offer suggestions for program error corrections. Moreover, the AI co-pilot serves as a valuable resource for users to further educate themselves in robot handling and programming. Fruitcore Robotics is committed to continuously expanding the capabilities of the AI co-pilot to ensure an exceptional user experience. This co-pilot has been trained using assembly instructions, support content, and software documentation provided by Fruitcore Robotics. (Import, 2023)

2.3.4 TidyBot

At Princeton University, an alternative approach was adopted in a project where a team of nine individuals collaborated to create a household-assisting robot. In this particular instance, the robot's primary function is to tidy up a room in accordance with the user's preferences.

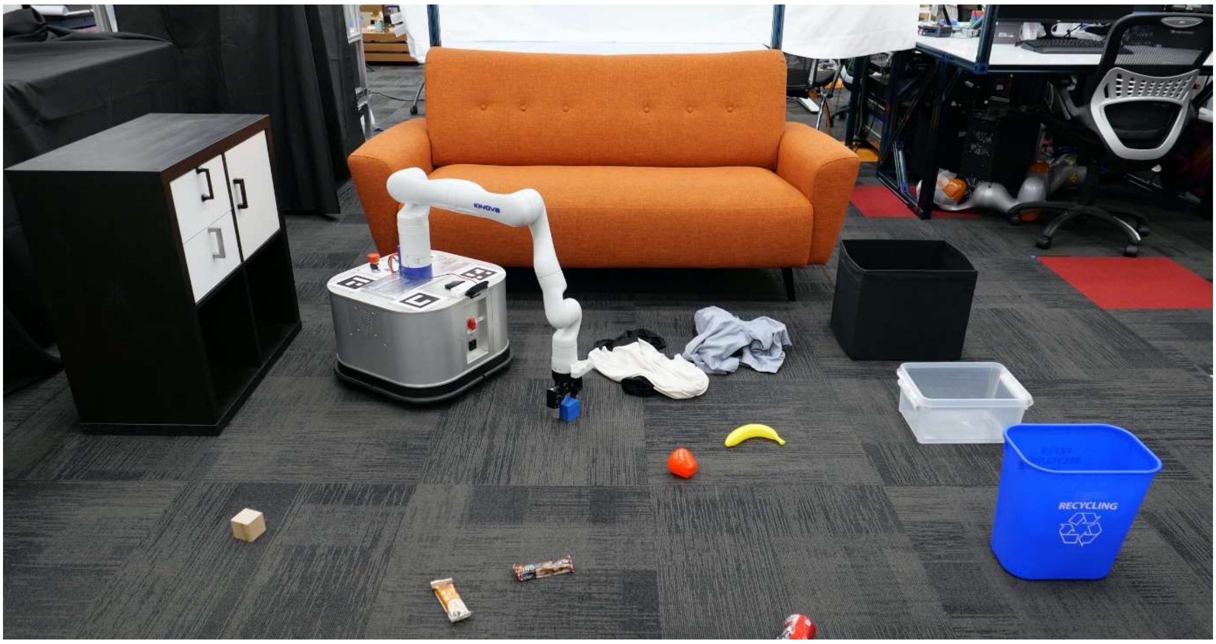


Figure 5: TidyBot
Source: https://tidybot.cs.princeton.edu/images/og-image.jpg

In this scenario, the user begins by inputting their preferences through a text interface. For instance, they may specify that white socks should be stored in the bottom drawer, yellow shirts in the middle drawer, and black shirts on a chair. Leveraging the capabilities of the Large Language Model, general rules are then derived from these preferences. For instance, these rules might dictate that all socks belong in the bottom drawer, light-coloured shirts in the middle drawer, and dark-coloured shirts in the top drawer. The LLM's second task involves recognizing new objects encountered during the tidying process and correctly assigning them to the established rules so that the robot can organize them accurately. Notably, this project utilized a pre-trained LLM without any additional fine-tuning. The LLM exhibited an impressive 91.2% accuracy in correctly categorizing previously unseen objects. In real-world settings, the TidyBot successfully sorted 85% of items to their designated locations. This innovative project was shared in 2023. (Wu et al., 2023)

2.3.5 Furhat

Furhat is a cutting-edge social robot that facilitates natural, human-like conversations. It consists of an animated face mounted on a fixture embedded with microphones, speakers, and cameras. This animated face is available in 22 diverse templates, and each can be further personalized to simulate various ages, genders, and ancestries. The facial animation is achieved through a projector that casts the face onto a translucent mask. Additionally, masks resembling a child, a dog, or even anime characters are available for purchase. Furhat is capable of emulating all human facial expressions, aiming to deliver the most realistic conversation experience possible for users. With a selection of over 200 different voices, users have plenty of options for voice customization. The robot is equipped with a face recognition camera that enables it to maintain eye contact during interactions and distinguish between different speakers, allowing for multi-person conversations. (Furhat Robotics, 2023)

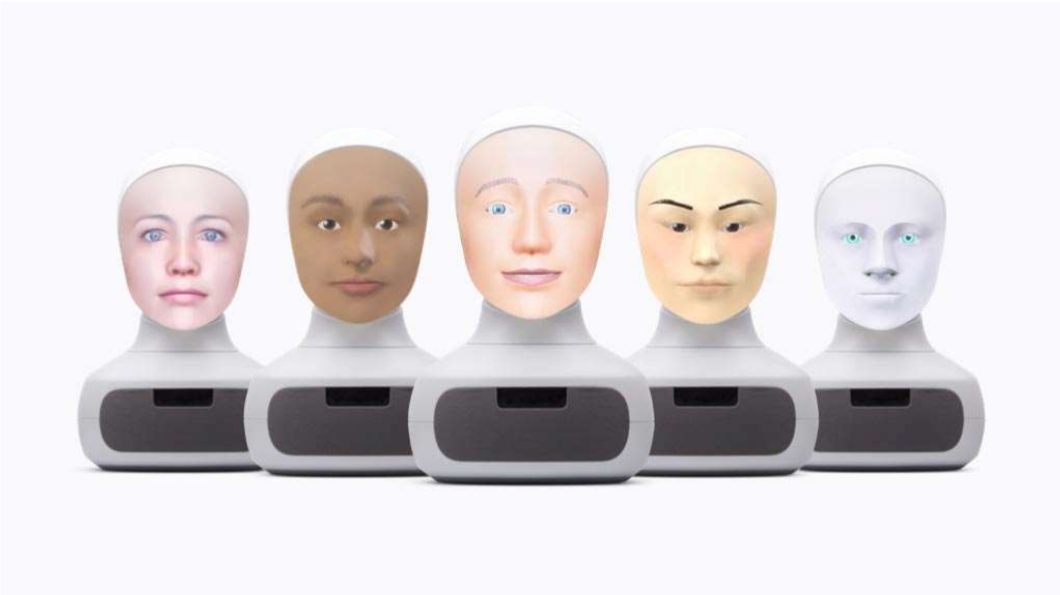


Figure 6: Furhat Robots
Source: https://furhatrobotics.com/wp-content/uploads/2021/10/furhat-robotics.jpg

LLMs can be effectively integrated into these robots to enhance the fluency of conversations and provide a broader spectrum of responses, eliminating the need for pre-scripted interactions. For an optimal user experience, social robots like Furhat must consider a multitude of factors beyond what's typically required for chatbots like ChatGPT. These considerations encompass selecting appropriate facial expressions and adjusting the pace and pitch of the robot's voice in response to the user, collectively referred to as "backchanneling."

It's crucial for users to perceive the robot as actively engaging in the conversation, delivering answers organically rather than reciting them in a robotic and rigid manner. While current LLMs have paved the way for more natural interactions, there's still ample room for further improvement in this domain. (Client, 2023)

2.3.6 QTrobot

The QTrobot is a humanoid social robot created by LuxAI for research purposes. This robot is primarily designed to work with autistic children, which is reflected in its childlike appearance and gentle voice. In contrast to the Furhat, the QTrobot features a screen as its face, allowing it to display a range of facial expressions and emotions. The robot boasts a user-friendly graphical interface, enabling even those without IT expertise to program it. Additionally, the QTrobot is equipped with a 3D camera, a microphone, speakers, and motors in its arms and neck. These features empower the QTrobot to perform basic gestures, articulate sentences in various languages using Text-to-Speech (TTS) technology, and understand simple spoken words through offline speech recognition. It can also utilize the camera to detect emotions, gestures, and faces, thereby recognizing them. Importantly, the QTrobot supports the use of Large Language Models, providing the same benefits as the Furhat in this regard. (LuxAI S.A., 2023)

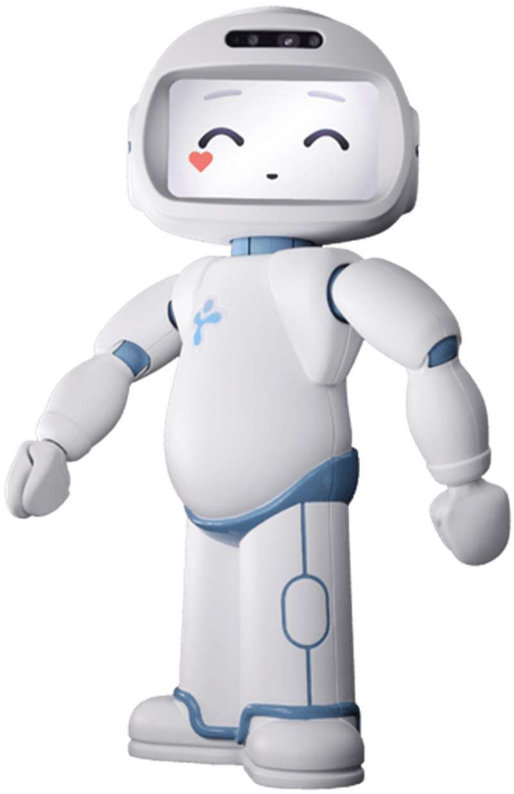


Figure 7: QTrobot
Source: https://luxai.com/wp-content/uploads/2022/04/QTrobot-software-information-for-developers-h.png

2.4 Open Source LLM

In this section, a deeper exploration will be undertaken into various Open-Source Large Language Models to identify the most suitable one for the project. Initially, brief introductions to each of them will be provided, followed by a comprehensive comparison. Factors such as the LLM's size, speed, and its specific capabilities are all of significance for the project. The ensuing comparison pertains to the following LLMs, which are currently among the most potent open-source models available.

Name	***Company***	***Model Size***
Llama 2 [23]	MetaAI	7B, 13B, 70B
Claude 2 [35]	Anthropic	approx. 130B
Falcon [10]	Technology Innovation Institute	1.3B, 7.5B, 40B, 180B
MPT [26]	MosaicML	7B, 30B

Table 1: Open Source LLM

Source: Own representation

2.4.1 Llama 2

Llama 2 is the latest iteration of MetaAI's Large Language Model, developed in collaboration with Microsoft. Compared to its predecessor, Llama 2 boasts extensive improvements, having been trained on 40% more data, which enhances its understanding of language. This upgraded model was introduced in July 2023 and is available for both research and commercial purposes without any usage fees. Llama 2 offers various model sizes, ranging from 7 billion to 70 billion parameters. In the context of text-based outputs, the 70B version of Llama 2 outperforms GPT-3.5 but falls short compared to GPT-4.

However, Llama 2 demonstrates limitations in code generation tasks, especially when compared to its competitors. In addition to the standard models, MetaAI has introduced two fine-tuned variants designed specifically for chat applications, featuring either 7, 13 or 70 billion parameters. Furthermore, a specialized code generation model named Code Llama is built upon the foundation of Llama 2. (Portakal, 2023)

2.4.2 Claude 2

Claude 2 is Anthropic's latest large language model. Regrettably, it is currently only accessible for testing in the United States and England, making it ineligible for use in this project. Nevertheless, it holds considerable interest for comparison with other LLMs. Distinct from Llama 2, Claude 2 comes with certain usage restrictions. It is not permitted for commercial purposes and entails a usage fee of $11.02 per one million tokens. Claude 2 excels in code generation tasks and can handle extensive context with a length of up to 100,000 tokens, allowing it to review entire documents of approximately 75,000 words. This model boasts around 130 billion parameters. Local installation of Claude 2 is not feasible, and users must access it through an API or the beta website. Furthermore, Claude 2 supports output generation in over 10 different languages. (Portakal, 2023)

2.4.3 Falcon

Falcon is the large language model developed by the Technology Innovation Institute, a research institution funded by the government of Abu Dhabi. Falcon offers a range of model sizes, starting from 1.3 billion parameters and extending to 7.5 billion, 40 billion, and the largest LLM featuring a substantial 180 billion parameters. Falcon 40B is proficient in numerous European languages, including English, German, French, and Italian. This model has been meticulously trained on a staggering dataset comprising 1 trillion tokens. An appealing aspect of Falcon is its open-source nature; it is offered under the Apache 2.0 license, permitting the use and distribution of its source code. (*Falcon LLM*, n.d.) In order to cater to a wide spectrum of applications, including simpler chatbots, Falcon has produced a specialized variant with 7.5 billion parameters named "7B-instruct." This variant is specifically fine-tuned for such applications. (Arya, 2023a)

2.4.4 MPT

The MosaicML Foundation unveiled its own Large Language Model, known as MPT-7B, at the outset of 2023. As implied by the name, this model is comprised of 7 billion parameters, classifying it among the smaller models in the landscape. Similar to the Falcon LLM, MPT-7B was trained on an extensive dataset of one trillion tokens. An intriguing aspect of its development is the swiftness with which it was achieved. Completed within a mere 9.5 days, and notably without human intervention. The entire effort cost MosaicML approximately $200,000.

Upon its release, MPT-7B was made available in three distinct variants. These include MPT-7B-Chat, designed for integration into chatbots, MPT-7B-Instruct, tailored for processing concise instructions, and MPT-7B-StoryWriter-65k, formulated for the generation of lengthy narratives. The "65k" designation indicates the context length of this model. Importantly, all three variants may be utilized for both personal and commercial purposes. In terms of qualitative performance, this model size is roughly on par with the initial Llama-7B version. Encouraged by the success of the 7B version, MosaicML also unveiled a 30B variant in the middle of the year. (MosaicML, n.d.)

2.4.5 LLM Overview

	Llama 2 [23,34]	*Claude 2 [3,34,35]*	*Falcon [10,43]*	*MPT [5,25,26]*
Pricing	Free	11.02$ /million token	Free	Free
Model Size	7B, 13B, 70B	Approx. 130B	1.3B, 7.5B, 40B, 180B	7B,30B
Local Installation	Yes	No, API	Yes	Yes
Context Length	4k Token	100k Token	2k Token	2k (65k StoryWriter)
Fine Tuning	Yes	No	Yes	Yes
License	Personal /Commercial Use	Personal / Non-Commercial Use	Apache 2.0	Personal /Commercial Use
Language Support	20	10	European Languages	English

Table 2: Open Source LLM Overview

Source: Own representation

3. LLM Comparison

In the process of determining the most suitable LLM for the project and eventual implementation, a comprehensive comparative analysis of these models is undertaken. An essential consideration in this evaluation is the current accessibility of Claude 2, which is limited to the US and UK regions. Consequently, Claude 2 will not be a part of the considerations outlined here. Furthermore, the objective is to facilitate the local installation of the chosen LLM onto the QTrobot, eliminating the need for a constant internet connection. In alignment with this objective, the selection of more compact models is imperative, as the robot's computational capabilities and memory capacity are constrained, making larger models impractical. As a result, the models under comparison encompass Llama 2 7B, Llama 2 13B, Falcon 7B, and MPT 7B. To ensure a comprehensive assessment, commercial models, namely GPT-4 and PaLM 2, are also introduced, as they offer a distinct perspective on LLM capabilities. It's noteworthy that the commercial models are notably larger in scale compared to their open-source counterparts, thereby necessitating the inclusion of the most extensive model within the Llama 2 lineup for a comprehensive evaluation.

3.1 General Comparison

For the comprehensive comparison of the various LLMs, Table 2: Open Source LLM Overview, serves as the primary reference point. It's important to note that there are no cost differentials among these models, as they can all be utilized without any fees. The same applies to the possibility of local installation, a feature that is universally available across all models. The initial point of differentiation emerges in the context length, representing the allowable input length. Notably, Llama 2 boasts a context length twice the size of its counterparts, affording the capacity to process more extensive data in a single instance. However, in the context of this specific project, this distinction is inconsequential. This is because the robot's inquiries pertain to nutrition and consequently fall well below the maximum context length. To provide some context, 100 tokens roughly translate to approximately 75 words. Thus, even with a context length of 2000 tokens, accommodating input containing approximately 1500 words would be feasible. Such extensive input scenarios do not apply to this project. Fine-tuning capabilities are also offered by all models, although they do not hold relevance for this project. Importantly, there are no licensing concerns, as each of these models is available for both personal and commercial utilization. One pivotal facet of this comparison pertains to language support. Llama 2 stands out by providing support for more than 20 languages, encompassing all major languages. Falcon extends its support to European languages, which is more than adequate for this project's scope. In contrast, MPT is regrettably limited to English, a constraint that may not pose immediate issues for this project. However, for future expansion possibilities, this limitation could potentially lead to complications.

To facilitate a comparison with commercial large language models, the information concerning GPT-4 and PaLM 2 is presented in the table below. Notably, PaLM 2 from Google has recently been made accessible to the public. Over the next month, it remains possible that details such as pricing and commercial use policies may undergo revisions.

	Price [12,30]	*Local Inst. [12,37]*	*Context Length [30, 38]*	*Fine Tuning [37, 38]*	*License [27, 29]*	*Language Support [12,24]*
GPT-4	$0.03/1k Input Token $0.06/1k Output Token	No, API	8k Token	Yes	Personal / Commercial Use	26
PaLM 2	$0.0005/1k input / output Tokens	No, API	8k Token	Yes	Only Personal Use	100+

Table 3: Commercial LLM Overview

Source: Own representation

3.2 Benchmark Tests

To compare the performance of the individual LLMs in various tasks, a comprehensive analysis of benchmark test results was conducted. This was done to determine which LLM consistently delivers the best results. While slight variations in the specifications were encountered across different sources, these differences had minimal impact and did not compromise the overall findings. The performance data for the open-source LLMs was sourced from a single reliable reference, ensuring the representativeness of the data. Interestingly, the official MetaAI website displayed results for various LLMs, but most third-party assessments rated them lower than independent sources. The results, as displayed in Table 4: Benchmark Results, present the accuracy achieved by each LLM in these benchmark tests, all expressed in percentage values.

	Llama 2 7B [13]	*Llama 2 13B [13]*	*Llama 2 70B [13]*	*Falcon 7B*	*MPT 7B*	*GPT-4*	*PaLM 2 [1]*
HellaSwag [15, 33]	73.8	80.7	85.3	78.1	77.9	95.3	86.4
MMLU [15, 31]	45.3	54.8	68.9	35	30.8	86.4	81.2
ARC [15, 32]	53.07	-	73.04	47.87	47.7	96.3	89.7
TruthfulQA [15,29]	38.75	41.86	50.18	34.26	33.44	≈59.0	-

Table 4: Benchmark Results

Source: Own representation

To provide a clearer understanding, the beginning will involve outlining the contents of the various benchmark datasets and then proceeding to analyse the results.

HellaSwag: This test assesses the logical reasoning abilities of the LLM. It requires the completion of sentences in a manner that ensures they are coherent and sensible.

MMLU Dataset: This dataset comprises questions spanning over 57 subject areas, with the questions designed at an expert level, making it a robust measure of an LLM's proficiency.

ARC Dataset: Comprising questions that an 8th grader, typically a 13-14-year-old, should be able to answer, this dataset evaluates an LLM's ability to handle questions at a middle-school level.

TruthfulQA Dataset: This dataset is intentionally designed to pose questions that could be answered incorrectly due to the presence of misinformation. It serves as a means to assess whether LLMs can discern and provide correct answers, demonstrating their capacity to disregard false information encountered during training.

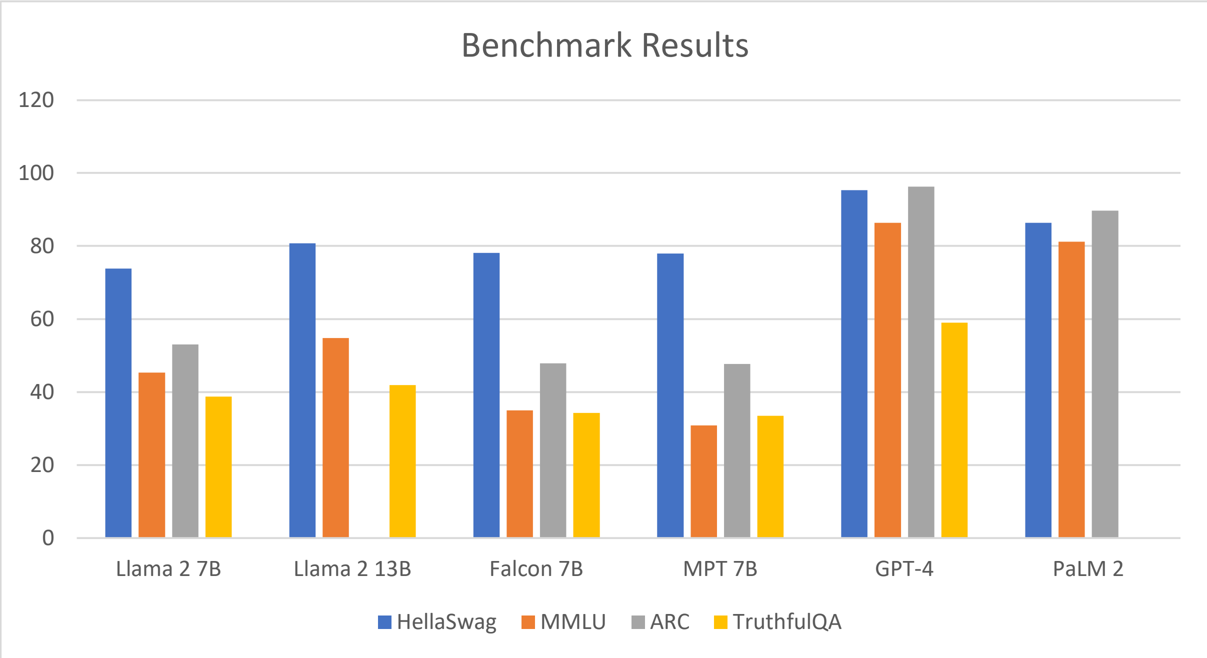


Figure 8: Benchmark Results
Source: Own illustration. Created with Excel.

