Sign & Spell VR: ISL Alphabet Practice and Sentence Construction in Virtual Reality Interim Report

TU858
BSc in Computer Science International

Student Name: Karl Negrillo

Student Number: C22386123

Supervisor: Colette Kirwan

School of Computer Science

Technological University, Dublin

Date

Abstract

The Sign & Spell VR project aims to create an immersive virtual reality application designed to facilitate the learning of the Irish Sign Language (ISL). This project addresses the need for accessible and interactive tool for sign language education, focusing specially on ISL, which is less commonly represented in existing digital learning platforms. The application utilizes the Meta Quest 3/3S virtual reality headset and the Godot 4 game engine to provide an engaging and interactive learning environment.

The core functionality of the application includes real-time tracking of user’s hand movements to recognize and assess ISL alphabet letters, words, and simple sentences signed by the user. This features immediate feedback to aid learning effectiveness and user engagement. The project progresses through sequential learning modules that introduce the alphabet, enable word spelling, facilitate sentence construction, and conclude with an interactive testing phase to evaluate the user’s comprehension and retention.

The final deliverable is a functional VR prototype that demonstrates the potential of virtual reality technology in language learning and accessibility. By integrating gesture recognition with VR immersion, the project offers a novel approach to learning ISL, which benefits both learners and the deaf community by promoting effective communication skills. The initiative not only supports language acquisition but also raises awareness about the cultural significance of ISL. The Sign & Spell VR project thus stands as an innovative educational tool with possibilities for further expansion to include additional sign language and more complex linguistic features.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Karl Negrillo____________________

Karl Negrillo

Date

Acknowledgements

I would like to thank my supervisor, Colette Kirwan, for her invaluable guidance and support throughout the course of this project. Her feedback and encouragement helped shape the direction of my work. I also wish to acknowledge the School of Computer Science at Technological University Dublin for providing the resources and environment necessary for completing this report.

1. Contents

1. Introduction

Irish Sign Language (ISL) plays a vital role in accessible communication and inclusion in Ireland. With the increasing demand for interactive and engaging learning tools, this project explores the use of virtual reality to enhance ISL education, making it more practical and user-friendly for beginners.

1.1 Project Background

Sign Language is the primary means of communication for over 70 million Deaf People worldwide, yet many learners struggle to find effective, accessible resources to master it independently. Learning sign language is important for communication and inclusion, but it can be difficult for beginners to learn on their own. Many people use books, online videos, or take ISL classes from the Irish Deaf Society. While classes provide feedback, books and videos don't correct mistakes. The app ASL Fingerspeller on SideQuest notes that VR and hand tracking make learning "more immersive and interactive," showing how new technology can make signing more engaging [1]. VR allows users to move in a 3D environment, helping them better understand hand shapes and motion.

This project is also important because there are very few tools for learning Irish Sign Language (ISL). Most existing apps focus on American Sign Language (ASL), which uses one hand. ISL uses two hands and different grammar rules, so it needs its own learning tools. A VR-based ISL system could make learning more accessible and inclusive for the Deaf community in Ireland.

Some VR apps already teach sign language, but mostly ASL. For example, ASL Fingerspeller [1] lets users spell words in VR, and Silent Classroom VR [2] offers a classroom-style setup. ASL Champ! [3] uses AI for real-time feedback, but these apps don't include sentence practice or ISL content.

In Ireland, DEF-ISL [4] and CrowdSign exist, but they mainly show static videos or pictures with no interaction. The study Design of Sign Language Learning Media Based on Virtual Reality developed an interactive VR sign language app with visual feedback and found users gained confidence after using it [5].

Since this project uses Godot, features like OpenXR or controller gestures can be handled through rule-based detection instead of machine learning. Novaliendry et al. [5] describe using "a software testing technique that focuses on the functionality of the software without considering its internal structure or implementation details," which suits early-stage testing in Godot. Research also shows that VR learners perform better with "immediate and interactive feedback" [3]. The video VR Sign Language Learning with Hand Tracking [6] demonstrates how visual feedback helps users correct signs in real time. Together, these findings suggest a rule-based, feedback-driven VR system is both feasible and effective for teaching ISL.

1.2 Project Description

What should this immersive VR Sign Language assistant do?

The goal of this project is to help users learn Irish Sign Language (ISL) in an immersive, hands-on way similar to real-world practice. Using the Meta Quest 3/3S VR headset, the application will allow users to engage with interactive lessons and receive feedback on their signing performance. Learners will practice signing the 3D hand shapes of the ISL alphabet, common words that deaf people use, and phrases. To provide accurate recognition tailored to each user, the system will require an initial configuration phase where users create an account and calibrate their hand movements. The hand tracking data collected during this configuration will be stored in a JSON file, which serves as a database of user-specific hand coordinates for future sign recognition and analysis.

Users will begin with learning the alphabet, then progress to signing common words and phrases. After practice, users can evaluate their learning through an interactive testing section. The system allows users to revisit and reinforce any learning modules as needed to improve proficiency.

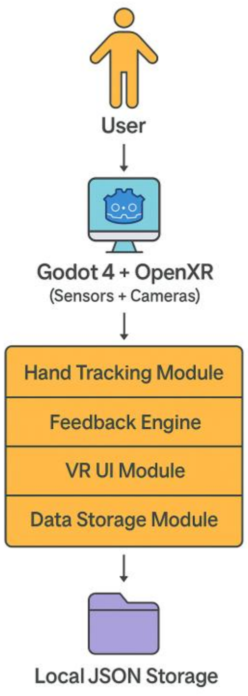


Figure 1: System Architecture Diagram for Sign & Spell VR

1.3 Project Aims and Objectives

To design and implement an interactive virtual reality application that enables users to learn and practice Irish Sign Language (ISL) efficiently through immersive lessons, personalized hand tracking, and real-time feedback.

Objectives:

- Develop a VR application for Meta Quest 3/3S using Godot 4
- Integrate Hand-Tracking

- Implement account creation and user-specific hand configuration system, with data saved in JSON format
- Portray and Create the ISL alphabet
- Create a learning space for common words and phrases in ISL taken from users.
- Develop a testing/assessment section to evaluate user progress
- Allow users to revisit and practice previously learned content as needed.
- Gather user feedback to refine usability and overall learning experience

1.4 Project Scope

The project scope focuses on the VR application. It clarifies the specific features and deliverables that will be developed, as well as what falls outside the project's responsibilities. By setting clear limits, the project scope ensures that work remains targeted, achievable within available resources and time, and helps manage expectations for stakeholders and users.

In Scope:

- **ISL Learning Modules –** Users practice the ISL alphabet, common words, and simple phrases in an interactive VR Environment
- **User Registration and Configuration –** Users create accounts and calibrate their hand configuration from the alphabet, which is then saved to a JSON database for tailored sign recognition.
- **Real-Hand Tracking & Feedback –** The system tracks hand movements and provides immediate visual feedback to guide users and correct errors.
- **Progress Assessment –** Users can test their knowledge and review feedback to evaluate learning and identify areas of improvement
- **Intuitive VR Interface –** The application is designed for the Meta Quest 3/3S headset, ensuring a comfortable and accessible user experience
- **Single-User Mode** – The learning experience is focused on individual progress, with users able to repeat modules and revisit lessons as needed.

Out of Scope:

- Development for VR platforms other than Meta Quest 3/3S
- Cloud Database integration (data stored locally in JSON)

1.5 Thesis Roadmap

This report is structured to provide a comprehensive overview of my project Sign & Spell VR. The following chapters are included:

- Literature Review – This chapter will explore existing solutions, relevant technologies, and related research, providing an evaluation that informs the project direction.
- System Analysis – This chapter will outline the system requirements, including gathering and analysing user needs, and specifying system objectives

- System Design – In this chapter, I will be writing about the selected software development methodology, system architecture, and user interface design.
- Testing and Evaluation – This chapter will cover the testing strategies I used, user evaluation, and results that assess the system’s usability and effectiveness.
- Issues and Future Work – Finally this chapter reflects on the challenges faced during the project and proposed direction for future improvements.

2. Literature Review

2.1 Introduction

In this chapter, I will present the research and exploration carried out by me to inform the development of this project. A range of alternative existing solutions relating to sign language education and learning technology will be analysed, evaluating their strengths and limitations. I will outline and compare technologies considered for use in Sign & Spell VR, focusing on the options available for core features such as hand tracking, user interaction, and immersive feedback. Further research is also discussed to understand key factors that shape accessible learning, with particular attention given to community needs and technical feasibility. Previous final year projects relevant to this topic will be reviewed, illustrating how their findings influenced the overall direction of my project and the decisions made throughout its lifecycle.

2.2 Alternative Existing Solutions

2.2.1 Introduction

There are quite a few sign language apps out there these days, and they all do things a bit differently. Since I wanted to compare them in a fair way, I decided to use the FURPS model, which stands for Functionality, Usability, Reliability, Performance, and Supportability. It's just a simple way to see which apps work well and what they might be missing. For my project, I'm going to look at ASL Fingerspeller, Silent Classroom VR, ASL Champ, DEF-ISL, and this YouTube video about VR sign language learning with hand tracking. I'll go through each one and talk about how they help people learn sign language, what's good, what could be better, and how they compare to what I'm trying to build.

2.2.2 ASL FingerSpeller – SideQuest

ASL Fingerspeller is a VR app that lets you practice fingerspelling in American Sign Language on Meta Quest. The description says you can "practice spelling names, places, and random words" and there's even a timer that makes it a bit more challenging. There are a few comments saying it's good for learning the ASL alphabet quickly but can get kind of repetitive. For my project, I thought it was interesting how they use challenges to help you remember so maybe something like a quiz mode in my own ISL app would work. [1]

2.2.3 Silent Classroom VR – Meta

Silent Classroom VR puts you in a virtual classroom to practice ASL with hand tracking. On the Meta page, it says, "Step into the Silent Classroom and interact with lessons designed to immerse you in a realistic ASL learning environment". People say in reviews that it feels like being in a classroom, just in VR. I think this kind of immersion could help make learning ISL more fun and interactive, so I'd like to take some of that immersion idea into my app but keep it simple so it's easy for beginners. [2]

2.2.4 ASL Champ! – Science Direct

ASL Champ! is a VR game that uses AI to give you feedback on your signs. The ScienceDirect summary puts it: “Players receive real-time feedback on their signing accuracy and are motivated to practice more”. Real-time feedback seems to be what helps most users get better at signing, so I want to use real-time feedback in my project (even if it’s rule-based and not AI for now). [3]

2.2.5 DEF-ISL – App Store

DEF-ISL is mainly for Indian Sign Language, but it’s still a helpful example. The developers say it “makes learning easy through Indian Sign Language,” and users say it’s good for all ages and easy to pick up. It looks like it has a lot of signs and videos, which is a big plus. My ISL project is smaller for now, but I want it to be as easy to use and accessible as DEF-ISL. [4]

2.2.6 Design of Sign Language Learning Media Based on Virtual Reality

This article describes a VR sign learning system that improved people’s confidence: “VR-based sign language learning media with visual feedback improves user confidence and engagement”. Visual feedback in VR really seems to make people willing to try more. For my app, adding feedback when someone tries a sign (even just a basic ‘correct/wrong’ pop-up) should help keep users motivated to keep practicing. [5]

2.2.7 VR Sign Language Learning with Hand Tracking – YouTube

This video shows how hand tracking works without any controllers, and says, “We believe that by enabling people to use their real hands in VR, hand tracking opens up many new capabilities and use cases”. That’s basically what I want to do for my project, but with Irish Sign Language instead. [6]

2.3 Technologies Researched

For my project, I had to investigate which technologies would work best for building a VR sign language learning app. There are actually a few options out there like Unity and Unreal, but I decided to go with the Godot 4 Engine because it’s free, open source, and people online say it’s easier to learn if you haven’t made games before. Also, it’s getting more popular with indie developers and there are tutorials for doing VR stuff in Godot, so it didn’t feel as overwhelming as the bigger engines. Also, I am learning to use Godot Engine 4 with the current module I am studying XR Prototyping. [7]

2.3.1 Meta Quest 3/3S

For my project, I decided to use the Meta Quest 3 headset. One of the main reasons is that it works by itself and doesn’t need a powerful gaming PC, so it’s easier for me and others to set up and use. The headset supports hand tracking, which is really important for this kind of app, since it can track finger movements accurately without using controllers. This helps make the signs more precise and natural, which is what I want for Irish Sign Language learning. The wireless design means you can move around freely while practicing, and it makes the whole system less complicated.

I also found that there are a lot of guides online for developing apps with the Meta Quest, especially for educational and training purposes, which is helpful since I'm still learning how all the VR stuff works. Plus, the Meta Quest is starting to be used in schools for different learning activities, so I think it's a suitable choice for my sign language learning app. [14]

Figure 2: Meta Quest 3/3S

2.3.2 Godot 4 Engine

I am using the Godot 4 game engine for making the project. The main reason is it's free and open source, so I don't need to worry about paying for anything, and it works on my laptop. Godot isn't as popular as Unity or Unreal, but it's getting a lot better, especially now with better support for VR and the OpenXR plugin, which is exactly what I need for the Quest headset. Godot also lets you work with 3D and has good tools for making educational projects. Since my project is about making an environment where users can practice ISL and get feedback, I think Godot should be good enough for what I want. Because it is open source, there's a cool community with lots of questions and tutorials, which is important since I still look stuff up all the time. [8]

GODOT
Game engine

OpenXR™

Figure 3: Godot and OpenXR

2.3.3 GDScript

The coding for my project is mostly in GDScript. I picked GDScript because it's built into Godot and looks a lot like Python, so it's easier for me to learn and I already know a little Python. I tried out other languages like C#, but it felt a bit more complicated to set up, while GDScript just works with the engine and is faster to test stuff. For my app, all the things like checking how the hands are moving, showing if the sign was correct or not, and saving user progress will be handled with GDScript code. It just makes it easier to make changes and try new things if needed, which is important since I still make mistakes or have to fix stuff a lot. [8]

2.3.4 JSON File Storage & Cloud Sync

I decided that for storing the calibration and user data, I'm going to use JSON files. The main reason is, you can save and load JSON files right in Godot without extra plugins or anything, and it's pretty simple to understand. Each user's hand position or progress can go in a JSON file, so when the app starts, it loads all their settings back up. Also, later on, I want to add syncing to the cloud, so if a user logs in from another headset or device, they can get their calibration and progress back. Using JSON with cloud storage (like Firebase or my own server) should be possible by sending the files with web requests, and there are lots of examples about it online. I thought about using a whole database system, but it seemed too complicated for what I need right now. [13]

2.3.5 OpenXR

For my project, I am using the OpenXR plugin with Godot 4 because it is a key part of getting virtual reality working properly on the Meta Quest 3 headset. OpenXR is an open standard that helps different VR hardware and apps talk to each other, so it means my project can work more smoothly with the Quest. Godot 4 supports OpenXR natively, so it was easy enough to enable the plugin and follow the instructions online.

