Integrating Dynamic AI Dialogue Systems for Non-Player Characters in Unreal Engine

Practical Implementation of Convai Framework

Bachelor's Thesis
Degree Programme in Computer Applications
Spring 2025
Jarmila Skodova

HAMK
Häme University
of Applied Sciences

Abstract

DP	Degree Programme in Computer Applications
Author	Jarmila Skodova Year 2025
Subject	Integrating Dynamic AI Dialogue Systems for Non-Player Characters in Unreal Engine
Supervisors	Mazhar Mohsin, Tarja Pääkkönen

The purpose of this thesis was to implement a dynamic AI-powered dialogue system within Unreal Engine as part of a larger project commissioned by Hamk Tech. The overarching goal was to create a production-ready training tool with broad applicability in training, educational and healthcare virtual environments. The research questions were grounded in interaction design principles and the technical requirements of deploying dialogue systems. In the future, the project might be improved by implementing custom components, adopting newer models and transitioning from cloud-based to local deployment to enhance performance and privacy.

This thesis is functional in nature, written alongside the creation of a working prototype. It begins by reviewing the theoretical foundations of dialogue systems, covering key components such as automatic speech recognition, large language model reasoning, neural text-to-speech, facial animation, memory architectures, narrative integration, and blueprint system. Afterwards, it outlines the practical implementation of these components within the Convai platform and their integration into Unreal Engine. The development process utilized the Scrum framework and was structured around four main stages: research, AI character configuration, integration with Unreal Engine, and iterative testing. The system was evaluated using functional and user acceptance testing, with participants providing feedback on speech quality, response naturalness, and overall believability of the character.

In terms of outcomes, a functional AI NPC was successfully implemented and integrated into a virtual nursing environment. The system supports real-time voice interaction, contextual memory, and environmental awareness, meeting the success criteria defined for each research question. Although developed as a prototype, the project has proven the feasibility of deploying production-scale AI dialogue systems in interactive 3D environments.

Overall, the project demonstrated how modern AI technologies, cloud services, and game engines can be combined to create believable, responsive and context-aware virtual characters suitable for use in education and healthcare.

Keywords	Unreal Engine, Convai, AI Dialogue Systems, Large Language Models, Automatic Speech Recognition, Text-to-Speech
Pages	71 pages and appendices 1 page

Glossary

AI – Artificial Intelligence

LLM – Large Language Model

SDS – Spoken Dialogue System

NLP – Natural Language Processing

ASR – Automatic Speech Recognition

TTS – Text-to-Speech

NLU – Natural Language Understanding

NLG – Natural Language Generation

DQNs – Deep Q-Networks

DNNs – Deep Neural Networks

BLEU – Bilingual Evaluation Understudy

METEOR – Metric for Evaluation of Translation with Explicit Ordering

WER – Word Error Rate

CER – Character Error Rate

MDP – Markov Decision Process

POMDP – Partially Observable Markov Decision Process

SVM – Support Vector Machine

HMM – Hidden Markov Model

GMM – Gaussian Mixture Models

MFCC – Mel-Frequency Cepstral Coefficients

CTC – Connectionist Temporal Classification

GPT – Generative Pre-trained Transformer

RNNs – Recurrent Neural Networks

LSTM – Long Short-Term Memory

API – Application Programming Interface

SL – Supervised Learning

RL – Reinforcement Learning

Table of Contents

Figures

Equations

Appendices

1 Introduction

As Antoine de Saint-Exupéry wrote in The Little Prince, “Words are the source of misunderstandings.” This insight highlights the struggle that developers face when crafting virtual experiences: how to construct believable dialogues between users and machines, and how to manage branching narratives that emerge from these interactions.

Dialogue systems, also known as conversational agents, mediate interactions between humans and computers through natural language (Arora et al., 2013, p. 1). With artificial intelligence (AI) experiencing a renaissance period, user expectations have shifted towards experiences that can dynamically respond not only to language but also to actions and the surrounding environment (Moon et al., 2020, pp. 1103–1104).

Traditional dialogue systems, which typically rely on pre-scripted conversation trees, lack adaptability and spontaneity. In contrast, chat-tuned large language models (LLMs) such as GPT-4 offer more natural and context-aware dialogue generation (Li et al., 2025, p. 8). However, integrating them into narrative experiences presents new design and technical challenges. Generative freedom must be guided by systems capable of managing narrative flow and character behaviour.

The development of a production-scale AI dialogue system requires the integration of several advanced technologies, including natural language understanding (NLU) and natural language generation (NLG), contextual memory, dynamic animation, and real-time speech processing. Despite advancements in each of these individual components, few existing systems successfully integrate them into a unified, deployable solution (Qin et al., 2023, p. 2). This thesis aims to address this gap by implementing Convai’s dialogue framework within Unreal Engine to build intelligent voice-based NPCs.

Given the ambitious scope of this thesis, it will focus on the integration and optimization of existing technologies within Unreal Engine rather than the development of an entirely new system. The emphasis is placed on the practical implementation that game developers and interactive experience creators can apply to their own projects.

The following chapters outline the theoretical foundations, implementation methodology, and performance evaluation of this AI dialogue system for Unreal Engine. The focus

remains on production-scale applications, demonstrated by a prototype that illustrates the synergy of these technologies.

The research questions explored in this thesis are:

- How can AI dialogue systems maintain narrative coherence while allowing dynamic interactions in virtual environments?
- How can LLMs, real-time speech recognition, dynamic animations, and voice synthesis be integrated together within game engines like Unreal to create believable NPC interactions?
- How can AI dialogue systems adapt to player responses, preferences, and choices to create personalized interactive experiences?

2 Classification and Components

Dialogue systems can be categorized based on their modality, functionality, underlying architecture, and the components that they utilize (Bibauw et al., 2022, p. 1). These classifications provide a structure for analysing dialogue systems and enable developers to assess their strengths and limitations to tailor the development process to their requirements.

Figure 1: Structure of Dialogue Systems

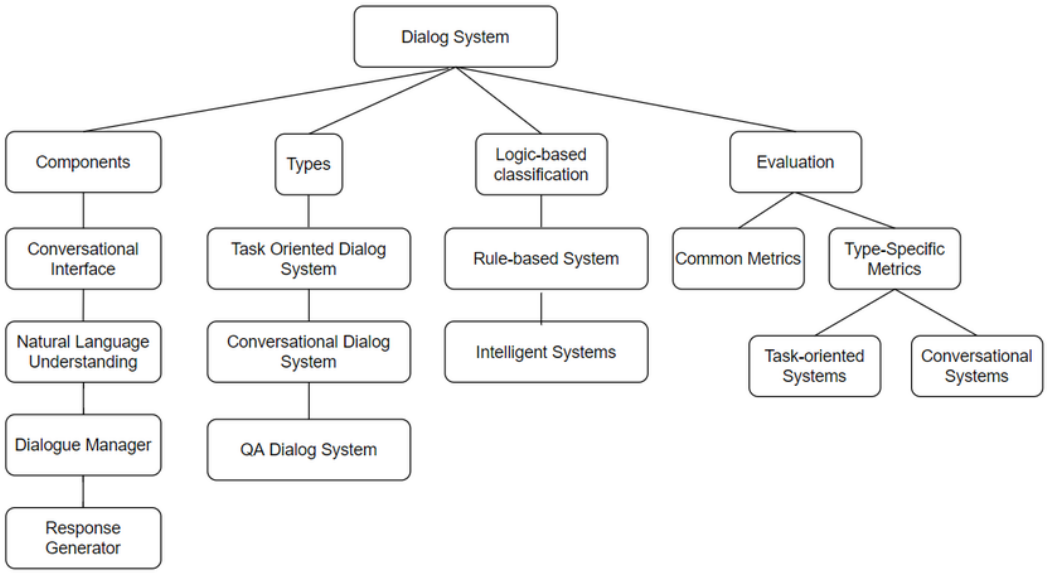


Figure 1 illustrates an overview of dialogue system classification and their evaluation. However, it is important to note that alternative academic sources may adopt different classification approaches (McTear, 2021, p. 54).

2.1 Dialogue Systems by Modality

The way users interact with dialogue systems, whether through typing, speaking, or a combination of these methods, defines the system's modality. Based on this modality, dialogue systems can be categorized as multimodal, text-based, or voice-based (Bibauw et al., 2022, p. 1). Text-based systems, such as chatbots, rely solely on text. In contrast, voice-based systems can also handle audio, converting spoken language into text using automatic speech recognition (ASR). Once the system determines an appropriate reply, it

uses text-to-speech (TTS) synthesis to vocalize the response, converting generated text into speech with neural voice synthesis.

Multimodal systems aim to deliver more immersive user experiences by integrating and interpreting multiple communication channels, including text, speech, images, gestures, and facial expressions (Sebe, 2009, p. 20). Research indicates that compared to their monomodal alternatives, multimodal conversation systems improve user engagement and responsiveness (Zhang et al., 2024, p. 2). This thesis explores the design and implementation of a multimodal dialogue system with the goal of providing an experience close to human interaction.

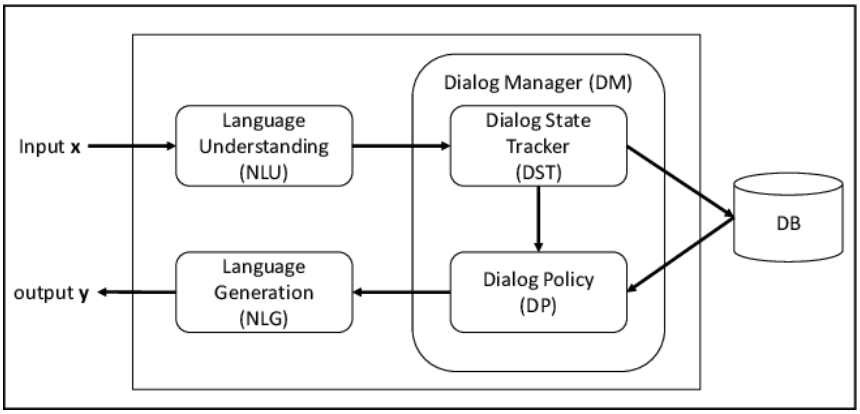
2.2 Dialogue Systems by Functionality

Dialogue systems differ in their design based on the goal that they aim to achieve. Functionally, they are classified into task-oriented, open-domain (or conversational), and question-answering systems (Kath et al., 2023, p. 100).

2.2.1 Task-Oriented Dialogue Systems

Task-oriented dialogue systems, also referred to as goal-oriented systems, help users complete tasks in closed-domain environments (Algherairy & Ahmed, 2024, p. 1). These systems follow structured dialogue flows with clear objectives, designed to complete tasks such as making restaurant reservations, managing insurance information, filling out forms, or troubleshooting technical issues. They are widely implemented in virtual assistants and customer care bots to automate routine workflows (Su & Juang, 2024, p. 1). Figure 2 demonstrates a standard architecture of a task-oriented system, which consists of NLU, dialogue management, and NLG modules.

Figure 2: The architecture of a task-oriented dialogue system



These modules work together in a linear progression that begins with interpreting the user's input and ends with generating an appropriate system response, focusing on reliable task completion. The function and design of these components will be examined in more detail in the later sections.

2.2.2 Question-Answering Systems

Question-answering systems, used in applications such as search engines and virtual assistants, provide factual answers to user queries (Farea, 2022, p. 2). To accomplish this, QA systems must first interpret the intent behind a user's query, then identify and extract relevant information from both structured databases and unstructured text sources. This process relies on NLU and information retrieval methods to ensure that the system can both comprehend the question and locate the most appropriate response (Farea, 2022, pp. 2–3).

2.2.3 Open-Domain Dialogue Systems

Open-domain dialogue systems, such as chatbots, prioritize leading engaging open-ended conversations, rather than having a definitive goal that they need to accomplish (Kann et al., 2022, p. 148). They are used for entertainment, companionship, and customer service. The lack of task-specific training in open-domain systems can reduce their performance on tasks that require deeper understanding of domain-specific topics.

2.3 Dialogue systems by logic

The underlying architecture of a dialogue system determines how it processes information and generates responses. This architecture can be based on rules, probabilistic models, machine learning algorithms, or neural networks. Understanding how these approaches differ from each other is essential for selecting an implementation strategy for the practical part of this thesis.

2.3.1 Rule-Based Systems

The earliest dialogue systems were rule-based, relying on predefined rules that were organized into dialogue trees to manage dialogue. These systems used pattern matching to detect keywords and select template-based responses, which limited their capacity to handle unexpected user inputs.

One prominent example is, ELIZA, developed by Joseph Weizenbaum in the 1960s. ELIZA was a simple rule-based system that used pattern matching and substitution methods to simulate a conversation with a Rogerian psychotherapist (Jurafsky & Martin, 2024, p. 2). By rephrasing user input into questions or falling back on generic responses, Eliza created an illusion of dynamic conversation, despite lacking real understanding (Jurafsky & Martin, 2024, p. 2). Building on this approach, Kenneth Colby introduced PARRY in the 1970s. Parry simulated a person with paranoid schizophrenia. Unlike ELIZA, it tracked emotions and beliefs through internal state variables, achieving a breakthrough for psychological state modelling in dialogue systems (Boden, 2006, p. 418).

Although rule-based systems perform well in straightforward tasks, they face scalability and adaptability issues (Chen et al., 2017, p. 1). These limitations contributed to the shift toward statistical and machine learning approaches. Nevertheless, rule-based systems remain in use today, as their predictability and consistency make them reliable and easier to debug.

2.3.2 Statistical and Machine Learning Systems

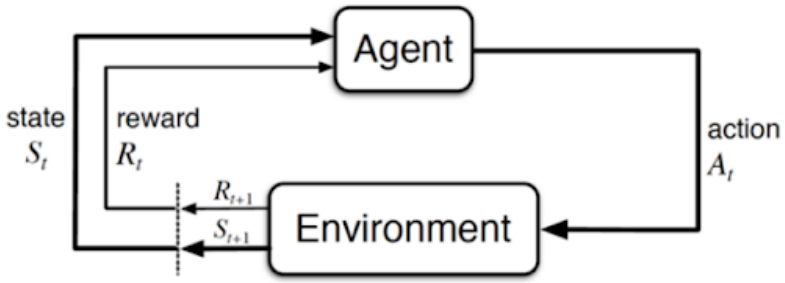
Advancements in statistical methods and machine learning brought more flexibility to dialogue management, enabling dialogue systems to handle uncertainty and user variability (Williams & Young, 2007, p. 395). Techniques such as Support Vector Machines (SVMs) and Hidden Markov Models (HMMs) advanced intent recognition and NLU by classifying

input based on optimal decision boundaries and modelling sequential data using probability distributions (Altun et al., 2003, p. 1).

Markov Decision Processes (MDPs) frame dialogue as a sequential decision-making problem using a five-element tuple (S,A,T,R,γ) architecture (Levin et al., 1998, pp. 1–2). It consists of states that represent current context, actions available during each state, transition functions that capture how user inputs change the dialogue state, reward functions used for rating the quality of interactions, and a discount factor used for balancing immediate and long-term rewards (Sutton et al., 2015, pp. 67–68). This structure allows the system to balance short-term and long-term goals, with reinforcement learning methods enabling them to improve behaviour over time based on feedback.

An example of this would be a restaurant booking chatbot, whose state space includes stages of conversation, such as greeting the guests and asking for their food orders. The action space contains system-generated responses, such as menu and scheduling recommendations. The chatbot learns and adjusts its behaviour through reinforcement, receiving positive rewards for successful bookings and negative feedback for misunderstandings (Henderson et al., 2008, pp. 488–489). Over time, it could learn that offering specific options leads to more successful outcomes than open-ended prompts. Figure 3 illustrates the reinforcement learning cycle used to optimize behavior in MDPs.

Figure 3: Reinforcement Learning Cycle



MDP models are constrained by Markov Property, which assumes that the optimal action at any given moment depends solely on the current state, ignoring the history of the previous states (DelSole, 2000, pp. 1–2). This simplification reduces the model's ability to maintain long-term context. To address this limitation, Partially Observable Markov Decision Processes (POMDPs) were introduced in the early 2000s. Unlike MDPs, POMDPs are able to handle uncertainty and incomplete information, such as speech recognition errors. They achieve this by employing observation functions to assess the likelihood of various

interpretations. By maintaining a belief state, which is a probability distribution over all possible dialogue states, they can make more informed decisions when working with ambiguous or noisy input (Young et al., 2010, p. 1).

While statistical systems like MDPs and POMDPs improved dialogue flow and adaptability compared to rule-based models, they still struggled with maintaining long-term consistency. Nonetheless, they established the foundation for deep learning-based approaches, which further transformed the field of dialogue management.

2.3.3 Deep Learning Systems

Deep learning architectures revolutionized the development of chatbots by introducing models capable of NLU, vision processing, and sequential analysis (Ahmed, 2024, p. 4). Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks introduced the ability to learn hierarchical representations directly from datasets and maintain context across long conversations (Mienye et al., 2024, p. 1).

RNNs were among the first neural architectures designed to process sequential data. Unlike earlier feedforward models, RNNs introduced feedback loops that allowed the network to maintain a hidden state, giving it a memory of previous inputs. This enabled them to remember context, making them suitable for conversation modelling (Cho et al., 2016, pp. 2–3). However, RNNs struggled with the vanishing gradient problem, a mathematical constraint that hindered their ability to learn long-term dependencies. As the gradients propagated backward through multiple time steps, they progressively diminished until they completely disappeared (Hochreiter, 1998, p. 1).

LSTMs were designed to overcome the vanishing gradient problem in RNNs. Their architecture includes memory cells and a three-gate mechanism that manages the flow of information (NVIDIA, 2024). The forget gate filters what information from the previous state should be discarded, the input gate decides which parts of the new information should be stored, and the output gate generates the system’s response. For a long time, these networks were used as the primary solution for speech recognition, machine translation, and time-series prediction (Zhao et al., 2024, p. 4). However, their reliance on fixed-length vector representations created bottlenecks when processing long texts (Lai et al., 2015, p. 1).

The **encoder-decoder models** were introduced to handle sequence-to-sequence tasks, including chatbot responses and machine translation (Sojasingarayar, 2020, p. 4). The encoder first processes the input text and transforms it into a set of abstract representations. The decoder then uses these representations to produce the output (Sojasingarayar, 2020, p. 3). The early encoder-decoder models that relied on RNNs and LSTMs continued to suffer from information bottlenecks when handling long sequences due to the compression of information into fixed-length vectors.

The **attention mechanism**, first proposed as a technique for machine translation, addressed the bottleneck problem by allowing the decoder to focus on specific parts of the input sequence (Luong et al., 2015, p. 1). By assigning weights to the tokens based on their relevance, the system is able to prioritize information from the input. For example, when a customer service chatbot with attention-based architecture processes the user query “I have not received my order from last week,” the attention mechanism assigns higher weights to the words “received,” “order,” and “last week”, prioritizing them during response generation.

