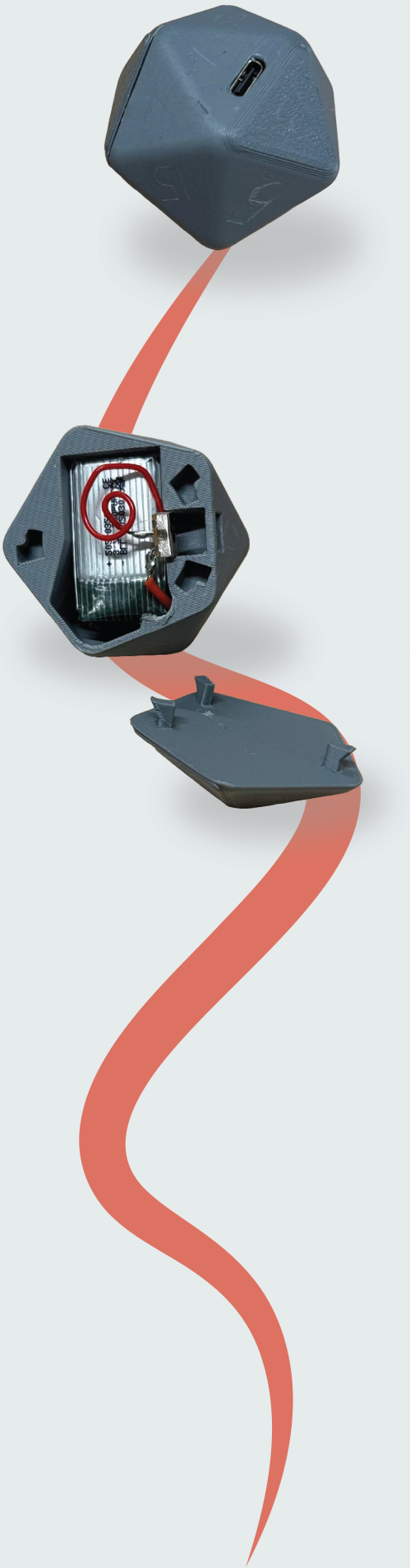

BSc Creative Technology
Graduation Project

AI GM

Creating a Speech to Speech AI to Play Dungeons and Dragons with.

By: Matt Hassing

Supervisor: Marcello. A. Gómez Maureira
Critical Observer: Thérèse S.L. Bergsma

January 2025

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Abstract

Tabletop role-playing games often rely on human Game Masters to manage complex narratives and mechanics, but with artificial intelligence, human game masters could be helped and, in some cases, not even needed anymore.

This project explores how artificial intelligence can address these challenges by automating game master responsibilities through a speech-to-speech conversational system. Leveraging a large language model for narrative generation, combined with speech recognition, text-to-speech, and a custom electronic 20-sided die.

We developed an artificial intelligence game master that is capable of real-time interactions and has integrated dice throws through the electronic 20 sided die. Informed through iterative design and backed by a user experience study with a score of 83 on the system usability scale, this prototype demonstrated strong engagement, particularly for novice players.

This work underscores the potential of artificial intelligence to enhance table top role playing and other board games through adaptive, immersive, and accessible gameplay.

Keywords: Artificial Intelligence, Tabletop RPG, Multi-modal Interaction, Embedded Systems, Speech-to-Speech, Dungeons and Dragons

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisor, Maro, and my critical observer, Thérèse, for their guidance, support, and above all, the enthusiasm they brought to this project. Their insights and encouragement have been instrumental in shaping this work.

I would also like to extend my heartfelt thanks to my parents and brother for their unwavering support, guidance, and the values they have instilled in me.
Thank you, Mom, Dad and Job.

Furthermore, I am sincerely grateful to the following individuals for their support, feedback, and contributions throughout this journey: Thomas Rendon, Nina Kwaks, Daisy Baars, Andres Mirabel, Maja Lamminga, Bart Oude Voshaar, Emma Sanders, Onne Iping, Max Vroomans, Femke Stockmann, and Fee Wollseiffen. Your assistance, encouragement, and belief in my work have been truly invaluable. "It truly takes a village."

Finally, to everyone who has supported me, either directly -through for example play testing- or indirectly, throughout this process, even if you are not mentioned by name. I am grateful to you, thank you. This work would not have been possible without you.

Contents

Chapter 1

Introduction

AI has become an ever-increasing part of people's everyday lives. With the advent of open-AI's ChatGPT and similar services, there's a large uptick in the number of available services using artificial intelligence (AI). This increase in interest and the accessibility of large AI models opens up new possibilities across different domains. One such domain is the gaming and entertainment industry. More specifically, tabletop role-playing games (TTRPGs) such as Dungeons & Dragons might benefit from integrating AI. Usually the game is played with a group of players and a Game Master (GM). It's the Game Master's duty to create and guide the story alongside their players, interpret dice rolls, and know all the rules. With current AI technology, it should be possible to aid human game masters, by allowing the AI to act as a game master, for testing out story ideas or having the AI GM suggest alternative scenarios.

The role of Game Master is multifaceted. Game Masters need to be able to create immersive and vast worlds with plenty of adventures, which means they need to spend a significant amount of time preparing the game world, and they need a vast knowledge of the game rules. The set of requirements are vast, and present for almost any tabletop role-playing game that requires someone to be a game master. AI could be utilized to help human GMs by acting as a GM itself to run scenarios by, to do test runs or to suggest alternative scenarios. Not only that AI could also be utilised for players that cannot find a human game master, making the game accessible to a broader audience.

With recent advancements in AI, it has become feasible to develop AI-based Game Masters. By integrating speech-based interaction, to make communication feel more natural with the AI, and AI-driven storytelling, an AI GM could assist human game masters or even serve as a GM for the people that do not have access to one. However, the effectiveness of such an AI-driven GM remains uncertain, particularly in terms of how players perceive it.

A big element of what makes the TTPRG a game is the fact that you role dice as an element of chance to make decision for your characters, to make the AI GM be able to interpret these dice rolls a 20 sided electronic die is needed, which lets the AI GM know what has been rolled.

Both requirements of the electronic D20 and the AI GM leads to the following question: **"How can speech-to-speech AI be used in conjunction with an electronic 20**

sided dice to serve as a 'Game Master' for tabletop role-playing games?"

To explore this, this research takes a user-centred approach, evaluating how players interact and perceive an AI-based GM. The study focuses on UX and usability using the System Usability Scale (SUS), User Engagement Scale (UES-sf), and qualitative metrics to evaluate the immersion of the system, aesthetic appeal, and overall usability. Through iterative prototyping based on the latest state of accessible AI technologies, this piece of literature describes the development and UX-based evaluation of an AI-based GM in real gameplay scenarios. Thus, this thesis investigates the question:

Ultimately, our findings indicated that, it is feasible to create a functioning and engaging AI GM. The participants found the AI GM highly usable, with SUS scores averaging above 80 and strong engagement ratings from novice and experienced players. This suggests that speech-to-speech AI can effectively handle much of the traditional GM workload, though some limitations remain in handling complex rules due to hallucinations and ensuring consistent narrative depth. These findings demonstrate that the AI-driven Game Master is a promising development for TTRPGs, offering an accessible, immersive experience that could be an GM for people that cannot find a human one, but there is some work still needed to achieve this.

Chapter 2 outlines, the background research on multi-modal AI systems, AI applications in gaming, and the technical foundations that enable an AI-driven Game Master. Chapter 3 outlines our design process, detailing the technology-driven iterative approach used to develop the system. Presented in chapter 4 is, the specification and realization of the AI-driven Game Master prototype. The specification and realisation encompassing both hardware integration and software development for the electronic D20 and the AI GM. In chapter 5, the final realization of the prototype is described. In chapter 6 the AI GMs usability and player engagement is detailed through utilisation of a user experience study. Chapter 7 discusses key findings, limitations, and potential directions for future work. Finally, in chapter 8, a conclusion about the work is detailed.

Keywords: Artificial Intelligence, Game Master, TableTop RPG, procedural storytelling, AI-driven narratives, procedurally generated content, interactive storytelling, gaming.

Chapter 2

Background Research

Following the ideas introduced in chapter 1, such as the AI GM, and the physical D20, some relevant background information has been outlined in chapter 2. Things such as, the basics of multi modal AI, the need for web interfaces to interact with them and the core elements of TTRPGs are detailed within this chapter. This background knowledge is necessary for understanding the design choices and technical challenges discussed in later chapters.

2.1 Multi-Modal Artificial Intelligence Systems

Before multi-modal artificially intelligent systems can be explained, it should be noted what artificial intelligence is.

2.1.1 Artificial Intelligence (AI)

Artificial intelligence, is a facet of computer science that focuses on making machines simulate human intelligence [28]. Usually, simulating human intelligence by machines is done through the usage of large language models (LLMs), which, when given a piece of text, will create the words after that piece of text by evaluating which word is most likely to follow. Finding which word should follow next, is done through the use of tokenisation, which is done by programs called tokenisers [19]. Tokenisers take the given input, such as a piece of text or an image and then translate that given input into tokens. The large language model is given these tokens and predicts the most likely next token. These predictions are then de-tokenised and returned back into human understandable output through the use of detokenisers [19].

A multi-modal AI system is when a suite of different AI systems or tokenisers are combined to make it accept different kinds of inputs or provide different kinds of outputs [5, 21]. For example, combining a model that understands and creates text, together with a model that understands what is in images, and makes a model out of it that understands both images and text. A common example of such a model is OpenAI's chatGPT [9]. Many powerful AI systems exist: image generation models, intelligent text

models that can understand questions and partake in complex reasoning, and models that turn speech into text and vice-versa. So, there are plenty of available models that can be used for speech to text conversion, text to speech conversion and a plethora of other applications, combining such models into a multi-modal conversational model.

2.1.2 Early and Late Fusion

Pereira et al [34], mention two methods to achieve multi modality when talking about AI systems, "early fusion" and "late fusion". In both of these two methods, the term fusion refers to the end result being a combination of different in- or output modalities, with the distinct difference between the two methods being whether the fusion happens early or late in the combination process.

Early fusion is described as combining different features from all the modalities into a single feature vector, in simpler terms, combining the tokenised version of different input modalities into the same input [34, 20]. This combining, results in a unified input that the AI understands and responds to. In contrast, late fusion refers to combining the outputs of multiple separate models operating on their own modality after the models are already done processing [34, 20].

2.1.3 AI Modalities and State of the Art Implementations

Currently, there are many popular AI implementations for different modalities,

One such implementation is for translating speech into text. which is done through the utilisation of automatic speech recognition systems or, as it is also commonly referred to, speech-to-text (STT) models. STT systems translate speech into text, giving computers a way to interpret humans' most common form of communication [8]; STT systems have historically had many of practical applications such as Voice assistants like Amazon Alexa and Google Assistant, Tools for people with disabilities and live or automated transcription services. Currently, these STT systems have found another use-case in functioning as the additional input modality of AI large language models. These examples illustrate that text-to-speech AI can utilised for translating audio into words that computer programs can interpret.

Another implementation that has gained much interest from the general public recently is the large language model (LLM). These models are specialised in processing and understanding natural language in texts. More so, these models can understand, generate and interact with texts in natural human language [50]. These abilities make it so that LLMs lend themselves well to various applications, ranging from conversational chat-bots to understanding and explaining university-level mathematics. LLMs have become a central part of AI, so much so that the acronym AI has become synonymous with mentioning large language models.

Another implementation that has seen considerable advancement similar to the recent LLMs is the Text-To-Speech (TTS) model. These TTS models convert, as their name might suggest, text into speech, which results in a more natural way for computers to

communicate with humans [8]. Many practical applications have already been found for TTS, such as screen readers for visually impaired people and adding human-like speech to virtual assistants [42].

Lastly, an AI modality that covers another big human sense is image generation AI. Models like DALL-E, stable diffusion and mid-journey have re-established what an image generation AI should be capable of. These AI have fundamentally transformed the ability of AI to create images and visuals from text descriptions. These AI make use of deep learning -a way to train AI- architectures known as Generative Adversarial Networks (GANs) and Diffusion Models to create very detailed and accurate images based on a text prompt [44].

These different AI modalities, and many more, represent powerful tools to make computers able to help humans. Such tools cover more and more of the different aspects of human communication for natural language processing. LLMs excel at natural language processing, speech-to-text AI helps computers interpret humans' speech, and text-to-speech allows the system to respond to natural human speech. Image generation, on the other hand can be utilised to help translate textual description to visuals, which leads to the advent of human-like interfacing with computers [8].

2.2 Web technology and webdevelopment

Currently, much of our communication is done through the internet, more specifically through websites. Adding to that, in the field of AI, many commercial products have been made accessible through websites, for example large language models and image generation AI can all be easily accessed through public websites. This abundance of online access is why many resources for creating AI applications pertain to the integration of said applications into websites. Therefore, it is important that some concepts surrounding web design are discussed before delving into the AI application, especially since this application also utilizes a web interface.

2.2.1 Website Development Stacks

Web development stacks are groups of technology that are often visualised as a layered hierarchy, which makes a certain website function. These stacks differ from website to website. Often, a distinction is made somewhere within the stack between two parts of the application, which is between the front-end component and the back-end component [46].

The front-end is the client facing part of the application, to simplify: The front-end is what the client sees, it encompasses everything from the lay-out, to the design, and the interactive elements. All of these elements come from programmed rules defined in the CSS, HTML and JavaScript, to enable developers to create more elaborate applications, certain so called frameworks have been created and are often preferred over writing plain JavaScript CSS and HTML. Popular front-end frameworks that are often used to

create dynamic layouts are:

1. **React**: Developed by Facebook, React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components and effectively manage the application's state [36].

2. **Angular**: Maintained by Google, Angular is a TypeScript-based framework for building web applications. It offers a complete solution for developing single-page applications with a robust set of features [3].

3. **Vue.js**: Vue.js is a progressive JavaScript framework that is approachable, versatile, and performant. It is designed for building user interfaces and single-page applications but has been expanded to be capable of much more [43].

The back-end is the server-side component of web development, which handles everything that is needed to keep the application functional, for example, a webshop needs to handle product inventory and pricing management, payment and more in the back-end [47]. Traditionally, PHP was created to fulfil this role, but currently, many popular back-end languages and frameworks exist, each having their own strengths and specific implementations:

1. **Node.js**: An open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside a web browser. When combined with Express.js, it allows for efficient server-side development [31].

2. **Python**: A programming language made to be readable and simple to understand, Python is used with frameworks like Django and Flask to build secure and scalable web applications [17].

   (a) **Django**: A high-level Python Web framework that encourages rapid development and clean, pragmatic design [14].

   (b) **Flask**: A micro web framework for Python based on Werkzeug and Jinja2. It's lightweight and easy to get started with [45].

3. **Ruby on Rails**: A server-side web application framework written in Ruby. It follows the convention over configuration principle, providing a default structure for databases, web services, and pages [38].

4. **PHP**: A widely-used open-source scripting language that is especially suited for web development. Frameworks like Laravel provide expressive, elegant syntax to ease common tasks [35].

2.3 Tabletop Role-Playing Games

Since the AI GM described in this thesis assumes the role of Game Master in Dungeons & Dragons, some basic understanding of the TTRPG genre can be useful to provide ad-

ditional context on how such a system should function, as such TTRPGs and Dungeons & Dragons

Tabletop role-playing games are a genre of games where players play as fictional characters to create or re-enact fictional stories with the help of a set of rules. Unlike board games, TTRPGs are open-ended stories that develop with the player's input. There are many TTRPGs, but one of the most notable examples would be Dungeons & Dragons (DnD).

In DnD, like most TTRPGs, the game works through a series of interactions between the players and a game master who, in the case of DnD, is sometimes also referred to as the Dungeon Master. Players describe what they want their fictional characters to do. The GM then dictates the result of their actions according to the rules of the game. The results are often decided through a combination of the fictional characters' traits, such as their strength, intelligence, or constitution, and an element of chance. The element of chance is often provided through dice rolling, making it so that even the strongest characters can lose. To make the strongest characters lose, a GM can set a difficulty class, which is a number between 0 and 20, after which the GM asks the player to throw a 20-sided die, and if the player's dice throw is higher than that DC they pass the DC check, the player often also gets bonuses on their rolls depending on the strengths and weaknesses of the character.

2.4 AI in Games [22]

Many different use cases for AI in games have been discussed in the academic literature, not much literature discusses the use of AI in TTRPGS particularly, but the things discussed about AI in games is also relevant to AI in TTRPGs.

2.4.1 What Roles Are AI Currently Fulfilling in Game Development and Gameplay?

According to Filipović [16], Artificial Intelligence (AI) plays an important role in improving the behaviour of non-player characters (NPCs). Filipović mentions that AI is utilized to manage NPC behaviour, which results in engaging NPCs with distinct personalities. Filipović did this by creating decision-making methods that allow NPCs to change their actions based on what is happening in the game and how the user is responding to the AI. Other than that, Karpouzis and Tsatiris [26] also wrote about the use of AI for planning the NPC's behaviour. In their version, content generation algorithms choose NPC actions and dialogues based on digital personalities and the player's behaviour. These advancements make NPC interactions more personalized, improving the overall gaming experience

Additionally, we can state that AI can improve the results of procedural content generation (PCG). Lara-Cabrera et al. [27] discuss in their research how PCG can generate game content such as levels, textures, and characters algorithmically, reducing

the need for someone to design all these assets manually. They also emphasize that integrating AI techniques in these PCG algorithms not only improves the process of creating these assets but also makes it possible to generate a more complex and bigger variety of assets. This potential in reduction in work needed for a result aligns with the findings of Karpouzis and Tsatiris [26]; they state that the resulting assets of these PCG techniques that require minimal input and are widely used in the industry can be adapted by AI based on the player's behaviour and preferences, again making games more personalized. Therefore, AI-enhanced PCG not only simplifies game development but also helps players have a more personalized gaming experience.

In the same way, AI-driven narrative generation can also improve the depth and interactivity of gaming narratives. An example showing this, is the work done by Buongiorno et al. [7], who developed the PANGeA model, which leverages generative AI to create stories that change with the user's input and remember their conversations. the model by Buongiorno et al, uses a personality framework, which is similar to Karpouzis and Tsatiris [26] attempt to give NPC's personality. They use this framework to drive in-game dialogue between players and NPCs. Beyond the creation of personalities for AI, Karpouzis and Tsatiris [26] also discuss the success of narrative generation techniques in role-playing games. They have found that generating adaptive narratives results in longer-lasting gameplay and helps keep players motivated. Interestingly enough, Avin et al. [4] have found a practical use for the narrative capabilities of AI, by utilising it to simulate future scenarios.

Overall, AI stories could make games more engaging by offering players unique and personalized stories.

2.4.2 How Does AI Impact the Player Experience and Engagement in Games?

Artificial Intelligence (AI) enhances player immersion in video games by simulating NPC's and changing narratives. Research by Jiang [25] discusses how the simulation of more realistic emotions in NPCs elicits stronger emotional responses from players, which results in increased player engagement. The simulation of more realistic emotions is also a topic present in research done by Fraser et al. [18]. They researched the management of emotions in dialogue with spoken conversational AI. They showed that processing the user's emotions and having NPCs react to them leads to more realistic interactions with NPCs and increases user engagement. An increase in user engagement by utilising AI based NPC is a topic in research done by Hernandez et al. [23]. They focused on creating AI experience managers in interactive storytelling. Their managers guided players through "emotional trajectories" by adapting the story based on player decisions. These studies demonstrate that emotionally responsive AI systems can significantly increase player immersion and satisfaction.

Artificial Intelligence (AI) plays an important role in improving the behaviour of non-player characters (NPCs). In the research done by Filipović [16], AI is utilized to manage NPC behaviour, which results in engaging NPCs with distinct personalities.

They did this by creating decision-making methods that allow NPCs to change their actions based on what is happening in the game and how the user is responding to the AI. Other than that, Karpouzis and Tsatiris [26] also wrote about the use of AI for planning the NPC's behaviour. In their version, content generation algorithms choose NPC actions and dialogues based on digital personalities and the player's behaviour.

Therefore, AI's role in personalising NPC behaviour could improve game mechanics and, in turn, could improve player engagement.

2.4.3 What Are the Challenges Faced by Implementing AI Systems in Games?

One major challenge faced when implementing AI systems in games is that when implementing AI in PCG, the generation process can become really complex. Karpouzis and Tsatiris [26] highlight that procedural content generation (PCG) algorithms must be very intricate to make sure that generated content is meaningful and not too complex. The PCG must be able to change according to many different player behaviors and preferences, this requirement for robustness leads to an increase in complexity. De Lima et al. [13] further elaborate that narrative generation needs complex natural language processing to create coherent and relevant narratives, and it needs to do this while keeping players engaged, leading to a complex web of requirements. Therefore, the complexity that is part of AI-driven content generation creates significant development and computational challenges.

Another key challenge involves making the NPC act more human-like. Lara-Cabrera et al. [27] discuss how NPCs need to mimic human decision-making and emotional responses to make the games more engaging. They faced difficulties making sure that these NPCs could adapt to different game environments without re-training NPCs for their specific tasks. A potential solution was proposed as part of the PANGeA framework by Buongiorno et al. [7]. They use the Big Five Personality model to give NPCs unique characteristics.

Although the results where impressive it is apparent that it takes much effort to get engaging results. Consequently, creating engaging NPCs that respond naturally is still a big challenge in-game AI development.

Decision-making under uncertainty still is a big barrier to implementing good AI in games. Yin et al. [49] point out that AI in games often have to deal with information that is far from perfect, which leads to AI having to make split second decisions in unpredictable environments. This challenge becomes more apparent in complex games where there is many things happening concurrently, and decisions have to be made efficiently. According to Riedl and Zook [37], developing such AI that can adapt to online games that run continuously with non-stop player interactions and the context of the real world adds such complexity. As a result, helping AI to make informed decisions when there is much uncertainty is still considered a big challenge in game development.

