

**TECHTUTOR: AR-INTEGRATED LEARNING APPLICATION FOR
COMPUTER AND TECHNOLOGY USE**

BY
LOH CHIA HEUNG

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

© 2025 Loh Chia Heung. All rights reserved.

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr. Ng Hui Fuang, for his guidance, patience, and encouragement throughout this project. He has not only provided me with valuable advice and technical support but also given me the freedom to explore new ideas in the areas of Augmented Reality (AR) and Artificial Intelligence (AI). This project has been a meaningful learning journey for me, and without his support, it would not have been completed smoothly.

I am also deeply grateful to my parents and family for their constant love, care, and encouragement. Their patience and understanding gave me the strength to push forward whenever I felt overwhelmed.

A special thanks goes to my friends, who have always been there during my difficult times, reminding me not to give up and motivating me to keep moving forward. Their encouragement, whether through small words or simple companionship, made a big difference in keeping me focused and positive throughout this journey.

Lastly, I would like to thank everyone who has, directly or indirectly, supported me in completing this project. Each contribution, no matter how small, has played a part in helping me reach this stage.

ABSTRACT

In today's digital era, the ability to operate basic computer technologies has become increasingly important. However, many beginners and older adults continue to face difficulties in learning digital tools such as email, word processors, and calculators, mainly due to the lack of accessible and interactive resources. This project addresses this issue by developing TechTutor, an Augmented Reality (AR)-integrated and Artificial Intelligence (AI)-assisted mobile learning application designed to provide a simple, intuitive, and engaging platform for digital literacy. TechTutor was developed using Unity 2022.3.57f1, AR Foundation (ARCore and Vuforia), and OpenAI integration, following the Agile Software Development methodology. The system consists of 3 main components: **AR Learn**, which enables users to identify computer components, interact with 3D models, and attempt AR-based quizzes; **AR Guide**, which provides step-by-step AR tutorials for tasks such as sending an email, using Microsoft Word, the Snipping Tool and a calculator; **AI Tutor**, which offers chatbot-based assistance with both text and image inputs, enhanced by EasyOCR for keyword detection and contextual guidance. Speech-to-Text (STT) and Text-to-Speech (TTS) technologies were integrated across the modules to enable hands-free navigation, interactive feedback, and accessibility for diverse user needs. The novelty of TechTutor lies in combining AR-driven experiential tutorials and AI-powered assistance with keyword highlighting, creating an adaptive learning environment tailored for users with minimal prior exposure to computers and technology. By delivering immersive, interactive, and personalized guidance, TechTutor aims to reduce the digital divide and promote inclusive technology use among underserved groups, particularly the beginners and older adults.

Area of Study: Augmented Reality (AR), Artificial Intelligence (AI)

Keywords: Digital Literacy, Augmented Reality Learning, Interactive Tutorials, OpenAI Integration, Speech-to-Text (STT), Text-to-Speech (TTS), EasyOCR, Unity Development

TABLE OF CONTENTS

TITLE PAGE	i
COPYRIGHT STATEMENT	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	ix
LIST OF ABBREVIATIONS	x
CHAPTER 1 – INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	3
1.3 Project Scope	4
1.4 Contributions	5
1.5 Background Information	6
1.6 Report Organisation	7
CHAPTER 2 – LITERATURE REVIEW	8
2.1 Review of the Technologies	8
2.1.1 Augmented Reality (AR)	8
2.1.2 Augmented Reality (AR) Development Platform	9
2.1.3 Augmented Reality (AR) Manuals	10
2.1.4 Optical Character Recognition (OCR)	11
2.2 Review of Previous Works	14
2.2.1 Quiver – 3D Colouring App	14
2.2.2 Visible Body	16
2.2.3 ARChem	18
2.2.4 Jigspace	20
2.2.5 Assemblr EDU	22
2.2.6 EasyOCR vs Tesseract OCR	24
2.3 Comparison of Previous Works	26

CHAPTER 3 – SYSTEM METHODOLOGY	29
3.1 Methodology	29
3.2 System Requirements	32
3.2.1 Hardware	32
3.2.2 Software	33
3.3 Functional Requirements	35
3.4 Non-Functional Requirements	38
3.5 Project Timeline	40
 CHAPTER 4 – SYSTEM DESIGN	 41
4.1 System Flowchart	41
4.2 Use Case Diagram	42
4.3 System Architecture	44
4.4 Activity Diagram	47
 CHAPTER 5 – SYSTEM IMPLEMENTATION	 53
5.1 Software Setup	53
5.1.1 Unity Setup	53
5.1.2 Vuforia Developer Portal	54
5.1.3 Vuforia Engine AR Package	54
5.1.4 ARCore XR Plugin	55
5.1.5 Speech-to-Text (STT) Plugin	56
5.1.6 Native Camera	56
5.1.7 Native Gallery	57
5.1.8 AR Foundation and XR Plugin Management	58
5.1.9 Android Studio	58
5.1.10 Sketchfab	59
5.1.11 GitHub	60
5.1.12 Meshy.ai	60
5.1.13 Figma	61
5.1.14 Icons8	61

5.2	Hardware Setup	62
5.3	System Configuration	63
5.3.1	Vuforia Engine Configuration	63
5.3.2	ARCore & AR Foundation Configuration	65
5.3.3	OpenAI API Configuration	66
5.3.4	EasyOCR Configuration	68
5.3.5	System Prompt Engineering for OpenAI	70
5.4	System Operation	75
5.4.1	Main Menu Scene	75
5.4.2	AR Learn	76
5.4.3	AR Guide	103
5.4.4	AI Tutor	117
5.5	Implementation Issues and Challenges	128
5.6	Concluding Remark	129
CHAPTER 6 – SYSTEM EVALUATION AND DISCUSSION		130
6.1	System Evaluation and Discussion	130
6.2	Testing Setup and Result	130
6.3	Project Challenges	148
6.4	Objectives Evaluation	149
6.5	Concluding Remark	150
CHAPTER 7 – CONCLUSION AND RECOMMENDATIONS		151
7.1	Conclusion	151
7.2	Recommendations	152
7.3	Closing Remark	154
REFERENCES		155
APPENDICES		A-1
Appendix A: Poster		A-1
Appendix B: Credits for 3D Models		B-1

Appendix C: Completed Tutorial Steps for Microsoft Word	C-1
Appendix D: Completed Tutorial Steps for Sending Email (Gmail)	D-1
Appendix E: Completed Tutorial Steps for Snipping Tool	E-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	Reality-Virtual Continuum Schematic [17]	9
Figure 2.2	Example of Usage of AR Manuals [24]	11
Figure 2.3	Example of CAPTCHA usage in OCR [43]	12
Figure 2.4	Example of CRAFT Detecting Texts in Horizontal, Curved, and Arbitrary Shapes [44]	13
Figure 2.5	Architecture of Convolutional Recurrent Neural Network (CRNN) for Text Recognition [45]	14
Figure 2.6	Logo of Quiver – 3D Colouring App [25]	15
Figure 2.7	Examples of Marker-based Coloring Papers [26, 27]	15
Figure 2.8	Logo of Visible Body [31]	17
Figure 2.9	Example of Usage of AR Technology in Visible Body [29, 30]	17
Figure 2.10	Logo of ARChem [32]	19
Figure 2.11	An Example Screenshot of Demonstration using Two Marker Papers [34]	19
Figure 2.12	Logo of JigSpace [35]	21
Figure 2.13	Example of Creation of ‘Jig’ [36]	21
Figure 2.14	Logo of Assemblr EDU [40]	23
Figure 2.15	Customizable Image Markers [39]	23
Figure 2.16	Comparison of Character Recognition Results Between EasyOCR and Tesseract OCR [46]	25
Figure 3.1	Agile Software Development Approach [41]	30
Figure 3.2	Gantt Chart for FYP 1	40
Figure 3.3	Gantt Chart for FYP 2	40
Figure 4.1	System Flowchart of TechTutor	42
Figure 4.2	Use Case Diagram of TechTutor	43
Figure 4.3	System Architecture Diagram of TechTutor	44

Figure 4.4	Activity Diagram for AI Identify Submodule	47
Figure 4.5	Activity Diagram for AR Models Submodule	48
Figure 4.6	Activity Diagram for AR Quiz Submodule	49
Figure 4.7	Activity Diagram for AR Guide Tutorials (Microsoft Word, Snipping Tool, Send Email)	50
Figure 4.8	Activity Diagram for AR Guide – Calculator Tutorial	51
Figure 4.9	Activity Diagram for AI Assistance Submodule	52
Figure 5.1	Unity 2022.3.57f1(LTS)	53
Figure 5.2	Vuforia Developer Portal Setup	54
Figure 5.3	Vuforia Engine AR Package Setup	55
Figure 5.4	ARCore XR Plugin Setup	55
Figure 5.5	Speech-to-Text (STT) Plugin Setup	56
Figure 5.6	Native Camera Plugin Setup	57
Figure 5.7	Native Gallery Plugin Setup	57
Figure 5.8	AR Foundation and XR Plugin Management Setup	58
Figure 5.9	Android Studio Logcat Window	59
Figure 5.10	Android Studio Version	59
Figure 5.11	Sketchfab Website	59
Figure 5.12	GitHub Repository for TechTutor	60
Figure 5.13	AR Robot Generated from Meshy.ai	60
Figure 5.14	Figma Workspace	61
Figure 5.15	Icons8 Website Interface	61
Figure 5.16	Generate License Key	63
Figure 5.17	Obtained License Key	63
Figure 5.18	Open Vuforia Engine Configuration	64
Figure 5.19	Paste License Key	64
Figure 5.20	Image Target Behaviour Configuration	65
Figure 5.21	Activated Google ARCore	65
Figure 5.22	AR Session and XR Origin (Mobile AR) Prefabs	66
Figure 5.23	Creating OpenAI Secret Key	67
Figure 5.24	Generated OpenAI Secret Key	67

Figure 5.25	OpenAI API Integration using UnityWebRequest	68
Figure 5.26	Flask Server Code	69
Figure 5.27	Flask Server Running	70
Figure 5.28	AR Quiz Configuration Code Snippet	71
Figure 5.29	AI Tutor Configuration Code Snippet	72
Figure 5.30	AI Identify System Prompt Section	74
Figure 5.31	AI Identify Configuration Code Snippet	74
Figure 5.32	Main Menu Screen	75
Figure 5.33	AR Learn Module Main Screen	76
Figure 5.34	AI Identify Main Screen	77
Figure 5.35	Capture Scene	77
Figure 5.36	OpenAI Response	78
Figure 5.37	Mouse Info Panel	78
Figure 5.38	Capture Scene for Keyboard	79
Figure 5.39	Updated Navigation Content	79
Figure 5.40	Accessing the History List	80
Figure 5.41	History List Interface	80
Figure 5.42	Keyboard Information Panel	81
Figure 5.43	AR Models Main Screen	81
Figure 5.44	Plane Detection	82
Figure 5.45	Model Selection Interface	82
Figure 5.46	Keyboard Model Selected	83
Figure 5.47	Enlarged Preview of Keyboard Model	83
Figure 5.48	Keyboard Model Placed	84
Figure 5.49	Information Panel Displayed	84
Figure 5.50	Key Labels Overlay	85
Figure 5.51	Movement of Keyboard Model	85
Figure 5.52	Position Reset	86
Figure 5.53	Arrow Keys Information Panel	86
Figure 5.54	Information Panel for Other Keys	87
Figure 5.55	Selection Another Model	87
Figure 5.56	Information Panel and Labels for Monitor Model	88

Figure 5.57	Information Panel for All Components (Monitor)	89
Figure 5.58	Information Panel and Labels for Mouse Model	90
Figure 5.59	Information Panel for All Components (Mouse)	90
Figure 5.60	Information Panel for Speaker Model	91
Figure 5.61	Speech-to-Text Action	92
Figure 5.62	Information Panel for Laptop Model (Laptop)	93
Figure 5.63	Information Panel for All Components (Laptop)	94
Figure 5.64	AR Quiz Marker	94
Figure 5.65	AR Quiz Main Screen	95
Figure 5.66	Model Selection Interface on Marker	95
Figure 5.67	Model Switching	96
Figure 5.68	Quiz Panel Spawned	97
Figure 5.69	Correct and Wrong Label Feedback	97
Figure 5.70	Finished Quiz Message	98
Figure 5.71	Unchoose Model Action	99
Figure 5.72	Triggering the Generate Quiz Button	99
Figure 5.73	Second Feedback Label	100
Figure 5.74	New Question Generated by OpenAI	100
Figure 5.75	Other Questions Generated by OpenAI	101
Figure 5.76	Quiz History List	102
Figure 5.77	Quiz Detail Panel	103
Figure 5.78	AR Guide Main Scene	104
Figure 5.79	Plane Detection in AR Guide	105
Figure 5.80	Desk Simulation Model Placement	106
Figure 5.81	Information Panel Control Options	107
Figure 5.82	Whole Model Control Panel	108
Figure 5.83	Tutorial Progression with Navigation Buttons	109
Figure 5.84	Monitor Screen and Visual Cues	110
Figure 5.85	Change Mode and Canvas Synchronization (Canvas Mode)	111
Figure 5.86	Switching Back to Desk Simulation Mode	112
Figure 5.87	Main Screen and Info Panel of Calculator Tutorial	114

Figure 5.88	Tutorial Steps in Calculator Tutorial	114
Figure 5.89	AR Learn Module Main Screen	117
Figure 5.90	AI Assistance Main Screen	118
Figure 5.91	Click the Gallery Button	119
Figure 5.92	Select Photo from the Gallery	120
Figure 5.93	Selected Image from the Gallery	121
Figure 5.94	Clicking to Enlarge the Image	122
Figure 5.95	User Prompt with Question and Send Action	122
Figure 5.96	OpenAI Response	123
Figure 5.97	Canvas Mode	124
Figure 5.98	Highlight Text using EasyOCR	124
Figure 5.99	Remaining Steps with EasyOCR	125
Figure 5.100	Chat History List	126
Figure 5.101	Chat History Detail Panel	126
Figure 5.102	Lists of All FAQ Questions	127

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Comparison between EasyOCR and Tesseract OCR	25
Table 2.2	Comparison Table of Existing Applications vs TechTutor (Proposed Project)	26
Table 3.1	Sprint Cycles for Each Sprint	31
Table 3.2	Hardware Specifications	32
Table 3.3	Software Specifications	33
Table 5.1	Laptop Specifications	62
Table 5.2	Mobile Device Specifications	62
Table 6.1	AI Identify Test Case	131
Table 6.2	AR Models Test Case	133
Table 6.3	AR Quiz Test Case	135
Table 6.4	Simple Calculator Test Case	138
Table 6.5	Microsoft Word Test Case	139
Table 6.6	Send Email Test Case	141
Table 6.7	Snipping Tool Test Case	144
Table 6.8	AI Assistance Test Case	146

LIST OF ABBREVIATIONS

<i>AR</i>	Augmented Reality
<i>AI</i>	Artificial Intelligence
<i>3D</i>	Three-Dimensional
<i>2D</i>	Two-Dimensional
<i>API</i>	Application Programming Interface
<i>UI</i>	User Interface
<i>XR</i>	Extended Reality
<i>VR</i>	Virtual Reality
<i>RE</i>	Real Environment
<i>AV</i>	Augmented Virtuality
<i>STT</i>	Speech-to-Text
<i>TTS</i>	Text-to-Speech
<i>CRAFT</i>	Character Region Awareness for Text Detection
<i>CRNN</i>	Convolutional Recurrent Neural Network

Chapter 1

Introduction

This chapter introduces the foundation of the project by presenting the problem statement and motivation behind the development of the TechTutor. It outlines the project objectives, scope, contributions, and background information followed by the organisation of the report. These sections provide a clear roadmap for readers to understand the purpose, direction, and the significance of this project.

1.1 Problem Statement and Motivation

1. Digital Literacy Gap

In today's advanced digital world, computer technology plays an essential role in the daily life. While younger generations have grown up surrounded by digital devices, there remains a significant gap in helping users, particularly older adults to become comfortable with modern technology. Although many older adults are aware of the capabilities and features of digital tools, they often struggle to utilize them effectively [6]. According to the Pew Research Center [1], 65% of seniors reported experiencing difficulties with using modern technology, mainly due to the **lack of simple, clear guidance and support** [2]. This situation often leads to frustration and discouragement when learning new technologies.

2. Challenges in Basic Computer Skills

Despite the availability of educational resources, many current computer learning applications **fail to offer an interactive, step-by-step approach** necessary to simplify technology use for older adults [3]. Designing educational tools for this demographic requires **careful consideration of factors** such as vision, hearing limitations, simplicity of design, and the user's prior experience to ensure the usability and effectiveness of system [4]. Older adults, particularly those without extensive digital exposure may need user interfaces that **do not assume prior knowledge** and are **tailored to their needs**.

Furthermore, basic computer operations, such as using Microsoft Word and sending emails remain challenging for many beginners. Research shows that older adults take

significantly longer time to learn word processors like Microsoft Word, and often struggle with email tasks without detailed guidance [5], [6]. Specific difficulties include navigating menus, understanding functions like the “Subject” field in email, and confidently executing basic operations [7]. These challenges, if unresolved, may contribute to feelings of isolation, digital exclusion, and reduced quality of life for the older adults [8].

3. Limitations of Existing Learning Methods

In addition to issues faced by older adults, broader challenges exist within the **education field** itself. Traditional teaching methods often rely heavily on **passive learning approaches**, such as lectures, textbooks, and static online materials, which can **limit student engagement** and **hands-on understanding** [9]. Practical fields, especially digital literacy, require more **experiential** and **interactive learning** methods to bridge the gap between theoretical knowledge and real-world application. Moreover, the success of the learning process depends not only on instructional methods but also on the **quality of the learning media utilized**.

Studies have highlighted that the **integration of appropriate technological tools** can enhance the effectiveness and efficiency of teaching, therefore educators are increasingly expected to master technological tools [13]. For instance, the use of ICT in classrooms has been shown to help students access information more efficiently and support more student-centred, creative learning experiences [42].

Motivation

To address these problems, the project proposes the development of an **Augmented Reality (AR)-integrated learning application** known as **TechTutor**. AR technology offers an immersive, interactive learning environment by overlaying digital content onto the real world, enabling users to experience and interact with virtual objects in real time. According to [10], previous studies have shown that AR **enhances the understanding, increases learner motivation, and improves experiential learning outcomes**.

Motivated by these findings, this project aims to create a simple, intuitive, and adaptable learning platform that supports different learning speeds and styles. Through interactive 3D computer components, simulated software interfaces, and guided

tutorials delivered via mobile devices, the system aims to make digital learning more approachable and less overwhelming. By combining AR-based step-by-step tutorials with AI-driven assistance, TechTutor specifically addresses the digital literacy gap among the older adults, while remaining inclusive for users of all ages who wish to strengthen their basic computer skills. Therefore, this project seeks to bridge the digital divide by leveraging the AR and AI technologies to provide an interactive, inclusive, and accessible learning platform.

1.2 Objectives

The main purpose of this project is to design and develop TechTutor, an Augmented Reality (AR)-integrated and AI-assisted mobile application that provides an intuitive platform for users of all ages, particularly beginners and older adults, to learn essential computer and technology skills. The project focuses on delivering a solution that teaches basic computer skills, simplifies technology use through interactive, step-by-step learning experiences.

- **To develop an AR-based Learning Modules, AR Learn and AR Guide.**

The main objective of these modules is to enable users to recognize and interact with virtual computer components and follow step-by-step tutorials for basic tasks such as sending emails, using Microsoft Word, and operating calculators. These modules combine digital overlays, 3D models, synchronized narration and visual cues to create immersive and engaging learning experiences.

- **To develop an AI Tutor Module.**

This module provides real-time support and personalized guidance through chatbot interaction, image-based queries, and OCR-powered keyword highlighting and navigating. It ensures that users can access on-demand explanations and receive context-aware assistance beyond the AR tutorials.

- **To implement adaptive learning features.**

The system includes quiz history, narration review, and flexible navigation using Speech-to-Text (STT) and Text-to-Speech (TTS). These features allow

users to progress at their own pace, support different learning speeds and styles, and enhance inclusivity for users from diverse backgrounds and abilities.

1.3 Project Scope

This project focuses on the design and development of **TechTutor**, an **Augmented Reality (AR)-integrated** and **AI-assisted** mobile application aimed at improving the digital literacy. The scope of the project is defined as follows.