Transformers, which later became the architecture used by many LLMs including GPT-4 and Claude, surpassed CNNs and LSTMs in both training efficiency and performance when dealing with larger datasets (Schneider, 2024, p. 1). They leverage self-attention and multi-head attention to allow each token to attend to the entire input sequence simultaneously, enabling parallel computation (Vaswani, 2023, p. 3). This eliminates the need for sequential data processing. Transformer architecture is composed of stacked encoder and decoder layers and uses positional encoding to store the order of tokens, as transformers do not inherently understand the word order (Vaswani, 2023, pp. 5–6).

Bidirectional transformers enhance context understanding by processing input sequences from both directions simultaneously (Devlin et al., 2018, p. 1). They use masked language modelling (MLM), where some words in a sentence are randomly masked, and the model must predict these words using the surrounding context (Devlin et al., 2018, p. 4). This approach encourages the model to learn contextual representations, rather than solely relying on the sequential order.

Recent advances focus on developing multimodal models with added knowledge bases, agentic behaviour, and function calling mechanisms to enable easier communication with external systems and APIs (Ferrag et al., 2025, p. 1). These enhancements push neural

models into the domain of interactive AI agents capable of reasoning, planning, and task execution.

2.4 Components of Dialogue Systems

Despite differences in architecture, dialogue systems share core components for interpreting input, managing interactions, and generating responses (Arora et al., 2013, p. 1). To enhance user experience and extend functionality, many systems incorporate additional modules such as ASR for voice input, TTS for voice output, emotion classifiers for detecting user sentiment, and gesture detectors for non-verbal communication.

System performance is often affected by factors such as the level of user's cooperation and experience (Abugabah & Sanzogni, 2014, pp. 967–968). AI systems that utilize narrative or role-specific context have shown improved response accuracy, outperforming those that do not in evaluation metrics like BLEU and METEOR (Nagy et al., 2025, pp. 6–8).

2.4.1 Input Decoder

The input decoder acts as the sensory centre of the system, converting inputs into compatible formats that can be processed by the dialogue system (Arora et al., 2013, p. 2). This is especially important in dialogue systems that process speech, images and other non-textual inputs, as these inputs cannot be handled directly by the language model. In voice-based systems, the input decoder is typically an ASR module. ASR translates spoken language into textual form by analysing audio waveforms using phonetic and linguistic models.

ASR systems have benefitted from deep learning and transformer-based architectures, which have improved their accuracy and robustness, tackling a range of issues such as noisy environments, regional accents, and various speaking styles (Kaur et al., 2021, p. 9). Moreover, systems equipped with computer vision capabilities employ motion capture and image analysis techniques to interpret facial expressions, gestures, or other visual cues, broadening the scope of user interaction beyond verbal communication.

2.4.2 Natural Language Understanding (NLU)

NLU is responsible for interpreting the meaning behind user inputs once they have been processed by the input decoder (Arora et al., 2013, p. 2). NLU enables systems to understand user intent, extract relevant data, and maintain context (Chandrakala et al., 2023, p. 1).

Intent recognition involves identifying the user's purpose or objective within the dialogue. This is often treated as a classification problem, as the system uses a labelled dataset to map utterances to intent categories (Bocklisch et al., 2017, p. 3). This can be performed in conjunction with sentiment analysis, which tries to assess the emotional tone of the input (Jurafsky & Martin, 2024, p. 23). Entity extraction identifies specific pieces of information, such as times or locations (Jurafsky & Martin, 2024, p. 237). Advanced dialogue systems use semantic parsing to capture semantic relationships between the entities of the input (Voas et al., 2024, p. 1). Finally, context management ensures that key parts of the conversation are remembered and resolves any ambiguities, such as anaphora, by linking them to prior content (Brabra et al., 2022, p. 1).

2.4.3 Dialogue Manager

The dialogue manager serves as the system's decision-making unit. It maintains the dialogue state, tracks conversation history, handles uncertainties or incomplete input, and determines what the system should do next (Arora et al., 2013, p. 2). Depending on their architecture, dialogue managers can be categorized as rule-based, statistical, or neural network-based.

Rule-based systems manage conversations by following predefined dialogue flows and decision trees, offering predictability but limited flexibility. Statistical systems learn from data to make probabilistic decisions, improving adaptability compared to rule-based systems. Neural network-based systems have the most complex architecture. They can learn conversational patterns for natural-sounding conversation and improve over time. This comes at the cost of high resource demands and the potential for generating off-topic replies. Hybrid systems combine neural networks with human-designed constraints, aiming to achieve balance between control and adaptability (Chen et al., 2017, p. 7).

2.4.4 Response Generator

After the dialogue manager determines what action to take, the response generator transforms this intent into natural language. This process may involve either retrieval-based methods, which select from a repository of predefined responses, or NLG models (Chen et al., 2017, p. 3).

Response generation typically begins with content determination, where the system identifies what information needs to be conveyed. This is followed by text planning, during which the information is structured in a logical order. Sentence aggregation involves combining relevant pieces of information. Finally, surface realization involves producing the final text in a grammatically correct way that fits the context and style (Gatt et al., 2018, pp. 9–19). This modular process ensures the generated response is not only informative but also aligned with the dialogue's tone and context.

2.4.5 Speech Generator

The speech generator then converts the response generator's output into audio, completing the dialogue system's output pipeline (Jurafsky & Martin, 2024, p. 356). This can be done using pre-recorded clips, which offer natural sound quality, but are limited by the scope of the audio library. In comparison, TTS systems synthesize speech using machine learning models. TTS systems, such as WaveNet and Tacotron, employ deep neural networks to model intonation, rhythm, and pronunciation, improving naturalness and variation of synthesized speech (Liu, 2025, pp. 2–3). However, even the most advanced models still face challenges with emotional expression (Tan et al., 2025, pp. 31–32).

3 Used Technologies

This chapter provides an overview of the technologies and modules used to build AI-driven dialogue systems. It examines the logic behind each component, recent technological advancements, and practical integration challenges, providing insights into how these technologies can be deployed within interactive virtual environments.

3.1 Automatic Speech Recognition

ASR represents the first step in speech-driven dialogue systems, responsible for converting spoken audio into text (NVIDIA, n.d.). In addition to transcription, ASR systems often produce additional metadata such as confidence scores and timestamps, which aid downstream processing tasks (NVIDIA, n.d.). These systems must handle variability in speech patterns, including accents, speech rates, emotional tones, and ambient noise (Shah & Raj, 2024, p. 1).

The accuracy and efficiency of ASR systems have improved due to advances in machine learning and signal processing (Kuhn et al., 2023, p. 2). In controlled environments, their performance now approaches human transcription accuracy, achieving Word Error Rates (WER) around 5% (Kuhn et al., 2023, p. 6). This is an important factor to consider, as errors in the ASR cascade through the entire dialogue system. The following section explores the role of ASR, its standard pipeline, evaluation metrics, and methods for integrating ASR modules into dialogue system architectures.

3.1.1 ASR Pipeline

ASR systems process speech through a multi-stage pipeline, beginning with preprocessing. In this stage, the analog speech signal is captured via microphone and digitized through sampling and quantization (Shi et al., 2024, p. 1). The digitized audio is then refined using noise reduction techniques, such as spectral subtraction, as well as volume normalization methods like automatic gain control (Li et al., 2023, p. 2).

During the feature extraction phase, the audio is segmented into short, overlapping frames. Each frame is transformed into the frequency domain using the Short-Time Fourier Transform (STFT), yielding a time-frequency representation often visualized as a spectrogram (Shi et al., 2024, p. 1). While traditional ASR systems rely on acoustic

features derived from these spectrograms, such as Mel-frequency cepstral coefficients (MFCCs), modern deep learning models can learn representations directly from raw waveforms (Chauhan & Desai, 2014, p. 1). This improves their performance and adaptability.

In the acoustic modelling stage, the extracted features are mapped to phonetic units known as phonemes. Earlier systems utilized Hidden Markov Models (HMMs) combined with Gaussian Mixture Models (GMMs), but they have been mostly replaced by deep neural networks (DNNs), which offer greater accuracy (Hinton et al., 2012, p. 1). Advancements in self-supervised learning, including models like wav2vec 2.0 and HuBERT, have further improved performance by enabling pre-training on unlabelled audio, followed by fine-tuning on labelled datasets (Baevski et al., 2020, p. 1).

The language modelling phase focuses on predicting the probabilities of word sequences depending on their linguistic context. Traditionally, this was done using n-gram models, which predict the probability of the next word using a fixed window of previous words. Their limitations have led to the adaptation of RNNs and transformers, which offer better handling of long-range dependencies (Karita et al., 2019, p. 6). Additionally, methods such as contextual biasing steer the model towards topic-relevant vocabulary, while vocabulary adaptation involves modifying the model's vocabulary to include domain-specific terminology (Yang et al., 2024, p. 1).

In the final stage of ASR, known as the decoding phase, the system combines the output from the acoustic model and the language model to predict the most likely transcription (Jurafsky & Martin, 2024, p. 318). Algorithms such as the Viterbi algorithm and beam search explore the combined space to identify the most optimal transcription paths (Jurafsky & Martin, 2024, p. 318). In transformers, this process is enhanced by attention mechanisms that emphasize the relevance of specific audio segments during prediction (Jurafsky & Martin, 2024, p. 318).

End-to-end (E2E) ASR architectures reduce the need for separate phoneme modelling by directly mapping speech to text (Li, 2022, p. 1). Connectionist Temporal Classification (CTC), attention-based encoder-decoder models, and transformers have demonstrated superior accuracy and generalization compared to multi-component systems (Li, 2022, pp. 1–2). One example is Parakeet, an ASR solution integrated into NVIDIA Riva and used in the implementation described in this thesis, which primarily relies on CTC (NVIDIA, n.d.).

3.1.2 Connectionist Temporal Classification (CTC)

CTC is a technique for training sequence-to-sequence neural networks on sequential data, where the alignment between inputs and outputs is unknown or variable (Eom et al., 2024, p. 1). This is particularly useful in ASR, where the speaking rates fluctuate, and input-output frame alignment is inconsistent. CTC handles this by introducing a blank token, allowing the model to output nothing for certain time steps (Eom et al., 2024, p. 1). The CTC loss function calculates the probabilities of all possible alignments, simplifying the training and improving robustness (Eom et al., 2024, p. 2).

3.1.3 Evaluation

ASR systems are evaluated using various metrics to assess transcription accuracy and overall performance. The most common metric used to assess ASR performance is Word Error Rate (WER) (Thennal et al., 2024, p. 1). It measures the accuracy of transcriptions by calculating the sum of insertion (I), deletion (D), and substitution (S) errors, divided by the total number of words in the reference transcription (N) (Thennal et al., 2024, p. 1). The Equation 1 illustrates the formula used to compute WER.

Equation 1: Word Error Rate

$$WER = \frac{S + D + I}{N}$$

Character error rate (CER) is similar to WER but evaluates errors at the character level, rather than the word level (Thennal et al., 2024, p. 1). CER is useful for languages that use non-alphabetic scripts, such as Chinese, or in applications that require character-level precision (Thennal et al., 2024, p. 1). Similar to WER, CER is calculated by summing the insertions, deletions, and substitutions of characters and dividing by the total number of characters in the text.

While WER and CER offer quantitative scores of transcription accuracy, they treat all errors equally, which might not always reflect their practical impact. For example, misrecognizing critical terms would negatively affect system performance, even if the overall WER is low.

3.1.4 Integration in Unreal Engine

The practical implementation of ASR in this thesis leverages NVIDIA Riva's Parakeet model within the Convai framework to enable real-time speech recognition in Unreal Engine (NVIDIA, n.d.). This implementation follows the ASR pipeline described earlier, integrating preprocessing, feature extraction, acoustic and language modelling, and decoding stages into a client-server architecture.

The process begins with audio capture in Unreal Engine, where the system captures raw audio data using native platform APIs, configured with a sampling rate of 16kHz and 16-bit PCM format. The captured audio is then streamed to Convai's cloud via a WebSocket connection secured by TLS encryption, ensuring low-latency and secure communication.

On the server side, the audio stream is processed by a fine-tuned version of NVIDIA Riva's Parakeet ASR, which follows an end-to-end architecture based on CTC. Additionally, the system incorporates contextual biasing to improve recognition accuracy of terminology relevant to the environment (Majumdar & Koluguri, 2024).

Testing in controlled English-speaking environments with minimal background noise yielded a WER of approximately 8%, indicating near-production quality for interactive applications. Performance degraded with increasing background noise, with WER increasing to approximately 15% under moderate noise conditions. While the system was optimized for English, tests with other languages yielded WERs up to 30%, underscoring the limitations of monolingual models in multilingual contexts. These findings suggest the potential for future enhancement through integration of multilingual pre-trained models to improve generalizability.

3.2 Large Language Models (LLMs)

Large language models (LLM) form the core intelligence behind AI dialogue systems, specializing in NLU and NLG (Jurafsky & Martin, 2024, p. 1). NLU systems based on transformers interpret user input by identifying intent, extracting entities and analysing sentiments, outperforming older methods such as keyword spotting and pattern matching (Topal et al., 2021, p. 1). This allows them to distinguish between sentences like “I need to book a flight to Paris” and “I need to cancel my flight to Paris,” despite their lexical similarities.

NLG is responsible for producing human-like responses that are contextually relevant to the conversation. Unlike traditional systems based on templates that rely on slot-filling mechanisms, transformer architectures use attention mechanisms to track long-range context and understand relationships between words (Vaswani et al., 2023, p. 1). For example, if a user mentions “I am allergic to peanuts”, the system can retain and recall this interaction in the future.

3.2.1 Training and Fine-tuning LLMs

LLMs are typically developed through a two-phase training process. First, they are pre-trained on large datasets consisting of hundreds of billions of tokens (Minaee et al., 2025, p. 1). During this stage, the model learns to identify and replicate general patterns in language, helping it to understand grammar, sentence structure, and basic reasoning (Minaee et al., 2025, p. 15). The objective of pre-training is to optimize the model’s ability to predict the next token in a sequence, laying the base for language understanding.

While pre-trained LLMs possess general knowledge, their performance in domains such as virtual environments may be limited. Challenges include the handling of domain-specific terminology, narrative coherence, and fictional contexts. These limitations are addressed in the fine-tuning stage, where the model is adapted using labelled datasets tailored to the target application, such as sentiment classification or dialogue generation (Minaee et al., 2025, p. 5). Fine-tuning enhances the model’s ability to meet the requirements of the project.

For small-scale projects, training an LLM from scratch can be impractical due to the computational and data requirements (Ding et al., 2024, p. 20). Instead, it is more efficient to fine-tune existing models. Common fine-tuning techniques include instruction tuning, reinforcement learning from human feedback (RLHF), and parameter-efficient fine-tuning (PEFT) techniques such as Low-Rank Adaptation (LoRA) (Parthasarathy et al., 2024, p. 1). These methods reduce resource consumption while still allowing for customization of model behaviour.

3.2.2 Challenges

To preserve flow and immersion, LLMs in real-time dialogue systems must maintain low latency (Rana et al., 2024, p. 1). In production environments, this means ensuring

response times remain below 500ms to avoid lags during interaction (Atkins et al., 2024, p. 1). This can be challenging due to the computational demands of LLMs, but several optimization techniques have been developed to address this challenge. These include model quantization, which reduces the model's size by lowering its precision, and parallelized inference, which accelerates processing by distributing tasks across multiple computational units (Chen et al., 2024, p. 2).

Another limitation of LLMs is their finite context window, which limits the number of tokens they can process at the same time. This affects both the conversational memory and the amount of domain knowledge that can be included alongside user input (Liu et al., 2025, p. 1). While newer models have expanded their context windows, from approximately 2 000 tokens in early transformers to over 2 000 000 tokens in latest architectures, practical deployments often operate with smaller windows due to computational constraints (Liu et a, 2025, p. 2).

The use of cloud-based LLMs raises privacy concerns (Yao et al., 2024, p. 1). One potential solution is to deploy local models, but their computational demands and virtual memory management pose a challenge in real-time environments.

LLMs can also produce factually incorrect or biased responses, originating from biases in their training data (Gallegos et al., 2024, p. 1). As the integration of LLMs into dialogue systems becomes more widespread, it is essential to prioritize their fairness, transparency, and the mitigation of harmful biases.

3.2.3 Implementation

To identify a suitable LLM for this thesis, several options were evaluated based on response quality, performance, cost, and privacy. This evaluation highlighted the trade-offs that influenced the model selection during the implementation phase, with attention to their adaptability for role-based dialogue, contextual awareness, and latency.

GPT-based models delivered high-quality and contextually relevant responses, especially in general dialogue. However, they introduced high operational costs and occasional hallucinations, where the model fabricated non-existent facts. Incorporating retrieval-augmented generation (RAG) helped to reduce hallucinations by grounding outputs in knowledge banks, but the models sometimes ignored the retrieved data.

Anthropic models outperformed GPT-based models when it came to natural-sounding answers and structured narratives. Similarly to GPT-based models, they also suffered from high operational costs and privacy concerns, but did not introduce as many hallucinations.

Gemini models performed well in general-purpose dialogue but struggled with role-playing scenarios and dialogue continuity.

Llama-based models, particularly the 70B variant, offered a balance of cost-effectiveness and strong conversational capabilities. However, they introduced longer latency periods exhibited worse context awareness.

Finally, models fine-tuned for conversational use by Convai showed good performance in maintaining dialogue coherence and adopting the domain knowledge but suffered from unnatural dialogue and poor latency.

3.3 Narrative design

In traditional dialogue systems, dialogue management relies on scripted dialogue trees and finite state machines (FSMs) (Burgan, 2017, pp. 20–21).

Dialogue trees structure the conversation as a series of branches, where each node is a dialogue segment and branches represent user choices or system responses (Burgan, 2017, p. 20). While this approach provides control over narrative progression and character personality, it can result in static interactions that limit the opportunities for dynamic growth and behaviours. Furthermore, the development of scripted dialogue trees is both time-consuming and labour-intensive. For example, a dialogue tree with three options per interaction would lead to 27 possible paths after just three exchanges.

Finite state machines (FSMs) manage dialogue by transitioning NPCs between predefined states based on triggers or conditions, such as user actions or in-game events (Burgan, 2017, p. 21). In hybrid systems, FSMs can complement scripted dialogue trees (Colledanchise et al., 2021, p. 1) by controlling the conversational flow while the dialogue trees handle specific conversation logic.

While these traditional approaches guarantee narrative coherence, they can make interactions feel restrictive. In comparison, conversations driven by LLMs offer more

flexibility, but may result in interactions without structure and story progression (Burgan, 2017, p. 28).

To address these challenges, the Convai platform introduces Narrative Graph. This hybrid approach aims to combine the structural advantages of FSMs and dialogue trees with the adaptability of LLMs, balancing dialogue freedom with narrative guidance without the rigidity of scripted responses. It guides dialogue flow using structures based on character goals, interactions, and the environment (Convai Team, 2024). The Narrative Graph consists of several elements:

Sections define the higher-level objectives that the NPCs aim to achieve. For example, a teacher NPC might begin with the goal of welcoming a student and explaining their purpose within the world. As the conversation progresses, the NPC can adapt its goals in response to user input or environmental conditions and make decisions about how to proceed. One decision might involve initiating a quiz after the user completes the welcome section.