In conclusion, the basics of multi-modal AI, including how different AI modali-

ties such as speech-to-text, text-to-speech, and large language models interact to create multi modal systems have been outlined. Additionally, the chapter explored the importance of web-based user interfaces in facilitating seamless interactions with AI, as well as the core mechanics of tabletop role-playing games such as the role of dice rolls in determining things that happen in the game. Finally, an overview of AI used in games showed how AI has been used to improve player engagement, personalize content, and improve NPC behavior. These topics form the foundation for the following chapters, where the design, implementation, and evaluation of an AI-driven Game Master, in conjunction with an electronic D20, will be discussed.

Chapter 3

The Design Process

The complex nature of creating an AI GM warrants the creation of a novel design process based on the needs for this project. The novel process followed in this project is based on the Creative Technology design process by Mader and Eggink [30], the "Technology driven iterative design" (TDID) process and is depicted in figure 3.1. This variant of the Creative Technology design process has been created to place technology at the centre of the process.

The design process in Figure 3.1 starts out the same as the Creative Technology design process [30], where the first ideation phase would lead to a creative idea. After the initial creative idea is where the TDID process diverges from the Creative Technology design process. In the TDID process, the next step after the creative idea, in this case, Creating an AI GM, would be to create design criteria. Creating design criteria is most easily realized through the utilization of the MoSCoW method, first developed by Dai Clegg in 1994 [11], where all criteria are split into Must, Should, Could and Will not. Based on these criteria, similar real-world implementations that apply to some of the design criteria should be found and discussed. After which, it should be figured out how their implementations work and how a similar implementation could be made, from which a similar implementation should be made with the other design criteria in mind. Then this implementation should be evaluated for its adherence to the design criteria or leading to the reconfiguration of the design criteria, following the loop again of finding implementations that meet your design criteria, copying them and evaluating them until a satisfactory prototype is reached that adheres to all of the criteria sufficiently.

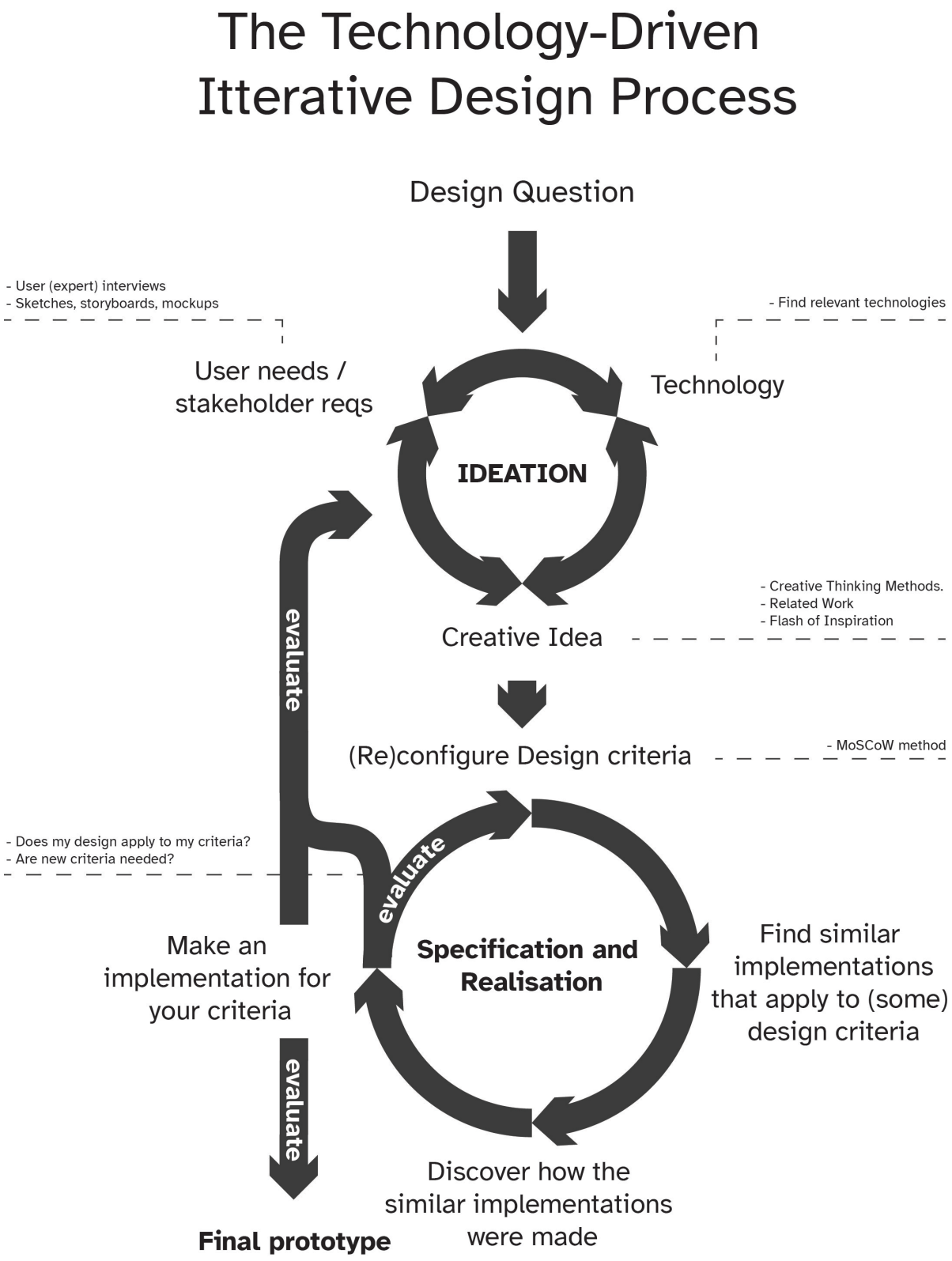

Figure 3.1: Figure depicting the design process used in this project.

In the next chapter, the outlined TDID will be used twice separately, once for the creation and iteration of the AI GM and another time for the creation and iteration of the electronic D20. For both the electronic D20 and the AI GM separate lists of design criteria will be configured based on the MoSCoW method after which similar implementations are detailed, remade and evaluated for their adherence to the configured design criteria. After multiple separate iterations for both the AI GM and the D20 a final prototype will be outlined in chapter 5, which will later be evaluated for its user experience.

In conclusion, The adapted Creative Technology design process, named Technology Driven Iterative Design (TDID) created specifically for this project, because of the complex nature of developing an AI-driven Game Master and an electronic D20. The TDID method has been detailed to provide a structured framework, prioritizing technology as the focus for iterative development. Allowing for iterative improvement, through a set of criteria and evaluations until the final prototype hits the design goals.

Chapter 4

Specification and Realization

The application of the TDID outlined in the previous chapter, as part of the specification and realisation phase, will be applied to configure the appropriate prototype in an attempt to answer the research question: "How can speech-to-speech AI be used in conjunction with an electronic 20 sided dice to serve as a 'Game Master' for tabletop role-playing games?" This chapter contains two distinct prototypes being iterated over, the first distinct prototype is the AI GM itself and the second prototype is the electronic D20. Both prototypes together form the final system that is presented in Chapter 5.

4.1 Ideation

The first phase of the design process is based on [30]. The design question that needs answering is the same as the research question, which is "How can speech-to-speech AI be used in conjunction with an electronic 20 sided dice to serve as a 'Game Master' for tabletop role-playing games?" From this question, several relevant technologies were identified, such as AI systems like ChatGPT, Arduino-based microcontrollers such as ESP32s and inertial measurement units to figure out what number is thrown. The following idea was outlined from the research question and these initial technologies: **Create an interactive AI-based GM that works through speech to speech, together with a physical input modality in the form of an electronic D20.**

4.2 Technology Driven Iterative Design: AI GM Iteration 1

In the TDID process, the ideation phase should result in a creative idea. The identified idea should then be used to configure design criteria, discover similar implementations and how they are made, etc., following the rest of the process illustrated in figure 3.1.

4.2.1 Configure Design Criteria

After the identification of the initial creative idea, the following set of Design criteria were outlined through the MoSCoW method [11], as seen in Table 4.1.

Category	Criterium
Must	The user must be able to talk to the AI GM through voice. The AI GM must be able to respond through speech. The user must be able to roll a dice that the AI GM knows the outcome from. The AI GM must be able to deal with the input of real users. The system must handle interruptions correctly, identify that it is being interrupted, and respond accordingly. The AI GM must have an interface. The user must be able to understand what is happening in the system at all times.
Should	The user should be able to write to the AI GM. The AI GM should be able to respond through text. The AI GM should be able to generate adaptive narratives based on the input from the user. The AI GM should be able to respond in a timely manner.
Could	The user could be able to create their own character, and the AI GM could be aware of this character's properties. The AI GM could be able to simulate NPC responses.
Will Not	The AI GM will not support all different kinds of dice. The AI GM will not support multiple campaigns. The AI GM will not support multiple users in the same campaign at the same time.

Table 4.1: Table containing MoSCoW criteria for the AI GM sorted by criteria

4.2.2 Identifying Similar Implementations

The next step in the TDID is the identification of similar implementations. Based on the outlined criteria, There are no current implementations of an AI GM with all outlined criteria. However, there are implementations that satisfy parts of the criteria.

For the AI, relevant MoSCoW criteria are: "The AI must have an interface." and "The user must be able to understand what is happening in the system at all times." an example of an application that satisfies this criteria is the ChatGPT web-application, by OpenAI [9]. They use a complicated tech stack to allow, according to Sam Altman, CEO of OpenAI 100's of millions of users to connect to their services [1], but in this case, just

the user interfaces were relevant, which is why. Plain HTML, CSS and a simple FLASK back-end were identified to be a good fit for the implementation of AI GM according the MoSCoW criteria.

4.2.3 Discover how the Implementation is Made and Create a Similar Implementation

The next step in the TDID is creating a similar implementation to, in this case, ChatGPT's web application, that incorporates as many of the design criteria as possible. First up, as a phase of familiarisation with the technologies, a dashboard was created through which users could have been able to do various different tasks, such as creating, managing characters, creating and managing the items for said characters and other tasks. This dashboard was later discarded, but it was a useful step to get more familiar with the programming environment and the different technologies utilised. After the initial phase of familiarization, the implementation of the AI according to the MoSCoW criteria and identified relevant third party implementations, this is done by interfacing with the OpenAI API. Through this API it is possible to access sophisticated TTS, STT, and LLM models, the idea is to utilize these sophisticated models to create an AI-chat bot implementation off of which, the AI GM can be built. To this end, a preliminary chatbot was created to start interfacing with the OpenAI API. OpenAI makes it relatively simple to interface with their API. Only an API-key is needed, this is an identification number that lets OpenAI know who to charge with the expenditures related to the usage of their API. The first step is to set up a client through the OpenAI package in Python in the Flask back-end with the aforementioned API key. Next up, we tell the flask application to start listening to the /ai-chat route to serve both users and requests from the website itself. If it is a User that connects to the website, the Flask backend will send the user to the HTML page. The HTML page consists of a user input textbox, a chat history box and a submit button, this is depicted in figure 4.1.

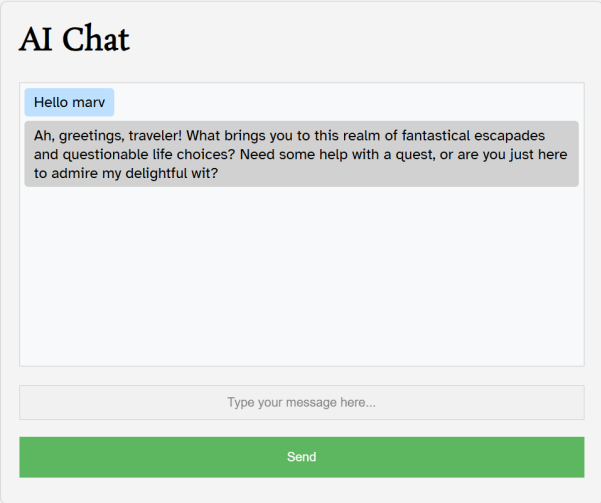

Figure 4.1: displaying an image taken of the AI-chat in the middle of the first iteration of the TDID for the AI GM

As described before, the /ai-chat route also handles requests from the website itself, and if it is a request from the website itself, it is the user using the buttons to post a message, which means that the AI should respond, this is done through OpenAI by calling their client chat completions create endpoint and sending the answer received from OpenAI for that request, back to the website. The responding and requesting of chats is all done through http post requests, the user sends one of these requests, and since post inherently waits for the server to answer, the server answers with the generated response.

4.2.4 Evaluation

Unfortunately, adding the speech-to-text before the LLM and the text-to-speech afterwards would lead to a dramatic uptick in delays, so much so that it would not even be considered a conversation anymore, another problem is the complexity involved with chaining all these different modalities together, dealing with the transmission of audio data both ways while keeping latency into account is a complex problem. This leads to the next step in the TDID, which is evaluation of the current prototype against the set MoSCoW criteria. In the round of evaluation it is imperative to evaluate wether or not the criteria have been met, the following table describes wether or not a criterium has been met, met criteria are marked with an “X”, and irrelevant criteria for this implementation have been marked with a “-”.

Category	**Criterium**	**X**
Must	The user must be able to talk to the AI GM through voice.	
	The AI GM must be able to respond through speech.	

Category	Criterium	X
	The user must be able to roll a dice that the AI GM knows the outcome from.	-
	The AI GM must be able to deal with the input of real users.	X
	The system must handle interruptions correctly, identify that it is being interrupted, and respond accordingly.	
	The AI GM must have an interface.	X
	The user must be able to understand what is happening in the system at all times.	X
Should	The AI GM should support multiple users in the same campaign at the same time.	
	The user should be able to write to the AI GM.	X
	The AI GM should be able to respond through text.	X
	The AI GM should be able to generate adaptive narratives based on the input from the user.	X
	The AI GM should be able to respond in a timely manner.	X
Could	The user could be able to create their own character, and the AI GM could be aware of this character's properties.	X
	The AI GM could be able to simulate NPC responses.	
Will Not	The AI GM will not support all different kinds of dice.	-
	The AI GM will not support multiple campaigns.	-

Table 4.2: Table containing MoSCoW criteria the AI GM with implementation status

Apparent from this is that although the current implementation hits many of the "must", "should", and "could" criteria, almost half of the "must haves" are left unresolved, the current implementation does not respond to speech using speech as shown in figure 4.2, due to the complications mentioned. Seeing as these criteria were considered to be relevant, another round of the TDID was needed.

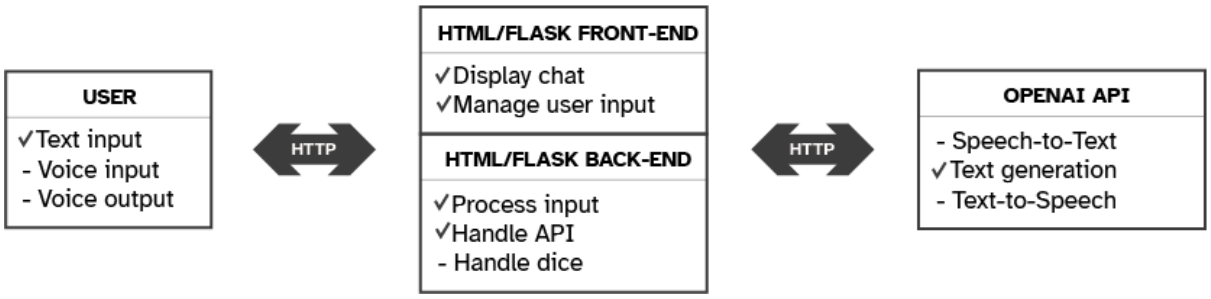

Figure 4.2: Final prototype of this iteration with check marks showing what was implemented.

4.3 Technology Driven Iterative Design: AI GM Iteration 2

In the previous section, the first round of TDID was completed, and another round was deemed necessary in the evaluation step; this means that, in this part, a second round of TDID will be discussed. The reconfiguration of the design criteria was deemed unnecessary, so that step will be skipped, and only the steps of finding similar implementations, discovering how they were made, and evaluation will be discussed. This iteration will be focused on the must haves failed to be covered by the previous iteration.

4.3.1 Identifying Similar Implementations

The second relevant implementation is related to the AI GM, is OpenAI's ChatGPT voice mode [9], in voice mode, "You can now use voice to engage in a back-and-forth conversation with your assistant." [9] This is an application that sends back and forth audio data through the web and gives users the ability to use speech instead of text to interface with the popular LLM named ChatGPT. This is similar to what is part of our criterium for an AI, according to our "must haves" a user should be able to interface with the AI using speech, voice mode is also able to handle interruptions, which is another one of our must have criteria. LLMs inherently are able to deal with the input of real users, crossing off yet another must-have criterium, and finally, this application does it all in a timely manner. OpenAI's voice mode works through the late fusion method described in 2.1.2 Early and Late Fusion. There was speculation about how OpenAI made use of a popular open-source webRTC (web-based Real-Time Communication, read phone calls and video calls over the web.) platform named LiveKit, this was later confirmed when OpenAI launched their real-time API [29], but at the time of designing, this was mere speculation. Based on speculation, LiveKit was identified as the relevant platform for building the AI part.

4.3.2 Discover how the Implementation is Made

LiveKit was originally a webRTC platform, this is apparent in the way things are structured. Inside of LiveKit there is this concept of rooms and people or systems that can subscribe to these rooms. These rooms can have a multitude of different functionalities: video, audio, and text chats. Every person that enters the room has a certain set of rights. This way, some people are allowed to do some things that others can't. One person might have the right to publish a video that others can only watch. Based on this system of rules, automated people can be created, for example, AI chatbots. So, an automated person in the form of an audio and text chatbot acts as the AI GM, and the system gives it access to both the audio and text of the room. Another important part of making the assistant is that it should chain the STT to the LLM to reason and talk back using the STT. Typically, as outlined in the previous section, this would lead to a significant sum of delays; there are, however, ways to minimize the delays. The delay to be minimized is the delay between the time that the user has finished speaking and the time the AI has started talking, which is called "time until the first readout". It is a big challenge to minimize this delay when doing late-fusion, but luckily, LiveKit has a built-in AI assistant class that does all of the before mentioned, including the minimization of the time until the first readout.

4.3.3 Creation of a Similar Implementation

In the implementation, the user creates/joins a room, this is done through a couple of steps, firstly the program is set to fetch an identification token from the back-end dynamically, then it identifies and authenticates itself with LiveKit this allows the user to join a LiveKit room. The room provides the WebRTC infrastructure (the ability to share audio and text in real-time), and notifies LiveKit that an AI is needed inside the room. The AI then joins the room and only responds to the user who joined the room first. AI can be in a couple of different states, including listening, thinking, speaking, connecting, and disconnected. These different states are communicated to the user through the audio visualizer in the user interface. Some are shown in figure 4.3.

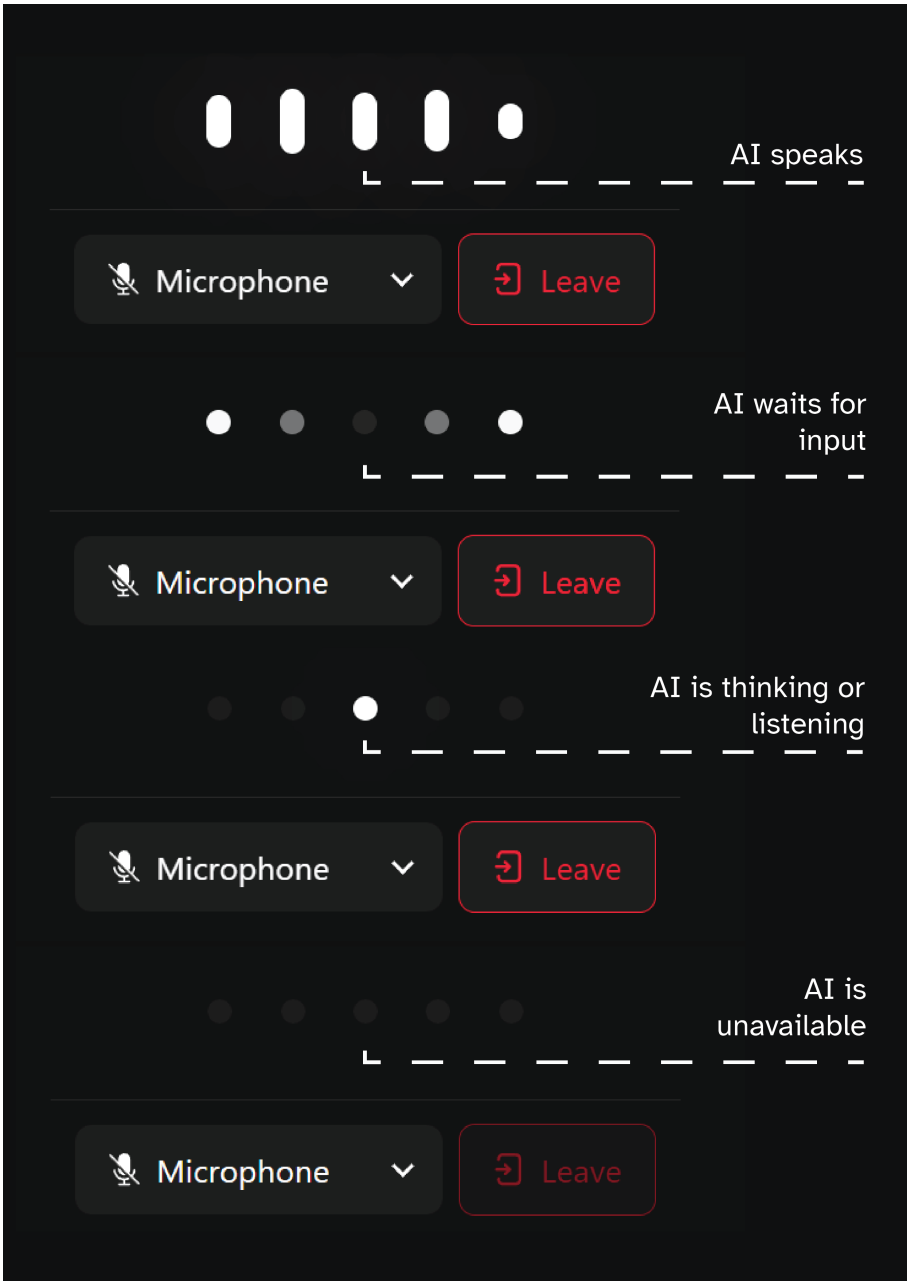

Figure 4.3: Overview of different states of the AI and the way it communicates this to the user through the audio visualizer in the UI.