- **AR Learn**

This module enables users to recognize and interact with **virtual computer components** such as the keyboard, mouse, monitor, laptop, and speaker. Through **3D models, labelling, and narration**, users can explore hardware functions in an engaging way. Additionally, an **AR Quiz** feature is included to assess knowledge through **AI-generated questions** in a marker-based AR environment, with **quiz history** stored for later review.

- **AR Guide**

This module provides **step-by-step tutorials** for performing basic computer tasks, including sending emails, using Microsoft Word, operating the Snipping Tool, and performing basic calculator operations. Tutorials are delivered through **synchronized narration, visual cues, and interactive overlays**. Two viewing modes are provided to suit different user needs.

- The first mode is **3D AR Mode**, which uses interactive models and is most suitable for complete beginners who benefit from immersive, hands-on visualization.
- The second mode is **Canvas Mode**, which presents simplified panel-based guidance, designed for users who already have some familiarity and prefer concise step-by-step instructions.

At the same time, **Speech-to-Text (STT)** is supported for hands-free navigation commands such as ‘Next’ or ‘Previous’.

- **AI Tutor**

This module offers **real-time assistance** and **personalized guidance**. Users can interact with the system through **text or image queries**. With **integrated Optical Character Recognition (OCR)**, the system identifies keywords from

screenshots and highlights them within the tutorial interface. This allows users to receive **contextual guidance** in addition to just chatbot-style explanations.

- **Supporting Features**

To enhance accessibility, TechTutor incorporates the **Text-to-Speech (TTS)** for narration and **Speech-to-Text (STT)** for hands-free control. User adaptability is supported through features such as **repositioning and scaling AR objects, saving quiz history, and storing narration audio files** for review. These features ensure inclusivity for learners with different needs and learning speeds.

- **Target Users**

The application is designed for **beginners of all ages** who wish to acquire fundamental computer skills, with particular emphasis on **supporting older adults** who often face greater barriers in digital literacy.

- **Project Boundaries**

The current implementation is limited to **Android devices with ARCore support**, and internet connectivity is required to access AI Tutor features. Also, it does not cover hardware troubleshooting or repair.

In summary, TechTutor provides an **interactive, adaptive, and accessible platform** for learning basic computer skills through **AR tutorials, quizzes, and AI-driven support**, while remaining focused on foundation skills within defined project boundaries.

1.4 Contributions

The contributions of this project can be summarized as follows:

- **Integration of AR and AI for digital literacy**

Unlike existing applications that focus on STEM subjects or professional training, TechTutor applies Augmented Reality (AR), and Artificial Intelligence (AI) specifically to the field of **basic computer and technology education**. This unique integration provides an immersive and intelligent learning environment that is currently lacking in widely available solutions.

- **Comprehensive learning workflow**

TechTutor introduces a structured workflow that combines **AR Learn, AR Guide, and AR Quiz**, which enables users to progress from recognition of computer components to guided step-by-step tutorials, and finally to knowledge assessment with quiz history review. This approach ensures that learners not only acquire knowledge but also reinforce and evaluate their understanding in an interactive manner.

- **Personalized and accessible learning support**

Through features such as **AI Tutor, Speech-to-Text (STT), Text-to-Speech (TTS), and OCR-based keyword highlighting**, the system accommodates diverse learning speeds, styles, and accessibility needs. This inclusivity makes TechTutor suitable for users of all ages. At the same time, it provides additional support for beginners and older adults who face greater challenges in digital literacy.

- **Contribution to bridging the digital divide**

By making basic computer education more approachable and less overwhelming, TechTutor addresses one of the critical barriers to digital literacy. This project not only educates, but also empowers users, fostering confidence in technology use and contributing to more inclusive digital participation in this society.

1.5 Background Information

In today's technology-driven society, digital literacy has become a fundamental, but essential skills for individuals of all ages. Digital literacy is broadly defined as the ability to access, create, evaluate, and communicate information by using the digital technologies responsibly and effectively [12]. To support digital literacy, advanced technologies such as **Augmented Reality (AR)** and **Artificial Intelligence (AI)** have gained significant attention in the education sector.

Augmented Reality (AR) is a technology that overlays digital information, such as images, sounds, and 3D models onto the real-world environment through devices like smartphones and tablets. Unlike Virtual Reality (VR), which immerses users in a fully virtual space, AR enhances the real-world experiences by blending digital content with

the physical surroundings [10]. Therefore, it creates interactive and engaging learning opportunities for users. Recent studies also have shown that AR can significantly improve motivation, experiential learning, and retention in education [9].

Artificial Intelligence (AI) also plays an important role in modern learning systems, especially in this digital-growing era. For example, AI-driven chatbots simulate human-like conversations by providing a real-time support by understanding user queries and generating relevant responses [11]. These chatbots are increasingly applied in education area to assist learners by answering questions, offering guidance, and providing on-demand explanations. When combined with natural language processing and other recognition technologies, AI facilitates personalized and accessible learning pathways effectively and efficiently.

In summary, the integration of AR and AI into educational applications represents a convergence of innovative technologies that aimed at enhancing the digital literacy and user engagement. By combining immersive AR tutorials with AI-driven guidance, these technologies can bridge the learning gaps, simplify access to digital tools, and create more inclusive educational environments for users from diverse backgrounds.

1.6 Report Organisation

This report is organized into seven chapters. **Chapter 1** introduces the project by presenting the problem statement and motivation, project scope, objectives, contributions, background information, and the overall report organisation. **Chapter 2** provides a detailed literature review, including an overview of related technologies, a review of previous applications, and a comparison of existing works that form the foundation of this project. **Chapter 3** outlines the research methodology, project requirements, and project timeline. **Chapter 4** describes the system design through system flowchart, diagrams, and architecture. **Chapter 5** presents the system implementation, covering software and hardware setup, configuration, operation of modules, and challenges faced. **Chapter 6** discusses the system evaluation, testing approach, results, project challenges, and the evaluation of objectives. Finally, **Chapter 7** concludes the report and provides recommendations for future improvements and extensions.

Chapter 2

Literature Review

This chapter summarizes the technologies and prior works relevant to the development of TechTutor. It first reviews key technologies such as Augmented Reality (AR), AR development platforms, AR manuals, and Optical Character Recognition (OCR). Then, it examines several existing AR-based educational applications, including Quiver, Visible Body, ARChem, Jigspace, and Assemblr EDU, alongside OCR tools such as EasyOCR and Tesseract. Finally, the chapter compares the strengths and limitations of these works, highlighting gaps that inform the design of the proposed TechTutor application.

2.1 Review of the Technologies

2.1.1 Augmented Reality (AR)

According to [16], the concept of Augmented Reality (AR) was first introduced in 1992 by Thomas Caudell at Boeing, who created an AR application to assist with industrial assembly tasks. It is a technology that enhances the real-world environment by overlaying digital information such as images, sounds, and 3D models in real time [9], [10]. Over time, AR has been defined in various ways, but it generally revolves around a ‘**Reality-Virtuality Continuum**’, which outlines a scope of environments ranging from real world to the fully virtual environment [17]. Figure 2.1 illustrates the Reality-Virtuality Continuum [17], which includes:

- **Real Environment (RE):** The physical world where people live.
- **Augmented Reality (AR):** The real world enhanced with digital virtual elements to further enhance the reality.
- **Augmented Virtuality (AV):** A predominantly virtual environment enriched with real-world elements.
- **Virtual Reality (VR):** A completed simulated digital environment with no real-world view.

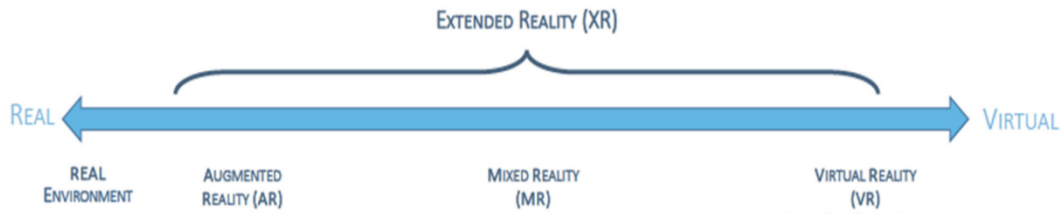


Figure 2.1 Reality-Virtual Continuum Schematic [17]

AR works by linking digital information to the physical world, enhancing the user's perception and interaction with their surroundings. Key AR technologies include

- **Marker-based AR:** It uses visual markers (such as QR Codes) to trigger the virtual content.
- **Markerless AR:** It relies on GPS, accelerometers, or other environmental sensors to display virtual elements without specific markers.
- **Project-based AR:** It projects light onto the real-world surfaces to detect interactions.
- **Overlay AR:** It replaces real-world objects with digital information for enhanced visualization.

Until today, AR is applied across diverse fields, from industrial maintenance to immersive entertainment and education [15], [18]. With the rise of Industry 4.0 and mobile technologies, AR continues to evolve as a tool for both productivity and learning enhancement.

2.1.2 Augmented Reality (AR) Development Platforms

In recent years, the development of AR applications has been significantly advanced by specialized platform such as Apple's ARKit and Google's ARCore, launched in 2017 and 2018 respectively [19], [20]. These platforms provide powerful tools for motion tracking, plane detection, and environmental understanding, making AR development more accessible to a wide range of developers [21].

According to Apple's official documentation, ARKit enables the precise mapping of 3D spaces with advanced capabilities such as 3D object detection, scene reconstruction,

and people occlusion [14]. ARKit is exclusive to iOS devices, which is known for its high frame rate performance and advanced features like 3D object detection and scene reconstruction. It enables the precise mapping of 3D spaces, making it a preferred platform for high-demand AR applications. In contrast, ARCore offers cross-platform compatibility, supporting both Android and iOS devices [21]. Although ARCore's optimization on iOS devices may not match that of ARKit, its broad device support makes it an attractive choice for developers who are targeting for a diverse audience.

To further address the cross-platform challenges, Unity introduced ARFoundation, a development framework that allows developers to create AR applications deployable on both ARKit and ARCore platforms using a unified database [22]. AR Foundation provides key functionalities like plane tracking, light estimation, and environment probes, simplifying the development of immersive AR experiences [22].

2.1.3 Augmented Reality (AR) Manuals

An emerging application of AR is in the development of AR Manuals, which replace traditional paper-based manuals with interactive digital overlays. The supported picture is shown illustrated in Figure 2.2. According to [23], AR manuals allow user to point their mobile device cameras at a piece of equipment, then it will instantly display the relevant instructions, maintenance guides, or 3D models that superimposed onto the real world. This process is enabled through the object recognition and real-time guidance.

AR manuals significantly enhance the visual clarity, reduce the error rates, and improve the user productivity [15]. They offer numerous advantages, including hands-free operation, faster troubleshooting, and eco-friendly documentation by eliminating the reliance on printed materials. According to [16], highlighted that AR manuals in industrial settings have proven to be highly effective tools in reducing downtime and streamlining complex maintenance processes. Beyond industrial use, the use of AR manuals is expanding into educational and training environments, demonstrating AR's versatility and long-term potential for improving learning outcomes [16].



Figure 2.2 Example of Usage of AR Manuals [24]

2.1.4 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is a fundamental technology that enables the automatic conversion of printed, handwritten, or scene text into machine-readable form [43]. Traditionally, OCR system relied on handcraft features and segmentation-based methods, but these approaches often struggled with complex backgrounds, irregular text arrangements or varied fonts. According to the survey by Singh et al., the author highlights that OCR has revolutionized industries such as banking, healthcare, and digital libraries by automating document digitization, reducing the manual data entry and supporting large-scale information retrieval [43]. With the integration of advanced segmentation algorithms and optimization techniques, OCR has expanded its applications to domains such as CAPTCHA solving, music score recognition, and also the vehicle plane detection, demonstrating its versatility in real-world contexts [43]. Figure 2.3 shows an example of how OCR techniques can be applied to segment and recognize the distorted text in CAPTHCA images [43].

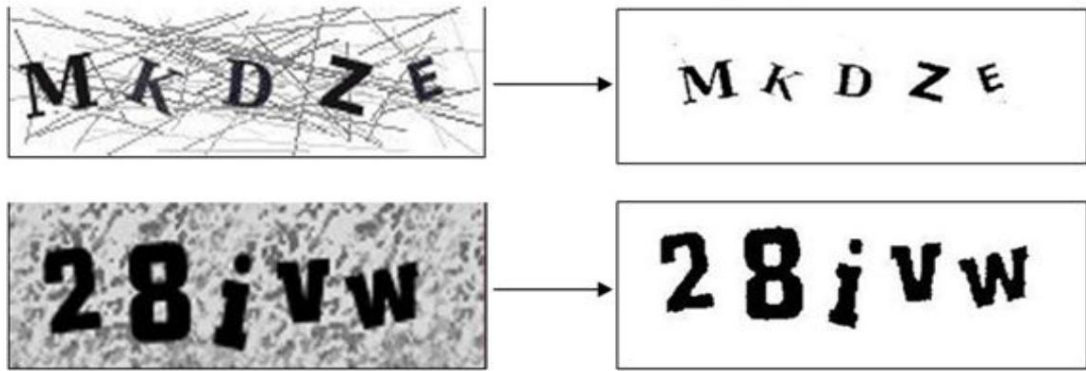


Figure 2.3 Example of CAPTCHA usage in OCR [43]

Recently, advances in deep learning have redefined the OCR pipeline, which typically consists of 2 major stages, text detection and text recognition. In the detection stage, models identify regions containing text, which may appear in arbitrary orientations or curved forms [44]. The Character Region Awareness for Text Detection (CRAFT) model proposed by Baek et al. introduced a character-level awareness mechanism that localizes each character and estimates the affinity between adjacent characters, enabling robust detection of texts with irregular shapes [44]. Unlike the earlier word-level detection methods, CRAFT's bottom-up strategy improves the accuracy effectively in complex scenarios such as curved or deformed text, achieving state-of-the-art performance on multiple public benchmarks. As shown in Figure 2.4, CRAFT can accurately capture text regions across different orientations, demonstrating the higher flexibility compared to conventional regression-based methods [44].



Figure 2.4 Example of CRAFT Detecting Texts in Horizontal, Curved, and Arbitrary Shapes [44]

For the recognition stage, deep learning approaches such as the Convolutional Recurrent Network (CRNN) proposed by Shi et al. have proven highly effective. CRNN integrates the convolutional layers for feature extraction, recurrent layers for sequence modeling, and a transcription layer using Connectionist Temporal Classification (CTC) to output variable-length text sequences [45]. This end-to-end trainable framework eliminates the need for character-level segmentation and demonstrates superior performance on scene text recognition benchmarks [45]. Its compact model size and robustness make it practical for real-world applications, including mobile and AR-based systems. Figure 2.5 illustrates the CRNN architecture which enables it to transcribe variable-length text sequences directly from images without requiring the explicit character-level segmentation.

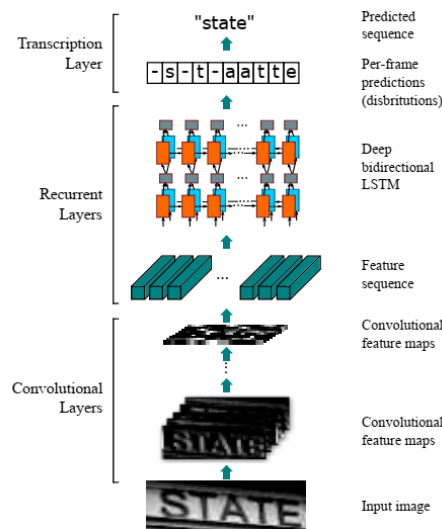


Figure 2.5 Architecture of Convolutional Recurrent Neural Network (CRNN) for Text Recognition [45]

2.2 Review of Previous Works

Since there is currently **no existing application** that integrates Augmented Reality (AR) and Artificial Intelligence (AI) for teaching basic computer and technology operations, this literature review focuses on analyzing related AR-based educational applications. By reviewing existing applications such as Quiver, Visible Body, ARChem, JigSpace and Assemblr EDU, their advantages, limitations, and technological approaches are identified. Through this analysis, insights are abstracted and used to improve in design of proposed TechTutor application, ensuring that it leverages the strengths and addresses the gaps observed in similar educational solutions.

2.2.1 Quiver – 3D Colouring App

Quiver – 3D Coloring App is an Augmented Reality (AR) application that enhances the traditional colouring activities by combining them with interactive 3D rendering technologies. The app transforms 2D colouring pages into animated 3D models that can be viewed, manipulated, and interacted with in real-time which target primarily at younger audiences. Through this innovative approach, Quiver offers an engaging way to stimulate creativity and learning among children [25].



Figure 2.6 Logo of Quiver – 3D Colouring App [25]

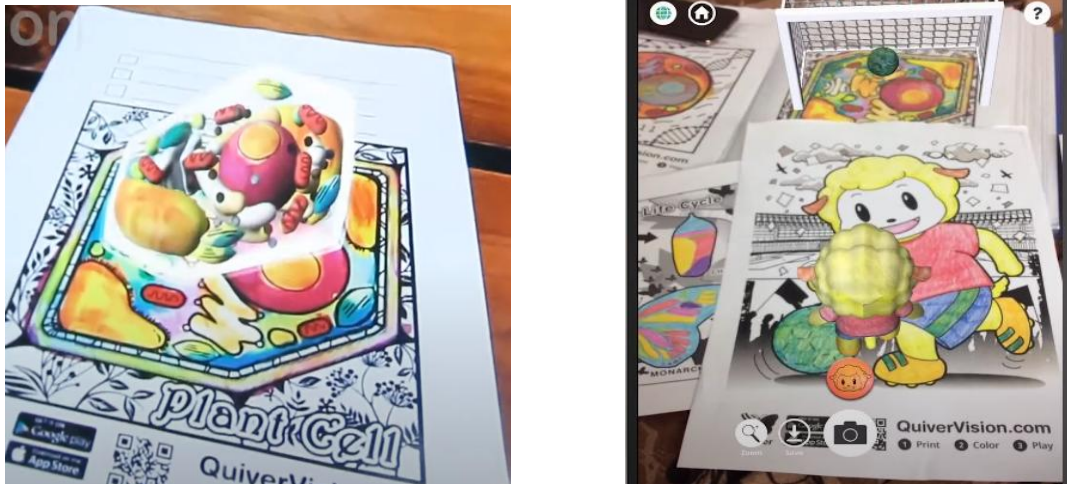


Figure 2.7 Examples of Marker-based Coloring Papers [26, 27]

Advantages

Quiver utilizes a marker-based approach, where specially designed colouring pages serve as AR markers. The applications apply computer vision techniques to detect the outlines and coloured regions on the printed page, extracts the colour information, and then maps it onto a corresponding 3D model. The dynamic colour mapping process creates a personalized 3D AR experience, where the model appears exactly as coloured by the user. This method aligns with standard practices in AR colouring apps that use computer vision for marker detection and real-time texture mapping [28]. The example about this technique can refer to the Figure 2.1.

In addition, its' primary advantage will be the simplicity and accessibility of the application itself. It is easy for children or beginners to use without any prior technical knowledge. The app delivers smooth AR rendering and support intuitive interaction through basic touch gestures such as rotating, scaling and moving 3D models. By blending familiar manual activities like colouring with digital interactivity, Quiver

effectively lowers the barriers to AR adoption and promotes the visual and experiential learning.

Limitations

Despite its strengths, its interactivity is relatively basic, it confined mainly to touch manipulation of static 3D models. The system heavily depends on the pre-printed colouring sheets provided by Quiver, limiting the flexibility and creative freedom. It also lacks advanced educational features such as voice-guided interaction to enhance the auditory effect for children.

Suggested Improvements

To further expand its educational potential, Quiver could integrate the voice recognition or sound effects, such as triggering animal sounds for specific models, to enhance the immersion and engagement. Also, it may introduce the markerless AR technologies, so that it allows users to animate any artwork without the need for predefined sheets, offering greater flexibility of the application. Furthermore, incorporating the interactive storytelling, gesture-based controls, and user progress tracking could transform Quiver from a simple AR colouring tool into a more comprehensive learning platform.

2.2.2 Visible Body

Visible Body is an educational application that leverages AR and 3D visualization to provide an immersive, interactive experience for studying human anatomy. It is designed primarily for medical education and offers detailed customizable 3D models of the human body, allowing users especially for medical students to explore the anatomical systems more effectively than the traditional textbooks or online resources. It supports both iOS and Android devices, ensuring accessibility across a wide range of users.