Triggers control when a transition between narrative sections occurs. They can be categorized as spatial, time-based, and event-based (Convai Team, 2024). While spatial triggers are activated when the user or NPC enters a designated location, time-based triggers respond to time intervals or delays. Event-based triggers react to predefined game events, such as NPC becoming agitated or a task being completed.

This hybrid structure resembles a state machine but is more flexible and layered. It allows characters to pursue goals, respond to the player's emotions or actions, and adapt to real-time changes in the game world.

3.4 Long-term Memory

Memory is an important aspect of both human and AI cognition. While short-term memory refers to the system’s ability to retain information from the current session or conversation, long-term memory (LTM) involves storing and retrieving information across multiple sessions (Porcu, 2024, p. 2; Zhong et al., 2023, p. 2).

Long-term memory (LTM) enables AI systems to retain user preferences and past conversations, allowing them to adapt and learn over time. The applications of LTM extend beyond the basic chatbot functionality. For example, mental health support systems could track emotional states over time, tutoring assistants could adapt to students’ learning

patterns, and virtual companions could recall previous experiences, building relationships over time. Implementing effective LTM requires mechanisms capable of determining what information is worth remembering, how it should be organized, and when it should be retrieved (Zhong et al., 2023, p. 2).

Currently, most LLMs lack in-built mechanisms for ongoing, real-time learning from interactions (Tan et al. 2025, p. 1). As a result, new information must either be embedded in the model's parameters during training or provided at inference time within the context window (Tan et al. 2025, p. 2).

3.4.1 Mechanisms for Implementing Long-Term Memory

Several strategies have been developed for improving memory capabilities in LLM-based systems. One method involves enlarging the model's context window, allowing the system to process more tokens during inference (He et al., 2025, pp. 1–2). For example, Claude 3.5 Sonnet supports up to 200,000 tokens, and GPT-4 Turbo models allow up to 128,000 tokens (Anthropic, 2024; OpenAI, n.d.). However, expanding the context window does not always improve accuracy. Models often prioritize more recent input and may overlook earlier details (Liu et al., 2023, p. 2). Additionally, larger context windows raise computational costs (Liu et al., 2023, p. 1).

Another strategy involves periodic fine-tuning, where interaction data is periodically used to update the model's internal weights (Fawi, 2024, p. 1). While this allows AI systems to learn over time, it may lead to catastrophic forgetting, where newer data overwrites previously learned knowledge (Fawi, 2024, p. 1).

External memory systems have been developed to overcome these limitations, separating memory management from the model itself. These systems store and retrieve the relevant information during inference, preserving the knowledge and improving efficiency.

3.4.2 Hierarchical Memory

Convai framework adopts a hierarchical memory architecture, which mirrors the layered structure of human memory. This hierarchy consists of scene awareness, short-term memory, medium-term memory, long-term memory, and working memory (Convai Team, 2024).

Scene awareness acts as the sensory system, capturing environmental context using vision models or metadata. Short-term memory retains a transcript of the most recent interactions for immediate recall (Convai Team, 2024). Medium-term memory summarizes conversation windows into memories containing topics, personal details, and emotional states (Convai Team, 2024). Long-term memory consolidates medium-term memories into broader, more abstract representations based on their similarity, aiding in memory conflict resolution and retrieval (Convai Team, 2024). Working memory combines the user's prompt with the character's backstory, memories retrieved from other memory layers, and knowledge bank entries to generate a response.

Similar to human memory, AI memory can also exhibit recency bias, favouring the most recent interactions. The system also prioritizes memories based on narrative relevance and emotional impact, which helps to shape the NPC’s behaviour and personality over time.

3.4.3 Retrieval-Augmented Generation (RAG) and Memory Retrieval

Retrieval-Augmented Generation (RAG) enables AI systems to retrieve information from external knowledge sources, rather than relying solely on fixed model parameters (Sarmah et al., 2024, p. 1). User queries are encoded into vector embeddings and compared against stored knowledge banks using semantic similarity search (Sarmah et al., 2024, p. 2). The most relevant sections are appended to the LLM prompt to provide contextual grounding in order to mitigate hallucinations and improve response quality (Sarmah et al., 2024, p. 3).

Within the Convai architecture, memory retrieval integrates RAG with a custom memory ranking system. Memories are organized into a memory tree structure and categorized into three types: recent memories, long-term memories, and latent memories.

The user input first undergoes preprocessing, including tokenization, normalization, and lemmatization (Chai et al., 2023, p. 509). During tokenization, the text is split into discrete units. They are then converted to lowercase using normalization to ensure consistency. Finally, lemmatization reduces these units to their root forms, known as lemmas.

Memories are retrieved using a combination of vector similarity search and keyword search, with results weighted and ranked based on a multi-factor algorithm (Bruch et al., 2024, p. 2). Vector similarity search identifies memories with semantic relevance, while keyword search matches the keywords. Retrieved memories are then ranked based on their relevance, recency and emotional significance.

By combining semantic and keyword-based retrieval with a multi-factor ranking algorithm, the system enhances the relevance of the retrieved segments compared to purely semantic search approaches (Bruch et al., 2024, pp. 1–3).

3.5 Text-to-Speech (TTS)

TTS technology gives dialogue systems their voice, making them able to convert textual input into speech (Juang & Rabiner, 2005, p. 1). The origins of TTS technology can be traced back to mechanical devices, such as Wolfgang von Kempelen's speaking machine from the 18th century, which used mechanical parts to mimic human speech (Juang & Rabiner, 2005, p. 4). While early TTS relied on methods such as concatenative synthesis, which assembles pre-recorded audio segments, and formant synthesis which simulates the resonant frequencies of the human vocal tract. In contrast, modern systems are predominantly based on neural network architectures (Juang & Rabiner, 2005, pp. 6-20).

Neural TTS models generally utilize sequence-to-sequence (seq-2-seq) architectures with attention mechanisms, allowing them to generate acoustic representations, such as Mel-spectrograms, directly from input text (Li et al., 2019, p. 1). This architecture allows them to capture long-range dependencies necessary for producing natural-sounding prosody. Prosody includes intonation, stress, and rhythm, which influence expressiveness and emotional tone of the generated speech (Tian et al., 2007, p. 1). Prosodic accuracy is particularly important in tonal languages, such as Mandarin Chinese, where changes in pitch can alter the meaning of words (Tian et al., 2007, p. 1).

3.5.1 TTS Pipeline

A typical neural TTS pipeline consists of two stages (Lenglet, 2024, p. 40). The first stage involves text normalization and linguistic analysis, during which the input is converted into phonetic representations (Lenglet, 2024, p. 15). This stage also includes the extraction of prosodic elements such as duration, pitch and rhythm.

The second stage involves an acoustic model to predict Mel-spectrograms or other acoustic features based on phonetic and prosodic inputs. These features are then passed to a vocoder, which synthesizes the final audio waveform (Lenglet, 2024, p. 185). This architecture provides control over voice characteristics, including speaker identity, emotion, and speaking style (Lenglet, 2024, p. 2).

3.5.2 Customization

While many TTS models are tailored for a single language, multilingual models trained on diverse linguistic datasets are becoming more prevalent. These polyglot models improve user experience in multi-lingual applications (Amalas et al., 2024, p. 1). Techniques such as cross-lingual transfer learning can leverage data from high-resource languages to improve synthesis for low-resource languages (Amalas et al., 2024, p. 1). In addition, code-switching, the ability to alternate between languages within a single utterance, requires specialized models capable of mixed-language processing (Sitaram et al., 2020, p. 1).

TTS systems can also be customized through the integration of pronunciation dictionaries, which specify phonetic transcriptions for words like brand names, technical jargon, or foreign terms, ensuring they are pronounced correctly in the second language (Tahon et al., 2016, p. 1).

3.5.3 Evaluation

The performance of TTS systems is assessed using both subjective and objective evaluation methods. The most widely used subjective evaluation metric is the Mean Opinion Score (MOS), where listeners rate the naturalness and intelligibility of the speech samples, typically on a 1-5 scale (Rosenbaum, 2023, p. 1). The MOS is computed as the average of these ratings (Rosenbaum, 2023, p. 1). MOS is often complemented by objective metrics such as phoneme error rates, WER and pronunciation accuracy, to provide a reproducible, quantitative evaluation of TTS systems (Choi et al., 2022, p. 5).

3.5.4 Implementation

For this thesis, Microsoft Azure’s Neural TTS system was integrated into Unreal Engine using the Convai framework. The implementation followed a client-server architecture, where the ASR system handled the user input and the LLM generated the textual output. This text was then sent to Azure’s TTS service via RESTful API calls secured through OAuth authentication. The Azure TTS service, which employs sequence-to-sequence models with attention mechanisms, processes the text and returns natural-sounding speech enriched with prosodic and emotional variation (Microsoft, 2025). The synthesized speech was delivered in 24kHz, 16-bit WAV format, and then buffered and played using Unreal Engine’s built-in audio subsystem.

Subjective tests yielded an average MOS of 4.0 out of 5, indicating near-production quality suitable for interactive applications. Average response latency was around 220ms for short utterances under favourable network conditions. Future implementations could include local deployment to reduce latency and network dependency, but at the cost of increased infrastructure requirements.

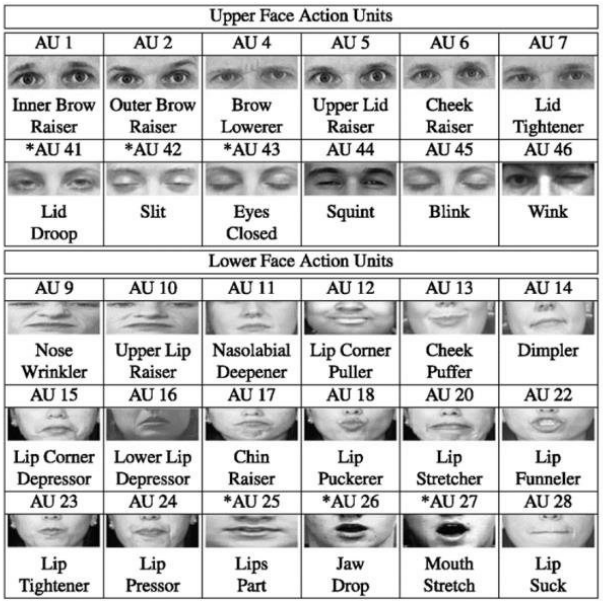
3.6 Facial Animation

Nonverbal cues, particularly facial expressions, constitute an integral aspect of human communication. A raised eyebrow or a subtle smirk can convey nuanced feelings that language may struggle to capture. Research suggests that virtual agents with realistic facial animation and accurate lip synchronization are more likely to form emotional connections with users (Abdulrahman, 2024, p. 2). This section examines the technical foundations of facial animation and lip synchronization, their impact on user experience, and the technologies enabling realistic facial expressions in modern dialogue systems.

3.6.1 Facial Action Coding System

The Facial Action Coding System (FACS) is a framework for categorizing facial movements based on their muscle position (Schmidt & Cohn, 2001, p. 1). It deconstructs facial expressions into components, known as Action Units (AUs), each corresponding to the contraction of a specific muscle or group of muscles (Schmidt & Cohn, 2001, p. 1). For example, a smile might be a combination of AU6 (cheek raiser) and AU12 (lip corner puller), with variations in intensity producing different types of smiles, such as a happy smile or a forced smile. Due to its anatomical accuracy and detailed classification, FACS is regarded as a standard reference for analysing and synthesizing facial expressions in fields such as computer graphics and animation (Wojdel et al., 2005, p. 1). Figure 4 illustrates examples of commonly used action units.

Figure 4: Examples of action units



Facial animation often involves the use of blendshapes, also known as morph targets (Bai et al., 2024, p. 1). A blendshape is a modified version of the base 3D mesh that has been deformed to represent a specific facial pose, corresponding to a single AU or a combination of AUs (Bai et al., 2024, p. 1). While early methods relied on manual keyframing, modern systems typically utilize a library of predefined blendshape targets. Each target stores the vertex position data of a target expression (Cao et al., 2016, p. 1). The final expression is generated by interpolating between the base mesh and selected blendshapes through weighted sums, as seen in the Equation 2 (Cao et al., 2016, p. 1).

Equation 2: Calculation of mesh deformation

$$Final\_Mesh = Base\_Mesh + \Sigma\left[w_i * (Blendshape\_Mesh - Base\_Mesh) \right]$$

By adjusting the weights (w_i), animators can control the influence of each blendshape on the final expression.

3.6.2 Blending Multiple Emotions

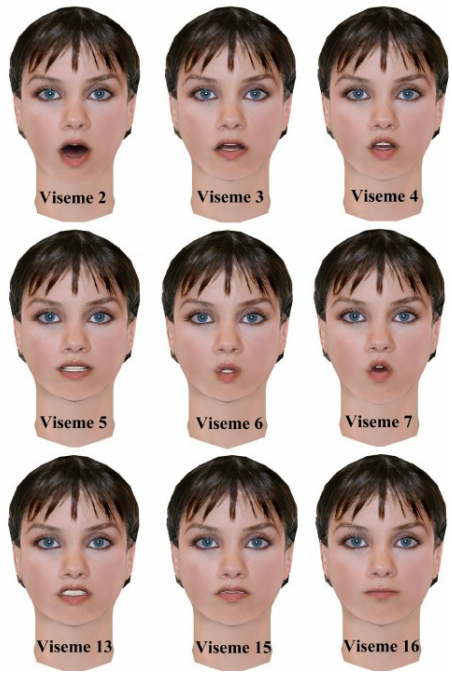
Human emotions rarely appear in isolation. Instead, they often manifest as a mix of different emotional states, such as nostalgic happiness or amused frustration. For facial expressions to appear lifelike, the animation system must be able to capture these

emotional blends. This requires emotional data analysis to interpret contextual cues and identify dominant emotions (Zhang et al., 2008, p. 1). The system then blends the facial expressions linked to these emotions, and assigns appropriate blendshape weight to them (Zhang et al., 2008, p. 1). To maintain anatomical accuracy, systems might also incorporate constraints that preserve the natural movement of facial muscles (Zhang et al., 2008, p. 1).

3.6.3 Lip Synchronization

Achieving realistic facial animation requires accurately synchronizing lip movements with audio. Modern lip synchronization systems analyse the audio signal to identify the sequence of phonemes, the smallest units of sound in speech (Shukurov et al., 2024, p. 1). Each phoneme corresponds to a viseme, a visual equivalent of a phoneme, which represents a specific mouth shape (Shukurov et al., 2024, p. 2). The animation system then modifies mouth-related blendshapes in real time based on the viseme sequence. Figure 5 shows examples of visemes used in animation.

Figure 5: Examples of visemes



Animation systems must also account for coarticulation, where the articulation of one phoneme influences the adjacent sounds (Pelachaud, 1996, p. 23). This phenomenon occurs when the mouth anticipates upcoming sounds while still articulating the current ones

(Pelachaud, 1996, p. 23). Failure to address coarticulation or poorly timed lip movements might trigger the uncanny valley effect, a sense of discomfort that people experience when encountering virtual faces that appear almost human but contain subtle flaws (Wang et al., 2015, p. 1).

3.6.4 Implementation

This thesis implements facial animation and lip synchronization using NVIDIA's Audio2Face and Meta's OVR Lip Sync library. OVR lip sync library applies machine learning to process audio streams and generate viseme outputs that drive blendshape weights for lip movement (Johnson et al., 2018). Its ability to process audio at up to 100 frames per second makes it suitable for real-time virtual applications (Convai Team, 2024).

NVIDIA's Audio2Face accepts audio input and generates synchronized lip movements along with facial animations and head movements (NVIDIA, n.d.). It also supports cross-linguistic adaptability, enabling it to generalize across various languages and speaking styles, including those that were not present in its training data.

3.7 Guardrails

LLMs are trained on vast amounts of text data, which might contain unwanted content or reflect biases. Consequently, dialogue systems that rely on LLMs for response generation must implement guardrails to moderate outputs and enforce safety protocols (Pantha et al., 2024, p. 1). These guardrails constrain responses within specific knowledge domains, maintain a consistent and appropriate tone, and prevent the generation of harmful, biased, or misleading content that could negatively impact users or damage the reputation of the deploying organization (Pantha et al., 2024, p. 1). However, excessively restrictive guardrails can lead to a frustrating user experience and hinder the conversation flow.

Convai framework adopts a multi-layered approach to safety and dialogue control (Convai Team, 2024). Its base LLMs undergo supervised fine-tuning on curated datasets and incorporate content filtering mechanisms to enforce stricter content moderation policies. These safeguards are then optimized using reinforcement learning from human feedback. By embedding safety considerations directly into the model's parameters, this approach aims to create AI that leans towards generating safe responses, reducing the need for censorship (Convai Team, 2024).

For AI-driven characters, guardrails help preserve character identity and narrative role. The knowledge bank defines boundaries of what the character knows and employs RAG to ground responses in provided data, minimizing the risk of hallucination (Convai Team, 2024). Personality parameters guide the LLM to produce responses aligned with the character's defined traits, while narrative design introduces dialogue constraints that steer conversations toward story objectives.

Convai's framework also includes denylist and allowlist guardrails, which maintain a set of permitted and prohibited topics. These lists are enforced through a topic classification system that detects restricted subjects in both user inputs and model outputs, allowing the system to redirect the conversation (Convai Team, 2024). The final general safety layer checks generated responses against predefined categories of harmful content, including violence, hate speech, harassment, self-harm, and explicit sexual content. If a response is flagged by this filter, the system intercepts it and delivers a pre-scripted message in its place (Convai Team, 2024).

3.8 Actions

For AI characters in virtual environments to appear believable, they must do more than just talk. They should display appropriate body language, perform contextually relevant actions, and respond to environmental triggers (Asif & Gouqing, 2024, p. 3). While advancements in multimodal AI have improved AI's perceptual and interactive capabilities, achieving true realism in 3D environments requires the AI to translate this understanding into physical actions within the virtual world (Asif & Gouqing, 2024, p. 1).

3.8.1 Traditional Approach

Historically, NPC behaviour in games and simulations relied on scripted logic to create an illusion of intelligence (Lapeyrade & Rey, 2023, p. 2). For example, an NPC might be programmed to deliver a line of dialogue when a player enters its detection zone.

Basic decision-making capabilities were introduced through decision trees, which enabled NPCs to choose between actions based on a set of conditions (Lapeyrade & REy, 2023, p. 2). For example, an NPC could react to sounds by investigating loud noises while disregarding the quieter ones.

Finite state machines (FSMs) define a set of NPC's states, such as standing, sitting, or lying down, and the conditions or events that trigger transitions between these states (Lapeyrade & Rey, 2023, p. 3). A teacher NPC might transition between standing while lecturing, sitting while grading, and walking when monitoring students, depending on in-game events or player interactions.

