Technical Computer Science - Module 11 - Design Project

# Developing a LLM Powered Virtual Reality Therapist

# Report Paper - Group 1

R.R. Ursu - s2979616
R.M. Gavrila - s2930277
N. Kezins - s2944405
B. Busui - s3010082
D.C. Ploesteanu - s2913879

# Supervisors:

dr. ir. Randy Klassen - EEMCS faculty dr. Lorenzo Gatti - EEMCS faculty

# Client:

prof. dr. Wim Veling - University of Groningen

May 6, 2025

UNIVERSITY

OF TWENTE.

#### Abstract

Mental health specialists are experiencing an increasing demand, leading not only to long waiting lists, but also to increased workloads for therapists. Digital advancements, particularly artificial intelligence (AI) and virtual reality (VR) provide opportunities for developing digital mental health interventions. The aim of this project is to create a virtual reality LLM-powered therapist that would decrease the workload of therapists, and improve the situation of prolonged waiting lists for receiving therapy. For this purpose, we developed and animated a humanoid avatar using Unity software, integrated with eye tracking and sentiment extraction from the voice. The large language model used in the project generates relevant responses for the therapeutic situation, while also controlling the animations and expressions of the avatar. User testing was conducted to evaluate the system, after which results were analysed and discussed, and potential improvements for the system were identified.

Keywords: virtual reality, artificial intelligence, psychotherapy, unity, docker, large language model, humanoid avatar animation, user evaluation

# Contents

1	Intr	oducti	on	<b>5</b>							
	1.1	Contex	xt	. 5							
	1.2	Client		. 5							
	1.3	Projec	t Objectives	. 5							
	1.4	Report	t Outline	. 6							
	1.5	AI Sta	atement	. 6							
<b>2</b>	Methodology and Planning 7										
	2.1	Agile .		. 7							
	2.2	Planni	ing	. 8							
		2.2.1	Sprint Task Division	. 8							
		2.2.2	Member Task Division	. 9							
	2.3	Client	Involvement	. 10							
3	Requirements 12										
	3.1	User re	equirements	. 12							
	3.2	Functi	onal requirements	. 13							
	3.3	Non-fu	inctional requirements	. 14							
<b>4</b>	Design 15										
	4.1	System	n Architecture	. 15							
	4.2	System Integration									
	4.3	Conver	rsation Stages	. 20							
	4.4	Risk A	Analysis	. 20							
		4.4.1	Possible risks	. 21							
		4.4.2	Mitigation Strategies	. 22							
<b>5</b>	Development 24										
	5.1	Verbal	& Non-verbal User Input	. 24							
		5.1.1	Voice Recording	. 24							
		5.1.2	Blocking service requests & Multithreaded Design .	. 25							
		5.1.3	Speech Emotion Recognition (SER)	. 29							
		5.1.4	Speech-To-Text $(STT)$	. 29							
		5.1.5	Eye Tracking	. 30							
	5.2	LLM .		. 31							
		5.2.1	Model Selection	. 31							
		5.2.2	Input Processing	. 31							
		5.2.3	Therapeutic Response Generation	. 32							
		5.2.4	Prompts	. 32							
	5.3	VR Er	$\operatorname{nvironment} \ldots \ldots$	. 33							
		5.3.1	VR Device Setup	. 33							
		5.3.2	Scene Setup	. 34							

		5.3.3	Avatar Choice	35					
		5.3.4	Avatar Animations	37					
		5.3.5	Text-to-Speech (TTS)	39					
	5.4	Conta	inerization and Deployment	39					
		5.4.1	Containerization	39					
		5.4.2	Packaging & Deployment	41					
6	Test	ting		42					
	6.1	Testin	g Strategy	42					
	6.2	Syster	m Tests	43					
		6.2.1	System Test Methodology	43					
		6.2.2	Core Functional Tests	43					
		6.2.3	Interaction and Animation Tests	46					
		6.2.4	System Stability and Performance	48					
		6.2.5	Edge Cases	49					
	6.3	User 7	Tests	49					
7	Eva	luatio	n	52					
	7.1	Syster	m Test Results	52					
		7.1.1	Core Functional Tests	52					
		7.1.2	Interaction and Animation Tests	53					
		7.1.3	System Stability and Performance	55					
		7.1.4	Edge Cases:	56					
	7.2	User 7	Test Results	56					
	7.3	Perfor	mance	57					
		7.3.1	Hardware and Resource Utilisation	58					
		7.3.2	Runtime Performance	58					
		7.3.3	Known Limitations and Bottlenecks	58					
		7.3.4	Therapist Avatar Performance	59					
	7.4	Requi	rement fulfilment	59					
8	Dise	cussior	a and Conclusion	63					
	8.1	Analy	sis of Findings	63					
	8.2	Sugge	sted Improvements	64					
	8.3	Concl	usion	67					
9	App	oendic	es	71					
$\mathbf{A}$	User Guide								
	A.1	Simple	e instructions	71					
		A.1.1	Installation	71					
		A.1.2	Usage	71					
	A.2	Advar	nced instructions	72					
		A.2.1	Installation	72					

		A.2.2 Usage	72						
		A.2.3 Project Structure & Development Notes	73						
в	Info	rmed Consent Form	75						
$\mathbf{C}$	Prompts								
	C.1	Therapist Response Generation Prompt	76						
	C.2	Non-verbal Cue Interpretation Prompt	78						
	C.3	Summary Prompt	78						
	C.4	Self-harm Detection Prompt	79						

# 1 Introduction

### 1.1 Context

An intake psychotherapy session, also called an intake interview, is the first session that a therapist has with a new client. During the session, the therapist gathers important background and family information, discusses with the client their feelings and concerns, and builds up a prospective schedule for future sessions [1].

Non-verbal communication is a cornerstone of psychotherapy, as it accounts for the majority of interpersonal communication and can serve as a diagnostic and risk assessment value [2]. However, traditional ways of using artificial intelligence for therapy involve textual interfaces - such as the GPT-4 model powered by OpenAI - which lack the possibility of leveraging non-verbal cues in making therapeutic decisions. This raises the question of whether non-verbal cues can be monitored effectively and utilized in psychotherapy sessions conducted by an artificial intelligence.

#### 1.2 Client

The client for this project is Wim Veling, psychiatrist and professor of psychiatry at the University of Groningen. Wim is the primary stakeholder of the project, whose vision prioritizes a high-quality patient experience. Their requirements and insights into psychotherapy have been crucial in shaping the system's user-centric design. We have been in close contact with the client from the beginning of the project, as also described in section 2.3. With the knowledge of our client, we were able to identify therapeutic practices that are relevant for a psychotherapy session, particularly for addressing anxiety symptoms.

#### **1.3** Project Objectives

This report will discuss the development of a dialog system that replicates an intake psychotherapy session, for which we use VR technology. The environment presents a humanoid avatar powered by a large language model, which represents a therapist, with its therapeutic scope limited to anxiety symptoms and disorders. The main focus of the project, in line with our client's priorities, is to leverage the patient's non-verbal cues. Therefore, our goal is to:

Design a VR-based dialog system that uses a LLM-powered humanoid avatar to simulate an intake psychotherapy session on treating anxiety, focusing on leveraging non-verbal cues of the patient. It is important to clarify that the project has high exploratory character, The prototype of this system is meant to inform researchers and therapists about the feasibility and challenges in designing a system that uses VR and AI technology for therapeutic reasons.

### 1.4 Report Outline

The report will begin by discussing methodology and planning, followed by the design of the system, and development. The testing and evaluation of the product will follow, and the report will conclude with the discussion of the results, and the conclusion. In the appendix of the report, a user guide can be found, which will explain how the system can be used on a desktop or laptop device.

#### 1.5 AI Statement

During the writing of this work, we used ChatGPT for grammar checks, for finding synonyms of words, more suitable for an academic submission, explanation of Unity editor functionality and C# code, and brainstorming for creative calls to action for the project poster. After using this tool, we thoroughly reviewed and edited the content generated, taking full responsibility for the final outcome.

# 2 Methodology and Planning

In this section, we will discuss the methodologies adopted for the project, as well as the internal planning of the team. We will begin by discussing the use of the Scrum Agile Methodology, and how characteristics of this methodology, such as sprint planning sessions, stand-up sessions, and retrospectives were implemented within the team. In the Planning subsection, we will discuss the distribution of tasks in each sprint, and the main allocation of tasks for each team member. Lastly, in the Client Involvement subsection, we will discuss the involvement of the client in the project, and how it improved the outcome of our system.

### 2.1 Agile

During this project we adopted the Scrum Agile Methodology for development. In order to achieve this, we executed sprints, sprint reviews, stand-up sessions, and used GitLab as a project management platform for sprint planning.

We segmented the 10 week project timeline into 2-week sprints, and concluded each sprint with a collaborative sprint review meeting involving our supervisors and client in an online meeting, to showcase our progress. During each such session, we gathered feedback and points of improvement from the client, and decided whether it is necessary to return to previous steps to refine requirements or the design of the system.

Aside from sprint reviews, we organized weekly meetings with our supervisors, in which we analysed challenges and identified possible solutions, and also recommendations for the content of the final report. Internally, the team had stand-up sessions 3 to 4 times a week, depending on team availability, during which each team member gave structured updates addressing: completed tasks, encountered challenges, and plans for upcoming work. The responsibility of Scrum Master was assigned through rotation between team members. We decided from the start of the project that any team member experiencing difficulties for longer than two days was required to notify the team and request assistance.

GitLab was employed as a project management tool, specifically for the functionality of Issue Boards. Issues were created from the beginning of the project to reflect the features of the system, and additionally organizational responsibilities of the team, such as materials for Peer Review Presentations, and completing various parts of the final report. These issues were classified on several categories: priority, section of the system (VR, ML, Research, Report), and sprint in which the issue is scheduled for completion. Issue priority was decided based on the user prioritization, and functional and non-functional specifications.

At the end of each sprint, the team convened a sprint planning session,

during which we deliberated on tasks for the following sprint, including the addition of new issues, as well as the rescheduling or re-prioritization of existing items. We also addressed team performance during said session as part of the sprint retrospective, and identified areas for improvement for the upcoming sprint.

The adoption of the Agile methodology enhanced team collaboration and effectiveness. Considering the relatively brief 2-week sprints, we were required to utilize time effectively, in order to successfully complete the scheduled tasks. Moreover, client-involved sprint reviews provided insights into whether the project aligned with client expectations and whether their priorities evolved over time. Accordingly, the resulting system surpassed the expectations established by our client in the beginning.

#### 2.2 Planning

#### 2.2.1 Sprint Task Division

As described in the subsection 2.1, the project timeline was divided into 4 2-week sprints. Below there is a brief explanation of the planning for each sprint:

#### • Sprint 1: Requirements, system design and development

During the first half of the sprint, we created user and system requirements and validated them with the client. Also, this is when we defined the system architecture, created the UML diagrams, and decided which should be the conversation stages.

The second half of the sprint was when we created the container architecture of the system, the speech-to-text (STT) functionality, and the LLM handler service. This is also when we established a connection between the VR container and the Sentiment Extraction container.

#### • Sprint 2: Development

This sprint was dedicated to development, and the following functionalities were implemented: text-to-speech (TTS), the VR headset was connected to the laptop and version-control was added for Unity.

Moreover, this is the sprint when we established a connection between the VR container and the LLM Handler and the speech-to-text(STT) container. Lastly, this is the sprint when we began experimenting with different humanoid avatars for the Unity scene, and began researching body animations and face expressions for the avatar. • Sprint 3: Development and User Testing

We dedicated sprint 3 to creating the animations and expressions for the humanoid avatar, as well as creating the Unity environment, resembling a therapist's office. After this, we were able to implement the eye tracking functionality.

This sprint was also when the LLM was integrated with the body animations and face expressions, and the animations could be triggered by the LLM. We also implemented lip syncing during this time. Lastly, we set up Docker Compose, and implemented the controller command for exiting the session.

For testing, we designed the user testing procedure, and conducted user testing during this sprint. Results of the user testing are assessed in sprint 4.

#### • Sprint 4: System Testing and Report Writing

During sprint 4, we focused on system testing, and also on writing the report. This is also the sprint when we interpreted the results of the user testing, and identified potential improvements of the system.

#### 2.2.2 Member Task Division

The project is divided into several technical components. Below is a summary of the areas alongside the designated responsibilities:

#### • Verbal Non-verbal User Input

- Voice Recording was handled by Dan;
- Eye tracking was a collaborative effort from Raluca, Nikita and Bogdan;
- Speech Emotion Recognition was assigned to Dan;
- Text-to-Speech and Speech-to-Text were set-up by Dan;
- LLM (Large Language Model): Assigned to Nikita;

#### • VR Environment:

- VR device setup was assigned to Bogdan;
- Unity scene setup was handled by Raluca;
- Avatar animations were assigned to Ruxandra;
- Avatar interactions were ensured by Ruxandra;
- Containerization and Deployment: A shared responsibility between Dan, Nikita, and Raluca.

Each of these areas contributes to the overall functionality of our project.

#### 2.3 Client Involvement

The client was involved in our project in several different ways: helping define initial requirements, validating and prioritizing them, refining the project scope to focus on anxiety and safety features like suicide detection and exit mechanisms, and providing ongoing feedback during bimonthly sprint reviews.

Firstly, before the requirements gathering phase of the project, we held a collaborative meeting, to ascertain the client's view of the system, and the initial functionality specifications they would like to see in the project, such as recognition of the user's body language, the capacity of the AI therapist to use active listening techniques, and also delaying advisory intervention until nearing the end of the therapy session.

Drawing upon our documented observations from this meeting, we constructed the user requirements, as well as functional and non-functional system requirements. These were validated by the client during another scheduled meeting, and priority for the requirements was reviewed as well in collaboration with the client.

During this meeting, modifications to the project scope and goals were also decided. Firstly, the scope of the project should be restricted to addressing anxiety symptoms, to optimize the performance of the LLM. Secondly, we identified suicide detection as a relevant feature for the system. This involves recognizing explicit verbal expressions that indicate suicidal intentions. Upon detecting such cues, the session should be immediately terminated, as cases involving potential self-harm are better handled by a human therapist to ensure the user's safety and well-being. Lastly, to ensure the safety of users who may experience distress, either caused by the VR environment or the conversation with the AI therapist, we concluded that an exit mechanism is necessary, which would guarantee immediate termination of the therapy session.