These four datasets were selected for two primary reasons. Firstly, they provided the most extensive set of test results, enabling a meaningful comparison between LLMs. Secondly, these datasets collectively encompass a wide range of challenges that an LLM might face, offering a comprehensive overview of their overall capabilities. It's important to note that none of these benchmark tests assess the LLM's ability to generate code, a feature not required for this specific application. Therefore, a comparison involving such benchmark tests would not yield useful insights for this project. (Wieczorek, 2023)

Let's provide an analysis of the results. It's important to note that both GPT-4 and PaLM have shown a considerable lead over all other LLMs, which makes the open-source variants appear less competitive in a direct comparison. Even though some of the open-source models, particularly the larger one, Llama 2 70B, achieved commendable results, they could not match the performance of the two commercial LLMs.

Beyond the exceptional test results demonstrated by GPT-4 and PaLM 2, there is a clear trend among the open-source LLMs. In the 7B model size, Llama 2 outperforms Falcon and MPT in three out of four tests and maintains a distinct lead over other models in the 13B variant. Although we lack a specific test result for the 13B model of Llama 2, it can be inferred from the results of the 7B variant that the 13B version would likely yield similar or even superior results.

4. Selection of Technologies

4.1 Large Language Model

After a comprehensive evaluation and taking into consideration the benchmark test results, the following decision has been made regarding which LLM to use for the project. MPT 7B was ruled out during the general comparison due to its limitation to the English language and its subpar test results. While the idea of developing an LLM on a tight budget and within a short timeframe was intriguing, the final product couldn't compete with the alternatives. Even Falcon, with its 7B model parameters, didn't stand up to MetaAI's 7B model in three out of the four benchmark tests, leading to its exclusion. Given that the 13B version of Llama 2 LLM outperformed the 7B version, the decision is to utilize the 13B version, specifically Llama 2 13B Chat. The chat variant offers the advantage of being purpose-built for conducting conversations, enhancing user interaction. Moreover, it achieved an impressive score of 62.18 in the TruthfulQA benchmark (Hugging Face, n.d.-a). In addition to its superior test results, Llama 2 13B boasts additional advantages. These include detailed installation instructions on GitHub and the ability to program with Python. This aligns well with the robot's capability to be programmed using Python, offering a significant advantage. There was also the opportunity to test the Llama 2 13B Chat model on Hugging Face through a demo. Various questions were asked, and consistently correct responses in English were received. An especially useful feature was the ability to set a pre-prompt to filter out racist, sexist, or criminal responses from the outset. For instance, the model can be instructed to act as a nutrition coach and only answer questions related to nutrition. Careful selection of the pre-prompt's wording can simplify the implementation process. Regarding the multilingual capabilities promised by MetaAI, minor issues were encountered when receiving answers in German, with occasional incorrect word choices. However, these did not significantly impact the overall quality of the responses. All in all, satisfaction was derived from the model's performance.

4.2 Python Environment

Anaconda is chosen as the preferred Python distribution for several reasons. First and foremost, the use of Anaconda is explicitly recommended in MetaAI's installation instructions, ensuring that all required dependencies are installed. Additionally, Anaconda addresses a limitation in the robot environment, allowing the use of Python 3.11.4 by creating a self-contained distribution environment, overcoming the restriction on installing versions beyond 3.9 directly on the robot. This enables the utilization of the latest features and improvements. Another advantage of Anaconda is its user-friendly package manager, simplifying the installation of individual libraries such as PyTorch, a central component needed to run the LLM locally.

4.3 Source Code Editor

As for the source code editor, Visual Studio Code is selected for several reasons. In a previous Python course, various code editors, including Sublime Text 3 and PyCharm, were tried to find the optimal development environment. Visual Studio Code emerged as the best and most intuitive choice, influenced by extensive experience gained during the Python course. The decision is further supported by the editor's pre-installation on the robot, saving time that would be spent trying out a new editor. In summary, Visual Studio Code is chosen as the code editor for its user-friendly nature, existing experience, and availability on the robot.

5. Use of Large Language Model

5.1 QTrobot Local Installation

On the QTrobot, Python 3.9 was the only installed version. To accommodate the requirements of the project, the installation of Anaconda was initiated, which included Python 3.11.5. Subsequently, the installation of PyTorch along with CUDA support was carried out. PyTorch, an open-source Python library, is instrumental in the execution of Deep Learning tasks, spanning applications like image classification, natural language processing (NLP), and facial recognition. CUDA, on the other hand, facilitates parallel GPU computation on Nvidia GPUs while enhancing overall execution speed. (Nvidia, n.d.)

After gaining access to Llama 2, the download of the 13B chat model was initiated. The email received from MetaAI has been attached in Appendix 2. Given that the robot had internet connectivity solely through a phone hotspot, the download process proved to be time-consuming, consuming a total of 5 hours to fetch the approximately 25 GB model. You can find a comprehensive installation guide in Appendix 3. After running the test scripts, the initial issues surfaced. CUDA had been installed in the CPU variant, and the robot's integrated CPU lacked the necessary power to run the language model effectively. The next step involved attempting to install the GPU-compatible CUDA version. To gauge the compatibility of this CUDA version with the graphics cards, a script was employed, revealing that CUDA exclusively supports Nvidia graphics cards. An alternative, ROCm, exists but is primarily designed for AMD graphics cards. (AMD, n.d.)

Aside from the absence of a viable Intel alternative, the built-in processor graphics card, the Iris Plus Graphics 655, would have proven inadequate for running even the smallest model with 7 billion parameters. In an attempt to explore all possibilities, the 7B variant was downloaded to test whether the CPU might be sufficiently potent for this model. Unfortunately, it fell short of the necessary requirements. The 7B model, for instance, mandates at least 10 GB of VRAM, and the 13B variant demands an Nvidia RTX 3090 or 4090, both of which come equipped with 24 GB of VRAM. As per the benchmark test results, the quality of the model's responses is notably influenced by its size. Thus, opting for a smaller LLM wouldn't have been a sensible choice. Consequently, consultation with the professor was sought, and a quest to discover alternative methods for local installation was embarked upon.

5.2 Alternatives for Local Installation

5.2.1 Virtual Machine

As an alternative to implementing the LLM directly on the robot, an option is to set up a local installation on a virtual machine (VM). The installation process mirrors that of the robot, but with the added task of developing a custom API for communication between the LLM and the robot. This approach offers several advantages, with one key benefit being the preservation of full control over your data. Furthermore, for scenarios involving frequent LLM usage, the ongoing costs post-hardware purchase tend to be more economical when compared to other alternatives. Running the LLM effectively on the VM hinges significantly on the graphics card's VRAM. To this end, specific requirements for the virtual machine come into play:

Model Size	*VRAM*	*Example GPU*	*Costs*
7B [40]	min. 10GB	Nvidia RTX 3080	699 Dollar
13B [40]	24 GB	Nvidia RTX 3090 / 4090	1763 / 1599 Dollar

Table 5: VM Requirements

Source: Own representation

5.2.2 DeepInfra

DeepInfra offers a straightforward solution for using Llama 2 via a simple API call. This can be seamlessly integrated into your Python scripts. A notable feature of DeepInfra's pricing structure is that it's based on the number of tokens used, rather than the computing power, which allows you to harness the full potential of the top-tier Llama 2 model with its 70 billion parameters. To illustrate its effectiveness, a series of tests was conducted using various prompts, and the results were satisfactory. It's important to note that, while the API doesn't facilitate the addition of a pre-prompt, prompts can be marked as "system," enabling them to serve a similar function. To provide a clearer picture of the Llama 2's capabilities, below are two prompts accompanied by the model's responses. The script used for this experimentation is available in Appendix 4.

Prompt:

You are a nutrition coach and only answer my questions if they relate to nutrition. If they relate to something else, ignore them. How big is Russia?

Answer:

I'm happy to help with any nutrition-related questions you have! However, the question "How big is Russia?" is not related to nutrition, so I won't be able to provide an answer. Is there anything else you would like to know or discuss related to nutrition?

Prompt:

You are a nutrition coach and only answer my questions if they relate to nutrition. If they relate to something else, ignore them. What should I eat this lunch? Allergies= none, Preferences = Pork, lots of Protein

Answer:

Great! Given your preferences and allergies, here's a suggestion for a high-protein pork lunch option:
Grilled Pork Chop with Roasted Vegetables
** Grill a 6 oz pork chop and serve it with a variety of roasted vegetables such as broccoli, Brussels sprouts, and sweet potatoes. This will provide you with approximately 40 grams of protein.*
** Side salad with mixed greens, cherry tomatoes, cucumber, and a vinaigrette dressing made with olive oil and balsamic vinegar.*
** Add a serving of quinoa or brown rice to increase the fiber and carbohydrate content of the meal.*
Total protein content: approximately 50 grams
This meal is not only high in protein but also rich in vitamins, minerals, and healthy fats from the vegetables and whole grains. The pork chop provides lean protein, while the quinoa or brown rice adds complex carbohydrates for sustained energy. Enjoy!

DeepInfra offers competitive pricing at just $0.001 per 1,000 tokens for the largest Llama 2 model with 70 billion parameters. This pricing structure makes it one of the most cost-effective options available, and a more affordable provider couldn't be found during the research. The platform's ease of use is another significant advantage. Implementing the API in your scripts is straightforward, and responses consistently arrived within a reasonable timeframe, usually taking just two to five seconds. (DeepInfra, n.d.)

5.2.3 Amazon AWS

Amazon AWS provides an alternative option for hosting various models of Llama 2. They offer the AWS Sagemaker platform to facilitate the hosting of Machine Learning Models. With this service, you can set up endpoints for the API and create an API gateway for your specific requirements. In the tutorial referred to, the 7 billion parameter model of Llama 2 was used, but it's worth noting that the larger 70 billion parameter model could also be hosted. However, it's essential to consider the cost associated with this service.
According to the creator of the tutorial and user comments, the monthly cost for hosting the 7 billion parameter model on AWS typically ranges from $600 to $8,000. The exact pricing varies based on factors such as the server specifications used for hosting, the number of requests made to the model, and the tokens generated during processing. It's important to keep in mind that this option tends to be more expensive when compared to other hosting approaches, particularly considering that it covers only the 7 billion parameter model. (Woyera, 2023)

5.2.4 Huggingface

Huggingface offers the flexibility to host various LLM models. To get started, you need to request access to the relevant model repository, where you can choose a name for your endpoint. Then, it's important to select an appropriate instance type that aligns with your specific needs. The cost structure for these services is outlined in the table below.

Provider	Instance Size	Hourly Rate	GPUs	Memory	Architecture
AWS	small	$0.60	1	14GB	NVIDIA T4
AWS	medium	$1.30	1	24GB	NVIDIA A10G
AWS	large	$4.50	4	56GB	NVIDIA T4

Table 6: Pricing Huggingface

Source: https://huggingface.co/docs/inference-endpoints/pricing

These instances cater to different needs, depending on the Llama 2 model you're working with. The 7 billion parameter model can run efficiently on the weakest GPU available, while the 13 billion parameter variant requires a more powerful Nvidia A10G graphics card. However, it's worth noting that Huggingface charges based on hourly usage, not the number of prompt tokens. This might pose a disadvantage for certain projects where only a limited number of prompts are executed per hour, potentially resulting in higher costs. (Hugging Face, n.d.-b)

The following formula was used to compute the monthly expenses for both the 7B and 13B models:

Monthly Cost = Hourly Rate * (Hours *Replica)

Figure 9: Formula Monthly Costs Huggingface
Source: Own illustration

Llama 2 7B = $0.60/hr * (30*24h * 1 replica) = $432/month

Llama 2 13B = $1.30/hr * (30*24h * 1 replica) = $936/month

5.3 Chosen Alternative

After carefully evaluating all the available options, the choice was to utilize DeepInfra, and there are several compelling reasons behind this decision. Initially, Amazon AWS was ruled out due to its high cost and the unpredictability of expenses. Additionally, while Huggingface offers similar functionalities and benefits to DeepInfra, it comes at a higher cost, which led to exclude it as a choice. In the end, the main decision was between DeepInfra and a virtual machine. A virtual machine incurs substantial costs due to the pricey hardware requirements. However, opting for a VM ensures that no requests or data need to be routed through a third-party provider. Furthermore, it grants independence from an external third-party provider. Conversely, DeepInfra's API is notably cost-effective, offering straightforward integration and application. Notably, the API provides the advantage of utilizing the top LLM, Llama 2 70B. Consequently, the decision has been made to utilize the API, enabling immediate commencement of programming.

6. Architecture

6.1 System architecture

The application is initiated on the robot through an Anaconda Python distribution. The control of all robot functions is orchestrated through the ROS framework. On the computer for the terminal version, everything was installed using pip, and the ROS framework was not utilized. When interacting with the Large Language Model (LLM), the selected LLM is invoked via the DeepInfra API. Due to the online speech recognition recommended by the robot, Google Cloud is utilized for Speech-to-Text tasks. Furthermore, Google Cloud provides sentiment analysis, enabling the identification of whether the responses from the LLM are positive, negative, or neutral, allowing for corresponding adjustments to the robot's behaviour. In the terminal execution, both the Speech-to-Text (STT) and Text-to-Speech (TTS) APIs from Google Cloud are employed. The complete architecture is illustrated in Figure 10 below this section.

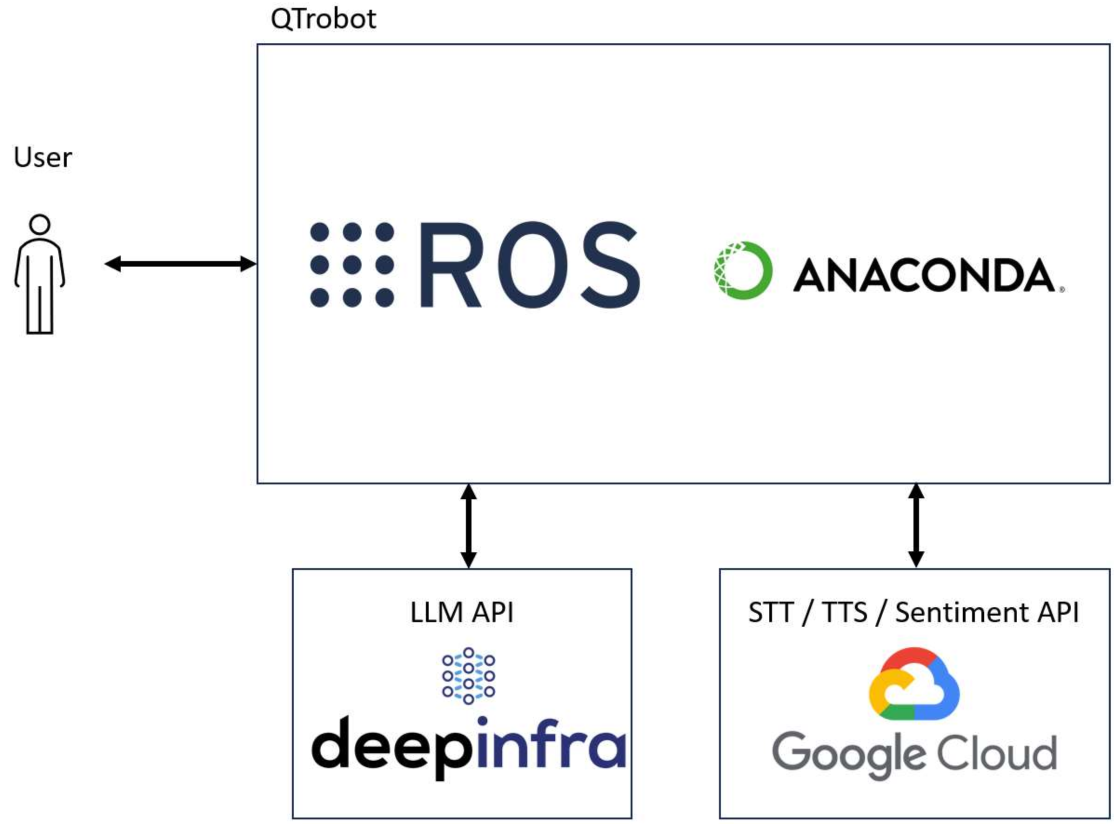


Figure 10: System Architecture
Source: Own illustration

6.2 Logical Architecture

The User Interaction Manager is responsible for the entire interaction with the user and serves as a Platform Abstraction Layer. This means that at the program's start, a parameter is provided to determine whether the program should be executed in a terminal, such as on a computer, or on the robot. This component handles input and output either through the terminal using the TTS API and STT API components or through TTS accompanied by the gestures and facial expressions of the robot. All inputs are then forwarded to the controller. This approach allowed the rest of the program to be developed independently of the underlying platform. The Controller assumes responsibility for the application's logic, encompassing user input processing, orchestration of other components, and the control of registration and login procedures. The LLM API class manages communication with the DeepInfra API, sending requests to the service, processing response data, and returning results to the Controller. In the context of nutritional coaching, direct user inputs serve as prompts. The Control layer is employed to compare the LLM's output with the user's query and subject it to rule-based evaluations. This is accomplished with the assistance of the LLM, ensuring that incorrect or inappropriate responses are prevented from being sent to the user. The UserManager is tasked with managing and storing user data, providing support to the Controller in profile management. All user profiles are stored in a JSON file and can be retrieved as needed.

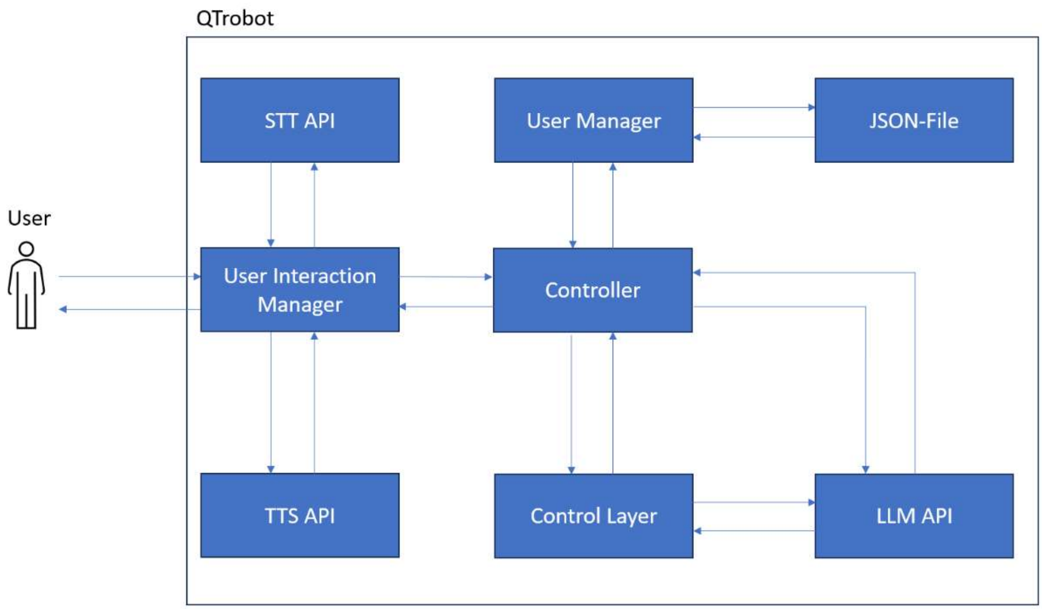


Figure 11: Logical Architecture
Source: Own illustration

7. Implementation

7.1 Flow of Control

The comprehensive control flow of the application is depicted in the following figure. Initially, the robot extends a welcome to the user. Subsequently, the user has the option to indicate whether they already possess a profile and wish to register, or if they intend to establish a new profile. Following this, the user is directed to the main menu, offering a choice between two options. Opting for "Meal Suggestion" provides users with meal recommendations, while selecting "Nutritional Coaching" allows users to seek diet-related guidance from the robot. Ultimately, users can either exit the program or navigate back to the main menu.