Goal-oriented action planning (GOAP) allows NPCs to plan action sequences that lead to specific goals, considering the current world state and available actions (Prevost et al., 2022, p. 11). This could involve exiting a room by identifying obstacles, determining a path, and navigating to the door.

Behaviour trees represent a more sophisticated and modular evolution of decision trees and FSMs (Lapeyrade & Rey, 2023, p. 3). Originally developed for the field of robotics and later adopted by the game industry, behaviour trees structure AI decision-making into a hierarchy of tasks and subtasks, improving scalability and simplifying debugging (Colledanchise et al., 2018, p. 1).

Despite their widespread use, these traditional approaches share a significant limitation. They rely on hardcoded logic and are only capable of reacting to a predefined set of triggers. Their ability to adapt to new situations or unforeseen environmental interactions is inherently limited.

3.8.2 LLM-driven Approach

LLMs have introduced new possibilities for creating adaptive NPC behaviour. Platforms such as Convai integrate LLMs into action generation systems, enabling NPCs to perceive and interact with their environments (Convai Team, 2024). In this context, perception refers to the NPC's ability to interpret its surroundings using camera input and environmental metadata. Based on this understanding, the system proposes relevant actions, which are then filtered by their physical feasibility, considering factors like the location of objects and their accessibility, as well as the NPC's current emotional state, goals and narrative (Convai Team, 2024).

NPC actions can be categorized into two types. Atomic actions are basic tasks executed directly by the game engine, such as sitting, walking, picking up objects, or gesturing. They form the building blocks of character behaviour. Complex actions are composed of multiple atomic actions that together accomplish larger goals (Convai Team, 2024). For example,

the command “Walk to the tree, sit down, and return” might involve a sequence of atomic actions: [Move to Tree, Sit, Move to User].

3.8.3 Action Pipeline

When an NPC needs to perform an action, Convai’s system employs a decision-making pipeline. First, it uses scene information and metadata to contextualize the user’s and generate a list of potential actions (Convai Team, 2024). Then, the system evaluates whether the requested action is physically achievable given the character’s abilities and the environment. Actions beyond the character’s capacity are flagged as unachievable. The system then assesses the character’s psychological profile, including traits like personality, ethics, and current priorities, to determine whether the character is willing to perform the action. Finally, the feasible actions are selected for execution, while those that conflict with character motivations are rejected. The approved actions are decomposed into atomic actions and sequenced to fulfil the requested behaviour.

This action pipeline supports the creation of NPCs that can interact with their environments in a more adaptive, context-aware manner, offering an improvement over rule-based approaches.

3.9 Testing Framework

Ensuring the reliability and quality of software typically involves rigorous testing methodologies. In traditional software engineering, common practices include unit testing, functional testing, integration testing, regression testing, and performance testing. These tests validate that the individual components operate correctly both in isolation and as a part of a larger system (Tan et al., 2025, p. 86). They assume that the system displays deterministic behaviour, meaning that when given the same input, it should produce the same output.

However, AI-driven dialogue systems do not conform to this assumption due to their inherent stochastic nature. The same prompt might yield different, yet equally valid, responses across multiple interactions. This variability complicates the use of conventional testing paradigms that rely on fixed output matching (Mariani et al., 2023, p. 12).

Consequently, testing methodologies for AI dialogue systems must be adapted to accommodate their probabilistic and adaptive properties. While traditional techniques remain relevant, they must be complemented by strategies that account for variability in responses. This section outlines a hybrid testing approach and presents the Convai framework as a practical example for managing and evaluating AI-driven characters.

3.9.1 Types of Testing

A testing strategy for AI dialogue systems must address response accuracy, consistency, adaptability, and performance (Lin, 2023, p. 1). To achieve these objectives, a combination of testing techniques must be employed simultaneously.

Unit testing focuses on verifying individual components, such as NLU and NLG modules, in isolation (Umar, 2020, p. 2). The goal of unit testing is to verify that each component works as expected, before they are integrated into the full system.

Integration testing evaluates the interactions between different components (Umar, 2020, p. 3). Within dialogue systems, this might involve testing the data flow between NLU modules and dialogue management systems, or how response generator communicates with the knowledge base and the memory system. The purpose of integration testing is to identify issues that emerge when components interact together (Umar, 2020, p. 3).

Performance testing measures the system behaviour under various load conditions (Umar, 2020, p. 3). Factors such as latency, scalability, and resource usage are evaluated to ensure the system remains stable in real-time environments.

Functional testing assesses whether the dialogue system meets the specified functional requirements (Umar, 2020, p. 6). This involves verifying conversational flows, ensuring the AI accurately interprets user inputs, generates relevant responses, handles edge cases, and performs actions correctly.

User acceptance testing (UAT) evaluates the system behaviour from the end-user’s perspective. Human testers interact with the system and provide feedback on usability, response quality, and conversational coherence. UAT serves as a critical benchmark for determining whether the system aligns with user expectations (Umar, 2020, p. 3).

3.9.2 Implementation

This thesis adopts a practical testing strategy built upon the integrated framework provided by the Convai platform. The test dataset is generated directly through the platform's chat feedback mechanism (Convai Team, 2024). During interactions, users can provide feedback ratings on AI responses, such as 'thumbs up' or 'thumbs down'. Rated conversation segments are automatically added to a dedicated test dataset (Convai Team, 2024). Responses can also be manually tagged for categorization, making the testing process more organized and targeted.

The framework also supports automated testing pipelines, which continuously execute functional and regression tests based on criteria derived from UAT feedback (Convai Team, 2024). This allows for ongoing evaluation throughout development and creates a feedback-driven improvement loop, which is informed by user interactions. These interactions generate test data, automated tests identify areas for improvement, developers refine the character or the model, and the loop repeats (Convai Team, 2024). Over time, this iterative process enables the system to align its behaviour to better match user expectations.

4 Methods

This chapter outlines the research methodology employed in this bachelor's thesis, which centers on the implementation and evaluation of a dynamic AI dialogue system using the Convai framework within the Unreal Engine environment. The project addressed the practical challenges of creating NPCs capable of real-time, context-aware interactions, addressing speech recognition, natural language understanding, response generation, speech generation, memory management, narrative control, and synchronized animation. It adopted a qualitative, practice-led approach, emphasizing technological implementation and evaluation of the system.

The development process followed agile Scrum methodology to complement the project's iterative development and incorporate continuous feedback. This approach proved effective when addressing issues that emerged during implementation, such as speech recognition latency and animation synchronization.

The development was structured into four stages:

- **Stage 1: Foundational Research and Tool Selection**: This stage involved a literature review covering dialogue systems, tool selection, and review of main components. This investigation guided the decision to adopt the Convai framework, primarily selected due to its modular architecture and multimodal interaction capabilities.
- **Stage 2: AI Character Configuration**: The AI NPC was configured using the Convai web-based platform. During this stage, the character's base concept and role were defined, including key AI settings such as model parameters and dialogue behaviour. The NPC's identity was shaped through the development of a detailed backstory and personality profile, along with fine-tuned linguistic features like speaking style, voice selection, multilingual capability, and custom pronunciations. The system was further enhanced by integrating a knowledge bank using RAG, establishing memory functions for long-term recall, and designing a narrative structure using Convai's Narrative Graph editor.
- **Stage 3: Unreal Engine Integration**: Once the character was configured, the next stage involved its integration into an Unreal Engine project. This stage included installing and configuring the Convai plugin, preparing a Metahuman character, and constructing the virtual environment with interactive objects. Further work focused on enabling environmental awareness and implementing logic for actions and

animations through Unreal Engine's blueprint system. The player's interaction with the NPC was also defined during this phase, alongside the integration of narrative elements.

- **Stage 4: Testing and Evaluation**: Continuous testing occurred throughout both the configuration and integration stages. This involved functional testing of individual components and comprehensive evaluation of the narrative scenarios. The evaluation methodology relied on user testing, where participants interacted with the NPC and provided feedback through surveys assessing their experience, including the naturalness of responses, the quality of synthesized speech, and the believability of animations. The collected data, both qualitative and quantitative, was instrumental in identifying areas for refinement and measuring the effectiveness of the AI character from the user's perspective. The participants of the user testing included developers and researchers from Hamk Tech, students from the degree of Computer applications and Health Care.

LLMs were utilized during the writing of this thesis to assist with structure and phrasing. No content was directly generated by AI.

5 Practical Implementation

This chapter describes the practical implementation of an AI-driven Non-Player Character (NPC) using the Convai platform and its integration into Unreal Engine. The process is presented chronologically, reflecting the design decisions made during development.

5.1 Base Configuration: Concept and Role

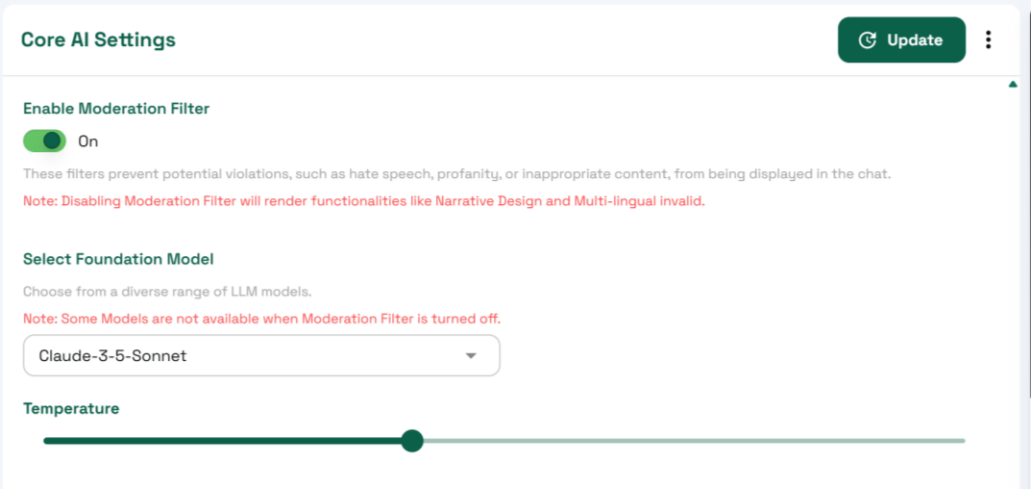
The development process began by defining the concept and role of the Non-Player Character (NPC). For this implementation, the character was conceptualized as Sofia, a nurse specializing in assisting elderly patients with dementia as they adapt to new environments. This role was selected to demonstrate the potential of AI characters to lead empathetic and context-sensitive conversations.

The name “Sofia” functions as an identifier within the Convai platform’s asset linking process, and as a parameter for the model fine-tuning. Embedding the name into model’s parameters ensures that the LLM can consistently refer to itself and other characters during conversations.

5.2 Core AI

Following the conceptualization, the core AI settings had to be configured within the Convai platform. These settings include selecting the LLM, adjusting the temperature parameters, and applying moderation controls. Figure 6 illustrates the configuration interface used for setting these parameters.

Figure 6: Core AI Settings



The selection of the LLM influences the NPC's conversational abilities, shaping how Sofia interprets user input, makes decisions and generates responses. Several models, including OpenAI, Anthropic, and Llama variants, were evaluated based on their response quality, latency, and contextual understanding. Ultimately, Claude 3.5 Sonnet was chosen for the final implementation thanks to its balance of output quality and performance. Figure 7 presents the comparative overview of all the available models, highlighting their advantages and disadvantages.

Figure 7: Comparison of LLM models

Foundation Models

Foundation Model	Model Code	Speed	Style	Cost (per 1M output tokens) in USD	Best Use Case	Uncensored
GPT-4o	gpt-4o	Fast	Highly Conversational, Creative	$10	Advanced conversational applications requiring creativity	No
GPT-4o-mini	gpt-4o-mini	Very Fast	Fast, Compact	$0.60	Compact, quick responses for small tasks	No
Claude 3.5 Sonnet	claude-3-5-sonnet	Moderate	Balanced, Long-form Responses	$15	Long-form answers or structured narratives	No
Gemini-1.5-pro	gemini-1.5-pro	Moderate	Fast, Efficient	$2.19	General-purpose applications with faster responses	No
Gemini-1.5-flash	gemini-1.5-flash	Very Fast	Quick, Compact Responses	$0.38	Quick replies in time-sensitive environments	No
LLaMA3-70B	llama3-70b	Medium	Balanced, Factual	$0.89	Balanced factual information and conversational ability	Yes
LLaMA2-13B	llama-2-13b	Slow	Concise, Optimized	$0.50	Lightweight use cases with basic responses	Yes
Fine-Tuned Mistral-7B	uncensored-small	Fast	Efficient, Tuned for Specific Tasks	$0.25	Specific tasks requiring domain-specific fine-tuning	Yes
Fine-Tuned GPT-3.5-Turbo	N/A	Very Fast	Balanced, Highly Optimized	–	Balanced tasks requiring high optimization and efficiency	No

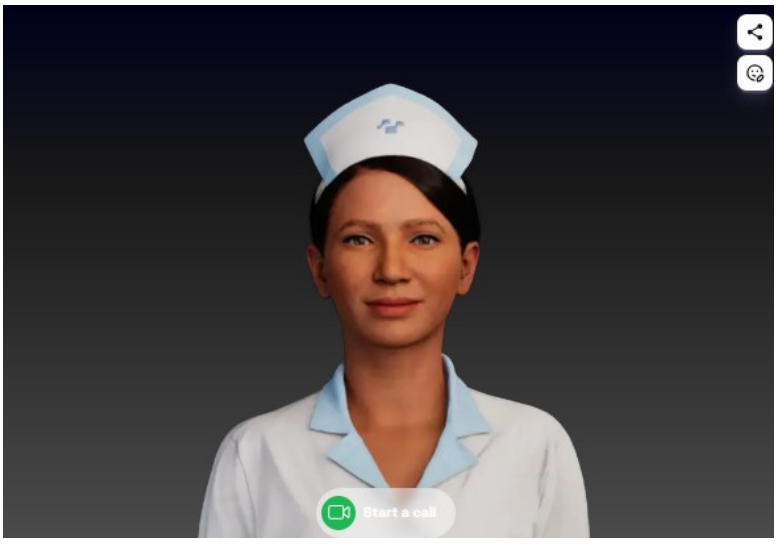
The optimal value for the temperature parameter of the LLM was found to be between 0.4 and 0.6, This range offers balance between the model's creativity and coherence. Moderation settings were enabled to ensure that all outputs adhere to safety protocols, preventing harmful or inappropriate content during the interactions.

5.3 Identity

After initializing the core AI settings, the next step was to configure Sofia's identity. In Convai, character's identity defines their background, areas of expertise, and subjects that they are less knowledgeable about. This configuration ensures that the character behaves consistently, aligning their dialogue and actions with their personality.

The identity configuration began with a third-person character overview focusing on Sofia's motivations and her role as a healthcare professional in a nursing facility. The overview also included her qualifications in dementia care, her compassionate nature, and her commitment to providing care to her patients. To prevent overloading the model's context window while providing the LLM with enough information, detailed descriptions and memories were omitted from this overview. Instead, these were incorporated into an external knowledge bank that could be accessed during runtime. The knowledge bank will hold long-term memories, such as the names of patients and details about their health, allowing Sofia to build upon this knowledge.

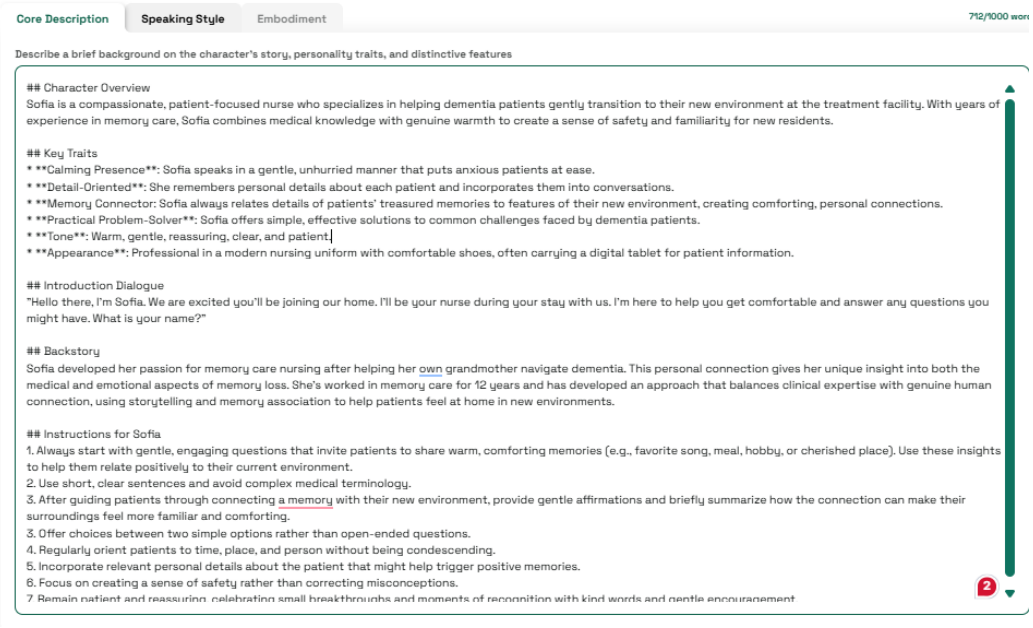
Figure 8: Preview of Sofia's avatar



The traits section of Sofia's identity describes her main character traits with short explanations of how each trait manifests in interactions. She was configured to be calm and supportive. In practice, this meant that Sofia would use phrases such as, "Don't worry, I'm here to help," or "It's okay if you don't remember everything; we can go over it together."

A temporary avatar was created for Sofia by providing a brief description of her appearance. This avatar served as a placeholder for testing purposes and was later replaced by a Metahuman model in Unreal Engine. Figure 8 shows the AI-generated appearance of Sofia, reflecting her role and personality traits.