OpenXR is especially useful because it lets my app use features like hand tracking and headset movement without having to build everything from scratch. This means the hand tracking module can take real data from the Quest's sensors and send it straight into the app while practicing signs. It also helps make the VR experience smoother and less glitchy, which is good for beginners.

I followed the steps from the Godot docs to get started, which were clear even for someone who hasn't worked much with XR before. Once it was set up, my app was able to show my hands in VR using the Quest, which was cool to see. Overall, OpenXR is important for my project because it means users get a proper VR experience with modern devices, and I don't have to worry about writing loads of extra code. [8]

2.3.4 Why not other Technologies for my project?

2.3.4.1 Why not Unity?

I considered using Unity since it's one of the most popular engines for VR and there are lots of built-in features and assets for working with virtual reality. However, I found that Unity is complicated for beginners, especially when you want to get hand tracking and VR stuff working quickly. The set-up process seemed overwhelming, and there are lots of things to learn that aren't directly related to my project. Another problem is Unity's licensing—sometimes you run into limits or costs if you want to use more advanced features, which isn't great for students or small projects.

Also, I'm currently doing a course called XR Prototyping, where we learn specifically about using Godot for VR and how to set up hand tracking with it. Since my course is already focused on Godot, it made more sense to stick with what I'm learning rather than starting over with a completely different engine. That way, I can use what I've learned and spend more time building my project, instead of trying to figure out everything from scratch in Unity.

2.3.4.2 Why not Unreal Engine

Unreal Engine looks impressive, especially for making high-end games with really good graphics. But after looking at it, I realized it's kind of overkill for what I need. Most of the guides and samples for Unreal are aimed at big game studios or super advanced developers. For a smaller educational project like mine, it looked like too much work to get started, and it focused a lot more on things like visuals rather than easy hand tracking or quick prototyping. I wanted to spend my time working on sign language learning, not just figuring out the basics of Unreal.

2.3.4.3 Why not a Database?

At one point, I thought about using a database (like SQL or Firebase) to store user data or calibration information. But after looking into it, I realized it was way more complicated than I needed. My project doesn't have hundreds or thousands of users, and I mostly just want to store some settings and hand calibration for whoever uses the app. Using JSON files works much easier with Godot and it's less stuff for me to set up and break. If I ever need to scale up or share data with loads of people, then a database might make more sense, but for now, files are just simpler.

2.4 Other Research

2.4.1 Ethics

2.4.1.1 The FATE Landscape of Sign Language AI Datasets

This paper highlights critical ethical concerns around sign language AI datasets, focusing on fairness, accountability, transparency, and ethics (FATE). The authors point out that "signing videos are deeply personal and culturally specific data," requiring careful consideration of consent and control by Deaf individuals. They emphasize the importance of involving Deaf communities "to avoid cultural appropriation and to build trust." They also talk about challenges in creating datasets that are both representative and respectful to the linguistic diversity of sign languages. [9]

2.4.1.2 Sign Language Recognition and Interdisciplinary Challenges

This study underscores the necessity of cross-disciplinary collaboration in sign language AI research. It stresses ethical issues where “technologists often develop systems without Deaf community input, leading to poor quality and mistrust”. The paper argues that sign languages are complex and culturally rich, stating, “Technologies must respect linguistic identity, or risk producing harmful misrepresentations.” The involvement of linguists, computer vision experts, and Deaf scholars is essential to produce usable and ethical technologies. [10]

2.4.1.3 AI Fairness for Deaf or Hard of Hearing (DHH) Users

The authors address fairness issues in AI systems designed for DHH users, noting that many models “perform poorly with Deaf users due to limited and biased data”. They stress that lack of transparency and interpretability in AI models creates challenges for accessibility. The study calls for “developing trustworthy AI tools with inclusive data and evaluation metrics designed for DHH needs,” emphasizing researchers’ ethical responsibility toward marginalized users. [11]

2.4.1.4 WFD and WASLI Statement on Signing Avatars

The World Federation of the Deaf (WFD) and the World Association of Sign Language Interpreters (WASLI) caution that signing avatars are not a substitute for human interpreters. They state, “Avatars cannot replicate the complex grammar and cultural nuances of sign languages”. For critical communications, human interpreters remain essential. The statement advises, “Use of avatars should be guided by Deaf community consultation and limited to non-urgent content.” [12]

2.4.1.5 Conclusion

These papers collectively highlight that ethical sign language AI development requires respecting Deaf culture, ensuring fairness through community involvement, and maintaining transparency in technology. Sign Spell VR, as an educational VR application, must consider these principles carefully. Incorporating Deaf feedback and providing clear, accurate sign representations aligns with the recommendation that technologies support rather than replace human expertise –. Also, given the limitations of AI and avatars noted in these works, my project focuses on providing accessible, immersive learning with real-time feedback, rather than automated interpretation, to empower users while respecting linguistic complexity and cultural significance.

2.4.2 FURPS Model

The FURPS model is a framework developed at Hewlett-Packard to help evaluate and organize software requirements and quality attributes. The acronym FURPS stands for:

- **Functionality:** The features, capabilities, and security of the software. This covers what the system actually does, what functions it performs, and whether it meets user or project needs. In sign language learning apps, functionality could be about how well it recognizes signs, supports quizzes, or provides helpful feedback.
- **Usability:** How easy and intuitive the software is for real people to use. This includes user interface design, consistency, documentation, and how quickly a new user can learn to operate the system. For educational apps, usability is about making sure learners feel comfortable and can navigate without getting lost or frustrated.
- **Reliability:** The likelihood that the software runs smoothly without crashing or making mistakes, and how well it recovers from errors. It also includes uptime, error rates, and how predictable the system is. For example, learning apps should not freeze, lose user progress, or misunderstand simple inputs.
- **Performance:** How fast the software responds, how efficiently it uses resources, and its ability to handle many users or complex tasks. In a VR sign language app, this would mean smooth hand tracking, minimal lag, and no stuttering, even during busy lessons.
- **Supportability:** How easily the software can be maintained, updated, debugged, or enhanced later. This includes aspects like testability, adaptability, installing updates, and support for localization or different devices. For a student project, good supportability means that new features can be added or bugs fixed without starting over.

FURPS
Usability
Reliability
Performance
Supportability
Functionality

Figure 4: FURPS Model

2.4.2.1 Why Use FURPS

The main reason I'm using the FURPS model is to make sure I think about all the important qualities of my software—not just features, but also how it feels to use, how robust it is, and how easy it will be to maintain. It's a useful checklist when comparing different technologies or solutions and helps ensure nothing major is forgotten in the design and review stages.

2.4.3 Human-Computer Interaction in Sign & Spell VR

The design of Sign & Spell VR incorporates widely recognized HCI principles such as Nielsen's Usability Heuristics, which include guidelines for system feedback, user control, consistency, error prevention, and simplicity. For example, providing immediate feedback after users perform signs aligns with Nielsen's emphasis on visibility of system status, helping learners know if their sign was correct or needs adjustment.

Other key principles come from Schneiderman’s Eight Golden Rules of Interface Design, which stress consistency, error prevention, and minimizing memory load, all relevant to VR environments where users must focus on learning and performing gestures naturally. By reducing cognitive load through simple and predictable navigation, the application supports these golden rules. [15]

Additionally, accessibility standards such as the Web Content Accessibility Guidelines (WCAG) influence the design to ensure inclusivity for users with different abilities, such as left-handed signers or VR novices. These foundations provide a structured approach to enhancing usability and engagement in the virtual learning experience. [16]

2.5 Existing Final Year Projects

2.5.1 Sign Language Translator (SLT)

Title: Sign Language Translator (SLT)

Student: Andrei Botnari

Description (brief): This project is about creating a system that can translate Irish Sign Language (ISL) gestures into text using the Xbox Kinect Sensor. This student developed a recognition pipeline that captured depth data, segmented the hands from the background, and matched the input against predefined gestures. From this project, the aim is to evaluate the accuracy the Kinect sensor under different environmental conditions such as lighting, distance, and motion.

What is complex in this project: The real time gestures recognition using low-resolution depth data. From the project, ISL contains many hand shapes that appear visually similar, making classification and tracking difficult. Managing environmental interference and differentiating between similar gestures (e.g., E, S, and T) required careful calibration and algorithm tuning.

What technical architecture was used: The student used the Kinect V2 depth sensor connected to a PC and developed in Python and C#. This project followed the Spiral Development model, which emphasises the iterative testing of gesture recognition accuracy and usability. The architecture included modules like image capture, segmentation, and features extraction before translating gestures to text output.

Explain key strengths and weaknesses of this project, as you see it:

Strengths: Hard focus on experimental validation, detailed testing across varied conditions. The spiral model ensured improvements. It provided an early exploration into ISL recognition, which remains a niche and research area.

Weaknesses: This project depended heavy on the Kinect hardware, which is now outdated and sensitive to lighting and distance as the resolution is low. It lacked interactive feedback, which means that users could not correct or practice signs dynamically, which limits learning effectiveness. [18]

2.5.2 VR Music Learner

Title: VR Music Learner

Student: Maciej Golubski

Description (brief): This student developed a virtual reality environment where users could learn musical instruments such as guitar or piano in an immersive, game-like setting. He integrated VR Gameplay, sound recognition, and a connected mobile app to track user progress and statistics. It aimed to make music learning more engaging though interaction, rhythm matching, and score tracking.

What is complex in this project: The challenge I have read from this project is the VR motion input, audio recognition, and mobile data communication. Maintaining accurate timing between the players performance and in-game feedback required strong optimization.

What technical architecture was used: This project is developed in Unity using C#, and using the Oculus Quest Headset for VR interaction, a web server and database for the scoring system, and a mobile companion app for data visuality. The student used Agile/Scrum workflow for iterative development and user testing.

Explain key strengths and weaknesses of this project, as you see it:

Strengths: Good integration with VR, mobile, backend technology. The focus is strongly seen on gamifying it and feedback loops that enhance learning and motivation. This shows well-thought user experience for the user.

Weaknesses: It has limited educational depth. Mainly focuses on music theory. System performance may change depending on network and the accuracy of audio input. The student could have designed it from a broader perspective of testing and adding additional learning elements. [17]

2.6 Conclusions

To sum up, the research and reviews explored in this chapter have been really valuable for shaping the direction and design of my Sign & Spell VR project. I started by looking at existing solutions and apps for sign language education—like ASL Fingerspeller, Silent Classroom VR, ASL Champ!, DEF-ISL, and various research projects and videos—and found that each one has its own way of helping people learn sign language in a digital way. Most of them focus on American or Indian Sign Language, which means there's not a lot out there specifically for Irish Sign Language (ISL), so my project can help fill that gap.

The FURPS model helped me break down each existing solution in terms of what they do well and what could be improved. For example, features like real-time feedback (as seen in ASL Champ!) and an immersive classroom setting (like Silent Classroom VR) really show that learning is more effective when users get instant responses and feel like they're in a realistic environment. These findings encouraged me to design my own project so that it keeps things simple for beginners, but also offers useful feedback and motivation for repeated practice. Visual feedback, as highlighted in educational VR research, seems super important for giving users confidence and prompting them to keep learning.

Throughout the section on technologies researched, I compared different platforms and tools and explained why I picked Godot 4, Meta Quest 3, GDScript, and JSON file storage. I found out that Godot is a good fit for someone with my level of experience, especially since my XR Prototyping course is teaching me exactly how to use it for VR and hand tracking. Similarly, Meta Quest 3 was chosen not just for how portable it is, but because of its finger tracking and widespread use in education now. GDScript made the development process easier for me, and using JSON files for calibration and user data keeps the project simple and manageable for now, while still letting me plan for cloud syncing in the future.

I also learned a lot about the ethical and community aspects of sign language technology by reading recent academic papers and official statements. It's clear that tech projects involving sign language need to involve the Deaf community, respect the complexities and culture of sign languages, and make learning resources accessible and fair. This means my app should not try to replace human interpreters and must be designed in a way that's open to feedback from people who actually use ISL every day.

Reflecting on previous final year projects (like Sign Language Translator and VR Music Learner) showed me how hardware choices, feedback mechanisms, and user engagement techniques influenced both their successes and their limitations. It reinforced that choosing the right tools, focusing on real-time feedback, and designing for actual user needs (instead of just technical novelty) are critical to project success.

All this research guided me to make informed decisions for Sign & Spell VR. My project aims to be easy to use with an intuitive interface for beginners, offer real-time feedback for learning, and be accessible on affordable technology. It is not trying to solve everything about ISL sign language education, but to make a useful and ethical VR learning app that stands out from what is already out there, while continuing to learn from both the technology and the community as development goes on.

3. System Analysis

3.1 System Overview

3.1.1 What is Sign & Spell VR?

Sign & Spell VR is a virtual reality app focused on helping people learn the basics of Irish Sign Language (ISL). It's designed for beginner users, like students or anyone interested in ISL, who want to practice at home or even in a classroom. The idea is to make sign language learning more accessible and interesting by making it interactive—so you're not just watching videos but practicing with your hands in a VR environment.

3.1.2 How the System Works

When the user puts on the Meta Quest 3 headset, they are brought into a virtual room that's kept simple and uncluttered. The system uses the Quest's cameras and sensors to track the user's hands in real time. Their real hand movements translate into virtual hands they can see in front of them. This way, users can practice making ISL letters, words, and maybe even sentences, with feedback straight away about how close their sign is to the correct one.

3.1.3 User Experience and Interface

From a non-technical view, the app should be easy to use. The menus are meant to be clear, with options like "Calibrate Hands," "Practice Letters," "Practice Words," and "View Progress." There's a short tutorial the first time they use it, and plenty of help tips in the interface for when people get stuck or don't know what to do next. I tried to plan the app so it doesn't overload users with too many choices at once. All navigation happens in VR using simple gestures or big buttons.

3.1.4 Practice and Feedback

Each practice session starts with calibration, so the hand tracking is as accurate as possible. Then, users can select which part of ISL they want to practice. If the system recognizes their sign correctly, the app provides a "Well done!" type message, and if not, it suggests trying again or gives a hint. Gamified elements like streaks or quizzes might be added so learners stay motivated, based on what's worked in other language learning apps.