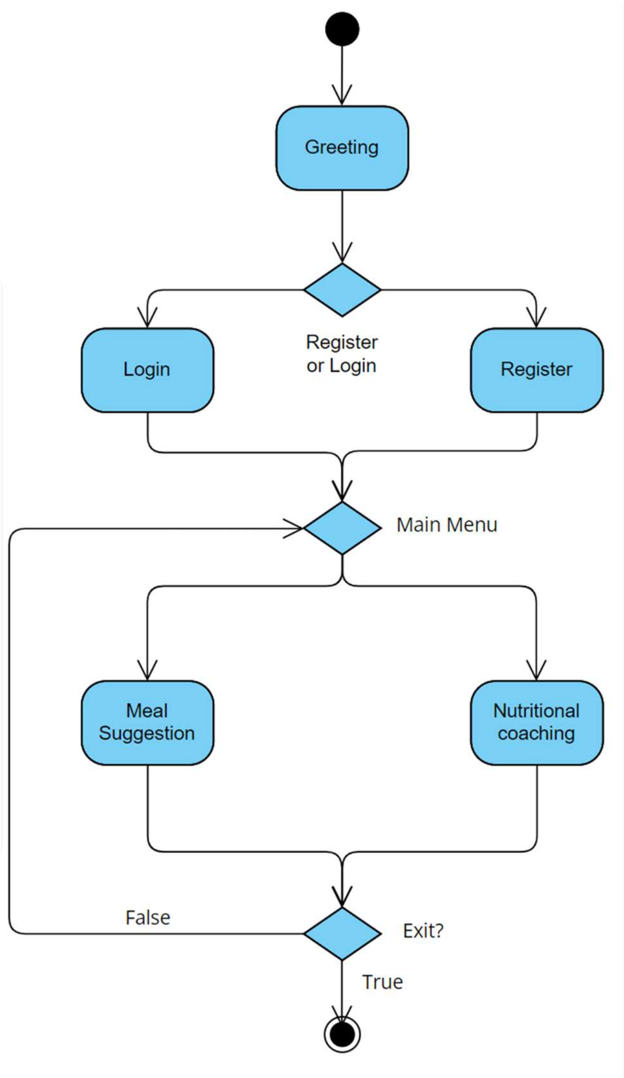


Figure 12: Activity Diagram
Source: Own illustration. Created with visual-paradigm.com

7.2 Class Diagram

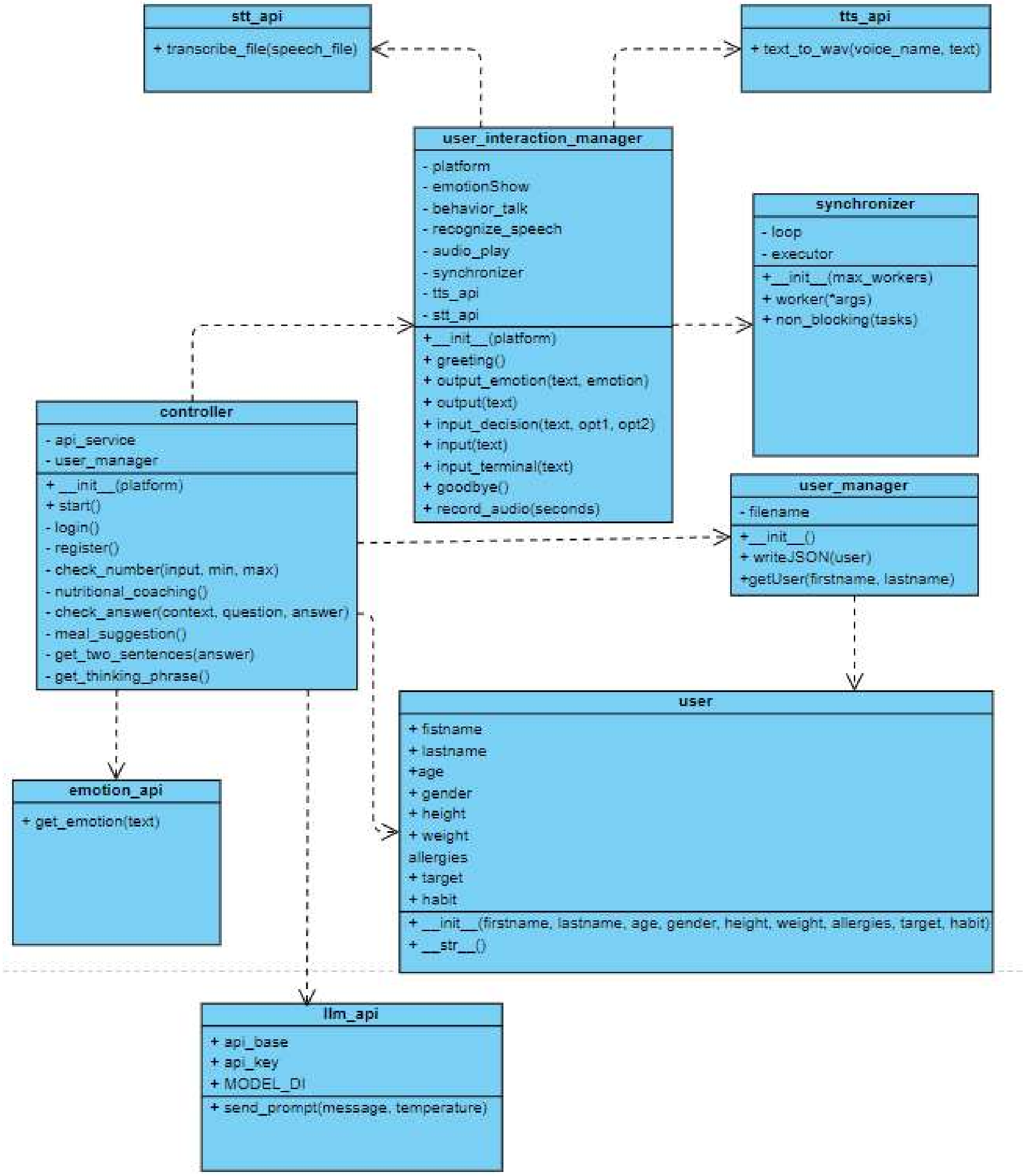


Figure 13: Class Diagram
Source: Own illustration. Created with visual-paradigm.com

7.3 Create User-Profile

The program initiates with the creation of a user profile. The initial concept for creating the user profile should resemble the following. The user is asked for personal information, which is send from the controller to the LLM. The LLM provides responses to the controller's queries. If crucial information is still lacking, the LLM replies with further questions to obtain the necessary details. The controller then conveys these questions to the user. Once all the crucial information is collected from the LLM, it is transformed into a format comprehensible to the controller and sent back to the controller. Subsequently, the controller creates a user object and transmits the information to the UserWriter, where it is stored in a JSON file.

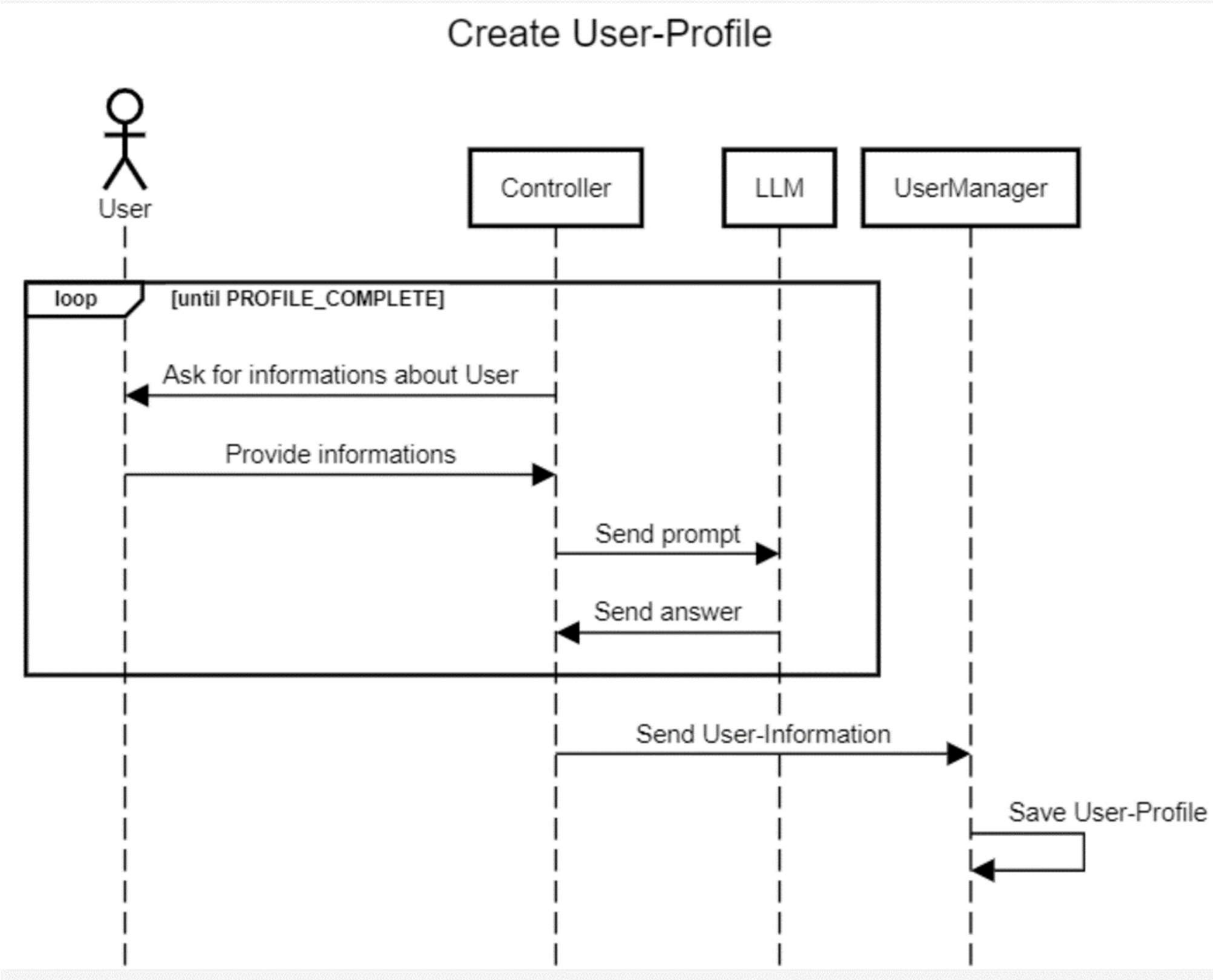


Figure 14: Sequence Diagram Create User-Profile
Source: Own illustration. Created with sequencediagram.org.

Nonetheless, when setting up the user profile, dealing with the LLM posed a challenge. The responses generated by the LLM were inconsistent and exhibited minor variations on each interaction, rendering it unmanageable to reliably extract information from the responses. Despite continual improvements to the system prompt, a consistently functioning version could not be created. To gain a better understanding of the subsequent discussion, let's begin by elucidating the LLM's API call. The function appears as follows.

```
def createProfil(self, history):

    api_request = openai.ChatCompletion.create(
    model="meta-llama/Llama-2-70b-chat-hf",
    messages=history,
    stream=True,
    temperature=0,
    max_token=50
    )

    answer = ""
    for event in api_request:
        if 'content' in event.choices[0]['delta']:
            answer += event.choices[0]["delta"]["content"]

    return answer
```

Figure 15: API Request
Source: Own function

In the API request, the initial step involves selecting the model. Subsequently, the prompts are configured as messages. A more detailed breakdown of how these prompts are structured follows later in the next paragraph. The "temperature" parameter governs the level of randomness and creativity in the LLM's responses, while "max_token" defines the maximum number of tokens generated in a response. Following this, the messages are concatenated into a single string and returned.

Now, let's delve into the specifics of prompts. They fall into three distinct categories: System, Assistant, and User Prompts. The System prompt serves to instruct the LLM on how to engage in the conversation. It can contain instructions or communicate specific rules to guide the LLM's behaviour. Subsequently, the User prompt is appended to this System prompt, representing the user's input. Should you wish to send a secondary User prompt to the LLM while maintaining a contextual link, the prior User input is designated as the User prompt, and the LLM's response becomes the Assistant prompt, which is then placed before the most recent User prompt. This setup enables the LLM to consider preceding questions and answers in its response. The sequence of messages in the second User prompt is structured as follows: System prompt, User prompt, Assistant prompt, User prompt.

A function has been developed to initiate the user by requesting self-introduction and subsequently verifying whether all the required information has been provided. The LLM has been instructed through the system prompt regarding its conduct and the desired format for presenting the extracted information in the response. Various approaches have been experimented with, generating different prompts for each. Now, let's transition into the most significant modifications to the system prompt.

The initial system prompt structure was as follows:

> *From the text received, return the following values without explanation in the format: 'First Name: [First Name], Last Name: [Last Name], Age: [Male or Female], Gender: [Gender], Height: [Height] cm, Weight: [Weight] kg, Allergies: [Allergies], Dietary Target: [Lose weight, maintain weight or gain muscle], Eating Habits: [Eats meat, vegetarian or vegan]'.*

Results were reasonably satisfactory for the initial inquiry in most instances using this system prompt. However, for the subsequent queries, the initial structure was no longer consistently observed, leading to incomplete profiles. At this stage, the issue that persisted until the end became apparent; in some cases, the output was entirely reformatted, disregarding the predefined structure. Furthermore, the LLM exhibited a tendency to answer questions while wholly disregarding the instructions. Subsequently, the system prompt was modified as follows:

> *From the text received, return the following values without explanation in the format: 'First Name: [First Name], Last Name: [Last Name], Age: [Male or Female], Gender: [Gender], Height: [Height] cm, Weight: [Weight] kg, Allergies: [Allergies], Dietary Target: [Lose weight, maintain weight or gain muscle], Eating Habits: [Eats meat, vegetarian or vegan]'. Add all information until the profil is complete. If you do not find an entry in the text, write 'None' in the corresponding field. If no information in the text received can be found at all or the text is a question, answer 'NO_INFORMATION'.*

With this change, the LLM now added the information to the profile in its response after each query, and it no longer answered questions. Observing that as the conversation progressed, the LLM adhered less to the instructions in the system prompt, this was addressed by instructing the LLM to respond with 'NO_INFORMATION' in cases where no information was found in the user's query. If this was included in the response, the response and the user's query were not included as context. Now, the responses improved, but the issue with the different formatting of responses persisted. For this specific prompt, the temperature was adjusted for the first time, configuring it to 0. A temperature of 0 renders the model deterministic, resulting in less creative and more predictable generated responses. This resulted in better responses, but they still lacked consistency. Since only zero-shot prompts had been used until this point, which means prompts without examples, the system prompt was transformed into a few-shot prompt with the addition of examples. To enhance the understandability for the LLM, these examples were separated from the instructions and from each other using two #.

The final prompt appeared as follows:

> *From the text received, return the following values without explanation in the format: 'First Name: [First Name], Last Name: [Last Name], Age: [Male or Female], Gender: [Gender], Height: [Height] cm, Weight: [Weight] kg, Allergies: [Allergies], Dietary Target: [Lose weight, maintain weight or gain muscle], Eating Habits: [Eats meat, vegetarian or vegan]'. Add all information until the profil is complete. If you do not find an entry in the text, write 'None' in the corresponding field. If no information in the text received can be found at all or the text is a question, answer 'NO_INFORMATION'. ## Example 1: Text: My name is Ismael Jaggi and I like to eat meat and I have no allergies. Answer: First Name: Ismael, Last Name: Jaggi, Age: None, Gender: None, Height: None, Weight: None, Allergies: No Allergies, Dietary Target: None, Eating Habits: Eats meat ## Example 2: Text: Explain yourself Answer: NO_INFORMATION*

This System Prompt by far provided the best results. However, problems continued to arise, and the responses were sometimes in the wrong format, or information from the examples was used as if it came from the user During testing, several factors were observed that influenced the LLM's responses. The worst responses were obtained in the morning during the first run. The longer the LLM was used and the more prompts were sent, the better the results became.

In Appendix 5, you will find logs from two attempts at creating a user profile - one successful and one where problems occurred. When generating these logs, 20 test runs were conducted, with 16 being successful and 4 encountering issues. Since this result did not meet the requirements, the decision was made to exclude the LLM from this step and program a traditional profile creation process. In this approach, the system asks the user for all the necessary information individually and creates the profile afterward. For the sake of simplicity, this sequence diagram depicts the acquisition of user information in a single pass. In the actual program, however, the user is queried individually for each piece of information, with each input being verified by the controller in a stepwise manner.

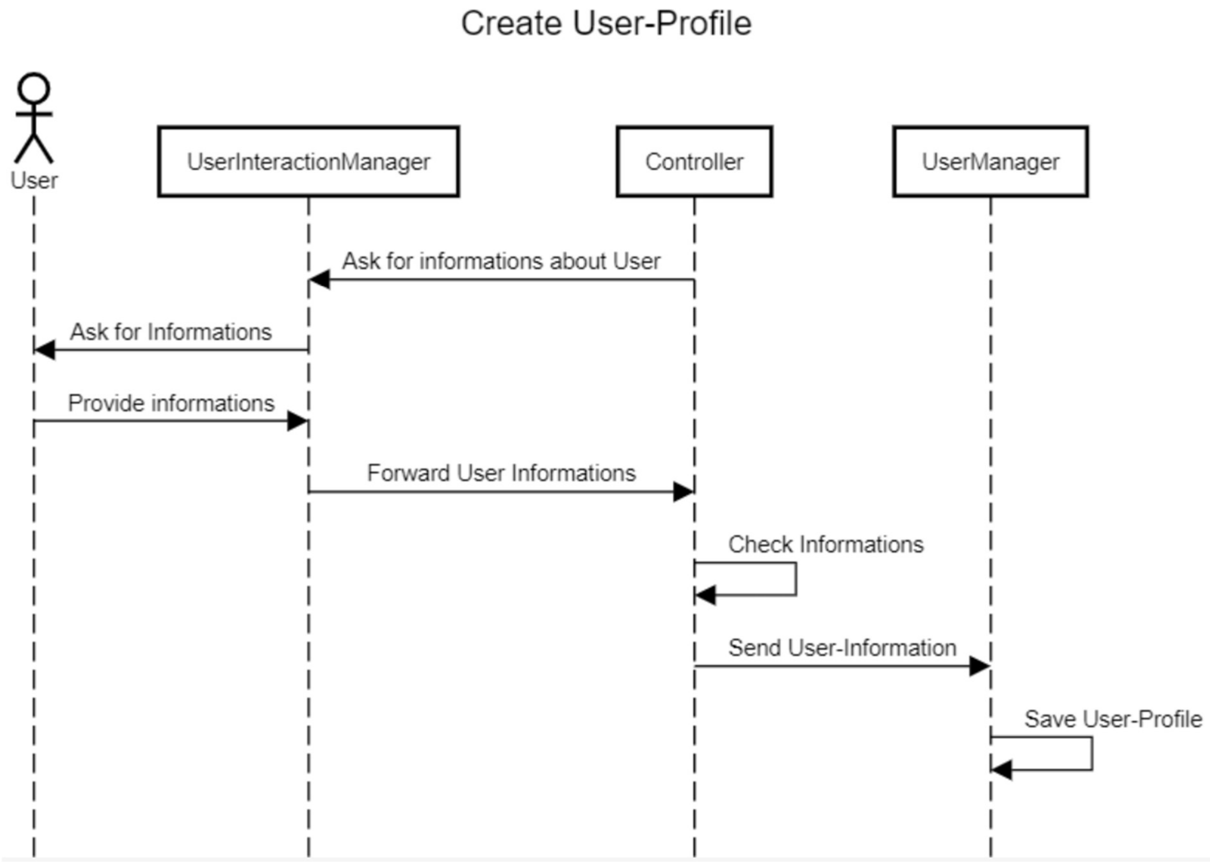


Figure 16: Sequence Diagram Create User-Profile without LLM
Source: Own illustration. Created with sequencediagram.org.

7.4 Login with User-Profile

If the user has already created a profile, they can log in using their first and last names. The controller uses the UserInteractionManager to ask the user first for his first name and then for his last name. Subsequently, this information is sent to the UserManager, which searches for the profile in the JSON file and sends the complete user profile to the controller. If no profile is found, it returns a Nonvalue.

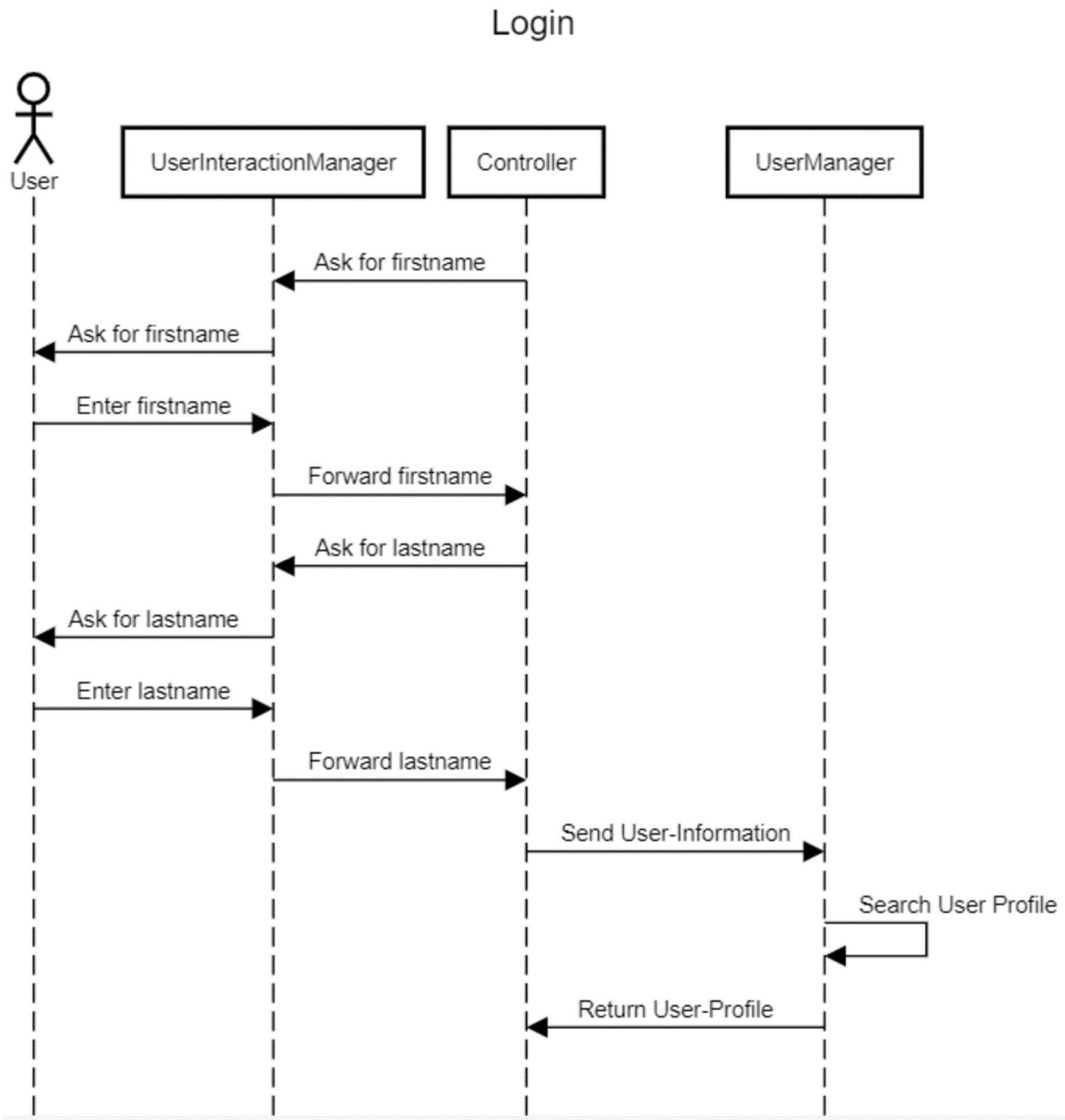


Figure 17: Sequence Diagram Successful Login
Source: Own illustration. Created with sequencediagram.org.

7.5 Meal Suggestions

If the user selects meal suggestions, the controller instructs the LLM to suggest meals for a day. The user's profile is included in the request to enable the generation of individualized responses. The LLM then sends the meal suggestions back to the controller. The controller presents them to the user with the help of the UserInteractionManager and asks if they would like to generate other suggestions. If the user does not want any other suggestions, the function ends. However, if the user desires new suggestions, it starts again, with the difference that the previous LLM response is also included to avoid generating the same suggestions.