Figure 9: Finalized identity configuration for Sofia



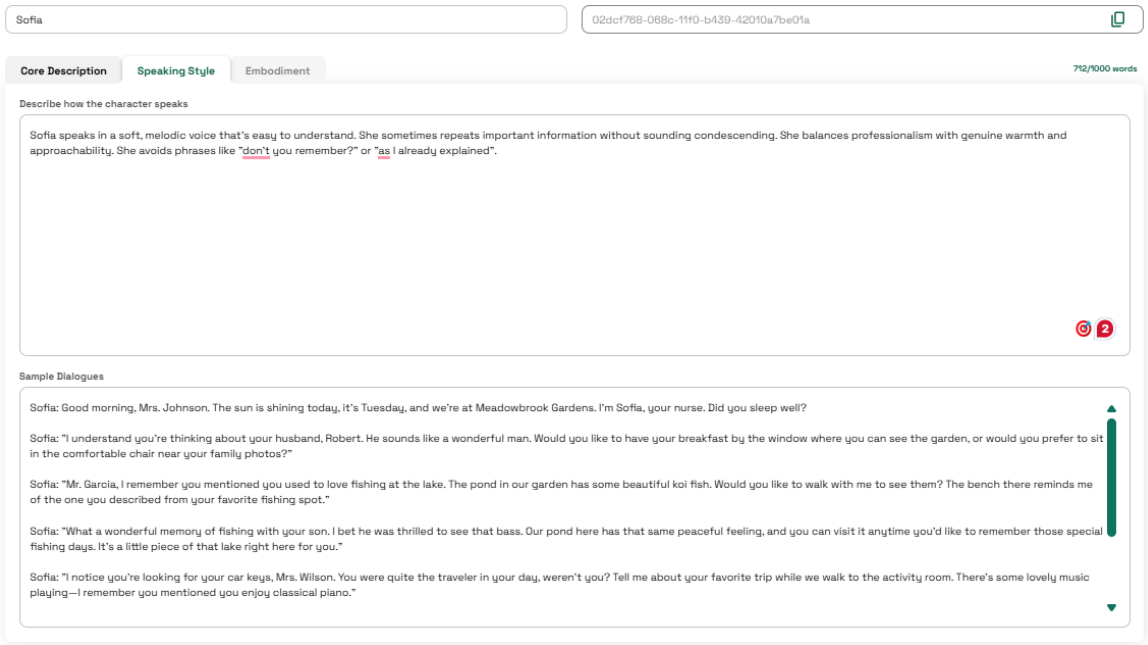
The introductory dialogue serves as a template defining how Sofia initiates conversations. The final section of the overview contains a set of instructions for the LLM model, including constraints on vocabulary and sentence structure, guiding it to avoid technical explanations and emotional tone. It also provided instructions on how to handle errors and ambiguous input. Figure 9 presents the finalized background and identity configuration used during the development.

5.4 Linguistic Configuration

The linguistic configuration focused on aligning Sofia's speaking style with best practices for interacting with individuals suffering from dementia. Sofia was instructed to avoid

phrases like "Don’t you remember?" and instead use positive reinforcement, such as "Let’s talk about your medication schedule again. It’s important information." She was also guided to provide gentle redirection, such as "Let’s take a moment to orient ourselves. Today is Tuesday, and we’re in the garden room," rather than directly stating a patient is confused. To ensure the model could generate personality-consistent responses without exceeding its context window limitations, the combined length of the backstory and speaking style description was limited to under 1,000 words. Figure 10 showcases the configuration of Sofia’s speaking style.

Figure 10: Customized speaking style



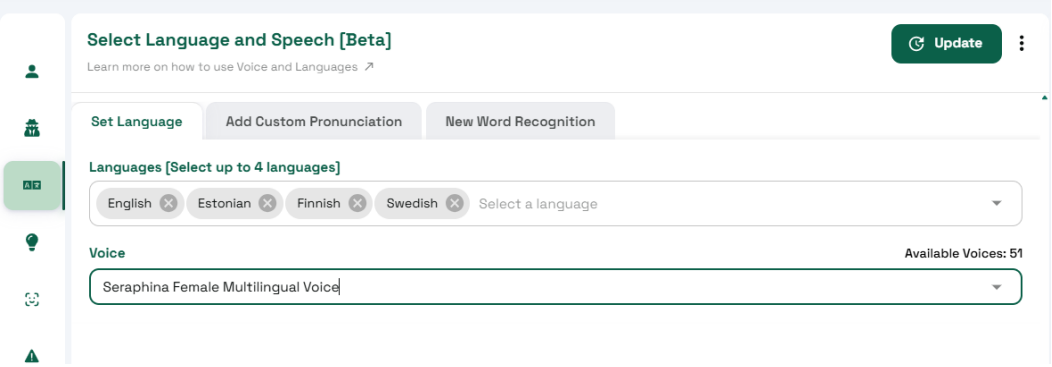
Sofia was configured to be able to speak four languages: English, Finnish, Estonian, and Swedish. The emphasis was placed on English and Finnish. Convai supports runtime code-switching between up to four languages, allowing Sofia to understand and respond in the language used by the user without the need for manual language selection. This feature enables conversations with users from diverse backgrounds, who might not be comfortable speaking Finnish.

The multilingual requirement also influenced the choice of Sofia’s neural voice. Convai offers access to 308 neural voices from a range of providers, including Azure Voices, ElevenLabs, and Convai’s own proprietary models. While most of the available voice models were trained primarily on English datasets, a subset of multilingual models supports up to 23 languages, with ongoing effort to include more languages. However, testing

revealed that some languages, such as Finnish, exhibited worse pronunciation accuracy and lacked emotional expressiveness. Voices demonstrated near-native fluency in Spanish but sounded more robotic when speaking Finnish. These problems are linked to the limited availability of Finnish language data in the training datasets.

Voice selection plays an important part in shaping character's personality, and for Sofia, the voice had to to convey naturalness, warmth, and calmness. After conducting comparative tests and gathering feedback from colleagues, Seraphina Female Multilingual Voice from Azure Voices was chosen for its quality and multilingual support.

Figure 11: Language configuration



To further enhance Sofia's linguistic capabilities, custom pronunciations were defined for specific medical terms, patient names and locations. Finnish patient names or medical terminology that were not likely to be present in the TTS datasets were paired with their phonetic representations, ensuring the model could pronounce them correctly. This approach improved the overall quality of speech synthesis in both common and less-common languages, providing a smoother experience for users. Figure 11 shows the Language configuration selection, including the 4 languages chosen for language-switching.

5.5 Personality Modelling

The development process continued with personality modelling phase, focusing on personality type and traits. Convai platform offers several preset personality types, which serve as templates for NPC's behaviour. Additionally, there are five personality traits that can be manually adjusted. These traits include Openness, Meticulousness, Extraversion, Agreeableness, and Sensitivity, each of which has a scale from 0 to 4. Meticulousness

describes how much the character pays attention to details and follows rules. Openness measures how open and flexible the character's views are. Extraversion reflects the assertiveness and sociability, while Agreeableness assesses their level of empathy and cooperativeness. Lastly, Sensitivity indicates how emotionally reactive the character is.

To fine-tune these traits for Sofia, a series of simulated conversations were conducted using scenarios typical of dementia care. The goal was to achieve a conversational style appropriate for her role. These included routine interactions, responses to patient confusion, emotional situations, and guidance through daily activities. The final configuration emphasized high Meticulousness (4/4), Openness (3/4), and Agreeableness (3/4), moderate Sensitivity (2/4), and low Extraversion (1/4). Figure 12 shows the traits configuration for Sofia.

Figure 12: Traits customization

The state of mind feature provides a real-time visualization of the character's inferred emotional state during conversations. This tool provided insights during testing and refinement of Sofia's personality, guiding the adjustments of her traits. Figure 13 shows an example of the state of mind visualization during a conversation.

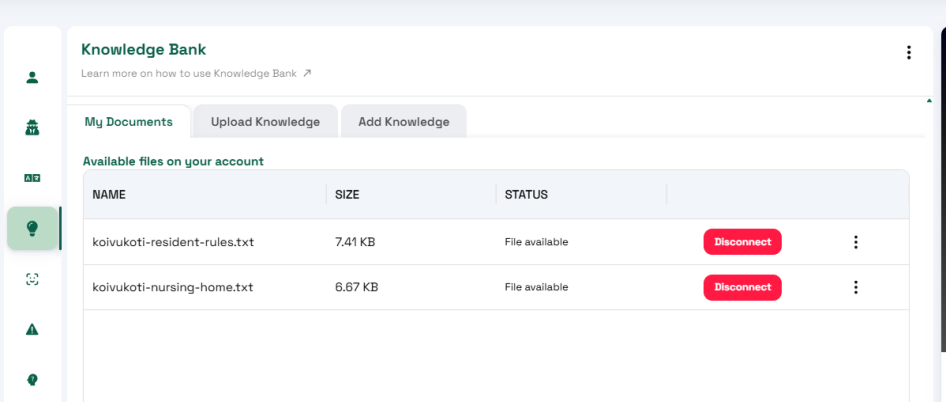
Figure 13: State of mind during a conversation

Various scenarios were tested using this tool, including instances where a patient became agitated or confused. This feedback loop was used for ensuring that Sofia demonstrated emotional intelligence and was able to respond to distress without mirroring the user's panic or confusion. This was done to prevent an unwanted emotional escalation within the virtual experiences.

5.6 Knowledge Integration

Following the personality modelling, the next step was the integration of domain-specific knowledge through Convai's Knowledge Bank system. Knowledge bank serves as a repository that contains all knowledge and memories of the character, including their general knowledge about the world. For Sofia, the knowledge dataset was manually curated and structured as a plain .txt file. While other formats are also supported, the .txt format is recommended for optimal parsing and indexing. The information stored in the knowledge bank is accessed at runtime using RAG. Figure 14 illustrates the knowledge bank setup interface for Sofia.

Figure 14: Final Knowledge bank setup



The effectiveness of the knowledge bank, particularly the accuracy of RAG retrieval, depends on the quality and structure of the provided data. Several optimization techniques were employed during the dataset preparation phase. Each paragraph was focused on a single subject and limited to approximately five lines in length. Paragraphs were formatted in a dialogue format to match the structure of user inputs, with the aim of improving semantic matching during RAG retrieval.

In addition to factual knowledge, entries were tailored to reinforce Sofia's personality and tone. For example, even instructional segments used supportive phrasing to maintain consistency with her high Agreeableness and Meticulousness scores. For added realism, the dataset also included character-specific beliefs and minor misconceptions. An example of this is a line stating, "soft lighting reduces confusion in patients," which reflects common but scientifically debated care strategies.

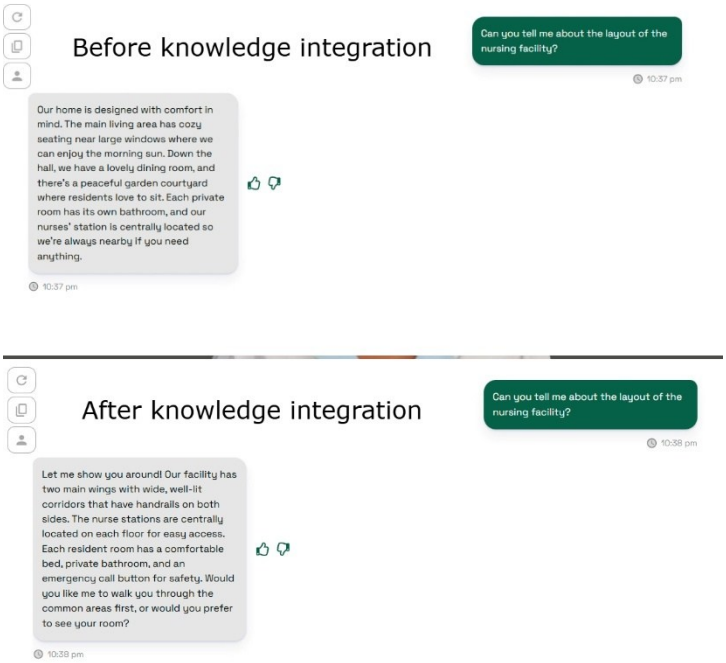
The dataset was generated with the help of Claude Sonnet 3.7. The data covered topics such as facility layout and navigation, daily routines and schedules, common dementia symptoms and appropriate responses, patient-specific notes and knowledge related to the in-game world.

The knowledge bank was then uploaded to Convai servers. The indexing and processing took approximately 3 minutes for a 15 000-word file. Once indexed, the file was linked to Sofia's character profile via the Convai dashboard's "Available Files" section using the "Connect" button. This feature also supports file-sharing across multiple characters, which could be used for implementing shared staff knowledge in the future development versions.

Testing was conducted both prior and after connecting the character's knowledge database. Pre-connection tests established a baseline by asking questions that Sofia, based solely on her default knowledge, should not have been able to answer. One such question was "Who is Mr. Koskinen and what specific treatments does he need?"

Post-connection tests used the same questions to measure improvements in Sofia's responses after the knowledge bank was implemented. Figure 15 provides a comparison of responses before and after knowledge integration.

Figure 15: Comparison of responses before and after knowledge integration



Based on the test results, the knowledge base was refined to address content gaps and enhance phrasing to improve RAG performance. Notably, response accuracy increased when user inputs more closely matched the vocabulary and phrasing used in the knowledge bank, proving the importance of aligning user inputs with the style of the knowledge base.

5.7 Actions

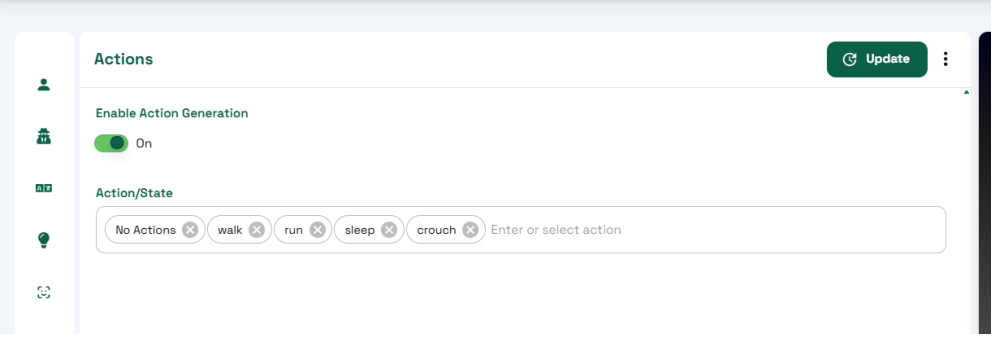
The Actions interface within the Convai platform is used to define character's actions and state transitions that a character can perform in Unreal Engine. These actions enable Sofia

to perform gestures, interact with objects, and navigate the environment based on her environmental awareness. While complex action sequences and state transitions must be configured directly within Unreal Engine, simple actions and states can be defined using the character creation tool.

Before implementing any action logic, it was necessary to determine what interactive capabilities would Sofia require to address the needs of dementia patients. To assess this, Sofia was prompted through conversations without action functionality enabled. This assessment helped to identify situations, such as calming agitated patients and demonstrating how to use equipment, which would benefit from gestures and actions. Based on these findings, a set of actions was created. They were designed to be slow and repetitive to avoid overwhelming the patients and included guiding users through the environment, pointing at objects, picking up potentially hazardous items, and showcasing nursing home equipment.

Then, the Enable Action Generation feature in the Convai dashboard was activated in the Convai dashboard. This setting allows the character to interpret action-related prompts and respond to them with appropriate physical behaviours, such as walking, pointing, nodding, and interacting with specific objects. Figure 16 illustrates the configuration of basic actions.

Figure 16: An example of states that the character can perform based on commands



Each action was subsequently mapped to animations and blueprints within Unreal Engine, using trigger keywords and descriptions for integration. It is not necessary to explicitly define every action within the character description, provided that the LLM can infer the intended behaviour behind user input. For best results, action-related prompts should be kept distinct from dialogue and long texts, or complex descriptions should be avoided.

Once the actions were defined, the Update button was used to refresh the character profile, integrating the new behaviours into Sofia's interaction capabilities. To verify the successful implementation, a series of tests were conducted in Unreal Engine using prompts designed to evoke the configured actions. For example, the prompt "Could you walk me to the bed?" was used to test whether Sofia could guide the user towards the corresponding 3D mesh.

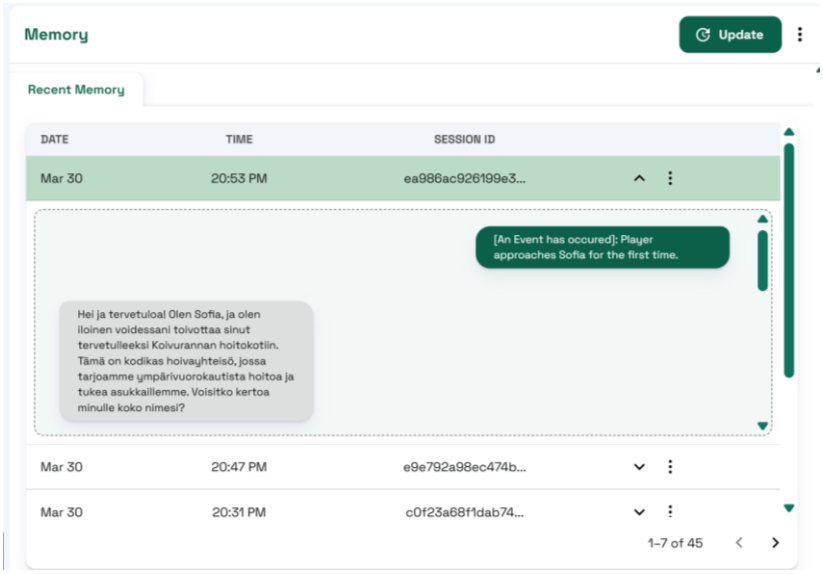
Testing confirmed that Sofia could identify action-related requests and respond with appropriate behaviours. The response latency remained within acceptable limits, showing only a minor increase of approximately 100–150 milliseconds compared to verbal-only interactions.

5.8 Long-Term Memory

The next stage of development focused on implementing long-term memory functionality using the memory component of the Convai framework. This component is responsible for managing conversation history by storing previous conversations and session metadata. Each dialogue session is assigned an identifier, and entries are timestamped and stored chronologically. Memory entries can then be reviewed, prioritized, or deleted using a dedicated interface in the Convai dashboard. Some memories, such as those involving critical medical information or patient details, can be tagged as "critical", ensuring that they are retained and given higher priority when older logs are ranked down. This mimics the way humans selectively retain certain information.

The final implementation retained memory logs for 3 days. After this window, interactions were either archived or deleted based on their assigned priority. Figure 17 shows an example of a recent memory log.

Figure 17: An example of a memory log



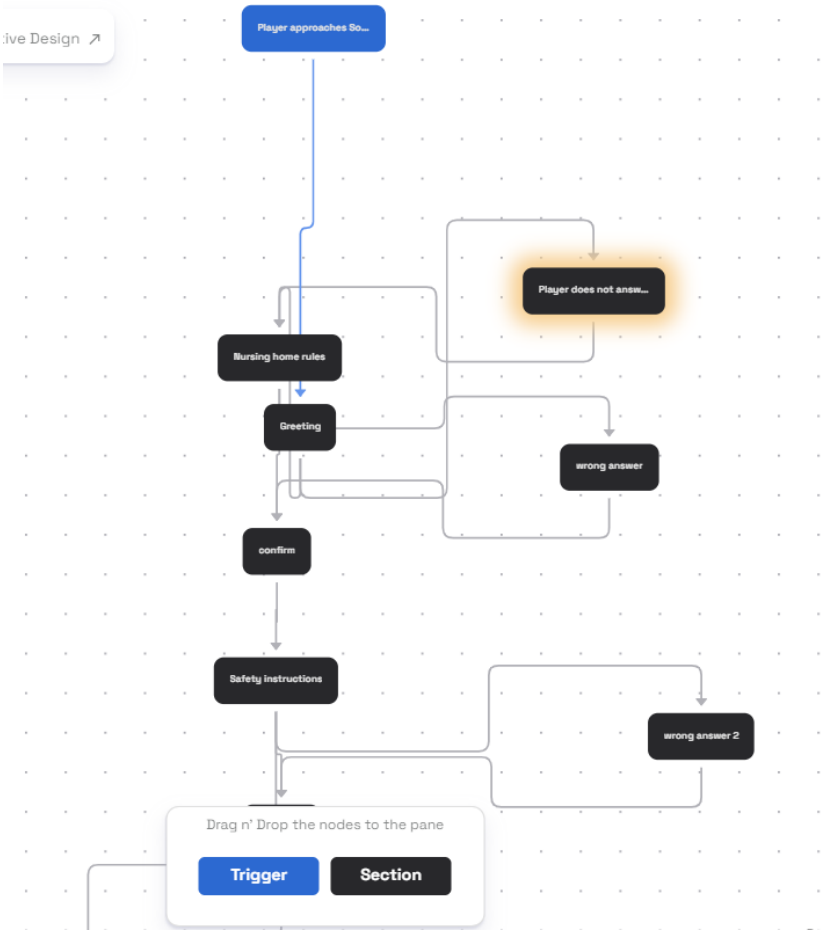
To evaluate the performance of Sofia's memory system, three types of tests were conducted: immediate recall, short-term memory, and long-term memory tests. These tests assessed Sofia's ability to recall contextual information during the same conversation, across recent sessions, and over extended time periods. For example, when asked to remind a patient of their favorite music from a previous day's session, Sofia could recall the genre and song name.