3.1.5 Saving Progress and Accessibility

The system automatically saves calibration data and user progress, so learners can come back later and keep going from where they stopped, even if they change devices (once cloud sync is working). Accessibility is also a priority—I want left-handed users, or people who aren't used to VR, to be able to use the app without issues. The design avoids small, fiddly controls or anything that could make someone feel lost or out of depth.

3.1.6 Inspiration and Aims

I chose this layout and approach because of what came up in my proposal and the literature review: real-time feedback is important, the interface must be welcoming for beginners, and the whole experience should support users learning at their own pace. I'm also aiming for it to feel a bit like the more interactive VR apps I researched but tailored to make ISL learning simpler and more fun.

3.2 Requirements Gathering

3.2.1 Who Are the Stakeholders?

For this project, figuring out who the main stakeholders are is important because it helps me decide what features to prioritize and how to design the app. Every group of people involved will have different needs and expectations, so I need to think about all of them, not just the end users.

3.2.1.1 New Learners

These are mostly people starting out with Irish Sign Language and virtual reality. Their main goal is to learn ISL in a way that feels approachable and supportive.

Persona: Sophie

Sophie is a college student who's heard about ISL but has never tried learning it before. She also hasn't used a VR headset, so everything feels a bit new and maybe intimidating. Sophie hopes that the app will give her clear instructions, let her go at her own pace, and provide lots of encouragement if she gets signs wrong. If things get too technical or confusing, Sophie might lose motivation, so keeping things simple is key.

3.2.1.2 Teachers and Instructors

Teachers will want to use the app as a learning aid in lessons or for assigning practice at home. Their needs include easy setup, management of different users, and being able to monitor or support their students' learning.

Persona: Martin

Martin runs ISL classes for adults. He wants a tool that's quick to set up with minimal fuss, so students can jump straight into practicing. Martin likes being able to see who needs extra help and would appreciate ways to check progress after each lesson. He also values any notes in the app that help him explain features to students, especially those who aren't tech-savvy.

3.2.1.3 Deaf Community

Members of the Deaf community (ISL users, interpreters, advocates) help ensure the app is respectful, accurate, and culturally sensitive.

Persona: Aisling

Aisling is an ISL interpreter and strong advocate for Deaf awareness. She's keen to see more tools making ISL accessible but is cautious about apps teaching incorrect signs or oversimplifying the language. She's happy to offer feedback if asked and hopes the app developers listen to her suggestions, particularly about sign accuracy and cultural respect.

3.2.1.4 Potential Stakeholders

In the future, more people might use the app—like families, researchers, or schools. Being aware of this early on helps keep the design open for improvements and wider use.

Persona: Mrs. O'Connor
Mrs. O'Connor has a Deaf son and wants tools that help their family learn ISL together. She values apps that are gentle for beginners, easy to follow, and work across different devices. She's especially happy if an app lets her review lessons at her own pace after the kids have gone to bed.

3.2.2 How I Collected Requirements

To figure out exactly what features and functions my ISL VR app should have, I used a mix of research, feedback, and practical exploration. First, I looked at a bunch of existing apps and reviewed research papers to see what works well for sign language learning, especially in VR. I paid close attention to feedback from real users in app reviews and on forums—people often mention what annoys them or what really helps them learn.

Next, I talked informally with classmates and friends who are interested in learning sign language and with a couple of deaf friends I know, just to see what stuff they might expect if they were to use a VR learning tool. Right now, my plan is to use simple questionnaires or online forms once I've got an early version running, so people can give feedback on what's easy, what's confusing, and what's missing.

I'm also relying on my own experience from my XR Prototyping course. Sometimes, when you start building, you notice things that sound good on paper aren't very practical—so I try to keep notes every time I run into a challenge or notice a feature that makes my life easier as the developer.

3.2.3 Initial Requirements List

From everything I've learned so far (other apps, research, talking to people, and my own experiments), these are the main requirements I've collected:

- The app must use Meta Quest 3's hand tracking accurately, so users don't need controllers.
- There should be an easy-to-follow calibration system, so all users get good results, no matter their hand size or starting position.
- The user interface must be clear and simple, using big buttons and straightforward instructions.
- Each time a user makes a sign, the app should give instant feedback—either confirming it was correct or suggesting what to fix.
- Progress and calibration settings must be saved, so users don't have to start over every session.

- If possible, there should be an option to sync progress online, so users can keep track of their learning even if they switch devices.
- The app must run smoothly on Meta Quest 3 hardware, with no crashes or big slowdowns.
- The system should be built in Godot 4 (since that's what my XR course covers), using GDScript and JSON files for simplicity.
- Accessibility should be considered, making the app usable for beginners and possible left-handed users.

3.3 Requirements Analysis

3.3.1 Organising the Requirements

After gathering my requirements, I realised it was important to sort them properly so I could keep everything clear and not get overwhelmed. First, I put them into two categories. Functional requirements are basically what the app needs to do for the user—like recognising hand signs, showing feedback, or letting someone calibrate their hands. Non-functional requirements are more about the "feel" of the app, like making sure it's simple, doesn't lag, or that data is kept safe.

For example, "the system must let users save progress" is functional, because it's an actual feature. But "the system should be easy for beginners" is non-functional, since it's about the overall user experience instead of a specific function. I tried to keep separate lists as I wrote them down. I probably didn't get all of them the first time, but I aim to add more if I see problems during development or testing.

3.3.2 Prioritising Features

With so many things to include, I saw I had to decide which parts were vital and which would be "extras" if I have more time. For my project, the absolute top priorities are hand tracking that works, recognising the ISL signs, basic feedback for the learners, and being able to save and load user progress. Other things like making pretty menus, adding quizzes, or even the cloud sync are important, but honestly, they can wait until the main parts work.

To help stay on track, I made a little list for myself:

- Must-have: Core sign tracking, accurate feedback, calibration, basic navigation, and saving data.
- Should-have: More lesson modes, cloud sync, better accessibility, maybe progress charts.
- Nice-to-have: Fancy graphics, custom avatars, very detailed analytics, advanced quizzes.

I think this makes the whole thing less stressful because I know it's better to have a simple working app than an unfinished fancy one.

3.3.3 Making an Initial System Model

To get a better idea of what to build, I tried to picture how different parts of the system will work together. Here are the modules I think I'll need:

- **Hand Tracking Module:** This will oversee getting the data from Meta Quest 3's sensors and figuring out what gestures the user is making with their fingers and hands. Without this, nothing else in the app really works.
- **Feedback Module:** Every time a user tries a sign, this module compares what they did to what the sign should look like and then gives them a message or score ("Great job!" or "Try again"). This helps learners know how they're doing right away.
- **User Interface (UI):** The UI is supposed to be super clear, with menus that make sense and buttons that are big enough for VR. It also includes the tutorial, help tips, and any graphics or text that guide users during calibration and practice.
- **Data Storage:** I'll use JSON files (because I'm familiar with them from class), and maybe later add cloud saving if I figure it out. It's not very fancy, but it keeps progress between sessions.

I'm hoping that by keeping things modular, I can debug one part at a time and possibly upgrade them later without breaking everything else.

3.3.4 Risk and Challenges

While thinking all this through, I started noticing some issues that could come up. One big challenge is making sure the hand tracking is accurate enough—sometimes VR hand tracking skips or gets confused with certain gestures, especially with ISL letters that look similar. If calibration isn't good, users might get frustrated fast. Another risk is performance: if the feedback or graphics are too heavy, the headset could lag or crash. I'm also a bit worried users could find the VR controls confusing or even a bit intimidating if they aren't VR gamers already.

To manage these, my plan is to focus on really clear calibration and simple user flows. I also want to get feedback from real users (probably friends at first) as soon as I have a working prototype, so I can spot big issues before it's too late to fix them.

3.3.5 Reviewing Gaps

Even after all this, I know my plan isn't perfect. There are still things I need to research, like how cloud syncing actually works in Godot or if there are any security risks with saving data online. I also still need more feedback from teachers or the Deaf community—my current list of requirements relies a lot on what I think and what I found online. Maybe I'll find out something's missing or needs changing when people actually try the app.

3.4 Initial System Specification

This section is where I try to put together all the main stuff my system needs so I actually have a plan to follow when I start developing. I break it down into what the app should be able to do (functional requirements), what qualities it should have (non-functional requirements), and my first idea for the system architecture—like the main building blocks and how they'll connect.

3.4.1 Initial Functional Requirements

These are the things the app must actually do for users (so I can say the main features are working).

- The system must track both hands and all five fingers using the Meta Quest 3 headset's sensors, so that users can practice ISL with proper finger shapes and positions.
- The app must let users calibrate the system for their own hands before starting, since different people might have different hand shapes or positions, and this can affect accuracy.
- The system should recognize and check ISL signs, giving the learner instant feedback on whether they signed correctly or need to try again.
- There needs to be a simple navigation system (menus for lessons, practice, calibration, and feedback) that is easy to use, especially for people new to VR.
- All user progress (like what signs have been learned, calibration data, and any scores or streaks) must be saved automatically, so users can pick up where they left off.
- Ideally, the app should include practice modes (like the alphabet, words, maybe even sentences later), and a way to track personal progress or see simple stats.
- If I get that far, the system can have an option to sync data online (cloud sync) so users aren't limited to one headset or device.

3.4.2 Initial Non-Functional Requirements

These are about the overall feel, stability, and usability—what the experience should be like (even if it's not a "feature" the user clicks).

- The app should be really easy to start using, even for people with no VR or ISL experience. Menus and instructions need to be clear, big enough for VR, and not overloaded.
- It should run smoothly on Meta Quest 3; lag, crashes, or slow responses are likely to frustrate users and make them quit.
- The system should be reliable—progress and calibration data shouldn't go missing, and the app should give clear error messages if something goes wrong.
- Accessibility should be good, so features like left-handed mode, and large, readable text are considered, making sure as many people as possible can use it.
- Data privacy is important especially if there is cloud saving, so user progress and settings should only be visible to them and not shared without permission.
- The app should be easy to maintain or update later on—if I need to fix bugs or add new features in the future, it shouldn't require starting over from scratch.

3.4.3 Initial System Architecture

I know having a plan for how different parts will connect makes life a lot easier later. Here's how I see the architecture at this stage (it might change as I progress):

Main Components/Modules:

1. **Hand Tracking Module:**
    - Uses Meta Quest 3's built-in sensors and software (probably OpenXR) to track and get hand/finger position data in real time.
    - Feeds this data into the feedback engine.
2. **Feedback Engine:**
    - Compares user's hand positions to stored, "correct" ISL signs.
    - Gives feedback instantly in the app, like "Correct!" or shows what was wrong.
3. **User Interface (UI) Module:**
    - Handles all menus, calibration screens, practice lessons, and feedback pop-ups.
    - Designed in Godot 4 with big VR-friendly buttons and simple navigation.
4. **Data Storage Module:**
    - Stores user progress and settings using local JSON files.
    - Might connect to a cloud server/API for syncing later, depending on progress and what I can manage to learn in time.

3.4.4 How it all Works

- The user puts on the headset and chooses a practice mode from the UI.
- The Hand Tracking Module tracks their signs and sends info to the Feedback Engine.
- Feedback Engine checks if the sign matches the database of ISL gestures and returns feedback (correct/incorrect, hints).
- The UI displays response, score, or any helpful messages.
- Data Storage records new progress or calibration automatically.

3.5 Conclusions

Overall, this System Analysis section helped me get a much clearer picture of what I want my Sign & Spell VR app to be. By breaking everything down—like who my users are, what features really matter, which parts of the system are most important, and what risks might show up later.

A lot of what I wrote comes from looking at other sign language learning apps, doing research for my proposal, and asking a few people what they'd want to see in a project like this. Figuring out requirements and then sorting them into “must have” and “nice to have” was helpful, because it means I’m less likely to get distracted by extras that aren’t necessary. Focusing on basics—like hand tracking, feedback, and ease of use—should give users something practical and not overwhelming, which I noticed was important from my research.

The architecture I planned out isn’t super fancy, but it gives me a good starting point and makes it easier to imagine how things will come together. I know there will be problems and I’ll probably have to change some things along the way.

There are still things I need to figure out—like the technical side of cloud sync, handling mistakes in hand tracking, or making the app accessible for more people. But having an initial specification and identified risks means if I get stuck, I at least know where my gaps are. I’ll keep checking with my user test group as I go to make sure the app is useful and respectful.

In summary, this section sets out a realistic plan that takes what I learned from my research and applies it to the project itself. The next step will be to start building the core features, keep gathering feedback as I go, and be flexible when something doesn’t work out exactly as planned.

4. System Design

4.1 Introduction

4.1.1 Translating Requirements into Design

The purpose of this section is to describe how the requirements and specifications identified earlier will be transformed into a concrete design for the Sign & Spell VR application. This involves both planning how the software will be structured and deciding on the technologies to be used, with a clear focus on making the app functional, usable, and scalable. Effective design is important because it ensures that development proceeds smoothly, and that the final system meets user needs effectively.

4.1.2 Design Principles and Goals

In designing this system, the main principles guiding the process are simplicity, modularity, and user-friendliness. Simplicity helps ensure that users, especially beginners, can interact with the app without confusion. Modularity allows me as a developer to isolate and manage different parts of the system separately, making it easier to debug and add new features later. User-friendliness is vital to encourage learners to practice consistently, by providing clear feedback, easy navigation, and an immersive but not overwhelming VR environment. These goals reflect the requirements identified from prior research and user feedback.

4.2 Software Methodology

4.2.1 Incremental Model

For the development of the project, the Incremental Model has been chosen as the software development methodology. This process involves dividing the project into small parts or “increments,” delivering a functional system for each stage that progressively adds new features. The focus is on building and testing core features first—such as hand tracking, basic sign recognition, and user feedback—and then adding more advanced features in later increments.