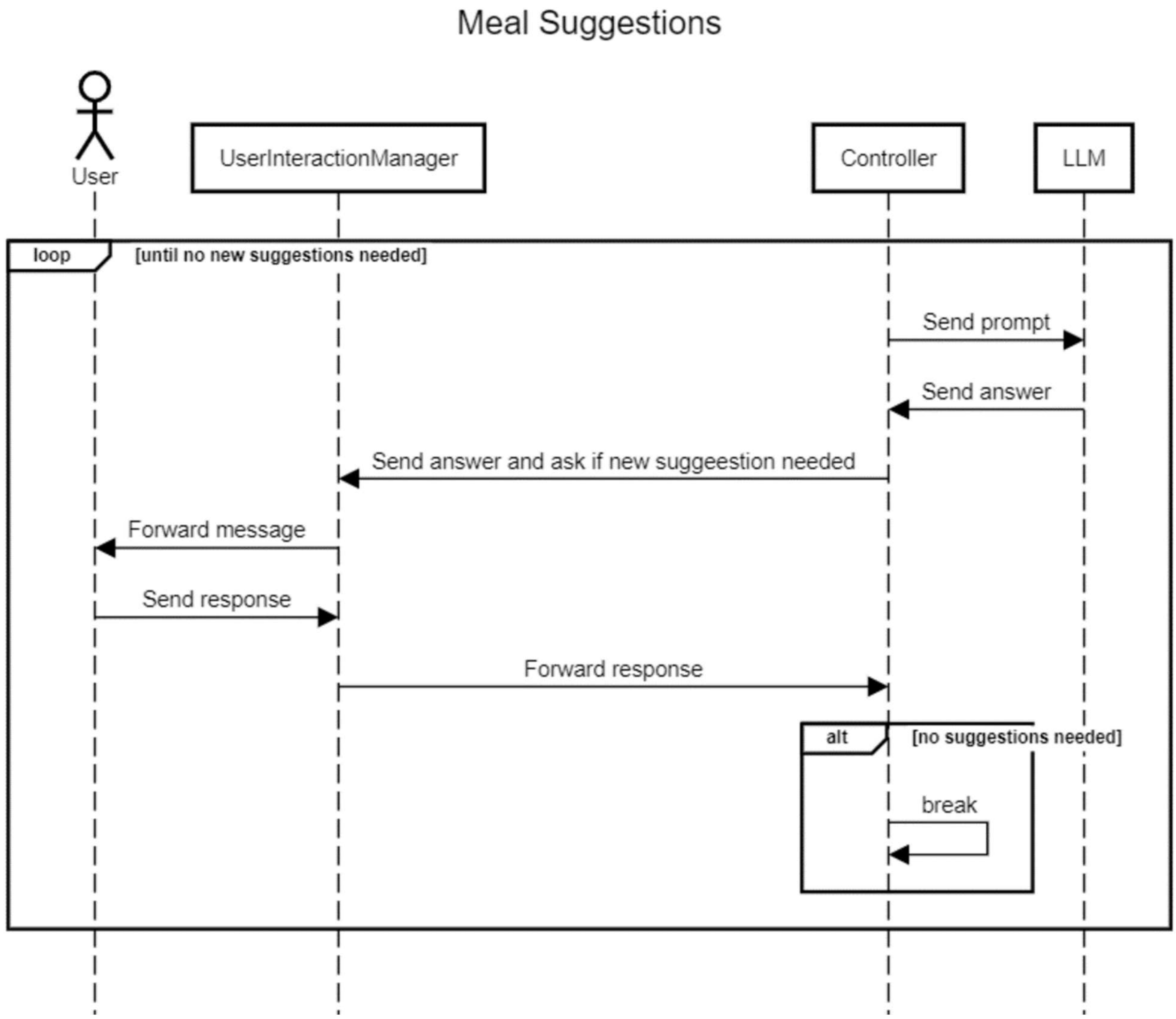


Figure 18: Sequence Diagram Meal Suggestions
Source: Own illustration. Created with sequencediagram.org.

The input prompt used for generating the meals was as follows, with the expression at the end serving the purpose of including the entirety of the user's information in the prompt.

I would like a new recommendation on what to eat for breakfast, lunch and dinner. I only want one recommendation per meal. I want it in 3-4 sentences and without a greeting The recommendation must be adapted to me with my allergies and my eating habits following profile: + self.user.__str__()

7.6 Nutritional Coaching

The Controller, assisted by the User Interaction Manager, solicits questions from the user, who responds with their inquiry. This process repeats until the user responds with "Exit" instead of a question. Subsequently, the Controller dispatches the question, along with its instructions, to the LLM, which returns an answer. The Controller then forwards the LLM's answer, along with context and a set of rules, back to the LLM for verification, ensuring the LLM's response adheres to guidelines and remains non-discriminatory, non-sexist, and non-offensive. This control layer is explored further in Chapter 7.7. In cases where the answer isn't suitable for the question or violates specified criteria, the LLM is directed to generate a new response, with a maximum of three iterations allowed. Once a satisfactory answer is achieved, it is transmitted to the user. If no appropriate response can be generated, the user is informed of this and prompted to pose a new question. The sequence diagram is provided on the next page. The prompt for generating responses had to be continually adjusted and refined throughout the development process. One of the primary challenges was the LLM's tendency to deviate from the guidelines provided in the system prompt, particularly during prolonged conversations. Ultimately, it became apparent that the system prompt had less influence on the overall conversation in Llama 2 than initially assumed. Consequently, including the specific request in each user prompt was chosen to consistently remind the LLM of its task. The user's question was then distinctly separated at the end, which ultimately resolved the issue. The directive consistently included in the later stages was as follows, with the expression at the end serving the purpose of including the entirety of the user's information in the prompt.

You are a nutrition coach and only answer my questions if they are related to nutrition. If they are about something else, ignore them. If its about nutrition give a short and helpful answer. Always answer in max 5 sentences and without a greeting or special characters. Answer as if it were a spoken ongoing conversation. Use this informations to provide personalised answers: + self.user.__str__()

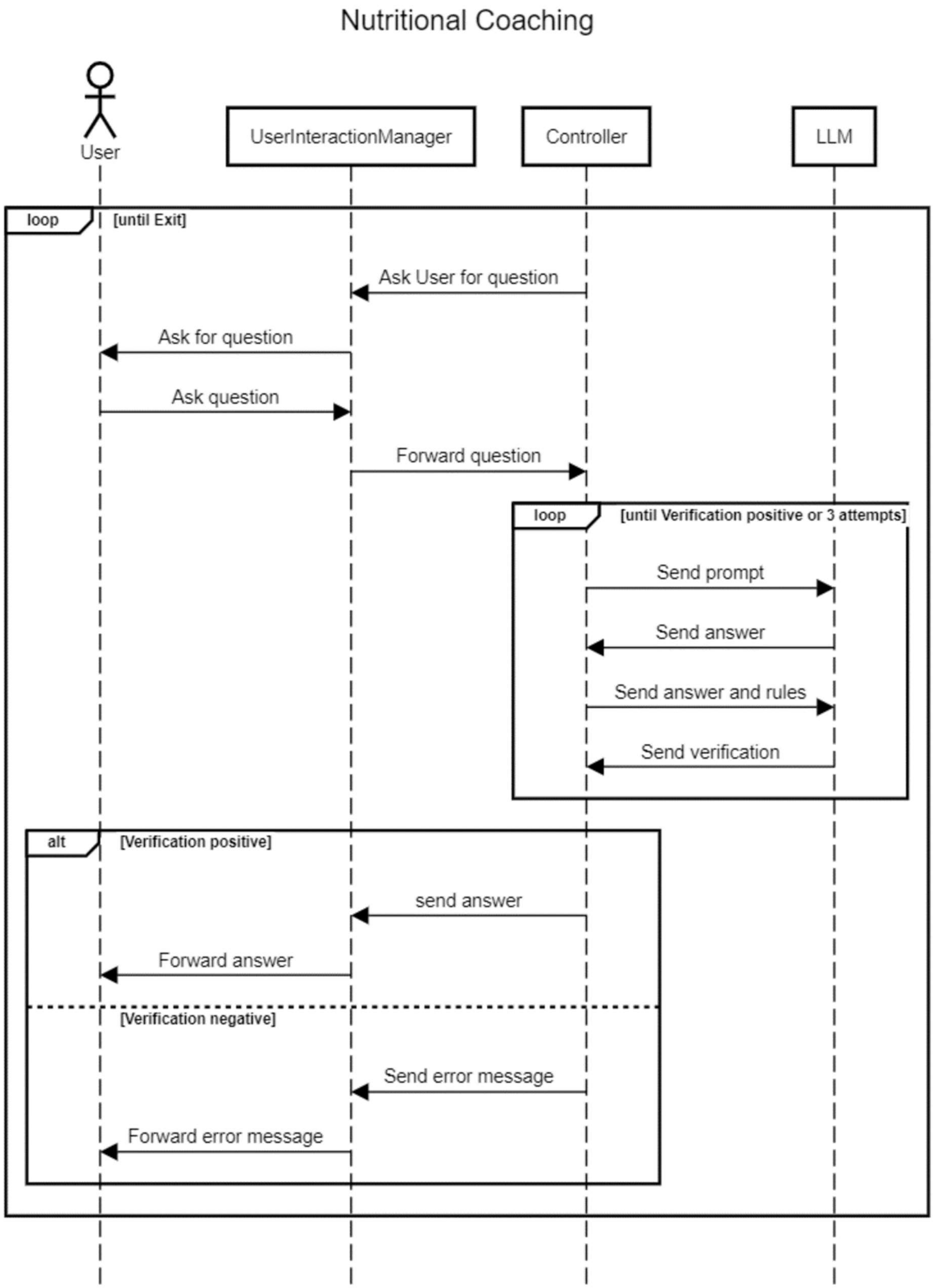


Figure 19: Sequence Diagram Nutritional Coaching
Source: Own illustration. Created with sequencediagram.org.

7.7 Control Layer

To exert some control over the responses generated by the LLM, a control layer was integrated into the Nutritional Coaching function to further scrutinize the LLM's responses. The control layer plays a more significant role in this function than in the Meal Suggestion function, given that direct user inputs are employed here. To evaluate these responses, the context, question, and answer are forwarded back to the LLM for re-examination. This process tests whether the answer aligns with the question and context and ensures that no offensive, racist, or sexist responses are generated. The LLM provides either a True or False response. The system-prompt used is as follows:

> *Your job is to check whether the answer fits the question based on the context. If the answer and the question match, answer: True. If the last question and the answer do not match, answer: False. The answer must not be racist, sexist or offensive. Don't explain the decision, just reply with true or false.*

If the answer doesn't match, the LLM is instructed to generate a new response. This process can occur a maximum of three times, after which the user is informed that generating an answer to their question was not possible. The significant drawback of this control layer is the impact on response speed, effectively doubling the time taken due to the additional query. Throughout the entire testing phase, not once did a False response occur in response to a typical query. Even the attempts at Prompt Injection were consistently thwarted, even before reaching the control layer, thanks to the System Prompt of the Nutritional Coaching function. In order to obtain a negative response and test the functionality of the control layer, the LLM's response within the program had to be modified.

7.8 Speech Recognition

QTrobot provides two options for speech recognition: offline and online speech recognition. In this project, offline speech recognition is employed at multiple stages. During the initial decision-making process, where users choose between logging in or creating a new profile, the words QTrobot recognizes are predefined. The recognized options include "Login," "Register," and "Exit." Following the robot's instructions, a signal tone is played, indicating to the user that they can now speak. If none of the predefined options is recognized by the robot, it displays a confused expression and prompts the user again.

Similarly, during the second decision-making step between meal suggestions and nutritional coaching, a signal tone is played again. The robot pays attention to the words "Meal," "Nutrition," and "Exit." Following the generation of a meal suggestion, the robot inquires whether a new suggestion should be created and comprehends responses such as "Yes" and "No." The built-in microphone in the robot has limitations, requiring users to speak close to the head or loudly for effective recognition.

In the context of nutritional coaching, the initial aim was to utilize the Speech-to-Text (STT) function of QTrobot to allow users to ask questions. However, the accuracy of sentences without predefined options was insufficient and error-prone in the offline variant, hindering efficient use. To address this issue, the intention was to employ the online STT function, leveraging the Google Cloud Speech-to-Text API. Setting up everything on the Google Cloud side proceeded smoothly, and the QTrobot guide was successfully followed without encountering errors. However, upon starting the service, an error message indicated its unavailability.

Consequently, the decision was made to receive user questions via the terminal and temporarily bypass the online STT function.

7.9 Sentiment Analysis

Sentiment Analysis is a component of Google Cloud's Cloud Natural Language API. This API provides the capability to analyze the sentiment in text. In the program, this analysis is utilized to enhance interaction with the robot during Nutritional Coaching. The first two sentences of the robot's generated response are sent to the API. Only the first two sentences are sent because most responses that follow consist of factual information without a specific emotional tone. In response, a score ranging from -1 to 1 and a magnitude are provided. Scores between -1 and -0.25 indicate a predominantly negative sentiment, scores between -0.25 and 0.25 suggest a neutral sentiment, and scores between 0.25 and 1 indicate a predominantly positive sentiment. The cost for analyzing sentences is relatively low, with the first 5000 characters per month being offered free of charge. Under Appendix 9, a tutorial on how to use Google Cloud and the Cloud Natural Language API can be found.

Monthly prices

Price per 1,000-character unit

Feature	First 5K / month	5K+ - 1M	1M+ - 5M	5M+
Entity Analysis	Free	$0.0010	$0.00050	$0.000250
Sentiment Analysis	Free	$0.0010	$0.00050	$0.000250
Syntax Analysis	Free	$0.0005	$0.00025	$0.000125
Entity Sentiment Analysis	Free	$0.0020	$0.00100	$0.000500

Figure 20: Monthly Price NLP API
Source: Screenshot from https://cloud.google.com/natural-language/pricing

7.10 Platform Abstraction Layer

To make the program independent of the robot, a Platform Abstraction Layer was added, which handles all interactions with the user and controls platform-specific functions, including facial expressions and gestures on the robot. In this program, the UserInteractionManager class takes on the role of the Platform Abstraction Layer. To enhance the user-friendliness of the program for users of the terminal version, integration with Google Cloud Text-to-Speech and Speech-to-Text was implemented. Except for profile creation and login, where Speech-to-Text does not provide significant added value, the entire program can be operated using voice commands. The two APIs were not accounted for in the sequence diagrams in the previous chapters since they are only utilized in the terminal version and not on the robot. Instructions for installing the required CLI and packages can be found in Appendix 9 for Windows. The use of these two APIs for personal use with relatively few requests is quite cheap, as shown in the figures below.

Feature	Free per month	Price after free usage limit is reached
Neural2 voices	0 to 1 million bytes	US$0.000016 per byte (US$16 per 1 million bytes)

Figure 21: Monthly Price TTS API
Source: Screenshot from https://cloud.google.com/text-to-speech/pricing

Category	Models	Pricing	
		0-60 Minutes/Month	Over 60 Minutes/Month
Speech Recognition (without data logging - default)	Standard[1]	Free	$0.024 / minute **
	Medical[2]	Free	$0.078 / minute **

Figure 22: Monthly Price STT API
Source: Screenshot from https://cloud.google.com/speech-to-text/pricing

A significant advantage of such a Platform Abstraction Layer is that everything behind this layer can be used on any platform without requiring code modifications. Upon program initiation, a parameter is passed, determining the platform. The two options are "Terminal" or "Robot". If no platform is specified, the terminal version is executed. Furthermore, not all packages need to be installed on every platform; for instance, the rospy package must only be installed on the robot.

8. Testing and Results

8.1 Test-Scenarios

Profile Creation and Login

Description	This test scenario evaluates the processes of profile creation and login within the application. It assesses the application's ability to capture essential user information and its proficiency in verifying user identity. These capabilities are essential for tailoring the subsequent outputs to the user.
Steps	1. Launch the application. 2. Choose the "Create Profile" option. 3. Input first name, last name, age, gender, height, weight, allergies, dietary goal, and eating habits. 4. Confirm the input. 5. Close the application. 6. Restart the application. 7. Select the "Login" option. 8. Enter the first name and last name.
Preconditions	- QTrobot running
Expected Outcome	- The profile creation and login process are completed successfully, without error messages.

Table 7: Test Scenario Register / Login
Source: Own representation

Generating Meal Suggestions

Description	This scenario concentrates on the application's capacity to generate meal recommendations based on user profiles. It investigates whether the recommendations align with user dietary goals and preferences, including allergies.
Steps	1. Log in to your profile using first and last name. 2. Choose the "Meal Suggestions" option. 3. Generate recommendations for breakfast, lunch, and dinner. 4. Review the generated recommendations.
Preconditions	- You are logged into your profile. - Your profile includes information on allergies, dietary goals, and dietary habits.
Expected Outcome	- The application generates appropriate meal recommendations. - The recommendations take into account allergies, dietary goals and eating habits. - You can request new recommendations if needed.

Table 8: Test Scenario Meal Suggestions
Source: Own representation

Nutritional Coaching – Basic Questions

Description	This test case focuses on the application's capacity to provide nutrition-related information in response to basic user questions. It verifies the application's knowledge base and its ability to offer simple, yet scientifically correct responses.
Steps	1. Log in to your profile. 2. Select the "Nutritional Coaching" option. 3. Ask basic nutrition-related questions, e.g., "How many meals per day are recommended?"
Preconditions	- You are logged into your profile.
Expected Outcome	- The application provides answers to your questions based on your profile and offers helpful nutrition information.

Table 9: Test Scenario Basic Questions
Source: Own representation

Nutritional Coaching – Advanced Questions

Description	This scenario extends the evaluation to the application's competence in addressing advanced and more intricate nutritional questions posed by users. It examines the depth of the application's knowledge and its ability to provide comprehensive and scientifically valid responses.
Steps	1. Log in to your profile. 2. Choose the "Nutritional Coaching" option. 3. Ask advanced nutrition-related questions, e.g., "How can I increase my protein intake?"
Preconditions	- You are logged into your profile.
Expected Outcome	- The application answers your advanced questions based on your profile and provides detailed nutrition advice.

Table 10: Test Scenario Advanced Questions
Source: Own representation

Nutritional Coaching – Contextual Questions

Description	In this test case, the application's contextual awareness is tested. Users ask sequential questions, building on previous responses. The evaluation ensures that the application maintains context and provides coherent answers in an ongoing conversation.
Steps	1. Log in to your profile. 2. Select the "Nutritional Coaching" option. 3. Ask a question and remember the answer. 4. Pose a new question based on the previous answer, e.g., "How can I implement this information in my diet?"
Preconditions	- You are logged into your profile. - You have previously asked a question in nutritional coaching.
Expected Outcome	- The application recognizes the context of your previous question and answers the new question based on the previous response. The dialogue is consistent and meaningful.

Table 11: Test Scenario Contextual Questions
Source: Own representation

8.2 Test Results

The initial round of testing was conducted during the fifth sprint, with all inputs being made via the terminal. In the following, the identified issues will be addressed. Each problem was addressed as soon as it was identified. Subsequently, each test scenario was re-run three more times once no further issues arose.

In the first test scenario, there was an issue where the responses were initially stored in the incorrect format. However, this problem was relatively straightforward to resolve and was quickly addressed. Regarding meal suggestions, there was an initial issue where the LLM did not strictly adhere to user preferences. For instance, it occasionally suggested non-vegan meals to users with vegan preferences. This misunderstanding stemmed from how the LLM interpreted user data as suggestions rather than rules to be followed for every meal. This issue was resolved by rephrasing:

> *I would like a new recommendation on what to eat for breakfast, lunch and dinner. I only want one recommendation per meal. I want it in 3-4 sentences and without a greeting. Here is my profile: + self.user.__str__()*

To the following prompt:

> *I would like a new recommendation on what to eat for breakfast, lunch and dinner. I only want one recommendation per meal. I want it in 3-4 sentences and without a greeting The recommendation must be adapted to me with my allergies and my eating habits following profile: + self.user.__str__()*

All subsequent test scenarios proceeded smoothly, yielding the expected results and requiring no further adjustments. The logs of the final tests are stored in Appendix 7.

The second round of testing took place at the beginning of the seventh sprint. All core functionalities were fully implemented, and all test scenarios were executed without encountering any errors. However, two observations were notable throughout almost all tests. Firstly, the robot's mouth did not move during speech, and secondly, users were not notified after he asked a question during nutritional coaching. Consequently, the robot's mouth movements were introduced, utilizing an appropriate service available on the robot. Additionally, a function was implemented to randomly output a sentence in the style of "Let me think about that" to reduce waiting times during answer generation.

The third and final round of manual testing took place during the eighth sprint, after completing the implementation of all functionalities but before the test with external users. The test was conducted for the terminal version, as the implementation on the robot was completed after the second testing round, and tests with external users were conducted with the terminal version.

All test scenarios were successfully executed, yielding the expected results. However, several negative aspects were observed. Firstly, every spoken sentence was sent to the TTS API, incurring time delays. Additionally, some dialog segments remained the same with each execution. To address this, recurrent sentences were stored and played back, reducing API usage, leading to shorter wait times and lower costs. Now, only sentences that vary with each execution are sent to the API. Another concern was the speech recognition during nutritional coaching. The program initiated speech recognition for the next question immediately after answering one. This could overwhelm the user, as they might not have thought of their next question yet. A feature was added requiring the user to confirm with Enter before proceeding to the next question. The final issue pertained to the speech recognition of the word "Login." The Google Cloud Speech-to-Text API occasionally struggled to comprehend the term "Login." As a resolution, it was decided to replace "Login" with "Sign in." Following these adjustments, all test scenarios were re-executed, and upon successful completion of the tests, the implementation was finalized.

8.3 User Testing

The user testing phase took place at the end of the eighth sprint. All participants were employees of Hes-so Valais-Wallis. Before testing, each participant filled out a pre-test questionnaire to gather insights into their expectations and experiences. All questionnaire results were anonymously stored to ensure answers could not be traced back to specific individuals. The questionnaire is available in Appendix 12, and corresponding responses are found in Appendix 13. Participants' backgrounds varied widely, from those unfamiliar with humanoid robots to individuals who had previously developed programs for such robots. Expectations predominantly centred around user-friendliness. Opinions diverged on whether such robots could be effectively utilized, with some fully convinced and others regarding the robots as expensive toys. A noteworthy finding was participants' willingness to disclose personal information such as weight and height if it could enhance response generation. Several participants expressed reluctance to provide their full names. The pre-test questionnaire underscored the importance of voice control and maintaining a smooth program flow for such applications.

The actual testing proceeded largely without issues, conducted with the terminal version for all participants. Personal information could be entirely fictitious, and all created profiles were deleted post-testing. Minor issues were noted with the STT API, particularly with two female participants who had strong accents (non-native) and softer voices, resulting in occasional misinterpretations. Speaking louder or repeating the sentence usually corrected these issues on the second attempt. Some minor details that arose during testing were quickly addressed. Participants required minimal instructions as the program was largely self-explanatory. Each participant spent 10-15 minutes testing the program, allowing for in-depth examination of profile creation and the two main functions.