Above this component, there is a chat shown as illustrated in figure 4.4, inside this chat, both live transcriptions of the user and the AI GM, as well as text messages, are shown, this way, a user is able to detect an error in transcription while talking, allowing them to correct the error as early as possible.

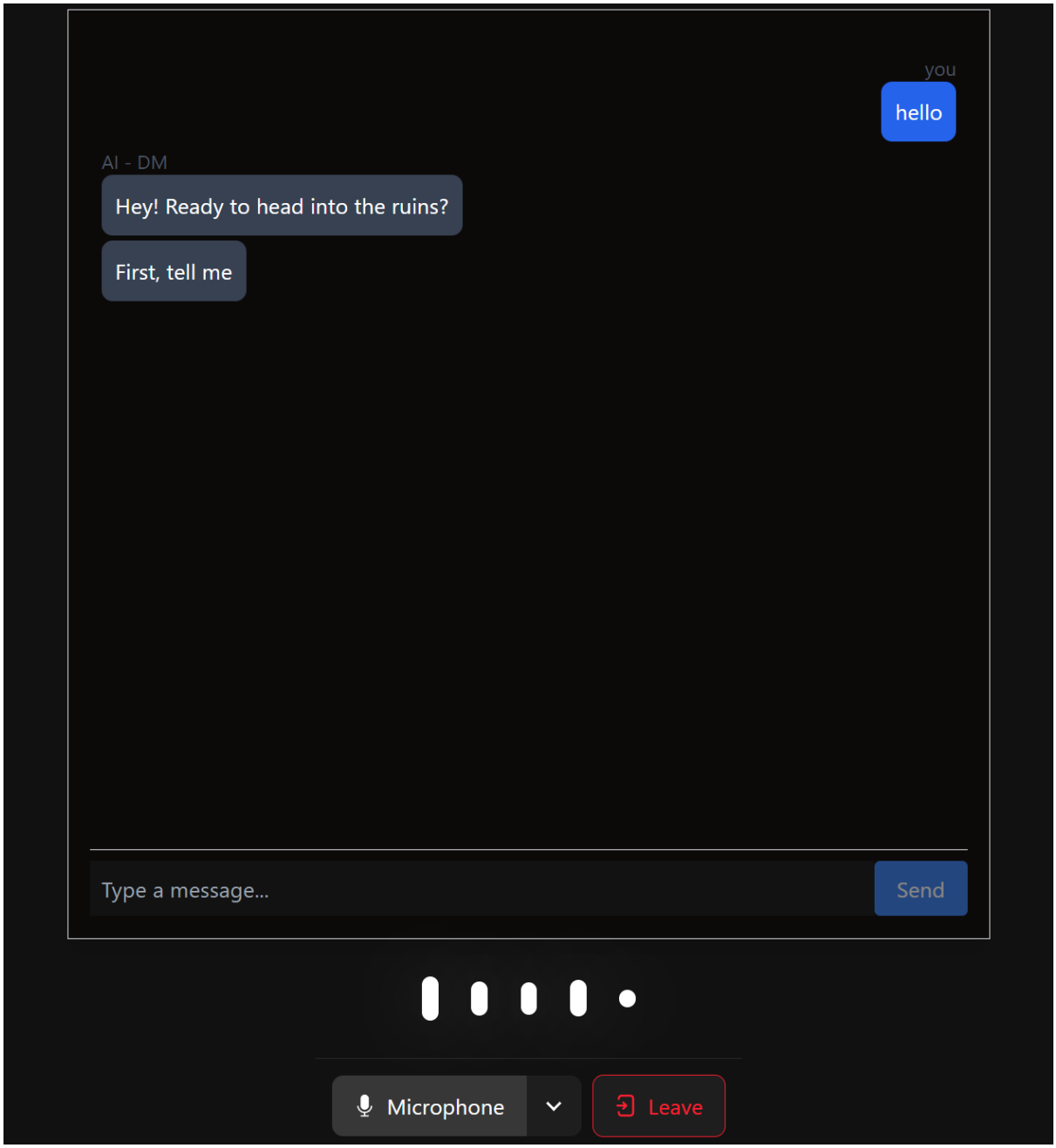

Figure 4.4: Overview of the UI of the AI GM

The AI is implemented through the LiveKit agents “VoicePipelineAgent”, the way in which LiveKit implements late fusion is through the STT, LLM, TTS chain, with a specification to make it also allow inputs directly to the LLM omitting the STT. Another important part of the voice pipeline agent is voice activity detection (VAD). This is a separate AI that detects when a human is speaking by returning a voice activity threshold, which can later be used by the voice pipeline agent to decide wether or not the human is speaking, which allows the AI GM to detect interruptions. For all four of these components, a plethora of modern choices are available, but for the purposes of this project, choices have been made based on latency, quality and cost. For the VAD, an open-source VAD from Silero has been chosen for its high performance and the fact

that it is free. For the speech-to-text engine, Deepgram's most recent STT model has been selected, mainly because it met the criterium to do streaming-based speech-to-text. For the LLM, the flagship LLM model of OpenAI is being used, which is, as of the time of writing, GPT-4o, since the 4o model has extended reasoning capabilities and a significant token window, which allows it to remember things much longer compared to the cheaper GPT-4o-mini. For the final step, which is the conversion from the output of the LLM back to speech, OpenAI's TTS is being used, OpenAI's TTS, however, requires a complete finished sentence before its output can be used, which is why the TTS is wrapped into a stream converter, which allows the program to bypass this restriction by cleverly deciding when (parts of) sentences are done, with the result that the AI can send (parts of) sentences audio earlier. A big advantage of this setup is that we can extend its reasoning using functions through the LLM. Specific sets of predefined functions can be exposed to the LLM, in this case these functions are utilized to detect what a user has thrown with a dice, allow the AI to search through a catalog of predefined monsters, and allow the AI access to the character sheet that the user has predefined.

4.3.4 Evaluation

As outlined in the previous chapter, the missing MoSCoW design criteria, including the lack of talking to and hearing the AI talk, as well as the delay that becomes too big, are now solved in this implementation. In table 4.3 is outlined which criteria are solved in this implementation.

Category	**Criterium**	**X**
Must	The user must be able to talk to the AI GM through voice.	X
	The AI GM must be able to respond through speech.	X
	The user must be able to roll a dice that the AI GM knows the outcome from.	-
	The AI GM must be able to deal with the input of real users.	X
	The system must handle interruptions correctly, identify that it is being interrupted, and respond accordingly.	X
	The AI GM must have an interface.	X
	The user must be able to understand what is happening in the system at all times.	X
Should	The AI GM should support multiple users in the same campaign at the same time.	
	The user should be able to write to the AI GM.	X
	The AI GM should be able to respond through text.	X

Category	Criterium	X
	The AI GM should be able to generate adaptive narratives based on the input from the user.	X
	The AI GM should be able to respond in a timely manner.	X
Could	The user could be able to create their own character, and the AI GM could be aware of this character's properties.	X
	The AI GM could be able to simulate NPC responses.	X
Will Not	The AI GM will not support all different kinds of dice.	-
	The AI GM will not support multiple campaigns.	-

Table 4.3: Table containing MoSCoW criteria the AI GM with final implementation status

Unfortunately, this implementation does not support multiple users at the same time, but since this criteria is a "should" and not a must, it can be concluded that this part of the prototype is successful enough for the final set of evaluations. A system overview is shown in figure 4.5. A set of criteria that is yet to be solved is the must criteria where the user must be able to roll a dice that the AI GM knows the outcome of. This criterium will be solved in the next section through the application of the TDID on the dice part of this project alone.

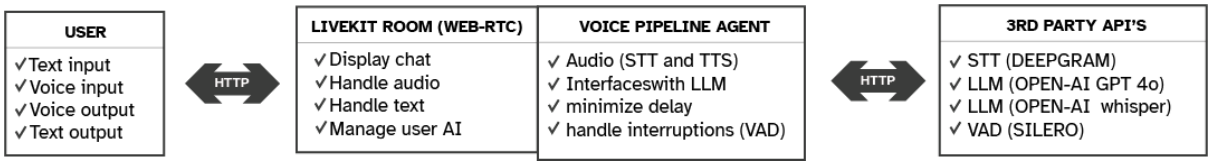

Figure 4.5: Final prototype of this iteration with check marks showing what was implemented.

4.4 Technology Driven Iterative Design: Dice Iteration 1

Currently, the prototype iterations only cover the digital part of the prototype, but the prototype also includes communicating with some sort of twenty-sided die (D20). This section of the report will describe how the D20 was developed, including how it communicates with the AI GM.

4.4.1 Configuring Design Criteria

According to the TDID explained in section 3.1, after outlining the initial idea, the design criteria should be configured, which is done once more through the MoSCoW method. The initial concept, in this case, is to have a dice with embedded electronics to measure which way the dice is facing and have the dice transmit this data wirelessly to the AI. This idea has several challenging constraints, with the two biggest being size and battery power, since the D20 needs to be thrown, it should be wireless, and as such, an embedded battery is required, which increases complexity somewhat since now the dice should be able to be charged, have an embedded rechargeable battery, which is big and conserve energy wherever possible, so the user doesn't have to recharge it often. Another problem that then comes into play is size, batteries can become big and with the criterium of wireless transmission built in battery management system (BMS), the size needed becomes bigger and bigger. Another functional criterium is that the delay between the dice being thrown and the results being visible to the user should be no more than 4 seconds, according to a study done by Peng et al. in 2020 on user perceptions of delays in robots' responses and GUI interactions [33]. To be comfortably under that boundary, even under poor conditions, such as with slow internet, we will be aiming for a delay of less than 3 seconds. From all these conflicting criteria, the MoSCoW set emerges as outlined in table 4.4.

Category	Criterium
Must	The dice must have embedded sensors to detect which face is up. The dice must wirelessly transmit data to the AI GM. The dice must have a rechargeable battery. The dice must use power efficiently to maximize battery life. The dice should have a time between thrown and data received of 3 seconds or less. The dice must be sturdy enough to withstand being thrown many times.
Should	The dice should have an embedded battery management system (BMS) to handle charging and power delivery. The dice should be small enough to be thrown and held comfortably.
Could	The dice could be small enough to resemble the size of a standard D20 as closely as possible. The dice could include customization options for LED colours or patterns for immediate feedback on the state of the AI.

Category	Criterium
Will Not	The dice will not be the exact size of a standard dice. There will not be multiple different types of dice, E.G., a 10-sided die or an 8-sided die. The dice will not be a fair dice The dice will not operate without a battery (i.e., no wired operation).

Table 4.4: Table containing MoSCoW criteria for the electronic dice sorted by criteria

4.4.2 Identifying Similar Implementations

The next step after the configuration of the design criteria is the identification of similar implementations and the technology used in them. For these dice that know what face is thrown, Pixels come to mind [10]. These are open-source dice that use inertial measurement units to detect which face is up, combined with embedded LEDs on each face to make the dice light up based on what was thrown. Some parts of these dice can be utilized for this implementation of an electronic D20 which can identify and communicate which face has been thrown to the AI GM.

4.4.3 Discover how the Implementation is Made and Create a Similar Implementation

According to a video by Jean Simonet posted in 2017 [24]. The early prototype of the Pixels was made using a flexible printed circuit board (PCB) with a multitude of LEDs, an ESP-32-based microcontroller, and an inertial measurement unit (IMU) to measure the way the dice are handled. Now the ESP-32 is a suitable candidate for this use-case, it has builtin Bluetooth, WiFi and a proprietary communications protocol that is more efficient [15], it draws really low currents for the type of performance it has and the tasks it does, it also has a sleep and deep sleep mode which means that when the ESP-32 is not in use, the amount electricity it uses it drastically reduced. All of these advantages lead to the employment of an ESP-32 for this project as well.

For the communication, the Pixels dice and the prototype published in 2017 Jean Simonet [24], Bluetooth is used, which is a great option for their use-case of local applications running on devices near the dice itself, however since this dice needs to communicate with the world wide web, to the AI server living somewhere else in the world, some form of internet connectivity is required. As such, the Bluetooth functionality will be replaced by internet connectivity through the use of WIFI for this prototype.

Another point of interest is the way they handled the IMU, they used an LIS2DE12TR as detailed in [12] by the project linked through the resources in the description of the

video [24]. This is a great choice for an IMU if the goal is to create a circuit board since it is a surface-mounted device (SMD), a type of component made to be on top of printed circuit boards. An alternative with much support for Do-It-Yourself or DIY electronics using the Arduino platform would be the MPU6050, an accelerometer and gyroscope combination. So, for this prototype, the MPU6050 IMU will be used due to its abundant support and high availability. The covering shell of the dice is made using three-dimensional (3D) modelling software such as Blender and Fusion360 and fabricated through the utilization of FDM-style 3D printing.

4.4.4 Creation of a Similar Implementation

To create the electronic die following the implementation outlined above, first a suitable ESP-32 variant should be selected. This should be done through the exploration of development boards, although these will add to the size of the project a bit, it means that the actual process of implementation and iteration is significantly sped up by the development platform boards. The criteria for said development board are:

1. **Built in BMS**, which is needed for reduced sizing, if the BMS is on the development board itself it will be way smaller.
2. **ESP-32C3 or newer**, which is the smallest platform of ESP-32 since it has a built-in programmer in the chip set. Thus, the USB programming logic does not require more circuitry.
3. **Small footprint**, The ESP-32 should be as small as possible due to the size constraint of the D20 detailed in the MoSCoW 4.4.

A development board that fits these criteria is the Seeed Studio XIAO ESP-32C3 development board [48]. It has built-in BMS, is ESP-32C3 based and measures 21mm by 17.5mm, according to Kiwii electronics [41]. The next pick is the relevant MPU6050 development board, a common one is the GY-521, which is the one that was readily available, similarly sized as the ESP-32 development board we picked, so it was selected aswell. Lastly, a battery was selected, a folded lithium-ion, relatively small, no name, no brand, 300Mah battery was chosen as a suitable candidate, this battery was then soldered to the battery pads exposed to the bottom of the ESP-32, it is advisable to put a switch in between the lithium-ion battery and the contact for turning on and off the power to the ESP32-C3 and the MPU was soldered to the top of the ESP-32C3 following the schematic outlined in figure 4.6. It is important to note that the interrupt pin of the MPU6050 is tied to an analogue pin of the ESP32-C3, which is required for the so called deepsleep mechanism described later in this section.

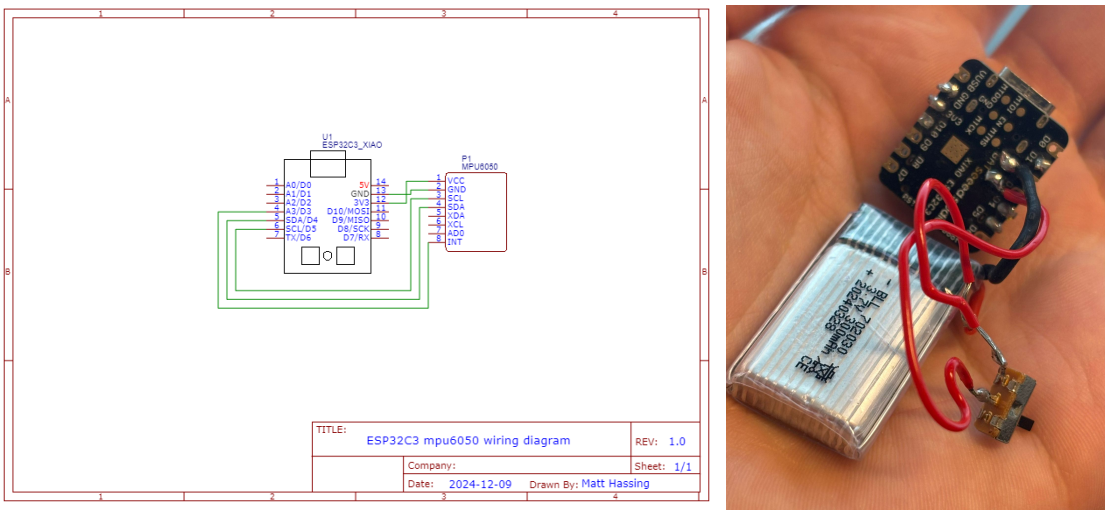

Figure 4.6: ESP32C3 wiring diagram with an MPU6050 left and image of the wired prototype right

After the prototype is assembled, as can be seen in 4.6 , a program can be made to make the prototype detect dice throws and send them over WIFI to the AI, and then go to sleep, the program should wake up again once the dice is moved, and this cycle should be repeated until the battery is empty or the device is turned off. The plan is to have the ESP32 only be on if it has to send its die face, which can be done through interrupts. If the ESP32 is in deepsleep, a certain pin, or sets of pins, can be designated as wake up pins, which means that if the pin is set to high or low, the ESP32 will wake up and perform its duties. We can utilize this by sending interrupts from the MPU6050 to the ESP32 if the MPU detects sufficient movement. Then, the ESP32 keeps track of said interrupts, and if it has not detected any interrupts within a certain time period, the ESP32 should send the current die face to the server.

To determine which face of the die is oriented upwards, we utilize vector geometry by comparing the measured gravitational acceleration with predefined reference vectors for each face.

1. **Measure Acceleration:**
   Obtain the raw acceleration vector from the MPU6050 sensor:

   $$\mathbf{a} = \langle a_x, a_y, a_z \rangle$$

2. **Normalize the Vector:**
   Convert the acceleration vector to a unit vector to represent the direction of gravity:

   $$\hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}, \quad \text{where} \quad \|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

3. **Define Reference Vectors:**
   For each die face i, define a reference unit vector $\hat{\mathbf{f}}_i$ that indicates the expected direction of gravity when that face is downward.

Key Step: Determine Active Face

Determine the upward-facing face:
Compute the dot product between the measured gravity direction and each reference vector:

$$\hat{\mathbf{a}} \cdot \hat{\mathbf{f}}_i = a_x f_{ix} + a_y f_{iy} + a_z f_{iz}$$

Identify the face k with the highest dot product:

$$k = \arg\max_i (\hat{\mathbf{a}} \cdot \hat{\mathbf{f}}_i)$$

The face corresponding to $\hat{\mathbf{f}}_k$ is the one currently facing downward. After identifying the correct face, the ESP32 ensures a wireless connection and sends the value to the AI GM.

The AI GM creates a dedicated HTTP server to receive said sent values from the electronic die after the die has decided the face value rolled through the algorithm. This happens through the following steps:

Step 1: Spin up an HTTP-based webserver at port 8081 as soon as the room is created. This implementation assumes only one room at a time is possible.

Step 2: A dice roll gets transmitted from the electronic die to the AI GM.

Step 3: Process the dice roll:

- **3A:** If the AI GM is actively requesting a roll, process the result and answer the request with the result, which makes the AI GM immediately respond.
- **3B:** If the dice roll is unsolicited, store the result as a hidden system message, which means that if the user prompts the AI GM with something to make it speak again, the AI GM will be aware of what was rolled, since the roll is stored in the system context.

4.4.5 Evaluation

The next step of the TDID is the evaluation phase. In the evaluation phase of the TDID process, the criteria for the electronic dice are evaluated against the outlined MoSCoW. Met criteria are marked with an “X” while irrelevant criteria for this implementation are marked with a “-”. The table 4.5, details the evaluation:

Category	Criterium	X
Must	The dice must have embedded sensors to detect which face is up.	X
	The dice must wirelessly transmit data to the AI GM.	X
	The dice must have a rechargeable battery.	X
	The dice must use power efficiently to maximize battery life.	X
	The dice should have a time between thrown and data received of 3 seconds or less.	
	The dice must be sturdy enough to withstand being thrown many times.	X
Should	The dice should have an embedded battery management system (BMS) to handle charging and power delivery.	X
	The dice should be small enough to be thrown and held comfortably.	X
Could	The dice could be small enough to resemble the size of a standard D20 as closely as possible.	X
	The dice could include customization options for LED colours or patterns for immediate feedback on the state of the AI.	
Will Not	The dice will not be the exact size of a standard dice.	-
	There will not be multiple different types of dice, E.G. a 10-sided die or an 8-sided die.	-
	The dice will not be a fair dice	-
	The dice will not operate without a battery (no wired dice throwing).	-

Table 4.5: Table containing MoSCoW criteria with implementation status for the electronic dice

As shown in table 4.5 all of the “must” and “should haves” are covered except for the delay, the current implementation has to follow many steps before the data is recieved.