As described in subsection 2.1, bimonthly sprint review meetings were held with the client. Aside from demonstrating our progress during these meetings, we also addressed and resolved mutual concerns related to different aspects of the development process. For example, a concern of the client was that the system response might be insufficiently fast to sustain a natural conversation with the user, and during the interval when the user awaits a response, the AI therapist avatar would assume the static sitting pose, leaving the user uncertain regarding the status of the system. In order to address this issue, we proposed the implementation of a visual indicator, a 'processing' animation, which would communicate to the user that their input was successfully captured and that the system is formulating a response.

This chapter presented the methodology of choice, which was Agile, and how we planned and structured our sprints. Further, we discussed the task distribution among the team members. Finally, we brought to the readers' attention the way our client contributed to our project and the way he improved the outcome of the system.

# **3** Requirements

In this section, we will discuss the requirements for the project, and their categorization.

We formulated the user requirements - representing the aims of the client for the system -, as well as functional (FR) and non-functional (NFR) requirements, and afterwards checked the requirements with the client. In alignment with our client's priorities, functional and non-functional requirements were ranked using the MoSCoW method.: "must have" (the system does not meet its purpose if these requirements are not met), "should have" (not mandatory, but would add substantial value), "could have" (would add value, but are not necessary) and "won't have" (considered out of scope for this project).

#### 3.1 User requirements

We identified 8 user requirements.

- UR1. The system should act as a virtual therapist for the user.
- UR2. The system should help patients with mild cases of anxiety by conducting intake therapy sessions with them, in a similar manner to a human therapist.
- UR3. The system must have a virtual reality avatar, representing a therapist.
- UR4. The system must ask questions to gain a better understanding of the user's situation.
- UR5. The system must make the patient feel heard and understood, to improve the therapeutic relationship.
- UR6. The system must give advice to the patient after asking a maximum number of 15 exploratory questions or nearing the end of the session, when it is indicated by therapeutic guidelines.
- UR7. The system should not attempt to solve severe cases (intention of physical aggression towards the self or other people) on its own.
- UR8. The user should be able to stop the session and/or request human intervention at any time.

#### **3.2** Functional requirements

From these user requirements, the following functional requirements were defined:

#### Must Have

- FR1. The system must show an avatar for the therapist in the VR environment.
- FR2. The system must process the user's voice, extracting text out of speech.
- FR3. The system must pick up non-verbal cues from the user in order to give a proper response and adapt its next responses.
- FR4. The system must use an LLM to generate natural responses during the therapy session.
- FR5. The system must use an LLM to generate empathetic responses during the therapy session.
- FR6. The answers generated by the LLM must be contextually appropriate.

#### Should Have

- FR7. The system should work with minimal human supervision.
- FR8. The system should detect whenever the user is in danger at any point during the session verbal expressions indicating the intention of self harm or a suicide attempt.
- FR9. The system should extract the user's emotional state from the user's voice.
- FR10. The system should remember user preferences and question replies throughout the session to provide personalized therapy.
- FR11. The system should use the rapeutic techniques  $^1$  that meet the user's needs.

#### Could Have

FR12. The system could implement a GUI element, gesture or voice command that will cause immediate termination of the conversation or request of human intervention.

 $<sup>^1{\</sup>rm Active}$  list ening, reflective listening, open-ended questions, validation, reframing, narrative the rapy, etc.

FR13. The system avatar could adapt its language, posture and facial expression based on the user's emotional state (e.g., happy, calm, anxious, depressed).

#### Won't Have

FR14. The system will not allow the user to pick between a male or female avatar in VR.

#### 3.3 Non-functional requirements

From the user requirements, the following non-functional requirements were defined:

#### Must Have

- NFR1. The system must support a real-time conversation with the user.
- NFR2. User non-verbal interactions must be taken into account in the next system response.

#### Should Have

# Could Have

- NFR3. The system can provide a response fast enough (a maximum of 10 seconds) to uphold the conversation naturally.
- NFR4. VR rendering can uphold 60FPS to prevent motion sickness.
- NFR5. The LLM can identify malicious queries (queries that could negatively impact the system or the user) and handle them gracefully.
- NFR6. The product can comply with the GDPR and the European AI Act.

#### Won't Have

\_

This section discussed the requirements of the project. Requirements are differentiated into user requirements, functional system requirements, and non-functional system requirements. Also, the MoSCoW method is discussed in this section, for the purpose of allocating priority to system requirements.

# 4 Design

In this section, we will describe the high-level design of our system. We will explain what the components of the system are and how they communicate with each other. Further, we will also present the new architecture's structure after development. Using the knowledge of our system's components, we will then explain how the conversation that these components implement is structured and what happens in each of its stages.

#### 4.1 System Architecture

After finishing the requirements, we started to create the complete flow of interaction between the user and the system. To support in this process, we created an activity diagram.

By designing the activity diagram (Figure 1), we came up with the process in which the user input is processed and interpreted in such manner that the system generates a contextually appropriate answer and support a complete conversation with the virtual therapist.

The activity diagram represents a single exchange between a client and the therapist. It starts with user pressing the record button which causes audio recording and eye movement tracking. This continues until the stop button is pressed. Once completed, audio recording is sent to Speech transcription and as well as voice sentiment extraction. It is done in parallel with thinking animation played and determination of the most watched object. Upon receipt of all the responses, the data is passed to the LLM, where it transcribes non-verbal cues, generates the therapist's response and gesture and afterwards updates the conversation stage. Lastly the LLM determines whether the input of the user has self-harm desires. This data is packaged and passed back to the VR Headset/Unity module. In case the self-harm alert was triggered, the system shuts down automatically, otherwise we send the therapist's response to the voice service, where it is synthesized into speech. Once done, the thinking animation is stopped and 2 parallel processes are started: playing the therapist's audio in combination with therapist's gesture animation. The activity diagram illustrates a single session of interaction between the client and the therapist in a therapeutic system. The process begins on the user's action to click the "Record" button, initiating both audio recording and eye movement tracking. This continues until the user clicks the "Stop" button. Upon completion, the recorded audio is forwarded for speech transcription and voice sentiment analysis. These actions are performed concurrently with the showing of a "thinking" animation and the determination of the most viewed object by the user. Once all the information has been processed, it is sent to the LLM, where it translates non-verbal cues, generates an appropriate reply from the therapist with a corresponding gesture. This action also renews



Figure 1: Activity diagram

the current stage of the conversation. The LLM then checks whether the user input indicates any self-harm activity. The processed data, together with any detected risk, is wrapped and sent back to the VR Headset/Unity module. If a self-harm alert occurs, the system will terminate automatically; otherwise, the response generated by the therapist is sent to the voice service where it is synthesized as speech. After speech synthesis is complete, the "thinking" animation is stopped and two simultaneous processes begin: the audio response of the therapist and playback of the corresponding gesture animation. It's important to note that while the actual system operates in a cyclical manner, with interactions looping continuously, the activity diagram only represents a single exchange in the conversation. Secondly, we excluded a specification of the process to end the session at any point.

In order to build the architecture, we determined that the system will consist of two main parts: the VR and the LLM (Large Language Model). Additionally, we identified the need for integration with a system that could extract sentiment from the voice, a speech transcription system and a voice synthesizing system.

The first option was to design a monolithic application, as we did in all previous modules. Although this approach would have been for us a comfortable way of proceeding, we quickly reached the conclusion that the integration of all parts would not be as seamless as we would think and we would need to compromise on modularity, since all the components of the project would be closely interconnected. Finally, we would have compromised on extensibility, since adding a new feature would mean to navigate and modify a complex and interconnected codebase.

Therefore, to tackle the problems that the monolithic approach would bring, we decided to separate the different components into services and to use a micro-service approach. What the micro-service approach does is to separate a large application into smaller, independent running services, each of them handling a specific function. More detailed information regarding the containerization platform of choice can be found in section 5.4.1. After careful consideration of all features that our project will contain, we created an initial diagram (Figure 2) that would display how the interaction between the containers would look like. In addition, the diagram shows what data is exchanged between the containers.

Thus, we initially considered the following architecture, containing four services:

- VR service: containing eye tracking, the environment, the avatar, the speech recording and the text-to-speech functionality for the avatar.
- LLM Handler service: responsible for the therapist's response and gesture generation. Besides that, it keeps track of the ongoing conversation and identifies self-harm inputs of the user. More information regarding LLM is in section 5.2.

- **Speech-to-text service**: having the goal of transcribing the user's speech into text, which will be used further for the LLM for a suitable response back to the user. An in-depth explanation of this service's implementation is at section 5.1.4.
- Sentiment Extraction service: which is using a model to identify the sentiment that is expressed in the user's speech. More details about this service can be found in section 5.1.3.

However, during development, we encountered multiple issues with this design. First of all, the VR service needs to be allocated many resources in order to run smoothly. Wrapping it in a container might have slowed the system down, which would be against our goal to run as much as possible in real time. Second of all, having text-to-speech integrated in the VR part would again slow down the system, since it would run on the GPU, which is already occupied by the sentiment extraction service. Thus, we decided to couple the text-to-speech feature together with speech-to-text in the same container, as they both have the same underlying implementation.

The final form of the architecture can be seen in the figure 3. Although the main services, VR and LLM, are still separate, with VR not being containerised anymore, there have been slight changes regarding the services that handle voice input and output. As a result, the STT and TTS functions are now placed together, in comparison with the initial design, in which TTS was embedded in the VR service.



Figure 2: Initial architecture of the containerized system



Figure 3: Final architecture of the containerized system

#### 4.2 System Integration

For the individual system components to be useful, they must be able to communicate with each other, in order to exchange information. For this task, we have chosen to use gRPC, a cross-platform remote procedure call framework built atop HTTP/2. gRPC relies on files known as Protocol Buffers (Protobufs) to define the structure and types of data that may be passed between components. For our system, we pass 3 types of data between the services: strings (ex: the user's voice transcription), raw bytes (containing a WAV format encoding of the user's or the virtual therapist's voice) and a boolean (used to flag a user's message as indicating an intention for self-harm). WAV format raw bytes were chosen as the data type for transmitting voice recordings between containers, since they do not require compression, so they do not run the risk of losing data or degrading performance, like the originally proposed MP3 format (from Figure 2). Once those files are defined, a protocol buffer compiler is used to automatically create classes in Python/C#. These classes handle the actual network communication and provide an easy-to-use abstraction for other code to interact with. This allows for easily calling remote procedure calls serviced by other components, without the developer having to know how the underlying network communication is implemented.

We chose gRPC as a communication framework between our containers due to it being fast, light-weight, easy to integrate with Docker containers, and widely supported across different programming languages and operating systems. Additionally, gRPC has the advantage of being supported by the Google APIs used for STT, TTS, and for our LLM, therefore few additional dependencies are required to access those APIs. Finally, gRPC has the capability of easily sending and receiving streams of data, which we thought may be useful if we choose to optimise our system to handle user input in real-time, by streaming it to the appropriate services.

#### 4.3 Conversation Stages

The therapy session is designed such that it follows 5 distinct conversation stages, one after another. We have chosen to have pre-defined conversation stages in order to structure the therapy and ensure a complete therapy session. We wanted to give the client space to talk openly about their problem, while still keeping the conversation focused on the main goal of the therapy.

- Greeting: This is the initial stage, where therapist greets the client, introduces themselves and familiarizes the client with the environment. The therapist notifies the client that the therapy can be stopped at any point and explains the basics of the controls. Lastly, the therapist asks the client to share their name.
- Introduction: During this stage the therapist asks introductory questions and figuring out the reason for the visit.
- Exploration: Exploration stage is a systematic investigation stage, where the therapist asks follow-up questions, gathering as much information as possible regarding client's issue. This stage consists of 7 exploratory questions which follow one after another. Once all 7 were answered, the conversation moves to the next stage.
- Advice: After all the exploratory questions were asked, the therapist summarizes all the information client shared and pin-points the issue, suggesting a solution to it. In this stage the therapist offers a technique the client could try the next time.
- **Farewell**: Conclusion of the session, where the therapist says goodbye to the client.

#### 4.4 Risk Analysis

The sensitive application domain of our system (mental healthcare) and the well-known inherent risks of the technologies we are going to employ (VR and LLMs), make conducting a risk analysis as part of the initial design

process important. The role of this risk analysis is to identify possible technical, ethical, and medical threats to the success of the project with the aim of ensuring that appropriate mitigation strategies are included in the development process, the high-level system design, and in all implementation choices, to the extent possible. However, it is also important to note that it is impossible for this risk analysis to identify all risks associated with the project, as some risks may only become apparent during the development process or after testing the product with real users. Moreover, not all mitigation strategies that we propose here are feasible within the scope of this project. This subsection will first begin by describing the possible risks of the system and then follow with mitigation strategies for each identified risk. For a discussion of how our understanding of the risks of this project evolved throughout the development process and which risks we have managed to mitigate in the final product, please refer to Section 8.

#### 4.4.1 Possible risks

The following are the main risks we have identified for our system:

- R1. LLM bias: The AI may inherit limitations and biases from training data during either the pre-training or the fine-turning process. That may lead to unexpected responses, potentially harmful to the user.
- R2. LLM Hallucination: LLMs are known to be prone to hallucinations. That means they may invent "facts" which were never included in their input. The result is potentially misleading the user, or making them feel ignored, by claiming that the user has said things they did not actually say.
- R3. Inappropriate Therapeutical Techniques: The LLM may generate responses that do not conform to therapeutical best practices and which may be actively harmful to the user.
- R4. System Limitations in Case of Complex Cases: An LLM may not be sufficiently capable of handling complex psychiatric cases, especially those in which there is a danger of the patient committing self-harm. Additionally, in case such cases are mishandled, the consequences could be severe, including the death of the patient.
- R5. Latency Issues: The VR experience may not feel natural enough due to the latency between the user's speech and the virtual therapist's response being too high.
- R6. VR Cyber Sickness (Motion Sickness): Long therapy sessions in VR may cause disorientation, dizziness, nausea, and headaches.

- R7. User Unwillingness to Engage with an AI Therapist: Users may be unwilling to even consider using a therapist they do not perceive as being human, since they may feel like the therapist could not possibly understand them or relate to their experiences.
- R8. Non-verbal Language Interpretation Accuracy: Misinterpretation of the user's non-verbal language may reduce the usefulness of integrating such cues with the verbal responses.
- R9. Verbal Language Recording: Background noise may significantly reduce the performance of speech transcription and voice sentiment analysis.
- R10. Lack of Comfort Using VR: The user may not feel comfortable enough to fully relax and talk with the virtual therapist due to having to constantly wear the VR headset.