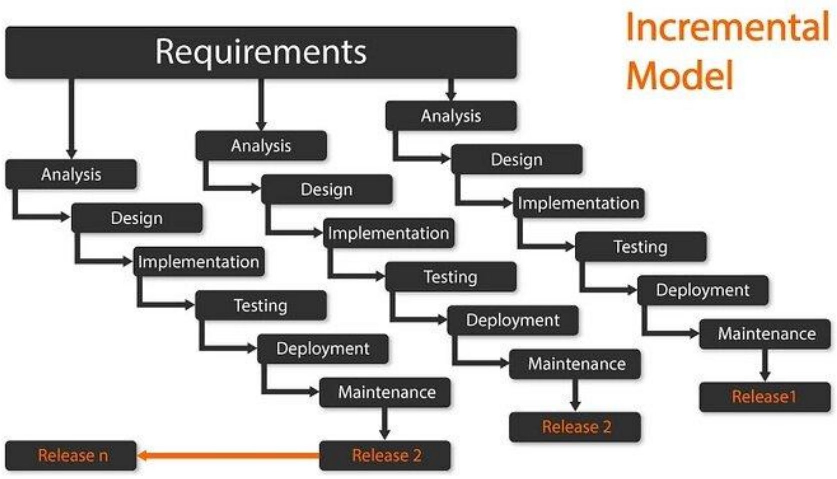


Figure 5: Incremental Model

4.2.2 Advantages of Incremental Development for my Project

This methodology offers several advantages for a solo project like mine. By developing the system in stages, I can continuously test and receive feedback on smaller parts, allowing me to identify and fix problems earlier. It also helps me stay motivated by achieving regular milestones and ensures that the most critical features are prioritized first. Additionally, incremental development supports flexibility, letting me adjust or reschedule features depending on progress and learning during development.

4.2.3 How Will I Apply it

In practice, I plan to start with a simple working prototype containing the core functionality: real-time hand tracking, recognition of basic ISL letters, and immediate feedback. This increment will focus on system stability and accurate tracking. Later increments will expand the practice set to include words and sentences, improve user interface polish, and implement features like data saving and cloud syncing. Each increment will be tested thoroughly before moving to the next.

4.3 Overview of System

4.3.1 Logical Architecture

The logical architecture is like a blueprint for the app. It shows how different parts of the system will work together to make the Sign & Spell VR app function smoothly. These parts are modules or components, each doing a specific job but communicating with each other, so the user experience is seamless. Having this clear makes it easier to build and test the app in pieces.

4.3.2 Main System Modules

Here are the key modules I plan to develop, along with what each does:

- **Hand Tracking Module:**
  This is the foundation of the app. It captures the user’s hand and finger movements via the Meta Quest 3’s built-in sensors. It uses the OpenXR plugin in Godot 4 to get accurate tracking data. This module sends the raw hand pose data to the rest of the app continuously during use, allowing the system to “see” what the user is signing.
- **Feedback Engine:**
  Once it receives the hand data, this module compares the user’s gesture to the database of valid ISL signs. It decides if the sign is correct, partially correct, or needs more practice. It then sends feedback signals back to the UI, like a “correct” message or suggestions on how to improve. This module helps provide real-time responses, which is important to keep learners engaged.
- **User Interface (UI) Module:**
  The UI module controls everything the user sees and interacts with inside VR. This includes menus for selecting practice modes, buttons for calibration, feedback pop-

ups after a sign attempt, and progress indicators. The UI is designed to be clear and simple, using larger elements suitable for VR so users aren't confused or overwhelmed.

- **Data Storage Module:**
  This module manages saving and loading personal user data, such as calibration settings, user progress, and any practice statistics. It works with local JSON files initially, which is straightforward and well supported by Godot. Later versions may expand this to include cloud storage or syncing features, allowing users to access their data across multiple headsets or devices.

4.3.3 Physical Architecture and Development

The entire system is designed to run directly on the Meta Quest 3 headset, which is a standalone virtual reality device. This means that all the processing required for the app, including tracking hands, giving feedback, displaying menus, and saving data, will happen within the headset itself rather than relying on an external PC or server. This makes the system more portable and easier for users to set up, because they don't need extra hardware or complicated connections.

Since everything runs on the device, users can just put on the headset and start practicing right away, without needing to be connected to a computer. Development of the app will be done using Godot 4, which supports OpenXR—this is the technology used by the Quest for VR applications. Godot makes it possible to build VR apps that work smoothly on the Quest without needing extra software.

All app files, user profile data, and temporary cache will be stored on the internal storage of the headset. This ensures fast access and reduces the chances of data loss. If I add future features like cloud syncing, those options will connect to online servers, but the core system will operate independently offline.

The goal is to deliver a user-friendly, wireless experience that is reliable and straightforward for anyone to use, whether they're teachers, students, or just beginners exploring sign language in VR. The system setup is intended to be simple and efficient, making it suitable for different environments like classrooms, homes, or labs.

4.3.4 Data Flow Between Modules

The system's data flow describes how information moves between the different parts of the system when a user interacts with the app. Here's how a typical session would go, step by step:

- **User Interaction:**
  When the user puts on the VR headset and opens the app, they select a lesson mode or practice activity from the main menu.
- **Hand Tracking Module:**
  This module is always active while practicing. It continuously collects data about the user's hand and finger positions using the sensors on Meta Quest 3. The module sends this positional data in real time to the Feedback Engine.

- **Feedback Processing:**
  The Feedback Engine receives the hand data and compares it against a stored database of correct ISL signs or gestures. It checks whether the user's signs match the correct signs closely enough to be considered correct. It then generates feedback, such as a visual message ("Good job!") or hints if the sign was wrong.
- **User Feedback Interface:**
  The feedback results are immediately sent to the UI module, which displays messages, score indicators, or visual cues directly in VR. This helps the user immediately understand their performance and adjust.
- **Progress Storage:**
  After the practice session, the system automatically saves the user's progress, including calibration settings, completed lessons, and scores. This data is stored locally in JSON files but could later be synced online if I add cloud features.

This data flow ensures that the system remains responsive, engaging, and easy to use. All modules coordinate smoothly to provide real-time feedback and keep track of the user's improvement.

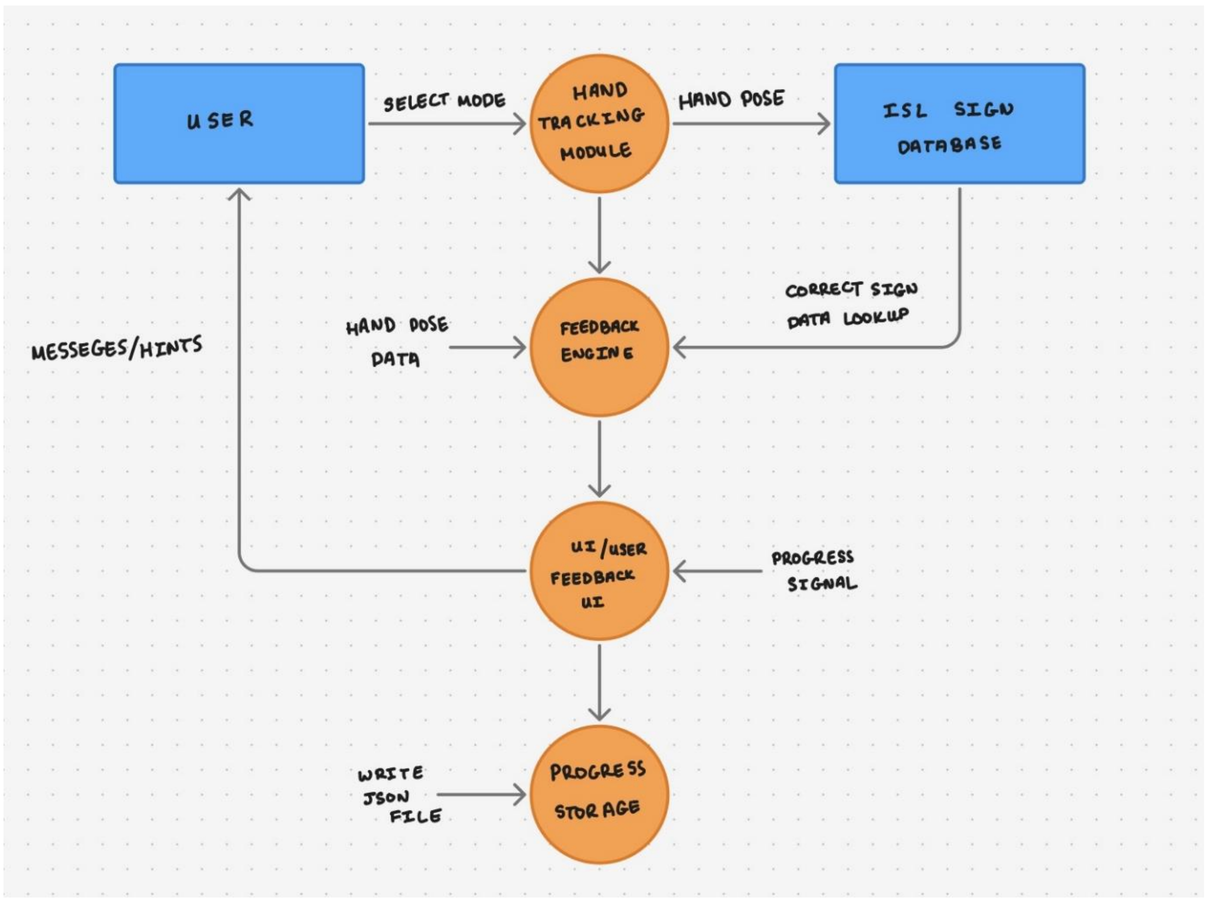


Figure 6: Data Flow Diagram Between Modules in Sign & Spell VR

4.3.4 Modular Benefits

Using a modular system design means I organize the app into different independent parts or modules, such as hand tracking, feedback, UI, and data storage. This approach offers many benefits for development:

- **Easier Testing:**
  I can develop and test each module separately. For example, I can perfect the hand tracking module without worrying about the feedback system or UI. If something doesn’t work, it’s easier to pinpoint and fix the problem.
- **Simpler Debugging:**
  When bugs happen, I know exactly which module is causing the problem, making it faster to troubleshoot and fix them.
- **Future Expansion:**
  If I want to add new features later, such as cloud syncing, progress charts, or more complex feedback, I can develop those parts independently and plug them into the existing system without rewriting everything.
- **Better Maintenance:**
  The system is less likely to break when I update or change parts of it because each module is separate. This makes it easier to keep the app running smoothly over time.

Overall, the modular design makes development more manageable, especially for a solo project like mine. It provides a clear structure, reduces complexity, and helps me stay organized as I build and improve the app.

4.4 Design System

4.4.1 Applying Design Approach

For the design of the Sign & Spell VR system, I have used a structured design methodology to translate the specifications and requirements into a detailed, workable plan. The goal of the design phase is to create clear models and diagrams that describe how each part of the system will function and interact, providing a roadmap for development. This methodology focuses on modularity, clarity, and scalability, which aligns well with the incremental development approach I am following.

4.4.2 Modular Design Approach

The system design follows a modular approach, dividing the application into distinct components, each responsible for specific tasks. This allows each module to be developed and tested independently before integration. The main modules, as identified earlier, include:

- **Hand Tracking Module:** Designed to interface smoothly with the hardware sensors, this module will interpret the raw data into user hand poses.
- **Feedback Module:** Responsible for interpreting hand poses and determining sign correctness, this module will use algorithmic matching techniques based on a sign database.
- **User Interface Module:** This module will manage all user interactions, including menus, practice sessions, feedback displays, and calibration workflows, all designed for clear VR visualization.

- **Data Storage Module:** Tasked with saving and loading user profiles, progress data, and calibration settings, this module will use lightweight JSON files for local storage with future cloud integration planned.

By designing these modules separately but ensuring clear interaction points, the system stays maintainable and flexible for future feature upgrades.

4.4.3 Use of UML Diagrams

As part of the design methodology, I have planned to create several UML (Unified Modelling Language) diagrams to visualize the system before starting detailed coding (none completed):

- **Use Case Diagrams:** To identify and represent the interactions between different users (learners, teachers) and the system functions.
- **Class Diagrams:** To define how data is structured within each module, such as classes for Hand, Gesture, User Profile, and Feedback.
- **Sequence Diagrams:** To map out the flow of interactions between modules, especially how hand tracking data flows through the feedback loop to the user interface.
- **Activity Diagrams:** To represent the step-by-step processes, such as user calibration, practicing a sign, and saving progress.

These diagrams help ensure completeness and correctness in the design, making development more straightforward and reducing potential errors.

(No image completed)

4.4.4 Prototyping and Iteration

The design methodology emphasizes early prototyping—building simple versions of each module to test concepts and gather feedback. For example, an initial prototype of the hand tracking will confirm how accurately the Meta Quest 3 sensors can detect finger positions in Godot 4. Similarly, a basic prototype of the user interface will help verify menu navigation and feedback readability in VR.

This iterative process allows me to revise the design continually based on testing results and user feedback, leading to a more robust and user-friendly final application.

4.4.5 Scalability and Flexibility Considerations

During the design, I have taken care to keep the system scalable and flexible. For example:

- The feedback engine is designed so that adding new signs or improving the recognition algorithm won't require a complete rewrite.
- The data storage module can be extended to support cloud syncing or networked data without affecting other parts of the app.

- The modular UI allows for easy changes to the layout or addition of new practice modes.

By keeping these principles in mind during design, I aim to build a system that not only meets the current requirements but can also accommodate future enhancements.

4.5 Conclusions

In this chapter, I have outlined how I plan to design the Sign & Spell VR system by translating the earlier requirements into a clear and manageable software design. By choosing the incremental development model, I can focus on delivering functional parts of the system step-by-step, allowing me to build core features first and add enhancements later. This method supports testing, feedback, and improvement at every stage, which I believe is ideal for this project.

My design adopts a modular approach, dividing the system into well-defined components such as hand tracking, feedback, user interface, and data storage. This separation makes the development process easier to handle and helps me focus on one part at a time, which is especially useful for a solo project. The modular design also provides flexibility for future upgrades, whether that's adding cloud syncing, more lesson types, or better accessibility features.

I have planned to use standard design tools like UML diagrams to visualize the system and its workflows before diving deeply into coding. This will help ensure I have a solid foundation and reduce bugs or mistakes. Additionally, early prototyping and iteration are important aspects of the design process, paving the way for a user-friendly and robust application.

Although there are challenges ahead, such as ensuring accurate hand tracking and smooth user interaction in VR, the structured design and incremental approach give me confidence that these issues can be addressed step-by-step. Overall, this design chapter lays a strong groundwork for the next phase of development, ensuring the system will be effective, expandable, and enjoyable to use.

5. Testing and Evaluation

5.1 Introduction

In this section, I will explain the testing and evaluation methods I plan to use for the system. Since I am still working on the project, I have focused on deciding how to test the parts I have made so far and how to check if they work well. Testing is important to make sure each part of the system does what it is supposed to do. Evaluation is also important to see if the system is easy to use and useful for learning.