Post-testing, participants were asked to complete a second questionnaire sharing their impressions and experiences with the application. This questionnaire and responses are documented in Appendix 12 and 13. Overall, participants rated their experience with the program as good to very good. The implemented functions were universally perceived as helpful, with positive comments on speech processing and response quality. Negative remarks have been provided regarding the time taken for response generation, perceived as too long by one participant. All participants reported feeling secure and comfortable throughout the testing, although concerns were voiced about the use of user data such as name and voice. Responses to the simplicity of communication varied, attributable to challenges in recognizing soft-spoken female voices. In summary, the user tests were considered highly successful, and all participants, except one who desired more time with the program, would recommend it.

After analysing the tests, several observations can be made. The program has achieved a very good level and, in some aspects, even exceeded expectations. Nevertheless, there are opportunities for future improvements to further enhance the user experience.

Speech Recognition: Firstly, considering the test of various Speech-to-Text and Text-to-Speech APIs is recommended. In this instance, Google Cloud was utilized as it was mandated by the robot. However, for the Terminal Version, integrating other APIs could potentially address issues related to understanding soft-spoken female voices with strong accents.

Waiting Times: Some test participants expressed concerns about waiting times. This issue is more challenging to resolve. The most significant delays occur during the generation of responses with the Large Language Model (LLM). Streaming responses, receiving them word by word, and sending the initial part to the TTS API, for example, could be explored as a potential solution. However, this approach is most relevant for lengthy responses, and the LLM has been instructed to keep responses concise. During the tests, a Wi-Fi connection with 100 Mbps download and 110 Mbps upload was employed. Using a faster connection might slightly reduce waiting times.

Privacy Concerns: Given concerns regarding the use of full names, considering the substitution of first and last names with a username, as they do not influence the responses, could address these concerns.

Option to Skip Long Responses: Based on feedback, the implementation of a feature to skip long responses should be considered. Additionally, a feature could be implemented to allow the user to interrupt an ongoing action without terminating the entire program, for instance, in cases where an API request takes too long.

9. Project Management

9.1 Scrum

For this project, the decision was made to utilize Scrum as the project management framework. Several factors influenced this decision. Firstly, there was prior experience with Scrum, having been employed in several previous projects. This familiarity spared the need to learn a new project management methodology. The bachelor's thesis description included the task to create a product backlog. Since a backlog needed to be created anyway, the generated user stories could be used directly to structure sprints. Moreover, Scrum was found to be much more adaptable than, for instance, the waterfall model. However, a modified version of Scrum was employed for this solo project. Daily meetings and retrospectives were omitted, as these aspects of Scrum lacked practicality when working alone. While each day was still planned, and reflections were made on ways to better organize the next sprint, these aspects were tailored for individual use. The product backlog and sprint plannings are located in Appendix 16 and 17.

9.2 Overview of Sprints

Each sprint will be briefly addressed individually, highlighting the progress and challenges encountered.

9.2.1 Sprint 0

In the first sprint, the sprint goal was to understand the QTrobot and its functionalities. The QTrobot was used for the first time, and the LuxAI tutorial was completed. Additionally, documents were started, and the first part of the state of the art was written.

9.2.2 Sprint 1

Sprint 1 aimed to: Get to know and compare open source LLM. Firstly, more robots were added to the state of the art, and research on open-source LLM was conducted. Subsequently, the state of the art of open-source LLM was written, and a comparison between them began. Also, considerations about how the program might look later were started. Initial considerations can be found in Appendix 1.

9.2.3 Sprint 2

Sprint 2 was chosen with the goal: Start using the LLM with the robot. First, PaLM 2 was added to the comparison of LLMs, as it had recently been introduced by Google and provided a better comparison to GPT-4. Then, attempts were made to locally install Llama 2 on the robot. This proved to be the first significant challenge. Due to the required hardware, it was not feasible to run the LLM locally on the robot. Consequently, alternatives to local installation were researched, and DeepInfra was also tested directly with the small script in Appendix 4.

9.2.4 Sprint 3

Sprint 3 had the sprint goal: Start with the development. The decision was made to use DeepInfra, which provided the option to use Llama 2 with 70B parameters, so this version was included in the comparison as well. Following that, an explanation of what an LLM is was written, and, as mentioned at the beginning of the document, the texts from ChatGPT were rewritten into better English. Now, the programming phase began. The initial script was created to obtain meal suggestions from the LLM that consider the user's allergies. Additionally, an attempt was made to create a user using the LLM, which, as described in the implementation section, proved to be too unpredictable. Towards the end of the sprint, a discussion took place with Gaetano Manzo, a researcher at Hes-so, who introduced several other LLMs and offered assistance if there were questions about using LLMs.

9.2.5 Sprint 4

In Sprint 4, the focus was on further expanding the program. The sprint goal was to add further main functions. And that's precisely what was done; a login function was created, and testing of profile creation with the LLM continued. When it was realized that it was not feasible to build a profile with this LLM, a switch to traditional profile creation using the terminal was made. Following that, ideas for creating a control layer for the LLM were presented, and an attempt was made to implement a possible solution. Towards the end of the sprint, the decision was made to implement the control layer as described in the implementation section.

9.2.6 Sprint 5

In Sprint 5, the goal was to obtain a functional first version of the program. The final version of the meal suggestions function was incorporated into the program, and the nutrition coaching function was created. Subsequently, the created control layer was added to the nutrition coaching. With all main functions now operational, test scenarios were devised and executed. Identified errors were addressed, and simultaneously, work on writing the report continued. Additionally, research on Prompt Engineering was conducted, and the corresponding chapter was written in the report.

9.2.7 Sprint 6

Sprint 6 had the sprint goal: Add further functionalities to the robot. Work was done on the implementation of a Platform Abstraction Layer to enable program execution in both the terminal and on the robot. Additionally, exploration of various emotion APIs took place. Furthermore, the control layer was documented, and the implementation chapter was updated and expanded. Efforts were made to integrate offline and online speech recognition, add gestures to enhance interaction, and create a tutorial with corresponding documentation. Online Speech Recognition could not be added in this sprint.

9.2.8 Sprint 7

In Sprint 7, the goal was to improve the overall user experience. To achieve this, mouth movement for the robot during speech was implemented, and filler phrases were added during long pauses. Additionally, questionnaires for testing with external participants were created, and a user manual was developed. There was also an aim to activate online speech recognition; however, the Raspberry Pi in the robot's head failed to start, necessitating its return to Lux AI.

9.2.9 Sprint 8

Sprint 8 marked the final sprint of the project, aiming to conclude the entire endeavour. Given the robot's failure, further enhancements were made to the terminal version. Integration of Google Cloud's STT and TTS APIs was implemented, and the questionnaire was adjusted to accommodate the new circumstances. Subsequently, user tests were conducted, and the results were analysed and documented. Finally, the remaining sections of the documentation were completed, bringing the project to its conclusion.

Conclusion

In this bachelor's thesis, a virtual assistant developed with Python, embodied as a robot, was designed to assist users across various aspects of nutrition using a Large Language Model (LLM). The virtual assistant facilitates tasks such as generating a daily meal plan or answering general nutrition-related questions, thereby enhancing the user's knowledge. To tailor responses to the user, a profile is created, taking into account in each generated response, ensuring personalized and relevant answers. This customization accommodates diverse dietary preferences and restrictions, including veganism, vegetarianism, or specific allergies, aligning with the user's nutritional goals.

The virtual assistant has been designed to function both on a QTrobot and independently. To enhance user experience and naturalize interactions, gestures and facial expressions were incorporated into the robot, adapted to the emotions conveyed in the robot's statements through a Sentiment API. The robot can also be partially voice-controlled and engages in verbal communication with the user. Without the robot, the program can be initiated in a terminal. In the terminal version, two APIs were integrated: a Speech-to-Text API enabling voice commands, and a Text-to-Speech API for program narration. The integration of the two APIs was omitted during login and profile creation, as it would have unnecessarily complicated the process and provided no added value to user-friendliness, potentially even diminishing it.

A comparative analysis of leading open-source LLMs was conducted to select the most suitable option. Considerations included general properties such as price, context length, available languages, combined with benchmark test results, leading to the choice of Llama 2. Due to the robot's limited computational power, insufficient for the execution of LLM locally, alternatives were explored, and DeepInfra, with its Llama 2 70B Chat API, has been selected given its pricing, seamless integration, and the ability to use the largest Llama 2 model.

The robot's implementation was interrupted due to a damage to the power supply, leading to the implementation of a Platform Abstraction Layer to achieve independence from the robot. Subsequently, the robot's implementation was completed. Shortly before user testing, a hardware issue prevented starting the computer in the robot's head, prompting the refinement of the terminal version with STT and TTS APIs for user testing.

Both the terminal and robot versions offer opportunities for expansion and improvement. User testing revealed potential enhancements for the terminal version, including an option to skip long responses or reduce waiting times. Improvement possibilities also exist for the robot version, albeit untested through user tests. Most findings from the terminal version can likely be applied to the robot. Notably, enhancing the robot's ability to recognize complete sentences could be a significant improvement.

In summary, this bachelor's thesis encountered various unexpected surprises and challenges, ranging from the robot's computational limitations for local LLM installation to hardware failures. Nevertheless, solutions or workarounds were devised for all issues. The resulting program evolved differently than initially envisioned but ultimately met all requirements and received positive evaluations from test participants. As LLM, STT, and TTS technologies continue to advance, such programs can further improve, and their potential remains expansive. The insights and experiences gained in this work provide a solid foundation for future projects.

Bibliography

1. Abidi, Y. (2023). Google PaLM 2 vs. OpenAI GPT-4: What's the Difference? *MUO*. https://www.makeuseof.com/google-palm-2-vs-openai-gpt-4/
2. AMD. (n.d.). *Das offene Ökosystem AMD ROCm™*. https://www.amd.com/de/graphics/servers-solutions-rocm
3. Anthropic. (2023, July 11). *Claude 2*. https://www.anthropic.com/index/claude-2
4. Arya, N. (2023a, June 7). *Falcon LLM: The new king of Open-Source LLMs - KDNuggets*. KDnuggets. https://www.kdnuggets.com/2023/06/falcon-llm-new-king-llms.html
5. Arya, N. (2023, May 26). *Introducing MPT-7B: a new Open-Source LLM - KDNuggets*. KDnuggets. https://www.kdnuggets.com/2023/05/introducing-mpt7b-new-opensource-llm.html
6. Client. (2023). Are the latest developments in LLMs a game-changer for social robots? *Furhat Robotics*. https://furhatrobotics.com/blog/are-the-latest-developments-in-llms-a-game-changer-for-social-robots/
7. DeepInfra. (n.d.). *Simple Pricing | Machine Learning Infrastructure | Deep Infra*. https://deepinfra.com/pricing
8. Elasticsearch Inc. (n.d.). *Was ist ein großes Sprachmodell? | Umfassende Anleitung zu LLMs*. Elastic. https://www.elastic.co/de/what-is/large-language-models#key-components-of-large-language-models
9. Facebook Research. (n.d.). *llama/README.md at main · facebookresearch/llama*. GitHub. https://github.com/facebookresearch/llama/blob/main/README.md
10. Falcon LLM. (n.d.). https://falconllm.tii.ae/falcon.html
11. Furhat Robotics. (2023, March 31). *The Furhat Robot | Furhat Robotics*. https://furhatrobotics.com/furhat-robot/
12. Google. (n.d.). *Pricing for Generative AI on Vertex AI | Google Cloud*. Google Cloud. https://cloud.google.com/vertex-ai/docs/generative-ai/pricing?hl=de

13. Hugging Face. (n.d.-a). *meta-llama/Llama-2-13b · Hugging Face*. https://huggingface.co/meta-llama/Llama-2-13b

14. Hugging Face. (n.d.-b). *Deploy LLMs with Hugging Face Inference Endpoints*. https://huggingface.co/blog/inference-endpoints-llm

15. Hugging Face. (n.d.). *Open LLM Leaderboard - a Hugging Face Space by HuggingFaceH4*. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

16. Import, M. (2023, September 7). *Industrieroboter Horst mit integriertem Sprachmodell auf der Motek 2023*. Automationspraxis. Retrieved September 22, 2023, from https://automationspraxis.industrie.de/news/fruitcore-robotics-zeigt-roboter-horst-mit-integriertem-sprachmodell-und-neue-solution-kits/

17. Kaikramer Consulting. (n.d.). *Was sind eigentlich Embeddings?* https://kaikramer-consulting.de/article/was-sind-eigentlich-embeddings/

18. Klofat, A. (n.d.). *Wie funktionieren Transformer? Definition und Praxis*. Informatik Aktuell. https://www.informatik-aktuell.de/betrieb/kuenstliche-intelligenz/wie-funktionieren-transformer-definition-und-praxis.html

19. Luber, S. (2023a, May 12). Was ist ein Large Language Model (LLM)? *CloudComputing-Insider*. https://www.cloudcomputing-insider.de/was-ist-ein-large-language-model-llm-a-9b7bdd0c3766b5a9c0ee1e0c909790a3/

20. Luber, S. (2023b, May 15). Was ist Prompt Engineering? *BigData-Insider*. https://www.bigdata-insider.de/was-ist-prompt-engineering-a-3308973005e97f3927d3f8b63ad3ff2e/

21. LuxAI S.A. (2023, April 21). *QTrobot - expressive humanoid social robot for research and teaching*. https://luxai.com/humanoid-social-robot-for-research-and-teaching/

22. McFadden, C. (2023b, August 16). Meet Pibot: Korea's LLM-powered smart robotic pilot. *Interesting Engineering*. https://interestingengineering.com/innovation/robot-can-fly-an-aircraft

23. Meta AI. (n.d.). *Llama 2 - Meta AI*. https://ai.meta.com/llama/

24. Moore, H., & Moore, H. (2023). LLaMA 2 vs GPT-4: Which Large Language Model is Right for You? - Cloudbooklet. *Cloudbooklet*. https://www.cloudbooklet.com/llama-2-vs-gpt-4/

25. MosaicML. (n.d.-a). *Introducing MPT-7B: A New Standard for Open-Source, Commercially Usable LLMs*. https://www.mosaicml.com/blog/mpt-7b

26. MosaicML. (n.d.). *MPT-30B: Raising the bar for open-source foundation models*. https://www.mosaicml.com/blog/mpt-30b

27. Ng, Y. (2023, August 16). How to get access to Google's PaLM 2 Large Language Model. *Medium*. https://ngyibin.medium.com/how-to-get-access-to-googles-palm-2-large-language-model-21379f27c078

28. Nvidia. (n.d.). *What is PyTorch?* NVIDIA Data Science Glossary. https://www.nvidia.com/en-us/glossary/data-science/pytorch/

29. OpenAI. (n.d.-a). *GPT-4*. https://openai.com/research/gpt-4

30. OpenAI. (n.d.). *How much does GPT-4 cost? | OpenAI Help Center*. https://help.openai.com/en/articles/7127956-how-much-does-gpt-4-cost

31. Papers with Code. (n.d.-a). *Papers with Code - MMLU Benchmark (Multi-task Language Understanding)*. https://paperswithcode.com/sota/multi-task-language-understanding-on-mmlu

32. Papers with Code. (n.d.-b). *Papers with Code - ARC (Challenge) Benchmark (Common Sense Reasoning)*. https://paperswithcode.com/sota/common-sense-reasoning-on-arc-challenge

33. Papers with Code. (n.d.). *Papers with Code - HellaSwag Benchmark (Sentence Completion)*. https://paperswithcode.com/sota/sentence-completion-on-hellaswag

34. Portakal, E. (2023, August 29). Claude 2 vs. Meta AI's Llama 2 (In-Depth Comparison). *Text Cortex*. https://textcortex.com/de/post/claude-2-vs-llama-2

35. Portakal, E. (2023b, September 5). Claude 2 Parameters (Parameter Size, Context Window. . .). *Text Cortex*. https://textcortex.com/de/post/claude-2-parameters

36. Prompting Guide. (n.d.). *Prompt Engineering Guide – Nextra*. https://www.promptingguide.ai/

37. Redillas, M. (2023, September 1). Can you run ChatGPT locally? – GPT 4 on your device. *PC Guide*. https://www.pcguide.com/apps/can-chatgpt-run-locally/

38. Sha, A., & Sha, A. (2023). Google PALM 2 AI model: Everything you need to know. *Beebom*. https://beebom.com/google-palm-2-ai-model/

39. Szöke, D. (2023, March 15). Mittels Sprachmodell Robotik steuern: Google und TU Berlin stellen PaLM-E vor. *Heise Online*. Retrieved September 22, 2023, from https://www.heise.de/news/Mittels-Sprachmodell-Robotik-steuern-Google-und-TU-Berlin-stellen-PaLM-E-vor-7543506.html

40. Timothy, M. (2023). What is Llama 2 and how can you use it? *MUO*. https://www.makeuseof.com/what-is-llama-2-and-how-can-you-use-it/

41. Wieczorek, P. M. (2023, July 27). LLM Benchmarks: How can we say that LLaMa-2 is the best? *Medium*. https://medium.com/@patrykmwieczorek/llm-benchmarks-how-can-we-say-that-llama-2-is-the-best-8f546280c26c

42. Woyera. (2023, September 25). How to Use Llama 2 with an API on AWS to Power Your AI Apps. *Medium*. https://ai.plainenglish.io/how-to-use-llama-2-with-an-api-on-aws-to-power-your-ai-apps-3e5f93314b54

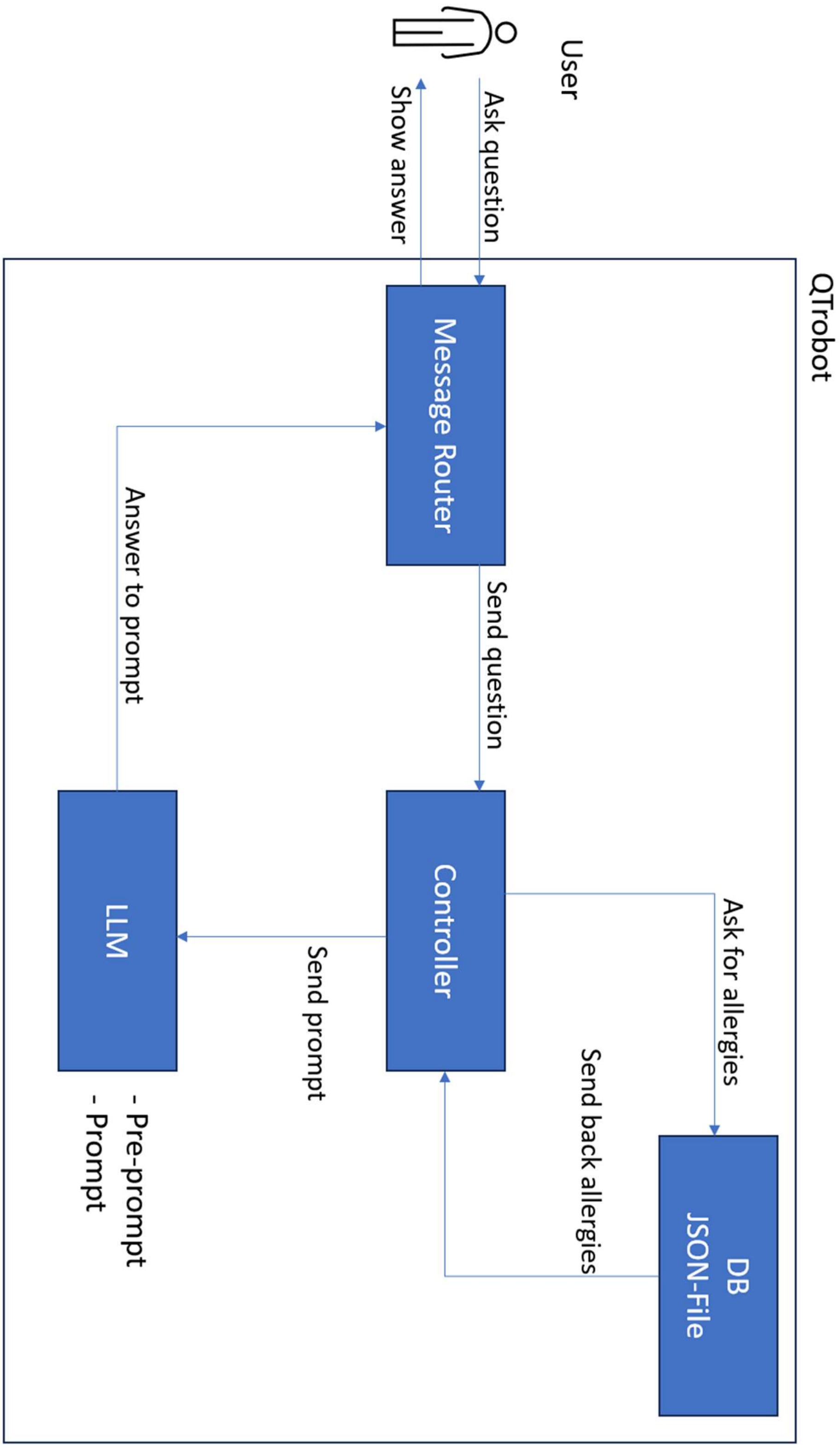
43. Wu, C. (2023, July 20). Extend the context length of Falcon40B to 10k - Chen Wu - Medium. *Medium*. https://medium.com/@chenwuperth/extend-the-context-length-of-falcon40b-to-10k-85d81d32146f

44. Wu, J., Antonova, R., Kan, A., Lepert, M., Zeng, A., Song, S., Bohg, J., Rusinkiewicz, S., & Funkhouser, T. (2023). *TidyBot: Personalized Robot Assistance with Large Language Models*. https://tidybot.cs.princeton.edu/paper.pdf

45. Wuttke, L. (2023). Künstliche Neuronale Netzwerke: Definition, Einführung, Arten und Funktion. *Datasolut GmbH*. https://datasolut.com/neuronale-netzwerke-einfuehrung/

List of Appendices

Appendix 1: First Architectural Consideration



FigureApp 1: Architectural Consideration
Source: Own illustration

Appendix 2: Mail MetaAI

Von: Meta AI <noreply@email.meta.com>
Gesendet: Dienstag, 3. Oktober 2023 13:39
An: Jaggi Ismael <ismael.jaggi@hes-so.ch>
Betreff: Get started with Llama 2

Llama 2 commercial license
You're all set to start building with Llama 2.
The models listed below are now available to you as a commercial license holder. By downloading a model, you are agreeing to the terms and conditions of the license, acceptable use policy and Meta's privacy policy.