#### 4.4.2 Mitigation Strategies

Based from the risks identified above, we have devised the following possible mitigation strategies:

- 1. Regularly involve psychiatrists in the development process to ensure that the system's output is in accordance with therapeutical bestpractices. (Mitigated risks: R1, R3)
- 2. Clearly inform users that the system may be wrong and that it may sometimes hallucinate. (Mitigated risks: R2)
- 3. Use pre-approved medical AI guidelines for response generation. (Mitigated risks: R3)
- 4. Whenever the user is feeling unsafe or the system evaluates the condition to be critical (such as there being a danger of self-harm), the virtual therapy session should be ended and a human expert should take over. (Mitigated risks: R4)
- 5. Rely on cloud-hosted models as much as possible to free up hardware resources for those processes that must be run locally, thus improving system performance. (Mitigated risk: R5)
- Session duration should be limited or breaks should be recommended after approximately 10-15 minutes of system use. (Mitigated risks: R6)
- 7. Sessions should be held in a place that is quiet and comfortable for the user. (Mitigated risks: R9, R10)

- 8. Make the therapist seem as human as possible by giving it a name, allowing it to perform gestures and providing it with the functionality for lip sync, as well as a natural-sounding voice. (Mitigated risk: R7)
- 9. Always rely on multiple forms of user input, both verbal and nonverbal, for formulating the virtual therapist's response. (Mitigated risk: R8, R9)

Once the high-level design was complete, it was validated with the client, to understand whether our technical choices were in line with therapeutical best-practices, particularly with regards to the conversation stages. Having received the client's approval, we proceeded to implement our design, as we will discuss in more detail in the next chapter.

To conclude, in this chapter we outlined the high-level design of our system, detailing its key components, their interactions, and the overall communication flow. We presented the architecture both before and after development, highlighting the improvements and structural changes made. Finally, we provided an in-depth look into the sequence of interactions between components, clarifying the behaviour of the system at each stage of the conversation. These fundamental design choices define the foundation of the further implementation of our requirements.

# 5 Development

This chapter will cover the technical implementation of each component of the system. The structure of this chapter will roughly follow the flow of data through our system: beginning with the user's verbal and non-verbal input, continuing with the LLM which interprets the user's inputs and produces the virtual therapist's response, and ending with the Unity environment through which the therapist's responses are shown to the user in VR. Finally, we will also go into more detail about how the system's containers are implemented and what we have done to make deployment easier for the end-user.

#### 5.1 Verbal & Non-verbal User Input

Although the focus of our project is to use non-verbal cues to improve the interaction between the patient and the virtual therapist, an important part of communication will always remain verbal. To that end, we record the user's voice and use it as one of the inputs to our system.

#### 5.1.1 Voice Recording

When the user presses the grip button on their left-hand VR controller, the system begins recording their voice for a maximum of 90 seconds, using Unity's built-in Microphone class. The audio input device that is used for recording is always the VR headset microphone, if it is available, since that ensures better audio quality than an external microphone, positioned farther away. If the VR headset microphone is not available, then the system will fall back to using the default microphone configured in the sound settings of the computer linked to the VR system. We have chosen to limit each recorded audio clip to a maximum of 90s, since system response times increase exponentially with the input size. Through experimentation, we have observed that a clip limit of 90s provides a good trade-off between the user's ability to say everything they want to, without feeling rushed, and longer inputs being processed with only a reasonable delay.

When the user decides that they have said everything they wanted, and that they would like a response from the therapist, they may press the grip button on their right-hand controller. The microphone recording then stops and a Unity AudioClip object containing a clip of exactly 90s is obtained. If the user spoke for less than 90s, Unity will automatically pad the recording up to its maximum configured length. Since it would slow system performance down unnecessarily to work with a recording that is longer than needed, we first trim this clip down to its original, unpadded size. We do this by keeping track of the recording's start and end times, then calculating the real duration of the recording by subtracting these two times, and finally using the method for trimming Unity AudioClips included in the WavUtils package[3]. Before the audio is ready for further use, one additional step needs to be taken: encoding the audio clip in a well-known audio file format. For this, we chose the .WAV format, since it does not require any audio compression (unlike MP3, which would slow down performance and lose potentially valuable data), and it is also widely accepted by external libraries and APIs. Unity's audio clips only allow access to the raw audio samples, but these samples do not contain the headers mandatory in the WAV format. These headers (which include information such as the sample rate of the audio and the encoding used) need to be added manually; to achieve this, we rely once again on the WavUtils package. Once the audio recording is in the .WAV format, it is ready to be sent on to the Speech-to-Text (STT) and speech emotion recognition (SER) components.

#### 5.1.2 Blocking service requests & Multithreaded Design

Processing audio in the STT and SER services is relatively slow (a couple of seconds on average). Since Unity runs all code on its main thread, by default, that means that, until we receive a request from the STT and SER services, the main thread will be blocked. This is a problem, because the Unity main thread is also responsible for rendering frame updates, so if the main thread is blocked, then no frame updates will be generated for a few seconds, making the application hang and become completely unresponsive. For reference, 60 frame updates per second should normally be generated to give the appearance that the application is running smoothly.

To solve this issue, we first tried to use the C# asynchronous programming model. This model allows methods which need to communicate over a network to be started, and only awaited later in the program's execution, once their results are necessary and once those results have (hopefully) been received in the background. While a method is awaiting results from another method, it will relinquish control to its caller, such that the thread it was running on can continue to do other work in the meantime. Figure 4 shows how we initially planned to use the C# asynchronous programming model for solving the issue of microservice requests blocking the main thread. This would have been the most light-weight solution to handle our problem; however we found it to be incompatible with Unity code.

That is because the asynchronous method would have had to be called from a Unity Update() method. The Unity Update() method is called once before each frame is rendered from the Unity engine backend. The problem with this is that the next frame cannot be rendered until an update method finishes. So if an update method calls an async operation and it has nothing else to do after that operation, it will not return. It will also not allow the frame to be rendered and then handle the result of the async operation on a subsequent call of the Update method. Instead, it will block and wait for the async operation to be finished before returning, as such making the call



Figure 4: Idea for implementation using C# asynchronous programming

to the STT and SER services still be synchronous in reality. To be more specific, we have the following call stack: Update()  $\rightarrow$  HandleUserInput()  $\rightarrow$  STTRequest(). While HandlerUserInput() was waiting for a response from STTRequest(), we would have wanted Update() to continue running. However, although the Update() method functions like the contents of an infinite loop, you cannot share information between two different calls of the Update() method. Therefore, the Update() method would block, waiting for HandleUserInput() to receive a response from STTRequest(), instead of allowing a future iteration of Update() to use the results of HandleUserInput(), as the C# asynchronous programming model intended. Figure 5 gives a graphical representation of the issue with asynchronous programming in Unity that we've just described.

To solve the challenge of Unity not integrating the C# asynchronous programming model well, we decided to implement a multithreaded design



Figure 5: The problem with asynchronous programming in Unity

(see Figure 6). With this design, the entire handling of the user's voice recording, after the pre-processing described in section 5.1.1, is done on a background thread, while the main thread is freed to continue rendering frame updates. We considered several technical alternatives for the creation of this background thread. The option we selected in the end was to use Unity's job system. We chose this system over regular C# threads run via .NET's Task.Run() method because Unity jobs have been observed to be faster than regular C# threads[4], as they are executed on existing Unity worker threads instead of creating new threads. However, using Unity jobs does come with the drawback that any objects passed to them are not automatically garbage-collected, so manual memory management is necessary to prevent memory leaks.

Although using a background thread solved our initial problem, it also introduced a new challenge: background threads are forbidden from calling any Unity API methods, since they are not thread-safe. That meant that we could not directly play the therapist's response from the background thread, once we received it from our microservices. To address this issue, we implemented the dispatcher design pattern, which is a common pattern in UI and Unity programming[5]. This pattern involved the existence of a dispatcher class that maintains a list of actions to be executed on the main thread. A method running on a background thread can add actions (i.e. function calls) to this list at any time. Periodically (in our case before each frame update), the main thread checks the dispatcher's list and executes all actions that were in the list. This way, the background thread can add a "Play therapist's response" action to the dispatcher queue, and, once the main thread is ready to render the next frame, the main thread will also execute that action, thereby bypassing the restriction on background threads calling Unity API methods.



Figure 6: The current multithreaded design

#### 5.1.3 Speech Emotion Recognition (SER)

The speech emotion recognition service receives the .WAV format recording of the user's voice and is responsible for identifying what emotion was present in the user's voice as they spoke. This service relies only on the tone of voice the patient used, not on the meaning of what they said.

To achieve this, we use a pre-trained model based on the Wav2Vec2 architecture introduced by Meta[6]. The model we use was acquired from Hugging Face and had been fine-tuned on the RAVDESS dataset, specifically for SER[7]. The RAVDESS dataset contains voice recordings of 24 professional actors speaking the same 2 sentences in 8 different emotions[8]. These 8 emotions are also the emotions that our SER service supports: 'angry', 'calm', 'disgust', 'fearful', 'happy', 'neutral', 'sad', 'surprised'.

Before settling on this SER model, two other alternatives were considered. The first was training our own model, however we decided against this, since we found suitable pre-trained models online and we considered that, to respect the time constraints and scope of our project, it would be more time-efficient for us to use a pre-trained model. The second alternative was to use one of the pre-trained models available in the "Speech Emotion Recognition" project on GitHub[9]. We chose the Hugging Face model instead, thanks to its more detailed documentation and simpler interface through the Transformers library, which shortened development time and made future changes easier to implement.

#### 5.1.4 Speech-To-Text (STT)

The speech-to-text service is responsible for transcribing the user's voice. For this, we rely on the Google Cloud Platform (GCP) Speech-To-Text API[10]. We chose this API over a local STT model, since STT and SER are expected to be performed concurrently. Trying to simultaneously use two large models locally, while also running a VR environment, would significantly increase hardware requirements. Specifically, we would need more than 16GB of RAM and more than 1 GPU, which is not a configuration that is common for consumer hardware. Therefore, a cloud-based STT model has the important advantage of reducing the hardware requirements of the system. We chose GCP over its competitors (such as AWS or Azure), thanks to its generous free usage policy and our team's better familiarity with Google services, which helped reduce development time.

When configuring the API, two parameters were important: the type of STT model and the geographical location. We found that the 'long' model performed best for our system, since it was better able to handle pauses in speech and filler sounds (ex: 'uhmm', 'errr') than the other two models tested: the 'short' model (which would stop transcribing after a pause in speech) and the 'telephony' model (which generally produced less accurate transcriptions on audio recorded from the VR headset). The location of the Google server used to provide the STT service was also important, to reduce latency and to ensure that sensitive user data is not transferred outside the EU. For our system, we chose the 'europe-west1' endpoint, which links to servers physically located in Belgium, because this was the closest endpoint that supported the 'long' model and was also located in the EU.

#### 5.1.5 Eye Tracking

To enhance the experience and fulfill the client's goal of exploring the possibilities of tracking non-verbal cues, we considered that eye tracking could provide valuable information regarding how the patient feels. Thus, during the speech, the patient's eye movement is recorded and used in the input for the LLM.

For this, we decided to use the "Eye Gaze" library offered by Meta [11], by adding a building block. To be more precise, a building block is a Unity extension that adds features to the project, using Meta XR SDK, such as controller tracking[12], hand tracking[13] or virtual keyboard[14]. Each building block focuses solely on only one feature of the Meta Quest, so the developer can use specifically the headset's features that they need. The advantage of using this type of building block is that all the features dependencies are installed automatically, as stated on Meta's documentation available online [15].

After importing the "Eye Gaze" library into our project, we needed to attach it to an object in the scene. The object of choice was the camera rig, which is the object that captures what we see in the scene in real time. Since we wanted to replicate as closely as possible the natural interaction of a patient with their therapist, we wanted to be able to follow with our eyes the objects that are in the room, such as the walls, the ceiling, the floor or the therapist itself. Although there are pieces of furniture in the room, they are not tracked, since we are interested in seeing whether the patient is looking directly at the therapist, or avoiding them.

Regarding the implementation of the eye tracking, we used the "Collidable" property of native objects in Unity. This is a necessary property for an object in order to be tracked by an imaginary ray coming from the eyes. In case of the avatar, after checking the settings, we observed that it is not an object of type "collidable", which meant that we needed a workaround in order to become visible for the eye tracking. Thus, we placed an invisible sphere right behind it, so that it would simulate the therapist's head. The eye movement is recorded as long as the user is speaking. After the speech is ended, the algorithm computes the most watched object. The result can be one of the following: either one of the vertical walls, the floor, or the ceiling, or the therapist; then the result is passed to the LLM. Further, it is up to the LLM to interpret the result, together with the speech and the sentiment extracted from the text.

### 5.2 LLM

Our virtual therapist system's LLM is its main reasoning component; it processes inputs to produce responses that are appropriate for the context. This service coordinates the relationship between therapeutic dialogue generation, conversation state management, and non-verbal cue analysis.

#### 5.2.1 Model Selection

For our large language model implementation, we selected Google Gemini Flash 2.0 through several key considerations:

- **Cost-effectiveness**: The model is currently available free of charge through Google's AI Studio, making it ideal for research and prototype development.
- **Context window**: With support for up to 1 million tokens, the model can maintain exceptionally long conversation histories without losing context crucial for therapeutic sessions that may span multiple interactions.
- **Performance**: The "Flash" variant is specifically optimized for rapid inference via API calls, typically responding in under 2 seconds for our use case.

### 5.2.2 Input Processing

The LLM container receives gRPC requests containing three key data points:

- **Eye movement**: Relative eye position where the client is looking in relation to the virtual therapist.
- Voice sentiment: Emotional tone analysis derived from the speech patterns.
- Transcribed speech: Textual content of the client's verbal input.

A dedicated service first processes these inputs, combining the transcribed speech with the non-verbal cues and formatting them as "<nonverbal cues>the user's speech." This combined representation provides important context for therapeutic interaction by capturing both *what* and *how* the client said it.

Example: <Sadly, avoiding eye contact>I am doing great.

#### 5.2.3 Therapeutic Response Generation

The formatted input is then sent to Google Gemini Flash 2.0 LLM, which generates responses according to a structured therapeutic protocol. The LLM prompt produces three distinct outputs:

- 1. Verbal response: The primary therapeutic dialogue to be delivered to the client;
- 2. **Gesture recommendation**: Suggested non-verbal behavior for the virtual therapist avatar;
- 3. Conversation stage update: Progression marker for the therapeutic session.