Step 1: Wake up from deepsleep (≈200 ms)

Step 2: Connection to WiFi (≈2-5 seconds)

Step 3: Post request to the server (≈0-3 seconds)

From the outlined steps, it becomes apparent that connecting to WiFi when awoken from deepsleep, even with the credentials already stored, can take up much precious time. The most important delay to minimize is the time between when the dice stop rolling and when the rolled face is received. As such, an interesting way to mitigate some of the delays is to start connecting to the WiFi as soon as the die has been moved and awoken from deepsleep, but even with this adjustment in implementation, it is not feasible to hit the three-second mark.

4.5 Technology Driven Iterative Design: Dice Iteration 2

To make the dice follow the three-second delay criterium, another round of TDID is needed, however from the preliminary search in the previous iteration, the fact that the ESP32 has a protocol for sending data quickly to other ESP32's was briefly mentioned, based on this and the fact that connecting to WIFI takes a long time, the following idea was formed: Create a "base station" that is always connected to WIFI and have the ESP32 based electronic die send the thrown face data to the other ESP32 over ESP-now (the proprietary communications layer between two ESP's). Then have this base station ESP-32 handle communications with the server, this setup is displayed in figure 4.7

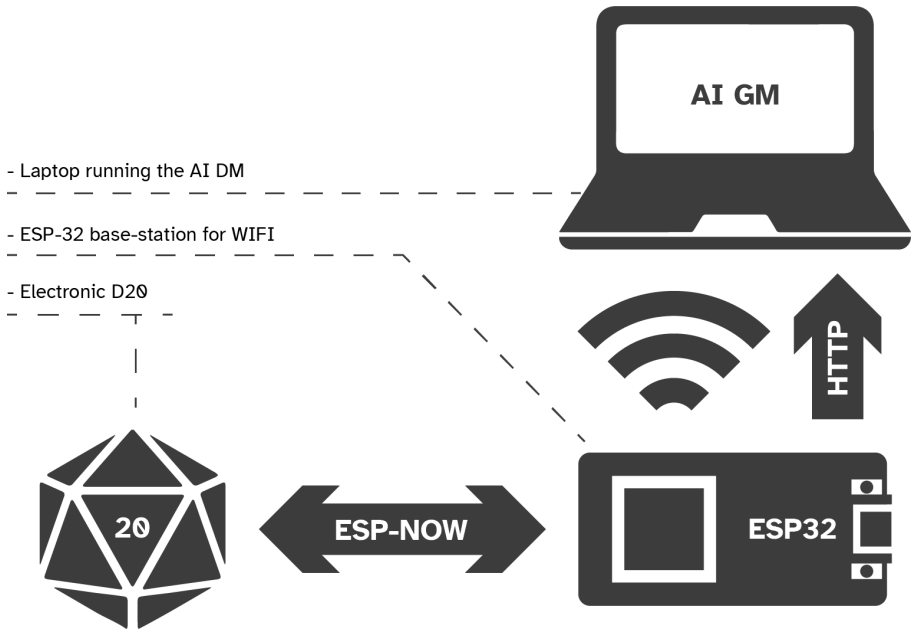

Figure 4.7: Communication diagram between the D20, base station and the AI GM

This setup theoretically should be able to send data in time, however writing the program to handle both WIFI and ESP-NOW simultaneously proved difficult, which is why in the final implementation, the base station consists of two ESP-32s, displayed in

figure 4.8

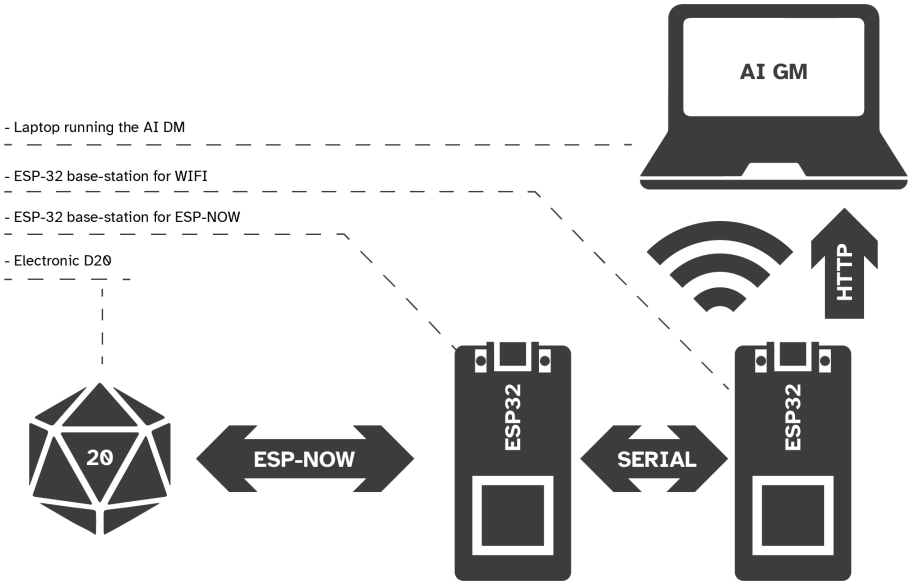

Figure 4.8: Updated communication diagram between the D20, base station and the AI GM

This setup shows a base station consisting of two ESP32s, where the first is responsible for maintaining communications with the D20, and the second ESP is responsible for maintaining WIFI connectivity. The information is being exchanged through serial between the two ESP32s; the final wiring diagram for both the die and the base station is displayed in figure 4.9.

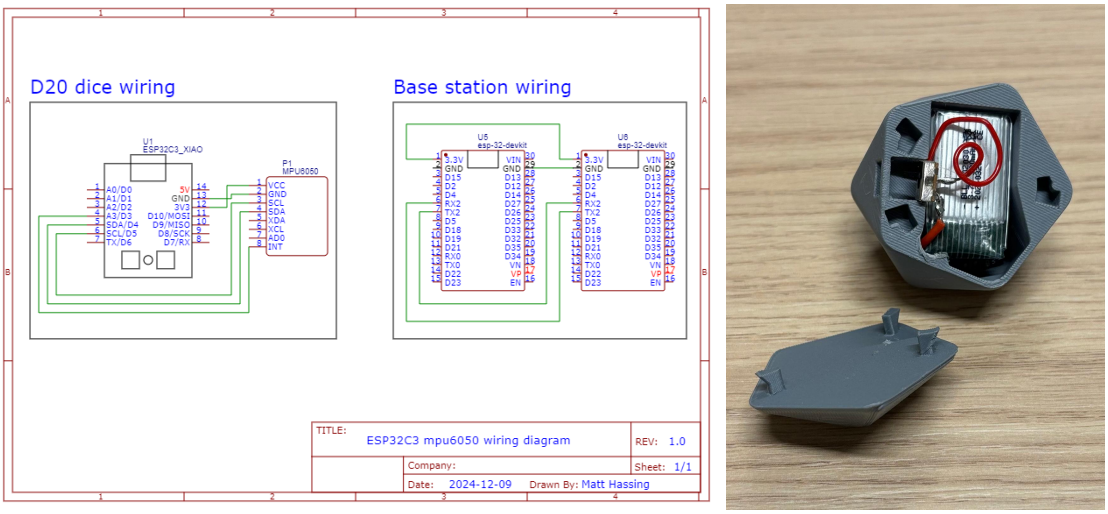

Figure 4.9: Updated ESP32C3 wiring diagram with an MPU6050, and the base-station for serial communication on the left and picture of the prototype on the right

After the addition of the base station, the approximate 2-5 seconds delay now no longer applies, instead being replaced with a couple of milliseconds of delay. This

marks the completion of the final MoSCoW “must haves” of a time between throw and data received of 3 seconds or less.

To conclude, through the application of the TDID an AI GM has been created, which through the fusion of different AI models, produces speech-to-speech AI GM experience, together with this AI GM an electronic D20 has been created to allow the AI GM to know which face has been thrown.

Initially the AI GM faced several difficult challenges related to delays of different modalities added up, however by leveraging the Live kits agents system, a real time storytelling AI GM was created. Similarly the electronic D20 underwent multiple iterations through the TDID. The primary challenge during development of the electronic D20 was balancing size, power efficiency and minimisation of delays.

While some “could have” criteria remain unexplored, the current implementations satisfies the core “must have” criteria it set out to satisfy.

Chapter 5

Final Realisation

The final realization as used during the experiment contains two main parts, the first part is the AI GM and how it will be presented to the participant and the second part will be about the electronic dice and how it will be utilized in the final realization.

The AI GM described in further detail in 4.3 Technology Driven Iterative Design: AI GM Iteration 2, is an agentic AI that, through late-fusion, can combine a speech-to-text model with VAD, a large language model and a text-to-speech model to create a speech-to-speech AI that is instructed to play the TTRPG Dungeons and Dragons together with the participant through speech. It is connected through Livekit's webRTC services to the participant and uses the APIs of multiple commercial vendors, it works through the web, so any browser could use the AI. This whole setup is illustrated in figure 5.1 below.

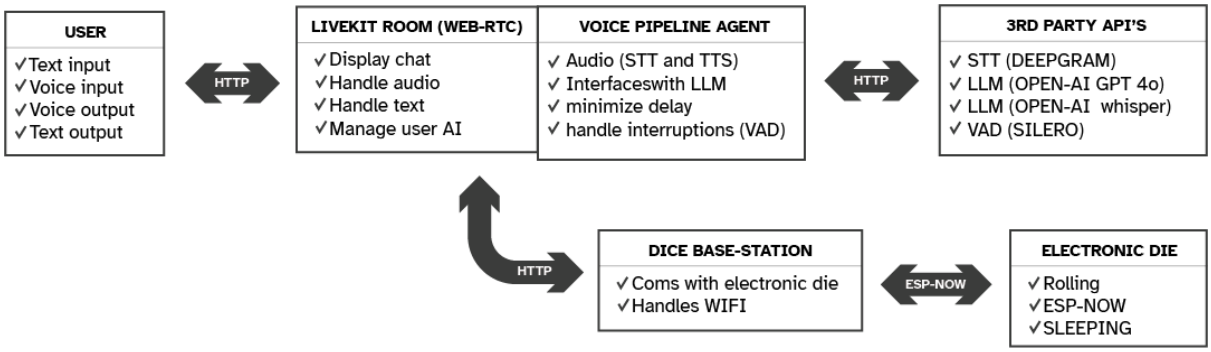

Figure 5.1: Final prototype with check marks showing what was implemented.

The AI is programmed to obey the participants' instruction through prompt engineering, this is a relatively new field of research, where you "program" a large language model through natural language, this way you can instruct the LLM to be pretend to be various things. In this case, this programming was done through the addition of numerous examples and normal instructions, as you would instruct a person to complete the task.

The second part of the final realization is the electronic die. The die, described in further detail in the "Technology Driven Iterative Design: AI GM Iteration 2" chapter, is made out of a three-dimensional printed light grey PLA shell with 20 faces resembling

the D20 shape. It consists of two halves with grooves and matching teeth on both halves so they can be twisted together to form the final D20. As shown in figure 5.2, the light grey D20 contains slots in one half of the print to house the electronics needed, this includes the MPU6050, the chosen inertial measurement unit, the ESP32 and the battery, cutouts have been made to give access to the USB-C port on the esp32 for charging the battery and a switch for powering on and off the D20. Battery life lasts up to a full day of continuous use without the base station present and even longer with the base station present, depending on the frequency at which the dice is thrown. Because of the setup and the slotted placement of the internal components inside the shell, the D20 is not a fair die. It does, however, still function as a die.

Figure 5.2: Image of the final 3D printed prototype of the dice

To conclude, the AI GM together with the electronic D20 form the final prototype that will tested as detailed in Chapter 6. This prototype is an AI GM that uses an electronic D20 as its input modality to know what a participant has thrown, it communicates through speech and understands speech from the participant and uses this to play D&D with them.

Chapter 6

User Experience Evaluation

The final realization, as discussed in Chapter 5, will be evaluated through an experimental setup. First, this experimental setup will be described, and the experiment conducted will be discussed.

6.1 Experimental-Setup

In the Final Realisation Chapter the final prototype has been discussed, including the dice and the final functionality of the web interface, an experiment to evaluate the prototype has been conducted and will be described in this chapter of the report. This experiment aims to understand the user experience (UX) and usability of this prototype, an AI-based GM for TTRPGs. The experimental setup contains three distinct stages, namely the:

1. Introduction and briefing stage.
2. The experiment.
3. The post-experiment questionnaire/interview.

These stages are designed so that the user experience can be evaluated constructively, which leads to consistent, quantifiable results from every participant.

6.1.1 Briefing

Firstly, participants are brought to a room to sign a consent form, which is found in appendix B. This form details the measures taken to protect the participants' privacy, that the study is voluntary, and more rights and information for the participants. After this, the participants will get a verbal briefing, found in appendix C, that contains the following text spoken out loud by the researcher present in the room:

"Next, you will play a Tabletop Role-Playing Game, TTRPG, called Dungeons & Dragons in a session managed by an AI Game Master. You will be playing as a

pre-made character provided to you, and during the game, you will use a 20-sided dice, D20, to interact with the game and resolve actions. Your role will involve making decisions for your character, responding to game scenarios, and rolling the die as needed. The session is designed to replicate a standard TTRPG experience and will last up to 20–30 minutes but can be cut short if you feel like you have had enough. Following the session, you will complete a short questionnaire to share your thoughts and feedback on the experience. Please remember that your participation is voluntary, and you may withdraw at any time. Let us know if you have any questions before we begin!"

A verbal instruction is chosen as this project focuses on the audio capabilities of this prototype, and this primes the participant to communicate through speech and will hopefully counteract the feeling of working in silence as people are used to doing when they are being placed in front of a laptop. The spoken briefing text is meant to inform participants of the experiment that they are about to participate in. To this end, a couple of important points have to be mentioned about the experimental setup to the participants, this is done through the briefing text. The points that should be mentioned and why they should be mentioned are the following :

- **AI GM,** the fact that the participant will speak to, and with, an AI.
- **Pre-made characters,** to inform participants they will be playing as a fictional character and that this character is already configured for them.
- **20-sided dice, D20,** both 20-sided and D20 are mentioned to hopefully familiarize the participant with both terms in the case that they have not yet played a TTRPG or seen a dice with more than 6 sides. This is important since the AI will most likely mention the term D20 during the experiment.
- **Participant role,** it is important to mention what participants are supposed to do to supply them with confidence and agency in the experiment.
- **Session duration,** to iterate that there is a set cut-off time.
- **Reminder that their participation is voluntary and that there is a questionnaire at the end.**

After the verbal briefing, the participant is placed inside of a room, The player is set in front of the laptop and D20, after which the participant is informed that the experiment has been started and that they should say something to the AI to start the session. After this, the participant is left alone with them and the AI, with the researcher remaining in the room to monitor the interactions the participant has with the AI GM. At the conclusion of the session, which is when either the time runs out, or the participant and AI GM have logically come to a conclusion to their story together, the participant will be asked to fill out a questionnaire.

6.1.2 Experiment

For this experiment participants were selected through the method of convenience sampling. through convenience sampling fellow students at the University of Twente will be requested to partake in the experiment. A criterion for participation in this experiment is that participants have to be able-bodied enough to use a laptop, speak, see, and have a method of rolling a D20. Except for these criteria, there are no exclusion criteria other than those outlined by the ethical committee CIS at the University of Twente regarding the ability to consent.

The experiment uses the hardware and software outlined in Chapter 5. The web-interface, will be shown through a laptop, and a D20 will be present that has electronics to connect to the AI. The AI is instructed to follow a single preset scenario, after which the AI will be left in charge with the agency of where the story goes, similar to an unstructured D&D sandbox experience [2], with the key difference being that this is done by an AI.

The preset scenario that will be used in the AI is an open ended dungeon entrance, this is chosen to have a semi-neutral starting point which the participant can use to familiarize themselves with the system, while being right at the edge of adventure, nudging the participant to partake in discovery in the dungeon. The full scenario can be found as part of the code in appendix D. The participant will be made aware by the AI, that there is another person in the party that is being managed by the AI, for this, an NPC named Finn has been designed. He is a trickster wizard who vehemently believes that anything that the player does to him is an attestment to their friendship. The AI is also informed of the following text detailing the details of the dungeon. This is done to give the AI a clear goal to work towards.

You are playing a small session in the world of Karathun. You are at the entrance of the ruins currently. Here is the lore:

- **Quest Giver:** A sage has hired you to find the *Staff of Verdant Whispers*, hidden deep within these ruins centuries ago by druids.
- **Ruins Description:**
  - Cracked mosaic floors depict scenes of ancient rituals.
  - Natural roots have burst through stone, weaving a strange tapestry of life reclaiming this underground space.
- **Challenges:**
  - Magical wards and living plants animated by old nature magic bar the way.
  - Puzzles that require knowledge of druidic lore must be solved.
  - Half-collapsed corridors, swarming kobold minions, and unstable magical crystals that hum ominously.

- **Objective:** Recover the *Staff of Verdant Whispers*.
- **Potential Reward:** A handsome payment and possibly a boon from a nature deity if handled respectfully.

This sandbox-style approach over a more structured experience is chosen since it more accurately shows what this AI GM is capable of. A linear story does not lend itself well to the AI GM, but the sandbox does. It also tests the AI's adaptability and narrative generation better. The experiment lasts roughly 20-40 minutes depending on the interactions between the participant and the AI GM, The AI GM itself is leading the story together with the player and will natural conclude the story in a logical manner.

6.1.3 Questionnaire and data collection

The purpose of the questionnaire is to evaluate the UX and usability of the prototype, to this end both quantitative and qualitative data is required. The questionnaire along with answers can be found in Appendix E. The questionnaire consists of four parts.

Firstly the questionnaire contains some baseline questions to gauge the prior experience with TTRPGs of the participant. Since it is a variable that is outside of the study's control, but will be used to determine which participants are eligible for the expert interviews, it could also have effects on the experience of a participant during the execution of the experiment.

Next, the participant is asked to fill in a System Usability Scale (SUS) [6]. The SUS evaluates a systems user experience, on its usability, it does so through a set of questions which are on the Likert-scale. The SUS was chosen since it is an academic and industry standard that has been used for decades. Advantages include its simplicity and widespread adoption, as well as the fact that the SUS gives clear numeric results, which can be easily interpreted through direct comparisons with benchmark thresholds. This widespread adoption allows for the objective determination of whether the AI GM system meets the usability standards, which is a big advantage it has over for example, heuristic evaluation, which is a method that also assesses UX but is focused on expert-based evaluations instead.

The section that follows after the SUS is the User Engagement Scale (UES-sf) [32]. This is a similar measuring scheme also using the Likert-scale, but instead of evaluating the experience, this is a way to measure engagement. The UES compliments the SUS, since the SUS is a benchmark and does not give insight as to what can be improved on the prototype, on the contrary the UES does give insights to what can be improved, but is used less frequently. The short format version of the UES, aptly named UES-sf (sf for short form), is the one being used in this experiment since the UES is a long list of questions, and the short form version is an version with only 10 questions making it more manageable for a participant to fill in in conjunction with the SUS. The UES-sf has four distinct categories of questions with two or three questions each. The categories are "Reward Factor", "Aesthetic Appeal", "Perceived Usability", and "Focused

Attention". Giving scores for each category allows for fine-grained insight into the prototype's performance from a UX standpoint. Both the SUS and the UES are Likert scale questions and, as such, have resulting quantitative values.

Finally open ended questions are needed to provide a collection of qualitative data, for more nuanced insights in user experiences, the responses to these questions will be read and re-occurring themes will be discussed, but no formal coding will be done to evaluate the responses from these questions.

Some people that self report as expert dungeon masters themselves in the questionnaire will be asked for a short follow up unstructured interview, in the form of a casual conversation about the prototype, combining insights from this interview with the open ended questions will give rise to a discussion in the results section.

6.2 Results

In this section the results of the experiment will be discussed, both the results from the SUS and UES-sf are going to be discussed, after which the findings from the questionnaire together with the interview will be represented. The results can be found in Appendix E.

In total 20 people partook in the experiment, most of the participants reported being slightly familiar with DND (35%), with some being not experienced at al (30%), some being expertly experienced (10%) and one participant reporting as very experienced. The majority has played DND at least once (60%), but a significant portion had limited exposure, having never played before (40%). Of the twelve people who reported having played D&D, only five people reported having played more than five sessions. This spread of players shows the importance of the system being catered towards less experienced players since they make up the majority of the group whilst remaining sufficiently robust for more experienced players.

From the quantifiable data with Likert scales, the following figures could be calculated. Firstly, the System Usability Scale returns a value of **83** out of 100. The rest can be extrapolated from the box plot in figure 6.1. To interpret these results, a curved grading scale can be applied, as detailed by Jeff Sauro and James R. Lewis in their book published in 2016 [40] and shown in table 6.1. According to the table, the prototype sits comfortably within the **90-95th percentile range**, and as such, is considered an excellent prototype. This agrees with remarks from the qualitative data, highlighting the system's ease of use. They state things such as: "I liked how it gave me the freedom to change stuff" by participant 3 and "It guided me nicely through everything." by the novice, participant 10, displaying that the system was intuitive even for less experienced players.