5.2 Plan for Testing

The system has different parts like hand tracking, giving feedback, user interface and saving data. Each part needs different ways to test it:

- For the Hand Tracking Module, I will test how well the system recognizes the signs of the Irish Sign Language alphabet and some words. This will involve me signing different letters first and then words to see if the system detects them correctly. I will also try to test the hand tracking in different lighting conditions and angles to see if it still works well, because lighting can sometimes affect VR sensors. I will also ask some friends to test this part so I can get more varied feedback.
- To get more input, I am giving my friends my VR headset to try out the system themselves. They will use and test the main features like signing letters and sentences and then tell me what they think about how well it works. They will also let me know if any parts are confusing or if the system fails to recognize their signs. Their feedback is important because they might find different problems than I do, and they might have ideas for improvements or new features.
- The Feedback Module needs to be tested by checking if the messages and hints shown to users are correct. For example, when the system detects a correct sign, it should say so clearly, and if it's the wrong sign, it should give hints or corrections. I will test this by signing right and wrong signs intentionally and observe if the feedback matches what I did. This part will be tested mostly manually, but I plan to write some small test cases in the prototype to double-check basic things automatically.
- Testing the User Interface (UI) means going through all the menus, buttons, and prompts to make sure they work how they should. I will test if buttons respond correctly when clicked and if the text instructions are easy to understand. I will also check if users can navigate between different parts of the application smoothly without getting lost or stuck. Again, I will ask my friends to try the UI as well and report any parts that feel confusing or awkward in VR.
- For the Data Storage Module, I will try saving my user settings, hand calibration data, and progress, then closing and reopening the application to check if the data loads back correctly. This will make sure users do not lose their settings or progress when they stop using the system. I will create multiple test profiles and check if data keeps correct values for each profile.

Testing will be done continuously and bit by bit while I keep working on the system. I plan to test every new feature, instead of waiting until the end. This way, I can find mistakes early and fix them before moving on to the next part. I think this approach helps prevent big problems from building up and makes it easier to improve the system step by step.

5.3 Plan for Evaluation

Before developing the system, I had my friends try existing Irish Sign Language learning apps and ASL apps to understand their features and limitations. Their feedback helped me identify requirements and informed the design of the Sign & Spell VR system. This early comparison also guides what aspects I should focus on when testing and evaluating my own system.

In this section, I will describe how I plan to check if each part of the system meets its goals and how I will understand what works well or what needs improvement. This includes different methods used for different components to get useful feedback.

- For the Hand Tracking Module, I will evaluate how accurate and responsive the system is when recognizing ISL signs. This will be done by having people use the system to sign letters and simple words, and I will observe whether the system correctly detects what they sign. The users will also give me their opinions on how well it feels, for example if it is smooth, fast, or delayed. This helps find areas that might need fixing to improve recognition.
- The Feedback Module will be evaluated by checking if the feedback messages or hints given to users are clear and helpful. Users will tell me if the feedback helps them correct their signs or if the messages are confusing. This evaluation focuses on matching the feedback with what the user expects to get.
- The User Interface evaluation will focus on how easy it is for users to navigate and use the features in the VR environment. I will watch users as they complete tasks and look for any confusion or errors. Then, I will ask them simple questions about whether they found the interface friendly and if instructions were clear. This kind of evaluation is important to make the system more user-friendly.
- For the Data Storage Module, evaluation will ensure that user data is saved and loaded correctly. I will simulate typical user behaviour, saving progress and settings, quitting, and then restarting the system to confirm data persistence. If there are problems with data not being saved or lost, this part of the system will need improvement.
- When evaluating the Overall System, I will check if users find it helpful and motivating for learning Irish Sign Language. I plan to ask users before and after they try the system about their confidence in signing. This can help measure if the system supports learning. Technical issues such as crashes or delays will also be monitored because these can affect user experience.

At this stage, the evaluation will be mostly informal and involve a small number of users, mainly friends testing the prototype. Their feedback will guide changes and improvements for the next phase of the project. In the final report, more formal evaluation with a larger user group and structured tests will be planned to better measure outcomes and usability.

5.4 Conclusions

Overall, the testing and evaluation plans I have set up so far give me a good way to check how well my system is working at this stage. Since the project is not finished yet, the testing done until now has been mostly informal and focused on core parts like hand tracking and feedback from existing apps and prototype. Having my friends try out my system, existing apps, and give feedback has been helpful to find problems I didn’t notice and get ideas for making it better.

In conclusion, this plan for testing and evaluation is a good starting point for making sure the Sign & Spell VR system will work well and meet users' needs. It also shows that I am thinking about how to improve the system based on real feedback. I will keep testing new features and asking users for their opinions so I can make the best possible VR learning tool for Irish Sign Language.

6. System Prototype

6.1 Introduction

For my prototype, I mainly wanted to see if I could get hand tracking and basic sign detection working in VR. I used Godot 4 because that's what we're learning in class, and it also lets me use the Meta Quest 3 headset to try out hand tracking. I didn't have much experience with VR projects before, so I tried to keep everything simple and focused on learning the basics first.

I started by adding a sign language alphabet chart to the environment so I could easily check what each sign should look like. This is the chart I found online that represents the Irish Sign Language.

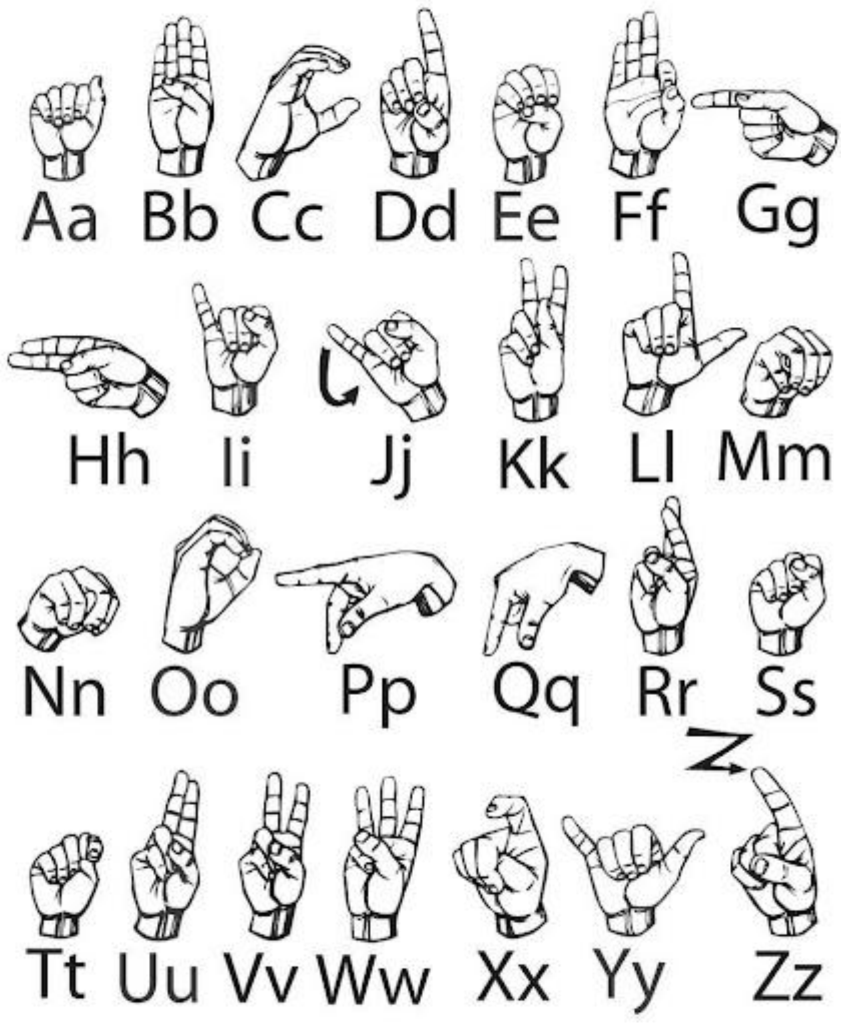
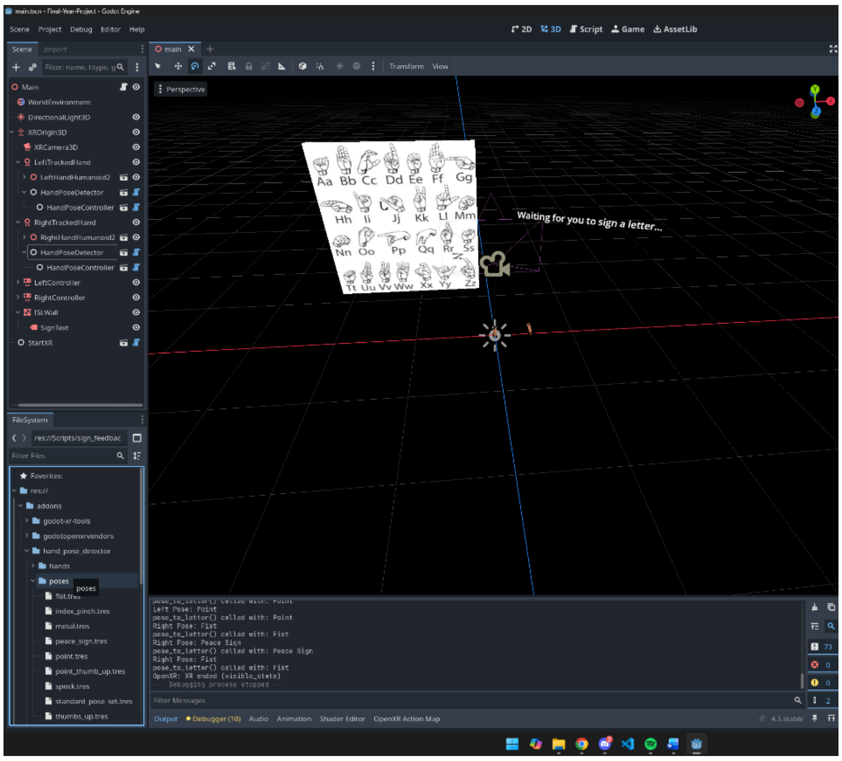


Figure 7: ISL Alphabet



```
Left Pose: Point
pose_to_letter() called with: Point
Right Pose: Fist
pose_to_letter() called with: Fist
Right Pose: Peace Sign
pose_to_letter() called with: Peace Sign
Right Pose: Fist
pose_to_letter() called with: Fist
OpenXR: XR ended (visible_state)
--- Debugging process stopped ---
```

Figure 8: Current Environment for Prototype

I installed some Godot addons, like the OpenXR plugin and the hand pose detector, to help with tracking the hands and recognizing sign shapes. Most of my time went into writing scripts that try to match the hand poses to ISL letters and show feedback in the VR scene.

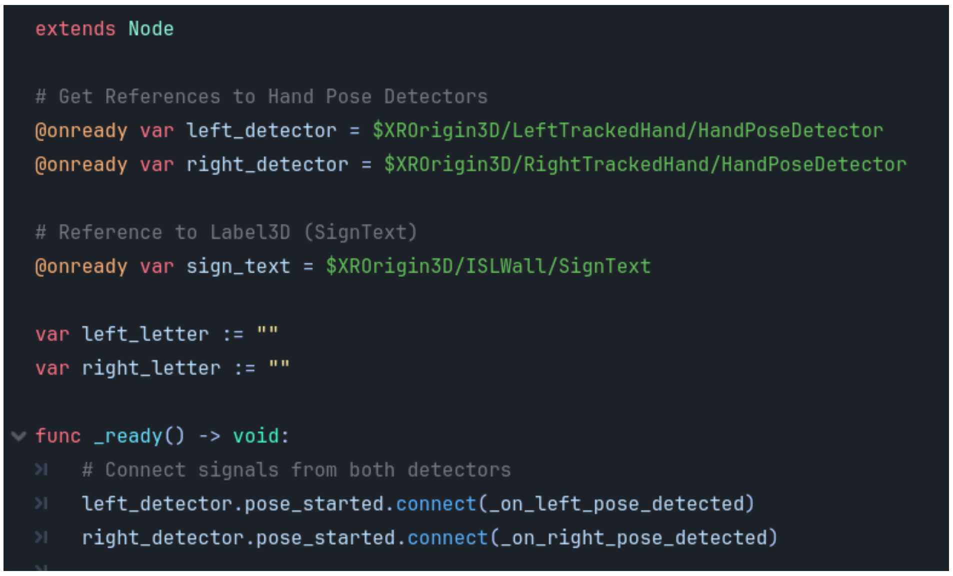
Currently, the code just handles a few signs, since some VR hand poses can look similar and I wanted to test if things worked before trying to cover every letter. My screenshots and code examples show how far I got—there's still a lot to improve, but I managed to get the basics running for this first version.

6.2 Prototype Development

For the prototype, I split my project into a few simple modules so I could work on each part without getting lost. Godot's OpenXR addon helped a lot, because it already includes built-in hand tracking and poses detection features. That saved me from having to write everything myself and let me focus on connecting the blocks in my app.

6.2.1 Hand Pose Detector Setup

I started by setting up hand tracking for both hands in the scene using the OpenXR plugin and its helper nodes. The plugin provides the basic controllers and detectors I needed, so I just referenced them in my script and connected the signals.



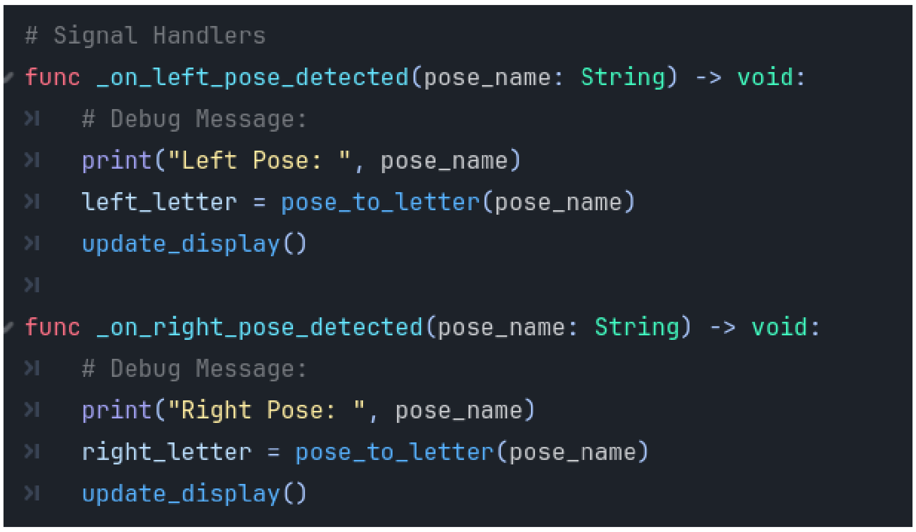
```
extends Node

# Get References to Hand Pose Detectors
@onready var left_detector = $XROrigin3D/LeftTrackedHand/HandPoseDetector
@onready var right_detector = $XROrigin3D/RightTrackedHand/HandPoseDetector

# Reference to Label3D (SignText)
@onready var sign_text = $XROrigin3D/ISLWall/SignText

var left_letter := ""
var right_letter := ""

func _ready() -> void:
	# Connect signals from both detectors
	left_detector.pose_started.connect(_on_left_pose_detected)
	right_detector.pose_started.connect(_on_right_pose_detected)
```

Figure 8: Code Snippet for Hand Pose Detector

This means the app can automatically listen for hand pose changes as soon as it starts. I didn't have to write the detector code from scratch—the OpenXR addon does it all behind the scenes.