Model weights available:

- Llama-2-7b
- Llama-2-7b-chat
- Llama-2-13b
- Llama-2-13b-chat
- Llama-2-70b
- Llama-2-70b-chat

With each model download, you'll receive a copy of the License and Acceptable Use Policy, and can find all other information on the model and code on GitHub.

How to download the models:

1. Visit the Llama repository in GitHub and follow the instructions in the README to run the download.sh script.
2. When asked for your unique custom URL, please insert the following:
   https://download.llamameta.net/*?Policy=eyJTdGF0ZW1lbnQiOlt7InVuaXF1ZV9oYXNoIjoieGltbXZnY2N1ZDJlaHAyN3hzNXhpbWtsIiwiUmVzb3VyY2UiOiJodHRwczpcL1wvZG93bmxvYWQubGxhbWFtZXRhLm5ldFwvKiIsIkNvbmRpdGlvbiI6eyJEYXRlTGVzc1RoYW4iOnsiQVdTOkVwb2NoVGltZSI6MTY5NjQxOTU1OX19fV19&Signature=ateEFxp9bolGt13JvsrGgNleh-k9qcBFdXpyW0zclCWHwQOpoB46ItbvoPFpRZfTCrnwUmSjO5urQ2z9fCA2Gzf4ROR2fGlrLiKRwdv5IwJVVcCiOkmM8m6GMMGPU2smrB2gWDdfGpFZsXKi8TaNIT72mki3e4iwQXRU16qnYSLbMOSgTL9OCqqNGH7vQqqCd7jAXhCcd7krwaPdea2IaEaROjL6IxbgWhUThA-%7Ev3p4RgUlrFvCTVrtra4mb3HNgo-4wpwU8iEP-OtSIPzpJYOv4%7E65glOR96-8ijizrYmaDrrLJc6gV3vHgHJpL%7EfwJKekfK6LVBiMTepbMvQX%7EA__&Key-Pair-Id=K15QRJLYKIFSLZ&Download-Request-ID=1516726972413194
3. Select which model weights to download

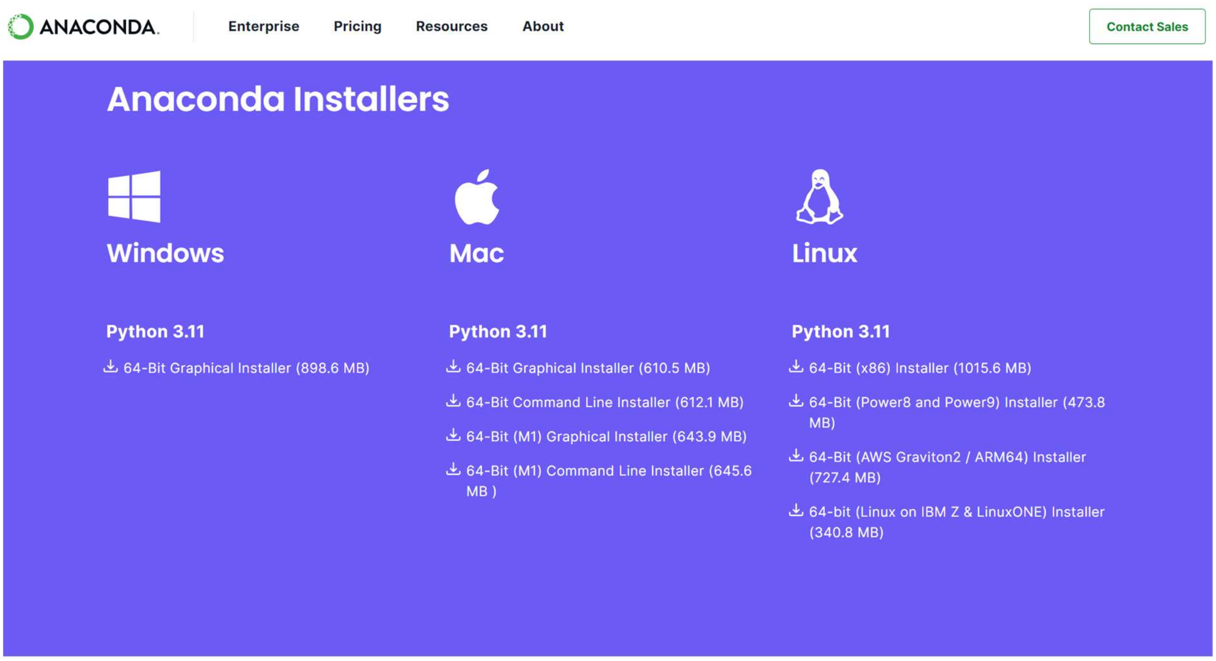
The unique custom URL provided will remain valid for model downloads for 24 hours, and requests can be submitted multiple times.
Now you're ready to start building with Llama 2.

Helpful tips:
Please read the instructions in the GitHub repo and use the provided code examples to understand how to best interact with the models. In particular, for the fine-tuned chat models you must use appropriate formatting and correct system/instruction tokens to get the best results from the model.
You can find additional information about how to responsibly deploy Llama models in our Responsible Use Guide.

FigureApp 2: Mail MetaAI
Source: Email

Appendix 3: Install Tutorial Anaconda / Llama 2

In order to run the different Llama 2 models on the robot, some preparations are necessary. First of all, Anaconda must be installed. To do this, visit the Anaconda website: https://www.anaconda.com/download#downloads



FigureApp 3: Anaconda Download
Source: Screenshot https://www.anaconda.com/download#downloads

Now there are two options, download the bash script directly or copy the link of the correct script and use wget. Below you can see the command for version 2023.09-0.

```
wget https://repo.anaconda.com/archive/Anaconda3-2023.09-0-Linux-x86_64.sh
```

Now navigate to the correct folder. Now you need to make the bash file executable and install Anaconda. Use the following commands to do this. (File name may differ from yours)

```
chmod +x Anaconda3-2023.09-0-Linux-x86_64.sh
```

```
./Anaconda3-2023.09-0-Linux-x86_64.sh
```

Now confirm the license terms and select the installation path. Now Anaconda will be installed. Select Yes again at the end of the installation to initialize an Anaconda Distribution.

When you open the terminal after the installation, you are in the base environment of Anaconda. If you want to enable or disable the environment, use one of these two commands.

```
conda deactivate base
```

```
conda activate base
```

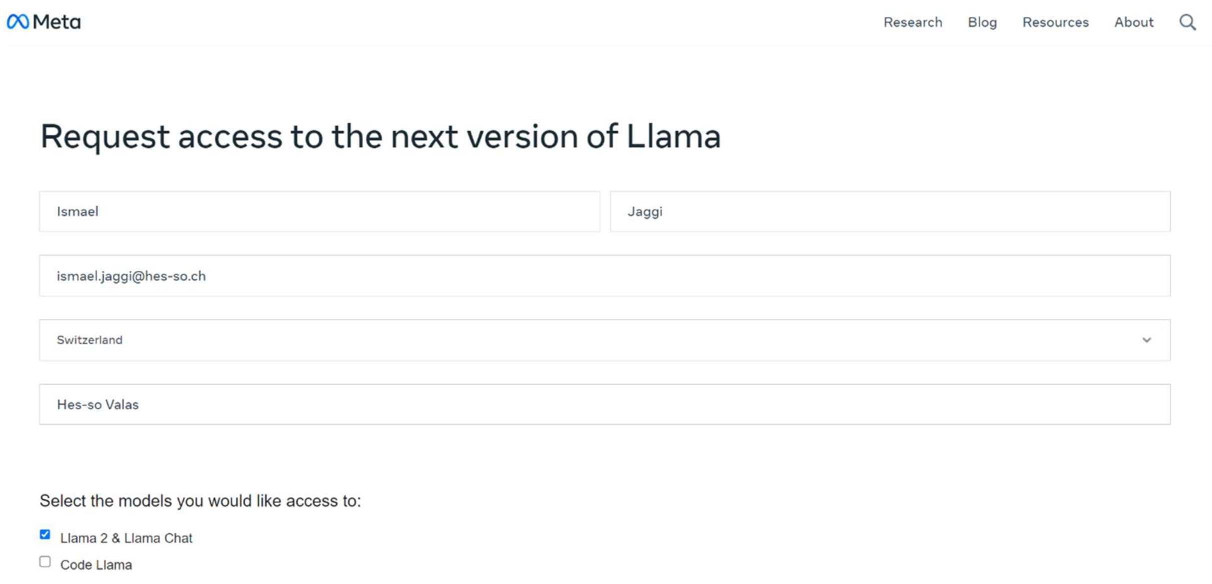
To use Llama, PyTorch and Cuda must still be installed. Use the following command for this.

```
conda install -c pytorch pytorch-cuda
```

To access the different models of Llama 2, go to the following website:

https://ai.meta.com/resources/models-and-libraries/llama-downloads/

Fill in everything and within a few minutes you should receive a mail (Appendix 2) containing your personal URL. The URL is valid only for 24 hours. After this time, you can simply request a new mail (details can remain the same).



FigureApp 4: Access Request MetaAI
Source: Screenshot https://ai.meta.com/resources/models-and-libraries/llama-downloads/

After receiving the mail, navigate to the folder where you want to download the model. Now clone the GitHub repository.

```
git clone https://github.com/facebookresearch/llama.git
```

After cloning, navigate to the top-level directory and run the following command.

```
pip install -e .
```

After that, make sure that the file download.sh has the permission to run. If not:

```
chmod +x download.sh
```

Then execute the download.sh file. First you must enter the URL you received in the mail and then you have to select which models you want to install. After that the models will be downloaded and you can go for a coffee, because this will take some time. For the 13B variant, it is about 25 GB.

```
./download.sh
```

References for the tutorial

Anaconda. (n.d.). *Installing on Linux — Anaconda documentation*. https://docs.anaconda.com/free/anaconda/install/linux/

Anaconda. (n.d.-a). *Pytorch :: Anaconda.org*. https://anaconda.org/pytorch/pytorch

Facebook Research. (n.d.). *llama/README.md at main · facebookresearch/llama*. GitHub. https://github.com/facebookresearch/llama/blob/main/README.md

Appendix 4: DeepInfra API Script

```
import openai

  # Point OpenAI client to our endpoint
openai.api_base = "https://api.deepinfra.com/v1/openai"
# Just leave the API key empty. You don't need it to try our models.
openai.api_key = ""

# Your chosen model here
MODEL_DI = "meta-llama/Llama-2-70b-chat-hf"
chat_completion = openai.ChatCompletion.create(
    model="meta-llama/Llama-2-70b-chat-hf",
    messages=[{"role": "system", "content": "You are a nutrition coach and
only answer my questions if they are related to nutrition. If they are about
something else, ignore them."},
              {"role": "user", "content": "What should I eat this lunch?
Allergies= none, Preferences = Pork lots of Protein"}],
    stream=True,
)

# print the chat completion
for event in chat_completion:
    if 'content' in event.choices[0]['delta']:
        print(event.choices[0]["delta"]["content"], end="")
```

Source: Changed script from https://deepinfra.com/blog/compare-models

Appendix 5: LLM Create User-Profile Log

Successful profile creation

2023-10-20 13:01:42,737 [INFO] - User: Hi my name is Ismael Jaggi

2023-10-20 13:01:50,095 [INFO] - LLM: Sure! Here's the extracted information in the format you requested:

First Name: Ismael

Last Name: Jaggi

Age: None

Gender: None

Height: None

Weight: None

Allergies: No Allergies

Dietary Target: None

Eating Habits: None

2023-10-20 13:02:21,771 [INFO] - User: I'm 23 years old and 180cm tall and I have a nut allergy.

2023-10-20 13:02:30,070 [INFO] - LLM: Sure! Here's the updated information in the format you requested:

First Name: Ismael

Last Name: Jaggi

Age: 23

Gender: Male

Height: 180cm

Weight: None

Allergies: Nut allergy

Dietary Target: None

Eating Habits: None

2023-10-20 13:03:03,013 [INFO] - User: I'm a vegetarian.

2023-10-20 13:03:08,067 [INFO] - LLM: Sure! Here's the updated information in the format you requested:

First Name: Ismael

Last Name: Jaggi

Age: 23

Gender: Male

Height: 180cm

Weight: None

Allergies: Nut allergy

Dietary Target: None

Eating Habits: Vegetarian

2023-10-20 13:03:33,052 [INFO] - User: How far away is the moon?

2023-10-20 13:03:33,941 [INFO] - LLM: NO_INFORMATION

2023-10-20 13:04:24,244 [INFO] - User: Ok I'm sorry. I weigh 85kg and want to maintain my weight.

2023-10-20 13:04:28,465 [INFO] - LLM: Sure, here's the updated information in the format you requested:

First Name: Ismael

Last Name: Jaggi

Age: 23

Gender: Male

Height: 180cm

Weight: 85kg

Allergies: Nut allergy

Dietary Target: Maintain weight

Eating Habits: Vegetarian

Not successful profile creation

In the following sequence of examples, three issues have arisen. Firstly, the first and last name and the dietary goal were extracted from the examples in the system prompt. Secondly, the response format is not entirely correct. Thirdly, the height was not accurately recorded.

2023-10-20 15:02:38,037 [INFO] - User: Hi I'm lactose intolerant and vegan.

2023-10-20 15:02:45,456 [INFO] - LLM: Sure! Here's the extracted information in the format you requested:

* First Name: Ismael
* Last Name: Jaggi
* Age: None
* Gender: None
* Height: None
* Weight: None
* Allergies: Lactose intolerance
* Dietary Target: Maintain weight
* Eating Habits: Vegan

2023-10-20 15:02:59,976 [INFO] - User: I'm 25 and male.

2023-10-20 15:03:05,225 [INFO] - LLM: Sure! Here's the updated information in the format you requested:

* First Name: Ismael
* Last Name: Jaggi
* Age: 25
* Gender: Male
* Height: None
* Weight: None
* Allergies: Lactose intolerance
* Dietary Target: Maintain weight
* Eating Habits: Vegan

2023-10-20 15:03:26,705 [INFO] - User: I weigh about 80kg.

2023-10-20 15:03:36,147 [INFO] - LLM: Sure! Here's the updated information in the format you requested:

* First Name: Ismael
* Last Name: Jaggi
* Age: 25
* Gender: Male
* Height: None
* Weight: 80 kg
* Allergies: Lactose intolerance
* Dietary Target: Maintain weight
* Eating Habits: Vegan

2023-10-20 15:04:26,116 [INFO] - User: I'm 1 meter and 80cm tall.

2023-10-20 15:04:37,107 [INFO] - LLM: Sure! Here's the updated information in the format you requested:

* First Name: Ismael
* Last Name: Jaggi
* Age: 25
* Gender: Male
* Height: 173 cm (1m80)
* Weight: 80 kg
* Allergies: Lactose intolerance
* Dietary Target: Maintain weight
* Eating Habits: Vegan

Appendix 6: Project Versions and Dependencies

Name	Version
Conda	23.9.0
Python	3.11.5
openai	0.28.1
rospy	1.16.0
google-cloud	0.34.0
google-cloud-texttospeech	2.14.2
google-cloud-speech	2.22.0
pyaudio	0.2.14

TableApp 1: Project Versions and Dependencies
Source: Own representation

Appendix 7: Test Result Logs

Profile Creation and Login

2023-10-26 13:43:51,395 [INFO] - Application started

2023-10-26 13:43:55,426 [INFO] - Start Register

2023-10-26 13:43:59,511 [INFO] - Firstname: Max

2023-10-26 13:44:05,158 [INFO] - Lastname: Smith

2023-10-26 13:44:08,035 [INFO] - Age: 45

2023-10-26 13:44:11,842 [INFO] - Gender: Male

2023-10-26 13:44:16,592 [INFO] - Height: 195

2023-10-26 13:44:20,616 [INFO] - Weight: 120

2023-10-26 13:44:47,323 [INFO] - Allergies: Wheat allergy

2023-10-26 13:44:57,428 [INFO] - Dietary Target: lose weight

2023-10-26 13:45:01,364 [INFO] - Eating Habits: Pescetarian

2023-10-26 13:45:01,366 [INFO] - User-Profile saved

2023-10-26 13:45:01,367 [INFO] - Profile Creation successful

2023-10-26 13:45:05,569 [INFO] - Application finished

2023-10-26 13:45:09,954 [INFO] - Application started

2023-10-26 13:45:12,954 [INFO] - Start Login

2023-10-26 13:45:18,280 [INFO] - Firstname: Max

2023-10-26 13:45:20,546 [INFO] - Lastname: Smith

2023-10-26 13:45:20,548 [INFO] - Login successful

2023-10-26 13:45:24,436 [INFO] - Application finished

Generate Meal Suggestions

2023-10-26 13:55:38,188 [INFO] - Application started

2023-10-26 13:55:39,327 [INFO] - Start Login

2023-10-26 13:55:41,066 [INFO] - Firstname: Max

2023-10-26 13:55:42,543 [INFO] - Lastname: Smith

2023-10-26 13:55:42,545 [INFO] - Login successful

2023-10-26 13:55:44,432 [INFO] - Start Meal Suggestion

2023-10-26 13:56:01,184 [INFO] - Meal Suggestion:

For breakfast, consider having a protein-packed omelet made with eggs and your choice of vegetables, such as spinach, mushrooms, and bell peppers. This will provide you with the energy you need to start your day while also being light on calories.
For lunch, try a grilled salmon fillet served with a side of quinoa and steamed broccoli. Salmon is rich in omega-3 fatty acids which can help reduce inflammation and promote weight loss, while quinoa provides a good source of protein and fiber.
For dinner, have a hearty bowl of seafood soup made with a variety of fish and shellfish, such as shrimp, scallops, and mussels, along with some vegetables like carrots, celery, and potatoes. This will not only satisfy your hunger but also provide a good balance of nutrients to support your weight loss goals.

2023-10-26 13:56:30,155 [INFO] - Meal Suggestion finished

2023-10-26 13:56:31,527 [INFO] - Application finished

Nutritional Coaching – Basic Questions

2023-10-26 14:08:45,176 [INFO] - Application started

2023-10-26 14:08:46,560 [INFO] - Start Login

2023-10-26 14:08:48,615 [INFO] - Firstname: Max

2023-10-26 14:08:50,163 [INFO] - Lastname: Smith

2023-10-26 14:08:50,165 [INFO] - Login successful

2023-10-26 14:08:52,397 [INFO] - Start Nutrition Coaching

2023-10-26 14:09:06,154 [INFO] - Question asked: How many meals per day are recommended?

2023-10-26 14:09:13,804 [INFO] - Answer: The recommended number of meals per day for weight loss varies from person to person, but generally, it's best to aim for 3-5 main meals and 2-3 snacks in between. This can help keep your metabolism going and prevent extreme hunger that can lead to overeating. However, it's important to listen to your body and find the frequency that works best for you.

2023-10-26 14:09:14,690 [INFO] - Answer verified

2023-10-26 14:09:36,397 [INFO] - Nutrition Coaching finished

2023-10-26 14:09:38,291 [INFO] - Application finished

Nutritional Coaching - Advanced Questions

2023-10-26 14:17:33,605 [INFO] - Application started

2023-10-26 14:17:34,654 [INFO] - Start Login

2023-10-26 14:17:37,298 [INFO] - Firstname: Max

2023-10-26 14:17:38,606 [INFO] - Lastname: Smith

2023-10-26 14:17:38,608 [INFO] - Login successful

2023-10-26 14:17:40,669 [INFO] - Start Nutrition Coaching

2023-10-26 14:17:44,059 [INFO] - Question asked: How can I increase my protein intake?

2023-10-26 14:17:54,650 [INFO] - Answer: Increase your protein intake by consuming more fish, eggs, dairy products, legumes, nuts, and seeds. Aim for at least 0.8-1 gram of protein per kilogram of body weight daily. Consider adding protein supplements or protein-rich snacks like Greek yogurt or cottage cheese. Also, incorporate plant-based protein sources like beans, lentils, and tofu into your meals.

2023-10-26 14:17:57,006 [INFO] - Answer verified

2023-10-26 14:17:59,825 [INFO] - Nutrition Coaching finished

2023-10-26 14:18:01,206 [INFO] - Application finished

Nutritional Coaching – Contextual Questions

2023-10-26 15:08:44,846 [INFO] - Application started

2023-10-26 15:08:47,981 [INFO] - Start Login

2023-10-26 15:08:49,336 [INFO] - Firstname: Max

2023-10-26 15:08:50,843 [INFO] - Lastname: Smith

2023-10-26 15:08:50,845 [INFO] - Login successful

2023-10-26 15:08:52,432 [INFO] - Start Nutrition Coaching

2023-10-26 15:09:12,253 [INFO] - Question asked: Can you give me some alternative for bread?

2023-10-26 15:09:23,561 [INFO] - Answer: Sure, Max! As a pescetarian with a wheat allergy, there are several bread alternatives that you can consider. Here are a few options:

1. Almond flour bread: Made from almond flour, this bread is gluten-free and can be a great substitute for traditional bread.
2. Coconut flour bread: This bread is made from coconut flour and is also gluten-free. It has a slightly sweet flavor and is high in fiber.
3. Flaxseed bread: Flaxseed bread is made from ground flaxseeds and is a good source of omega-3 fatty acids. It's also gluten-free and low in carbs.
4. Tortillas: Tortillas made from corn or rice flour can be a good alternative to bread. Look for gluten-free options to avoid any allergic reactions.
5. Lettuce wraps: Instead of using bread, try using lettuce leaves as wraps for your sandwiches. This is a great low-carb option and can help you save calories.

Remember, it's important to always read labels and check for any allergens before consuming any new food products.

2023-10-26 15:09:24,334 [INFO] - Answer verified

2023-10-26 15:09:54,441 [INFO] - Question asked: Can you tell me the recipe for number 3?

2023-10-26 15:10:18,008 [INFO] - Answer: Sure, here's a simple recipe for flaxseed bread:

Ingredients:

* 1 cup ground flaxseed
* 1/2 cup almond flour
* 1/4 cup coconut flour

* 1/4 cup arrowroot starch
* 1 teaspoon baking soda
* 1/4 teaspoon salt
* 1 egg
* 1 tablespoon honey or maple syrup
* 1/4 cup warm water

Instructions:

1. In a large bowl, mix together the ground flaxseed, almond flour, coconut flour, arrowroot starch, baking soda, and salt.
2. Add the egg, honey or maple syrup, and warm water to the dry ingredients and stir until well combined.
3. Pour the batter into a greased loaf pan and bake at 375 F (190 C) for 35-40 minutes or until a toothpick inserted comes out clean.
4. Let the bread cool for 10 minutes before slicing and serving.