SUS Score	Percentile Range	Grade	Rank
84.1 - 100	96 –100	A+	Best imaginable

SUS Score	Percentile Range	Grade	Rank
80.8 - 84.0	90 –95	A	
78.9 - 80.7	85 –89	A–	
77.2 –78.8	80 –84	B+	Excellent
74.1 –77.1	70 –79	B	
72.6 –74.0	65 –69	B–	
71.1 –72.5	60 –64	C+	Good
65.0 –71.0	41 –59	C	
62.7 –64.9	35 –40	C–	
51.7 –62.6	15 –34	D	Okay
0 –51.6	0 –14	F	Poor

Table 6.1: SUS Score grading table with percentile ranges, grades, and ranks

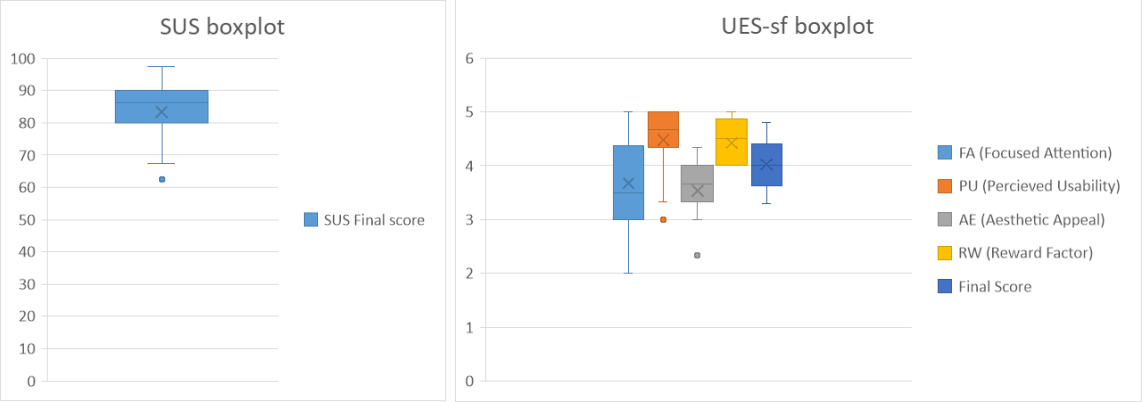

Figure 6.1: SUS and UES-sf grading scores, represented as boxplots

There is still room for improvement however, this becomes apparent from the evaluation of the UES-sf scores. The UES-sf asses a prototype based on four categories: Reward Factor (RW), Aesthetic Appeal (AE), Perceived Usability (PU), and Focused Attention (FA). It gives numerical grades from one to five for each category, to simplify our understanding, an extra column is added to give the grades a similar range as the SUS score. This column has changed them to be out of 100 by simply multiplying the scores by 20. The resulting scores are shown below in table 6.2

Metric	Original	Out of 100
RW (Reward Factor)	4.43	88.50
AE (Aesthetic Appeal)	3.53	70.67
PU (Perceived Usability)	4.48	89.67
FA (Focused Attention)	3.68	73.50
Original UES Score: 4.03		
Out of 100: 80.50		

Table 6.2: Table for UES-sf metrics, containing both original scores, and out of 100 scores

The overall score of 4.03 indicates a strong prototype, similar to how the SUS indicates an excellent prototype. The SUS score is further supported by the high Perceived Usability score (4.48/5). Another thing that hindered the enjoyment and could be used to explain the low FA score, is what participant 13 noted: "I got sidetracked and the GM didn't remind me of the original quest". This happened when one participant was never reminded of the original goal of their journey, this might be one of these immersion breaking moments that make a generated narrative feel thin. The low score in the AE category can be combatted by carefully redesigning the User Interface (UI). Multiple participants indicated the need for visuals during the narration of the adventures by the AI GM. This is supported by the answer of several participants who indicated that visuals of the characters, enemies or scenery would enhance their enjoyment, in the question about what they would improve about the experience. There was a noticeable difference between participant segments. Novice players reported that the system's guidance helped them understand what to do. Statements like: "It guided me nicely through everything and gave me options and inspiration." Matches the high SUS and PU scores for them. Experienced players, on the other hand, appreciated the open nature but wanted more structure around dice mechanics, which required further balancing of the spell, suggesting the need for more structured handling of such tasks. This might have led to reduced FA for them if they felt like the AI GM was allowing them to do things the experienced participants felt like they should not be able to do. One participant noted the AI GM allowed them to do actions that "should not be possible with my character class." expert 1.

6.2.1 Expert Interviews

In this section, the contents of two expert interviews conducted after the experiment are detailed. These interviews were in the form of open discussion and will be processed as such, including insights about them.

During the two interviews with two DnD experts, some things became apparent. One key observation was that the AI was very much a "Yes man," stating that the AI did not give adequate "pushback" against their wishes. This resulted in an experience where everything was allowed, leading to a break in immersion.

Expert 1, for example, did not work together with the AI but rather played against the AI, resulting in Expert 1's ability to manipulate the AI's decision-making process into agreeing with whatever they wanted to do, even going as far as making it so that the AI did not require this individual to roll the D20 for their choices, and ascending to "godhood" in the story. On the one hand, being able to overpower the AI in such a way is a known issue that is present in LLM-based applications in general and should be resolved. On the other hand, the second expert noted that if they were the GM and a player tried to overpower them, they would most likely not be invited to play DnD again with them next time. Expert 1's experience did, however, diverge significantly from the mean participant experience. This statement is further supported by the fact that Expert 1 gave the second to least high SUS score out of all participants. Being able to overpower the AI GM is a pressing problem that should not be ignored, especially since both experts agreed that the AI GM is a yes man, with even Expert 2 stating that "It is a good simulation of DnD if you play with a people pleaser as a Game Master". Another point of friction Expert 1 noted was the longwindedness of the explanations of current events during gameplay, especially during combat, this would be annoying. But they did say that they viewed this as a fun experience. Expert 2 agreed mostly with Expert 1's views, but while both noted the same issues, Expert 2 was less convinced about the level of impact the issues had on the experience, compared to Expert 1.

The experts did, however, have differing opinions on whether or not this would be a good onboarding experience for novice players. Expert 2 stated that this would be a very good introductory tool, while Expert 1 said this would give a "very weird view" of how DnD usually goes. Expert 2's opinion is more in line with the feedback received from the novice players, being that they enjoyed the game and found the system compelling. The feedback from Expert 1 largely makes sense regarding their experience, but due to the nature of AI, their experience was very different from the average experience with the AI GM, this becomes apparent from their individually SUS and UES-sf scores compared to the mean scores as well, as can be found in Table 6.3 and 6.4.

Participant	**SUS Score**
Overall Average	83.4
Expert 1	67.5
Expert 2	87.5

Table 6.3: Comparison of System Usability Scale (SUS) Scores

	FA	PU	AE	RW	Final Score
Overall Avg.	3.68	4.48	3.53	4.43	4.03
Expert 1	3.5	4.33	3.67	4.5	4.0
Expert 2	5.0	5.00	4.33	5.0	4.8

Table 6.4: Comparison of UES-sf Scores (Overall vs. Experts)

Another problem that needs addressing, on which both experts agreed, is that the AI was too lenient towards the players, expert 2 noted that: “The AI GM should be based on the rules for soft magic systems, which state that if something is not previously explained in a story, it should not solve a newly presented problem for the protagonist.” This aligns with a broader statement about story writing proposed by Brandon Sanderson [39] “An author’s ability to solve conflicts with magic is directly proportional to how well the reader understands said magic.” In a broader sense: If a solution is introduced suddenly without prior explanation, it weakens the story’s internal logic. The AI GM violated this rule by allowing players to create new items or abilities without justification, reducing the challenge and immersion of gameplay. There were times when the participant could say something along the lines of I grabbed this wand of solving the problem I am currently facing out of my pocket and used it to solve the current problem, to which the AI would go, okay, roll a D20 instead of stating that the participant had no such item in their possession.

Overall, both experts enjoyed the experience, with Expert 1 being more negative about the AI-GM, which could perhaps be attributed to their anomalous story experience with the AI-GM GM and Expert 2, who had a more typical story experience, being more positive about the AI-GM. Both experts highlighted that the AI-GM was too permissive and that it was immersion-breaking that the AI-GM allowed players to create items that they did not have before.

Chapter 7

Discussion

Overall, participants were generally positive about the AI GM, with an average SUS score of 83. There was, however, a clear difference between novice and experienced participants. Novice participants reported that they felt like they were guided properly, while experienced players appreciated the AI's strong narrative capabilities but found that the amount of freedom and openness lacked challenge and hardship, with some experienced participants explicitly requesting stricter rule adherence when asked if they wanted to change anything about the AI GM. Showing that the level of TTRPG experience is an influence on how the AI GM is perceived.

Based on both the qualitative data gathered from the questionnaire and the expert interviews, the most frequently cited critique on the AI GM was its eagerness to agree with the participants, with one expert even going as far as calling the AI GM a "people pleaser". The people pleasing, could be a part of the reason behind the relatively low immersion score found in the UES-sf part of the questionnaire. From a more technical standpoint, this issue likely stems from the origins of this AI. It illustrates a larger issue regarding the difficulty LLMs face in general when enforcing negative constraints. In contrast, most people also cited the AI GM's freedom as the part they enjoyed the most. There must be a design trade-off between avoiding player frustration by providing creative freedom and enforcing negative constraints. A balance which this prototype has not been able to find yet.

However, despite the lack of balance, another big issue was presented during the expert interviews, which the novice players never brought up in their feedback, which was the lack of firm rule enforcement. Participants exploited the AI GM by creating items or powers not previously defined, effectively bypassing normal game mechanics. Drawing on Sanderson's First Law of Magic, Expert 2 said that introducing unexpected solutions makes for poor storytelling, this resonates with prior research on procedural generation and narrative design, highlighting the need for a robust rule system to make sure that the AI GM does not undermine its own challenges. Creating such a system could be interesting future research avenue, however since no novice players reported the lack of rule enforcement anywhere in their qualitative answers, partly due to their unfamiliarity with the rules, this issue could therefore be more prominent for more experienced players.

Another point of interest for many participants was the physical electronic D20, participants generally strongly enjoyed engaging with the AI through rolling the D20, despite this large set of positive feedback, some participants raised the concern that due to the distribution of the weight in the D20, the D20 might not role fairly. During the UX study no such flaw became apparent, however this must be the case since no effort was made to balance the D20. This could be an interesting avenue for future research. However, it had little impact on the narratives experienced by the players when interacting with the AI GM.

While the AI GM's speech-based interactions were well received and praised, many participants felt that the user interface lacked a certain appeal, the UES-sf scores in the Aesthetic Appeal category indicated that many participants wanted more visuals, some participants reported that they would like to see for example imagery of their characters, surroundings or enemies they encountered. This form of multimodal feedback could help increase the low immersion score found in the UES-sf as well, especially during character interactions and fights.

Although great efforts were made to reduce latency at every point in the interactions possible, unfortunately, participants still noted occasional delays, for example, in speech recognition or responses. Though overall latency was mostly negligible, through improvements like the base station for dice interactions, there were still noticeable delays during both speech play-out and dice-throw recognitions. Most of these can be explained by the testing location and circumstances. The whole system was running off a mobile hot spot, 10 stories high, during poor weather conditions. It was surprising that the system held up so well under those conditions, in the future this system could be improved to not be reliant on a mobile hot spot, which should mitigate most if not all these issues.

The current implementation of the AI GM can only handle a single participant and a single GM. This is largely due to the inability of the speech-to-text AI to distinguish speech from multiple people in a room; in the future, if the speech-to-text AI is able to track who said what, this would be completely feasible with little to no changes to the logic of the rest of the AI GM.

Overall, the results show that the AI GM, despite its flaws, is a promising avenue for creating more accessible TTRPG experiences that all participants reportedly found pleasant. At the same time, it shows how complex the task of a GM can be and how much effort is needed to juggle both the complexity of maintaining tension, spinning up a creative narrative, delivering engaging gameplay and doing so while maintaining the rules of the game.

Chapter 8

Conclusion

Recent advancements in AI open up new opportunities for TTRPGs. This project set out to answer: **How can speech-to-speech AI be used in conjunction with an electronic 20 sided dice to serve as a 'Game Master' for tabletop role-playing games?** The Technology-Driven Iterative Design process led to the development of a speech-to-speech AI GM. This system uses late fusion to combine text-to-speech with voice activity detection, a large language model for natural language processing, and speech-to-text. This resulted in an AI GM that could be used with a single participant alongside an electronic D20 to play a simplified version of Dungeons and Dragons. The system was then evaluated against a small group of convenience-sampled participants (20 participants). The results from this experiment displayed a positive user experience for most test subjects, with a SUS score of 83, which is in the 90th to 95th percentile range and a UES-sf score of 4.03 out of 5. The UX study showed that particularly novice players without a concept of what D&D is and how it works were greatly supported by the system. The main criticism here being, mostly from more experienced players is that the AI GM does not keep track of what spells or items a player had and was allowed to use. Another obstacle for more experienced players was, that the AI GM easily agreed to suggestions by the players and not really pushed back, this was however also seen as a big strength of the system, with multiple participants citing the freedom of the AI GM as one of its enjoyable qualities.

Further research could focus on making the AI GM follow the rules more closely, as well as finding a more scalable version of the system. Another future avenue of research could be to make the AI GM allow for multiple players concurrently or add visuals since many participants said adaptive visual generation could greatly improve their enjoyment of the game. Ultimately, this project demonstrated that multi-modal AI can provide a novel foundation for creating adaptive story games that allow participants to pick their own destinies.

Bibliography

[1] *(26) ChatGPT hits 100M weekly users | LinkedIn*. url: https://www.linkedin.com/news/story/chatgpt-hits-100m-weekly-users-5808204/ (visited on 11/21/2024).

[2] AdornThemes. *How to Create a Sandbox Game in D&D*. en. Apr. 2024. url: https://thedmlair.com/blogs/news/how-to-create-a-sandbox-game-in-d-d (visited on 01/23/2025).

[3] *Angular - What is Angular?* url: https://v17.angular.io/guide/what-is-angular (visited on 01/18/2025).

[4] Shahar Avin, Ross Gruetzemacher, and James Fox. "Exploring AI Futures Through Role Play". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES '20. New York, NY, USA: Association for Computing Machinery, Feb. 2020, pp. 8–14. isbn: 978-1-4503-7110-0. doi: 10.1145/3375627.3375817. url: https://dl.acm.org/doi/10.1145/3375627.3375817 (visited on 10/09/2024).

[5] Tadas Baltrusaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal Machine Learning: A Survey and Taxonomy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2 (Feb. 2019), pp. 423–443. issn: 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2018.2798607. url: https://ieeexplore.ieee.org/document/8269806/ (visited on 09/21/2024).

[6] john Brooke. "SUS: A 'Quick and Dirty' Usability Scale". In: *Usability Evaluation In Industry*. Num Pages: 6. CRC Press, 1996. isbn: 978-0-429-15701-1.

[7] Steph Buongiorno et al. *PANGeA: Procedural Artificial Narrative using Generative AI for Turn-Based Video Games*. arXiv:2404.19721 [cs]. July 2024. doi: 10.48550/arXiv.2404.19721. url: http://arxiv.org/abs/2404.19721 (visited on 09/21/2024).

[8] Charles B. Callaway and K. Sima'an. "Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship". In: July 2005. url: https://www.semanticscholar.org/paper/Wired-for-Speech%3A-How-Voice-Activates-and-Advances-Callaway-Sima'an/09d9e89983b07c589e35196f4a0c161987042670 (visited on 01/18/2025).

[9] *ChatGPT can now see, hear, and speak*. en-US. url: https://openai.com/index/chatgpt-can-now-see-hear-and-speak/ (visited on 11/21/2024).

[10] *Circuit Board - Game With Pixels*. url: https://gamewithpixels.com/features/circuit-board/ (visited on 12/09/2024).

[11] Dai Clegg and Richard Barker. *CASE method fast track: a RAD approach*. eng. 1. pr. Computer aided systems engineering. Wokingham, England Reading, Mass: Addison-Wesley Pub. Co, 1994. isbn: 978-0-201-62432-8.

[12] *Components | Pixels | Hackaday.io*. url: https://hackaday.io/project/28377/components (visited on 12/09/2024).

[13] Edirlei Soares De Lima et al. "ChatGeppetto - an AI-powered Storyteller". en. In: *Proceedings of the 22nd Brazilian Symposium on Games and Digital Entertainment*. Rio Grande (RS) Brazil: ACM, Nov. 2023, pp. 28–37. isbn: 9798400716270. doi: 10.1145/3631085.3631302. url: https://dl.acm.org/doi/10.1145/3631085.3631302 (visited on 09/21/2024).

[14] *Django documentation | Django documentation*. en. url: https://docs.djangoproject.com/en/5.1/ (visited on 01/18/2025).

[15] *ESP32*. en. Page Version ID: 1261606949. Dec. 2024. url: https://en.wikipedia.org/w/index.php?title=ESP32&oldid=1261606949 (visited on 12/09/2024).

[16] Aleksandar Filipović. "THE ROLE OF ARTIFICIAL INTELLIGENCE IN VIDEO GAME DEVELOPMENT". English. In: *Kultura Polisa* 20.3 (2023). Publisher: [illegible] „[illegible]–[illegible]" [illegible], pp. 50–67. issn: 1820-4589, 2812-9466. url: https://www.ceeol.com/search/article-detail?id=1201751 (visited on 10/09/2024).

[17] *Flask vs Django in 2025: Which One to Choose and When?* url: https://www.netsolutions.com/insights/flask-vs-django (visited on 01/18/2025).

[18] Jamie Fraser, Ioannis Papaioannou, and Oliver Lemon. "Spoken Conversational AI in Video Games-Emotional Dialogue Management Increases User Engagement". In: Oct. 2018. doi: 10.1145/3267851.3267896.

[19] Robert Friedman. "Tokenization in the Theory of Knowledge". en. In: *Encyclopedia* 3.1 (Mar. 2023), pp. 380–386. issn: 2673-8392. doi: 10.3390/encyclopedia3010024. url: https://www.mdpi.com/2673-8392/3/1/24 (visited on 01/18/2025).

[20] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. "Early vs Late Fusion in Multimodal Convolutional Neural Networks". In: *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. July 2020, pp. 1–6. doi: 10.23919/FUSION45008.2020.9190246. url: https://ieeexplore.ieee.org/document/9190246/?arnumber=9190246 (visited on 10/14/2024).

[21] Sushant Gautam. “Bridging Multimedia Modalities: Enhanced Multimodal AI Understanding and Intelligent Agents”. en. In: *INTERNATIONAL CONFERENCE ON MULTIMODAL INTERACTION*. Paris France: ACM, Oct. 2023, pp. 695–699. isbn: 9798400700552. doi: 10.1145/3577190.3614225. url: https://dl.acm.org/doi/10.1145/3577190.3614225 (visited on 09/21/2024).

[22] Matt Hassing. *Final version Literature Review - Own Work*. Oct. 2024.

[23] Sergio Poo Hernandez, Vadim Bulitko, and Marcia Spetch. “Keeping the Player on an Emotional Trajectory in Interactive Storytelling”. en. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 11.1 (2015). Number: 1, pp. 65–71. issn: 2334-0924. doi: https://doi.org/10.1609/aiide.v11i1.12783. url: https://ojs.aaai.org/index.php/AIIDE/article/view/12783 (visited on 09/21/2024).

[24] Jean Simonet. *In-depth look at My Awesome Electronic Dice :)* Dec. 2017. url: https://www.youtube.com/watch?v=5uRe2gKeFX4 (visited on 12/09/2024).

[25] Zixuan Jiang. “Emotional Simulation in Game AI and Its Impact on Player Experience”. en. In: *International Journal of Education and Humanities* 13.2 (Apr. 2024). Number: 2, pp. 11–13. issn: 2770-6702. doi: 10.54097/pbhsqy34. url: https://drpress.org/ojs/index.php/ijeh/article/view/19621 (visited on 10/09/2024).

[26] Kostas Karpouzis and George Tsatiris. “AI in (and for) Games”. In: Jan. 2022, pp. 27–43. isbn: 978-3-030-76793-8. doi: 10.1007/978-3-030-76794-5_3.

[27] Raul Lara-Cabrera et al. “Game artificial intelligence: Challenges for the scientific community”. In: *CEUR Workshop Proceedings* 1394 (Jan. 2015), pp. 1–12.

[28] David B. Leake. “Artiicial Intelligence”. In: 2001. url: https://www.semanticscholar.org/paper/Artiicial-Intelligence-Leake/86d763790bb7da53d8ad69c0 (visited on 09/21/2024).

[29] *LiveKit*. en. url: https://livekit.io (visited on 11/21/2024).

[30] Angelika H. Mader and Wouter Eggink. “A Design Process for Creative Technology”. Undefined. In: *Proceedings of the 16th International conference on Engineering and Product Design, E&PDE 2014*. The Design Society, Sept. 2014, pp. 568–573. url: https://research.utwente.nl/en/publications/a-design-process-for-creative-technology (visited on 11/04/2024).

[31] *Node.js - MDN Web Docs Glossary: Definitions of Web-related terms | MDN*. en-US. Aug. 2024. url: https://developer.mozilla.org/en-US/docs/Glossary/Node.js (visited on 01/18/2025).

[32] Heather L. O'Brien, Paul Cairns, and Mark Hall. "A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form". In: *International Journal of Human-Computer Studies* 112 (Apr. 2018), pp. 28–39. issn: 1071-5819. doi: 10.1016/j.ijhcs.2018.01.004. url: https://www.sciencedirect.com/science/article/pii/S1071581918300041 (visited on 01/18/2025).

[33] Zhenhui Peng et al. "Understanding User Perceptions of Robot's Delay, Voice Quality-Speed Trade-off and GUI during Conversation". en. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. Honolulu HI USA: ACM, Apr. 2020, pp. 1–8. isbn: 978-1-4503-6819-3. doi: 10.1145/3334480.3382792. url: https://dl.acm.org/doi/10.1145/3334480.3382792 (visited on 01/23/2025).

[34] Luis Manuel Pereira, Addisson Salazar, and Luis Vergara. "A Comparative Analysis of Early and Late Fusion for the Multimodal Two-Class Problem". In: *IEEE Access* 11 (2023), pp. 84283–84300. issn: 2169-3536. doi: 10.1109/ACCESS.2023.3296098. url: https://ieeexplore.ieee.org/document/10185035/ (visited on 10/14/2024).