It uses the markerless AR, enabling users to project and manipulate 3D models directly onto flat surfaces detected by the device's camera and sensors. Plane detection ensures that the models remain accurately positioned within the physical environment. Furthermore, object recognition and motion tracking technologies are utilized to

maintain the model's spatial stability even as users move around, enhancing the real-time interaction and immersion [31].



Figure 2.8 Logo of Visible Body [31]

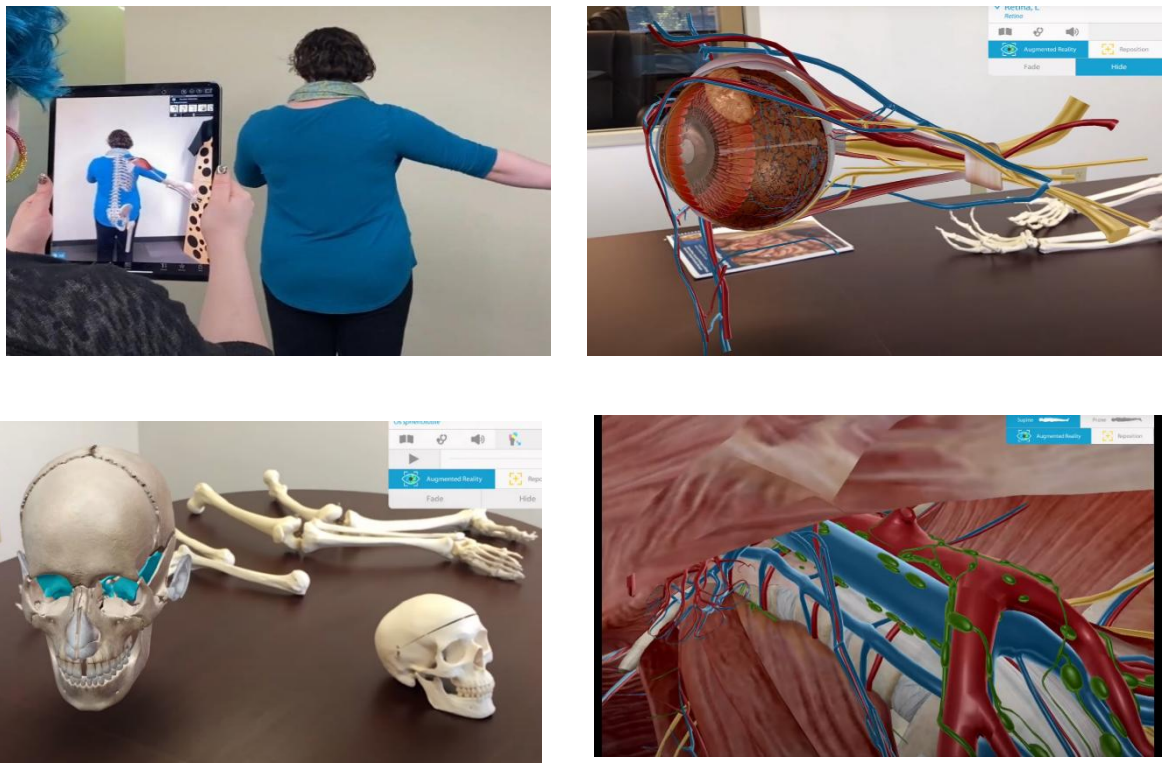


Figure 2.9 Example of Usage of AR Technology in Visible Body [29, 30]

Advantages

It delivers high-resolution, anatomically accurate 3d models that users can intuitively manipulate through zooming, rotating, and isolating specific anatomical structures. This interactive approach fosters deeper visual understanding and supports experiential learning, which is important for students who specialize in medical and healthcare fields. Through the AR technology used in the application, users are able to gain practical, hands-on experience in exploring human anatomy in realistic spatial contexts, enhancing their preparation for real-world clinical environments.

Furthermore, the use of markerless AR eliminates the need for external markers or printed materials, simplifying the setup process and improving user convenience. By removing the reliance of physical media, the application not only reduces the preparation time but also contributes positively to the environmental sustainability by minimizing the paper waste.

Limitations

Despite its strengths, Visible Body presents a learning curve for users who are unfamiliar with AR environments. It may potentially hinder the initial engagement. First-time users may require additional time to adapt to interacting with the virtual anatomical models in an Augmented space. Additionally, the app's coursework and educational materials is currently available in English, which limit the accessibility for non-English speaker users. Furthermore, the app is mainly specialized in human anatomy and does not support broader educational customization such as creation of user-generated tutorials, adaptive learning pathways, or content modification beyond the provided 3D models.

Suggested Improvements

To further enhance its usability and accessibility, Visible Body could consider expanding its educational content into multiple languages, improving the accessibility for global audience. Also, it may provide optional onboarding tutorials for first-time AR users, which may help reduce the initial learning curve and improve user confidence. In addition, it should allow for basic customization features, such as adding personalized notes or labels to the anatomical models, could increase the interactivity and educational flexibility for the educational contexts.

2.2.3 ARChem

ARChem is an educational Augmented Reality (AR) application developed to provide an interactive, virtual chemistry laboratory experience. It allows users to engage in hands-on lab activities without the need of physical laboratory equipment, making it an accessible tool for educational institutions with limited resources. This was particularly beneficial during the Movement Control Order period (MCO) in Malaysia in 2021, when the COVID-19 pandemic restricted the physical classroom access.

The application primarily relies on marker-based AR technology, utilizing the Vuforia Engine for object tracking and interaction [33]. Through the camera, physical markers are detected, and a virtual laboratory apparatus and chemical simulations are overlaid onto the real-world environment. To enhance realism, ARChem integrates the OBI Fluid physics engine to simulate fluid dynamics, allowing the visualization of the liquid behaviour during the chemical experiments [34].

In addition, ARChem incorporates several supporting modules, including an AR Engine, a Reaction Engine, a Reaction Database, and advanced Fluid Renderers, which work together to deliver smooth interactions, realistic experiment simulations, and computational efficiency on mobile devices [34].

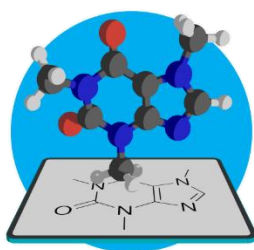


Figure 2.10 Logo of ARChem [32]

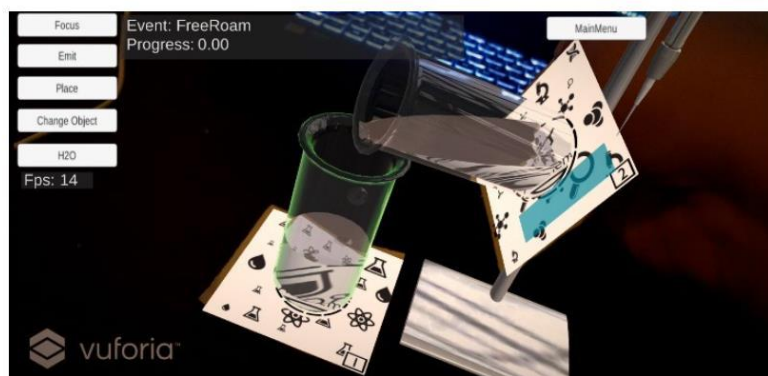


Figure 2.11 An Example Screenshot of Demonstration using Two Marker Papers [34]

Advantages

One of the ARChem's major advantages is its ability to deliver highly realistic and interactive virtual chemistry experiments, promoting experiential learning without the risks associated with handling any dangerous chemicals. The application supports high frame rate performance and optimizes visualization, ensuring smooth operation even on low-end mobile devices. Therefore, it broadens its accessibility to students from

various socioeconomic backgrounds. Additionally, by simulating the real-time fluid behaviour during the experiments, ARChem offers an authentic and engaging representation of chemical reactions, enhancing the student's conceptual understanding and interest in the subject matter.

Limitations

Despite its strengths, ARChem exhibits limitations primarily related to its reliance on marker-based AR. If the physical marker is moved, poorly lit or partially obscured during use, alignment issues may occur, causing the virtual objects to misalign or drift from their intended position. Such disruptions can affect the continuity of the learning experience and reduce overall immersion for students. Furthermore, the system currently focuses solely on chemistry-related experiments and does not offer customizable experiment creation or personalized feedback pathways, which could broaden its application to wider educational fields.

Suggested Improvements

To improve usability, ARChem can consider integrating markerless AR technologies to reduce its dependency on physical markers, thereby enhancing system stability and ease of use across different environments. Besides, ARChem may consider adding basic adaptive features, such as adjustable difficulty levels for experiments or real-time feedback based on user interactions to further personalize the learning experience. Finally, it can expand its subject coverage to include broader science education topics such as physics or biology experiments, so that it could significantly increase its versatility and impact within science education.

2.2.4 Jigspace

JigSpace is an Augmented Reality (AR) platform that enables users to explore interactive 3D visualizations, known as 'Jigs', across both iOS and Android devices. It is developed using Unity 3D and utilizing Unity's AR Foundation framework to ensure cross-platform compatibility. AR Foundation acts as a bridge between ARKit (for iOS) and ARCore (for Android), automatically adapting the application at runtime based on the device's operation system [35].



Figure 2.12 Logo of JigSpace [35]

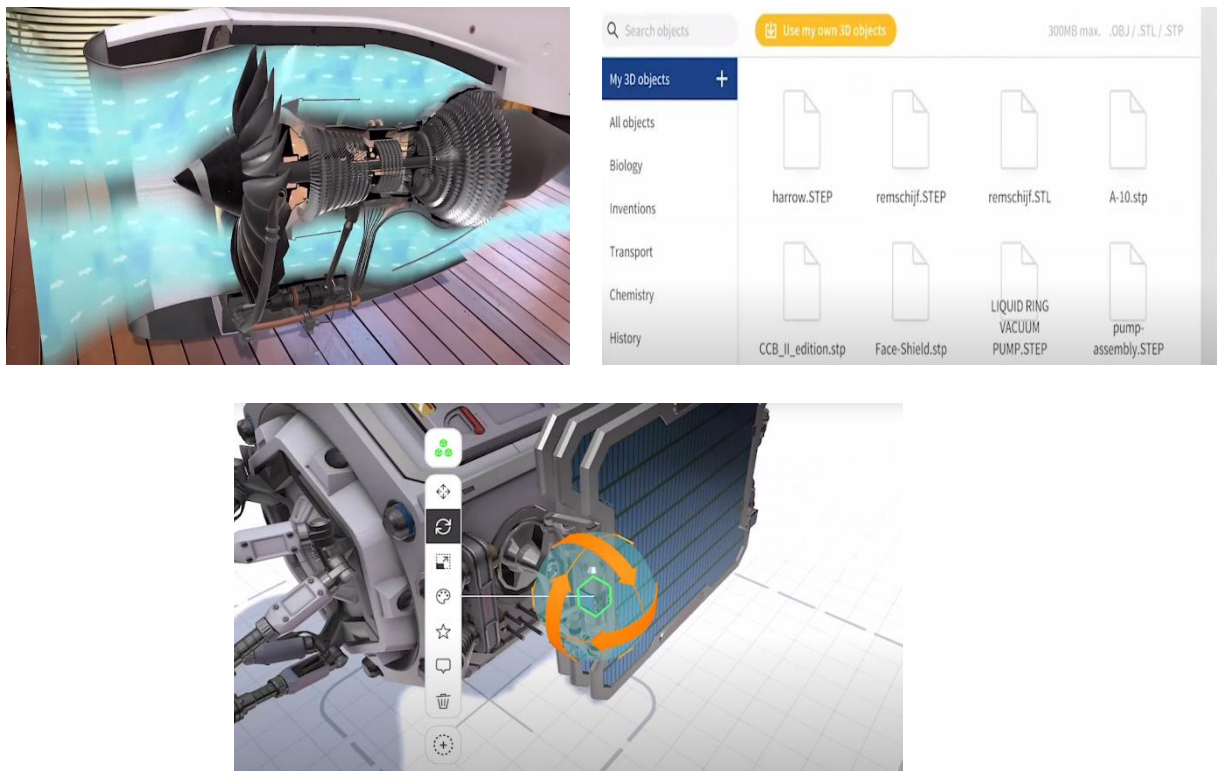


Figure 2.13 Example of Creation of 'Jig' [36]

Advantages

One of the key advantages of JigSpace is its cross-platform support, offering AR experiences for both iOS and Android users without require separate applications. At the same time, the platform provides high-resolution 3D models that user can zoom, relate, and interact intuitively with it. It helps to promote better visual learning and understanding of complex structure. Additionally, JigSpace hosts its 3D models in the cloud through the Jig Library [35], allowing users to easily access a broad range of pre-built Jigs without consuming the extensive local storage. These features make it

particularly effective for industries that require clear visual communication, such as manufacturing, medical devices and engineering [35].

Limitations

Despite its strengths, JigSpace is primarily designed for business use cases, including the industrial training, product demonstrations and engineering visualizations. As a result, it may not be well-suited for beginners, older adults, or general educational use. Creating a new ‘Jig’ from scratch requires lots of technical effort, making it difficult for non-technical users to develop customized content. Furthermore, JigSpace focuses heavily on pre-built topics, offering limited support for creating custom tutorials or personalized, adaptive learning paths. The platform also lacks integrated user progress monitoring, real-time adaptive feedback, making it less suitable for structured educational programs.

Suggested Improvements

JigSpace could be improved by integrating sound elements, such as voice narrations and interactive audio cues, to enhance the user engagement during AR interactions. Besides, expanding support for creating customized, step-by-step tutorials will further increase its adaptability for educational environments.

2.2.5 Assemblr EDU

Assemblr EDU is an educational augmented reality (AR) application that enables users, particularly educators, to create customized AR lessons for their students. It allows the creation of both QR-based markers and customized image markers, onto which 3D models, animations, and other digital content can be anchored. It leverages the AR rendering techniques built using Unity 3D, integrating both ARKit and ARCore frameworks to deliver cross-platform augmented reality educational experiences [37], [38].



Figure 2.14 Logo of Assemblr EDU [40]

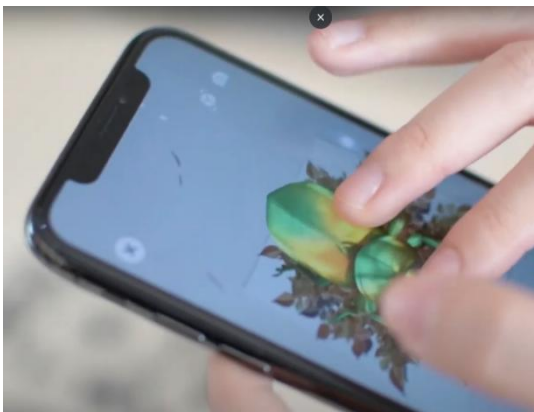


Figure 2.15 Customizable Image Markers [39]

Advantages

One of the key advantages of Assemblr EDU is its accessibility for non-technical users. It offers intuitive tools that allow educators and students to easily create their own AR lessons by attaching 3D models to markers or real-world surfaces. The platform is specifically designed to support the learning and teaching process, promoting collaborative and interactive experiences in educational settings [13]. Through visual and hands-on engagement, it enhances students' understanding, particularly for younger learners. Besides, it also offers a flexible licensing model, providing a free package with access to basic 3D objects and subscription packages that unlock a broader range of educational content.

Limitations

Despite its strengths, Assemblr EDU presents several limitations. While it allows users to create their own AR content, the range of available free 3D objects is limited, with many educational resources are restricted to pay the subscription plans. Furthermore,

although users can create interactive scenes, the platform focuses primarily on static 3D model placement without offering real-time adaptive learning feedback, AI-guided assistance, or dynamic learning adjustments. The platform also lacks comprehensive student progress tracking, which reduces its effectiveness for structured educational programs that require continuous monitoring of learning outcomes.

Suggested Improvements

Assemblr EDU could be further enhanced by integrating AI-driven feedback mechanisms that guide learners based on their interactions with AR models. By adding the basic progress tracking features and hint-systems for self-correction would make it more powerful tool for a personalized education. Furthermore, expanding the range of freely accessible 3D educational models would lower the adoption barriers, especially for schools and institutions with limited budgets, making the platform more widely usable across diverse educational environments.

2.2.6 EasyOCR vs Tesseract OCR

Optical Character Recognition (OCR) methods such as EasyOCR and Tesseract have been widely used in various applications, and their comparative performance has been evaluated in real-world contexts. Vedhaviyassash et al. conducted a study on automatic license plate recognition, integrating the YOLOv5 for object detection and applying both EasyOCR and Tesseract OCR for character recognition [46]. The results demonstrated that EasyOCR consistently outperformed Tesseract, achieving an accuracy of approximately 95%, whereas Tesseract reached around 88 – 90% [46]. This performance gap was largely attributed to EasyOCR's deep learning pipeline, which employs CRAFT for text detection and CRNN for text recognition, making it more robust under challenging conditions such as varying fonts and noisy backgrounds [44, 45].

Beyond accuracy, an important feature of EasyOCR is its ability to return both the recognized text and the positional coordinates of each character or word. This functionality is particularly useful in applications where position-aware outputs are required, such as the project proposed, TechTutor. In contrast, Tesseract OCR, while effective for printed and structured text, was reported to support fewer languages and

relies on its adaptive classifier for recognition accuracy [46]. Therefore, EasyOCR is selected for this project as it better satisfies the requirements of real-time, position-aware text recognition in AR-based learning. As shown in Figure 2.16, EasyOCR returns not only the recognized text but also additional information such as confidence scores (and in its full output, positional coordinates). In contrast, Tesseract OCR outputs only the recognized text, which is sufficient for document digitization but less advantageous in AR-based contexts [46].

Parameter	Test Data 1	Test Data 2
Original Image		
Character Recognition using EasyOCR	'HR.26 BR.9044', 0.5167916087717548)]	'POLBCAF5030', 0.30806908725519727)]
Character Recognition using Tesseract		

Figure 2.16 Comparison of Character Recognition Results Between EasyOCR and Tesseract OCR [46]

Table 2.1 shows the comparison table to further highlight the strengths and limitations of EasyOCR and Tesseract. Also, it provides a clearer understanding of why EasyOCR is more suitable for the proposed project, TechTutor, particularly in terms of its ability to provide position-aware text recognition and support AR-based applications.

Table 2.1 Comparison between EasyOCR and Tesseract OCR


Features	EasyOCR	Tesseract OCR
Accuracy	Around 95% on license plate recognition [46].	Around 88 to 90% on license plate recognition [46].
Core Approach	Deep learning pipeline (CRAFT for detection +	Classical OCR engine with adaptive classifier [46].

	CRNN for recognition) [44, 45]	
Language Support	70+ languages, including complex scripts [46].	Fewer languages supported [46].
Dependencies	Lightweight, PyTorch-based API with minimal setup [46].	Requires additional setup and dependencies [46].
Text Position Output	Return recognized text with positional coordinates [46]	Outputs primarily plain text
Best Use Case	Real-time, multilingual, position-aware OCR	Document digitization and structured printed text.

2.3 Comparison of Previous Works

Table 2.2 shows the comparisons for several existing AR-based educational applications, highlighting their technologies, advantages, limitations and potential improvements. Based on this analysis, the proposed TechTutor application is designed by combining the best features across these platforms while addressing their gaps. TechTutor uniquely integrates AR-guided tutorials with AI-based feedback and user progress tracking, targeting an underserved need for accessible, intuitive digital literacy education, especially for beginners and older adults.