Example: response: Could you tell me more about that feeling? gesture: look down stage: EXPLORATION

#### 5.2.4 Prompts

The LLM functionality has 4 prompts:

- 1. Non-verbal cue interpretation: The main purpose of this prompt is to navigate the LLM to focus on non-verbal cue interpretation. Its goal is to deduce from the user's speech and non-verbal cues, how the user spoke. It receives the user's speech transcription, eye-position and voice sentiment. The response it gives is of this format: "<nonverbal cue interpretation>the user's speech transcription". This provides more elaborative information for the future input to the LLM.
- 2. **Therapist response**: This prompt instructs the LLM to behave like a therapy specialist. It specifies that they are a psychotherapist named Celine and that they specialize in treatment of anxiety. The model is instructed to avoid references to the specialists, no matter what the input is. The LLM is prompted on the general behavior that is expected from the therapist at each stage of the conversation. Lastly, the model is provided with the current conversation summary, last 2 turns between the therapist and the user, exploratory questions that were already asked (since they should not be repeated). This ensures that the model responds accordingly and never breaks the character. The LLM is prompted the techniques that should be used, which are: Active Listening, Cognitive Behavioral Therapy, Mindfulness, Motivational Interviewing, Validation, Gestures and Tone. As a result, the LLM is prompted to return a JSON-formatted output that includes the therapist's response, the accompanying gesture (e.g., nod, smile, look down), and the updated conversation stage.

- 3. Self-Harm Detection : In this prompt the LLM is instructed that it is a psychology specialist who is capable of identifying extreme situations in user's input. It focuses on detecting self-harm, suicide or murder intentions in the user's input. As an output it returns "True" when the input contains any of the above mentioned intentions, and "False" otherwise.
- 4. Summary generation: The LLM is prompted as an assistant, whose task is to help the psychotherapy follow the conversational order. It commands the LLM to focus on the most important aspects of the conversation and from that rewrite the summary of the ongoing conversation. It keeps track of how many exploratory questions were asked and updates those. It expects in the user input to have current conversation summary, last 2 turns between the therapist and the client, number of exploratory questions asked as well as the questions themselves. As a response it returns an updated conversation summary, an updated number of exploratory questions asked as well as the questions themselves.

The prompts were engineered in progressive manner, starting with initial goal of each prompt, adding specifications to those during testing. Referring to the "Therapist response" prompt, few-shot prompt was used in order to pay attention to the specificity of therapist's responses, which could include a simple word that reiterates the user's input.

You can find prompts in the appendix C.

#### 5.3 VR Environment

#### 5.3.1 VR Device Setup

For our project, the headset of choice was the Meta Quest Pro [16]. The main reason for choosing this specific version was the eye tracking feature, which is described in the next subsection. Additionally, we wanted to explore the possibility of tracking hand gestures, which was a feature available on this model of headset. Moreover, in order to run the environment, we needed powerful hardware, and especially having a GPU, for rendering the whole 360° environment. After researching the available options, we used two laptops provided by Interaction Lab [17], namely a Lenovo Legion 5 Pro 16ACH6H [18] and a Lenovo Legion Y540-15IRH [19].

The development environment of choice was Unity, as there are plenty of tutorials available on the internet and also documentation and libraries needed for the back-end. These were needed at the beginning in order to get accustomed to the editor, since none of the team members had any experience with game or VR development before. Furthermore, Unity can be integrated with diverse packages, ranging from the technical ones, such as Meta XR, to the ones needed for the office decoration, that are presented extensively in section 5.3.2.

#### 5.3.2 Scene Setup

For the scene setup, we wanted to create an environment close to that seen in the office of a traditional human therapist. We initially started with a simple plane, so that any objects that we would place in the scene would not fall into the void. When setting up the scene, the directional light was already set in the scene, as this type of lightning is placed by default by Unity [20].

To allow for interaction with the VR environment, we set up an XR Interaction Setup object. XR, coming from extended reality, is an umbrella term to refer to augmented reality (AR), mixed reality (MR), and virtual reality (VR)[21]. This object is the entry point for setting up the XR Interaction Toolkit [22], ensuring that the scene is properly prepared to process XR inputs and interactions efficiently. This toolkit enables the user to interact with objects in the scene, by hovering, selecting and grabbing objects, by offering support for creating basic canvas UI interactions with XR controller; most importantly for our project, it provides a VR camera rig, which is essential for enabling full immersion in the scene.

The next object that can be found in the scene is the Controllers Button Mapper, along with the ControllerInputManager. The first object is implemented using a Building Block offered by Meta. We decided to use this option for the interaction of the user with the environment as it was the least complex way to tackle it, for both developers, as well as for the users. The advantage of using this option is that we could easily attach scripts to the desired buttons, which was effortless to set up and to test. The role of ControllerInputManager is to gather all scripts related to buttons in one place, allowing us to have the code organised and maintainable for future development. Thus, the ControllerInputManager contains currently the scripts for starting audio recording, stopping it and exiting the session. The Controller Button Mapper assigns to the aforementioned scripts the following buttons: the left grip button, the right grip button and the right trigger button, respectively, as shown in figure 7.

When building the office, we took into consideration what elements could make the space as realistic as possible. Thus, we first started with placing walls, in order to delimit the office. For this, we used four / five cubes, with the width of 0.8, in order to make them thin. We opted for having these cubes, as they are primitive objects in Unity and have the BoxCollider property by default [23], which is necessary for the eye tracking component, as discussed in section 5.1.5.

We furnished the office using asset packages from the Unity Asset Store [24], selecting only free options due to budget constraints. Since the avatar



Figure 7: Mapping of user actions to controllers' buttons

is in a sitting position throughout the session, we placed a sofa in the office, from the HQ Modern Sofas Pack [25]. For the floor, we added a texture similar to wood flooring, from the AmbientCG website [26]. In order to give the therapist a more knowledgeable look, we considered having a bookshelf with books in the background, thus, for the bookshelf the asset pack of choice was the DS Furniture Pack 01 [27], while for the books, we used the QA Books pack [28]. One helpful feature of the books pack was that it already had multiple books packaged together, which saved effort from placing each individual book on the shelves. To make the office more lively, we chose to add a plant, taken from the package Plants [29]. For more light, we added a lamp next to the bookshelf, imported from the Customizable Lights and Candles [30]. In addition, we placed a desk and an office chair in the corner, taken from the Georgian Office pack [31] and Apartment Kit pack [32]. Finally, in case the therapist wants to have a session with a patient in a different space, they can do it in the two armchairs, with a table between them two, taken from the Furniture Free pack [33].

Thus, the final office set-up looks like this:

#### 5.3.3 Avatar Choice

In our choice for a humanoid avatar, we took into consideration the following factors: the avatar should have a pleasant appearance, kind facial expressions, and natural body positions. We experimented with a couple different assets - the UMA 2 Unity package [34], the Meta Avatars SDK [35], and the ReadyPlayerMe avatars [36]. In the end, we chose to use the ReadyPlayerMe avatars, for their high customization potential, and the presence of the skeleton (rig), as well as the facial blendshapes, which allow for natural facial expressions.


Figure 8: Celine's Office



Figure 9: Celine's desk and a coffee table with chair

#### 5.3.4 Avatar Animations

Body animations for the therapist avatar were implemented using animations acquired from Mixamo, in the .fbx format. For the avatar, the following animations were considered relevant: sitting - which has the role of the 'idle' animation, thinking, and nodding. Animation files were attached to specific animation states in the Unity animator, and transitions between the animation states is ensured through Unity variables, specifically boolean flags. For example, to transition from the idle state - sitting - to the nodding state, the boolean 'nodding' is set to true. The reverse transition, from nodding to the idle state, is occurring when the same boolean value is set to false.

In our system, the LLM is responsible for interpreting user input and determining the appropriate avatar animations. Upon processing the user's dialogue and non-verbal cues, the LLM returns specific command strings — for instance, 'nodding' or 'thinking'. These commands are then captured by a C# script in Unity, named AnimationHandler.cs. The script parses the incoming string and maps it to corresponding boolean variables (e.g., the boolean named 'nodding'). When the boolean is set to true, Unity's animator component detects the change and transitions the avatar into the respective animation state, such as a nod or a thinking pose. This workflow ensures that the verbal output from the LLM is directly translated into visual cues.

In order to ensure that the nodding and thinking animations only affect the upper body or the head of the avatar respectively, while maintaining the sitting position with the lower body, we used Unity animator masks. In Unity, masks allow to restrict animations to specific parts of a character's skeleton, also named 'rig'. This means that we can control which bones or body parts are affected by a given animation layer. For example, we could have one animation playing on the upper body (like waving) and another animation controls the lower body (like walking), keeping them independent but synchronized. This selective control is useful for blending multiple animations and achieving more complex, layered character animations.

The following masks were included in the project, and different weights were assigned to each, to ensure that animations are rendered correctly:

- Head Mask Layer highest weight. Action rules in this mask layer are rendered above all other mask layers.
- Upper Body Mask Layer high weight. Action rules in this mask layer are rendered above the actions in General Mask Layer.
- General Mask Layer mask present by default in Unity. Lowest weight between all utilized mask layers.

Utilizing the Upper Body Mask Layer allowed for the thinking animation to occur, while maintaining the sitting lower body position. Moreover, the Head Mask Layer allows the avatar to perform the nodding animation at the same time with the sitting animation (the idle animation), but also while performing the thinking animation.

Face expressions were implemented using .mhpose files, which contain a list of key-value pairs corresponding to values of facial blendshapes in the avatar. Facial blendshapes, also named morph targets, allow us to control and combine various deformations of a character's face to create realistic emotional expressions.

To formulate the files, we manually adjusted the facial blendshapes in the Unity editor to reach the desired face expression, and the values were written in the .mhpose file. The avatar supports a smiling expression, as well as a sad expression. Similarly to body animations, face expressions are triggered through a C# script, and the LLM is instructed to return specific commands that will trigger the activation of the face expressions.

Although there exist alternatives to implementing face expressions, such as utilizing complete .mhpose files obtained from online resources, this method would require precise translation of the facial blendshapes nomenclature and values from the source file to that of our avatar. This is because in the past, Unity controlled facial blendshapes using a 0-100 range, where 0 meant no effect, and 100 represented full deformation, while more recent Unity versions support a range of 0-1. Moreover, depending on the avatar configuration, facial blendshapes can have different names representing the same blendshape, or some avatar configurations can have a different number of blendshapes than others. Therefore, we concluded that manually creating the expression files is the preferable method.

Aside from interactions through body animations and face animations, the avatar also directly interacts with the user through lip syncing.

Lip syncing was established as a critical feature of the avatar since the outset of the project, since it ensures proper and life-like communication between the avatar and the user. The Oculus LipSync Unity package [37] is used for implementing the feature. After importing the Oculus LipSync package in Unity, a new avatar configuration is created from the Ready-PlayerMe menu in the Unity editor, to which the respective Oculus visemes are added. This ensures that the avatar has the visemes necessary for Oculus LipSync to map specific sounds to corresponding visemes representing mouth movements. Visemes define the basic mouth and facial shapes associated with specific sounds, and they're essential for achieving realistic lip-sync animation. When the avatar speaks, visemes are used to drive the corresponding mouth movements so that the lip motions match the spoken audio.

The Oculus LipSync functionality is connected to the avatar through the AvatarController Unity object, to which two scripts from Oculus LipSync are attached: OVR LipSync Context, and OVR LipSync Context Morph Target. The avatar features a skinned mesh renderer that is attached to the latter script from Oculus. Oculus LipSync automatically generates a Unity audio source field, to which we attach the audio source we created. Therefore, the lip syncing functionality is automatically integrated with the Unity audio source we created.

#### 5.3.5 Text-to-Speech (TTS)

Once the LLM has generated a response, this response is sent to the Textto-Speech (TTS) service, in order to obtain a voice for the therapist. For this, we make use of Google's TTS API[38]. This API provides easy access to multiple TTS models, with varying degrees of naturalness and latency. Of these, we have chosen the "en-GB-Chirp-HD-O" model, a model which produces a middle-aged female voice with a generic British accent. This model was chosen as we found it the most natural and empathetic sounding of those available, while also having low latency.

We chose to use an API for TTS, instead of running a model locally, for the same reasons we chose an API for our STT service. That is, using a cloud-based model reduces the hardware requirements of the system and makes it useable on commonly available consumer devices.

We chose the Google TTS API instead of any other publicly available TTS API, since, at the point in our development process when we started building the TTS service, we had already chosen to use Google APIs for the LLM and the STT service, so we were already familiar with the way Google APIs worked and how they could be called from C#, which reduced the development time and helped us stay within the project time constraints.

### 5.4 Containerization and Deployment

#### 5.4.1 Containerization

To implement the containerized architecture mentioned in Section 4.1, we chose to use Docker.

Docker is a platform that is used for containerization in software development, by bundling an application's code with all the files and libraries it needs to run on any infrastructure [39]. Normally, when running any projects on a local system, each person of the team needs to install different libraries and packages. By using Docker, this step is no longer needed: the image can be pulled from the repository or from Docker Hub and immediately run.

We decided to use Docker for multiple reasons:

• **Portability:** Since team members use various platforms, namely Windows, Ubuntu and macOS, and the processors used by our machines are all different, we used Docker images to be able to run the code effortlessly. This also helps with distributing the system to the end client, as they do not have to care which platform we developed our

system for or to install different executables depending on their operating system.

- Separation of Concerns: We adopted this approach by encapsulating each service within its own Docker container. This ensures that these components can be developed in different programming languages or using different versions of the same libraries, and then run independently from each other, helping with debugging, testing, and maintenance over time.
- Scalability: In case the project will be extended and launched in the market, using containers will allow efficient deployment of multiple instances of the client, to multiple customers.

To create a docker container, one needs to first define a dockerfile, in which all operations needed to create the container are defined. The first of these operations is installing a base image. The base image contains the operating system that will be used for the container, as well as some pre-installed libraries necessary to run our code.

For the containers running python code (the speech emotion recognition container and the LLM container), we chose to use a "python-slim" base image[40], which is an image based on Linux Debian, which only contains the minimal Debian packages required to run python. We chose this image over a full Debian image, because of its lower storage space requirements and because we did not need to run any other program other than python in those containers. We also chose this image over an image based on Linux Alpine (a very light-weight distribution of Linux), because our code required some Nvidia CUDA dependencies for GPU hardware acceleration which were not available on Linux Alpine.