[35] *PHP: Documentation*. en. url: https://www.php.net/docs.php (visited on 01/18/2025).

[36] *React*. en. url: https://react.dev/ (visited on 01/18/2025).

[37] Mark Owen Riedl and Alexander Zook. "AI for game production". In: *2013 IEEE Conference on Computational Inteligence in Games (CIG)*. ISSN: 2325-4289. Aug. 2013, pp. 1–8. doi: 10.1109/CIG.2013.6633663. url: https://ieeexplore.ieee.org/document/6633663 (visited on 10/09/2024).

[38] *Ruby on Rails*. en. url: https://rubyonrails.org/ (visited on 01/18/2025).

[39] Brandon Sanderson. *Arcanum Unbounded: the Cosmere Collection*. eng. OCLC: 962850091. London, England: Gollancz, 2016. isbn: 978-1-4732-1804-8.

[40] Jeff Sauro and James R. Lewis. *Quantifying the User Experience: Practical Statistics for User Research*. en. Google-Books-ID: USPfCQAAQBAJ. Morgan Kaufmann, July 2016. isbn: 978-0-12-802548-2.

[41] *Seeed Studio XIAO ESP32C3*. en. url: https://www.kiwi-electronics.com/en/seeed-studio-xiao-esp32c3-11039 (visited on 12/09/2024).

[42] Akshay Sharma, Abhishek Srivastava, and Adhar Vashishth. "An Assistive Reading System for Visually Impaired using OCR and TTS". In: *International Journal of Computer Applications* 95.2 (June 2014), pp. 13–18. issn: 09758887. doi: 10.5120/16566-6231. url: http://research.ijcaonline.org/volume95/number2/pxc3896231.pdf (visited on 01/18/2025).

[43] *Vue.js*. en-US. url: https://vuejs.org/ (visited on 01/18/2025).

[44] Ruixi Wang. “Review of Generative Models”. In: *Applied and Computational Engineering* 8.1 (Aug. 2023), pp. 524–529. issn: 2755-2721, 2755-273X. doi: 10.54254/2755-2721/8/20230269. url: https://www.ewadirect.com/proceedings/ace/article/view/2488 (visited on 01/18/2025).

[45] *Welcome to Flask — Flask Documentation (3.1.x)*. url: https://flask.palletsprojects.com/en/stable/ (visited on 01/18/2025).

[46] *What is Full Stack Development? - Full Stack Development Explained - AWS*. en-US. url: https://aws.amazon.com/what-is/full-stack-development/ (visited on 01/18/2025).

[47] *What is Full Stack Development? - Full Stack Development Explained - AWS*. url: https://aws.amazon.com/what-is/full-stack-development/ (visited on 01/18/2025).

[48] *XIAO ESP32C3*. en. url: https://docs.nordicsemi.com/bundle/ncs-latest/page/zephyr/boards/seeed/xiao_esp32c3/doc/index.html (visited on 12/09/2024).

[49] Qi-Yue Yin et al. “AI in Human-computer Gaming: Techniques, Challenges and Opportunities”. In: *Machine Intelligence Research* 20 (Jan. 2023). doi: 10.1007/s11633-022-1384-6.

[50] Wayne Xin Zhao et al. *A Survey of Large Language Models*. Version Number: 15. 2023. doi: 10.48550/ARXIV.2303.18223. url: https://arxiv.org/abs/2303.18223 (visited on 01/18/2025).

Appendices

Appendix A

AI-notice

During the preparation of this work, I used ChatGPT, Elicit, litmaps and paperdigest to help structure content, aid in programming and find academic sources. After using these tools and services, I thoroughly reviewed and edited the content as needed, taking full responsibility for the final outcome.

Appendix B

Consent Form

The ethical consent form participants had to sign to enter the research

Consent form

Project Title: *Dicey DM'ing: Developing an AI-based Physical Roleplaying Artifact*

Researcher: Matt Hassing, B-CREA, University of Twente

Contact Information: m.hassing@student.utwente.nl

Purpose of the Study:

This study aims to explore the integration of AI into tabletop role-playing games (TTRPGs) like *Dungeons & Dragons*. Specifically, we will test an AI Game Master system compared to traditional human Game Masters. Participants will engage with the AI system or human GM during a playtest and complete a questionnaire afterward.

Study Procedures:

1. You will be introduced to the research and sign this consent form.
2. You will play a session with either the AI Game Master or a traditional Game Master.
3. Following the session, you will fill out a questionnaire to evaluate their experiences.

Please note that the AI system we are using is developed by OpenAI and is thus created by a third-party, commercial entity. While OpenAI has implemented safeguards to minimize risks, the AI may still produce outputs that reflect biases or stray into topics that could be considered harmful or inappropriate. We do not have control over these biases.

To ensure your safety, the session will be monitored in real time by the researcher, who will intervene or stop the interaction if biased or unsafe content appears.

Voluntary Participation:

Your participation in this research is completely voluntary. You may withdraw at any time without providing a reason, this will lead to the destruction of all information gathered within the research, including transcripts and consent forms.

Data Protection and Confidentiality:

- The only personally identifiable information collected is the signed consent form.
- Responses to the questionnaire will be anonymized.
- We will use only the transcript of your conversation with the AI from your session, any other information used by the AI during the session will be permanently deleted within 5 minutes of the session's conclusion
- All data will be used exclusively for this research project and destroyed after its conclusion anticipated to be January 31, 2025.

Participant Rights:

- You may refuse to answer any question or terminate your participation at any time.

- Anonymized data may be included in published research but will not be linked to your identity.

Contact for Questions or Concerns:

If you have any questions about the study, please contact the researcher at m.hassing@student.utwente.nl. For concerns about your rights as a participant or to report a problem, you may contact the Ethics Committee at ethicscommittee-cis@utwente.nl.

Consent Declaration:

By signing this form, you agree to the following:

☐ I have read and understood the provided information.

☐ I agree to participate in this study voluntarily.

☐ I understand that I can withdraw at any time without explanation.

☐ I consent to the collection and anonymization of my questionnaire responses.

☐ I understand that all data used by the AI during the session (except transcripts) will be deleted within 5 minutes of the session's conclusion.

☐ I understand that the AI may produce outputs that reflect biases or stray into topics that could be considered harmful or inappropriate.

☐ I understand that the session will be monitored by the researcher remotely

Signature:

Participant Name: _____

Participant Signature: _____

Date: _____

Appendix C

Briefing

Debriefing at the Start of the Experiment

Following will be read to each participant

Next, you will play a Tabletop Role-Playing Game (TTRPG) called *Dungeons & Dragons* in a session managed by an AI Game Master. You will be playing as a pre-made character provided to you, and during the game, you will use a 20-sided die *(D20)* to interact with the game and resolve actions. Your role, will involve making decisions for your character, responding to game scenarios, and rolling the die as needed. The session is designed to replicate a standard TTRPG experience and will last approximately 20–30 minutes. Following the session, you will complete a short questionnaire to share your thoughts and feedback on the experience. Please remember that your participation is voluntary, and you may withdraw at any time. Let us know if you have any questions before we begin!

Figure C.1: Briefing read verbally to a participant during the experiment

Appendix D

Programs, Back-end

The back-end python program keeping the LiveKit AI GM running

```python
import asyncio
from typing import Annotated
from livekit import agents, rtc
from livekit.agents import JobContext, WorkerOptions, cli, tokenize, tts, stt, llm
from livekit.agents.llm import ChatContext, ChatMessage
from livekit.agents.voice_assistant import VoicePipelineAgent
from livekit.plugins import deepgram, openai, silero
import logging
from dotenv import load_dotenv
import os
from aiohttp import web

# Load environment variables
load_dotenv()

# Access your environment variables
openai_api_key = os.getenv('OPENAI_API_KEY')
deepgram_api_key = os.getenv('DEEPGRAM_API_KEY')
livekit_url = os.getenv('LIVEKIT_URL')
livekit_api_key = os.getenv('LIVEKIT_API_KEY')
livekit_api_secret = os.getenv('LIVEKIT_API_SECRET')

logger = logging.getLogger("Assistant")
logger.setLevel(logging.INFO)

# Initialize an asynchronous queue for dice rolls
dice_roll_future = None  # Future to manage active waiting for dice rolls

async def run_http_server(chat_context, answer_fnc):
```

```python
    app = web.Application()
    # Store references in app's state
    app['chat_context'] = chat_context
    app['answer_fnc'] = answer_fnc

    app.add_routes([web.post('/dice_roll', handle_dice_roll)])

    runner = web.AppRunner(app)
    await runner.setup()
    site = web.TCPSite(runner, '0.0.0.0', 8081)
    await site.start()
    logger.info("HTTP server started on port 8081 for dice roll submissions.")

    # Keep the server running indefinitely until this task is canceled
    await asyncio.Future()

async def handle_dice_roll(request):
    global dice_roll_future
    raw_body = await request.text()  # Log raw body for debugging
    logger.info(f"Raw request body: {raw_body}")

    data = await request.json()
    dice_result = data.get("roll")

    if dice_result is None:
        logger.warning("No 'roll' field in received data.")
        return web.Response(text="Invalid data format.", status=400)

    logger.info(f"Dice roll received: {dice_result}")

        # # If let_player_roll_dice is waiting, fulfill its future; otherwise, _answer with "rolled <dice_result>"
        # if dice_roll_future and not dice_roll_future.done():
        #     dice_roll_future.set_result(dice_result)
        #     logger.info("Dice roll delivered to waiting function.")
        # else:
        # Instead of adding a message to the chat_context, call _answer
    answer_fnc = request.app['answer_fnc']

    chat_context = request.app['chat_context']
    chat_context.messages.append(
        ChatMessage(role="system", content=f"Dice roll result: {dice_result}")
    )
```

```
72
73      asyncio.create_task(answer_fnc(f"rolled {dice_result}"))
74
75      return web.Response(text="Dice roll received.")
76
77
78  class AssistantFnc(llm.FunctionContext):
79      @llm.ai_callable(
80      description="Called to retrieve character data from the database.",
81      )
82      async def get_character_info(
83          self,
84          character_name: Annotated[str, llm.TypeInfo(description="The name
            ↪ of the character to search for, or leave empty for
            ↪ all_summaries")] = None,
85      ):
86
87          return "A big burly minotaur"
88
89
90  async def entrypoint(ctx: JobContext):
91      await ctx.connect()
92      print(f"Room name: {ctx.room.name}")
93      global chat_context
94      chat_context = ChatContext(
95          messages=[
96              ChatMessage(
97                  role="system",
98                  content=(
99                      """ You are an GM for a dnd 5e campaign, you also have
                        ↪ a bunch of functions to your disposal to help you
                        ↪ with your campaign.
100                     if you CAN you NEED to call a function.
101                     You need to ask the user to roll the D20 similar to
                        ↪ how its done in DND
102
103                     You are playing a small session in the world of
                        ↪ Karathun, you are at the entrance of the runes
                        ↪ currently. here is the lore:
104                     'A sage has hired you to find the Staff of Verdant
                        ↪ Whispers, hidden deep within these ruins centuries
                        ↪ ago by druids.'
```

```
105                     'Cracked mosaic floors depict scenes of ancient
                        ↪  rituals. Natural roots have burst through stone,
                        ↪  weaving a strange tapestry of life reclaiming this
                        ↪  underground space.'
106                     'Magical wards and living plants animated by old
                        ↪  nature magic bar the way. Puzzles that require
                        ↪  knowledge of druidic lore must be solved.
                        ↪  Half-collapsed corridors, swarming kobold minions,
                        ↪  and unstable magical crystals that hum ominously.'
107                     'Recover the Staff of Verdant Whispers. Potential
                        ↪  reward: a handsome payment and possibly a boon
                        ↪  from a nature deity if handled respectfully.'
108
109                     you have a party of 2, one player and one NPC, the
                        ↪  player is the one you are interacting with
                        ↪  currently and the NPC you have to play yourself.
110                     the npc you play is a wizard named "Finn" (chaotic
                        ↪  good personality ) who is a bit of a trickster but
                        ↪  has a good heart.
111                     adhere strictly to the DND 5e rules! but more
                        ↪  importantly make the player have fun.
112                     as part of the adhering to the rules be sure to stick
                        ↪  to the game loop including combat and exploration.
113                     make a user roll the dice, if they do not want to do
                        ↪  not progress the game.
114                     ask the player which character they want to play then
                        ↪  look it up using the function calls.
115
116                     remember to act out scenarios and describe the world
                        ↪  to the player also keep encounters interesting and
                        ↪  challenging.
117                     strictly adhere to the game loops of DND 5e, and
                        ↪  remember to have fun!
118
119                     keep it short and sweet, and remember to have fun!
                        ↪  Good luck!
120                     keep responses short, no one has a long attention
                        ↪  span.
121                     Do not use unprenounce-able markup, you're words are
                        ↪  being spoken not read.
122                     be direct and a little bit abbrasive, take inspiration
                        ↪  from famous dnd GM's such as mathew mercer in the
                        ↪  way he sets up scenarios and plays with the
                        ↪  players.
123                     KEEP IT SHORT!
```

```
124
125                     example message:
126                     (user rolls poorly on an attack roll):
127                     "You attack the goblin with your sword, Finn screams
                        ↪  'WATCH OUT FOR THE ROCK!' as you swing your sword,
                        ↪  but you miss the goblin and get hit in the head
                        ↪  with a rock, you take 2 damage"
128                     finn then goes in for the counter attack and rolls a
                        ↪  15, he hits the goblin and deals 5 damage, the
                        ↪  goblin is looking pretty rough now.
129
130                     another example:
131                     (user rolls for initiative and gave instructions to
                        ↪  attack the goblin with a pastry in towards its
                        ↪  face)
132                     "You rolled a nat 20! way to go, with an epic launch
                        ↪  you throw a pastry directly into the goblins face,
                        ↪  it's neck snaps back and it falls to the ground,
                        ↪  dead."
133                     "Finn looks at you and says 'I wanted to eat that!'
                        ↪  while taking advantage of the situation and
                        ↪  casting a firebolt at the other goblin, he rolls a
                        ↪  <make up a good number> and hits the goblin for
                        ↪  <make up a better number> damage."
134                     "The goblin is looking rough and charges in for an
                        ↪  attack, he rolls a <number> for <number> damage.
                        ↪  The goblin heaves a sigh of relief as he lifts up
                        ↪  his club and smashes the shit out of you."
135                     <give a summary of the current state of the world make
                        ↪  it snappy>
136
137                     another example
138                     "You tiptoe carefully, but your foot catches on a
                        ↪  loose cobblestone, sending you tumbling to the
                        ↪  ground with a loud crash. The guard's eyes snap to
                        ↪  you, and he shouts, 'Halt! Who goes there?' as he
                        ↪  draws his sword. You take 3 damage as you scrape
                        ↪  your knee on the rough stone."
139                     "Finn, seeing the situation go sideways, whispers a
                        ↪  spell and fires a bolt of ice at the guard. He
                        ↪  rolls a 17, and it strikes true, freezing the
                        ↪  guard's arm. The guard takes 6 damage and is
                        ↪  slowed. He's now stumbling, struggling to break
                        ↪  free from the ice."
```

```
                    "The street is tense; shadows flicker from torches,
                    ↪  and the sound of distant footsteps suggests more
                    ↪  guards are on the way."

                    DO NOT OFFER THE PLAYER THE OPTION TO PLAY AS FINN THE
                    ↪  TRICKSTER WIZARD THAT IS AN NPC: NON PLAYABLE.
                    KEEP IT SHORT,

                    """
                )
            ),
            # ChatMessage(
            #     role="system",
            #     content="You are in development mode, aid the
            ↪  development of you by helping the user as much as
            ↪  possible. This includes saying exactly what you are doing"
            # ),
            ChatMessage(
                role="assistant",
                content="Ah, Another band of heroes thinking they can
                ↪  handle the ruines Karathun. Lets see what adventures
                ↪  await them... note to self: do not to laugh when they
                ↪  trip over their own sword.",
            ),

        ]
    )
    # gpt = openai.LLM(model="gpt-4o-mini")
    gpt = openai.LLM(model="gpt-4o")


    # Since OpenAI does not support streaming TTS, we'll use it with a
    ↪  StreamAdapter
    # to make it compatible with the VoiceAssistant
    openai_tts = tts.StreamAdapter(
        tts=openai.TTS(voice="echo", speed=1),
        sentence_tokenizer=tokenize.basic.SentenceTokenizer(),
    )

    assistant = VoicePipelineAgent(
        stt=deepgram.STT(),  # We'll use Deepgram's Speech To Text (STT)
        llm=gpt,
        tts=openai_tts,  # We'll use OpenAI's Text To Speech (TTS)
        fnc_ctx=AssistantFnc(),
        chat_ctx=chat_context,
```

```
        allow_interruptions=True,
        vad=silero.VAD.load(
            min_speech_duration=0.1,
            min_silence_duration=0.5,
        )
    )


    chat = rtc.ChatManager(ctx.room)

    async def _answer(text: str):
        """Answer the user's message with the given text."""
        chat_context.messages.append(ChatMessage(role="user",
        ↪  content=text))

        stream = gpt.chat(chat_ctx=chat_context)
        assistant._interrupt_if_possible()
        await assistant.say(stream, allow_interruptions=True)

    @chat.on("message_received")
    def on_message_received(msg: rtc.ChatMessage):
        """This event triggers whenever we get a new message from the
        ↪  user."""
        if msg.message:
            asyncio.create_task(_answer(msg.message))

    @assistant.on("function_calls_finished")
    def on_function_calls_finished(called_functions:
    ↪  list[agents.llm.CalledFunction]):
        """This event triggers when an assistant's function call
        ↪  completes."""
        if len(called_functions) == 0:
            return

        logger.info(f"Function call completed:result:
        ↪  {called_functions[0].result}")

        user_msg = called_functions[0].call_info.arguments.get("user_msg")
        if user_msg:
            asyncio.create_task(_answer(user_msg))

    assistant.start(ctx.room)

    # Start the HTTP server in its own background task
```

```
    server_task = asyncio.create_task(run_http_server(chat_context,
    ↪  _answer))

    try:
        # Main loop: This will run while the connection is active
        while ctx.room.connection_state ==
        ↪  rtc.ConnectionState.CONN_CONNECTED:
            await asyncio.sleep(1)
    finally:
        # Optional: If you want to shut down the server once the room is
        ↪  disconnected
        server_task.cancel()
        with contextlib.suppress(asyncio.CancelledError):
            await server_task


if __name__ == "__main__":
    cli.run_app(WorkerOptions(entrypoint_fnc=entrypoint))
```

Appendix E

Data from the Questionnaire

ID	How familiar are you with TTRPGs	Have you ever played Dungeons and Dragons before?	amount of sessions of D&D	Have you ever acted as a GM?	I think that I would like to use this system frequently.
1	Not at all	No		No	Strongly disagree
2	Slightly	No		No	Agree
3	Slightly	Yes	1-5	No	Agree
4 - Expert 1	expert	Yes	20+	Yes	Disagree
5	Not at all	No		No	Agree
6	Slightly	No		No	Agree
7	Slightly	Yes	1-5	No	Neither agree nor disagree
8	Slightly	Yes	1-5	No	Agree
9	Very	Yes	1-5	No	Agree
10	Slightly	Yes	1-5	No	Agree
11	Moderately	Yes	1-5	No	Agree
12	Slightly	Yes	6-10	No	Agree
13-E2	expert	Yes	20+	Yes	Agree
14	Not at all	No		No	Agree
15	Not at all	No		No	Neither agree nor disagree
16	Not at all	No		No	Agree
17	Not at all	No		No	Agree
18	Moderately	Yes	1-5	No	Agree
19	Moderately	Yes	11-20	Yes	Agree
20	Moderately	Yes	11-20	Yes	Neither agree nor disagree

I found the system unnecessarily complex.	I thought the system was easy to use.	I think that I would need the support of a technical person to be able to use this system.	I found the various functions in this system were well-integrated.
Disagree	Strongly agree	Strongly disagree	Neither agree nor disagree
Strongly disagree	Strongly agree	Strongly disagree	Agree
Disagree	Agree	Disagree	Agree
Strongly disagree	Disagree	Disagree	Agree
Strongly disagree	Strongly agree	Strongly disagree	Agree
Disagree	Neither agree nor disagree	Agree	Agree
Strongly disagree	Strongly agree	Strongly disagree	Strongly agree
Strongly disagree	Strongly agree	Strongly disagree	Strongly agree
Strongly disagree	Agree	Neither agree nor disagree	Strongly agree
Disagree	Strongly agree	Disagree	Agree
Strongly disagree	Agree	Disagree	Strongly agree
Strongly disagree	Strongly agree	Strongly disagree	Agree
Strongly disagree	Strongly agree	Strongly disagree	Agree
Strongly disagree	Strongly agree	Disagree	Agree
Disagree	Strongly agree	Strongly disagree	Agree
Strongly disagree	Strongly agree	Strongly disagree	Strongly agree
Strongly disagree	Agree	Disagree	Strongly agree
Disagree	Strongly agree	Disagree	Strongly agree
Disagree	Strongly agree	Strongly disagree	Agree
Strongly disagree	Strongly agree	Disagree	Agree