Table 2.2 Comparison Table of Existing Applications vs TechTutor (Proposed Project)

Application	Technology Used	Advantages	Limitations	Potential Improvements
Quiver 3D Coloring App 	- Marker-based AR - Computer Vision for Color Mapping	- Personalized 3D models based on user colouring - Simple and kid-friendly	- Limited interactivity - Rely on pre-printed colouring papers.	- Add voice/sound interactivity - Move to markerless AR - Expand educational storytelling
Visible Body	- Markerless AR	- High-resolution anatomy models	- Initial learning curve	- Add more tutorials

	<ul style="list-style-type: none"> - Plane Detection - Object Tracking 	<ul style="list-style-type: none"> - Multi-angle exploration 	<ul style="list-style-type: none"> is high for AR beginners - English-only coursework - Limited content customization 	<ul style="list-style-type: none"> - Multi-language expansion - Allows user annotations and labels
ARChem 	<ul style="list-style-type: none"> - Marker-based AR (Vuforia) - OBI Fluid physics engine 	<ul style="list-style-type: none"> - Realistic virtual chemistry experiments - Optimized for low-end devices 	<ul style="list-style-type: none"> - Marker misalignment issues - Limited to chemistry topics 	<ul style="list-style-type: none"> - Introduce markerless AR - Broaden subject range - Add adaptive feedback systems
JigSpace 	<ul style="list-style-type: none"> - Markerless AR - Cloud-based Jig Library - High-fidelity 3D rendering 	<ul style="list-style-type: none"> - Highly interactive object manipulation - Good for visual learning 	<ul style="list-style-type: none"> - Business-focused - Hard to create custom Jigs - No user progress monitoring 	<ul style="list-style-type: none"> - Provide educational templates - Add easier Jig creation tools - Integrate progress tracking system
Assemblr Edu 	<ul style="list-style-type: none"> - Marker-based AR & Markerless AR - Unity with ARCore/AR Kit integration 	<ul style="list-style-type: none"> - Easy AR content creation for education - Collaboration classroom use 	<ul style="list-style-type: none"> - Limited free 3D assets - No adaptive feedback or learning analytics 	<ul style="list-style-type: none"> - Add AI-guided hints - Broaden free content - Introduce progress analytics features
TechTutor (Proposed Project) 	<ul style="list-style-type: none"> - Markerless AR (Vuforia + AR Foundation) - Marker-based AR - Unity 3D 2022.3.57f1 	<ul style="list-style-type: none"> - Combined AR Learning modules (AR Learn, AR Guide, AI Tutor) - Multi-modal interaction 	<ul style="list-style-type: none"> - Requires stable Internet connection for OpenAI API (except offline stored content) 	<ul style="list-style-type: none"> - Add more computer-related tutorials such as Excel, PowerPoint - Enhance offline functionality with

	<ul style="list-style-type: none"> - OpenAI API integration - EasyOCR for text detection & recognition - Speech-to-Text (STT) and Text-to-Speech (TTS) - Local storage for history and quiz records 	<ul style="list-style-type: none"> (voice, touch, image input) - Interactive 3D models with labelling and manipulation. - Real-time keyword highlighting via OCR (Canvas Mode) - AI-generated quizzes and chatbot guidance - Synchronized AR + Canvas modes for tutorials - Designed for beginners and older adults to improve digital literacy. 	<ul style="list-style-type: none"> - Limited range of tutorials compared to full software suits. - Dependent on mobile device performance. 	<ul style="list-style-type: none"> pre-downloaded AI responses - Improve AR stability and tracking on low-end devices. - Expand adaptive feedback and personalized learning analytics. - Integrate multi-language support for wider accessibility.
--	---	--	--	--

Chapter 3

System Methodology

This chapter presents the overall methodology adopted for the development of TechTutor application. It outlines the software development approach used, the system requirements in terms of hardware and software, and the functional and non-functional requirements that guided the design of the system. In addition, the project timeline is provided to illustrate the structured progression of tasks throughout both Final Year Project I (FYP I) and Final Year Project II (FYP2).

3.1 Methodology

The methodology chosen for this project is the **Agile Software Development Methodology**. Agile development is suitable for this project because it emphasizes the **iterative development, continuous feedback** and **flexible adaptation**, which make it ideal for the rapid development of the TechTutor application, which consists of multiple AR and AI-integrated features.

Since TechTutor is not a safety-critical system, but a mobile learning application, Agile methodology offers the flexibility needed to adapt to changes and continuously refine the user experience throughout the development process. The system was divided into smaller modules, including **AR Guide, AR Learn, and AI Tutor**. Each module was developed, tested and improved independently through the short iterative cycles (sprints). In addition, this approach supports early testing of AR functionality, AI integration and user interaction features, which allows the issues to be identified and resolved at each stage of the sprints through the continuous feedback. The process flow diagram is shown in Figure 3.1.

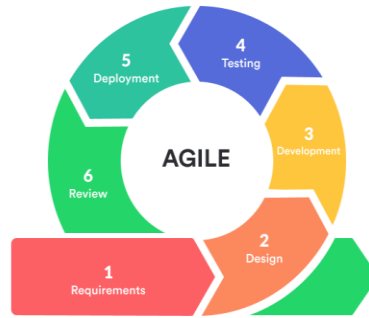


Figure 3.1 Agile Software Development Approach [41]

Planning Phase

In the planning phase, user requirements (functional and non-functional requirements) were identified. The project objectives, problem statements, and project scope were discussed and finalized. Wireframe of user interfaces and system flow diagrams were prepared. The 3 main modules were prioritized according to their complexity and importance:

- **AR Learn Module** (AI Identify, AR Models, AR Quiz)
- **AR Guide Module** (Tutorials for Microsoft Word, Snipping Tool, Calculator, and Send Email)
- **AI Tutor Module** (AI Assistance, Frequently Asked Questions (FAQ))

Design Phase

System architecture diagrams, use case diagrams, and activity diagrams were created to model the system structure and user interaction flows. The modular breakdown of the system was finalized, which involves:

- **AR Learn Module** – Support image target detection, plane detection, and marker-based AR for quizzes.
- **AR Guide Module** – Provides step-by-step tutorials using markerless AR overlays and marker-based AR overlays with synchronized Text-to-Speech narration.
- **AI Tutor Module** – Provides text-based and image-based assistance, powered by OpenAI integration and EasyOCR keyword highlighting.

Development Phase

The development followed an iterative, sprint-based approach, where each sprint focuses on building specific features. Table 3.1 illustrates the detailed for each of the sprint.

Table 3.1 Sprint Cycles for Each Sprint

Sprint	Focus Area
Sprint 1	Setup Unity project, integrate Vuforia and Google ARCore XR Plugin.
Sprint 2	Build AR Guide tutorials (Microsoft Word and Snipping Tool).
Sprint 3	Develop AI Tutor chatbot with OpenAI integration and EasyOCR support.
Sprint 4	Build AR Learn features (AI Identify and AR Models).
Sprint 5	Implement AR Quiz with marker-based triggers and question generation.
Sprint 6	Extend AR Guide with Send Email and Calculator tutorials.
Sprint 7	Add FAQ modules and refine Text-to-Speech (TTS) and Speech-to-Text (STT) functions.
Sprint 8	Final integration, UI refinement, optimization and testing.

Testing Phase

Testing will be conducted iteratively at the end of each sprint to ensure the quality of the feature and overall system stability.

- **Unit Testing:** Each module was tested independently, such as AR detection accuracy, quiz functionality, and AI chatbot responses.
- **Integration Testing:** Verified smooth interactions between AR modules, AI Tutor, and multimedia features (STT/TTS).
- **User Testing:** Evaluated usability, accessibility, and clarity of tutorials for beginners and older adults.

- **Final System Testing:** End-to-end testing was conducted after full integration to ensure stability across all modules.

Deployment Phase

After thorough testing, the completed application was deployed to Android devices. Final APK builds were generated and tested on real devices to ensure compatibility with ARCore-supported phones. Minor adjustments were made based on device performance results.

Review Phase

Feedback was collected regarding the user experience, system usability, stability, and also the functionality. Based on the feedback, further refinements and improvements were implemented. Final project documentation, reports, and deliverables were then completed.

3.2 System Requirements

3.2.1 Hardware

The hardware requirements for this project include a development machine and a mobile device for deployment and testing. The specifications ensure that the system can run Unity smoothly, support AR rendering, and validate Speech-to-Text (STT) and Text-to-Speech (TTS) features. Table 3.2 shows the minimum recommended hardware specifications.

Table 3.2 Hardware Specifications




Hardware	Minimum Requirement	Description
Development Machine (Laptop / PC)	Intel Core i5 processor or equivalent (recommended i7 or above), 8GB RAM (recommended 16GB), Dedicated GPU with OpenGL 4.5+ support, 500GB storage (SSD recommended), Windows 10 or above.	Used for Unity development, system configuration, and integration of AR, AI, and backend services.






Mobile Device (Android)	Google ARCore-supported device, Octa-core CPU, 4GB RAM (recommended 6GB-8GB), Android 10 or above, Rear camera with autofocus.	Used to deploy the APK, test AR functionality, and validate STT and TTS performance.
--------------------------------	--	--

3.2.2 Software

The development of this project involves several software applications. Table 3.3 shows the related software that will be used to develop the system.

Table 3.3 Software Specifications

Software	Description
1. Unity 	<ul style="list-style-type: none"> It is the main platform that used to develop the application. It is a powerful cross-platform game engine that enables the creation of 3D AR environments, user interactions and scene management that is essential for the AR-based learning experiences.
2. Vuforia Engine 	<ul style="list-style-type: none"> It is an Augmented Reality (AR) software development kit that will be used to integrate with Unity. Used to detect the real-world images and objects, enabling the application to anchor the digital 3D models and tutorial panels onto recognized surfaces.
3. ARCore XR Plugin 	<ul style="list-style-type: none"> Used to allow the application to access ARCore features on supported Android mobile devices. It provides the enhanced capabilities such as light estimation, surface detection and motion tracking to provide a more stable and realistic AR experiences.

<p>4. Sketchfab</p> 	<ul style="list-style-type: none"> • Used for discovering and downloading the 3D models.
<p>5. Figma</p> 	<ul style="list-style-type: none"> • Used to design the wireframes, UI layouts, buttons, icons for the application.
<p>6. GitHub</p> 	<ul style="list-style-type: none"> • Used for version control and project management throughout the development process.
<p>7. Android Studio</p> 	<ul style="list-style-type: none"> • Used for debugging while coding. For example, when deploying to Android phone, the console.log results will appear in the Android Studio log.
<p>8. Meshy.ai</p> 	<ul style="list-style-type: none"> • Used to convert the image (AR Robot) into a 3D model and display it in the AR Quiz.

3.3 Functional Requirements

The functional requirements define what the TechTutor application must be able to perform in order to meet the project objectives.

1. AR Learn Module (AI Identify, AR Models, AR Quiz)

AI Identify

- The system shall allow users to identify basic computer components such as keyboard, mouse, monitor, laptop, and speaker using AI-based recognition.
- The system shall provide voice narration (TTS) of the identified component's name and description.
- The system shall allow users to review previously scanned objects from a local history list.
- The system shall provide access to detailed information on scanned objects and allow replay of the narration if required.

AR Models

- The system shall generate quizzes dynamically using marker-based AR for the same five components (keyboard, mouse, monitor, laptop, speaker).
- The system shall present quiz questions in both text form and audio narration (TTS).
- The system shall allow users to answer quiz questions and receive immediate feedback.
- The system shall allow users to regenerate new quiz questions.
- The system shall maintain a local history of past quiz attempts, including questions and audio files, for later review.

AR Quiz

- The system shall generate quizzes dynamically using marker-based AR for the same five components (keyboard, mouse, monitor, laptop, speaker).
- The system shall present quiz questions in both text form and audio narration (TTS).

- The system shall allow users to answer quiz questions and receive immediate feedback.
- The system shall allow users to regenerate new quiz questions.
- The system shall maintain a local history of past quiz attempts, including questions and audio files, for later review.

2. AR Guide Module (Microsoft Word Tutorial, Snipping Tool Tutorial, Send Email Tutorial, Calculator Tutorial)

Microsoft Word Tutorial, Snipping Tool Tutorial, Send Email Tutorial

- The system shall provide step-by-step tutorials overlayed in AR with synchronized TTS narration.
- The system shall highlight required actions on the monitor, keyboard, or mouse to guide the user.
- The system shall allow users to toggle between 3D AR Mode and Canvas Mode, ensuring both are synchronized with instructions.
- The system shall support hands-free navigation using STT commands (“next”, “previous”).
- The system shall update the virtual monitor display dynamically at each tutorial step to reflect the expected outcome.

Tutorial Objectives (Specific)

- **Microsoft Word Tutorial:** The system shall teach users to create, edit, format text and save a document.
- **Snipping Tool Tutorial:** The system shall teach users to open the Snipping Tools, select capture modes, and save the screenshot.
- **Send Email Tutorial:** The system shall teach users to compose an email with recipients, subject, body text before sending.

Calculator Tutorial

- The system shall start the tutorial when the users scan a calculator in their laptop or monitor.
- The system shall provide step-by-step instructions overlayed in AR.
- The system shall highlight relevant calculator buttons to explain their functions.

- The system shall guide the user through an example calculation (e.g., addition) to demonstrate the practical use.
- The system shall allow hands-free navigation through STT commands such as “next” or “back”.

3. AI Tutor Module (AI Assistance, FAQ)

AI Assistance (Chatbot)

- The system shall allow users to ask questions through text input.
- The system shall allow users to upload an image or capture a screenshot for analysis.
- The system shall process questions using OpenAI and return concise and step-by-step answers.
- The system shall extract keyword from images using EasyOCR.
- The system shall highlight detected keywords in the “Canvas Mode” to guide the user.
- The system shall allow users to review past chat interactions in a history list, including text and image context.

FAQ

- The system shall provide a FAQ section accessible from the AI Tutor menu.
- The system shall include examples questions and tips to guide users on how to phrase their queries effectively.
- The system shall allow users to browser and read FAQs without requiring an Internet connection.

3.4 Non-Functional Requirements

1. AR Learn Module (AI Identify, AR Models, AR Quiz)

AI Identify

- The recognition process shall display results within 3 to 5 seconds after scanning.
- The voice narration shall use a natural and clear voice for accessibility.
- The history list shall persist locally even if the application is restarted.

AR Models

- The AR models shall render at a minimum of 30 FPS to ensure smooth interaction.
- Visual labels shall remain anchored to the model even when repositioned or rotated.
- Voice command recognition shall achieve at least 85% accuracy in quiet environments.

AR Quiz

- Quiz question generation shall complete within 5-7 seconds after the user clicked the “Generate Quiz” button.
- All saved quiz history and audio files shall remain accessible offline.
- The quiz interface shall be designed with large buttons and clear fonts for usability by beginners and older adults.

Microsoft Word Tutorial, Snipping Tool Tutorial, Send Email Tutorial

- The system shall render AR Overlays and monitor updates with a response time of less than 2 seconds per step.
- The AR highlights shall remain stable and accurately anchored to the target during user movement.
- The tutorials shall be designed with simple, beginner-friendly language to accommodate users with limited computer knowledge.
- The TTS narration shall be clear and paced slowly enough for older adults to follow comfortably.

- The interface shall provide large, clearly visible navigation buttons for manual control.

Calculator Tutorial

- The marker-based detection shall initialize within 3 seconds after the calculator is scanned.
- The AR highlights shall remain stable on the calculator surface during normal hand movement.
- The tutorial steps shall be limited to essential instructions to avoid overwhelming the user.

AI Assistance (Chatbot)

- The chatbot responses shall be displayed within 5 seconds after a query is submitted.
- The highlighted keywords in Canvas Mode shall remain clearly visible with distinct colour contrast.

FAQ

- The FAQ section shall be written in clear, beginner-friendly language to accommodate beginners and older adults.
- The FAQ interface shall use a simple layout with large, readable fonts.
- The FAQ content shall remain available offline once downloaded with the application.

3.5 Project Timeline

Figure 3.2 and Figure 3.3 show the Gantt chart outlining the project timeline for the development of the TechTutor application, covering both Final Year Project I (FYP I) and Final Year Project II (FYP II). The two Gantt Chart detail the major tasks, starting from the initial proposal and system design stages in FYP 1, through to the development, testing, deployment, and the final report writing in FYP 2. Each task is organized with specific start and end dates, durations, and dependencies to ensure a structured and progressive workflow throughout the overall project duration.

The timeline also aligns with the Agile methodology adopted in this project, where major modules such as AR Learn, AR Guide, and AI Tutor were developed iteratively across multiple sprints. Key milestones, including prototype completion in FYP I and full system integration and testing in FYP II, are clearly indicated to track progress effectively.

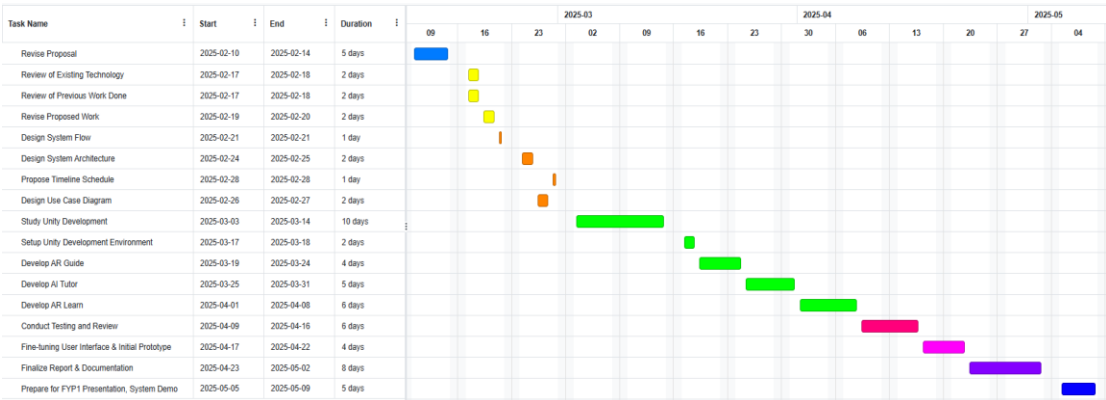


Figure 3.2 Gantt Chart for FYP 1

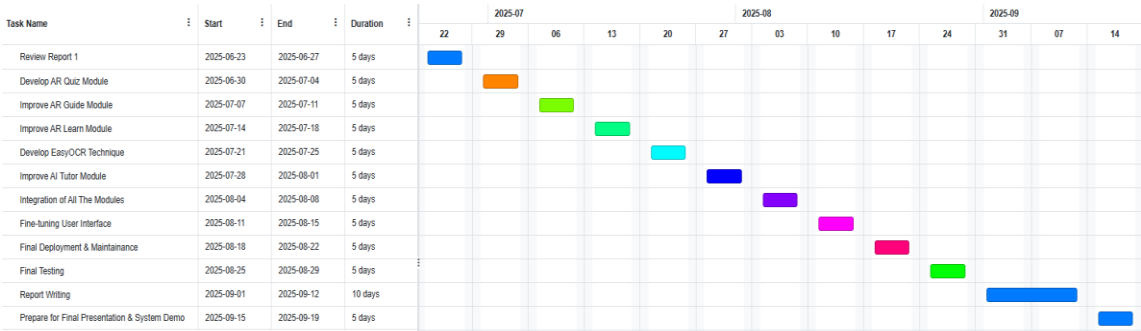


Figure 3.3 Gantt Chart for FYP 2

Chapter 4

System Design

This chapter illustrates the overall system design of the TechTutor application. It presents the system flowchart, system architecture, use case diagram, and activity diagrams that describe the structure, components, and workflows of the system. These diagrams provide a clear representation of how the application module interacts, how user engage with the system, and how the functionalities are executed to achieve the project objectives.

4.1 System Flowchart

Figure 4.1 presents the system flowchart of the TechTutor application. After launching the application, users are presented with 3 main features: **AR Guide**, **AR Learn**, and **AI Tutor**.

- In **AR Guide**, the user selects between marker-based or markerless tutorials. For marker-based tutorials, the system detects real-world objects such as the calculator and provides step-by-step instructions. For markerless tutorials, the system overlays 3D models and instructions for applications such as Microsoft Word, Gmail, and the Snipping Tool. Each tutorial continues with synchronized narration until completion.
- In **AR Learn**, the user selects different tutorials options such as identifying computer components, interacting with 3D models, or taking AR Quizzes. For AI Identify module, the capture request is sent to OpenAI, and the response is shown to the user, with the information stored in local storage. For AR Quizzes, the user selects models, and the system generates the AI-based quiz questions. The quiz results are displayed and saved for later review.
- In **AI Tutor**, the user can ask questions in text or uploading in image. For text queries, the system sends the prompt directly to OpenAI to generate a step-by-step answer. For image queries, the image is first analyzed by OpenAI to identify relevant keywords. These keywords are then passed to EasyOCR to detect their positions in the screenshot if requested by the user. The system highlights the identified regions in Canvas Mode and delivers the step-by-step

guidance. All chatbot interactions and responses are stored in local storage for future reference.

This flow ensures that users are guided through a structured and interactive learning experience by combining AR interaction and AI assistance, while maintaining local history records for review and continued learning.