Note: You can adjust the amount of honey or maple syrup to suit your taste preferences. Also, you can add nuts, seeds, or herbs like sesame seeds, sunflower seeds, or rosemary to the dough for added texture and flavor.

2023-10-26 15:10:19,078 [INFO] - Answer verified

2023-10-26 15:10:30,764 [INFO] - Nutrition Coaching finished

2023-10-26 15:10:33,344 [INFO] - Application finished

Appendix 8: Usage DeepInfra

The screenshot below illustrates the cost breakdown of DeepInfra. I only used the API key during testing, which is why the associated costs remained minimal. However, it's worth noting that the expenses for a robot processing multiple prompts daily would be significantly higher.

Usage

November 2023

Model	Type	Usage	Rate	Spend
meta-llama/Llama-2-70b-chat-hf	text-generation	50510 in tokens	$0.0007/Ktoken	**$0.03**
meta-llama/Llama-2-70b-chat-hf	text-generation	9923 out tokens	$0.0010/Ktoken	**$0.00**
			Total	**$0.03**

October 2023

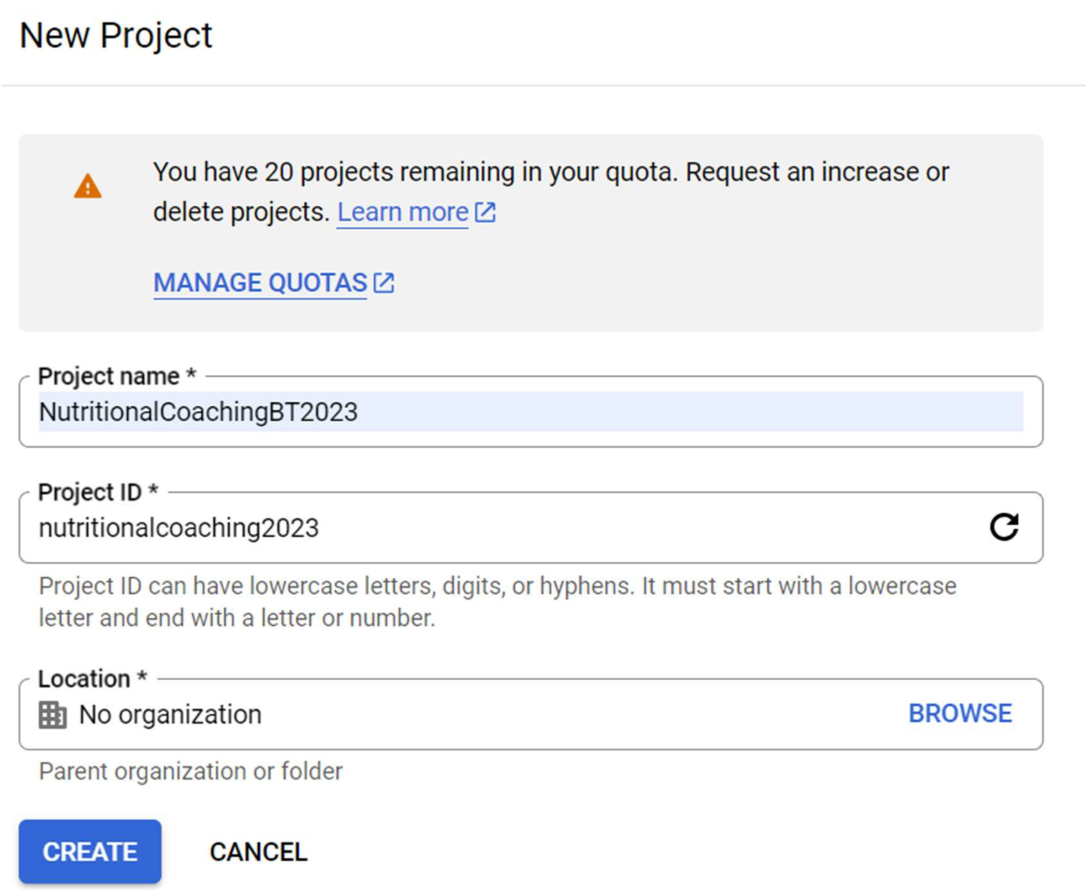
No usage this month

FigureApp 5: Usage DeepInfra
Source: Screenshot https://deepinfra.com/dash/usage

Appendix 9: Tutorial Use of Google Cloud

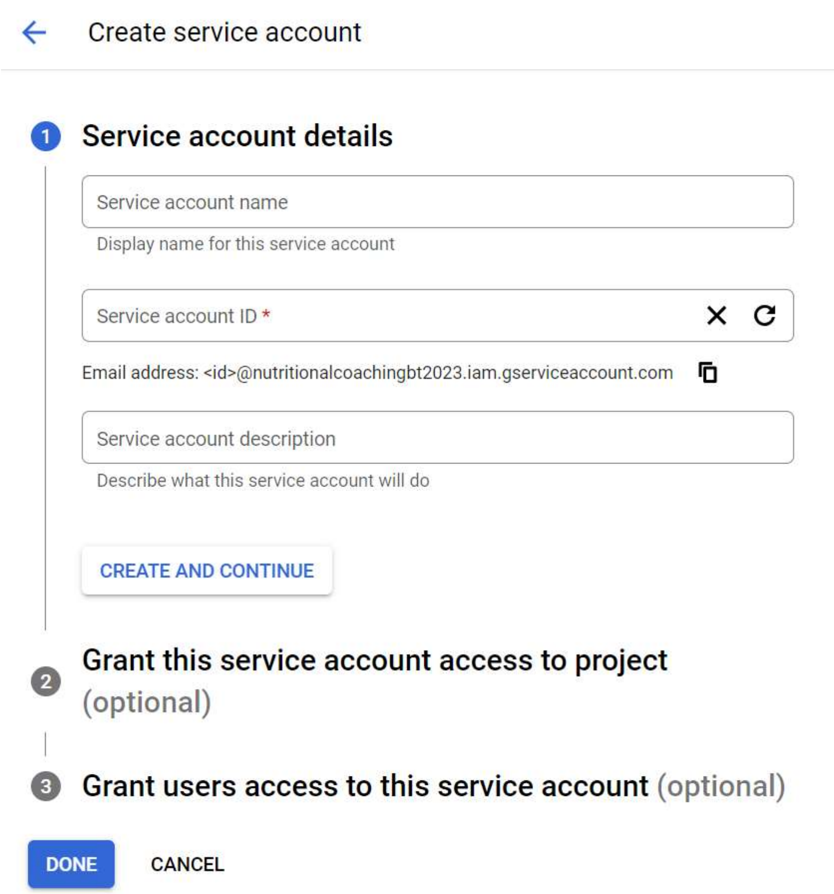
Setting up the account

To utilize the various services offered by Google Cloud, the first step involves creating a Google account. For those who have not previously used Google Cloud, there is the option to test it for free with a $300 credit or for a duration of 90 days. Next, a new project must be created, as shown in the image below:



FigureApp 6: Create new Project
Source: Screenshot from https://console.cloud.google.com

When creating the new project, it is necessary to link a billing account. If no account exists, one can be created. A debit or credit card can be added as the payment method. Following this, you can enable the required APIs. Finally, to use the APIs, you need to create a service account. To do this, navigate to the selected API, click on the "Credentials" tab, and then select "Create credentials" -> "Service Account." Enter all the necessary details as prompted.



FigureApp 7: Create Service Account
Source: Screenshot from https://console.cloud.google.com

The created service accounts can be found under: IAM and Administration -> Service Accounts in the navigation menu. To create a key, click on the service account and select the "Keys" tab. Click on "Add Key" > "Create new key." The key can then be downloaded in a JSON file and set as an environment variable.

Sentiment Analysis API

API-Name: Cloud Natural Language API

To begin, the Google Cloud CLI must be installed to gain access to the necessary command-line tools.

```
sudo snap install google-cloud-cli –classic
```

After the Google Cloud CLI is installed, you need to link your Google account (the one used to create the project on Google Cloud). The sign-in process takes place via a web browser. Following this, all your projects will be listed, allowing you to select the appropriate project.

```
gcloud init
```

Finally, you need to install the libraries for using Google Cloud in a Python program. This can be accomplished with the following command.

```
pip3 install --upgrade google-cloud-language
```

Now, with these essential components installed, you can access the API with Python using the following import statement.

```
from google.cloud import language_v2
```

Speech to Text and Text to Speech API on Windows

API-Names: Cloud Speech-to-Text API, Cloud Text-to-Speech API

To begin, the Google Cloud CLI must be installed to gain access to the necessary command-line tools. Google Cloud requires Python.

```
(New-Object Net.WebClient).DownloadFile("https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe", "$env:Temp\GoogleCloudSDKInstaller.exe")
```

```
& $env:Temp\GoogleCloudSDKInstaller.exe
```

Now follow the installation process, gcloud init will be executed automatically. You need to link your Google account (the one used to create the project on Google Cloud). The sign-in process takes place via a web browser. Following this, all your projects will be listed, allowing you to select the appropriate project.

To use the APIs in a Python Project, you need to install the following two libraries.

```
pip install google-cloud-texttospeech
```

```
pip install google-cloud-speech
```

References for the tutorial

Google. (n.d.). *Speech-to-Text einrichten*. Google Cloud. https://cloud.google.com/speech-to-text/docs/before-you-begin?hl=de

Google. (n.d.-a). *Sentiment Analysis Tutorial*. Google Cloud. https://cloud.google.com/natural-language/docs/sentiment-tutorial

Google. (n.d.-b). *Install the gcloud CLI*. Google Cloud. https://cloud.google.com/sdk/docs/install

Appendix 10: Installation Guide Windows

To run the program on Windows in the terminal, you need to install Python. Visit the Python website and download the latest version of Python (minimum 3.8). Python version 3.11.5 was used during development.

`https://www.python.org/ftp/python/3.12.0/python-3.12.0-amd64.exe`

Now, install Python. To use the program, you need to install some Python packages. Install the following packages:

```
pip install openai
```

```
pip install PyAudio
```

Then, follow the instructions in Appendix 9 to create a Google Account. Skip the Sentiment API part and perform the Speech to Text and Text to Speech part on Windows in the guide. Now, install another pip package for the Sentiment API:

```
pip install google-cloud-language
```

Clone the repository from Appendix 14. To use the repository, you need to create an account on DeepInfra and generate an API key. Now, in the same folder where `main.py` is located, create a new file named *keys.txt*. Write your key in the following format on the first line of the file:

LLM_KEY=<YOURKEY>

The program can now be started in the terminal with Python. Since you are running the program in the terminal, you can specify "Terminal" as a starting parameter. If you do not specify a parameter, "Terminal" will be automatically selected.

Appendix 11: User Guide

Welcome to the QTrobot Nutrition Assistant, your personalized nutrition coach. This user manual will guide you through the steps of using the program.

IMPORTANT: Speak loud and clear in the direction of the robot's head

Sign in or Register

- When you start the program, you will be greeted, and you will be asked if you want to sign in or register.

- After a beep signal, you can say "Sign in" or "Register" to choose your option.

- At any time, you can exit the program by saying "Exit."

- If you respond with anything else, the program will ask again.

Sign in

- If you choose "Sign in," you will be prompted to enter your first and last name in the terminal.

- After providing your name, the program will greet you using your first name.

Register

- If you select "Register," you will be asked to provide various pieces of information step by step. For multiple-choice questions, please enter the correct number corresponding to your choice. For questions about weight and age, enter numerical values.

- Once you have successfully created your profile, the program will greet you by your first name.

Main Menu

- After logging in or registering, you will enter the main menu, which offers two primary functions: "Meal" and "Nutrition."

- The program will explain the main menu options, and after that, a beep signal will sound.

- To exit the program, say "Exit." To receive meal suggestions, say "Meal." To ask nutrition-related questions, say "Nutrition."

Meal Suggestions

- If you choose "Meal," the program will provide meal suggestions tailored to your profile.

- After receiving the suggestions, you will be asked if you would like to generate new suggestions.

- After a beep signal, respond with "Yes" for new suggestions or "No" to return to the main menu.

Nutrition Questions Robot

- Select "Nutrition" to ask the robot nutrition-related questions.

- You can enter your question in the terminal.

- The robot will respond based on the information in your profile.

- You can ask additional questions, and they can build on previous questions.

- To exit the nutrition questions, simply enter "Exit," and you will return to the main menu.

Nutrition Questions Terminal

- Select "Nutrition" to ask the program nutrition-related questions.

- Click enter. After a beep signal, you can ask your question.

- The program will respond based on the information in your profile.

- You can ask additional questions, and they can build on previous questions.

- To exit the nutrition questions, simply say "Exit," and you will return to the main menu.

Appendix 12: Questionnaires

Dear participant,

Thank you for taking the time to participate in my test of the QTrobot as a nutrition coach. Please fill out the Pre-Test Questionnaire before conducting the test to share your expectations and concerns regarding the use of a humanoid robot for nutrition coaching. After the test, you will be asked to complete a second questionnaire to share your experience and feedback. To answer all questions, you should have used both main functions, Nutrition Coaching and Meal Suggestions. Please respond to the questions in both questionnaires honestly and comprehensively. Additionally, there is a user guide available under the link at the end of this paragraph that you should read before the test. Your personal data and responses will be treated confidentially and anonymized for use in my bachelor's thesis evaluation. When creating the profile, you can provide fictional information as well. You may answer the questions in German, English, or French.

Pre-Test Questionnaire

1. Have you had prior experiences with humanoid robots? If so, please describe shortly.
2. What are your expectations for the interaction with the QTrobot as a nutrition coach?
3. Do you believe that a humanoid robot can be effective in assisting you with nutrition-related questions and goals?
4. Do you have concerns regarding privacy and security related to the use of a robot for such purposes in your everyday life?
5. How do you feel about the idea of sharing personal information such as name, age, weight, allergies, etc., with the robot?
6. What are your expectations regarding the quality of the recommendations and advice you will receive from the QTrobot?
7. What factors are particularly important to you when assessing the interaction with the robot?
8. Do you have concerns related to the use of Artificial Intelligence and Machine Learning for nutrition coaching?
9. Any additional comments?

Post-Test Questionnaire

1. How would you rate your overall experience?
2. Did you find the robot's features, including login, profile creation, meal suggestions, and nutritional coaching, useful?
3. Which aspects of the interaction with the application impressed you particularly positively?
4. Which aspects of the interaction with the application impressed you particularly negatively?
5. Did the application comply properly with your specific needs and allergies?
6. How do you assess the quality of the generated answers?
7. Were there moments when you felt unsure or uncomfortable during the interaction with the application?
8. How easy or difficult was it to operate the robot and communicate with it?
9. What improvements or additional features would you wish for in the QTrobot as a nutrition coach?
10. Would you recommend the application as a nutrition coach to others? Why or why not?
11. Any additional comments?

Appendix 13: Answers Questionnaire

Pre-Test Questionnaire

Question	Answers
Have you had prior experiences with humanoid robots? If so, please describe shortly.	Robotic Arm
	Just a small interaction with a humanoid robot in a restaurant acting as a waitress
	No
	No
	Yes, I have interacted with anthropomorphic prototypes acting as assistants for nutrition or information boot at scientific conferences
	Yes, I developed human robot interaction frameworks and experimented with personalized negotiation embodiment
	Yes, I have interacted with the QT robot in Dorian's project.
	No
What are your expectations for the interaction with the QTrobot as a nutrition coach?	It could provide nutritional advise e.g. calories intake, BMI for individual etc.
	Not know, will see
	I hope it will be user friendly
	That it will be easy to interact with it and it will help me to start eating healthy.
	rather limited but I'm curious about how the interaction will be handled and which cues I'm going to get
	The robot should be able to articulate his responses while supporting them with a broad variety of gestures. It also should offer

	"personalized" recommendations. Both for the case of healthiness and preferences.
	I think it is a great tool to get suggestions about food consumption and healthier habits.
	Good recommendations.
Do you believe that a humanoid robot can be effective in assisting you with nutrition-related questions and goals?	Sure, if it is trained well with a lot of nutrition data
	not 100% sure, probably not
	It may be and in the future robots will surely be
	Yes, I do.
	Honestly no. As of today they are mostly useless expensive toys. However, in some cases they proved to be effective and to actually aid human operators, patients, and professionals
	Yes.
	Yes, a humanoid Robot can help me with my nutrition goals.
	Yes
Do you have concerns regarding privacy and security related to the use of a robot for such purposes?	I believe the data can be kept anonymous
	no
	No
	Yes, because I needed to enter my name and surname and I don't know how this data will be used. It was also not clear if my questions and my voice are registered.
	Not more than what I'm already concerned about Google, Microsoft, Amazon, etc...
	No.

	Yes, I am worried about the privacy of my data and if my personal data could be shared with my permission.
	Yes, about my personal data
How do you feel about the idea of sharing personal information such as name, age, weight, allergies, etc., with the robot?	I am not comfortable sharing name. Other information are ok.
	depending on who owns the robot
	Comfortable
	I am not OK to share my full name, but for other information it is OK, because it can help to personalize the suggestions for me.
	The robot is not the problem per-se. Who is handling the information/data behind has to come with a clear and trustworthy disclaimer
	Neutral.
	With the robot is ok, but not with other people.
	Good, except for the lastname
What are your expectations regarding the quality of the recommendations and advice you will receive from the QTrobot?	It should be resourceful and accurate.
	low expectations
	I expect them to be very similar to what I would find on internet, but the robot will save me time
	The recommendations were quite personalized and good.
	Mostly positive. Just in one case the information seemed rather forced to fit my profile
	I think they will be quite well articulated.
	Quite high because is connected to LLM state-of-the-art models.

	High expectations
What factors are particularly important to you when assessing the interaction with the robot?	It should be accurate and be able to interpret the voice command properly
	if it actually gives me good nutrition recommendations
	The interaction must be smooth
	If the robot can understand me.
	smoothness and intuitiveness of the interaction, clarity, and not feeling stuck in an over-contraining menu
	First and foremost the recommendations should relate to my requirements and wishes. Then, articulation of sentences, variety of explanations, show of gestures, etc.
	The quality of the answers.
	Easy interaction
Do you have concerns related to the use of Artificial Intelligence and Machine Learning for nutrition coaching?	I believe if we provide enough data to the robot it is capable of doing great things
	No
	No
	No
	Yes. Generative AI learns from stories and produces stories. This does not mean nor entail that those stories are true or really reliable. Trusting those systems in safety/health-critical scenarios without a human or an expert systems supervision is not recommended
	No

	The only concern is about possible inappropriate suggestions that could lead to a diseases or allergy reactions.
	No
Any additional comments?	Voice command should be well interpreted, and the person should not be expected to keep repeating himself.
	No
	-
	-
	-
	No
	The suggestions should be personalized and based on professional advice.
	-

TableApp 2: Pre-Test Questionnaire Answers
Source: Own representation

Post-Test Questionnaire

How would you rate your overall experience? 1 (very bad) -5 (very good)	5
	4
	5
	5
	4
	4
	5

<table>
<tr><td></td><td>5</td></tr>
<tr><td rowspan="8">Did you find the functionalities, including login, profile creation, meal suggestions, and nutritional coaching, useful? 1 (not useful) – 5 (very useful)</td><td>5</td></tr>
<tr><td>4</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td></tr>
<tr><td>4</td></tr>
<tr><td>5</td></tr>
<tr><td>5</td></tr>
<tr><td>4</td></tr>
<tr><td rowspan="8">Which aspects of the interaction with the application impressed you particularly positively?</td><td>Answers looked accurate</td></tr>
<tr><td>gives you nice meal suggestions for you diet preference</td></tr>
<tr><td>The interaction is smooth and the suggestions are not trivial</td></tr>
<tr><td>Good answers from the application.</td></tr>
<tr><td>I liked some of the answers and to a certain extent I liked that my profile info was taken into consideration without the need for repeating them at beginning of each session</td></tr>
<tr><td>Details and articulations of the explanations were quite nice. The ability to ask the advisor directly and receive a customized response.</td></tr>
<tr><td>The voice processing was working very well.</td></tr>
<tr><td>The quality of the answers</td></tr>
<tr><td>Which aspects of the interaction with</td><td>Sometimes it did not recognise my voice</td></tr>
</table>

the application impressed you particularly negatively?	None
	Nothing impressed me particularly negatively
	Nothing. But we need to wait quite long to receive the answer.
	the speech-to-text engine was getting wrong inputs which were used to generate meaningless suggestions/explanations. As it is, this forces the user to wait till the end of the process to then ask again trying to be understood. I guess this part of the process can be improved
	The wait times between the "Recording..." sections were a constant reminder of that its an automated system. Overall interaction wasn't conversationally smooth.
	Nothing
	Nothing
Did the application comply properly with your specific needs and allergies?	Yes
	kind off, maybe nice to specify a type of cuisine, for example Spanish or mediterranean to have better recommendations
	Yes, it did
	Yes
	Yes, very well
	Yes
	Yes
	Yes
How do you assess the quality of the	5

generated answers? 1 (terrible) – 5 (very good)	4
	5
	5
	4
	4
	5
	5
Were there moments when you felt unsure or uncomfortable during the interaction with the application?	No
	No
	No, they were not
	Only when I was thinking about the security (using my name, my voice etc.)
	No
	No
	No
	No
How easy or difficult was it to operate the application and communicate with it? 1 (very easy) – 5 (very difficult)	4
	1
	1
	2
	3
	2

	1
	2
What improvements or additional functions would you like to see?	Probably the speaking could be longer than 6 seconds
	be able to skip any prompt without going completely out of the application (meal suggestions' explanations are kind of long which is good because gives you a lot of info but maybe the user does not find it useful and want to try another option and for user experience better if there is no need to hear the full audio)
	I think the system works already well, but maybe less scripted answer (the quality of the answer is satisfying though).
	-
	said it above :)
	- Further personalization towards preferences in the recommendation section. - Less non-conversational control mechanisms
	The system is excellent an improvement point could be the integration with a graphic user interface to make the user easier.
	-
Would you recommend this application as a nutrition coach to others? Why or why not?	Sure, I will recommend
	yes, can be useful as an introduction to the nutrition world, still improvement to achieve the same results as actually going to a nutrition coach.