I thought there was too much inconsistency in this system.	I would imagine that most people would learn to use this system very quickly.	I found the system very cumbersome to use.	I felt very confident using the system.
Disagree	Agree	Disagree	Strongly agree
Strongly disagree	Strongly agree	Strongly disagree	Strongly agree
Neither agree nor disagree	Strongly agree	Strongly disagree	Agree
Disagree	Agree	Disagree	Neither agree nor disagree
Strongly disagree	Strongly agree	Disagree	Agree
Neither agree nor disagree	Agree	Neither agree nor disagree	Agree
Disagree	Agree	Strongly disagree	Agree
Strongly disagree	Strongly agree	Strongly disagree	Strongly agree
Disagree	Strongly agree	Strongly disagree	Agree
Disagree	Disagree	Disagree	Disagree
Strongly disagree	Strongly agree	Strongly disagree	Agree
Disagree	Strongly agree	Strongly disagree	Strongly agree
Disagree	Agree	Strongly disagree	Strongly agree
Strongly disagree	Strongly agree	Disagree	Neither agree nor disagree
Disagree	Strongly agree	Disagree	Agree
Strongly disagree	Agree	Strongly disagree	Agree
Disagree	Strongly agree	Agree	Agree
Strongly disagree	Agree	Strongly disagree	Agree
Neither agree nor disagree	Strongly agree	Strongly disagree	Strongly agree
Strongly disagree	Strongly agree	Strongly disagree	Disagree

I needed to learn a lot of things before I could get going with this system.	I lost myself in this experience.	The time I spent using the AI GM just slipped away.	I felt frustrated while using the AI GM.
Neither agree nor disagree	Agree	Disagree	Strongly Disagree
Strongly disagree	Strongly Agree	Strongly Agree	Strongly Disagree
Strongly disagree	Disagree	Strongly Agree	Neither agree nor disagree
Strongly disagree	Neither agree nor disagree	Agree	Disagree
Strongly disagree	Disagree	Agree	Disagree
Disagree	Neither agree nor disagree	Neither agree nor disagree	Agree
Strongly disagree	Agree	Agree	Disagree
Strongly disagree	Agree	Strongly Agree	Disagree
Strongly disagree	Agree	Agree	Disagree
Disagree	Agree	Neither agree nor disagree	Strongly Disagree
Strongly disagree	Agree	Agree	Disagree
Strongly disagree	Agree	Neither agree nor disagree	Disagree
Disagree	Strongly Agree	Strongly Agree	Strongly Disagree
Disagree	Disagree	Disagree	Strongly Disagree
Strongly disagree	Neither agree nor disagree	Agree	Strongly Disagree
Strongly disagree	Agree	Agree	Strongly Disagree
Strongly disagree	Strongly Disagree	Neither agree nor disagree	Strongly Disagree
Strongly disagree	Strongly Agree	Agree	Strongly Disagree
Strongly disagree	Strongly Agree	Strongly Agree	Disagree
Strongly disagree	Disagree	Agree	Neither agree nor disagree

I found this the AI GM confusing to use.	Using the AI GM was taxing	This the AI GM was attractive	This the AI GM was aesthetically appealing
Strongly Disagree	Disagree	Agree	Disagree
Strongly Disagree	Strongly Disagree	Agree	Agree
Disagree	Disagree	Agree	Strongly Agree
Disagree	Strongly Disagree	Agree	Neither agree nor disagree
Strongly Disagree	Disagree	Agree	Neither agree nor disagree
Neither agree nor disagree	Disagree	Agree	Neither agree nor disagree
Strongly Disagree	Strongly Disagree	Agree	Agree
Strongly Disagree	Strongly Disagree	Strongly Agree	Agree
Strongly Disagree	Disagree	Strongly Agree	Neither agree nor disagree
Disagree	Disagree	Neither agree nor disagree	Disagree
Disagree	Strongly Disagree	Agree	Agree
Strongly Disagree	Strongly Disagree	Neither agree nor disagree	Neither agree nor disagree
Strongly Disagree	Strongly Disagree	Agree	Strongly Agree
Strongly Disagree	Strongly Disagree	Neither agree nor disagree	Neither agree nor disagree
Strongly Disagree	Disagree	Agree	Agree
Strongly Disagree	Strongly Disagree	Strongly Agree	Neither agree nor disagree
Strongly Disagree	Strongly Disagree	Agree	Agree
Strongly Disagree	Strongly Disagree	Agree	Neither agree nor disagree
Strongly Disagree	Strongly Disagree	Strongly Agree	Neither agree nor disagree
Strongly Disagree	Agree	Agree	Neither agree nor disagree

The AI GM appealed to be visual senses.	Using the AI GM was worthwhile	My experience was rewarding.
Strongly Disagree	Strongly Agree	Agree
Agree	Agree	Strongly Agree
Agree	Agree	Agree
Agree	Agree	Strongly Agree
Neither agree nor disagree	Strongly Agree	Agree
Neither agree nor disagree	Agree	Agree
Agree	Strongly Agree	Strongly Agree
Neither agree nor disagree	Strongly Agree	Strongly Agree
Disagree	Agree	Agree
Disagree	Agree	Strongly Agree
Neither agree nor disagree	Agree	Agree
Agree	Agree	Strongly Agree
Agree	Strongly Agree	Strongly Agree
Neither agree nor disagree	Agree	Agree
Agree	Strongly Agree	Agree
Neither agree nor disagree	Strongly Agree	Strongly Agree
Disagree	Agree	Agree
Neither agree nor disagree	Agree	Agree
Neither agree nor disagree	Strongly Agree	Strongly Agree
Agree	Agree	Agree

What did you like most about using the system?

I could say anything and it would give me an elaborate answer. I could ask for more explanation without feeling like a burden. I liked the answers it gave, it was magical and surprising and it gave me a lot of encouragement. I liked that you could both chose for listening and reading, which makes it accessible for more people

I liked how it gave me the freedom to change something, if I did not like it.

It worked pretty well, and the story is nicely given

Absolute freedom to do all sorts of stupid shit

It was easy to use for beginners that have never played DandD and it guided you nicely

It provided a nice, interesting story I could interact with

The GM responded quickly, and was able to react to even some more random and obscure inputs.

that i could speak and did not have to type. it added a lot to the experience.

The way it created a story was very cool and the way it could incorporate randomness like the random spell I came up with.

I have not much experience playing DnD so I wasn't sure what I could do, but the system guided me nicely through everything and gave me options and inspiration for actions I could do. It was relatively easy to use and it was fun to play.

I enjoyed the interaction with the ai, being able to speak to it gives a similar experience as with regular DnD. I like the fact that the AI has a smooth voice and is consistent.

That I didn't have to type anything and I could physically throw a die

I loved my little goblin (kobold) squad, I also really like how responsive the AI GM was to my inputs

Although i did not fully understand my character and its purposes i was able to get a story which was to my liking. Adapted to what I told the AI what I wanted instead of adapted to the designated character.

That it read aloud while I could read the text.

The fact that it felt unique, and not like it was scripted. The AI actually used my input and made it feel like I had a lot of influence on the storyline. The dice was also a nice addition.

It has so much room for creativity and you could play for hours and create your own world where you could even progress without your friends(As id real life D&D)

The fact that the AI GM could come up with new adventures, side missions, puzzles and enemies on the go so its basically a endless game.

I like that I can really come up with things to do, and the AI will be able to structure DMing around it properly. It really felt like I was free to do anything, which traditional digital versions of TTRPGs lack (I.e. Divinity 2 original sin, text adventures, boldurs gate, etc.)

I was surprised at the flexibility of the AI GM and the amount of things it remembered from previous actions. Trying different weird actions to see how the AI responded was the most fun thing to do in my opinion.

What were the challenges you faced while using the system?

The audio lagged a bit, sometimes it was faster than the text wat could be irritating.

Sometimes the words were complex.

Sometimes the voice was a robot and therefore was behind the text, so I had to wait for the voice to be done

Didn't ask for any dice rolls after a while

i was confused about what exactly i needed to do at some points , maybe a short explanation on whats going on would be nice for those who have never played the game

The dice broke, and it briefly stopped at one point.

The die gave the wrong output once, but you are able to manually input the die roll by telling the AI GM

one time i misspoke or the master misheard me and then i tried typing but was too slow and it caused a bit of confusion and skipping in the storyline.

Rolling the dice as it sometimes did not know what I threw. And I could not talk during the dialogue since it would stop talking to interpret what I said.

I could not always remember all the information I got, so I had to go back. And this might be a lack of experience, but at times that there was no guidance I didn't know what the boundaries were. Could I ask anything? But everything I asked was smoothly resolved during this sessions. Also I felt I had no context on who I was and what I had or could do, but maybe I should have asked that to the system.

I did not experience any challenges during the game

It had a little trouble understanding the word orc and related words.

I got sidetracked and the GM didnt remind me of the orignal purpose of the quest

I also spoke when the AI was not supposed to hear me, which seemed to confuse it which in turn confused me.

Sometimes the speaking was before the text and then it was more difficult to use (I also watch series with subtitles and without I just get more lost)

The AI didn't catch my name correctly at the start, but this was okay because he changed it into a pretty sick name anyways.

Not being sure when to roll exactly, it wasn't always 100% clear that i should

None, It just worked and i didnt find any big glitches. The only glitch i found was when I said lets het back to the entrance and the voice to text heared lets head back to Claire Meyers, but the AI just took this name and named the quest giver Claire Meyers. I love how the AI could just take what I say and adapt the story accordingly

Minor challenge: it would be nice to know when it is registering a dice roll; Sometimes it took long to notice that it had in fact registered what I just rolled (no roll was registered incorrectly though, so it still worked flawlessly).

I felt like I had to be very quiet as to not disrupt the GM, which meant that I also was not really sure if laughing would also cause a disruption which was a bit of a shame.

If you could improve one thing about the system what would it be?

I would like to just see an (maybe AI generated) picture of my character, that way I feel more connected with the role I am playing. I would make it a bit prettier, but I am not sure that I am the right user group but maybe in the same style as DND?

Simpler words.

Fix the voice thingy

General information about the characters being played and limits on what they can do

maybe a short explanation on whats going on would be nice for those who have never played the game

A more consistent dice

Visuals of the characters or environment in the display

i would like the AI not to pick up my giggles

Add visuals to the story.

Maybe something visual. Now it is just text, and I can imagine something like a map would give inspiration and insight into what is possible. And at the moment I felt like the dice was very fragile, like i had to handle it with care. This stopped me from really rolling the dice like I would normally do.

The voice occasionally glitched out but aside from that it all went well

I feel like I skipped out or missed some puzzles or story steps. So maybe a bit more restriction on the AI that certain tasks need be fulfilled

Maybe have Fin every now an then get you to go to the right side of the system

Maybe more integration if the dice as i only rolled it twice. Some choices seemed to need the addition of a roll but instead were just accepted.

Make sure the text and the speaking is at the same tempo.

Maybe adding AI generated images based on the story to make it more aesthetically pleasing.

Maybe they ai could talk a bit less, some stuff were extra to add a bit of depthness into the story which i get, but some times the AI went on for a while

Sound effect and background sounds. This is would be a hard feature to add but just having a pool of 20 generic background tracks like forest, snow storm, cave, etc. and the AI pick on of those to play based on what is going on would add so much immersion

The AI should keep balancing in mind more. I was able to use very very very late game spells to "cheese" the puzzles just by rolling a high number. It might help if the AI is instructed to only allow the player to use spells/ items that Wth level character of class and race X and Y would have.
Because I'm annoying, a second thing to improve: I answered neutral for the aesthetic questions because I feel that it could maybe have a little bit more of a "DnD" feel (i.e. the text is all written on a burned paper background, a more RPG-like font, etc.)

In the DnD I am used to the GM usually states a problem without a solution since that is up to the players to solve, I felt like the GM AI would push me toward certain actions more then perhaps necessary.

Any other remarks?

The emoji do add something, but I would prefer them to be incorporated into a DND style as well, but I understand that for the user testing it can be a bit out of scope.

Visual of the characters might be cool and even sceneries. AMAZING WORK!

Nope

The AI stories were too long in my opinion

No! :)

i really liked the experience and it was fun to go through the story. Also the AI only said the number i rolled once, it would be cool if it had like a reaction so it would be like: oof a two, you try to listen to bla bla bla. Or like: Wow a natural 20! bla bla bla

I like the product and would buy a more refined verions of such a system.

It was fun:) And I'm not even a fan of games.

I really enjoyed the experience, it is a great way to experience DnD

Was fun!

Loved my kobold squad

very cool!

It was very nice. Eventhough I do not know D&D I did like it.

Really fun experience, I'm impressed!

Nope, I loved it!

Really fun to play with! I really saw myself trying stupid things as I would in a real game of DnD and laughing along with it. If it was more balanced, I could really see myself playing with this for several hours on end.

This was a very cool experience!

SUS score	FA (Focused Attention)	PU (Percieved Usability)	AE (Aesthetic Appeal)	RW (Reward Factor)	Final Score
70	3	4.666666667	2.333333333	4.5	3.6
95	5	5	4	4.5	4.6
80	3.5	3.666666667	4.333333333	4	3.9
67.5	3.5	4.333333333	3.666666667	4.5	4
90	3	4.333333333	3.333333333	4.5	3.8
62.5	3	3	3.333333333	4	3.3
87.5	4	4.666666667	4	5	4.4
97.5	4.5	4.666666667	4	5	4.5
85	4	4.333333333	3.333333333	4	3.9
67.5	3.5	4.333333333	2.333333333	4.5	3.6
90	4	4.333333333	3.666666667	4	4
92.5	3.5	4.666666667	3.333333333	4.5	4
87.5	5	5	4.333333333	5	4.8
82.5	2	5	3	4	3.6
82.5	3.5	4.666666667	4	4.5	4.2
92.5	4	5	3.666666667	5	4.4
80	2	5	3.333333333	4	3.7
87.5	4.5	5	3.333333333	4	4.2
87.5	5	4.666666667	3.666666667	5	4.5
82.5	3	3.333333333	3.666666667	4	3.5

Appendix F

Programs, Front-End

The front-end program for the web based UI that shows the AI GM

```
    // src/app/page.tsx
"use client";
import React, { useState, useEffect, useMemo, useRef } from "react";
import {
  LiveKitRoom,
  RoomAudioRenderer,
  BarVisualizer,
  ControlBar,
  ControlBarControls,
  useVoiceAssistant,
  useMaybeRoomContext,
  useChat,
} from "@livekit/components-react";
import "@livekit/components-styles";
import { RoomEvent, TranscriptionSegment } from "livekit-client";

export default function HomePage() {
  const [token, setToken] = useState<string | null>(null);
  const [url, setUrl] = useState<string | null>(null);

  const connectToRoom = async () => {
    const response = await fetch("/api/token");
    if (!response.ok) {
      console.error("Failed to fetch token");
      return;
    }
    const { accessToken, url } = await response.json();
    setToken(accessToken);
    setUrl(url);
  };
```

```
31
32    return (
33      <>
34        <main className="w-full h-full flex items-center justify-center">
35          {token === null ? (
36            <button
37              onClick={connectToRoom}
38              className="bg-gradient-to-r from-blue-500 to-blue-700
                ↪  hover:from-blue-700 hover:to-blue-500 text-white font-bold
                ↪  py-2 px-4 rounded-full shadow transition-colors
                ↪  duration-300"
39            >
40              Connect
41            </button>
42          ) : (
43            <LiveKitRoom
44              token={token}
45              serverUrl={url}
46              connectOptions={{ autoSubscribe: true }}
47              data-lk-theme="default"
48              className="h-full grid place-items-center"
49            >
50              <ActiveRoom />
51            </LiveKitRoom>
52          )}
53        </main>
54      </>
55    );
56  }
57
58  const ActiveRoom = () => {
59    const controlBarOptions: ControlBarControls = {
60      microphone: true,
61      camera: false,
62      chat: false,
63      screenShare: false,
64      leave: true,
65      settings: false,
66    };
67
68    const { state, audioTrack } = useVoiceAssistant();
69
70    const room = useMaybeRoomContext();
71
72    const { chatMessages, send } = useChat();
```

```tsx

  const [draft, setDraft] = useState("");
  const [messagesMap, setMessagesMap] = useState<
    Map<string, { sender: string; text: string; timestamp: number }>
  >(new Map());

  // Handle transcription events and update messages
  useEffect(() => {
    if (!room) {
      return;
    }

    const updateTranscriptions = (
      segments: TranscriptionSegment[],
      participant: any,
      publication: any
    ) => {
      const participantName = participant.name || participant.identity;

      segments.forEach((segment) => {
        const messageId =
          `transcription-${participant.identity}-${segment.id}`;

        setMessagesMap((prev) => {
          const newMap = new Map(prev);

          const timestamp = (segment.startTime || Date.now() / 1000) *
            1000; // Convert to ms

          newMap.set(messageId, {
            sender: participantName,
            text: segment.text,
            timestamp,
          });

          return newMap;
        });
      });
    };

    room.on(RoomEvent.TranscriptionReceived, updateTranscriptions);

    return () => {
      room.off(RoomEvent.TranscriptionReceived, updateTranscriptions);
    };
```

```
116   }, [room]);
117
118   // Handle chat messages and add them to messagesMap
119   useEffect(() => {
120     setMessagesMap((prev) => {
121       const newMap = new Map(prev);
122       chatMessages.forEach((msg) => {
123         const messageId = `chat-${msg.timestamp}-${msg.from?.identity}`;
124         if (!newMap.has(messageId)) {
125           newMap.set(messageId, {
126             sender: msg.from?.identity || "Unknown",
127             text: msg.message,
128             timestamp: msg.timestamp,
129           });
130         }
131       });
132       return newMap;
133     });
134   }, [chatMessages]);
135
136   const messages = useMemo(() => {
137     return Array.from(messagesMap.values()).sort(
138       (a, b) => a.timestamp - b.timestamp
139     );
140   }, [messagesMap]);
141
142   const onSend = async () => {
143     if (draft.trim().length && send) {
144       await send(draft);
145       setDraft("");
146     }
147   };
148
149   // Function to determine if the username should be displayed
150   // Function to determine if the username should be displayed
151   const shouldDisplayUsername = (
152     currentMsg: { sender: any; timestamp: string | number | Date },
153     previousMsg: { sender: any; timestamp: string | number | Date }
154   ) => {
155     // Always show if there's no previous message
156     if (!previousMsg) return true;
157
158     // Check if the current message is from a different sender than the
      ↪  previous message
159     if (currentMsg.sender !== previousMsg.sender) return true;
```

```
160
161     // Check if messages are spaced more than a minute apart
162     const currentTime = new Date(currentMsg.timestamp).getTime();
163     const previousTime = new Date(previousMsg.timestamp).getTime();
164     return currentTime - previousTime > 60000; // 60000 ms = 1 minute
165   };
166   const messagesEndRef = useRef<HTMLDivElement | null>(null); // For
      ↪  automatic scrolling
167
168   // Auto-scroll to the latest message
169   useEffect(() => {
170     messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
171   }, [messages]);
172
173   return (
174     <>
175       <RoomAudioRenderer />
176       <div className="flex flex-col w-full max-w-2xl min-w-[300px] mx-auto
        ↪  h-[80vh] border p-4 bg-stone-950 shadow-lg">
177         <div className="flex-1 overflow-y-auto mt-2 p-2 space-y-1
          ↪  scrollbar-thin">
178           {messages.map((msg, index) => (
179             <div
180               key={index}
181               className={`flex flex-col \${
182                 msg.sender === room?.localParticipant?.identity
183                   ? "items-end"
184                   : "items-start"
185               }`}
186             >
187               {shouldDisplayUsername(msg, messages[index - 1]) && (
188                 <div className="text-sm text-gray-600">
189                   {msg.sender === room?.localParticipant?.identity
190                     ? "you"
191                     : "AI - GM"}
192                 </div>
193               )}
194               <div
195                 className={`max-w-[75%] p-2.5 rounded-lg text-white ${
196                   msg.sender === room?.localParticipant?.identity
197                     ? "bg-blue-600" // Your messages color
198                     : "bg-gray-700" // Other people's messages color
199                 }`}
200               >
201                 <span>{msg.text}</span>
```

```
              </div>
            </div>
          ))}
          <div ref={messagesEndRef} />
        </div>
        <div className="flex border-t pt-2 mt-2">
          <input
            type="text"
            placeholder="Type a message..."
            value={draft}
            onChange={(e) => setDraft(e.target.value)}
            onKeyUp={(e) => {
              if (e.key === "Enter") {
                e.preventDefault();
                onSend();
              }
            }}
            className="flex-1 p-2 border-gray-300 focus:outline-none"
          />
          <button
            onClick={onSend}
            disabled={!draft.trim()}
            className="bg-blue-500 hover:bg-blue-700 text-white font-bold
             ↪  py-2 px-4 rounded disabled:opacity-50"
          >
            Send
          </button>
        </div>
      </div>
      <div style={{ width: "100%", height: "60px" }}>
        <BarVisualizer
          state={state}
          trackRef={audioTrack}
          barCount={5}
          data-lk-theme="default"
        />
      </div>
      <ControlBar controls={controlBarOptions} saveUserChoices={true} />
    </>
  );
};
```

Appendix G

Programs, Token Route

The route for retriving the token to connect the LiveKit AI to the LiveKit room.

```
import { AccessToken } from "livekit-server-sdk";

export async function GET(request: Request) {
  const roomName = Math.random().toString(36).substring(7);
  const apiKey = process.env.LIVEKIT_API_KEY;
  const apiSecret = process.env.LIVEKIT_API_SECRET;
  const at = new AccessToken(apiKey, apiSecret, {
    identity: "human_user",
  });
  at.addGrant({
    room: roomName,
    roomJoin: true,
    canPublish: true,
    canPublishData: true,
    canSubscribe: true,
  });
  return Response.json({
    accessToken: await at.toJwt(),
    url: process.env.LIVEKIT_URL,
  });
}
```