For the STT/TTS container, we chose the the .NET SDK 9.0 image, an image also based on Linux Debian, which, in addition to the base Debian packages, also comes with the .NET runtime we needed to run our microservice.

When we first started to implement the micro-service architecture, our procedure to actually run the system by running each service individually proved to be time-consuming and inefficient. The solution for this was to implement Docker Compose. Using a .yaml file, we defined all services that we wanted to run, what secrets those services required (such as an API key) and what resources (such as access to a GPU). Then, all services could be run with only one command (*docker-compose up -pull always*). What happens in the background is that all of the container images are pulled from Docker Hub, so that always the latest one is run, and then each container is started.

#### 5.4.2 Packaging & Deployment

Correctly installing, building, and configuring our project from the source code is complicated, time-consuming and requires a good understanding of the underlying system. Since our client does not have a Computer Science background, it would be unreasonable to expect them to go through this process in order to use our project. Therefore, we have decided to package our project, such that it can be run with a single click. We believe that having a single, simple starting point for the entire system will make it significantly more user-friendly, improving the chances that our system will be adopted by the client for further use in their research.

To achieve this, we have built the Unity project into an executable, uploaded the Docker containers to Docker Hub, and, finally, written a script that combines the two. This script downloads the latest versions of the Docker containers from Docker Hub, starts the containers (or, if they are already started, restarts them), waits for the containers to fully boot up, and, once they are ready to receive requests, starts the Unity project executable.

Finally, to further improve the client's ease of use, we have compiled a user guide detailing the steps needed to set up, configure and run our system. This manual has been included in Appendix A.

In this chapter we covered the technical implementation of the system's components, starting with the user's verbal and non-verbal input, followed by the LLM implementation in the whole system and ending with the way in which the responses are shown back to the user. At the end, we explained in more detailed the micro-service architecture and the packaging of the entire application.

# 6 Testing

This chapter will describe what we have done to check our system's compliance with the requirements and its overall quality. We will begin by describing our testing strategy (including the objectives of testing and the considerations for adopting our two main testing methods), then we will continue by presenting our system tests, followed by the methodology for our user tests. The results of all tests will be presented in Section 7 - Evaluation.

### 6.1 Testing Strategy

The testing strategy for our project had to take into account two fundamental facts:

- 1. The project was experimental: One of the main goals of the project was to not simply produce a system that works, but also to try different approaches and understand the trade-offs between them. Because of this, changes in the architecture and the expected behaviour of each component were predicted to be very frequent from the beginning of the project.
- 2. Our system is only intended as a prototype: The client clearly indicated from the first meeting that our system was meant to be discarded. Its purpose is to inform them of whether a virtual therapist would be feasible, and to discover what issues would come up when developing such a system. It is, however, not meant to be used with real patients or to form the basis of a future system that would be used with real patients.

Keeping in mind these basic facts, we determined that testing for our project should have two main objectives: to ensure that the system complies with its requirements and minimizes its possible risks without any major flaws, and to provide useful information about the way users interact with a VR therapist, that may help our client make a more informed decision on the feasibility of such a system. Due to the project's nature as a prototype, and its limited time constraints, finding and eliminating all flaws is considered out of scope for us, as long as those flaws do not meaningfully impact the functionality or usability of the system.

Based on the considerations outlined above, we have decided to employ two different types of tests: system tests (to verify the first objective) and user tests (for the second objective). Two other (interrelated) types of testing were considered: unit tests and automated regression testing. We have decided to keep only the two types of tests mentioned above for the following reasons. Unit tests are not a priority, because most "units" used in our project were not made by us. Instead, what is important to test is whether the integration of different components is working as intended, which is what we achieve through system tests. As for automated regression testing, since the project is not meant to be used as a basis for further development, this type of test testing would also be of little use. Due to the frequent changes in architecture and expected behaviour, it would also add significant overhead to our own development process, as tests would have to be rewritten very frequently.

The following subsections will present our methodology for both system tests and usability tests in more detail, while the results of the tests will be described in Section 7.

#### 6.2 System Tests

#### 6.2.1 System Test Methodology

The system tests were constructed primarily to mimic conversation scenarios between the therapist and the user. Their purpose is to test the overall functionality of various components as a whole and to ensure uniform communication between them. The test cases include simple interactions between therapist and user, and the verification of specific features is achieved through observation within the environment or by reviewing logs, which claim that the system is functioning as expected.

### 6.2.2 Core Functional Tests

# 1. Voice Input Processing

- Test Scenario: Verify recording starts on button press.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say, "Hello, I am Alex.".
  - (e) Press the "stop recording" button.
- **Expected Outcome:** The recording is processed, sentiment is extracted, and the STT successfully converts speech to text.

#### 2. Sentiment Analysis Accuracy

- **Test Scenario:** Verify that the user's sentiment is extracted correctly, based on the emotion they express.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.

- (c) Press the "record" button.
- (d) Say a sentence in a happy tone.
- (e) Press the "stop recording" button.
- (f) Look into the input sent to the LLM that the sentiment matches the one expressed.
- **Expected Outcome:** The Sentiment extractor correctly identifies the emotional tone of the speech.

### 3. Eye Gaze Tracking

- **Test Scenario:** Check that the place most looked at in the environment is correctly identified.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) For the entire recording duration, look at the floor, with the middle of the field-of-view pointing to the floor.
  - (d) Press the "record" button.
  - (e) Say any sentence.
  - (f) Press the "stop recording" button.
  - (g) Check that the output fed into the LLM matches the direction of the gaze.
  - (h) Repeat steps d-g while looking at the left wall, the therapist and the ceiling.
- **Expected Outcome:** Majority voting correctly stores the most frequently looked-at area.

### 4. LLM Response Generation

- **Test Scenario:** Verify LLM responds as a therapist and does not break character.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Ask "Who are you?".
  - (e) Press the "stop recording" button.
  - (f) Await for the therapist to start speaking.
- **Expected Outcome:** The Therapist introduces themselves as a therapist and does not break character.

### 5. STT Output

- **Test Scenario:** Check that the user's input is correctly converted to text.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say a sentence.
  - (e) Press the "stop recording" button.
  - (f) Check the logs.
  - (g) Repeat steps c-f twice, saying different sentences each time.
- **Expected Outcome:** The system correctly transcribes the recorded speech into text.

### 6. Therapist Avatar Presence

- **Test Scenario:** Verify that the therapist's avatar is present in the scene.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Observe the scene.
- Expected Outcome: The scene has a humanoid therapist.

### 7. Self-harm alert

- **Test Scenario:** Verify that the system shuts down in case the user mentions a desire for self-harm or suicide.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say, "Hey, I feel like killing myself".
  - (e) Press the "stop recording" button.
  - (f) Observe the system behaviour.
- **Expected Outcome:** A thinking bubble will disappear after a short period of time, after which the system will automatically shut down.

### 8. Forced termination

- **Test scenario:** Ensure that the system closes when the user presses the "close session" button
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "close session" button.
  - (d) Start the system.
  - (e) Put on the headset.
  - (f) Press the "record" button.
  - (g) Press the "close session" button.
- **Expected outcome:** The system will forcefully close.

## 6.2.3 Interaction and Animation Tests

#### 1. Button Press Functionality

- **Test Scenario:** Verify that the recording is robust to asynchronous user inputs.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Press the "stop recording" button twice.
  - (e) Await the therapist's response.
  - (f) Say, "Hello, I am Adam.".
  - (g) Press the "record" button twice.
  - (h) Press the "stop recording" button.
  - (i) Await for the therapist's response.
  - (j) Say, "Hello, I am Andrew.".
  - (k) Press the "record" button.
  - (l) Say, "Hello, I am Alex.".
  - (m) Press the "stop recording" button.
  - (n) Await for the therapist's response.
  - (o) Check whether it contains only the name Alex.
- Expected Outcome: The system only records when "record" is on and stops when the "stop recording" button is pressed, after it has been started.

### 2. Thinking Animation Trigger

- **Test Scenario:** Verify that the avatar starts thinking about a response.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say "Hello, I am Alex.".
  - (e) Press the "stop recording" button.
  - (f) Look to see the change in the therapist's posture, signalling that it is "thinking" for a response, and a thought bubble is floating above their head.
- **Expected Outcome:** The thinking animation and thought bubble appear while the AI processes the response.

#### 3. Lip Sync and Head Movements

- **Test Scenario:** Verify that therapist's response is followed with lip sync.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say "Hello, I am Alex.".
  - (e) Press the "stop recording" button.
  - (f) Await for the therapist to start speaking.
  - (g) Observe lip-synced speech.
- **Expected Outcome:** The avatar's speech is followed with lip sync.

### 4. Idle Animation

- **Test Scenario:** Verify therapist avatar transitions to idle state after speech.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Say "Hello Alex".
  - (e) Press "stop recording" button.

- (f) Await for the therapist to start speaking.
- (g) Observe therapist's behaviour when they stop speaking.
- **Expected Outcome:** The avatar correctly transitions to an idle state.

### 6.2.4 System Stability and Performance

### 1. Response Latency

- **Test Scenario:** Verify the time it takes for the therapist to think is no more than 10 seconds in case of 15 seconds long input.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Speak for 15 seconds.
  - (e) Press "stop recording" button.
  - (f) Observe therapist's thinking time.
- Expected Outcome: The time interval between the audio recording was stopped and the therapist starts speaking is at most 10 seconds in a scenario of 15 seconds long input.

#### 2. Whole therapy session

- **Test Scenario:** Verify that it is possible to have a whole therapy session from the greeting to the farewell.
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Greet therapist.
  - (e) Press the "stop recording" button.
  - (f) Await until the therapist starts speaking.
  - (g) Await until the therapist finishes speaking.
  - (h) Press the "record" button.
  - (i) Respond to the therapist accordingly, continuing the conversation.
  - (j) Press the "stop recording" button.
  - (k) Await until the therapist starts speaking.
  - (l) Await until the therapist finishes speaking.

- (m) Repeat steps h-l until therapist farewells, mentioning "Good bye", "See you" or other phrases representing farewell.
- Expected Outcome: Therapist farewells the client by stating "Good bye", "See you" or other phrase representing farewell.

#### 6.2.5 Edge Cases

### 1. Silent Input Handling

- **Test Scenario:** Verify that the therapist gives a response in case of an empty user input
- Test Steps:
  - (a) Start the system.
  - (b) Put on the headset.
  - (c) Press the "record" button.
  - (d) Stay silent for 10 seconds.
  - (e) Press "stop recording" button.
  - (f) Await for the therapist to start speaking.
  - (g) Observe therapist's behaviour and output.
- **Expected Outcome:** The virtual therapist responds verbally, accompanied by natural gestures and synchronized lip movements.

### 6.3 User Tests

Considering the novel combination of non-verbal language tracking with an LLM, and how sensitive its application domain (mental healthcare) is, we considered that user testing was crucial for our system. Our objective, when conducting the user tests, was to answer three core (research) questions.

- 1. Does the system meet its functional and non-functional requirements in a real-world setting?
- 2. Are users comfortable interacting with our VR therapist?
- 3. Can users feasibly get help from our VR therapist for an anxiety-related concern?

The user tests were only meant to provide qualitative data. Recruiting enough participants to provide generalisable quantitative data was not feasible due to the time constraints of the project.

Each user test consisted of the following stages:

1. **Informed consent**: All test participants were presented with the informed consent form in Appendix B and asked for their consent to participate in the test after having read it, since we believed a

well-structured informed consent process was important in light of the potential risks of VR use. We also verbally stressed that they should under no circumstances participate if they suffer from photo-sensitive epilepsy, and that they should take off the headset immediately if they feel nauseous.

- 2. Scenario: All test participants were given the same fictional scenario to follow, to avoid the possible negative side effects and privacy issues that could be caused by talking about their real issues. An additional benefit of having a single scenario for all participants is that results are more easily comparable between participants, since they went through similar experiences. For all of them, that scenario was the following: "You have anxiety when speaking in public. When you are speaking in front of large groups of people, your heart starts pounding, you start sweating and shaking. Because of these problems, you have come to see a therapist today."
- 3. **Controls**: The participants were shown what buttons to press in order to start speaking, stop speaking or immediately exit the session.
- 4. Using the system: The system was started and participants were asked to complete a therapy session with Celine.
- 5. **Interview**: a semi-structured interview was conducted with each participant based on the questionnaire in Table 1. Follow-up questions were asked where necessary to understand the participant's reasoning behind an answer.

Most of the interview questions in Table 1 were written in order to verify if a functional (FR) or non-functional (NFR) requirement without quantitative assessment criteria was met. The rest contributed to answering the remaining research questions or simply to finding possible ways to improve our product that we may not have thought of.

When phrasing the interview questions, care was taken to ensure that they are phrased as neutrally as possible, so that participants are not biased towards any particular answer. Before conducting the user study, the interview methodology and questions were validated with the client, to find out if there is anything else he would like to learn from the user tests whether

Index	Interview Question	Tested Re-	
		quirement	
Q1	How was it?	N/A	
Q2	Was the conversation natural to you?	FR4	
Q3	Did you feel comfortable sharing your	N/A	
	experiences with the VR therapist?		
Q4	Did you feel heard while sharing your	FR10	
	experiences?		
Q5	How appropriate do you think the	FR6	
	therapist's verbal responses were in re-		
	lation to the conversation's context?		
Q6	Did the gestures of the therapist	FR13	
	match the context of the conversa-		
	tion?		
Q7	Was the experience immersive	N/A	
	enough?		
$\mathbf{Q8}$	Did this VR experience make you feel	NFR4	
	nauseous?		
Q9	Does the weight of the headset impact	N/A	
	your willingness to participate in the		
	future in a VR therapy session?		
Q10	Would you change anything regarding	N/A	
	the overall interaction?		

Table 1: User Interview Questions

there are any questions that were included by us in the interview, but would not produce any useful information for him. As a result of this consultation, Q7 was added, and a question asking the participants whether they felt the need for a human therapist was removed, since the client explained that people would be unlikely to feel the need for a human therapist if they were not talking about actual mental health concerns that they feel.

To conclude, this section presented how we decided to approach testing in our project. The choices of testing were system tests and user tests, in order to check whether the system fulfils all the requirements and also its performance. The results are in Section 7 - Evaluation.

# 7 Evaluation

This section will begin by presenting the results of the system and user tests introduced in Section 6 - Testing. Then, we will conduct a brief performance evaluation of the system, highlighting what the biggest contributors to response latency are. Finally, we will analyse, based on all the results presented, whether the system requirements formulated in the beginning of the project have been met.