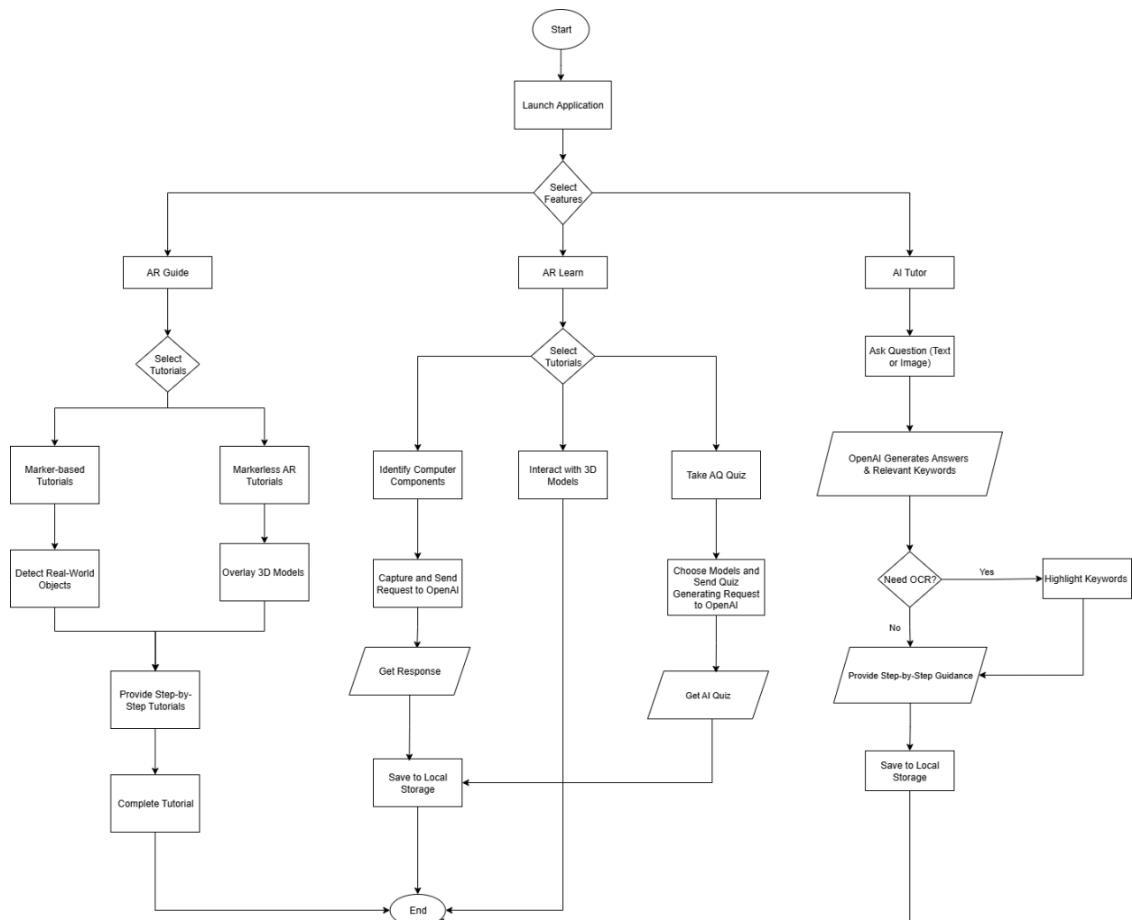


Figure 4.1 System Flowchart of TechTutor

4.2 Use Case Diagram

Figure 4.2 represents the use case diagram that shows the interaction between users and the TechTutor application. It includes the three main activities: **AR Learn**, **AR Guide**, and **AI Tutor**.

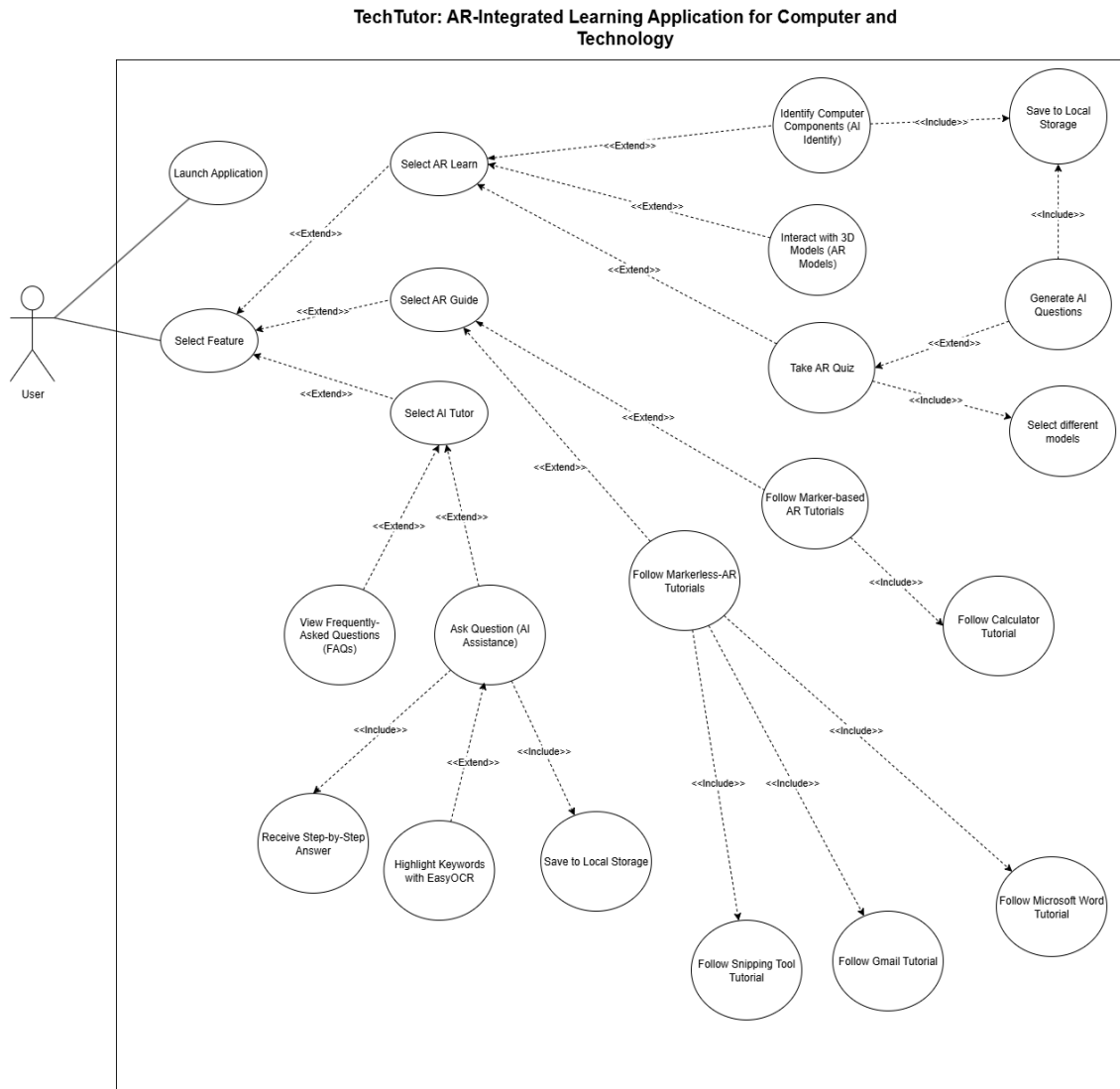


Figure 4.2 Use Case Diagram of TechTutor

- In **AR Learn**, users can choose between different learning options. They may identify basic computer components through AI Identify, interact with 3D models of computer components, or attempt an AR quiz. The system will automatically generate AI-based questions based on the models they selected if the user clicks the ‘Generate Questions’ button. All results and relevant data are saved to local storage for review.
- In **AR Guide**, user can follow both marker-based and markerless AR Tutorials. For example, marker-based tutorials include the ‘Basic Calculator’, while markerless tutorials cover applications such as Microsoft Word, Gmail, and the Snipping Tool. The system provides step-by-step guidance with synchronized narration, ensuring that users can follow instructions in an interactive and accessible way.

- In **AI Chatbot**, users can either ask questions directly (via text or image) or view frequently asked questions (FAQs). When a question is asked, the system provides a concise step-by-step answer. If an image is uploaded, EasyOCR will highlight the relevant keyword on the image and show them. All chatbot interactions and responses are also saved to local storage for future reference.

The diagram highlights how the system supports different learning actions through AR interaction and AI assistance, while maintaining local history records for review and continued learning.

4.3 System Architecture

Figure 4.3 shows the overview system architecture of the TechTutor application. The system integrates multiple components, and each of them plays a distinct role in enabling the AR interaction, AI assistance, and intelligent user guidance.

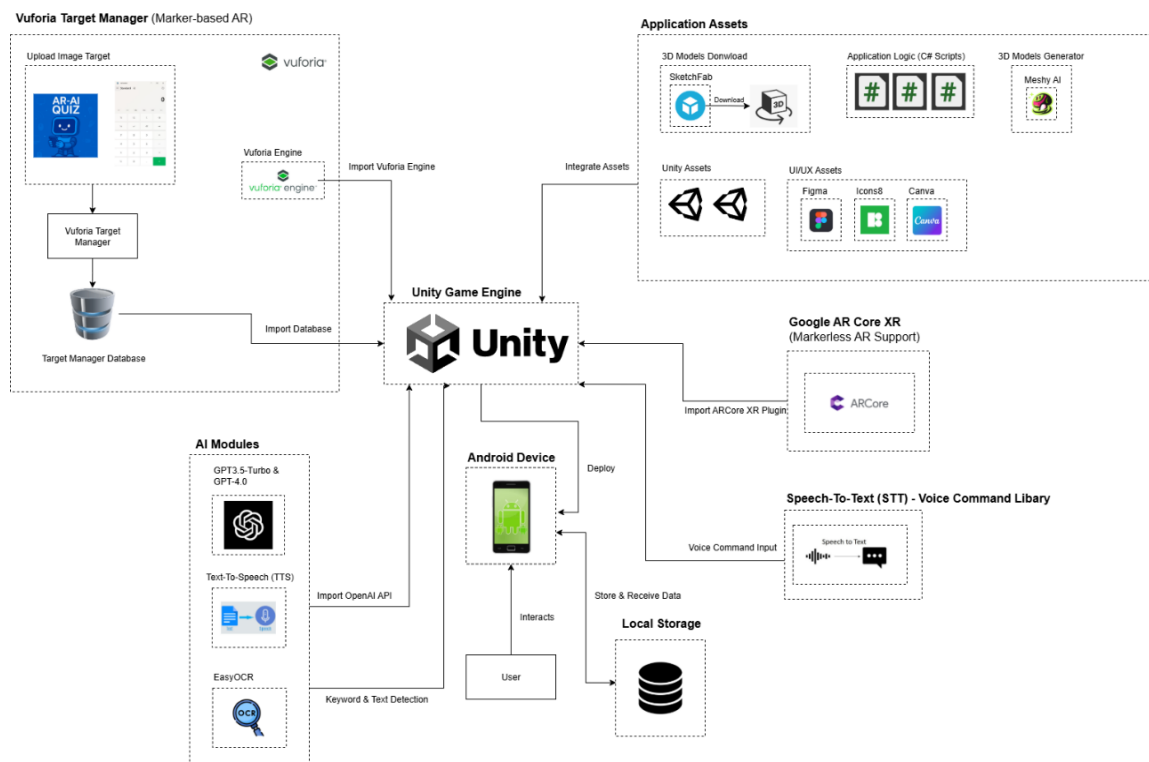


Figure 4.3 System Architecture Diagram of TechTutor

First and foremost, the **Vuforia Target Manager** is responsible for managing the image targets used in specific marker-based AR experiences, which are the AR Quiz

and ‘Simple Calculator’ tutorial. In these cases, predefined markers are uploaded as image targets, which are compiled into the **Target Manager Database**. This database is then imported into the **Unity Game Engine** via the Vuforia Engine plugin, which activates the object recognition functionality that is required for anchoring the AR content for these modules.

Besides, the **Application Assets** box includes all the elements from visual and interactive layers of the application. These includes:

- **3D Models** (It is downloaded from Sketchfab or generated using Meshy AI.)
- **C# Scripts** (It is used to define the application logic.)
- **Unity Assets** (e.g., colliders, materials, prefabs)
- **UI/UX Assets** (It is designed using Figma, Icons8, and Canva.)

All of these assets are then integrated into the **Unity Game Engine**, which acts as the main development environment for scene composition, AR rendering and the interaction handling.

At the same time, **Google ARCore XR Plugin** is imported into the Unity to enhance the AR experience with advanced environmental understanding features such as the image tracking, motion tracking and the plane detection. This helps ensure a stable and accurate placement of AR content within the user’s physical environment. **ARCore** is important for developing the **AR Guide Tutorials** and the **AR models module**, which are fully designed using the markerless AR. Examples include the ‘Send Email’ tutorial, ‘Microsoft Word’ tutorial, and ‘Snipping Tool’ tutorial, as well as the interactive 3D computer components models (keyboard, mouse, monitor, laptop and speaker).

For the **AI Modules** section, it highlights the intelligent features of the application which includes the **GPT-3.5 Turbo**, **GPT-4.0**, **Text-to-Speech (TTS)**, and **EasyOCR**.

- **GPT-3.5 Turbo and GPT-4.0** are integrated through the **OpenAI API**, and it allow users to interact with chatbot that provides real-time conversational support and generates the quiz questions and tutorial steps.
- **Text-to-Speech (TTS)** converts AI-generated contents or tutorial response into audio output, so that it can provide synchronized narration for quizzes and AR guides.

- **EasyOCR** is used to detect and extract the keywords from the images or screenshots so that it can be used to support the ‘Canvas Mode’ keyword highlighting feature.

In addition, a **Speech-to-Text (STT) Voice Command Library** is integrated separately to enable hands-free navigation. This allows the users to control tutorials or quizzes using simple voice commands such as “next” or “previous.

All the data, such as quiz history, TTS audio files, and the chatbot records are stored in **local storage** on the Android device. This ensures that the application functions offline without requiring cloud storage while still allowing users to review their past interactions and learning progress.

Lastly, once the application is developed successfully, it is deployed to an **Android Device**, where user interacts with the application in real-time. Depending on the module, Unity processes either **marker-based AR (via Vuforia)** for the AR Quiz and Calculator tutorial, or **Markerless-AR** for the AR Guide tutorials and AR Models module. The recognized objects or detected planes are then used to overlay relevant 3D content, synchronize tutorial steps, and manage the logic flow, which also includes the AI-based support such as narration, quizzes and keyword highlighting.

To conclude, this system architecture can ensure the smooth integration between AR content delivery, user interaction and the AI-enhanced learning support. It demonstrates a well-organized and scalable design for delivering an immersive, intelligent educational experiences for the users.

4.4 Activity Diagram

AR Learn Module – AI Identify

Figure 4.4 illustrates the activity diagram of the AI identify submodule under AR Learn module. The process begins when the user launches the feature and points the camera at a computer component. The captured image is then sent to the OpenAI service, which generates the recognition response. The system receives the result, displays the object information on screen, automatically plays the narration, and stores the record in the local history. The user can then choose to view more detailed information through an info panel, capture another object, review the saved history list, or exit the module to complete the activity flow.

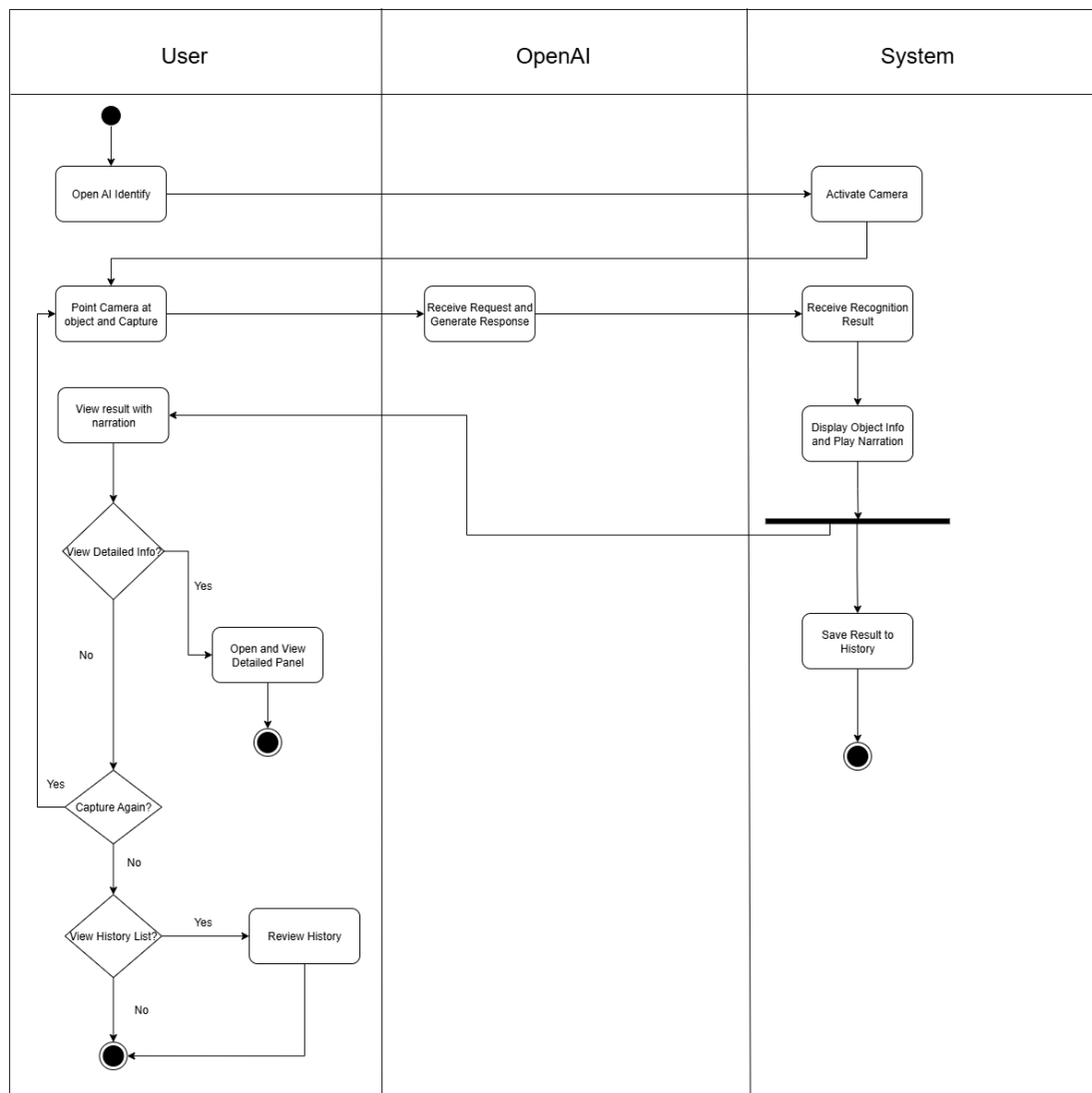


Figure 4.4 Activity Diagram for AI Identify Submodule

AR Learn Module – AR Model

Figure 4.5 illustrates the activity diagram for the AR Models submodule under the AR Learn module. The process begins when the user launches the AR Camera and moves their camera to scan the environment. Once the AR engine detects a plane, and then after user tapped and locked the plane, the system will prompt the user to select for a model. Once user selected, a preview model will spawn on the plane. The user can interact with the model through gestures such as drag, pinch, and rotate. The system will display an info panel with synchronized TTS narration. User may choose to select another model or exit the module to complete the activity flow.