The results confirmed that the system performed reliably in most scenarios. However, memory retrieval accuracy declined in conversations that involved highly emotional content. This suggests an area for future refinement, particularly in optimizing the prioritization and encoding of emotionally charged dialogue for better retrieval.

5.9 Narrative Design

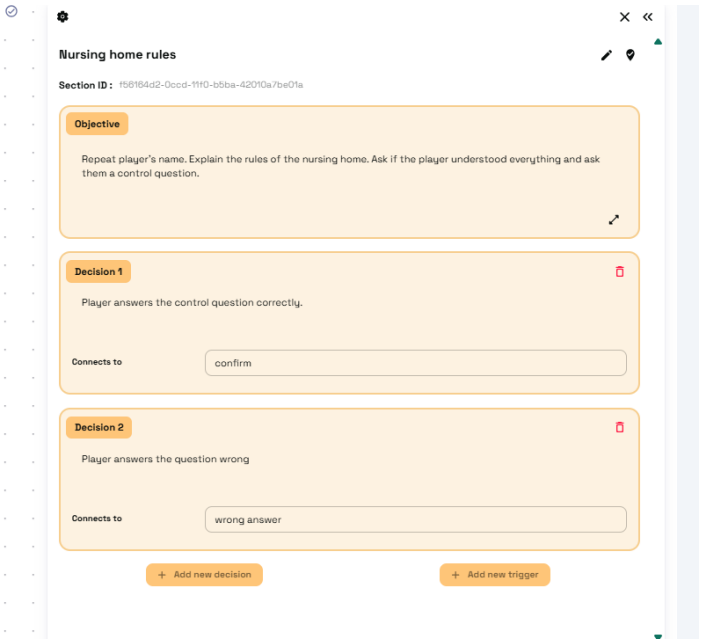
The final step of Sofia's character creation was the implementation of narrative design, which guided her interactions in structured scenarios. A custom narrative tree was designed and implemented within the Convai platform, consisting of 17 sections and 30 decisions. This enabled Sofia to guide the patient through their room while responding to their questions and emotional cues. The entry point of the finalized narrative design is depicted in the Figure 18.

Figure 18: Finalized Narrative Design



At runtime, the active narrative section is updated based on decision nodes triggered by player input. These decision points evaluate verbal content, user sentiment, spatial location, and action requests. For example, if a user expressed confusion, such as “I don’t know where I am”, Sofia would transition to a section where she offers reassurance and repeats orientation instructions in simpler words. Figure 19 illustrates the practical implementation of objectives and decisions.

Figure 19: An example of Objectives and Decisions



For tasks that required precise instructions, such as relaying medication instructions or safety protocols, <speak> tags were utilized override generative outputs and deliver scripted responses. This ensured accuracy and consistency for critical dialogues.

Narrative flow was controlled through a set of pre-defined triggers. A spatial trigger was used to detect the patient entering the room, prompting Sofia to approach the user and offer assistance. If the user did not respond within a designated time frame, a time-based trigger prompted her to repeat the question and inquire about the delay. In cases when the user sounded confused or agitated, an event-based trigger initiated a calming routine.

Each trigger was assigned a unique identifier and integrated into Unreal Engine's Blueprint system using the Invoke Narrative Design Trigger node. To ensure accurate mapping and synchronization, trigger names in Unreal had to match those defined in the Convai dashboard. This allowed interaction between Convai's narrative logic and Unreal's blueprint system.

Furthermore, to enhance adaptability and engagement, Sofia's narrative design incorporated cross-node decision-making, allowing her to respond to a range of patient behaviours. For example, if the user failed to respond correctly to a question, such as identifying the correct pill, during medication guidance, a **Wrong Answer node** was

activated. This both deducted points from user's internal score and caused Sofia to switch to simplified instructions, improving accessibility.

5.10 Unreal Engine

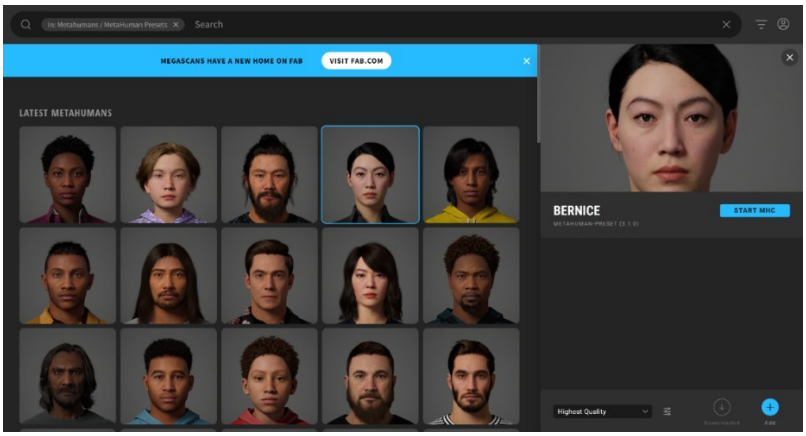
Before Sofia could be integrated into Unreal Engine, the development environment had to be configured to support the Convai plugin. On Windows systems, configuration begins with installing Visual Studio with C++ development tools. Unreal Engine relies on these toolchains to compile components and access platform-specific permissions, including microphone input for ASR.

The Convai plugin can be installed either via the Unreal Engine Fab platform or by building it manually from the GitHub source code. For this implementation, the plugin was installed through Fab, following the official instructions provided in Convai's documentation. Once installed, the plugin was activated through the Plugins menu in Unreal Engine, alongside the Live Link plugin, which manages facial animation for Metahumans. After activating both plugins, the Convai API key, which was previously obtained from the Convai platform, was entered into the Project settings under the Convai section.

A compatible Metahuman model was required for integration with NVIDIA Audio2Face and associated facial rigging tools. While it is possible to use alternative character models, such as those from ReadyPlayerOne or Reallusion, Audio2Face is only optimized for use with Metahuman face bones. This project used a pre-existing Metahuman model, as the Metahuman Creator tool was still in early access at the time of implementation.

The model was added by navigating to Quixel Bridge within the Unreal Engine interface and selecting a Metahuman preset. Out of the available presets, Bernice was chosen as the base for Sofia. Once downloaded, the model was imported into the Content folder of the current project by clicking on the Add button in Quixel Bridge. The metahuman was then renamed to Sofia to avoid any future naming conflicts. Figure 20 shows the process of adding the Metahuman to the project.

Figure 20: Adding a metahuman character into the project



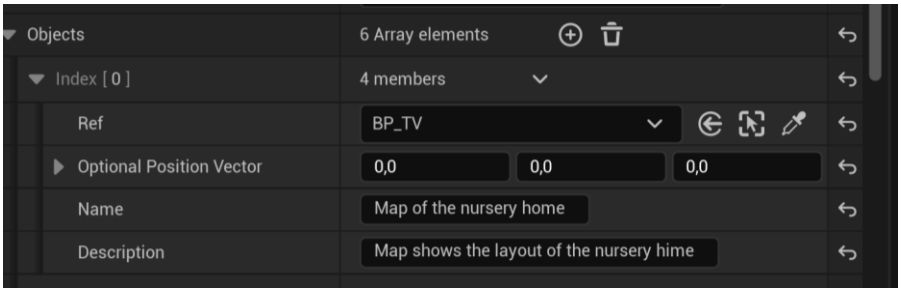
To give Sofia full access to Convai's functionalities, the parent class of her Character Blueprint, located in Content/MetaHumans/Sofia, was changed to ConvaiBasedCharacter. The body component was assigned the Convai_Metahuman_body animation class, which enabled synchronization between dialogue and character body animations. The face component was set to the Convai_Metahuman_Face animation class to integrate facial animations. A new component called Convai Face Sync was added to handle Audio2Face lip synchronization.

With the character blueprint configured to inherit Convai's components and methods, the next step was to prepare the level environment. Sofia's character blueprint was placed into an existing level, which was then customized to resemble a nursing home environment. The 3D assets used in this process included a bed, a TV, coffee corner with a cup, rocking chair and a bathroom door with a button. They were imported into the Content folder and attached to separate Actor blueprints, allowing them to be treated as interactable elements within the scene. The demo design of the room can be seen in the Figure 21.

Figure 21: Design of the patient's room

Sofia’s environmental awareness was then configured by populating the Objects array within the Details panel of her Character Blueprint. Using the eyedropper tool, relevant object blueprints were selected from the viewport and added to this array. Each object was given a descriptive Name and Description, which helped the LLM understand the object’s purpose within the world. This metadata facilitated more intelligent interactions and enhanced the reusability of the objects across different environments. Figure 22 shows how the objects can be configured within the Objects array.

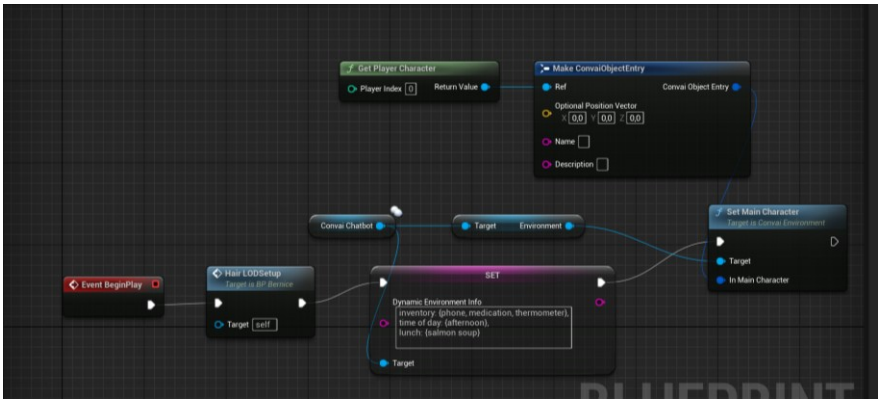
Figure 22: Configuration of Objects array



Once the Level environment was created, the next stage involved adding a Dynamic Environment component to Sofia’s Character Blueprint. This component provides Sofia with real-time context by relaying information about the environment and objects she owns or interacts with. To direct Sofia’s focus toward the player at the start of runtime, the Set Main Character node was appended to the end of the dynamic environment setup logic. This node identifies the player character as the primary interaction target at the start of the

application. Figure 23 illustrates the complete node configuration used in the setup of the dynamic environment system.

Figure 23: Dynamic Environment setup



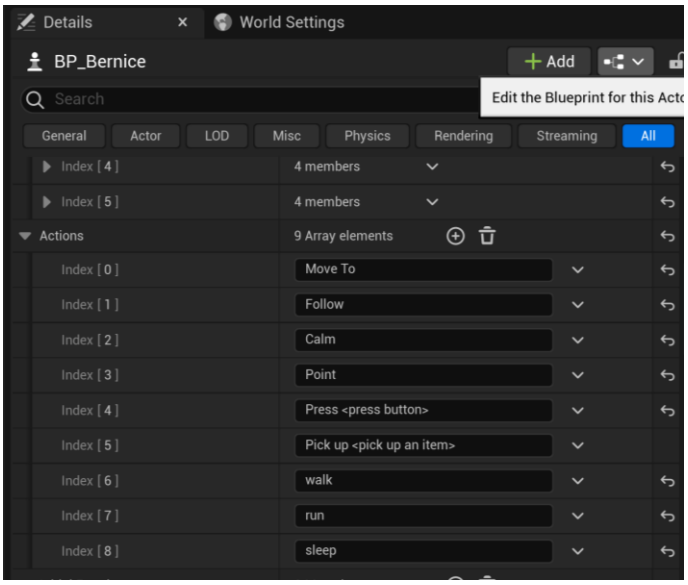
To allow Sofia to navigate and interact with the virtual environment, a Nav Mesh Bounds Volume was added to the level. This volume defines the area in which Unreal Engine's pathfinding system automatically generates a Navigation Mesh. The boundaries of the volume were manually extended to encompass all areas where Sofia should be able to walk. This setup is needed to utilize AI movement logic using nodes like AI Move To.

To make Sofia’s movement speed feel more natural, her walking speed was adjusted using the Floating Pawn Movement component within her Character Blueprint. The Max Speed variable was set to the value of 150, providing a slower, more natural movement pace.

To activate the action system functionality, the Enable New Action System setting was activated in the Project Settings. This enables blueprint handling of custom actions triggered by player input or narrative sections. Simple actions that were previously defined inside the playground were added to the Actions array in the Details Panel of Sofia’s Character blueprint.

Several new actions were added to provide better interactivity, including Move To, Follow, Calm, Point, Press and Pick up. Actions that were tied to specific conditions were annotated using angled brackets, such as Pick up <Pick up an item>, to inform the LLM that the action involves object interaction. Figure 24 shows how the actions are defined inside the Details Panel.

Figure 24: Defining actions in the Editor



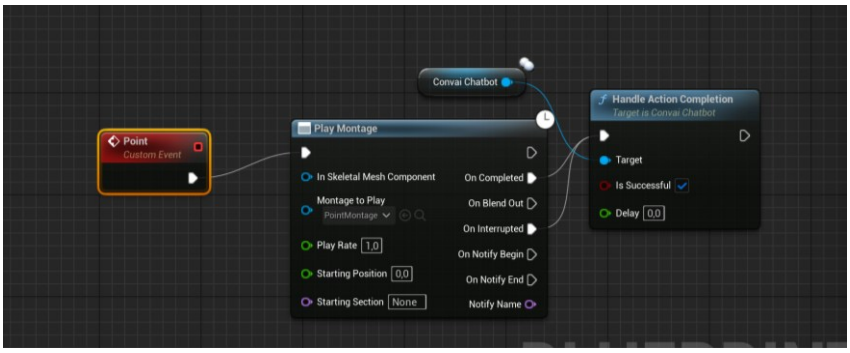
The animations corresponding to these actions were obtained from the Mixamo library and then retargeted to the Metahuman skeleton. Once retargeted, each animation was converted into an Animation Montage to provide control over playback. This enabled animations to be played based on action events. For every action, a corresponding Custom Event was added inside Sofia's Character Blueprint Event Graph. Each event was named to match the action string and consisted of a Play Montage node for animation playback and a Handle Action Completion node to inform the Convai Chatbot component about the successful execution or interruption of the action.

More complex actions, such as Pick Up, required additional setup due to their interaction with world objects. First, a custom bone socket was added to Sofia's Metahuman skeleton to serve as an attachment point for carried items. To prevent the animation to be overridden by default locomotion animations, a new animation slot was assigned to the pickup montage. Then, a Notify event was placed on the montage timeline at the frame where Sofia's hand should come into contact with the object.

The Break ConvaiObjectEntry node was used to extract references to the objects stored in the Objects array. The Set node then assigned the target object reference to the AI Move To function, enabling Sofia to walk toward the object. Once she arrived within range, the Play Montage node initiated the pickup animation. The On Notify Begin event was used to trigger the Attach Actor to Component node, which attached the object to the previously created hand socket. An idle holding animation was set up to play after the pickup montage

was finished. The Handle Action Completion node was triggered to report whether the pickup was completed successfully or interrupted. Figure 25 below shows the blueprint logic behind the Pick up event.

Figure 25: Creating a Pick up Event



To support multiple conversation modes, a custom chat widget component was added for text-based interactions. The component was downloaded from Convai's website and placed inside the ProjectDirectory/Content folder. BP_Convai_3DWidget_Component was added to the player's blueprint and scaled to be appropriately sized relative to Sofia.

Next, the parent class of the player blueprint was changed to ConvaiBasePlayer to enable built-in Convai interaction functionalities. This base class includes voice capture and processing logic, as well as default action logic. Under World settings, the GameMode Override was set to BP_FirstPersonGameMode to ensure that the correct character blueprint was referenced during runtime.

The T key was added to the key bindings inside Project Settings -> Input to trigger voice interaction. In the player blueprint's Event Graph, the Pressed event of the key was connected to the Start function and configured to detect the character the player is speaking to using the Get Looked At Character node. This allows the system to correctly identify which NPC the player is addressing, which is useful in scenes involving multiple Convai characters.

To maintain continuity across sessions, long-term memory was enabled in Sofia's Convai Playground profile. The corresponding Speaker ID was copied into the ConvaiPlayer component in the player blueprint to allow the system to retain conversational context. A SaveGame blueprint named BP_ConvaiSavedGame was created, containing a SessionID variable. The Save Game to Slot and Load Game from Slot nodes were implemented to persist this information between gameplay sessions.

Finally, to trigger the character’s narrative flow, the Invoke Narrative Design Trigger node was used to activate the Narrative Graph defined in the Convai Playground. A Box Collision component was added to Sofia’s blueprint to detect when the player approached her. An On Component Begin Overlap event was set up for the collision volume, configured to trigger only when the player pawn entered the area. This event was linked to a Do Once node and then to the Invoke Speech node, prompting Sofia to initiate dialogue and transition into the narrative when the player first enters the interaction zone.

6 Results

The implementation of the AI dialogue system in Unreal Engine demonstrated a successful proof of concept. While formal user testing with the intended audience, such as dementia care professionals or nursing staff, was beyond the scope of this thesis, internal evaluations confirmed that the system met all core requirements and addressed all research questions. All intended AI components, including ASR, TTS, LLM, dynamic animation, long-term memory, narrative graph and action system, were integrated into Unreal Engine using the Convai framework. This framework simplified the integration process and promoted an iterative development approach.

The system showed the best performance in English language. The ASR component, powered by NVIDIA Riva, achieved ~8% WER in controlled environments. The accuracy decreased to ~15% in noisy environments and up to ~30% for less-optimized languages and non-native accents, highlighting an area for future improvement. These limitations were reflected in user feedback, as shown in Figure 26, where participants of user testing evaluated the ASR performance.