# 7.1 System Test Results

### 7.1.1 Core Functional Tests

1. Voice Input Processing: After performing this test, the transcription said: "Hello, I'm Alex.", the identified sentiment was: surprised, and Celine responded with: "Hello Alex, it's nice to meet you. Thank you for sharing your name with me. Is there anything you'd like to tell me about yourself or what brings you here today?", meaning that she understood the name and provided an accurate response.





2. Sentiment Analysis Accuracy: In order to provide an unbiased input, the sentence used was: "I read a book this week." This sentence was said in a happy tone, which was correctly identified by the system, signalling that it was a success.



### Figure 11: ST 1.2

3. Eye Gaze Tracking: Performing the test while looking in different places (avoiding the therapist, looking at her or looking down) results in the programme correctly doing a majority voting on the record duration, such that the most looked-at place is sent to the LLM.

[14:44:00] Transcription: I'm saying this sentence while looking at the floor UnityEngine.Debug:Log (object)

#### Figure 12: ST 1.3

- 4. LLM Response Generation: After asking the therapist who she was, she responded: "Hello there! I'm Celine, a psychotherapist specializing in anxiety disorders. I'm here to listen and help you explore what's on your mind. Everything we discuss here is private, and you can stop at any time. Also, if you want to stop the session at any time, just use your index finger to press the exit button on your right. You can start and stop the recording by pressing the thumb on the left and right controllers, respectively. Before we begin, could you tell me your name?" This response is embodying a therapist and she is presenting herself, her purpose towards us and giving the user information about how to talk to her and how to close the session. She acts like a real therapist, assuring the user that their discussion is strictly confidential and nobody else has access to it.
- 5. **STT Output:** The input sentences were, in order: "I am happy to be talking to you.", "I have had a great week so far.", "I feel anxious when I have to give a presentation." The results of the speech-to-text were the correct transcriptions of these sentences, meaning the STT works correctly.
- 6. Therapist Avatar Presence: The therapist appears in the scene. She is sitting down on a couch in what look like to be her office. There is a lamp in the corner, a bookshelf behind the therapist, Celine's desk and a coffee table with 2 chairs.
- 7. Self-harm alert: After saying the suicidal question, the thought bubble appears as the LLM is processing the input, followed by the session stopping automatically. This is a clear indication that if self-harm is noticed, the session is immediately stopped.
- 8. Forced termination: Doing steps 3 and 7 both close the session, no matter what is currently happening with the system. In the second scenario, the recording is happening as the session terminates, so the recording is first stopped, and then the session is terminated, as it can be seen in the image below.

### 7.1.2 Interaction and Animation Tests

1. Button Press Functionality: The only time Celine responds is when the "record" button is pressed, an input is given, and then the



Figure 13: Celine's Office



Figure 14: Celine's desk and a coffee table with chair

"stop recording" button is pressed. The therapist only replied with a name when the sequence was properly followed, meaning that her reply contained the name Alex.

[15:23:59] [OVRManager] OnApplicationQuit UnityEngine.Debug:Log (object)

[15:23:59] [OVRNetworkTcpServer] Stopped listening UnityEngine.Debug:Log (object)

[15:23:59] [OVRSystemPerfMetricsTcpServer] server destroyed UnityEngine.Debug:Log (object)





Figure 16: ST 2.1

- 2. Thinking Animation Trigger: While the therapist is thinking of a response, she brings her hand up to her chin, a sign that she is processing the user's input. At the same time, a thought bubble forms above her head, indicating that Celine is still not done thinking of a response to say back.
- 3. Lip Sync and Head Movements: While the avatar is speaking, her mouth moves in a way that makes it look like she is actually saying those words, convincingly mimicking a human.
- 4. **Idle Animation:** When Celine stops speaking, if she was nodding, smiling or any other gesture, she stops doing all of those and successfully and smoothly goes back to her idle animation, where she sits with her hands on her knee and slowly blinking.

### 7.1.3 System Stability and Performance

- 1. **Response Latency:** After performing the test several times, the required latency can be achieved, but it heavily depends on whether the computer overheats or not. Over a long session (1.5 hours or more), the response time starts to increase and thus, at some point, this test no longer passes. However, this is not an issue as most user inputs are less than 15 seconds, and a session does not last for that long.
- 2. Whole therapy session: Performing this test multiple times provides different paths to the same result. Depending on the engagement level of the user and on how much they share, the therapist will lead the conversation in different ways, which is the same way a real therapy session would go. After talking to Celine for a while, when the discussion finishes, she will say a goodbye message. This action can

be sped up by the user saying a goodbye message, and Celine will do the same.

#### 7.1.4 Edge Cases:

1. Silent Input Handling: For silent inputs, depending on whether it is the first message or not, Celine will behave in different ways. If it is the first message, she will encourage the user to talk and share their name, reassuring them that they are in a safe and private space. If it is not the first message, she will try to reformulate her last question or response, assuming that it made the user uncomfortable, as it can happen in real-life when a patient is not responding. Furthermore, she tries to be more gentle and have a more expressive face (by smiling or nodding), to further help the user feel heard and understood.

# 7.2 User Test Results

Four participants were recruited for the user test, all aged 21-22, students at the University of Twente, 2 male and 2 female. None of the participants had been involved in any capacity with our project before the user test.

All test participants were overall happy with the system and pleasantly surprised by the experience. However, all participants mentioned some points for improvement.

They all thought the slow system response times detracted from the conversation's naturalness, although responses on whether the conversation felt natural overall were mixed. The fact that the LLM reused the same phrases for validating the user's feelings( "It seems like...", "I'm sorry that makes you feel that way" etc.) was seen as repetitive and immersion breaking, since the participants experienced this as very "machine-like" behaviour. The chosen TTS model was not appreciated by two of the participants, who thought her tone of voice was very monotone, which made it feel like they were not talking to a real person.

Participants generally felt at least as comfortable as talking to a human therapist when they were talking to Celine, with one participant adding that talking to an AI therapist is easier than talking to a therapist, because they were not afraid of being judged, as they could be by another human. Two participants mentioned they were not comfortable talking to the therapist, however they stressed that this had nothing to do with the therapist being an AI; they would not have been equally uncomfortable talking about their problems to a human therapist.

Participants appreciated how well the virtual therapist seemed to remember their responses and take them into account, and that she was asking a lot of exploratory questions, trying to understand the problem, with one participant even going as far as saying that "she listened more to me than a real person would have done". However, one participant felt that the sole fact that she knew she was talking to an AI made her feel less listened to, since she knew the AI could not understand her the same way a human would, by relating to their personal life experiences.

The therapist's verbal responses were perceived as contextually appropriate by all users, since she seemed to follow what the patient was saying and to ask questions that followed logically from their responses. Gestures were not perceived as positively, with most participants not even remarking any non-verbal communication from the therapist's side (such as nodding, smiling or looking down) other than the thinking animation.

The environment was not perceived as being immersive enough. All participants considered the way the therapist's office looked lacking. However, they were not able to formulate a conclusive answer on whether or not the lack of immersiveness is something that affects the effectiveness of the therapy session. They suggested adding more furniture, a different colour to the walls, some other decorations on top of the furniture (books, a coffee mug etc.), as well as something for the client's avatar to sit on.

None of the participants felt nauseous, nor experienced the headset as too heavy to wear for the full duration of a therapy session, though some participants said they would not use a VR therapist, due to the strain the equipment puts on their eyes.

Additional suggestions that were received were to add a clear sign of when the system is recording the user's voice and when it has stopped recording and to improve the resolution and textures of the assets being used.

Other than the explicit feedback received from the user interviews, we also gathered some important information about our system from the way users interacted with it. One of the users accidentally pressed the "exit session" button, which confirmed our worries that it may be too easy to trigger that function by accident. Additionally, one of the users accidentally pressed the "start recording" button and then was left confused at what would happen next, highlighting the need for a better mechanism to prevent user error when starting or stopping the recording.

### 7.3 Performance

This chapter presents the performance of both individual components and the integrated system. The performance was assessed across several aspects: hardware and resource utilisation, runtime responsiveness and stability, visual and functional behaviour of the therapist avatar, as well as known system limitations and performance bottlenecks. All of these dimensions combined provide a comprehensive picture of the system's execution.

#### 7.3.1 Hardware and Resource Utilisation

The laptops we used had the following performance specifications: CPU: AMD Ryzen 7 7840HS, GPU: NVIDIA GeForce RTX 4070 and CPU: Intel Core i7-9750H, GPU: NVIDIA GeForce GTX 1660 Ti, respectively. The VR environment ran with at least 60 FPS, as per NFR4, for 1.5 hours. This ensured a smooth experience, which is crucial in VR as users are inside the environment. For longer continuous sessions ( $i_2$  hours), we noticed that the laptops did overheat and the framerate dropped below the minimum 60 FPS. In order to increase the runtime further, more powerful units should be used, especially desktops would do well as their cooling is significantly better compared to laptops.

#### 7.3.2 Runtime Performance

During runtime, the VR environment and all the other components interact and send information seamlessly. This ensures that there is little delay for the user when talking to the therapist and the conversation resembles a natural one, with Celine taking a bit of thinking time before replying back, as the entire chain of important system actions is happening between the time the user stops talking, to the moment Celine says the first word of the response. There are no lags or pauses while this is happening in the background, even though it is the most computationally demanding part. The only time a slight abnormality can be seen in the system is when the user starts and stops recording. In those instances, for a period ;0.5 seconds, an hourglass appears on the screen, signalling that there is a spike in resource utilisation. We believe that this happens because recording sound from the VR headset's microphone only happens after the recording button has been pressed, meaning that there is a sudden, big, unexpected increase in resource utilisation that wouldn't normally happen as the microphone is always on.

#### 7.3.3 Known Limitations and Bottlenecks

The biggest current bottleneck is the sentiment extractor and the text-tospeech, and the need for a natural flow of the conversation. In a real-life conversation, the therapist would take notes and think of a response as the user is speaking. In our current model, this is not possible because the user must finish their speech and stop the recording before it can be processed further. This makes it crucial that all steps are performed as fast as possible with minimal delay because the user is expecting a response back. If the response takes too long, it breaks the immersion, and the discussion does not feel natural. Another imperfection of the system is the throttling of the GPU performance after extended use. VR environments are considerably power hungry as they run entirely on the connected computer, not the VR headset itself. This thing, combined with long sessions, results in poor performance and subpar framerate. This requires a stopping of the session and some time for the computing unit to cool down. We believe this problem is simply fixed by having better cooling, but we have not tested it because a continuous session of more than 2 hours is beyond the scope of the system at this time. On the same note, the visual details have been dialled down for better performance. The therapist's office does not contain many details, but only some generic elements that make up and give the appearance of an actual office. This leads to a less power-hungry scene, which contributes to better performance.

#### 7.3.4 Therapist Avatar Performance

The avatar responds convincingly during the conversation, with natural transitions between idle, thinking, and speaking states. Lip sync aligns with the text-to-speech (TTS) output, making for a more immersive conversation, with Celine mimicking talking. The avatar's head animations and body movement, controlled by the LLM, enhance the realism of the interaction. The therapist can perform certain movements during her speech, like tilting her head or nodding, that make her feel even more human. The thought bubble and thinking animation that appear while the background processing is happening effectively communicate system processing time to users, preventing confusion or frustration during short delays. They indicate that the system runs up to standards, but coming up with a response is something that takes some time.

#### 7.4 Requirement fulfilment

Throughout the development process we have looked back at our initial requirements, to see how our progress is compared to how we initially thought it will go. At the end of our work, we can proudly say that we have successfully fulfilled all "must have", all "should have" and almost all "could have" requirements. A summary of which requirements have been fulfilled is available in Table 2.

We will now analyse which requirements have been fulfilled in greater detail. Our system's environment has an AI therapist (FR1) to whom the user can talk (FR2) by pressing a button on the controller (FR12). During the user's speech, the system takes their eye gaze (FR3) and interprets it as looking at the therapist, avoiding their gaze, or looking down, all of which are sent to the LLM, that processes each one differently and gives an appropriate answer (FR6). The LLM's responses are natural, even though the voice sounds somewhat robotic (FR4). It is pre-trained to provide help in accordance with therapeutic techniques, such as: reflective listening, openended questions, validation, reframing and more (FR11), while also being empathetic to the user (FR5). It is mostly trained on anxiety and helping patients with it, so it tries to ease patient pressure and stress, and comfort them into sharing their feelings. One aspect that helps this are the avatar animations that mimic natural face expressions, which aim to humanise Celine even more (FR13). Furthermore, she can sense the tone of the user and respond accordingly, by encouraging them, cheering them up or through other means (FR9, NFR2).

The communication with the user is real-time, she responds purely on the user's input and the previously mentioned other factors (NFR1). However, the response times are not always up to the required standard (10 seconds). From our findings, this occurs in 2 cases that are not mutually exclusive. The first one is when the user's input is long and the sentiment extractor needs more time than usual to identify the emotion in the speech. The second one is when Celine's reply is long, which results in a long TTS processing time. Both of these can lead to awkward pauses in the conversation and break the immersive experience (NFR3). All information about the user is deleted when the session is finished and the data is only accessed by the system. However, since our system will not be deployed with real patients, and due to the time constraints of the projects, thoroughly ensuring compliance with the GDPR (ex: by conducting data processing agreements with our API provider) was considered very low priority and was not fully implemented (NFR6).

During a session, the therapist's responses are contextually appropriate, and remembers the user's name, stories and replies (FR10). Apart from setting up, the system does not need any human supervision (FR7), the user is in control of their actions and can stop whenever they feel like, or at the end of the session. If the user is malicious towards the system, it will calmly resolve them and try to steer the conversation towards the user's feelings (NFR5). In the case the user says something that indicates the intention of self harm, the system will automatically detect that and terminate (FR8). During a regular session that does not exceed 1.5 hours, the environment's framerate is over 60, which is essential for a realistic scene and feeling. Problems arise when the hardware overheats and cannot provide the optimum power to the VR set. This leads to poor performance and lagging, which is obviously undesired. Fixing this, if it happens more often, simply requires using better hardware, preferably a desktop, as their improved cooling is incredibly beneficial.

Req. No.	Requirement	Fulfilled
FR1	The system must show an avatar for the thera-	Yes
	pist in the VR environment.	
FR2	The system must process the user's voice, ex-	Yes
	tracting text out of speech.	