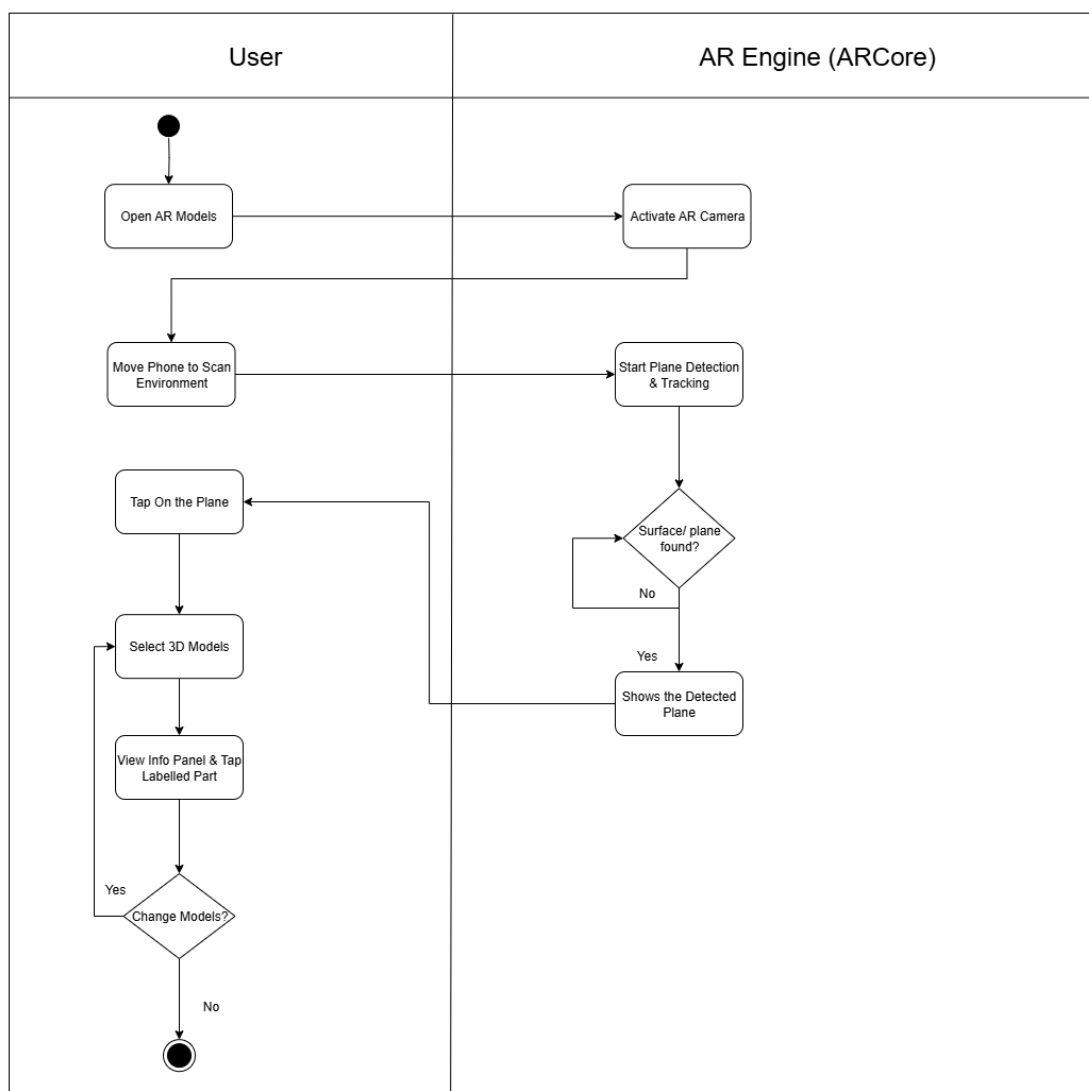


Figure 4.5 Activity Diagram for AR Models Submodule

AR Learn Module – AR Quiz

Figure 4.6 illustrates the activity diagram for the AR Quiz submodule under the AR Learn module. The process begins when the user launches the AR Quiz and points the camera at a quiz marker. The AR engine detects the marker and spawns the associated 3D model. Once the user selects a model, and requests for generating quiz, the system will send a request to OpenAI to generate quiz questions. The generated questions are received, displayed on a quiz panel and narrated using TTS. The user then answers the question, and the system provides feedback accordingly. All questions and attempts are stored in the quiz history. The user may choose to change the model, review the past quiz history, or exit the module to complete the activity flow.

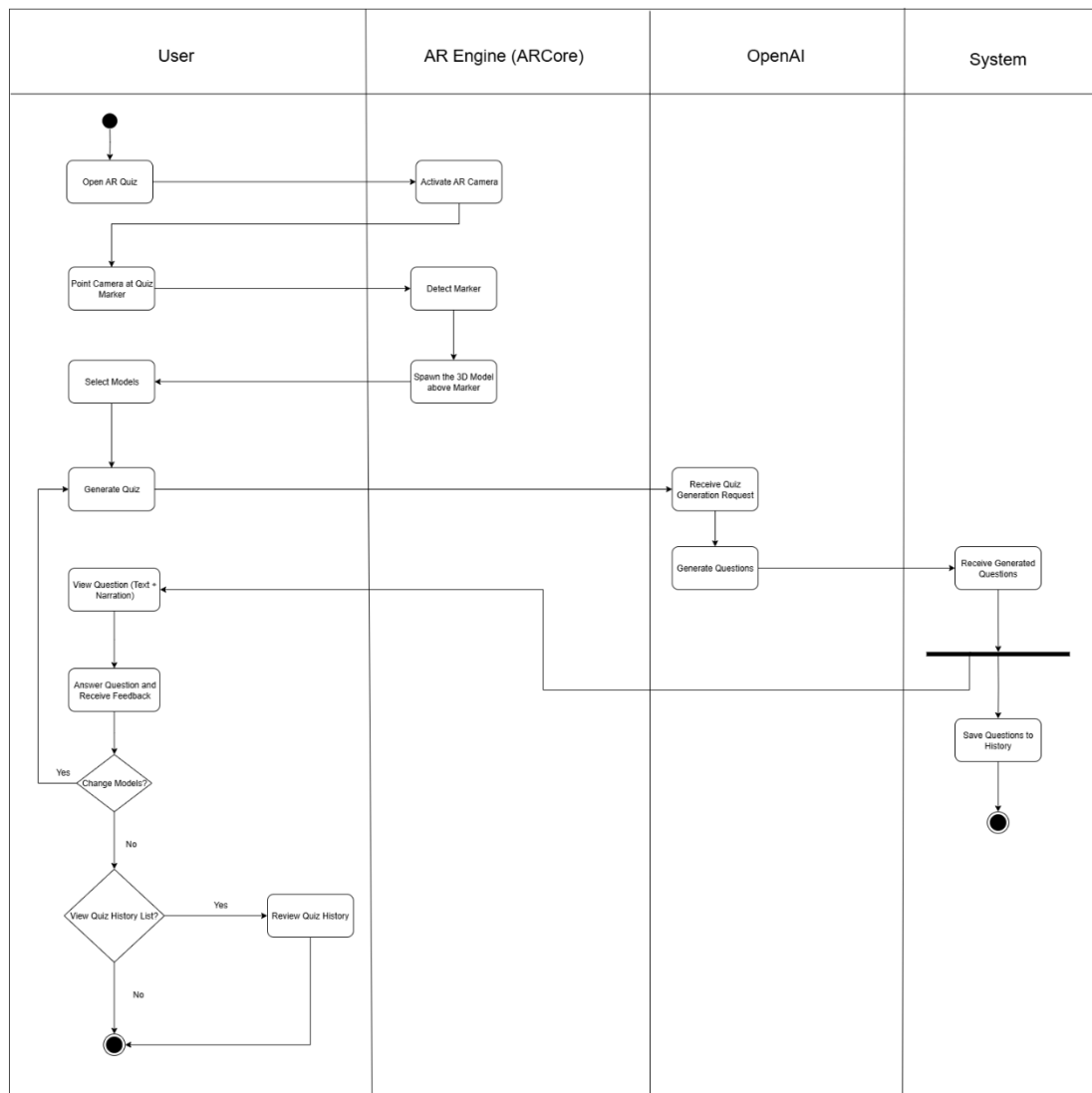


Figure 4.6 Activity Diagram for AR Quiz Submodule

AR Guide – Microsoft Word Tutorial, Send Email Tutorial, and Snipping Tool Tutorial

Figure 4.7 illustrates the shared activity diagram for the AR Guided tutorials covering Microsoft Word, Snipping Tool, and Send Email. The process begins when the user launched the AR Guide and move his phone to scan the environment. The AR engine detects a plane, and the system spawns the required virtual 3d models, which are monitor, keyboard, and mouse. Each tutorial step is displayed with synchronized TTS navigation and visual highlights. The user may proceed through the tutorial using the button on the virtual info panel or voice commands and can toggle between 3D AR Mode and Canvas Mode at any time while maintain the synchronized progress. The activity concludes one all tutorial steps are completed.

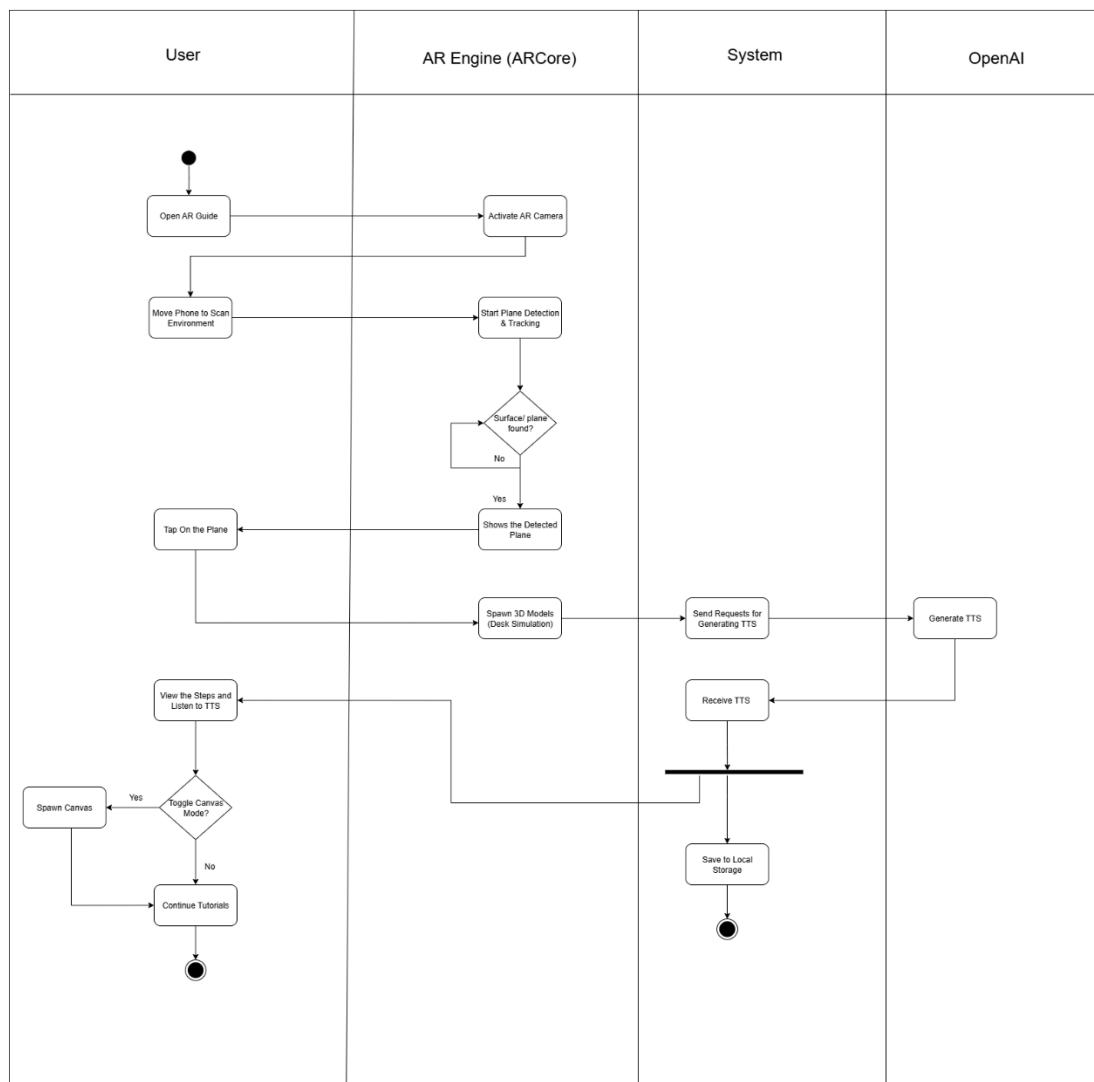


Figure 4.7 Activity Diagram for AR Guide Tutorials (Microsoft Word, Snipping Tool, Send Email)

AR Guide – Calculator Tutorials

Figure 4.8 illustrates the activity diagram for the calculator tutorial under the AR Guide module. The user begins by pointing the device at the calculator at their monitor or laptop and waiting for it to be detected and tracked by the AR engine. The system overlays an info panel besides the calculator and provides step-by-step instructions. The user navigates through the tutorial using the buttons on the panel or speech commands (“next”, “back”). The activity concludes after completing the guided example calculation, marking the end of the tutorial.

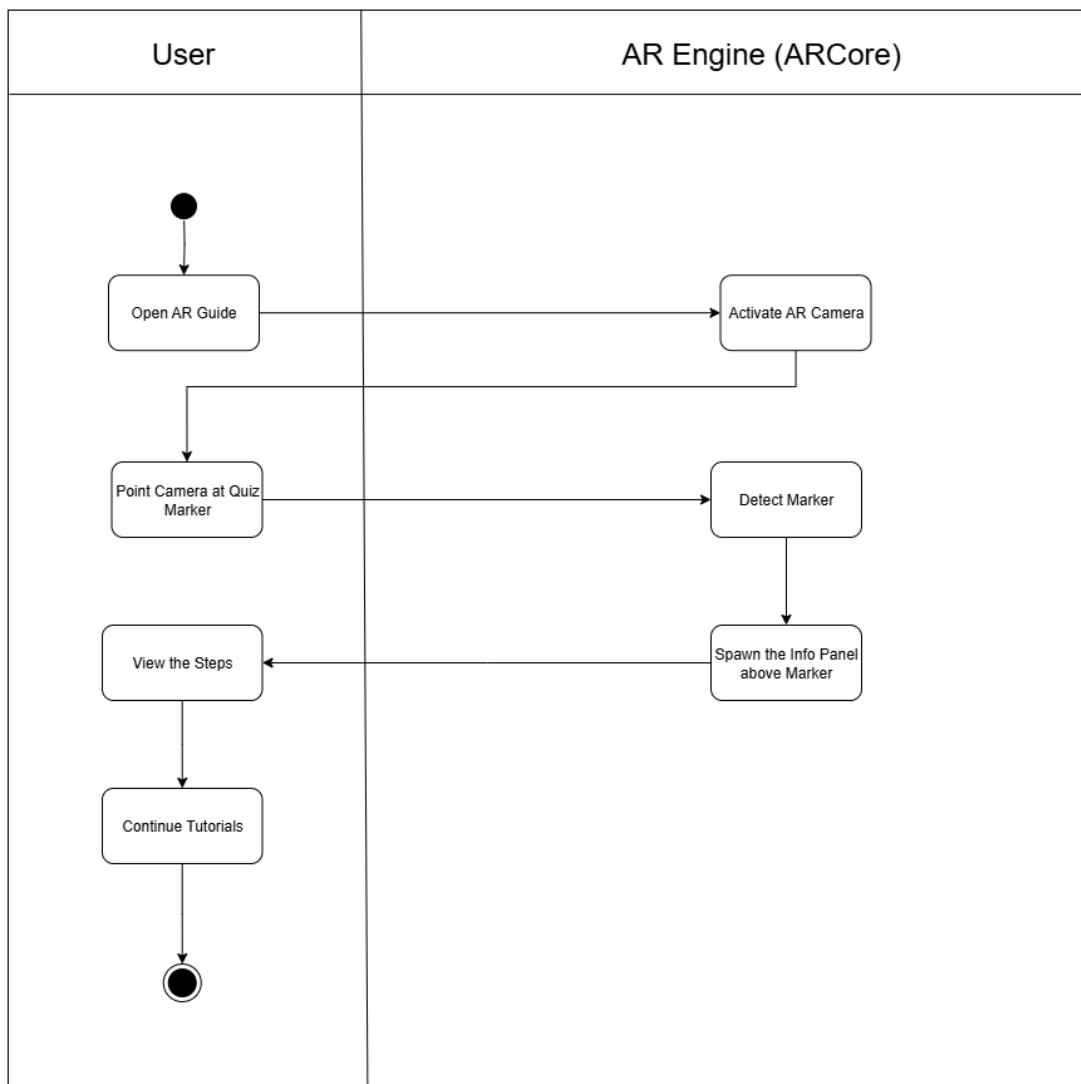


Figure 4.8 Activity Diagram for AR Guide – Calculator Tutorial

AI Tutor – AI Assistance (Chatbot)

Figure 4.9 illustrates the activity diagram for the AI Assistance submodule under the AI Tutor module. The process begins when the user opens the AI Tutor and chooses to input either a text question or an image. For text queries, the system builds the prompt and sends it directly to OpenAI. For image queries, the image is first sent to OpenAI to identify relevant keywords based on the user's request. These keywords are then passed to EasyOCR, which scans the image to determine the positions of the keywords within the screenshot. The system overlays the highlighted regions in Canvas Mode and displays OpenAI's step-by-step answer. The user may ask another question or review the history list. The steps are stored locally for later reference. The activity concludes when the user exits the module, completing the activity flow.

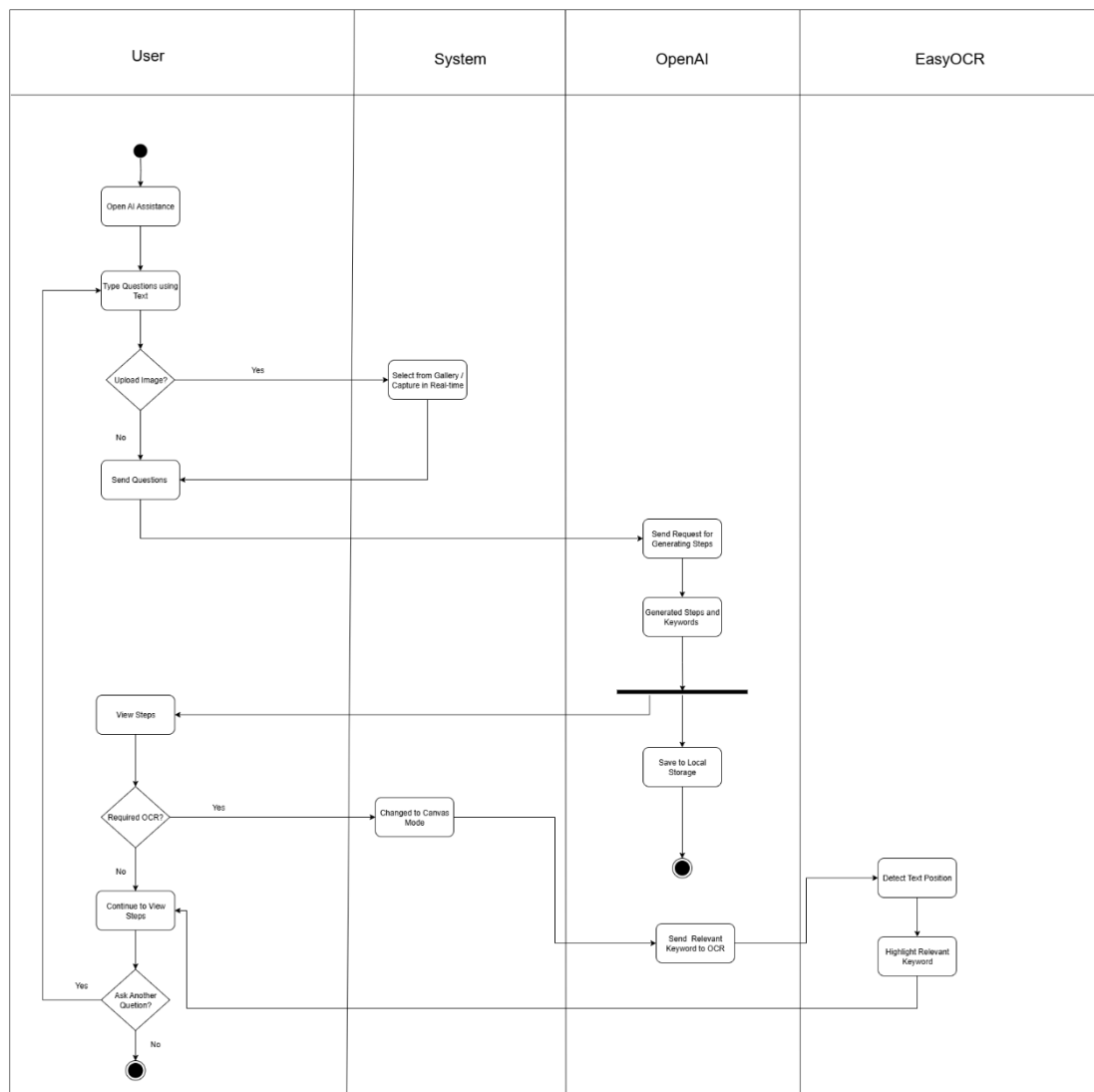


Figure 4.9 Activity Diagram for AI Assistance Submodule

Chapter 5

System Implementation

This chapter summarizes the practical realization of the TechTutor application. It describes the software and hardware environment required for development, followed by the configuration of the key technologies such as Unity, Vuforia, ARCore, OpenAI, and EasyOCR. The system operation is then detailed, presenting how the 3 major modules, AR Learn, AR Guide and AI Tutor function within the application. Furthermore, this chapter discusses the challenges encountered during implementation and the strategies employed to address them. The chapter concludes with a reflection on the overall development process and its alignment with the project objectives.