6.2.2 Signal Handlers for Pose Detection

Whenever the user makes a new hand shape, the detectors fire signals that call these functions. This is how the system figures out which pose was made and gets ready to update the UI.



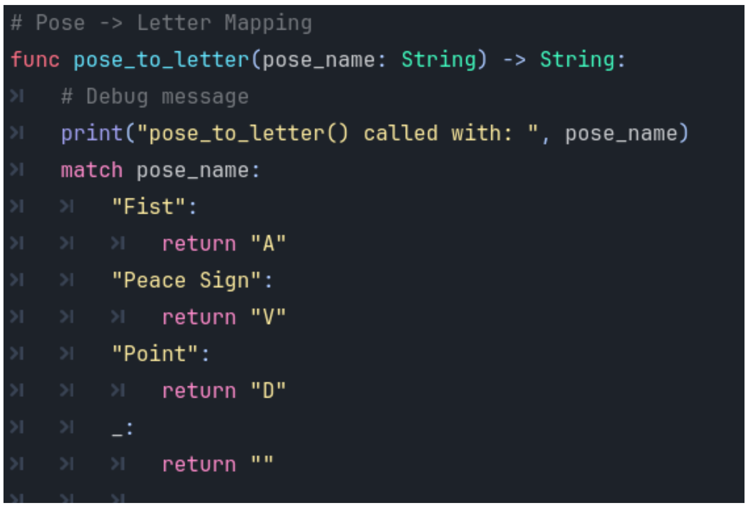
```
# Signal Handlers
func _on_left_pose_detected(pose_name: String) -> void:
	# Debug Message:
	print("Left Pose: ", pose_name)
	left_letter = pose_to_letter(pose_name)
	update_display()

func _on_right_pose_detected(pose_name: String) -> void:
	# Debug Message:
	print("Right Pose: ", pose_name)
	right_letter = pose_to_letter(pose_name)
	update_display()
```

Figure 9: Code Snippet for Signal Handling for Pose Detection

These handlers keep things simple—they react whenever a new pose is made and move on to checking which letter it might be. The debug messages help me show what pose I am doing on the terminal.

6.2.3 Mapping Hand Poses to ISL Letters

After a pose is detected, my code tries to match it to an ISL letter. Right now I only mapped three poses (fist, point, peace sign) to letters, since they were easiest to set up and test and some VR hand poses look really similar.



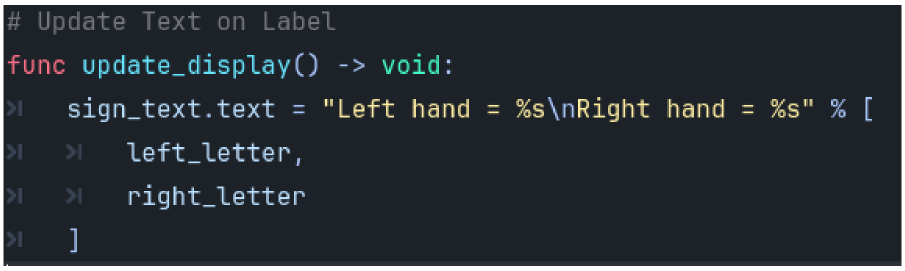
```
# Pose -> Letter Mapping
func pose_to_letter(pose_name: String) -> String:
	# Debug message
	print("pose_to_letter() called with: ", pose_name)
	match pose_name:
		"Fist":
			return "A"
		"Peace Sign":
			return "V"
		"Point":
			return "D"
		_:
			return ""
```

Figure 10: Code Snippet for Mapping Hand Poses to ISL Letters

This function is basic but does the job for now—I plan to expand it later so the app can recognise more signs and letters, and find a way to get all the letters to be not in the OpenXR Addon, I would have to make it myself.

6.2.4 Updating UI Feedback

Once the system knows what pose was made, it needs to let the user know. This small function updates the sign text label in VR, so the user can see in real time which letter was detected for each hand.



```
# Update Text on Label
func update_display() -> void:
	sign_text.text = "Left hand = %s\nRight hand = %s" % [
		left_letter,
		right_letter
	]
```

Figure 11: Code Snippet for Updating UI Feedback

Although it is not fancy, it works and lets the person practicing know if their poses match what the system expects.

6.2.5 Using Godot Addons, Environment Setup, and Hierarchy

To keep my project organized, I set up the scene using Godot's node hierarchy. The root node is called Main, which holds everything else in the environment.

Under Main, I have the WorldEnvironment and DirectionalLight3D nodes for lighting and general world setup. The XROrigin3D node is the starting point for setting up VR tracking. Inside XROrigin3D, there are nodes for the VR camera (XRCamera3D), and two main tracked hand nodes: LeftTrackedHand and RightTrackedHand.

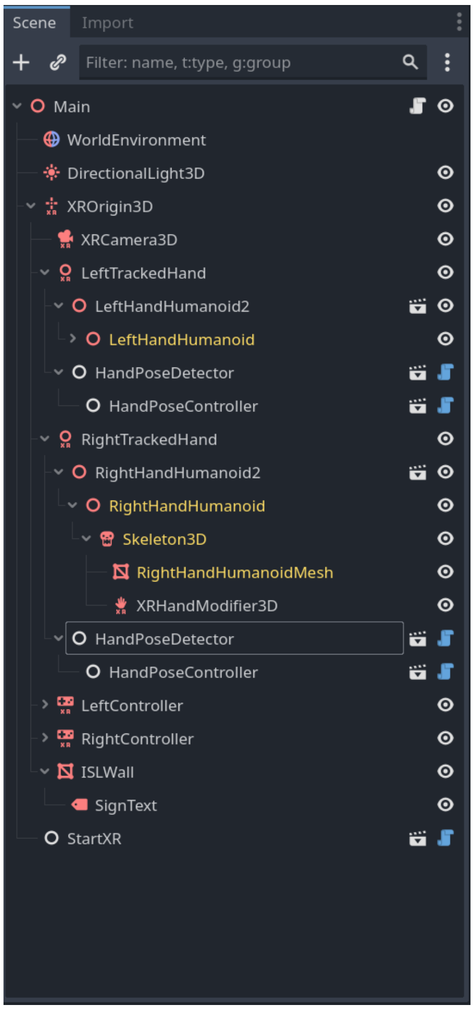
For each tracked hand, I added human hand models (LeftHandHumanoid, RightHandHumanoid), plus the important HandPoseDetector and HandPoseController nodes. These detectors and controllers handle most of the hand pose tracking and processing—the OpenXR addon already includes the code to do this automatically, so I just needed to include these nodes in my scene.

The right hand branch also has extra nodes for meshes and skeleton (Skeleton3D, RightHandHumanoidMesh, XRHandModifier3D), which help make the hand tracking more accurate and realistic.

In addition to the hands, there are simple controller nodes (LeftController and RightController), which I might use later for extra inputs.

Next, there's the ISLWall node. On this wall, I placed my sign language alphabet chart for easy reference. Attached to ISLWall is the SignText label node, which updates in real time to show the letter feedback for each hand.

Finally, StartXR is at the bottom, which handles starting up XR/VR mode in Godot.



```
Main
  WorldEnvironment
  DirectionalLight3D
  XROrigin3D
    XRCamera3D
    LeftTrackedHand
      LeftHandHumanoid2
        LeftHandHumanoid
      HandPoseDetector
        HandPoseController
    RightTrackedHand
      RightHandHumanoid2
        RightHandHumanoid
          Skeleton3D
            RightHandHumanoidMesh
            XRHandModifier3D
      HandPoseDetector
        HandPoseController
    LeftController
    RightController
    ISLWall
      SignText
  StartXR
```

Figure 12: Current Hierarchy for Prototype

6.3 Results

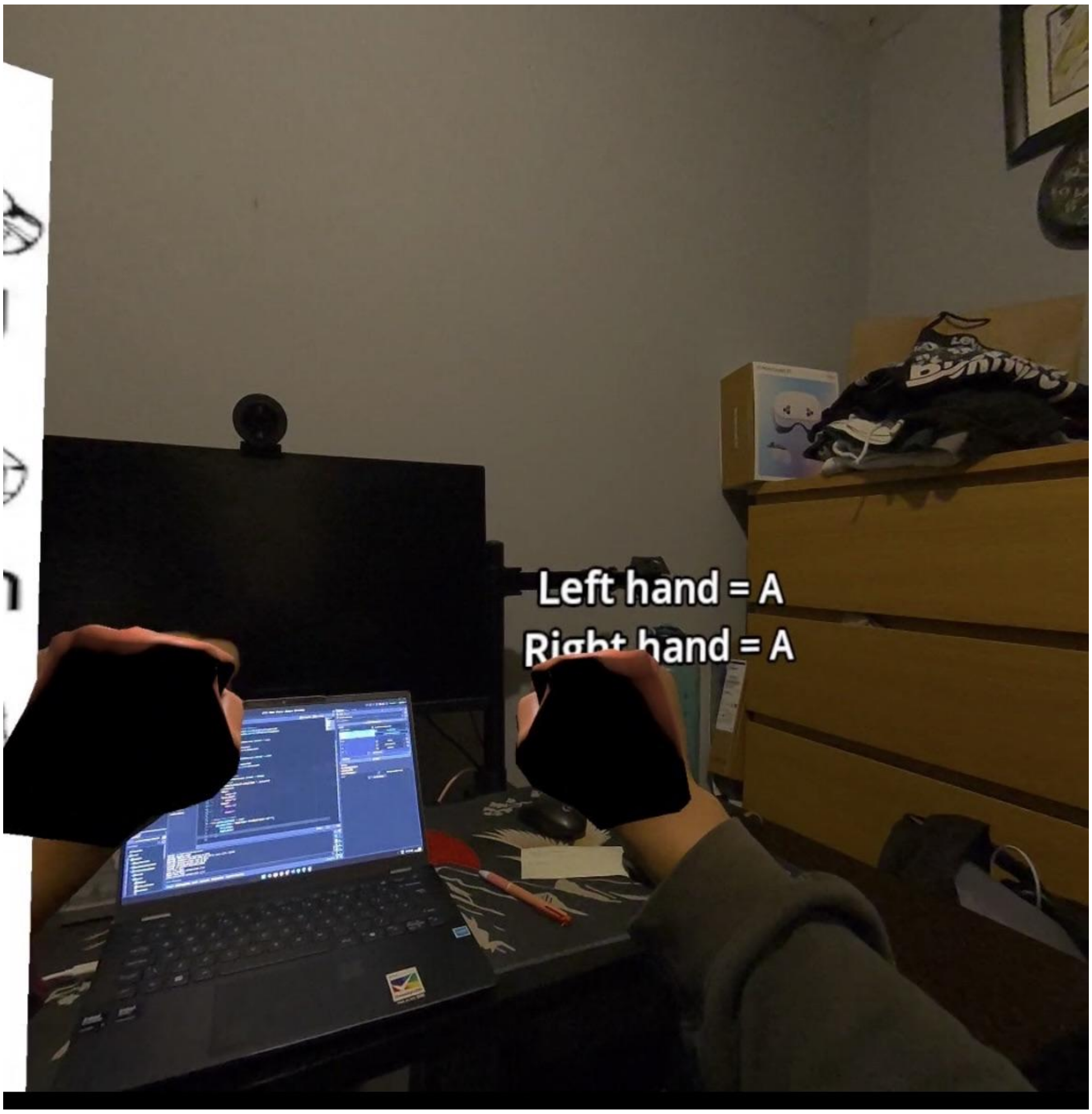


Figure 13: Test 1

This is my first test. On the ISL Alphabet, a fist is represented as “A”. It is successful as I both had my hands as a fist and the feedback says that I am signing “A”.