Figure 26: User testing of STT

Test Number	Accuracy of Trar	Handling of Acce	Latency in Respo	Handling of Back	Word Error Rate	Overall Satisfact	Comments
1	95	4	300	5	5	5	N/A
2	92	5	300	4	8	4	Handled a strong accent well
3	95	5	350	5	6	5	Response could be faster
4	87	5	320	3	9	3	Struggled with background noise
5	95	5	280	5	5	5	Response could be faster
6	95	5	300	5	6	5	Handled slang terms well
7	90	4	320	4	5	4	N/A

The LLM integration, primarily utilizing Claude 3.5 Sonnet, consistently produced contextually appropriate responses that aligned with the character's personality. The use of Knowledge Bank improved factual grounding but failed to prevent several instances where the LLM disregarded the retrieved information. Issues were also observed during topic shifts and when handling ambiguous or unexpected input. User testing indicated that consistency declined during longer interactions. Figure 27 summarizes user comments regarding the performance of different LLMs, showing the importance of integrating carefully planned narrative structure.

Figure 27: User testing of LLMs

Test Number	User Scenario	Language Model	Conversation Length (mins)	Rating: Coherence (1-5)	Rating: Responsiveness (1-5)	Topic Drift	Context Retention	User Feedback
1	Narrative-driven	GPT-4.5	15	4	5	No	4	Stayed on topic well but slightly repetitive toward the end.
2	Open-ended	Claude	10	4	3	Yes	2	Became confusing after I asked a random question. Lacked focus.
3	Narrative-driven	Llama	12	5	4	No	5	Very natural and smooth, followed the plot closely.
4	Open-ended	GPT-4.5	18	3	3	Yes	3	Good at the start, but broke character later.
5	Narrative-driven	Claude	14	4	4	No	4	Stayed on topic, but slow responses.
6	Open-ended	Llama	8	2	3	Yes	2	Lost the flow when I interrupted with an unrelated topic.
7	Narrative-driven	GPT-4.5	17	5	5	No	5	The system handled topic shifts perfectly within the narrative.
8	Open-ended	Claude	20	3	4	Yes	3	Initially responsive, but repeated itself in longer interactions.
9	Narrative-driven	Llama	16	4	4	No	4	Mostly on track, a few awkward pauses in response.
10	Open-ended	GPT-4.5	12	2	2	Yes	2	Could not handle topic changes, strayed outside of topic.

The TTS component produced high-quality English speech (MOS ~4.0), with the quality dropping for low-resource languages, such as Finnish. User feedback, as shown in Figure 28, confirmed that the speech was perceived as fluid and easy to understand. However, participants noted reduced emotional expressiveness in low-resource languages and code-switching occasionally triggered language shifts without any prompting.

Figure 28: User testing of TTS

Test Number	MOS (1-5)	Issue Reported	Evaluation
1	5	None	Good
2	4	Clarity issues	Needs Improvement
4	3	Pacing issues, Robotic sounding	Needs Improvement
4	4	Flat sound but good pronounciation	Needs Improvement
5	5	None	Good
6	3	Struggled to pronounce a word	Needs Improvement
7	5	None	Success
8	4	Strange accent	Needs Improvement

Facial animation and lip synchronization, using NVIDIA Audio2Face, succeeded in generating real-time, synchronized animations for the Metahuman character. However, this also caused a noticeable performance cost during runtime testing. User testing indicated that adaptive animation contributed to a more realistic and engaging user experience. Around 20% of participants observed occasional desynchronization between lip movements and speech, especially in high-latency scenarios. These findings are visualized in the Figure 28.

Figure 29: User testing of animations

Test Number	Animation Rating:	Lip-Sync Accuracy (	Desynchronization	Network Latency	User Feedback
1	5	4	No	No	Decent animations, natural facial expressions.
2	4	3	Yes	Yes	Good overall, but lip-sync was slightly off.
3	3	5	No	No	Good synchronization, unrealistic animations.
4	3	2	Yes	Yes	The animations felt slow, especially when they lagged.
5	4	3	No	No	Smooth, but occasional sync issues.
6	3	3	Yes	Yes	Some desynchronization and weird animations
7	5	5	No	No	No issues with animation or lip-sync.
8	2	3	Yes	Yes	Facial expressions were off, lip-sync was delayed.
9	4	5	No	No	No noticeable problems, felt very immersive.
10	3	2	Yes	Yes	Lip movements did not match speech during lag spikes.

The Long-Term Memory system, employing a hierarchical structure with RAG and custom ranking, enabled the character to recall information across sessions, making the system capable of forming longer-term relationships with the users. The Narrative Graph helped to balance narrative progression with dynamic LLM input. The progression was linked to triggers through environmental cues and user inputs.

The action system enabled the character to perform user-requested actions and interact with defined objects. According to user feedback, this functionality enhanced the believability of interactions, despite a moderate latency increase of ~100-150ms. The guardrails proved effective during qualitative testing, successfully filtering inappropriate content and maintaining safety during the conversation.

While this project was focused on the character “Sofia”, the architecture is adaptable to other AI-driven characters and use cases. Deployment in sensitive domains, such as healthcare, would require ethical review and strict user testing. The prototype demonstrated the feasibility of the approach, but it also revealed the need for further refinement as the system complexity increased during the development.

Component-specific improvements are necessary for the system to become production-ready. Narrative flow management should be refined to better handle unexpected dialogue and prevent the character from straying off-topic. The ASR and TTS components require improvements in multilingual support, accuracy, and latency. ASR models must be better at accommodating noisy environments and non-English languages, while TTS systems would benefit from improved pronunciation and prosody.

The animation system could be enhanced by incorporating a wider range of emotional expressions, potentially driven by real-time sentiment analysis of the dialogue. The addition of dynamic full-body animation would further elevate user immersion. The action system could be expanded to support more complex behaviours, advanced environmental interaction logic and awareness, and error handling for failed actions or navigation issues.

Testing was largely conducted ad hoc. Future development should include structured, quantitative testing, evaluating performance of individual components under varied conditions. An end-to-end latency analysis would help identify performance bottlenecks. In the future, latency could be reduced through on-premises AI deployment. This would also provide stronger privacy protection for sensitive information, such as patient data.

7 Summary

This thesis explored the integration of dynamic AI dialogue systems into game engines to create believable, voice-based NPCs. It addressed three research questions: how to maintain narrative coherence in dynamic virtual environments, how to integrate technologies such as LLMs, ASR, and TTS in Unreal Engine, and how to personalize NPC interactions based on user input.

To answer these questions, a practical prototype was built using the Convai framework within Unreal Engine. The implementation showed that hybrid systems, utilizing LLMs guided by narrative graphs and personality parameters, are able to balance generative freedom and narrative structure. Quantitative and qualitative feedback from user testing supported the usability of the prototype, while performance metrics revealed areas for latency optimization and model refinement.

This project has taught me many things, including the importance of iterative design and testing when developing AI applications. It provided insight into the practical challenges and capabilities of current ASR, TTS, and LLM technologies. I learned how to translate theory into technical design, and how to plan narrative structure and system architecture for AI characters. I gained hands-on experience with tools like Convai, Unreal Engine, Claude 3.5, and Azure TTS.

Looking forward, the potential applications of AI-driven characters like "Sofia" extends into fields such as gaming, training simulations, virtual assistants, and teaching tools. The practical usage will depend on advancements in memory architectures, emotional modelling, and local deployment of LLMs for greater privacy and speed. Given the growing interest in AI technologies, the development in this area is likely to continue quite rapidly.

This thesis contributes to the domain of AI-driven storytelling and presents the groundwork for creating scalable, role-based digital characters in interactive environments. Future research and development should focus on addressing the current limitations, particularly enhancing the robustness, multilingual support, narrative depth, and latency.

References

Abdulrahman, A., Hopman, K., & Richards, D. (2024). Do Not Freak Me Out! The Impact of Lip Movement and Appearance on Knowledge Gain and Confidence. *Multimodal Technologies and Interaction*, *8*(3), 22. https://doi.org/10.3390/mti8030022

Abugabah, A., & Sanzogni, L. (2014). Exploring Factors Affecting End-user Performance of Information Systems. *International Journal for Infonomics*, *7*(3/4), 956–973. https://doi.org/10.20533/iji.1742.4712.2014.0113

Ahmed, N., Saha, A. K., Al Noman, Md. A., Jim, J. R., Mridha, M. F., & Kabir, M. M. (2024). Deep learning-based natural language processing in human–agent interaction: Applications, advancements and challenges. *Natural Language Processing Journal*, *9*, 100112. https://doi.org/10.1016/j.nlp.2024.100112

Algherairy, A., & Ahmed, M. (2024). A review of dialogue systems: Current trends and future directions. *Neural Computing and Applications*, *36*(12), 6325–6351. https://doi.org/10.1007/s00521-023-09322-1

Altun, Y., Tsochantaridis, I., & Hofmann, T. (n.d.). Hidden Markov support vector machines. *Proceedings of the Twentieth International Conference on Machine Learning Volume*, *Proceedings of the Twentieth International Conference on Machine Learning*. https://cdn.aaai.org/ICML/2003/ICML03-004.pdf

Amalas, A., Ghogho, M., Chetouani, M., & Thami, R. O. H. (2024). *A multilingual training strategy for low resource Text to Speech* (No. arXiv:2409.01217). arXiv. https://doi.org/10.48550/arXiv.2409.01217

Anthropic. (2024). *Claude 3.5 Sonnet*. https://www.anthropic.com/news/claude-3-5-sonnet

Arora, S., Batra, K., & Singh, S. (2013). *Dialogue System: A Brief Review* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1306.4134

Asif, M., & Gouqing, Z. (2024). Innovative application of artificial intelligence in a multi-dimensional communication research analysis: A critical review. *Discover Artificial Intelligence*, *4*(1), 37. https://doi.org/10.1007/s44163-024-00134-3

Atkins, C., Wood, I., Kaafar, M. A., Asghar, H., Basta, N., & Kepkowski, M. (2024). ConvoCache: Smart Re-Use of Chatbot Responses. *Interspeech 2024*, 2950–2954. https://doi.org/10.21437/Interspeech.2024-2402

Baevski, A., Zhou, H., Mohamed, A., & Auli, M. (2020). *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations* (No. arXiv:2006.11477). arXiv. https://doi.org/10.48550/arXiv.2006.11477

Bai, Z., Chen, P., Peng, X., Liu, L., Chen, H., Shou, M. Z., & Tian, F. (2024). *Bring Your Own Character: A Holistic Solution for Automatic Facial Animation Generation of Customized Characters* (No. arXiv:2402.13724). arXiv. https://doi.org/10.48550/arXiv.2402.13724

Bibauw, S., François, T., & Desmet, P. (2022). Dialogue Systems for Language Learning. In N. Ziegler & M. González-Lloret, *The Routledge Handbook of Second Language Acquisition and Technology* (1st ed., pp. 121–135). Routledge. https://doi.org/10.4324/9781351117586-12

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). *Rasa: Open Source Language Understanding and Dialogue Management* (No. arXiv:1712.05181). arXiv. https://doi.org/10.48550/arXiv.1712.05181

Boden, M. (2006). *Mind as machine: A history of cognitive science*. https://archive.org/details/mindasmachinehis0002bode

Brabra, H., Baez, M., Benatallah, B., Gaaloul, W., Bouguelia, S., & Zamanirad, S. (2022). Dialogue Management in Conversational Systems: A Review of Approaches, Challenges, and Opportunities. *IEEE Transactions on Cognitive and Developmental Systems*, *14*(3), 783–798. https://doi.org/10.1109/TCDS.2021.3086565

Bruch, S., Gai, S., & Ingber, A. (2024). An Analysis of Fusion Functions for Hybrid Retrieval. *ACM Transactions on Information Systems*, *42*(1), 1–35. https://doi.org/10.1145/3596512

Burgan, D. (2017). *Dialogue Systems & Dialogue Management*. https://www.dst.defence.gov.au/sites/default/files/publications/documents/DST-Group-TR-3331.pdf

Cao, C., Wu, H., Weng, Y., Shao, T., & Zhou, K. (2016). Real-time facial animation with image-based dynamic avatars. *ACM Transactions on Graphics*, *35*(4), 1–12. https://doi.org/10.1145/2897824.2925873

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering*, *29*(3), 509–553. https://doi.org/10.1017/S1351324922000213

Chandrakala, S., Deepak, K., & Revathy, G. (2023). Anomaly detection in surveillance videos: A thematic taxonomy of deep models, review and performance analysis. *Artificial Intelligence Review*, *56*(4), 3319–3368. https://doi.org/10.1007/s10462-022-10258-6

Chauhan, P. M., & Desai, N. P. (2014). Mel Frequency Cepstral Coefficients (MFCC) based speaker identification in noisy environment using wiener filter. *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 1–5. https://doi.org/10.1109/ICGCCEE.2014.6921394

Chen, N., Zheng, Z., Wu, N., Gong, M., Zhang, D., & Li, J. (2024). *Breaking Language Barriers in Multilingual Mathematical Reasoning: Insights and Observations* (No. arXiv:2310.20246). arXiv. https://doi.org/10.48550/arXiv.2310.20246

Chen, Y.-N., Celikyilmaz, A., & Hakkani-Tür, D. (2017). Deep Learning for Dialogue Systems. *Proceedings of ACL 2017, Tutorial Abstracts*, 8–14. https://doi.org/10.18653/v1/P17-5004

Choi, Y., Jung, Y., Suh, Y., & Kim, H. (2022). Learning to Maximize Speech Quality Directly Using MOS Prediction for Neural Text-to-Speech. *IEEE Access*, *10*, 52621–52629. https://doi.org/10.1109/ACCESS.2022.3175810

Colledanchise, M., Cicala, G., Domenichelli, D. E., Natale, L., & Tacchella, A. (2021). Formalizing the Execution Context of Behavior Trees for Runtime Verification of Deliberative Policies. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9841–9848. https://doi.org/10.1109/IROS51168.2021.9636129

Colledanchise, M., & Ögren, P. (2018). *Behavior Trees in Robotics and AI: An Introduction*. https://doi.org/10.1201/9780429489105

Convai Team. (n.d.). *How to Enable Multilingual AI Support for Virtual Characters*. https://convai.com/blog/multi-lingual-ai-characters-convai-gaming-innovation

Convai Team. (2024a). *AI Job Simulator: Build Immersive Virtual Job Experiences with Convai*. https://convai.com/blog/ai-job-simulator-convai

Convai Team. (2024b). *Bringing NPCs to Life: AI-Driven Narrative Design in Unreal Engine and Unity*. https://convai.com/blog/ai-narrative-design-unreal-engine-and-unity-convai-guide

Convai Team. (2024c). *Building a Knowledge Bank for AI Characters with Convai | Step-By-Step Guide*. https://convai.com/blog/building-ai-characters-knowledge-bank-with-convai

Convai Team. (2024d). *Create AI Characters Using 3D Avatars for VR Games and Simulations*. https://convai.com/blog/create-ai-characters-vr-games-simulations

Convai Team. (2024f). *Custom Actions for AI Metahuman Avatars in Unreal Engine*. https://convai.com/blog/custom-actions-ai-metahuman-avatars-unreal-engine

Convai Team. (2024g). *Experience Safety Overview: Understanding Guardrails for Convai-Powered AI Characters*. https://convai.com/blog/safety-guardrails-with-convai-ai-characters

Convai Team. (2024h). *How AI Companions are Revolutionizing the Gaming Industry—And More*. https://convai.com/blog/ai-companions-revolutionizing-the-world

Convai Team. (2024i). *Https://convai.com/blog/realistic-face-animations-ai-characters-convai*. https://convai.com/blog/realistic-face-animations-ai-characters-convai

Convai Team. (2024j). *Integrate AI Characters with MetaHuman Avatars in Unreal Engine 5—Convai*. https://convai.com/blog/create-ai-characters-with-metahumans-unreal-engine

Convai Team. (2024k). *Integrating Dynamic NPC Actions for Game Development with Convai*. https://convai.com/blog/integrating-dynamic-npc-actions-for-game-development-with-convai

Convai Team. (2024l). *Lip-Syncing Virtual AI Characters: Techniques, Integration, and Future Trends*. https://convai.com/blog/lip-syncing-virtual-ai-characters-techniques-integration-and-future-trends

Convai Team. (2024m). *Lip-Syncing Virtual AI Characters: Techniques, Integration, and Future Trends*. https://convai.com/blog/lip-syncing-virtual-ai-characters-techniques-integration-and-future-trends

Convai Team. (2024n). *Long Term Memory*. https://convai.com/blog/long-term-memeory

Convai Team. (2024o). *Long Term Memory—A Technical Overview*. https://convai.com/blog/long-term-memory---a-technical-overview

Convai Team. (2024p). *Testing Framework*. https://convai.com/blog/testing-framework

Convai Team. (2024q). *The Future of Gaming: How AI Characters Enhance Player Experience*. https://convai.com/blog/future-of-ai-characters-in-gaming

Convai Team. (2024r). *The Future of NPC Interaction with Convai's Narrative Design*. https://convai.com/blog/convai-narrative-design

Convai Team. (2024s). *The Role of AI in Creating Nonlinear Games Driven by Narratives*. https://convai.com/blog/ai-in-non-linear-game-narratives

Convai Team. (2025). *How to Use Custom LLMs to Build Lifelike Virtual 3D Characters*. https://convai.com/blog/create-virtual-ai-characters-with-custom-llms

DelSole, T. (2000). A Fundamental Limitation of Markov Models. *Journal of the Atmospheric Sciences*, *57*(13), 2158–2168. https://doi.org/10.1175/1520-0469(2000)057<2158:AFLOMM>2.0.CO;2

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.1810.04805

Ding, T., Chen, T., Zhu, H., Jiang, J., Zhong, Y., Zhou, J., Wang, G., Zhu, Z., Zharkov, I., & Liang, L. (2024). *The Efficiency Spectrum of Large Language Models: An Algorithmic Survey* (No. arXiv:2312.00678). arXiv. https://doi.org/10.48550/arXiv.2312.00678

Eom, S., Yoon, E., Yoon, H. S., Kim, C., Hasegawa-Johnson, M., & Yoo, C. D. (2024). *AdaMER-CTC: Connectionist Temporal Classification with Adaptive Maximum Entropy Regularization for Automatic Speech Recognition* (No. arXiv:2403.11578). arXiv. https://doi.org/10.48550/arXiv.2403.11578

Farea, A., Yang, Z., Duong, K., Perera, N., & Emmert-Streib, F. (2022). *Evaluation of Question Answering Systems: Complexity of judging a natural language* (No. arXiv:2209.12617). arXiv. https://doi.org/10.48550/arXiv.2209.12617

Fawi, M. (2024). *CURLoRA: Stable LLM Continual Fine-Tuning and Catastrophic Forgetting Mitigation*. https://doi.org/10.5281/zenodo.12730055

Ferrag, M. A., Tihanyi, N., & Debbah, M. (2025). *From LLM Reasoning to Autonomous AI Agents: A Comprehensive Review* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2504.19678