5.1 Software Setup

5.1.1 Unity Setup

Unity 2022.3.57f1 (LTS) was installed as the primary development environment. The Long-Term Support (LTS) version was chosen to ensure stability and compatibility for AR development.

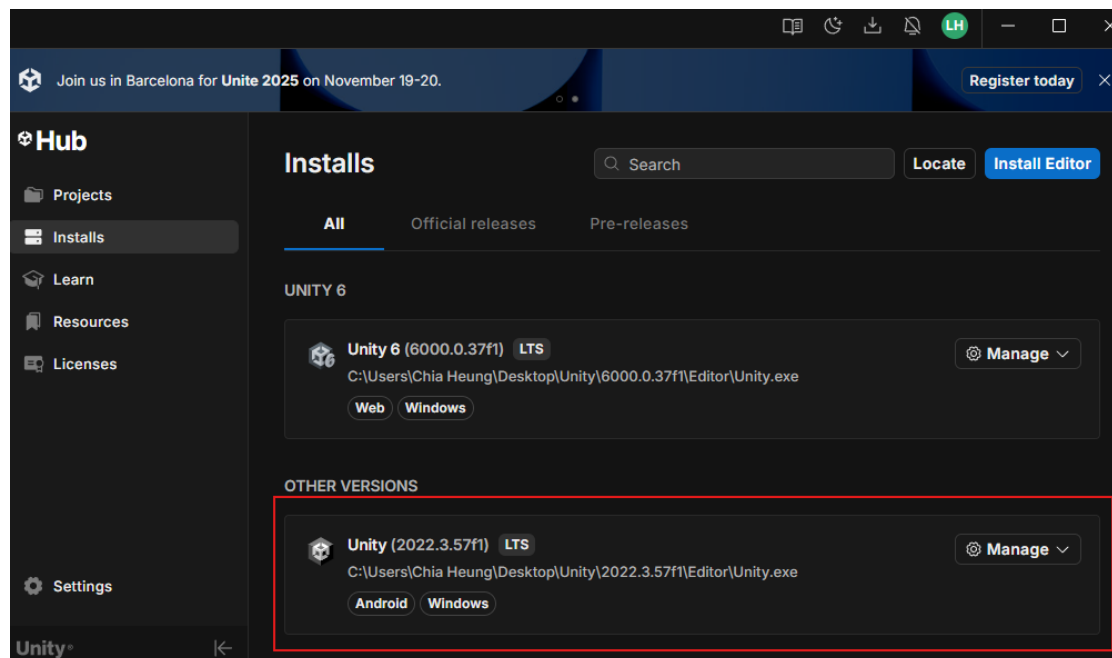


Figure 5.1 Unity 2022.3.57f1(LTS)

5.1.2 Vuforia Developer Portal

The **Vuforia Developer Portal** was used to prepare marker-based AR resources for the application. After logging into the portal, developer accessed to the ‘**Target Manager**’ tab to create a new database. Then, a database name was entered, and the target type was selected. Once confirmed, the database was generated successfully and made available for later use in the Unity Project.

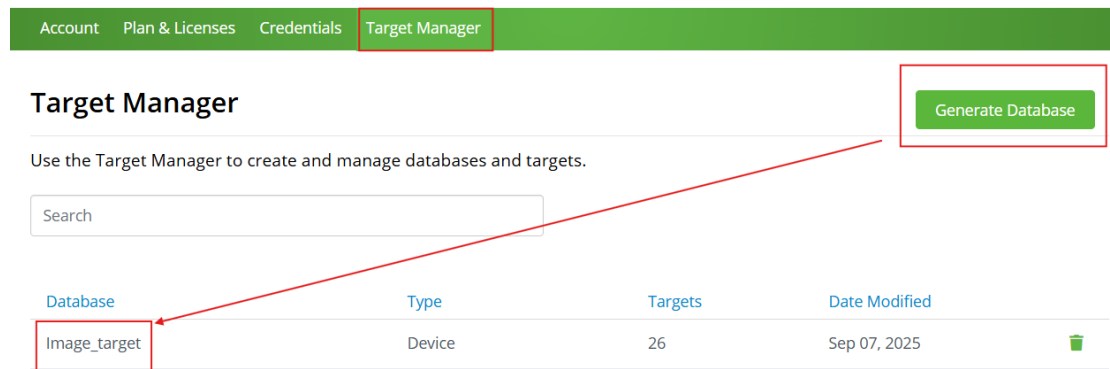


Figure 5.2 Vuforia Developer Portal Setup

5.1.3 Vuforia Engine AR Package

The **Vuforia Engine AR Package** was installed in Unity Package Manager. This package provides the core functionalities for marker-based AR on both the Android and iOS device. This package enables the application to recognize and render the image targets that were created in the **Vuforia Engine Developer Portal**. By integrating this package, the system was able to anchor 3D models and instructional content onto the detected markers. The installed version for this project is 11.0.4.

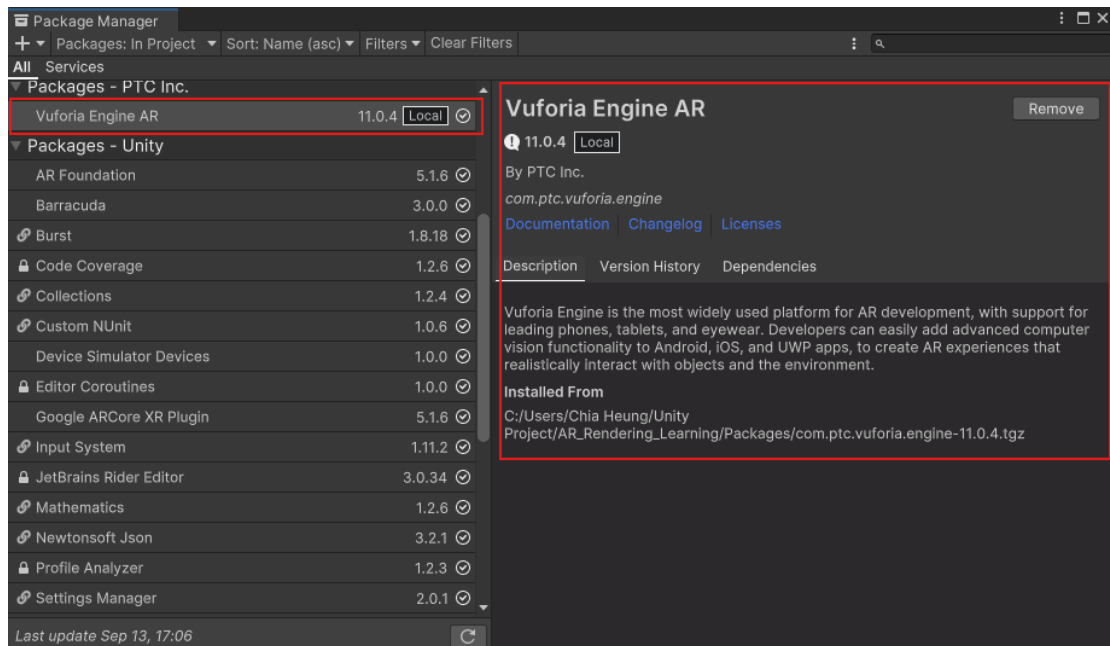


Figure 5.3 Vuforia Engine AR Package Setup

5.1.4 ARCore XR Plugin

The **ARCore XR Plugin** was installed through the Unity Package Manager to support markerless AR development on **Android devices**. This plugin enables plane detection, motion tracking, and environmental understanding, which were essential for implementing the AR Guide tutorials and AR Models features. The installed version for this project is 5.1.6.

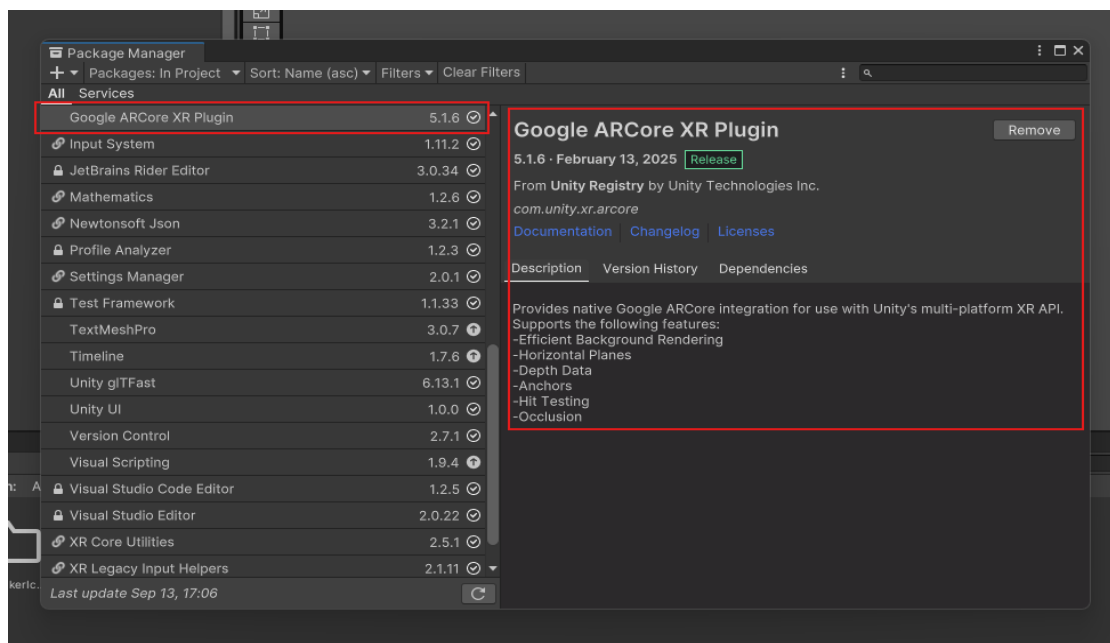


Figure 5.4 ARCore XR Plugin Setup

5.1.5 Speech-to-Text (STT) Plugin

The **Speech-to-Text (STT) plugin** was installed in Unity using a Git URL link through the Package Manager. This plugin allows the application to convert spoken voice commands into text on Android and iOS devices. It was primarily used to enable hands-free navigation in AR Guide tutorials (for example, the “next” and “previous” commands) and to support interactions in AR Learn modules. The installed version is 1.1.1.

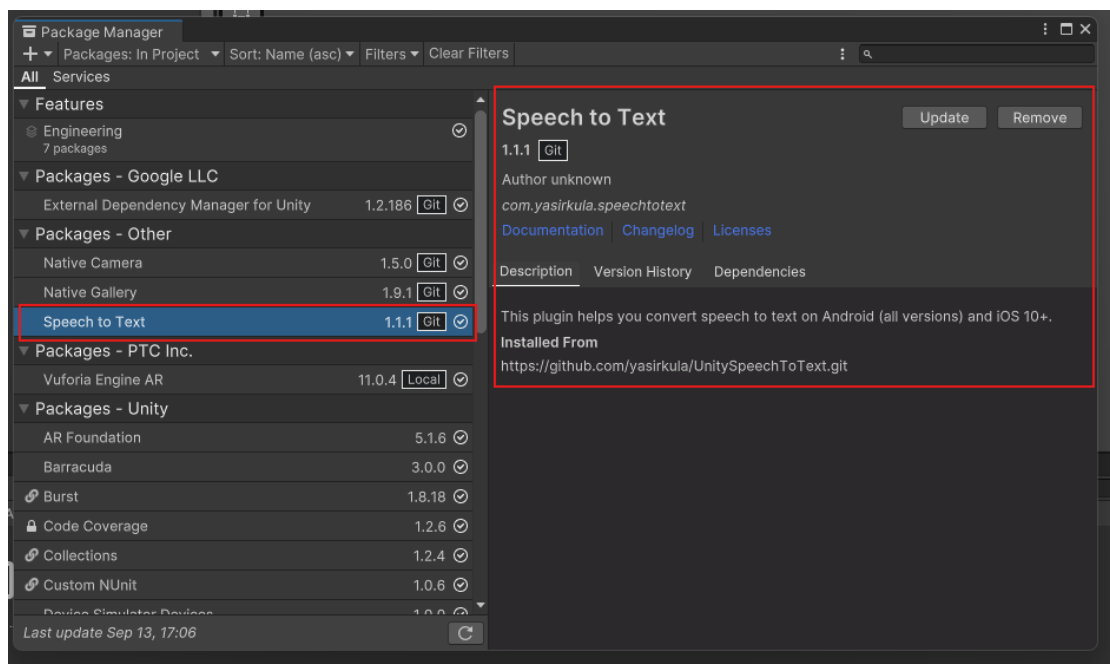


Figure 5.5 Speech-to-Text (STT) Plugin Setup

5.1.6 Native Camera

The **Native Camera plugin** was installed in Unity using a Git URL link. This plugin allows the application to capture photos and record videos directly from the device’s camera on both Android and iOS. It was mainly used in the AI Tutor module to let users take snapshots for image-based questions. The installed version is 1.5.0.

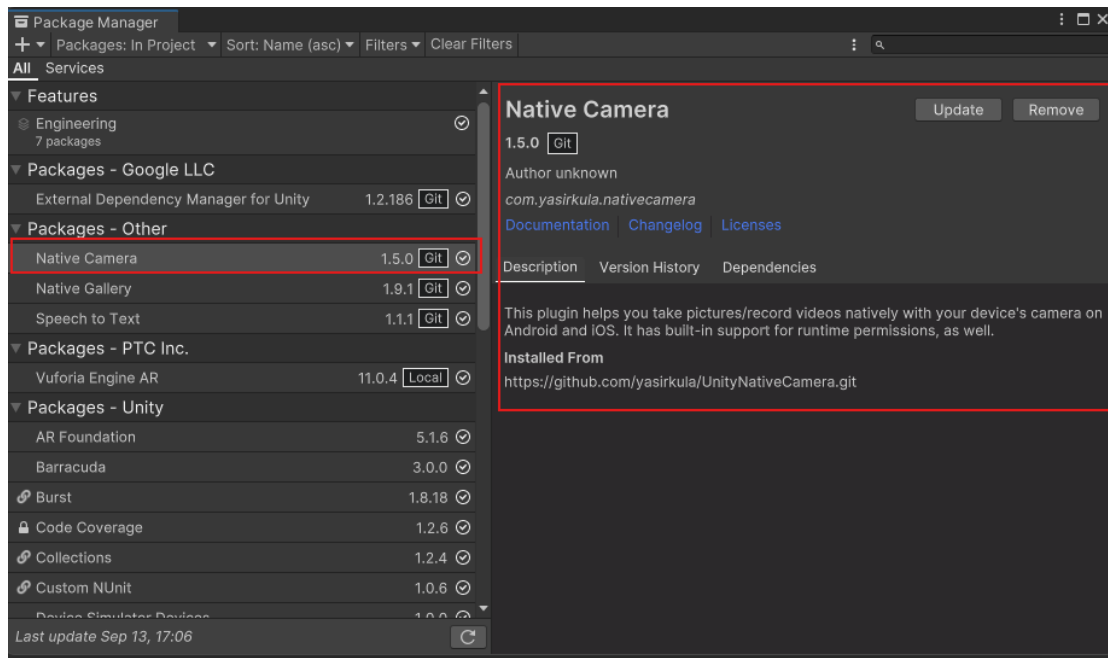


Figure 5.6 Native Camera Plugin Setup

5.1.7 Native Gallery

The **Native Gallery plugin** was installed in Unity using a Git URL link. This plugin provides access to the device's gallery, allowing the application to save images or videos, as well as to select files from the gallery. It was used in the AI Tutor module to let users upload images from their phone storage for OCR keyword detection. The installed version is 1.9.1.

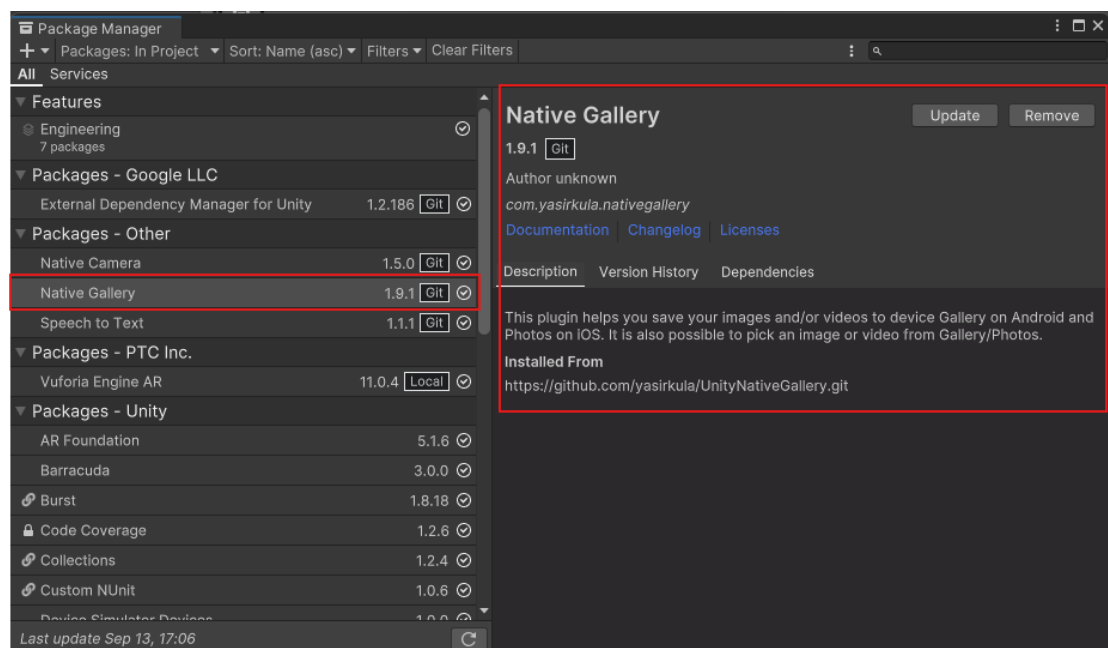


Figure 5.7 Native Gallery Plugin Setup

5.1.8 AR Foundation and XR Plugin Management

The **AR Foundation** package was installed through the Unity Package Manager as the core framework for augmented reality development. It provides essential cross-platform functionalities such as plane detection, anchors, hit testing, and face tracking, which are required for AR interaction. Alongside this, XR Plugin Management was enabled automatically to manage the active AR backends. Through this system, the ARCore XR Plugin was configured for markerless AR features on Android devices, while the Vuforia Engine was configured for marker-based AR features on both Android and iOS devices. The installed version is 5.1.6.