Appendix H

Programs, D20 itself

Arduino program that runs the esp32

```
#include <Wire.h>
#include "esp_sleep.h"  // Include the ESP32 sleep library
#include <esp_now.h>
#include <WiFi.h>

// ----------------- Define MPU6050 Registers -----------------
#define SIGNAL_PATH_RESET 0x68
#define I2C_SLV0_ADDR 0x37
#define ACCEL_CONFIG 0x1C
#define MOT_THR 0x1F
#define MOT_DUR 0x20
#define MOT_DETECT_CTRL 0x69
#define INT_ENABLE 0x38
#define INT_STATUS 0x3A
#define WHO_AM_I_MPU6050 0x75  // WHO_AM_I register address
#define MPU6050_ADDRESS 0x68


// ---------------------- dice settings ----------------------
#define SLEEP_DELAY 2000  // Time in milliseconds to wait before going to
↪  sleep

#define INTERRUPT_PIN 3

unsigned long lastMotionTime = 0;

struct Vector3 {
  float x;
  float y;
  float z;
```

```
};

const int numFaces = 20;

// Define vectors representing each face of the dice
Vector3 diceFaces[20] = {
  { -0.02, 1, 0.32 },       // Face 1 (Z-axis up)
  { 0.61, -0.69, -0.22 },   // Face 2 (Y-axis up)
  { -0.41, 0.21, 1.07 },    // Face 3 (Y-axis up)
  { -0.30, -0.20, -0.76 },  // Face 4 (Y-axis up)
  { 0.62, 0.43, -0.50 },    // Face 5 (Y-axis up)
  { -0.94, -0.34, 0.20 },   // Face 6 (Y-axis up)
  { 0.56, 0.70, 0.63 },     // Face 7 (Y-axis up)
  { -0.02, -0.82, 0.74 },   // Face 8 (Y-axis up)
  { -0.94, 0.39, 0.09 },    // Face 9 (Y-axis up)
  { 0.55, -0.45, 0.87 },    // Face 10 (Y-axis up)
  { -0.57, 0.45, -0.54 },   // Face 11 (Y-axis up)
  { 0.93, -0.35, 0.27 },    // Face 12 (Y-axis up)
  { 0.03, 0.86, -0.37 },    // Face 13 (Y-axis up)
  { -0.58, -0.66, -0.30 },  // Face 14 (Y-axis up)
  { 0.93, 0.38, 0.13 },     // Face 15 (Y-axis up)
  { -0.61, -0.42, 0.85 },   // Face 16 (Y-axis up)
  { 0.35, 0.23, 1.08 },     // Face 17 (Y-axis up)
  { 0.41, -0.17, -0.75 },   // Face 18 (Y-axis up)
  { -0.58, 0.72, 0.58 },    // Face 19 (Y-axis up)
  { 0, -0.98, 0 },          // Face 20 (Y-axis up)
};

// -------------------------ESP_NOW-------------------------
// Define the broadcast address (FF:FF:FF:FF:FF:FF) to send to all ESP-NOW
↪  devices
// adress                          08:3a:f2:b8:11:24
uint8_t broadcastAddress[] = { 0x08, 0x3a, 0xf2, 0xb8, 0x11, 0x24 };

// Structure to hold dice data
typedef struct struct_message {
  uint8_t diceID;
  uint8_t roll;
  uint8_t maxNumber;
} struct_message;

struct_message diceData = { 1, 0, 20 };  // Initialize with default values

// Peer info structure
esp_now_peer_info_t peerInfo;
```

```
74
75 void setup() {
76   Serial.begin(115200);
77   analogReadResolution(12);  // Set the resolution to 12 bits (0-4095)
78
79   setupEspNow();  //Setup ESP_NOW
80
81   configMPU();  //setup and configure the MPU6050
82   Serial.println();
83   Serial.println("MPU6050 initialized!");
84
85   // Initialize the last motion time
86   lastMotionTime = millis();
87 }
88
89 void setupEspNow() {
90   // Set device as a Wi-Fi Station
91   WiFi.mode(WIFI_STA);
92
93   // Initialize ESP-NOW
94   if (esp_now_init() != ESP_OK) {
95     Serial.println("Error initializing ESP-NOW");
96     return;
97   }
98
99   memcpy(peerInfo.peer_addr, broadcastAddress, 6);
100  peerInfo.channel = 0;
101  peerInfo.encrypt = false;
102
103  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
104    Serial.println("Failed to add peer");
105    return;
106  }
107
108  esp_now_register_send_cb(OnDataSent);
109 }
110
111 void loop() {
112
113  // Read the INT_STATUS register to check and clear the interrupt flag
114  uint8_t intStatus = readByte(MPU6050_ADDRESS, INT_STATUS);
115
116  // Check if the motion detection interrupt bit (bit 6) is set
117  if (intStatus & 0x40) {       // Bit 6 corresponds to motion detection
     ↪  interrupt
```

```
    lastMotionTime = millis();  // Reset the timer if motion was detected
    Serial.println("Motion detected, timer reset.");
  }

  // If no motion has been detected for the SLEEP_DELAY period, go to
  ↪  sleep
  if (millis() - lastMotionTime > SLEEP_DELAY) {
    HandleAccelAndComms();
  }

  delay(100);  // Small delay for loop stability
}

void HandleAccelAndComms() {
  // Get accelerometer vector
  Serial.print("Accel Vector: ");
  Vector3 accelVector = getAccelerometerVector();
  logVector(accelVector);

  // Determine the face
  int face = getDieFace(accelVector);
  Serial.print("Detected Die Face: ");
  Serial.println(face);
  sendFace(face);
}

void sendFace(int face) {
  diceData.roll = face;
  Serial.println("Start sending...");

  esp_now_send(broadcastAddress, (uint8_t *)&diceData, sizeof(diceData));
  // callback when data is sent
}

void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Succesfull" :
  ↪  "Unsuccesfull");
  Serial.print("MAC Address: ");
  if (mac_addr != NULL) {
    for (int i = 0; i < 6; i++) {
      Serial.printf("%02X", mac_addr[i]);
      if (i < 5) {
        Serial.print(":");
      }
```

```
    }
    Serial.println();
  } else {
    Serial.println("MAC Address not available");
  }

  if (status == ESP_NOW_SEND_SUCCESS) {
    goToSleep();
  } else{
    delay(200);
    esp_now_send(broadcastAddress, (uint8_t *)&diceData,
    ↪  sizeof(diceData));
  }
}


// Function to read the accelerometer data and return it as a vector
Vector3 getAccelerometerVector() {
  int16_t accelX = (readByte(MPU6050_ADDRESS, 0x3B) << 8) |
  ↪  readByte(MPU6050_ADDRESS, 0x3C);
  int16_t accelY = (readByte(MPU6050_ADDRESS, 0x3D) << 8) |
  ↪  readByte(MPU6050_ADDRESS, 0x3E);
  int16_t accelZ = (readByte(MPU6050_ADDRESS, 0x3F) << 8) |
  ↪  readByte(MPU6050_ADDRESS, 0x40);

  // Convert raw values to g's (gravity units) based on the accelerometer
  ↪  sensitivity
  const float accelScale = 16384.0;  // Assuming the sensitivity is set to
  ↪  +/- 2g (scale factor is 16384 LSB/g)
  Vector3 accelVector;
  accelVector.x = (float)accelX / accelScale;
  accelVector.y = (float)accelY / accelScale;
  accelVector.z = (float)accelZ / accelScale;

  return accelVector;
}

// Function to determine which face of the die is facing up
int getDieFace(Vector3 accelVector) {
  int closestFace = 0;
  float maxDotProduct = -1.0;

  // Normalize the accelerometer vector (make its length 1)
  float magnitude = sqrt(accelVector.x * accelVector.x + accelVector.y *
  ↪  accelVector.y + accelVector.z * accelVector.z);
```

```
  accelVector.x /= magnitude;
  accelVector.y /= magnitude;
  accelVector.z /= magnitude;

  for (int i = 0; i < numFaces; i++) {
    // Calculate dot product between accelVector and each face vector
    float dotProduct = accelVector.x * diceFaces[i].x + accelVector.y *
    ↪  diceFaces[i].y + accelVector.z * diceFaces[i].z;
    if (dotProduct > maxDotProduct) {
      maxDotProduct = dotProduct;
      closestFace = i + 1;  // Face numbers are 1-indexed
    }
  }

  return closestFace;
}

void logVector(Vector3 vector) {
  Serial.print("X=");
  Serial.print(vector.x);
  Serial.print(", Y=");
  Serial.print(vector.y);
  Serial.print(", Z=");
  Serial.println(vector.z);
}

void goToSleep() {
  esp_deep_sleep_enable_gpio_wakeup(1 << INTERRUPT_PIN,
  ↪  ESP_GPIO_WAKEUP_GPIO_HIGH);
  esp_deep_sleep_start();
}

// Function to write a byte to an MPU6050 register
void writeByte(uint8_t address, uint8_t subAddress, uint8_t data) {
  Wire.beginTransmission(address);
  Wire.write(subAddress);
  Wire.write(data);
  Wire.endTransmission();
}

// Function to read a byte from an MPU6050 register
uint8_t readByte(uint8_t address, uint8_t subAddress) {
  uint8_t data;
  Wire.beginTransmission(address);
  Wire.write(subAddress);
```

```
242   Wire.endTransmission(false);
243   Wire.requestFrom(address, (uint8_t)1);
244   if (Wire.available()) {
245     data = Wire.read();
246   }
247   return data;
248 }
249
250 void configMPU() {
251   Wire.begin(6, 7);                                                    //
      ↪  Initialize I2C on GPIO 6 (SDA) and GPIO 7 (SCL)
252   writeByte(MPU6050_ADDRESS, 0x6B, 0x80);                              // Reset
      ↪  device
253   delay(100);                                                          // Wait
      ↪  for the reset to take effect
254   writeByte(MPU6050_ADDRESS, 0x6B, 0x00);                              // Wake
      ↪  up MPU6050
255   delay(10);                                                           // Give
      ↪  the MPU6050 time to wake up
256   uint8_t whoAmI = readByte(MPU6050_ADDRESS, WHO_AM_I_MPU6050);  // Check
      ↪  WHO_AM_I register to verify connection
257
258   Serial.print("WHO_AM_I register: 0x");
259   Serial.println(whoAmI, HEX);
260   if (whoAmI != 0x68) {
261     Serial.println("MPU6050 not detected! Check your wiring.");
262     // THIS IS NOT GOOD, MAKE IT WAIT AND RETRY!
263     while (1)
264       ;  // Halt the program if MPU6050 is not detected
265   }
266
267   writeByte(MPU6050_ADDRESS, SIGNAL_PATH_RESET, 0x07);  // Reset signal
      ↪  paths
268   delay(10);
269   // Set accelerometer config (Digital High Pass Filter)
270   writeByte(MPU6050_ADDRESS, ACCEL_CONFIG, 0x01);
271   delay(10);
272   uint8_t accelConfig = readByte(MPU6050_ADDRESS, ACCEL_CONFIG);
273   Serial.print("ACCEL_CONFIG register: 0x");
274   Serial.println(accelConfig, HEX);
275
276   // Set motion detection threshold
277   writeByte(MPU6050_ADDRESS, MOT_THR, 5);
278   delay(10);
279   uint8_t motThr = readByte(MPU6050_ADDRESS, MOT_THR);
```

```
  Serial.print("MOT_THR register: 0x");
  Serial.println(motThr, HEX);

  // Set motion detection duration
  writeByte(MPU6050_ADDRESS, MOT_DUR, 20);
  delay(10);
  uint8_t motDur = readByte(MPU6050_ADDRESS, MOT_DUR);
  Serial.print("MOT_DUR register: 0x");
  Serial.println(motDur, HEX);

  // Motion detect control
  writeByte(MPU6050_ADDRESS, MOT_DETECT_CTRL, 0x15);
  delay(10);
  uint8_t motDetectCtrl = readByte(MPU6050_ADDRESS, MOT_DETECT_CTRL);
  Serial.print("MOT_DETECT_CTRL register: 0x");
  Serial.println(motDetectCtrl, HEX);

  // Enable motion interrupt
  writeByte(MPU6050_ADDRESS, INT_ENABLE, 0x40);
  delay(10);
  uint8_t intEnable = readByte(MPU6050_ADDRESS, INT_ENABLE);
  Serial.print("INT_ENABLE register: 0x");
  Serial.println(intEnable, HEX);

  // Configure INT pin as active high
  writeByte(MPU6050_ADDRESS, 0x37, 0x20);
  delay(10);
  uint8_t intPinCfg = readByte(MPU6050_ADDRESS, 0x37);
  Serial.print("INT_PIN_CFG register: 0x");
  Serial.println(intPinCfg, HEX);
}

```

Appendix I

Programs, D20 base-station receiver

Arduino program that runs on the receiver processor of the base station

```
#include <esp_now.h>
#include <WiFi.h>

// Define the data structure to receive
// Must match the sender's structure exactly
typedef struct struct_message {
  uint8_t diceID;     // Dice ID (0-100)
  uint8_t roll;       // Roll result (0-100)
  uint8_t maxNumber;  // Maximum possible number (0-100)
} struct_message;

struct_message receivedData;

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Initialize ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

  // Register the broadcast peer (FF:FF:FF:FF:FF:FF)
  esp_now_peer_info_t peerInfo = {};
  memset(peerInfo.peer_addr, 0xFF, 6);  // Set peer address to broadcast
  peerInfo.channel = 0;                 // Use default channel
```

```
  peerInfo.encrypt = false;

  // Add the broadcast peer
  if (esp_now_add_peer(&peerInfo) != ESP_OK) {
    Serial.println("Failed to add broadcast peer");
    return;
  }

  // Register for a callback function that will be called when data is
  ↪  received
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
}

void loop() {
  // Nothing to do here
}


void OnDataRecv(const uint8_t *mac_addr, const uint8_t *incomingData, int
↪  len) {
  // Copy the received data into our structure
  memcpy(&receivedData, incomingData, sizeof(receivedData));

  // Print to Serial to send the data to the other ESP32
  Serial.print("{\"roll\": ");
  Serial.print(receivedData.roll);
  Serial.println("}");
}
```

Appendix J

Programs, D20 base-station sender to server

Arduino program that sends dice data to the server

```
#include <WiFiManager.h>  // Ensure this library is correctly installed
#include <HTTPClient.h>
#include <Preferences.h>

// Initialize WiFiManager with your desired AP name
WiFiManager wifiManager("Serial-Sender");

// Buffer to store incoming serial data
String serialBuffer = "";

// Initialize HardwareSerial object for UART2 (since UART0 is used by Serial)
HardwareSerial SerialArduino(2);  // Use UART2

// Define the pins for UART2
const int RXD2 = 16;  // ESP32 RX pin (connect to Arduino TX pin)
const int TXD2 = 17;  // ESP32 TX pin (connect to Arduino RX pin)

// Define the server URL where data will be sent
const char* serverUrl = "http://172.20.10.3:8081/dice_roll";  // Replace with your server address

void setup() {
  // Initialize Serial for debugging
  Serial.begin(115200);
  delay(1000);  // Allow time for Serial to initialize
  Serial.println("ESP32 Serial Sender Started");

```

```
  // Initialize SerialArduino on specified RX and TX pins
  SerialArduino.begin(115200, SERIAL_8N1, RXD2, TXD2);
  Serial.println("SerialArduino initialized on pins:");
  Serial.print("RX (ESP32 GPIO");
  Serial.print(RXD2);
  Serial.println(") <-- TX (Arduino)");

  Serial.print("TX (ESP32 GPIO");
  Serial.print(TXD2);
  Serial.println(") --> RX (Arduino)");

  // Start WiFiManager to connect to Wi-Fi
  wifiManager.begin();
  Serial.println("Wi-Fi connected, awaiting serial data to send...");
}

void loop() {
  // Let WiFiManager handle its tasks
  wifiManager.loop();

  // Read incoming serial data from Arduino and store it in the buffer
  while (SerialArduino.available() > 0) {
    char incomingChar = SerialArduino.read();
    serialBuffer += incomingChar;

    // Check for newline to indicate end of line
    if (incomingChar == '\n') {
      // Remove the newline character before sending
      serialBuffer.trim();

      // Send the buffer data if there's new data in serialBuffer
      if (serialBuffer.length() > 0) {
        sendData(serialBuffer);
        serialBuffer = "";  // Clear buffer after sending
      }
    }
  }
}

// Function to send serial data to the server
void sendData(const String& jsonData) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(serverUrl);  // Initialize HTTP client with server URL

```

```
    http.addHeader("Content-Type", "application/json");  // Set content type to JSON

    int httpResponseCode = http.POST(jsonData);  // Send JSON data via POST

    if (httpResponseCode > 0) {
      Serial.print("Data sent successfully. Response code: ");
      Serial.println(httpResponseCode);
    } else if (httpResponseCode < 0) {
      Serial.print("HTTP Error: ");
      Serial.println(http.errorToString(httpResponseCode).c_str());
    } else {
      Serial.print("Error sending data. HTTP response code: ");
      Serial.println(httpResponseCode);
    }

    http.end();  // Close HTTP connection
  } else {
    Serial.println("Wi-Fi not connected. Cannot send data. Restarting");
    ESP.restart();
  }
}
```

wifimanager.h

```
#ifndef WIFIMANAGER_H
#define WIFIMANAGER_H

#include <WiFi.h>
#include <WebServer.h>
#include <DNSServer.h>
#include <Preferences.h>

class WiFiManager {
public:
  WiFiManager(const char* ap_ssid = "ESP32_AP");
  void begin();
  void loop();

private:
  Preferences preferences;
  const char* ap_ssid;
  WebServer server;
  DNSServer dnsServer;
```

```
  bool isAPMode;
  const byte DNS_PORT = 53;

  void setupAPMode();
  void setupCaptivePortal();
  void handleRoot();
  void handleSaveCredentials();
};

#endif  // WIFIMANAGER_H
```

wifimanager.cpp

```
#include "WiFiManager.h"

WiFiManager::WiFiManager(const char* ap_ssid)
  : ap_ssid(ap_ssid), server(80), isAPMode(false) {}

void WiFiManager::begin() {
  Serial.begin(115200);
  delay(1000);

  preferences.begin("wifiCreds", false);  // Open Preferences in RW mode

  String stored_ssid = preferences.getString("ssid", "");
  String stored_password = preferences.getString("password", "");

  if (stored_ssid != "") {
    // Try to connect to stored Wi-Fi network
    WiFi.mode(WIFI_STA);
    WiFi.begin(stored_ssid.c_str(), stored_password.c_str());
    Serial.println("Attempting to connect to WiFi network...");
    unsigned long startAttemptTime = millis();

    // Keep trying to connect for 10 seconds
    while (WiFi.status() != WL_CONNECTED && millis() - startAttemptTime <
    ↪  15000) {
      delay(100);
    }

    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("Connected to WiFi network");
      Serial.print("IP Address: ");
      Serial.println(WiFi.localIP());
      // Proceed with your application logic
```

```
    } else {
      Serial.println("Failed to connect to WiFi network");
      setupAPMode();  // Start captive portal
    }
  } else {
    // No stored credentials, start AP mode
    setupAPMode();
  }
}

void WiFiManager::loop() {
  if (isAPMode) {
    dnsServer.processNextRequest();  // Process DNS requests
    server.handleClient();           // Handle incoming client requests
  } else {
    // Place your code here to run when connected to Wi-Fi
  }
}

void WiFiManager::setupAPMode() {
  isAPMode = true;
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ap_ssid);

  IPAddress IP = WiFi.softAPIP();
  Serial.print("Access Point IP address: ");
  Serial.println(IP);

  dnsServer.start(DNS_PORT, "*", IP);

  setupCaptivePortal();
}

void WiFiManager::setupCaptivePortal() {
  server.on("/", std::bind(&WiFiManager::handleRoot, this));
  ↪ // Handle root URL
  server.on("/save", HTTP_POST,
  ↪ std::bind(&WiFiManager::handleSaveCredentials, this));  // Handle
  ↪ form submission
  server.onNotFound(std::bind(&WiFiManager::handleRoot, this));
  ↪ // Redirect all other URLs
  server.begin();
  Serial.println("Web server started with captive portal");
}

```

```
73 void WiFiManager::handleRoot() {
74   String stored_ssid = preferences.getString("ssid", "");
75   String stored_password = preferences.getString("password", "");
76 
77   String webpage = R"=====(
78         <!DOCTYPE html>
79         <html>
80           <head>
81             <title>Configure Wi-Fi</title>
82             <meta name="viewport" content="width=device-width,
    ↪  initial-scale=1.0">
83           </head>
84           <body style="font-family: Arial; text-align: center;
    ↪  background-color: #f0f0f0; color: #333;">
85             <h1>Configure Wi-Fi</h1>
86             <form action="/save" method="post">
87               <label for="ssid">Wi-Fi Name (SSID):</label><br>
88               <input type="text" id="ssid" name="ssid" value=")=====";
89   webpage += stored_ssid;
90   webpage += R"=====("><br><br>
91               <label for="password">Password:</label><br>
92               <input type="text" id="password" name="password"
    ↪  value=")=====";
93   webpage += stored_password;
94   webpage += R"=====("><br><br>
95               <input type="submit" value="Save and Restart">
96             </form>
97           </body>
98         </html>
99     )=====";
100 
101   server.send(200, "text/html", webpage);
102 }
103 
104 void WiFiManager::handleSaveCredentials() {
105   String ssid = server.arg("ssid");
106   String password = server.arg("password");
107 
108   // Save to preferences
109   preferences.putString("ssid", ssid);
110   preferences.putString("password", password);
111 
112   Serial.println("Credentials saved, restarting...");
113   server.send(200, "text/html", "<html><body><h1>Credentials saved,
    ↪  restarting...</h1></body></html>");
```

```
114   delay(1000);
115   ESP.restart();
116 }
117
```