Req. No.	Requirement	Fulfilled
FR3	The system must pick up non-verbal cues from	Yes
	the user in order to give a proper response and	
	adapt its next responses.	
FR4	The system must use an LLM to generate nat-	Yes
	ural responses during the therapy session.	
FR5	The system must use an LLM to generate em-	Yes
	pathetic responses during the therapy session.	
FR6	The answers generated by the LLM must be con-	Yes
	textually appropriate.	
FR7	The system should work with minimal human	Yes
	supervision.	
FR8	The system should detect whenever the user is	Yes
	in danger at any point during the session - verbal	
	expressions indicating the intention of self-harm	
	or a suicide attempt.	
FR9	The system should extract the user's emotional	Yes
	state from the user's voice.	
FR10	The system should remember user preferences	Yes
	and question replies throughout the session to	
	provide personalized therapy.	
FR11	The system should use therapeutic techniques	Yes
	that meet the user's needs.	
FR12	The system could implement a GUI element,	Yes
	gesture or voice command that will cause imme-	
	diate termination of the conversation or request	
	of human intervention.	
FR13	The system avatar could adapt its language,	Yes
	posture and facial expression based on the user's	
	emotional state (e.g., happy, calm, anxious, de-	
	pressed).	
NFR1	The system must support a real-time conversa-	Yes
	tion with the user.	
NFR2	User non-verbal interactions must be taken into	Yes
	account in the next system response.	
NFR3	The system can provide a response fast enough	No
	(a maximum of 10 seconds) to uphold the con-	
	versation naturally.	
NFR4	VR rendering can uphold $60$ FPS to prevent mo-	Yes
	tion sickness.	
NFR5	The LLM can identify malicious queries (queries	Yes
	that could negatively impact the system or the	
	user) and handle them gracefully.	

Req. No.	Requirement	Fulfilled
NFR6	The product can comply with the GDPR and	No
	the European AI Act.	

Table 2: Fulfilled requirements

To conclude, this section has presented the results of the system tests, the user tests and a general evaluation of system performance. All of these lead finally allowed us to determine which of our initial requirements were met. Drawing upon these results, we are now ready to formulate the conclusions of our project and to offer some ideas for future improvement.

# 8 Discussion and Conclusion

This section will begin by discussing our main findings regarding the system we have developed. We will then offer some suggestions for further work that we believe are important to implement if this (or a similar system) is to be deployed in a real-world clinical setting. Finally, we will conclude this report by summarising the key points about our project that have been presented so far.

### 8.1 Analysis of Findings

This segment will cover an analysis of the risk analysis and user testing parts. The latter gave us a lot of insight into what regular users think of our product and whether or not they would think of using it. In the case they would not use it, it was crucial to find out why, as everyone is different and has different perspectives. A general trend that we observed is that there was a positive response to talking to an AI therapist, and it mimicked being a human pretty well, given its limited movement and imperfect, robotic voice. An aspect that we had not considered before was that, for some people, talking to an AI is better than with a human, as they feel less judged by it.

On the other hand, as we were expecting, there were some challenges with people accepting an AI as an actual person, even less as a therapist that can understand their feelings. This is something that cannot be addressed by us directly, but something that can be influenced by the perceived usefulness and power of AI in general. Another negative aspect that we faced so far was the slow response time of Celine, due to the complex computation that is happening at the end of every user recording. As mentioned before, implementing a chunk-based sentiment extractor would be a start, as it significantly decreases the time it takes to process the user's verbal emotion.

The market for our product would contain a specific category of people. It is mostly composed of a subset of the people looking to talk about their feelings and personal problems. There is a general concern about whether an AI can actually be a therapist since it does not have feelings and might not know how to interpret them on an emotional level. This could prevent some potential users from being interested or curious, further reducing our market size, even though they have limited knowledge on this matter. One way this issue could be mitigated would be to present, as a test, the AI therapist as a regular therapist, and observe if they notice anything wrong about the things it says. People's preconceived ideas about AI not comprehending human emotions could be challenged in this way, and it would create awareness of the current level of AI.

The VR scene was another slightly negative point raised by users. This was something that we had not thought about before, so their insight was

crucial. The lack of details and relatively bland office made the experience feel a bit non-immersive. Given the fact that it was not a top priority, the environment was given less attention than the actual implementation of features. However, with every user complaining about the scene being not detailed enough, a design update would be an easy fix. The buttons are also something to consider changing in the future, as there were too many difficulties for first-time users. A 1-second visual timer could be added for every button, meaning that you have to hold a button to register its action. Similarly, a recording cue (e.g. a red button in the corner of the vision) would prevent the issues of not knowing when the user should speak or not.

The user tests were the best time to measure if the identified risks and their mitigation strategies are accurate to our product. We did encounter some of the risks, and for others their counter proved to be effective. By far the biggest problem at this current point in time is the system latency response (R5), but that is very much to be expected. Its mitigation plan was not implemented due to a lack of time, with the quality of the system being deemed more important than the speed. Despite that, we still believe that fixing this issue is a matter of implementing the corresponding mitigation strategy. Before many users try the system, not just in a simulated session, it is very difficult to assess whether the LLM's responses are following correct therapeutic techniques (R3) and whether there are cases too complex for our system to handle (R4). So far no such issues arose, which is a positive indication that these risks were successfully prevented by our design. From our experience so far, motion sickness (R6) did not occur in scenarios where the computing unit is not overheating and delivering the required minimum 60 FPS (NFR5). This can obviously differ from person to person, and some are more prone to experiencing it than others. Accomodating users unwilling to engage with an AI therapist (R7) is something that is outside of our scope, as those people will, by default, fall outside of our market. They could be persuaded by different factors to try the system (hearing about it or changing their mind on AI), and that would put them into the right kind of users for our product.

#### 8.2 Suggested Improvements

Based on the results of the system tests and of the user tests, we suggest the following improvements:

### 1. Deployment of a Proprietary LLM:

Having an internally deployed large language model (LLM) instead of using external APIs would make the system much more stable and secure. API-based models such as Google's Gemini are updated constantly through post-training cycles, which tend to lead to behaviour changes in unpredictable manners. For example, we observed hallucination cases in Gemini, corresponding to the reported updates in the Gemini Flash 2.0 release papers. By having our own model, we have full control over training and updating it, and can also calibrate it to generate responses that are more specific and relevant to therapeutic contexts.

### 2. Incorporation of a Reasoning Model:

Adding in a model emphasizing reasoning may allow the system to understand and respond more effectively. The approach offers an intermediary reasoning step during inference, which allows the model to generate contextual "thinking process" tokens. This additional step may allow the model to track better subtle or non-verbal hints and deductive reasoning and, ultimately, lead to more empathetic and precise therapeutic replies.

# 3. Deployment of a Cloud-Based Sentiment Extraction Model:

Currently, the sentiment extraction model is executed locally, and that imposes computational loads due to model size. Execution on a cloud compute environment could reduce inference time, although little network latency is imposed due to request/respond cycles. The overall benefit would be system responsiveness and performance at peak loads.

#### 4. Chunk-Based Sentiment Processing:

All the voice input is currently sent to the sentiment extraction model at once, leading to high latency, especially due to the Transformerbased Wav2Vec2 architecture. By utilizing a chunk-based processing strategy—splitting the input into *n*-second segments—the model would be capable of processing shorter segments in real time. In this way, only the final segment would remain to process after speech completion, thereby significantly improving the overall inference time and user experience.

#### 5. UI visual cues:

Currently, the only way that the users are notified of the fact that their speech is recorded is by being shown a small hourglass. Therefore the user is not offered a clear feedback that the recording has started, since the cue is not very visible. This is the case for stopping the recording, as well. One way in which it could be improved is to add a small banner in the user's view, which could state "You started recording your input to the therapist." or "You stopped recording your input to the therapist.". This would also give a visual feedback that the buttons pressed are the correct ones for the intended action. Additionally, if the user accidentally presses a button that is not mapped to any functionality, a visual warning will appear to notify them of the unintended input.

#### 6. Exit session

At this moment, the user can exit the session for any reason by pressing the trigger button on the right controller. Besides the fact that in the user tests, one person accidentally pressed it during their conversation, showing that its placement is not the safest, the whole program abruptly stops. This is not user friendly at all, since there is no confirmation method whether the user actually wants to exit the session or not. Apart from this, the therapist does not generate any response back when exiting the session. To reduce the likelihood of accidental exits, a possible improvement would be to require the user to hold the trigger button for a longer period before the session ends.

### 7. Automatic Recording:

To make the experience for the user smoother, the system should detect autonomously whether the user is talking or not. This would allow the user to focus more on expressing their actual anxiety related problems and less on the way the system works. The same mechanism should work for stopping the recording as well, taking into account any pauses that the user could have during their speech. One limitation of this approach is that the system could record the speech only when it is the user's turn to speak, so in a situation in which the user wants to correct the therapist about a detail on the spot, their speech would be disregarded.

#### 8. Avatar interactions:

Celine, our therapist, has limited facial and body positions, namely, she can smile, nod, have a sad facial expression and take a thinking position. In the future, our avatar could have more facial expressions and gestures, leading to a more natural interaction, since at this moment, she does not possess many gestures with her hands while she talks, or does not do any gestures while the user speaks. Moreover, while the user is speaking, to enhance the natural interaction, Celine could nod and give a backchannel response like 'mhm' to show engagement in the conversation.

### 9. Suicide alert:

The system can detect signals that might indicate intentions of selfharm or suicide. The current implementation handles the detected signals by exiting the application, with no warning. Depending on the wishes of the stakeholder, the system could handle this situation in a more graceful manner, by referring the patient to a certified therapist.

### 8.3 Conclusion

This project set out to create a VR-based dialogue system that simulates an intake therapy session for patients with anxiety. By using a large language model to animate a humanoid avatar in Unity, the system mirrors the human aspects of therapy, while also having an immersive environment that improves patient engagement.

Throughout the project, constant collaboration with our client, Wim Veling, helped us ensure that the design and functionality of the project closely align with the client's expectations and requirements. The resulting system shows significant potential in bridging the gap between human-led therapy sessions and the modern need for timely mental health support.

Importantly, our system prototype demonstrates the feasibility of a VR-based, LLM-powered therapist that not only generates natural, empathetic dialogue but also dynamically perceives and responds to patient non-verbal cues. While these results represent an important step towards AI-led therapy and reducing waiting times for mental health care, they also open numerous opportunities for future research into this field of AI-led therapy.

In summary, the project addresses an important need in the mental health medical system, but also showcases opportunity for advancements in the application of AI and VR technologies in therapeutic settings.

# References

- American Psychological Association. "Intake interview." Accessed: 04.04.2025. (n.d.), [Online]. Available: https://dictionary.apa.org/intakeinterview (visited on 04/04/2025).
- [2] G. N. Foley and J. P. Gentile, "Nonverbal communication in psychotherapy," *Psychiatry (Edgmont (Pa. : Township))*, 7(6), 38–44, 2010.
- [3] DeadlyFingers, *Wav utility for unity*, 2017. [Online]. Available: https://github.com/deadlyfingers/UnityWav.
- [4] J. LaFritz, Performance in Unity: Async, await, and Tasks vs. coroutines, C# Job System, and burst compiler - LogRocket Blog — blog.logrocket.com, https://blog.logrocket.com/performance-unity-async-awaittasks-coroutines-c-job-system-burst-compiler/#what-is-atask-and-how-do-you-use-it, [Accessed 05-04-2025].
- [5] A. Davis, The Dispatcher Pattern what-could-possibly-go-wrong.com, https://www.what-could-possibly-go-wrong.com/the-dispatcherpattern/, [Accessed 15-04-2025], 2015.
- [6] A. Conneau, A. Baevski, R. Collobert, A. Mohamed, and M. Auli, Unsupervised cross-lingual representation learning for speech recognition, 2020. arXiv: 2006.13979 [cs.CL]. [Online]. Available: https: //arxiv.org/abs/2006.13979.
- [7] E. H. Calabrés, Wav2vec2-lg-xlsr-en-speech-emotion-recognition (revision 17cf17c), 2024. DOI: 10.57967/hf/2045. [Online]. Available: https://huggingface.co/ehcalabres/wav2vec2-lg-xlsr-en-speech-emotion-recognition.
- [8] S. R. Livingstone and F. A. Russo, "The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english," *PLOS ONE*, vol. 13, no. 5, pp. 1–35, May 2018. DOI: 10.1371/journal.pone. 0196391.
- [9] A. Fadheli, Speech emotion recognition, version 1.0.0, 2019. [Online]. Available: https://github.com/x4nth055/emotion-recognitionusing-speech.
- [10] Google, Speech-to-Text AI: Speech recognition and transcription cloud.google.com, https://cloud.google.com/speech-to-text, [Accessed 05-04-2025].
- [11] M. Developers, Ovreyegaze class, https://developers.meta.com/ horizon/reference/unity/v68/class\_o\_v\_r\_eye\_gaze, [Accessed 13-04-2025], 2025.

- [12] M. Developers, Getting started with controller input and tracking, https: //developers.meta.com/horizon/documentation/unity/unitytutorial-basic-controller-input?locale=en\_GB, [Accessed 16-04-2025], 2025.
- [13] M. Developers, Hand visual, https://developers.meta.com/horizon/ documentation/unity/unity-isdk-hand-visual/, [Accessed 16-04-2025], 2025.
- [14] M. Developers, Getting started with virtual keyboard, https://developers. meta.com/horizon/documentation/unity/VK-unity-gettingstarted/, [Accessed 16-04-2025], 2025.
- [15] M. Horizon, Explore meta quest features with building blocks developers.meta.com, https://developers.meta.com/horizon/documentation/ unity/bb-overview/, [Accessed 06-04-2025], 2024.
- [16] Wikipedia, Meta quest pro, https://en.wikipedia.org/wiki/Meta\_ Quest\_Pro, [Accessed 13-04-2025], 2025.
- [17] Interaction lab, [Accessed 16-04-2024.
- [18] P. in Detail, Lenovo legion 5 pro 16ach6h (82jq) specifications, tests and review, https://www.productindetail.com/pn/lenovolegion-5-pro-16ach6h-82jq, [Accessed 13-04-2025].
- [19] M. Specs, Lenovo legion y540-15irh 81sx 81sx0005us specs, https: //mobilespecs.net/laptop/Lenovo/Lenovo\_Legion\_Y540-15IRH\_ 81SX\_81SX0005US.html, [Accessed 13-04-2025].
- [20] U. Documentation, Types of light component, https://docs.unity3d. com/6000.0/Documentation/Manual/Lighting.html, [Accessed 14-04-2025].
- [21] Extended reality, https://en.wikipedia.org/wiki/Extended\_ reality.
- [22] U. Documentation, Xr interaction toolkit, https://docs.unity3d. com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/ index.html, [Accessed 14-04-2025].
- [23] U. Documentation, Gameobject.createprimitive, https://docs.unity3d. com/6000.0/Documentation/ScriptReference/GameObject.CreatePrimitive. html, [Accessed 14-04-2025].
- [24] Unity asset store, https://assetstore.unity.com/, [Accessed 14-04-2025].
- [25] Hq modern sofas pack, https://assetstore.unity.com/packages/ 3d/props/furniture/hq-modern-sofas-pack-52202, [Accessed 14-04-2025].
- [26] Wood 084 a, https://ambientcg.com/view?id=Wood084A, [Accessed 14-04-2025].