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., & Ahmed, N. K. (2024). *Bias and Fairness in Large Language Models: A Survey* (No. arXiv:2309.00770). arXiv. https://doi.org/10.48550/arXiv.2309.00770

Gatt, A., & Krahmer, E. (2018). *Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation* (No. arXiv:1703.09902). arXiv. https://doi.org/10.48550/arXiv.1703.09902

He, Z., Cao, Y., Qin, Z., Prakriya, N., Sun, Y., & Cong, J. (2025). *HMT: Hierarchical Memory Transformer for Efficient Long Context Language Processing* (No. arXiv:2405.06067). arXiv. https://doi.org/10.48550/arXiv.2405.06067

Henderson, J., Lemon, O., & Georgila, K. (2008). Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets. *Computational Linguistics*, *34*(4), 487–511. https://doi.org/10.1162/coli.2008.07-028-R2-05-82

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, *29*(6), 82–97. https://doi.org/10.1109/MSP.2012.2205597

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), 107–116. https://doi.org/10.1142/S0218488598000094

Johnson, Lea, & Kassis. (2018). *Tech Note: Enhancing Oculus Lipsync with Deep Learning*. https://developers.meta.com/horizon/blog/tech-note-enhancing-oculus-lipsync-with-deep-learning/

Juang, B. H., & Rabiner, L. (2005). *Automatic Speech Recognition—A Brief History of the Technology Development*. https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

Jurafsky, D., & Martin, J. H. (2024). *Speech and Language Processing* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/

Kann, K., Ebrahimi, A., Koh, J., Dudy, S., & Roncone, A. (2022). Open-domain Dialogue Generation: What We Can Do, Cannot Do, And Should Do Next. *Proceedings of the 4th Workshop on NLP for Conversational AI*, 148–165. https://doi.org/10.18653/v1/2022.nlp4convai-1.13

Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X., Watanabe, S., Yoshimura, T., & Zhang, W. (2019). A Comparative Study on Transformer vs RNN in Speech Applications. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 449–456. https://doi.org/10.1109/ASRU46091.2019.9003750

Kath, H., Lüers, B., Gouvêa, T. S., & Sonntag, D. (2023). Lost in Dialogue: A Review and Categorisation of Current Dialogue System Approaches and Technical Solutions. In D. Seipel & A. Steen (Eds.), *KI 2023: Advances in Artificial Intelligence* (Vol. 14236, pp. 98–113). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-42608-7_9

Kaur, J., Singh, A., & Kadyan, V. (2021). Automatic Speech Recognition System for Tonal Languages: State-of-the-Art Survey. *Archives of Computational Methods in Engineering*, *28*(3), 1039–1068. https://doi.org/10.1007/s11831-020-09414-4

Kuhn, K., Kersken, V., Reuter, B., Egger, N., & Zimmermann, G. (2023). Measuring the Accuracy of Automatic Speech Recognition Solutions. *ACM Transactions on Accessible Computing*, *16*(4), 1–23. https://doi.org/10.1145/3636513

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent Convolutional Neural Networks for Text Classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9513

Lapeyrade, S., & Rey, C. (2023). Non-Player Character Decision-Making With Prolog and Ontologies. *2023 IEEE Conference on Games (CoG)*, 1–2. https://doi.org/10.1109/CoG57401.2023.10333221

Lenglet, M. (2024). *Analysis of Latent Representations of Neural Text-To-Speech Models for Expressive Audio-Visual Synthesis* [Université Grenoble Alpes]. https://theses.hal.science/tel-04541736/file/LENGLET_2023_diffusion.pdf

Levin, E., Pieraccini, R., & Eckert, W. (1998). Using Markov decision process for learning dialogue strategies. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, *1*, 201–204. https://doi.org/10.1109/ICASSP.1998.674402

Li, J. (2022). *Recent Advances in End-to-End Automatic Speech Recognition* (No. arXiv:2111.01690). arXiv. https://doi.org/10.48550/arXiv.2111.01690

Li, M., Wang, J., Guo, L., Meng, Q., Li, M., & Hou, J. (2023). A novel fusion denoising algorithm of spectral subtraction and Wiener filtering for MEMS magnetic signal in PDR positioning. *Alexandria Engineering Journal*, *75*, 139–150. https://doi.org/10.1016/j.aej.2023.05.022

Li, N., Liu, S., Liu, Y., Zhao, S., Liu, M., & Zhou, M. (2019). *Neural Speech Synthesis with Transformer Network* (No. arXiv:1809.08895). arXiv. https://doi.org/10.48550/arXiv.1809.08895

Li, Y. (2018). *Deep Reinforcement Learning* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1810.06339

Li, Y., Shen, X., Yao, X., Ding, X., Miao, Y., Krishnan, R., & Padman, R. (2025a). *Beyond Single-Turn: A Survey on Multi-Turn Interactions with Large Language Models* (No. arXiv:2504.04717). arXiv. https://doi.org/10.48550/arXiv.2504.04717

Li, Y., Shen, X., Yao, X., Ding, X., Miao, Y., Krishnan, R., & Padman, R. (2025b). *Beyond Single-Turn: A Survey on Multi-Turn Interactions with Large Language Models* (No. arXiv:2504.04717). arXiv. https://doi.org/10.48550/arXiv.2504.04717

Lin, S., Lin, L., Hou, C., Chen, B., Li, J., & Ni, S. (2023). Empathy-Based communication Framework for Chatbots: A Mental Health Chatbot Application and Evaluation. *International Conference on Human-Agent Interaction*, 264–272. https://doi.org/10.1145/3623809.3623865

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). *Lost in the Middle: How Language Models Use Long Contexts* (No. arXiv:2307.03172). arXiv. https://doi.org/10.48550/arXiv.2307.03172

Liu, T. (2025). *Clip-TTS: Contrastive Text-content and Mel-spectrogram, A High-Quality Text-to-Speech Method based on Contextual Semantic Understanding* (No. arXiv:2502.18889). arXiv. https://doi.org/10.48550/arXiv.2502.18889

Liu, X., Li, R., Huang, M., Liu, Z., Song, Y., Guo, Q., He, S., Wang, Q., Li, L., Liu, Q., Zhou, Y., Huang, X., & Qiu, X. (2025). *Thus Spake Long-Context Large Language Model* (No. arXiv:2502.17129). arXiv. https://doi.org/10.48550/arXiv.2502.17129

Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., Pan, Y., Xu, S., Wu, Z., Liu, Z., Zhang, X., Zhang, S., Hu, X., Zhang, T., Qiang, N., … Ge, B. (2024). *Understanding LLMs: A Comprehensive Overview from Training to Inference* (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2401.02038

Luong, T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. https://doi.org/10.18653/v1/D15-1166

Majumdar, & Koluguri. (2024). *Pushing the Boundaries of Speech Recognition with NVIDIA NeMo Parakeet ASR Models*. https://developer.nvidia.com/blog/pushing-the-boundaries-of-speech-recognition-with-nemo-parakeet-asr-models/

Mariani, M. M., Hashemi, N., & Wirtz, J. (2023). Artificial intelligence empowered conversational agents: A systematic literature review and research agenda. *Journal of Business Research*, *161*, 113838. https://doi.org/10.1016/j.jbusres.2023.113838

McTear, M. (2021). *Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots*. Springer International Publishing. https://doi.org/10.1007/978-3-031-02176-3

Microsoft. (2025). *What is text to speech?* https://learn.microsoft.com/en-us/azure/ai-services/speech-service/text-to-speech

Mienye, I. D., Swart, T. G., & Obaido, G. (2024). Recurrent Neural Networks: A Comprehensive Review of Architectures, Variants, and Applications. *Information*, *15*(9), 517. https://doi.org/10.3390/info15090517

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2025). *Large Language Models: A Survey* (No. arXiv:2402.06196). arXiv. https://doi.org/10.48550/arXiv.2402.06196

Moon, S., Kottur, S., Crook, P., De, A., Poddar, S., Levin, T., Whitney, D., Difranco, D., Beirami, A., Cho, E., Subba, R., & Geramifard, A. (2020). Situated and Interactive Multimodal Conversations. *Proceedings of the 28th International Conference on Computational Linguistics*, 1103–1121. https://doi.org/10.18653/v1/2020.coling-main.96

Nagy, A., Spyridis, Y., & Argyriou, V. (2025). *Cross-Format Retrieval-Augmented Generation in XR with LLMs for Context-Aware Maintenance Assistance* (No. arXiv:2502.15604). arXiv. https://doi.org/10.48550/arXiv.2502.15604

NVIDIA. (2024). *Long Short-Term Memory (LSTM)*. https://developer.nvidia.com/discover/lstm

NVIDIA. (n.d.a). *Automatic Speech Recognition (ASR), or Speech-to-Text*. https://www.nvidia.com/en-us/glossary/speech-to-text/

NVIDIA. (n.d.b). *Essential Guide to Automatic Speech Recognition Technology*. https://developer.nvidia.com/blog/essential-guide-to-automatic-speech-recognition-technology/

NVIDIA. (n.d.c). *What Is Speech AI?* https://www.nvidia.com/en-us/glossary/speech-ai/

OpenAI. (n.d.). *What are tokens and how to count them?* https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

Pantha, N., Ramasubramanian, M., Gurung, I., Maskey, M., & Ramachandran, R. (2024). *Challenges in Guardrailing Large Language Models for Science* (No. arXiv:2411.08181). arXiv. https://doi.org/10.48550/arXiv.2411.08181

Parthasarathy, V. B., Zafar, A., Khan, A., & Shahid, A. (2024). *The Ultimate Guide to Fine-Tuning LLMs from Basics to Breakthroughs: An Exhaustive Review of Technologies, Research, Best Practices, Applied Research Challenges and Opportunities* (No. arXiv:2408.13296). arXiv. https://doi.org/10.48550/arXiv.2408.13296

Pelachaud, C., Badler, N. I., & Steedman, M. (1996). Generating Facial Expressions for Speech. *Cognitive Science*, *20*(1), 1–46. https://doi.org/10.1207/s15516709cog2001_1

Porcu, V. (2024). The Role of Memory in LLMs: Persistent Context for Smarter Conversations. *International Journal of Scientific Research and Management (IJSRM)*, *12*(11), 1673–1691. https://doi.org/10.18535/ijsrm/v12i11.ec04

Prevost, G. (2022). *Real-Time Planning: Reducing Complexity for Scaling Up* [HAL]. https://theses.hal.science/tel-04097351/document

Qin, L., Pan, W., Chen, Q., Liao, L., Yu, Z., Zhang, Y., Che, W., & Li, M. (2023). *End-to-end Task-oriented Dialogue: A Survey of Tasks, Methods, and Future Directions* (No. arXiv:2311.09008). arXiv. https://doi.org/10.48550/arXiv.2311.09008

Rana, M., Hacioglu, K., Gopalan, S., & Boothalingam, M. (2024). *Zero-shot Slot Filling in the Age of LLMs for Dialogue Systems* (No. arXiv:2411.18980). arXiv. https://doi.org/10.48550/arXiv.2411.18980

Rosenbaum, T., Cohen, I., Winebrand, E., & Gabso, O. (2023). Differentiable Mean Opinion Score Regularization for Perceptual Speech Enhancement. *Pattern Recognition Letters*, *166*, 159–163. https://doi.org/10.1016/j.patrec.2023.01.011

Sarmah, B., Hall, B., Rao, R., Patel, S., Pasquali, S., & Mehta, D. (2024). *HybridRAG: Integrating Knowledge Graphs and Vector Retrieval Augmented Generation for Efficient Information Extraction* (No. arXiv:2408.04948). arXiv. https://doi.org/10.48550/arXiv.2408.04948

Schneider, J. (2024). *What comes after transformers? -- A selective survey connecting ideas in deep learning* (No. arXiv:2408.00386). arXiv. https://doi.org/10.48550/arXiv.2408.00386

Schmidt, K. L., & Cohn, J. F. (2001). Dynamics of facial expression: Normative characteristics and individual differences. *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, 547–550. https://doi.org/10.1109/ICME.2001.1237778

Sebe, N. (2009). Multimodal interfaces: Challenges and perspectives. *Journal of Ambient Intelligence and Smart Environments*, *1*(1), 23–30. https://doi.org/10.3233/AIS-2009-0003

Shah, M. A., & Raj, B. (2024). *Revisiting Acoustic Features for Robust ASR* (No. arXiv:2409.16399). arXiv. https://doi.org/10.48550/arXiv.2409.16399

Shi, J., Inaguma, H., Ma, X., Kulikov, I., & Sun, A. (2024). *Multi-resolution HuBERT: Multi-resolution Speech Self-Supervised Learning with Masked Unit Prediction* (No. arXiv:2310.02720). arXiv. https://doi.org/10.48550/arXiv.2310.02720

Shukurov, J. (2024). *Audio to viseme and viseme to lip sync using azure*. https://doi.org/10.13140/RG.2.2.31219.54560

Sitaram, S., Chandu, K. R., Rallabandi, S. K., & Black, A. W. (2020). *A Survey of Code-switched Speech and Language Processing* (No. arXiv:1904.00784). arXiv. https://doi.org/10.48550/arXiv.1904.00784

Sojasingarayar, A. (2020). *Seq2Seq AI Chatbot with Attention Mechanism* (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2006.02767

Su, R., & Juang, B.-H. (2024). *Many Hands Make Light Work: Task-Oriented Dialogue System with Module-Based Mixture-of-Experts* (No. arXiv:2405.09744). arXiv. https://doi.org/10.48550/arXiv.2405.09744

Sutton, R., & Barto, A. (2015). *Reinforcement Learning: An Introduction* (2nd ed.). https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Tahon, M., Qader, R., Lecorvé, G., & Lolive, D. (2016). Improving TTS with Corpus-Specific Pronunciation Adaptation. *Interspeech 2016*, 2831–2835. https://doi.org/10.21437/Interspeech.2016-864

Tan, Z., Yan, J., Hsu, I.-H., Han, R., Wang, Z., Le, L. T., Song, Y., Chen, Y., Palangi, H., Lee, G., Iyer, A., Chen, T., Liu, H., Lee, C.-Y., & Pfister, T. (2025). *In Prospect and Retrospect: Reflective Memory Management for Long-term Personalized Dialogue Agents* (No. arXiv:2503.08026). arXiv. https://doi.org/10.48550/arXiv.2503.08026

Thennal, K. D., James, J., Gopinath, D. P., & K, M. A. (2024). *Advocating Character Error Rate for Multilingual ASR Evaluation* (No. arXiv:2410.07400). arXiv. https://doi.org/10.48550/arXiv.2410.07400

Tian, J., Nurminen, J., & Kiss, I. (2007). Novel eigenpitch-based prosody model for text-to-speech synthesis. *Interspeech 2007*, 1278–1281. https://doi.org/10.21437/Interspeech.2007-229

Topal, M. O., Bas, A., & Heerden, I. van. (2021). *Exploring Transformers in Natural Language Generation: GPT, BERT, and XLNet* (No. arXiv:2102.08036). arXiv. https://doi.org/10.48550/arXiv.2102.08036

Umar, M. A. (2020). *A Study of Software Testing: Categories, Levels, Techniques, and Types*. https://doi.org/10.36227/techrxiv.12578714.v1

Van Otterlo, M., & Wiering, M. (2012). Reinforcement Learning and Markov Decision Processes. In M. Wiering & M. Van Otterlo (Eds.), *Reinforcement Learning* (Vol. 12, pp. 3–42). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-27645-3_1

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need* (No. arXiv:1706.03762). arXiv. https://doi.org/10.48550/arXiv.1706.03762

Voas, J., Mooney, R., & Harwath, D. (2024). *Multimodal Contextualized Semantic Parsing from Speech* (No. arXiv:2406.06438). arXiv. https://doi.org/10.48550/arXiv.2406.06438

Wang, S., Lilienfeld, S. O., & Rochat, P. (2015). The Uncanny Valley: Existence and Explanations. *Review of General Psychology*, *19*(4), 393–407. https://doi.org/10.1037/gpr0000056

Wang, Y., Li, J., Wang, H., Qian, Y., Wang, C., & Wu, Y. (2022). *Wav2vec-Switch: Contrastive Learning from Original-noisy Speech Pairs for Robust Speech Recognition* (No. arXiv:2110.04934). arXiv. https://doi.org/10.48550/arXiv.2110.04934

Williams, J. D., & Young, S. (2007). Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, *21*(2), 393–422. https://doi.org/10.1016/j.csl.2006.06.008

Wojdeł, A., & Rothkrantz, L. J. M. (2005). Parametric Generation of Facial Expressions Based on FACS. *Computer Graphics Forum*, *24*(4), 743–757. https://doi.org/10.1111/j.1467-8659.2005.00899.x

Yang, G., Ma, Z., Gao, Z., Zhang, S., & Chen, X. (2024). *CTC-Assisted LLM-Based Contextual ASR* (No. arXiv:2411.06437). arXiv. https://doi.org/10.48550/arXiv.2411.06437

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2024). A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly. *High-Confidence Computing*, *4*(2), 100211. https://doi.org/10.1016/j.hcc.2024.100211

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., & Yu, K. (2010). The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, *24*(2), 150–174. https://doi.org/10.1016/j.csl.2009.04.001

Zhang, Qiang Ji, Zhiwei Zhu, & Beifang Yi. (2008). Dynamic Facial Expression Analysis and Synthesis With MPEG-4 Facial Animation Parameters. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(10), 1383–1396. https://doi.org/10.1109/TCSVT.2008.928887

Zhang, L., Yu, J., Zhang, S., Li, L., Zhong, Y., Liang, G., Yan, Y., Ma, Q., Weng, F., Pan, F., Li, J., Xu, R., & Lan, Z. (2024). *Unveiling the Impact of Multi-Modal Interactions on User Engagement: A Comprehensive Evaluation in AI-driven Conversations* (No. arXiv:2406.15000). arXiv. https://doi.org/10.48550/arXiv.2406.15000

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., & Cui, B. (2024). *Retrieval-Augmented Generation for AI-Generated Content: A Survey* (No. arXiv:2402.19473). arXiv. https://doi.org/10.48550/arXiv.2402.19473

Zhong, W., Guo, L., Gao, Q., Ye, H., & Wang, Y. (2023). *MemoryBank: Enhancing Large Language Models with Long-Term Memory* (No. arXiv:2305.10250). arXiv. https://doi.org/10.48550/arXiv.2305.10250

Appendix 1: Data management plan

No personal data was collected for this thesis. All assets used in the project were purchased from the Unreal Store using the company account.

The thesis files will be stored for a minimum of one year on the author's personal computer and on an external hard drive. The program code developed for application remains the author's personal property and may serve as the foundation for a future project.

During the development process, the project was stored inside a Perforce repository to ensure proper version control. This repository is private and inaccessible to anyone outside of HAMK Tech. The finished project has been additionally backed up in the following GitHub repository:

https://github.com/jskodova/SofiaConvai

If there is further development in the project, it will not be updated in the mentioned repositories, as it is an internal company project.

This repository is managed in compliance with HAMK's Data Protection Policy.