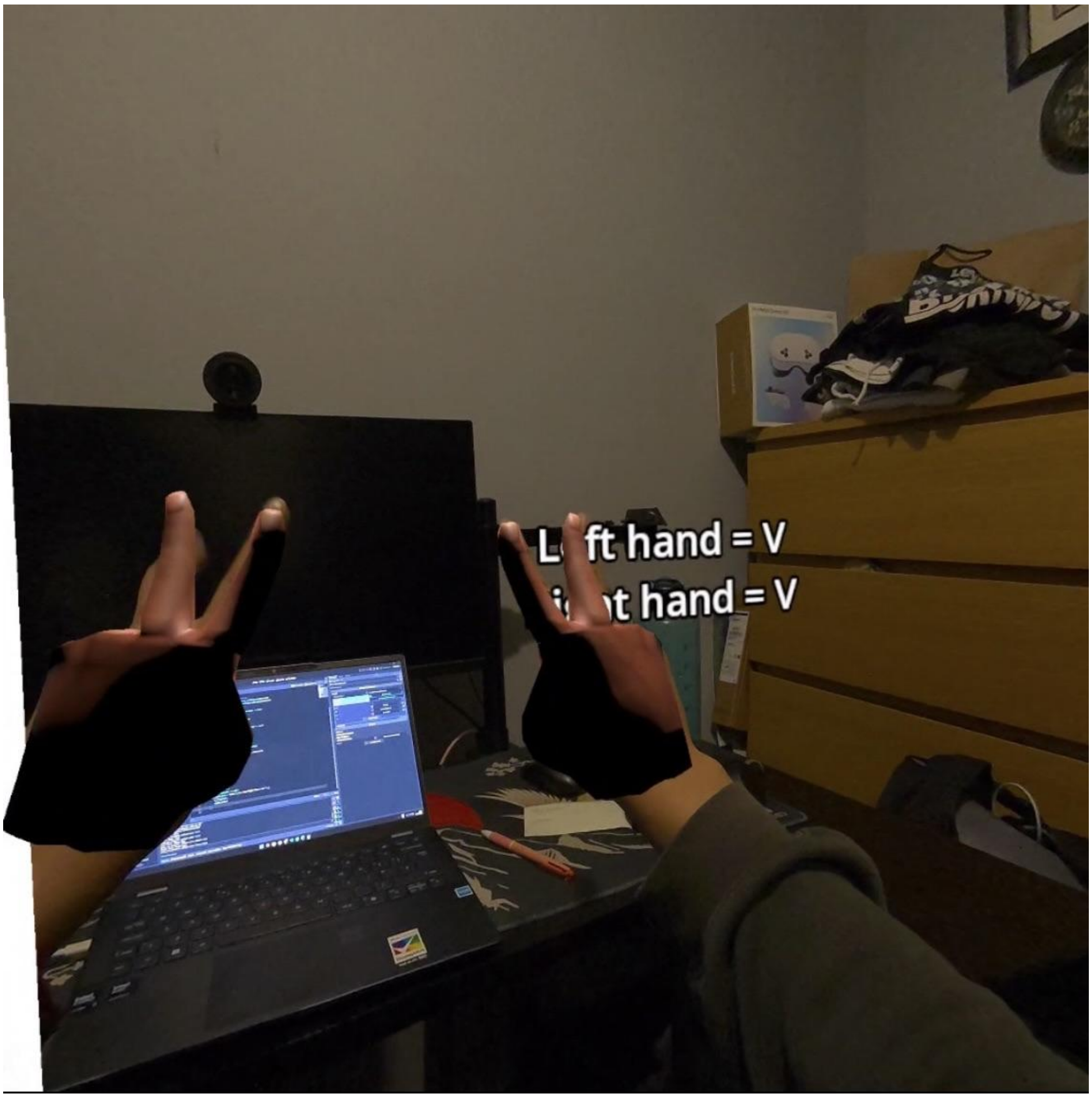


Figure 14: Test 2

This is my second test. Again, on the ISL Alphabet, posing a Piece Sign represents as "V". I both posed with both my hands and results were successful again. Both result in "V".

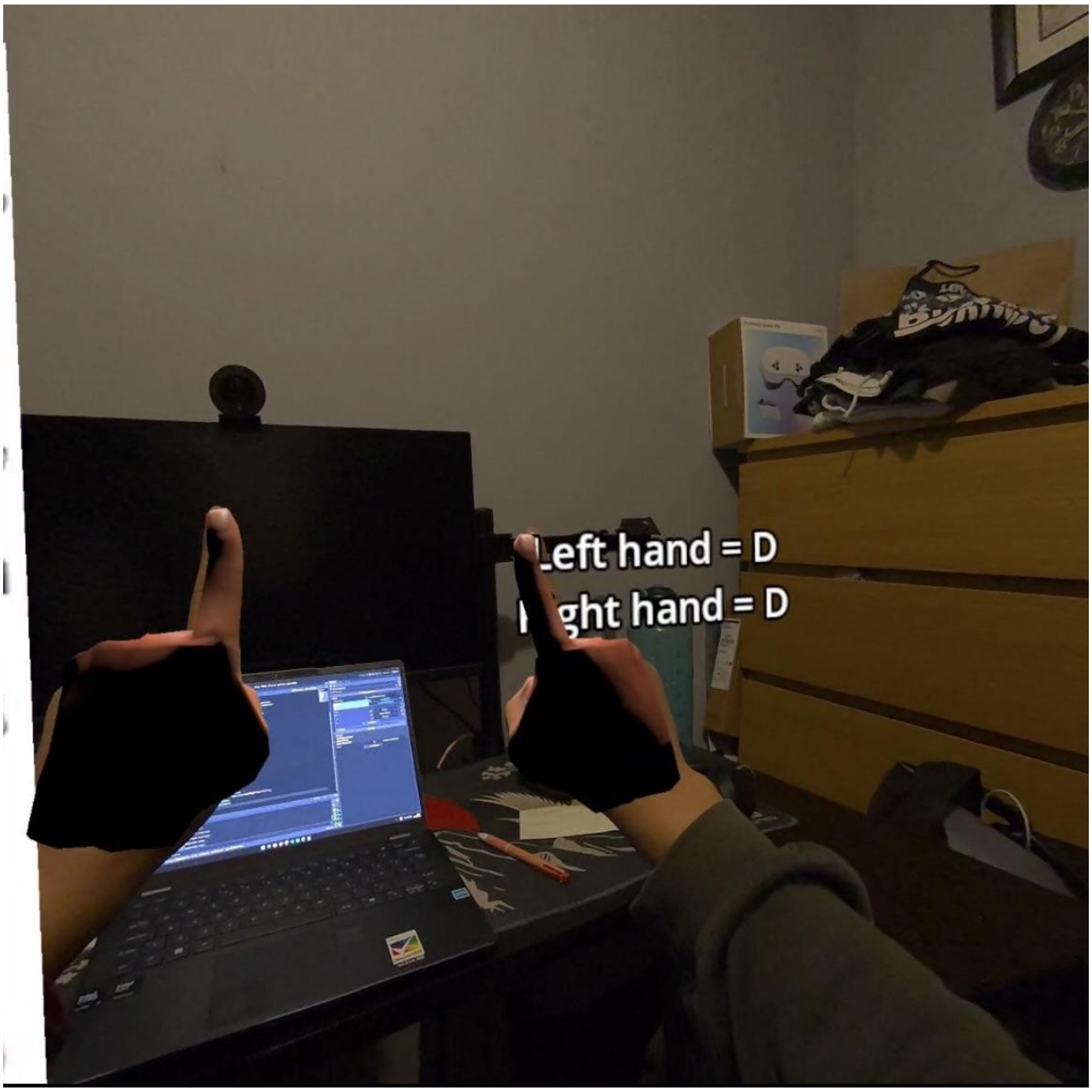


Figure 15: Test 3

This is my third test. On the ISL Alphabet, posing a Point represents as “D”. I both posed with both my hands and results were successful again. Both result in “D”.

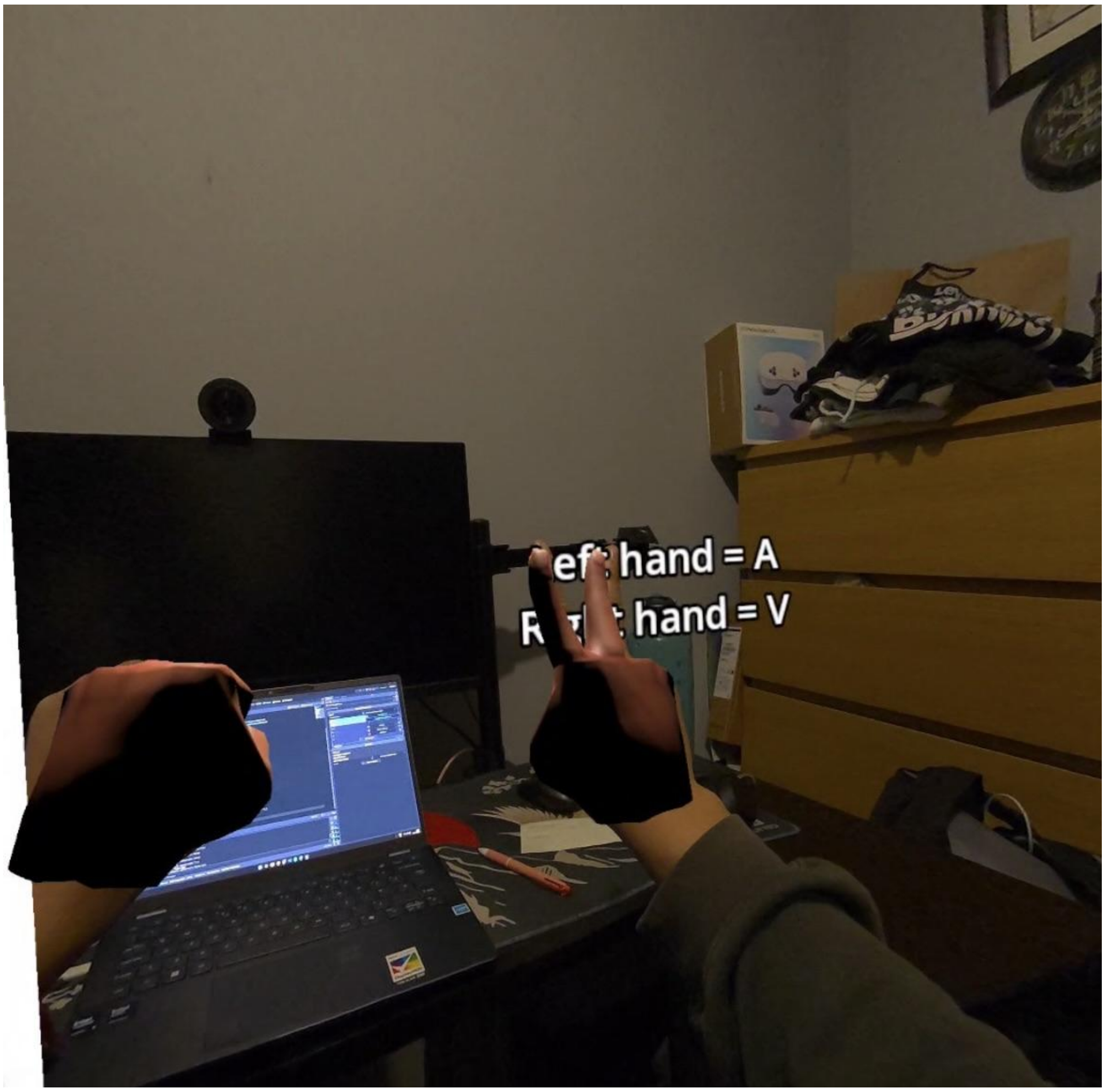


Figure 16: Test 4

This is my fourth test. I tried posing different signs on both my hand. In the image, my left hand is in a Fist which represents "A" and my right hand is a Piece Sign representing "V". The outcome is successful.

6.4 Evaluation

To evaluate my Sign Spell VR prototype, I used the testing and evaluation approaches from Section 5, focusing mainly on functional tests of hand tracking, pose recognition, feedback, and the user interface.

6.4.1 Hand Tracking Accuracy

I tested the hand tracking module by signing different ISL letters (using fist, point, and peace sign shapes) while wearing the Meta Quest 3 headset. The system was generally able to detect these core poses and print the correct letters to the screen. Sometimes, the hand pose detector would mix up similar signs (especially when finger positions weren't clear enough), but overall it worked well for basic shape recognition.

6.4.2 Feedback and UI

The feedback module successfully updated the VR sign label in real time whenever my hand changed poses. This helped me check if the system was responding instantly, which is important for a good learning experience. The alphabet chart in the VR space also made it easier to compare my hand shapes with the correct signs.

6.4.3 User Testing

I asked a few friends to try out my system, and their feedback helped me see what needs improvement. While most found the interface straightforward and easy to use once they got used to the basic gestures, they pointed out a couple of important limitations. Firstly, the app currently only supports recognition for a few letters, which made it seem limited—they suggested I should try cover the whole ISL alphabet in future versions, so the app is more useful.

Testers also noticed that the hand pose detection can depend a lot on lighting conditions in the room. Poor lighting or shadows made it harder for the system to consistently pick up accurate finger positions, resulting in some missed or incorrect letter detections.

On the positive side, everyone agreed the UI was simple and not distracting, which made it easier to focus on signing rather than figuring out controls. Their comments helped me spot bugs and understand which features to work on next.

6.4.4 Plans for Further Testing

Based on what I learned from user testing, I plan to reflect on their feedback and keep working on the main features that matter most. For my next version, I will write more automated test cases and try using the system with a wider group of users, so it's not just my friends testing it. The feedback highlighted that people want all ISL letters to work, so I aim to expand the pose recognition to cover the full alphabet and improve calibration for better accuracy. I also know now that lighting can affect hand tracking, so I'm going to try testing in different environments to make sure the system works reliably.

6.5 Conclusions

To sum up, I've managed to build a basic VR sign language system that can track hands and recognise a few signs using Godot and the Meta Quest 3. The prototype does the main things I wanted—detecting hand shapes and showing feedback on the letters—but it's still limited because it doesn't cover all the ISL alphabet yet and sometimes gets confused depending on lighting. User testing and my own experiments showed the system works okay for what it is, but there's a lot I still must improve. I plan to keep fixing things, add more features, and make the app better step by step as I learn more and get more feedback.

7. Issues and Future Work

7.1 Introduction

This section explains some of the problems or unexpected results encountered during the project so far. It also identifies risks that could affect finishing the project and outlines plans for continuing the work in the future, especially for next semester.

7.2 Issues and Risks

During development, some things worked, but a lot didn't go as planned. The hand tracking using OpenXR is still a bit dodgy—sometimes it doesn’t pick up certain gestures, and lighting in the room really affects how well it detects the hands. Because the prototype is only about halfway done, I haven’t built out all the main features yet, so I couldn’t do a full evaluation or lots of user tests.

Trying to juggle this project with other college work and assignments has also been hard. Time management was a big issue, and I ended up running out of time for some features I wanted to try. Another risk is that I only asked a few friends to test it, so I still don’t know if the app will work for everyone, especially for people new to VR or sign language. There’s also a chance that, as I add more modules (like saving data), some new bugs or crashes could happen and slow things down even more.

7.3 Plans and Future Work

Next semester, the main plan is to finish all the core features of Sign & Spell VR and make it more stable and usable. This includes improving hand tracking so it’s more accurate, make feedback system, good UI for the users so they don’t get confused, and a way to implement Data Storage.

To deal with my time management issue, I plan to organise the work into smaller tasks and spread them across the weeks of the semester. Setting clearer weekly goals should make it easier to keep track of progress and avoid leaving too much work until the end.

For evaluation and testing, I want to get more users involved, not just friends, so I can collect better feedback on how easy the system is to use and how helpful it is for learning Irish Sign Language. This might include short testing sessions with classmates or volunteers who are interested in ISL or VR. Their comments will help guide changes to the user interface and overall experience.

Fixing integration issues will also be an important part of the future work. Once more features are implemented, I will need to test how everything works together in the VR environment and look for crashes, delays, or unexpected behaviour.