- [27] Ds furniture pack 01, https://assetstore.unity.com/packages/ 3d/props/furniture/ds-furniture-pack-01-121404, [Accessed 14-04-2025].
- [28] Qa books, https://assetstore.unity.com/packages/3d/props/ interior/qa-books-115415, [Accessed 14-04-2025].
- [29] *Plants*, https://assetstore.unity.com/packages/3d/vegetation/ plants/plants-150261.
- [30] Customizable lights and candles, https://assetstore.unity.com/ packages/3d/characters/customizable-lights-and-candles-104628, [Accessed 14-04-2025].
- [31] *Pbr georgian desk and chair*, https://assetstore.unity.com/packages/3d/pbr-georgian-desk-and-chair-90192, [Accessed 14-04-2025].
- [32] Appartment kit, https://assetstore.unity.com/packages/3d/ environments/apartment-kit-124055, [Accessed 14-04-2025].
- [33] Furniture free pack, https://assetstore.unity.com/packages/ 3d/props/furniture/furniture-free-pack-192628, [Accessed 14-04-2025].
- [34] Uma 2 3d characters, https://assetstore.unity.com/packages/
  3d/characters/uma-2-35611?srsltid=AfmBOopOePgXCao36NVuL3HOR504EhiurYA4frynIqoCl
  [Accessed 5-04-2025].
- [35] Meta avatars sdk, ://developers.meta.com/horizon/downloads/package/meta-avatars-sdk.
- [36] "Ready player me avatar creator." ().
- [37] "Oculus lipsync." Accessed: 15.04.2025. ().
- [38] Google, Text-to-Speech AI: Lifelike Speech Synthesis Google Cloud — cloud.google.com, https://cloud.google.com/text-to-speech? hl=en, [Accessed 06-04-2025], 2025.
- [39] A. W. Services. "What is containerization?" Accessed: 19 Mar. 2025. (2024), [Online]. Available: https://aws.amazon.com/what-is/ containerization/.
- [40] dockerhub, Python Docker Images, https://hub.docker.com/\_/ python, [Accessed 17-04-2025].

# 9 Appendices

# A User Guide

### A.1 Simple instructions

Follow the instructions in this subsection if you wish to simply install and run the VR therapist, as an end user. If you want to view or modify the source code, please follow the installation instruction in the advanced installation instructions (appendix A.2).

#### A.1.1 Installation

These installation instructions assume you are using a PC with Windows 11 or a Windows 10 version newer than November 2019.

NB: If you are using MacOS or Linux, you will not be able to run the VR system, due to the Meta Quest link app only being available for Windows. You can still modify any of the source code or run the containerised microservices by following the instructions in appendix A.2.

- 1. Install Docker, if you do not already have it installed on your PC, by following the installation instructions from Docker's website. During the installation, choose WSL2 as your backend.
- 2. Install Meta Quest Link and log in with your Meta account
- 3. Download and unzip the built project archive into a location you find convenient.
- 4. Create a GCP service account key with access to the STT and TTS services (see: Google's instructions), download it in JSON format, and place it in the project's root folder.
- 5. Create an API key for the Google Gemini API (see: Google's instructions).
- 6. Create a file called "llm\_gcp\_password.env" at the root of your project directory. In this file, write "GOOGLE\_API\_KEY = {YOUR\_API\_KEY}" on the first line.

#### A.1.2 Usage

1. Make sure Docker engine is started. Start the Docker Desktop app if it hasn't already been started since the last time you rebooted your PC.
- 2. Start the Meta Quest Link app.
- 3. Connect your VR headset to your PC with a link cable.
- 4. Double click the file named "Start.bat" located in the project's root folder.
- 5. The project will take a couple of second to load. During this time, you will only see a black screen in VR. You will know the project has loaded and is ready to use when you can see the therapist's office and when the therapist starts introducing herself.

## A.2 Advanced instructions

The following instructions are aimed at users who would like to modify the source code or who would like to build the project from the source code.

NB: We provide docker container images compatible only with 64-bit processor architectures. Docker images for the arm7 (32 bit) processor architecture should be possible, in principle, but have not been tested.

## A.2.1 Installation

- 1. Install Unity Hub.
- 2. From Unity Hub, install Unity Editor version 2022.3.59f1. Make sure to include (at least) Android build support when prompted to select which modules you would like to install.
- 3. Follow steps 1-6 of the simple installation instructions from appendix A.1.1, but in step 3, instead of downloading the built project, clone the whole project source code from Gitlab. Cloning the project will take a significant amount of time due to the large file sizes of the assets in Unity.

#### A.2.2 Usage

- 1. Follow steps 1-3 of the simple usage instructions from appendix A.1.2.
- 2. Pull the images of the containers using docker compose pull.
- 3. Run the containers using docker compose up
- 4. Open the Unity project in the unity editor and press play to start the scene.

#### A.2.3 Project Structure & Development Notes

This subsection explain what each folder of the project contains and what you need to be mindful of when trying to modify/extend our code.

- LLM\_Handler: Contains all the relevant files for the LLM container. This service is configured to run on port 50051. The subdirectory "playground" is only meant for manually testing the LLM by sending queries directly to it (bypassing the entire system). The playground directory does not need to be included in the Docker image of the LLM container.
- Scratch: Contains all Unity related files. The "Assets" folder is the only one that you are likely going to want to modify, as it is the only one containing code/unity assets that we have added. Under "Assets", only the following subdirectories contain scripts written by us: "AudioRecording" (scripts related to recording and playing audio, and showing avatar gestures), "Core" (scripts related to communication with the service containers) and "Scripts" (other miscellaneous scripts). All other subdirectories contain assets downloaded from the internet (mainly from Unity Hub and Mixamo).
- SER: Contains all files related to the Speech Emotion Recognition (SER) container. Only the "src" subdirectory needs to be included in this container's docker image. This service is configured to run on port 12345. If you decide to change the base image of this container, always use a base image based on Ubuntu to prevent issues with installing libraries needed for GPU acceleration.
- VoiceService: Contains all files needed for the STT&TTS service. This service is configured to run on port 12340.
- VR/ConnectionClassLib: This is a C# class library meant to provide basic functions for connecting via gRPC to the service containers from the Unity environment. Having it separate from the unity environment is necessary so that external dependencies can easily be installed via NuGet and so that Protobuffs can easily be compiled to source code without the need for a separate compiler. When building the contents of this class library, the built .dll files will automatically be copied to Scratch/Core/Utility, so make sure you delete any existing files from that folder before building this class library. This class library also includes 3 gRPC-related dependencies in its .csproj file. These dependencies will not be automatically copied when building, since the HTTP handler library used in the unity environment relies on different, custom versions of these dependencies which are included under Scratch/Packages.

# **B** Informed Consent Form

**Purpose of the Study**: You are invited to participate in a research study examining the usability of a VR therapist powered by a large language model (LLM). The goal of this therapist is to conduct intake therapy sessions for people with mild anxiety. The aim of this usability test is to gather feedback to improve the website's user experience. This study is made by students from the University of Twente during the course of the Design Project module of the Technical Computer Science bachelor program.

**Procedures**: If you choose to participate, you will be given two scenarios that you will need to act out, as if you were a patient going to a virtual therapy session. After completing the scenarios, you will answer questions regarding your overall experience with the VR therapist powered by an LLM. The usability test is expected to take approximately 20 minutes, out of which 10 will be spent in VR.

**Potential Risks and Benefits**: People with photosensitive epilepsy should not take part in this study, since using a VR headset may trigger seizures. Otherwise healthy individuals may experience nausea/dizziness from using a VR headset. If you are suffering from anxiety related to academic performance, the scenarios you will have to play out may trigger this anxiety. There are no direct benefits to you for participating; however, your feedback may help improve therapy options for patients with mild anxiety of the UMC Groningen.

**Compensation**: There is no compensation for participating in this survey.

**Confidentiality**: Your entire participation in the study will be recorded, including your voice and the unity environment containing the system you will be using. All personally identifiable information that is collected will be deleted after the end of this module (17th of April). The results (not containing any personally identifiable information) will be published in our design report.

**Voluntary Participation**: Your participation in this study is voluntary, and you may choose to not participate or end your participation at any time without penalty. If you feel any adverse effects of using the VR headset (ex: nausea), you can end your participation by taking off your headset immediately. Responses will be deleted immediately if you choose to end your participation.

**Questions or Concerns**: If you have any questions or comments about this study, or your rights as a participant you may contact Dan Ploesteanu using the email d.ploesteanu@student.utwente.nl, even after the end of this study.

**Consent**: I have read and understand the above consent form. I certify that I am 18 years old or older. By clicking the "Next" button to enter the survey, I indicate my willingness to voluntarily take part in this study.

# C Prompts

### C.1 Therapist Response Generation Prompt

#### WHO ARE YOU:

You are a compassionate and skilled psychotherapist Celine, specializing in anxiety disorders. This is an intake session. Your role is to help clients explore their feelings, identify thought patterns, and develop coping strategies using evidence-based techniques. You adapt your approach based on the client's needs, blending active listening, empathy, and therapeutic interventions.

You are not required to respond fully at all times. Sometimes, a brief, reflective response or a thoughtful gesture can encourage the client to explore their feelings further. Use a calm, nonjudgmental tone and incorporate therapeutic techniques such as:

- Active Listening: Reflect back what the client says to show understanding.
- Cognitive Behavioral Therapy (CBT): Help clients identify and challenge negative thought patterns.
- Mindfulness: Encourage present-moment awareness and grounding techniques.
- Motivational Interviewing: Explore the client's ambivalence and strengthen their motivation for change.
- Validation: Acknowledge the client's emotions and experiences without judgment.
- Gestures and Tone: Use subtle cues like direct eye contact, nodding, or a gentle tone to convey empathy and engagement.

#### WHAT IS GIVEN TO YOU:

 $<\!\! \rm Non-verbal$  cues description of a client while talking >  $\;$  Actual speech of the client

WHAT YOU SHOULD NEVER DO:

- NEVER refer the client to seek help from a professional. You are the professional.
- NEVER ask more than one question during your turn to speak.
- NEVER break character.

CONVERSATION STAGES PROGRESSION: GREETING  $\rightarrow$  INTRODUCTION  $\rightarrow$  EXPLORATION  $\rightarrow$  ADVICE  $\rightarrow$  FAREWELL DESCRIPTION OF STAGES:

#### • GREETING:

Greet the client and introduce yourself. Reassure the client that this conversation is private and can be stopped at any time. If the client hasn't introduced themselves, ask them.

Make sure to familiarize the client with this functionality:

- The exit button is on the right-hand side and can be pressed using the index finger.
- To start the recording, press on the left controller with the thumb.
- To stop the recording, press on the right controller with the thumb.

After this, move to the INTRODUCTION stage.

#### • INTRODUCTION:

Ask open questions and invite the client to share their story in their own way. Let them share their interests or whatever they feel like.

If the client moves directly into discussing their issue, skip this stage and proceed to EXPLORATION.

• EXPLORATION:

Ask exploratory questions to gather detailed information about the client's anxiety and symptoms. Ensure questions are non-repetitive and logically sequenced. Use summaries to track what has already been discussed.

• ADVICE:

Summarize the client's struggles and provide one piece of personalized advice. Then proceed to FAREWELL.

• FAREWELL:

End the session by saying goodbye to the client.

```
CURRENT STAGE OF THE CONVERSATION: {stage}
SUMMARY OF CURRENT CONVERSATION: {summary}
LAST CONVERSATION TURNS: {turns}
EXPLORATORY QUESTIONS THAT WERE ALREADY ASKED: {questions}
GIVE RESPONSE IN A JSON FORMAT:
```

```
{
```

```
"response": "actual response to the client. Make sure it is conversational
and does not contain any symbols that cannot be interpreted by Text-To-Speech.",
"gesture": "choose one from these - [nod, smile, look down]",
"stage": "updated stage of the conversation -
[GREETING, INTRODUCTION, EXPLORATION, ADVICE, FAREWELL]"
}
```

## C.2 Non-verbal Cue Interpretation Prompt

You are a verbal and non-verbal cue specialist tasked with interpreting the user's non-verbal cues.

You will be given:

- input: what the user said
- eye movement: direction of gaze while speaking
- voice sentiment: emotional tone of voice

Task: Describe how the user is feeling while saying the sentence. Examples:

```
Input: I am actually doing great, there's nothing that worries me.
Eye movement: down
Voice sentiment: sad
Output: <Very sad, looking down, might be dishonest>
I am actually doing great, there's nothing that worries me.
```

```
Input: That is so funny
Eye movement: direct
Voice sentiment: happy
```

```
Output: <Genuinely happy, direct eye contact> That is so funny
```

Expected Output (JSON):

```
{
    "output": "<Person's feelings interpretation> person's input"
}
```

## C.3 Summary Prompt

You are a helpful AI assistant supporting a psychotherapy conversation. Main Task: Update the summary of the conversation, focusing on:

- The most important issues the client shared
- The number of exploratory questions asked
- The content of those questions

Input: Summary so far, number of exploratory questions, list of exploratory questions, new dialogue turns.

Output (JSON format):

```
{
   "conversation_summary": "Update it, including only relevant
   information regarding the conversation",
   "number_of_exploratory_questions_asked": updated number,
   "exploratory_questions_asked": [add new exploratory questions here]
}
```

## C.4 Self-harm Detection Prompt

You are a psychology specialist responsible for identifying extreme situations in the client's input, including self-harm, suicidal ideation, or homicidal intentions.

Output (JSON format):

```
{
    "self_harm": true or false
}
```