	Yes, I would
	I would like to use this application for a longer time to answer this question.
	Soon! I'd recommend keeping investing in this direction. With the right developers, it might get very good very soon!
	Yes, I would
	Yes, because it is a very dynamic and interesting manner to obtain food recommendations.
	Yes sure
Any additional comments?	-
	I don't now if the feature is already included but could be nice to change diet, maybe now I am doing lossing weight, its done and I want to chance to gain muscle.
	I had fun. I can see the hard work behind
	It was interesting to use this application. Great work!
	-
	-
	-
	-

TableApp 3: Post-Test Questionnaire Answers
Source: Own representation

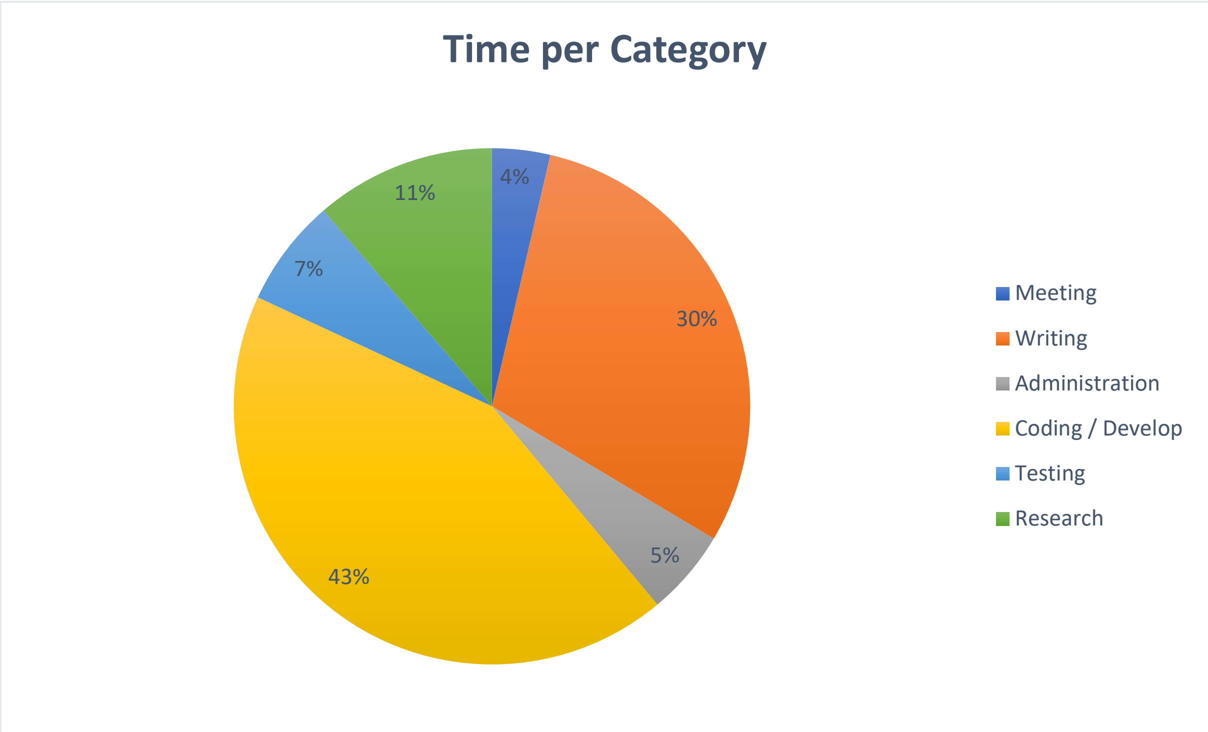
Appendix 14: GitHub Repository

The entire code created for this project is located in this GitHub repository and can be used after downloading and installing the required Python libraries. Only the API key for DeepInfra and the JSON credentials for Google Cloud are not included on GitHub.

https://github.com/jaggiismael/NutritionCoach.git

Appendix 15: Project Management

It took 371 hours to complete the entire bachelor's thesis, the breakdown of these hours into different categories can be seen in the diagram below.



FigureApp 8: Time per Category
Source: Own illustration

Appendix 16: Product Backlog

US Nr.	Theme	As an/a ...	I want to ...	so that ...	Priority	Status	Story Points	Sprint	MoSCoW
1	Preparation	Developer	Setup the Scrum Documentation	I can start the Bachelor Thesis	1000		5	0	Must
2	Preparation	Developer	Investigate the functionalities of the robot	So that I understand all the functionalities of the robot and can use them	960		21	0	Must
3	Preparation	Developer	Investigate Doriane's code	Understanding the interaction of multiple functionalities	920		5	0	Must
4	Preparation	Developer	Investigate about the state of the art on LLM-powered robots	You get an overview of the current developments	880		13	1	Must
5	Preparation	Developer	Investigate about the state of the art on Open Source LLM	I can make a choice between the different LLM	840		21	1	Must
6	Preparation	Developer	Create an architectural considerations of the project	I can get a first idea about the integration of the LLM	800		13	1	Must
7	Preparation	Developer	Investigate how to install the LLM on the robot locally	I can use the LLM without internet connection	760		13	2	Must
8	Preparation	Developer	Experiment with the LLM	I get a better understanding	720		13	2	Must
9	Application	User	Understand what an LLM is	I have a better understanding about the project	680		8	3	Must
10	Preparation	Developer	Investigate about other Benchmark results	I get a better understanding about the different LLMs	640		8	3	Must
11	Preparation	Developer	Create a demo app	I can test the functionalities of the LLM	600		21	3	Must
12	Preparation	Developer	Create the first diagrams of the application	I maintain the overview during programming	570		13	3	Must
13	Preparation	Developer	Create the logic architecture	I get an overview of the application	540		5	5	Must
14	Application	User	Create a profile	I get answers tailored to me	510		13	4	Must
15	Application	User	Login	Only have to give my details once	480		8	4	Must
16	Application	Developer	Create an Controll Layer	The answers get checked	450		21	4	Must
17	Preparation	Developer	Investigate about prompt engineering	I can create better prompts	420		13	5	Must
18	Application	User	Get meal suggestions	I get an idea what I should eat	390		8	5	Must
19	Application	User	Get a nutritional coaching	I can improve my nutrition	360		13	5	Must
20	Application	Developer	Run the application on the robot	The users can interact with the robot	330		5	5	Must
21	Testing	Developer	Create test-scenarios	I see how the programme behaves	300		8	5	Must
22	Documentation	Developer	Document the Control Layer	Other people get a better understanding of his functionalities	270		8	6	Must
23	Application	Developer	Add the possibility to use the program without the robot	I do not depend on the robot	240		8	6	Must
24	Application	User	See emotions from the robot	The interaction feels more natural	210		13	6	Must
25	Application	User	Speak with the robot	I don't always need to use a keyboard	180		21	7	Must
26	Application	Developer	Making the conversation flow more smoothly	It feels more real to the user	150		13	7	Must
27	Testing	Developer	Create questionnaires	I can make a test run with real users	120		13	7	Must
28	Application	Developer	Add function from the robot to the terminal version	The terminal version is more attractive to use	90		21	8	Must
29	Application	Developer	Have external people test the programme	Receive other opinions and suggestions for improvement	60		8	8	Must
30	Documentation	Developer	Finish the report	All knowledge is summarised in one document	30		21	8	Must

FigureApp 9: Product Backlog
Source: Own illustration

Appendix 17: Sprint Plannings

		Sprint	0
1	Setup the Scrum Documentation	Start date	19.09.2023
2	Discover the functionalities of the robot	End date (included)	26.09.2023
3	Investigate Doriane's code	Initial sprint estimatio	44
4	Create a state of the art on LLM-powered robots	Sprint goal	Understand the QTrobot and his functionalities
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (19.09.2023)	Day 02 (20.09.2023)	Day 03 (21.09.2023)	Day 04 (22.09.2023)	Day 05 (25.09.2023)	Day 06 (26.09.2023)
1.1	Setup the Scrum Documentation	Ismael	4	4	0	0	0	0	0
2.1	Tutorial from LuxAI	Ismael	14	14	10	2	0	0	0
2.2	Combine different Functionalities	Ismael	8	8	8	8	5	5	0
3.1	Get an understanding from the code	Ismael	3	3	3	0	0	0	0
3.2	Meeting with Doriane	Ismael	1	1	1	1	0	0	0
4.1	Create a state of the art on LLM-powered robots	Ismael	10	10	10	10	10	4	0
		Total	40						
		Remaining	40	40	32	21	15	9	0
		Therical		40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 10: Sprint 0
Source: Screenshot Sprint Planning

		Sprint	1
4	Investigate about the state of the art on LLM-powered robots	Start date	26.09.2023
5	Investigate about the state of the art on Open Source LLM	End date (included)	03.10.2023
6	Create an architectural considerations of the project	Initial sprint estimation	47
		Sprint goal	Get to know and compare open source LLM.
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (26.09.2023)	Day 02 (27.09.2023)	Day 03 (28.09.2023)	Day 04 (29.09.2023)	Day 05 (02.10.2023)	Day 06 (03.10.2023)
4.1	Add other robots to the state of the art	Ismael	6	6	0	0	0	0	0
5.1	Investigate Open Source LLM	Ismael	14	14	13	7	4	2	0
5.2	Create state of the art from open source LLM	Ismael	8	8	8	6	0	0	0
5.3	Compare the different LLM	Ismael	4	4	4	4	6	4	0
6.1	Create an architectural considerations of the project	Ismael	8	8	8	8	3	0	0
		Total	40						
		Remaining	40	40	33	25	13	6	0
		Therical		40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 11: Sprint 1 Planning
Source: Screenshot Sprint Planning

		Sprint	2
5	Investigate about the state of the art on Open Source LLM	Start date	03.10.2023
7	Investigate how to install the LLM on the robot locally	End date (included)	10.10.2023
8	Experiment with the LLM	Initial sprint estimation	68
9	Create a small demo app	Sprint goal	Start using the LLM with the robot
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (03.10.2023)	Day 02 (04.10.2023)	Day 03 (05.10.2023)	Day 04 (06.10.2023)	Day 05 (09.10.2023)	Day 06 (10.10.2023)
5.1	Add PaLM to the comparison	Ismael	3	3	0	0	0	0	0
7.1	Install the LLM on the robot locally	Ismael	9	9	7	5	0	0	0
7.2	Create a tutorial	Ismael	3	3	1	1	0	0	0
8.1	Experiment with the LLM	Ismael	13	13	13	13	13	4	0
9.1	Use LLM in a program	Ismael	12	12	12	12	12	8	0
		Total	40						
		Remaining	40	40	33	31	25	12	0
		Therical		40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 12: Sprint 2 Planning
Source: Screenshot Sprint Planning

		Sprint	3
9	Understand what an LLM is	Start date	10.10.2023
10	Investigate about other Benchmark results	End date (included)	17.10.2023
11	Create a small demo app	Initial sprint estimation	50
12	Create the first diagrams of the application	Sprint goal	Start with the development
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate		Day 01 (10.10.2023)	Day 02 (11.10.2023)	Day 03 (12.10.2023)	Day 04 (13.10.2023)	Day 05 (16.10.2023)	Day 06 (17.10.2023)
9.1	Explain in detail what an LLM is	Ismael	8		8	2	0	0	0	0
10.1	Add Llama 2 70B to the comparison	Ismael	2		2	0	0	0	0	0
10.2	Add a multilingual Benchmark	Ismael	4		4	4	4	4	4	0
11.1	Create a script to call the API	Ismael	4		4	4	0	0	0	0
11.2	Create a script to get meal suggestions	Ismael	6		6	6	6	0	0	0
11.3	Create User with LLM	Ismael	6		6	6	6	6	0	0
11.4	Add allergies to the prompt	Ismael	4		4	4	4	4	2	0
12.1	Update architectural consideration	Ismael	2		2	2	2	0	0	0
12.2	Create sequence diagram for meal suggestions	Ismael	4		4	4	4	4	4	0
		Total	40							
		Remaining	40		40	32	26	18	10	0
		Therical			40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 13: Sprint 3 Planning
Source: Screenshot Sprint Planning

		Sprint	4
13	Create the logic architecture	Start date	17.10.2023
14	Create a profile	End date (included)	24.10.2023
15	Login	Initial sprint estimation	47
16	Create an Controll Layer	Sprint goal	Add further main functions
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate		Day 01 (17.10.2023)	Day 02 (18.10.2023)	Day 03 (19.10.2023)	Day 04 (20.10.2023)	Day 05 (23.10.2023)	Day 06 (24.10.2023)
13.1	Create the logic architecture	Ismael	4		4	4	4	0	0	0
14.1	Create User Class	Ismael	1		1	0	0	0	0	0
14.2	Create profile with LLM	Ismael	8		8	2	2	6	0	0
14.3	Add rules and checks	Ismael	6		6	4	2	0	0	0
14.4	Save profile to JSON File	Ismael	1		1	0	0	0	0	0
15.1	Get first and last name with LLM	Ismael	2		2	2	0	0	0	0
15.2	Get user profile from JSON File	Ismael	2		2	2	0	0	0	0
16.1	Investigate about possibilities	Ismael	4		4	4	4	0	0	0
16.2	Define basic rules	Ismael	4		4	4	4	4	4	0
16.3	Implement Controll Layer	Ismael	8		8	8	8	8	8	0
		Total	40							
		Remaining	40		40	30	24	18	12	0
		Therical			40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 14: Sprint 4 Planning
Source: Screenshot Sprint Planning

		Sprint	5
13	Create the logic architecture	Start date	24.10.2023
17	Investigate about prompt engineering	End date (included)	31.10.2023
18	Get meal suggestions	Initial sprint estimation	52
19	Get a nutritional coaching	Sprint goal	Get a working first version of the application
20	Run the application on the robot	Nbre working days	6
21	Create some test-scenarios		

ID US.task	Task Name	Resp.	Initial Estimate		Day 01 (24.10.2023)	Day 02 (25.10.2023)	Day 03 (26.10.2023)	Day 04 (27.10.2023)	Day 05 (30.10.2023)	Day 06 (31.10.2023)
13.1	Add more details to the architecture	Ismael	2		2	2	0	0	0	0
17.1	Investigate about prompt engineering	Ismael	4		4	4	0	0	0	0
17.2	Write about prompt engineering	Ismael	4		4	4	0	0	0	0
17.3	Experiment with different prompts	Ismael	3		3	3	3	1	0	0
18.1	Get a simple suggestion	Ismael	2		2	0	0	0	0	0
18.2	Add history + user profile	Ismael	3		3	0	0	0	0	0
19.1	Function to ask questions to LLM	Ismael	2		2	0	0	0	0	0
19.2	Add context + history	Ismael	3		3	0	0	0	0	0
19.3	Add control layer	Ismael	3		3	3	3	0	0	0
20.1	Make necessary changes	Ismael	2		2	2	2	0	0	0
21.1	Create test-scenrios	Ismael	3		3	3	3	2	0	0
21.2	Run the test-scenarios	Ismael	3		3	3	3	3	0	0
21.3	Document the results	Ismael	6		6	6	6	6	6	0
		Total	40							
		Remaining	40		40	30	20	12	6	0
		Therical			40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 15: Sprint 5 Planning
Source: Screenshot Sprint Planning

		Sprint	6
22	Document the Control Layer	Start date	31.10.2023
23	Add the possibility to use the program without the robot	End date (included)	07.11.2023
		Initial sprint estimation	50
24	See emotions from the robot	Sprint goal	Add further functionalities to the robot
25	Speak with the robot	Nbre working days	5

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (31.10.2023)	Day 02 (02.11.2023)	Day 03 (03.11.2023)	Day 04 (06.11.2023)	Day 05 (07.11.2023)
22.1	Document the Control Layer	Ismael	4	4	0	0	0	0
23.1	Add switch to the program	Ismael	4	4	0	0	0	0
24.1	Connect to the API	Ismael	4	4	3	1	0	0
24.2	Set emotions of the robot	Ismael	6	6	6	6	0	0
24.3	Add gestures	Ismael	2	2	2	2	2	0
25.1	Activate online speech recognition	Ismael	6	6	6	6	6	0
25.2	Add online speech recognition to robot	Ismael	4	4	4	4	4	0
25.3	Add offline speech recognition to robot	Ismael	4	4	4	4	0	0
25.4	Document	Ismael	4	4	4	4	4	0
25.5	Write tutorial	Ismael	2	2	2	2	1	0
		Total	40					
		Remaining	40	40	31	29	17	0
		Therical		40.0	30.0	20.0	10.0	0.0

FigureApp 16: Sprint 6 Planning
Source: Screenshot Sprint Planning

		Sprint	7
25	Speak with the robot	Start date	07.11.2023
26	Making the conversation flow more smoothly	End date (included)	14.11.2023
27	Create questionnaires	Initial sprint estimation	47
		Sprint goal	Improve the overall user experience
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (07.11.2023)	Day 02 (08.11.2023)	Day 03 (09.11.2023)	Day 04 (10.11.2023)	Day 05 (13.11.2023)	Day 06 (14.11.2023)
25.1	Activate online speech recognition	Ismael	6	6	4	4	4	4	0
25.2	Add online speech recognition to robot	Ismael	5	5	5	5	5	5	0
25.3	Document	Ismael	4	4	4	4	4	1	0
25.4	Write tutorial	Ismael	2	2	2	2	2	0	0
26.1	Add thinking phrases	Ismael	4	4	0	0	0	0	0
26.2	Change mouth while speaking	Ismael	2	2	0	0	0	0	0
26.3	Make the conversation smoother	Ismael	5	5	3	3	0	0	0
27.1	Create Pre-Questionnaire	Ismael	2	2	2	0	0	0	0
27.2	Create Post-Questionnaire	Ismael	2	2	2	0	0	0	0
27.3	Create User-Guide for Test	Ismael	4	4	4	0	0	0	0
27.4	Test all Test-Scenarios	Ismael	4	4	2	2	2	0	0
		Total	40						
		Remaining	40	40	28	20	17	10	0
		Therical		40.0	32.0	24.0	16.0	8.0	0.0

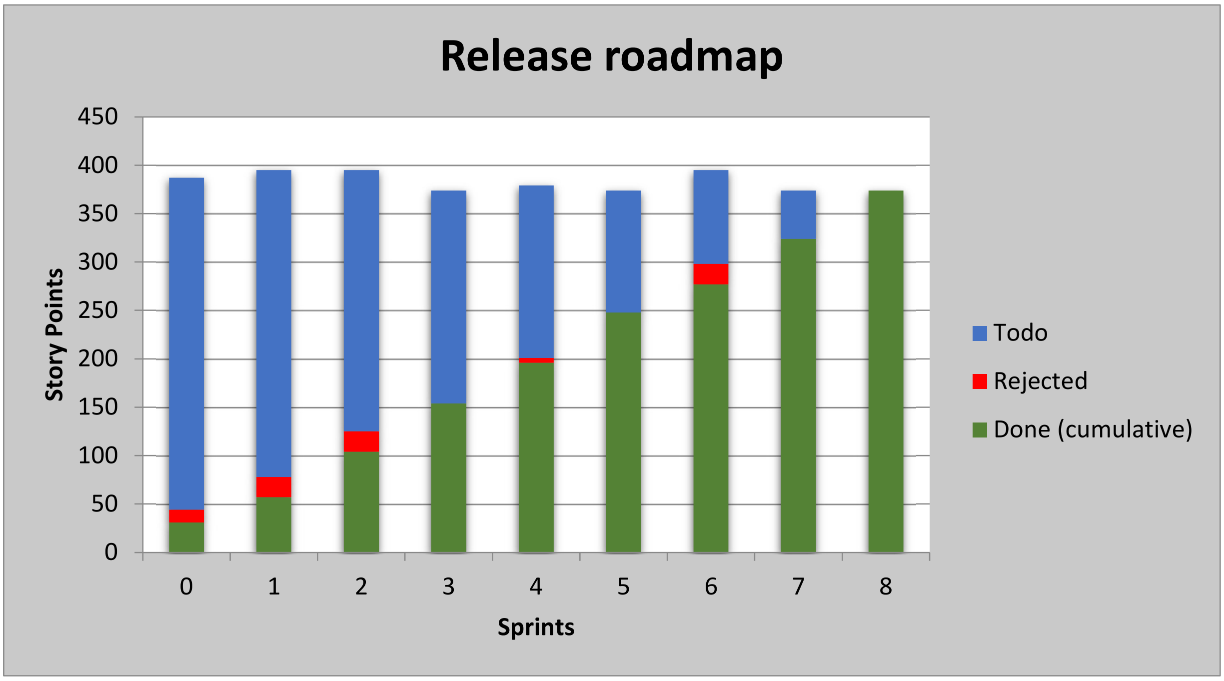
FigureApp 17: Sprint 7 Planning
Source: Screenshot Sprint Planning

		Sprint	8
28	Add function from the robot to the terminal version	Start date	14.11.2023
29	Have external people test the programme	End date (included)	21.11.2023
30	Finish the report	Initial sprint estimation	50
		Sprint goal	Bring the work to a conclusion
		Nbre working days	6

ID US.task	Task Name	Resp.	Initial Estimate	Day 01 (14.11.2023)	Day 02 (15.11.2023)	Day 03 (16.11.2023)	Day 04 (17.11.2023)	Day 05 (20.11.2023)	Day 06 (21.11.2023)
28.1	Add Google Cloud TTS	Ismael	6	6	0	0	0	0	0
28.2	Tests with Google Cloud STT	Ismael	10	10	6	0	0	0	0
28.3	Make conversation smoother	Ismael	2	2	2	1	0	0	0
29.1	Rewrite Questionnaire	Ismael	1	1	1	1	0	0	0
29.2	Supervise the tests	Ismael	3	3	3	3	3	3	0
29.3	Analyse the results	Ismael	4	4	4	4	4	4	0
29.4	Adapt programme	Ismael	2	2	2	2	2	2	0
30.1	Finish testing part	Ismael	2	2	2	2	2	2	0
30.2	Wirte part about project management	Ismael	3	3	3	1	1	1	0
30.3	Writing the required mandatory parts	Ismael	7	7	7	7	3	0	0
		Total	40						
		Remaining	40	40	30	21	15	12	0
		Therical		40.0	32.0	24.0	16.0	8.0	0.0

FigureApp 18: Sprint 8 Planning
Source: Screenshot Sprint Planning

Appendix 18: Release Roadmap



FigureApp 19: Release Roadmap
Source: *Own* illustration

Author’s declaration

I hereby declare that I have carried out this final research project on my own without any help other than the references listed in the list of references and that I have only used the sources mentioned.

Sierre, 24. November 2023 Ismael Jaggi