7.3.1 Project Plan with GANTT Chart

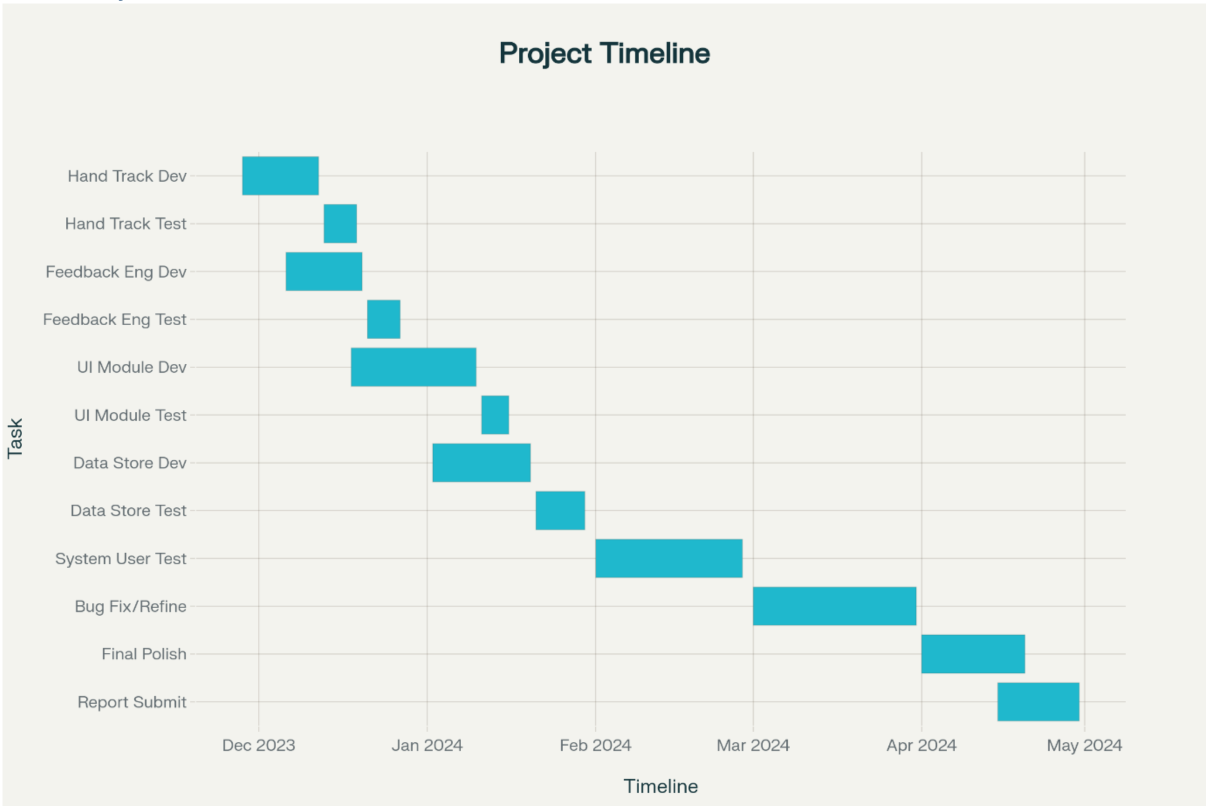


Figure 17: Gantt Chart for Next Semester Work

Based on my future plans, I will complete the project by focusing on finishing the core features first, like hand tracking, feedback, user interface, and data storage. I will develop each part step by step and test it right after to make sure it works before moving on. After these modules are working well separately, I will combine them into the full system and do more testing with users to find any problems.

To keep things realistic, I will break down my work into smaller tasks and spread them out over the weeks from now until the final deadline. I will set weekly goals, for example finishing the hand tracking module by mid-December, then its testing by the end of December, and so on for the other parts. This way I can track my progress better and avoid rushing everything at the end. Regular user testing and fixing bugs will be a big part of the schedule too.

Finally, I will leave enough time to polish the system and write the final report properly. This plan should help me finish the project on time and make sure it works as intended.

References

[1] ASL Fingerspeller – SideQuest. Available: https://sidequestvr.com/app/1317/aslfingerspeller

[2] Silent Classroom VR – Meta. Available: https://www.meta.com/experiences/silentclassroom-asl-vr/9249215595132131/

[3] ASL Champ! – ScienceDirect. Available:

https://www.sciencedirect.com/science/article/pii/S2949678024000096?

[4] DEF-ISL – App Store. Available: https://apps.apple.com/mt/app/def-isl/id6447841373

[5] D. Novaliendry et al., "Design of Sign Language Learning Media Based on Virtual Reality,"

Int. J. Online Biomedical Engineering (iJOE), vol. 19, no. 16, pp. 111–126, 2023.

https://www.researchgate.net/publication/375642351_Design_of_Sign_Language_Learning

_Media_Based_on_Virtual_Reality

[6] VR Sign Language Learning with Hand Tracking [Video]. YouTube, 2023. Available:

https://youtu.be/HeFut3Htrcw

[7] "Getting Started with Godot: The Open-Source Game Engine," *Meta.com*, 2022.

https://developers.meta.com/horizon/blog/getting-started-with-the-open-source-game-engine-godot/

[8] "XR," *Godot Engine documentation*, 2025. https://docs.godotengine.org/en/4.4/tutorials/xr/index.html

[9] Danielle Bragg, Naomi Caselli, Julie A. Hochgesang, Matt Huenerfauth, Leah Katz-Hernandez, Oscar Koller, Raja Kushalnagar, Christian Vogler, and Richard E. Ladner. 2021. The FATE Landscape of Sign Language AI Datasets: An Interdisciplinary Perspective. ACM Trans. Access. Comput. 14, 2, Article 7 (June 2021), 45 pages. https://doi.org/10.1145/3436996

[10] Danielle Bragg, Oscar Koller, Mary Bellard, Larwan Berke, Patrick Boudreault, Annelies Braffort, Naomi Caselli, Matt Huenerfauth, Hernisa Kacorri, Tessa Verhoef, Christian Vogler, and Meredith Ringel Morris. 2019. Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective. In Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19). Association for Computing Machinery, New York, NY, USA, 16–31. https://doi.org/10.1145/3308561.3353774

[11] Sushant Kafle, Abraham Glasser, Sedeeq Al-khazraji, Larwan Berke, Matthew Seita, and Matt Huenerfauth. 2020. Artificial intelligence fairness in the context of accessibility research on intelligent systems for people who are deaf or hard of hearing. SIGACCESS Access. Comput., 125, Article 4 (October 2019), 1 pages. https://doi.org/10.1145/3386296.3386300

[12] "WFD AND WASLI STATEMENT ON USE OF SIGNING AVATARS Click here for International Sign Version." Accessed: Nov. 06, 2025. [Online]. Available: https://wasli.org/wp-content/uploads/2023/07/WFD-and-WASLI-Statement-on-Avatar-FINAL-14032018-Updated-14042018-1.pdf

[13] "Saving games," *Godot Engine documentation*. https://docs.godotengine.org/en/stable/tutorials/io/saving_games.html

[14] L. Jones, "Using Meta Quest Headsets in Education: 3 Powerful Ways Educators Can Get Started," *Thinglink.com*, Oct. 27, 2025. https://www.thinglink.com/blog/using-meta-quest-headsets-in-education-3-powerful-ways-educators-can-get-started/

[15] Y. Ding and J. Hao, "The Application of Educational Games Based on Virtual Reality Technology in Teaching English in Universities," *Systems and Soft Computing*, p. 200380, Aug. 2025, doi: https://doi.org/10.1016/j.sasc.2025.200380.

[16] Z. Zhang, "Innovative Application and User Experience of Virtual Reality Technology in Human-Computer Interaction," *Highlights in Science, Engineering and Technology*, vol. 93, pp. 200–209, May 2024, doi: https://doi.org/10.54097/mx8yf812.

[17] M. Golubski, "VR Music Learner Final Project Report," BSc dissertation, School of Computer Science, Technological University Dublin, Mar. 2022.

[18] A. Botnari, “Sign Language Translator (SLT): Final Year Project,” BSc dissertation, School of Computing, Department of Science and Health, Technological University Dublin, 2017.

A) Appendix A: System Model and Analysis

Details of Requirements gathering and analysis method

B) Appendix B: Design

Details of design approach used

System elements, components, classes

C) Appendix C: Prompts Used with ChatGPT

Literature Review:

Prompt: “Could you suggest what I should write in this section—including relevant subheadings and topics for discussion—when structuring my literature review?”

Technologies Researched:

Prompt: “What should I write in this section, and what headings or subheadings would be appropriate for discussing the technologies I considered?”

System Analysis:

Prompt: “What are the important points to include in system analysis? Please advise on the main subheadings and structure for the section.”

System Design:

Prompt: “Tell me what I should write here, listing suitable headings and subheadings for an effective system design section.”

Prototype Development & Results:

Prompt: “Could you specify what needs to be included in the prototype and results section, as well as recommend potential subheadings and topics?”

Testing & Evaluation:

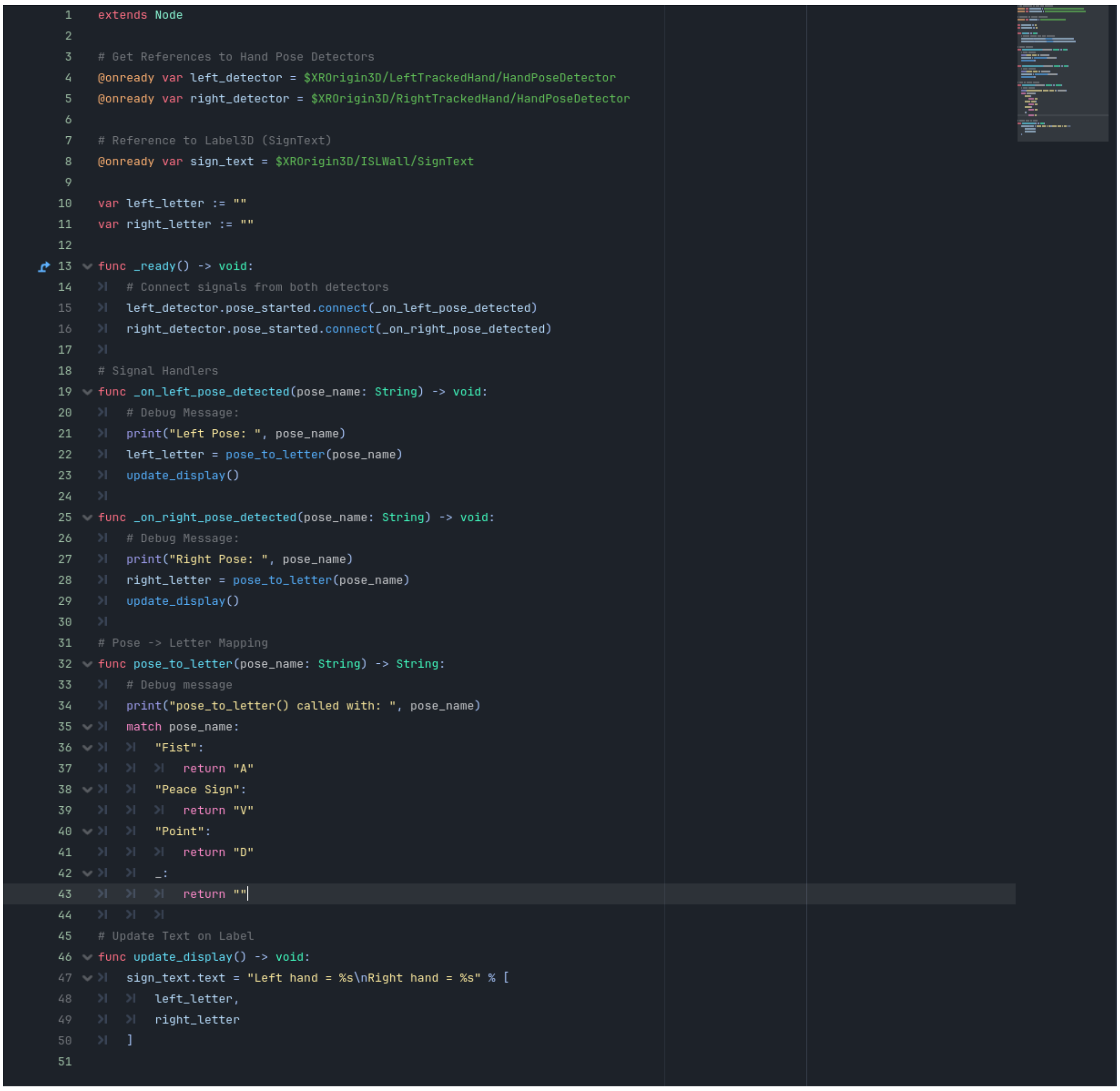
Prompt: “What should I cover while writing about testing and evaluation, and what subheadings or main headings would best organize this content?”

Issues and Future Work:

Prompt: “Suggest what to write about current issues and future improvements, including possible headings or subsections for clarity.”

What are good diagrams to show for this report (aka class diagram, DFD, etc)

D) Appendix D: Additional Code Samples



```
extends Node

# Get References to Hand Pose Detectors
@onready var left_detector = $XROrigin3D/LeftTrackedHand/HandPoseDetector
@onready var right_detector = $XROrigin3D/RightTrackedHand/HandPoseDetector

# Reference to Label3D (SignText)
@onready var sign_text = $XROrigin3D/ISLWall/SignText

var left_letter := ""
var right_letter := ""

func _ready() -> void:
	# Connect signals from both detectors
	left_detector.pose_started.connect(_on_left_pose_detected)
	right_detector.pose_started.connect(_on_right_pose_detected)
	
# Signal Handlers
func _on_left_pose_detected(pose_name: String) -> void:
	# Debug Message:
	print("Left Pose: ", pose_name)
	left_letter = pose_to_letter(pose_name)
	update_display()
	
func _on_right_pose_detected(pose_name: String) -> void:
	# Debug Message:
	print("Right Pose: ", pose_name)
	right_letter = pose_to_letter(pose_name)
	update_display()
	
# Pose -> Letter Mapping
func pose_to_letter(pose_name: String) -> String:
	# Debug message
	print("pose_to_letter() called with: ", pose_name)
	match pose_name:
		"Fist":
			return "A"
		"Peace Sign":
			return "V"
		"Point":
			return "D"
		_:
			return ""
			
# Update Text on Label
func update_display() -> void:
	sign_text.text = "Left hand = %s\nRight hand = %s" % [
		left_letter,
		right_letter
	]

```