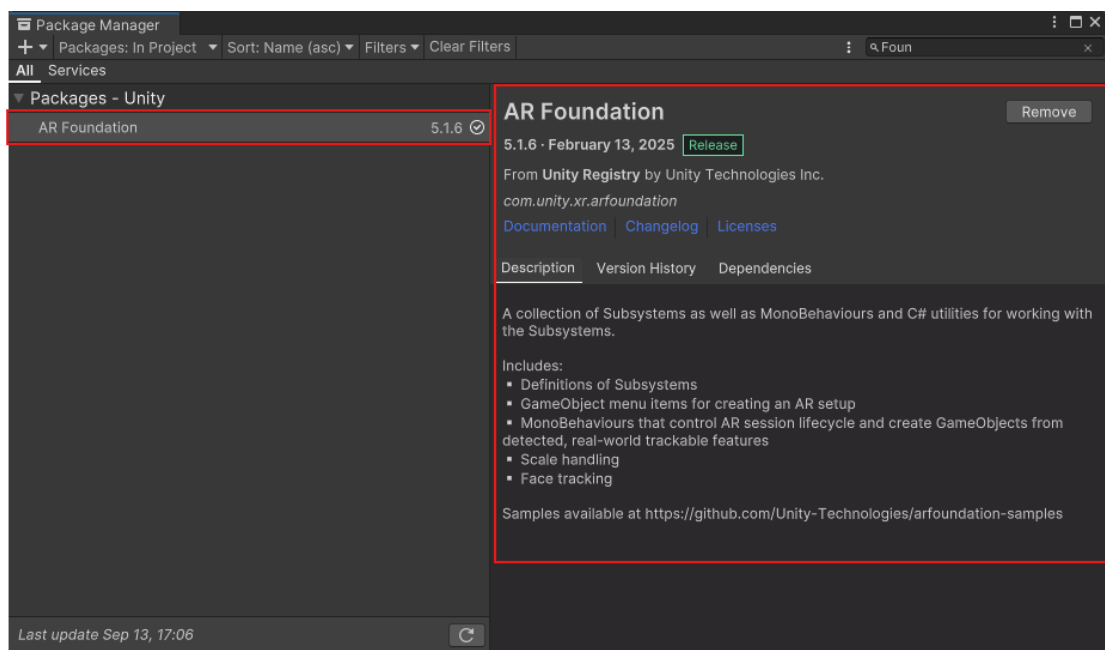


Figure 5.8 AR Foundation and XR Plugin Management Setup

5.1.9 Android Studio

Android Studio was installed to support application debugging on Android devices. It was primarily used for accessing the Logcat tool, which displayed real-time system logs, error messages, and console outputs generated by the Unity application during testing. The installed version is Android Studio Hedgehog 2023.1.1 Patch 2

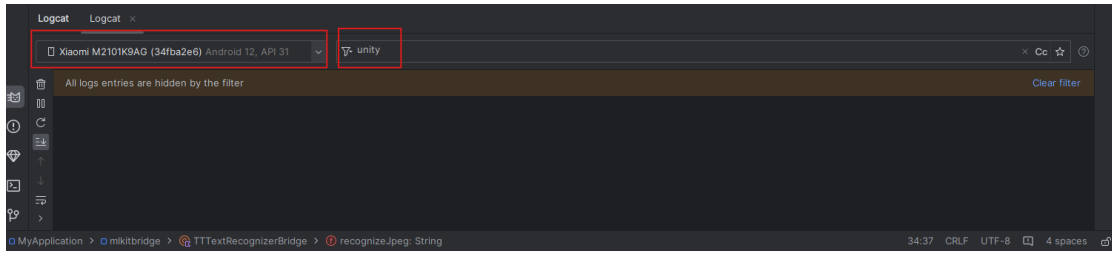


Figure 5.9 Android Studio Logcat Window

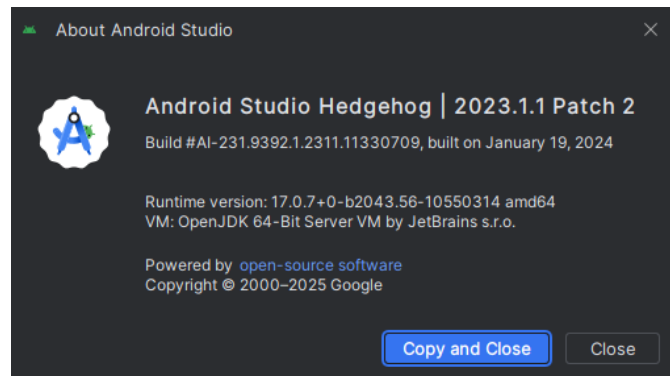


Figure 5.10 Android Studio Version

5.1.10 Sketchfab

Sketchfab was used as an online platform to source and download pre-built 3D models for the application. Computer component models such as the keyboard, monitor, mouse, and speaker were obtained from Sketchfab and imported into Unity for use in the AR Learn and AR Guide modules. This reduced development time while ensuring that realistic models were available for interactive learning.

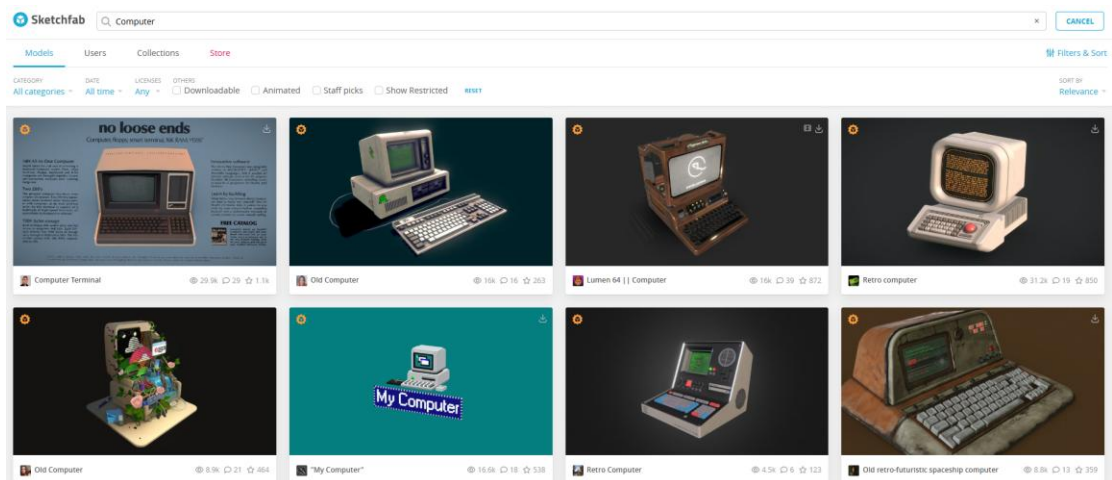


Figure 5.11 Sketchfab Website

5.1.11 GitHub

GitHub was used for version control and project backup throughout the development of the TechTutor application. The Unity project files were committed and pushed to a private GitHub repository, ensuring proper version tracking and recovery in case of errors. This also provided a structured workflow for managing updates and maintaining the source code.

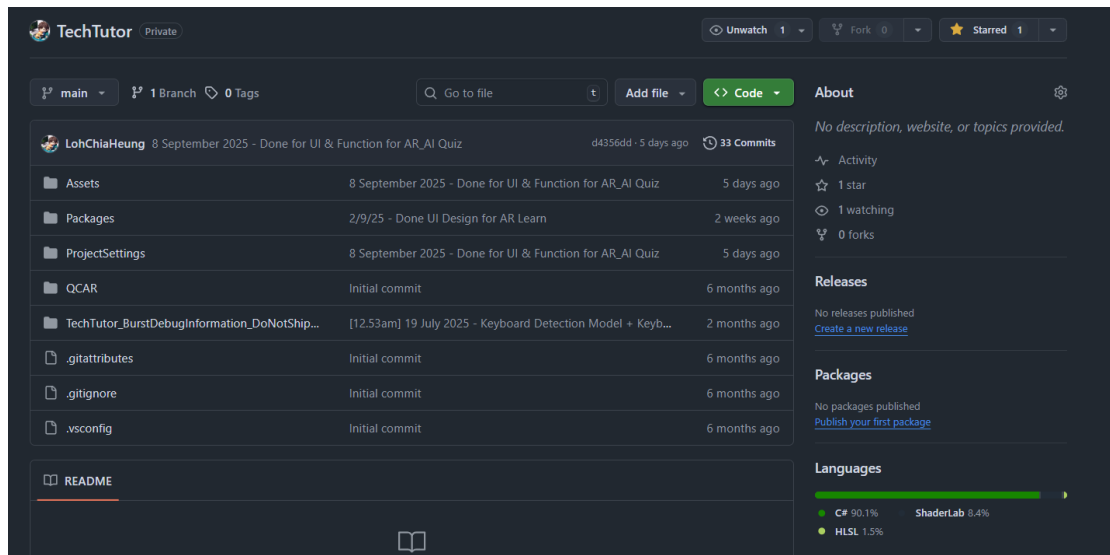


Figure 5.12 GitHub Repository for TechTutor

5.1.12 Meshy.ai

Meshy.ai was used to generate custom 3D models from 2D image inputs. For example, an AR Robot model was created and later integrated into the AR Quiz feature. This tool helped to supplement pre-built assets from Sketchfab by providing unique models tailored to the project's requirements.

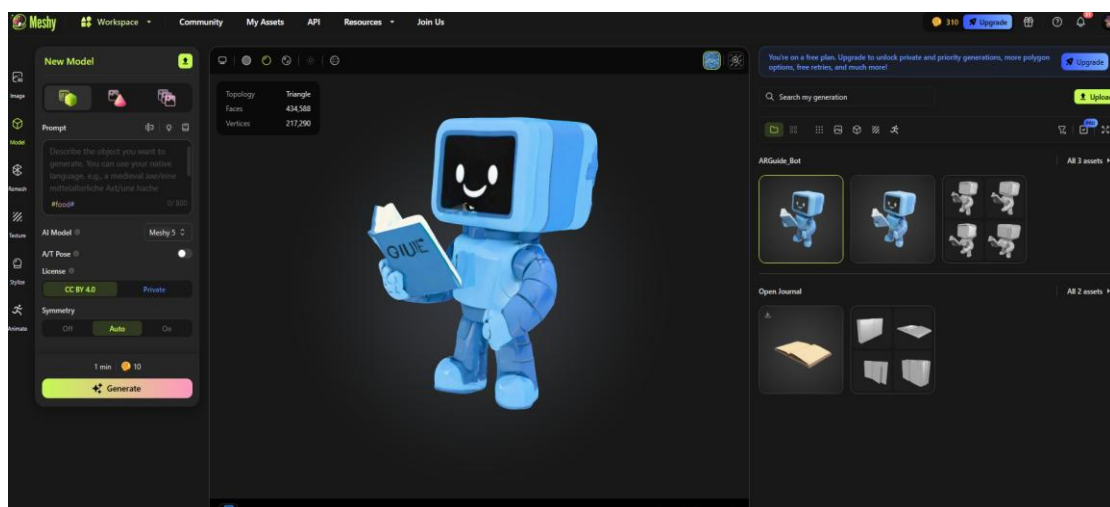


Figure 5.13 AR Robot Generated from Meshy.ai

5.1.13 Figma

Figma was used to design the user interface (UI) layouts and icons for the TechTutor application. The interface elements, including buttons, menus, and panels, were first prototyped in Figma and later exported for integration into Unity. This ensured a consistent and user-friendly design throughout the application.

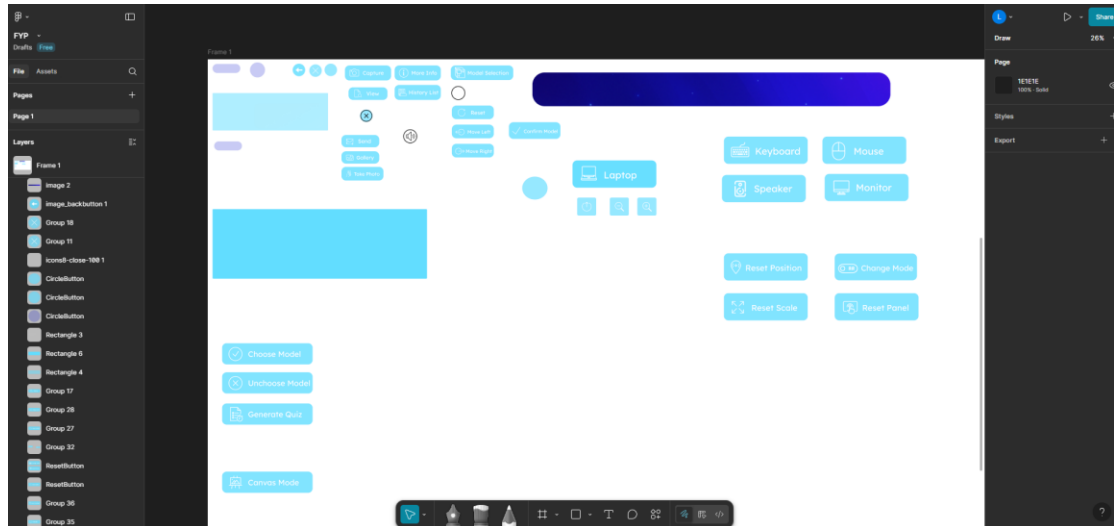


Figure 5.14 Figma Workspace

5.1.14 Icons8

Icons8 was used as an additional design resource to obtain icons and graphic assets for the TechTutor application. These assets were used to complement the user interface elements designed in Figma, ensuring a clean and professional appearance for the application's menus and panels.

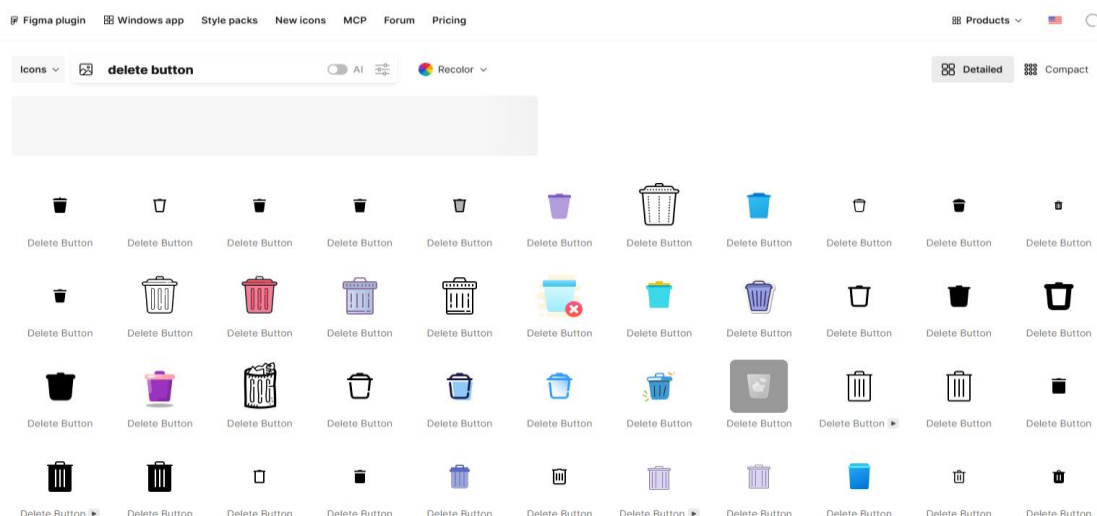


Figure 5.15 Icons8 Website Interface

In summary, the above software tools and packages were installed and configured to establish the development environment for the TechTutor application. These tools provided the necessary support for AR functionality, AI integration, 3D asset generation, and user interface design, ensuring a smooth workflow throughout the project.

5.2 Hardware Setup

The hardware used to develop and test the TechTutor application included a laptop and an Android mobile device. The laptop was primarily used for Unity development, system configuration, and hosting the Flask backend for EasyOCR, while the mobile device was used for deploying the APK, testing AR functionality, and validating the Speech-to-Text (STT) and Text-to-Speech (TTS) features. Detailed specifications are provided in Tables 5.1 and 5.2.

Table 5.1 Laptop Specifications

Description	Specifications
Model	ROG Strix G16
Processor	13th Gen Intel(R) Core™ i7-13650HX
Operating System	Windows 11
Graphic	NVIDIA GeForce RTX 4050
Memory	16GB
Storage	1TB SSD

Table 5.2 Mobile Device Specifications

Description	Specifications
Model	Mi 11 Lite
Processor	Qualcomm Snapdragon 732G (8nm) CPU: Octa-core Max 2.3GHz
Operating System	Android 11 (MIUI 12), upgradable to Android 13 (MIUI 14)
Graphic	Adreno 618
Memory	8GB
Storage	128GB
Camera	64MP primary camera (f/1.79, 6P lens) 8MP ultra-wide angle camera (119° FOV, f/2.2)

	5MP telemacro camera (f/2.4, Contrast AF, 3cm-7cm)
--	--

5.3 System Configuration

5.3.1 Vuforia Engine Configuration

To configure the **Vuforia Engine** for marker-based AR, a license key first needed to be obtained from the **Vuforia Developer Portal**. After logging in, the developer navigated to the ‘Plan & Licenses’ tab and clicked ‘Generate Basic License’ to create a new license. As shown in Figure 5.16, an example license “Test_App1” was generated. By selecting the license, the license key was displayed, as illustrated in Figure 5.17, and then copied for use in Unity.

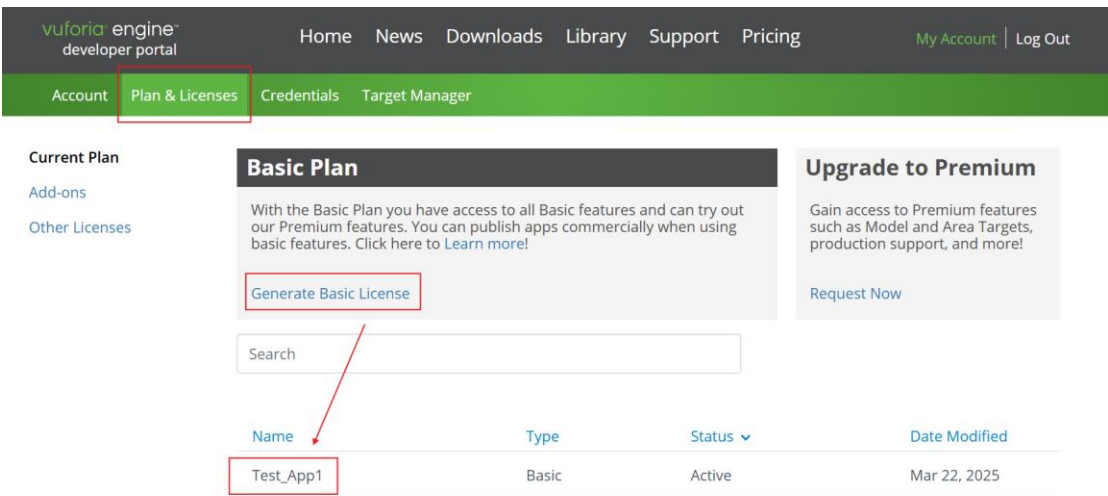


Figure 5.16 Generate License Key

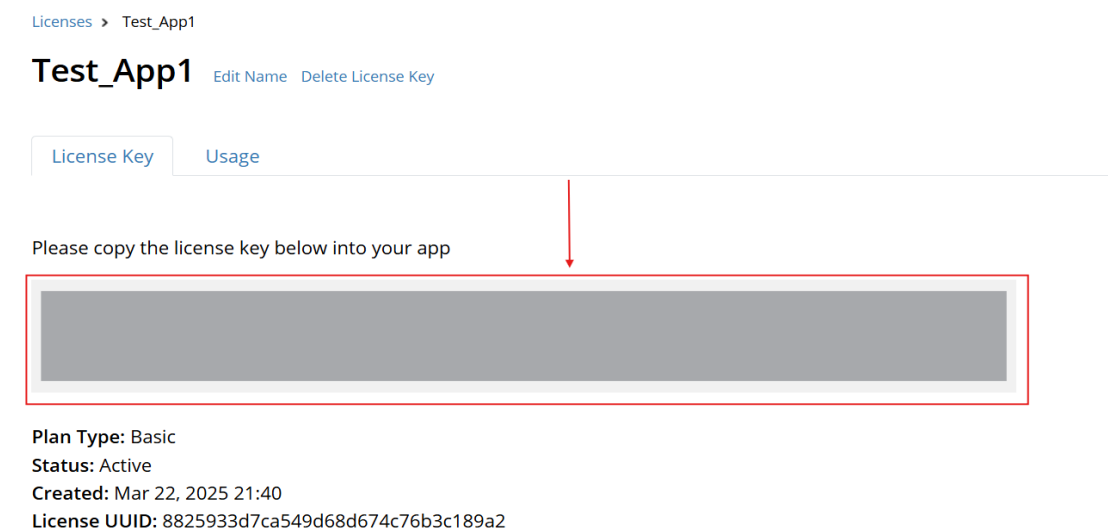


Figure 5.17 Obtained License Key

In Unity, the license key was applied by right-clicking in the Project panel and creating a **Vuforia Engine > AR Camera**. The AR Camera contained a Vuforia Behaviour script, where the developer could click ‘Open Vuforia Engine Configuration’ and paste the license key, as shown in Figure 5.18 and Figure 5.19. This step activated Vuforia in the project.

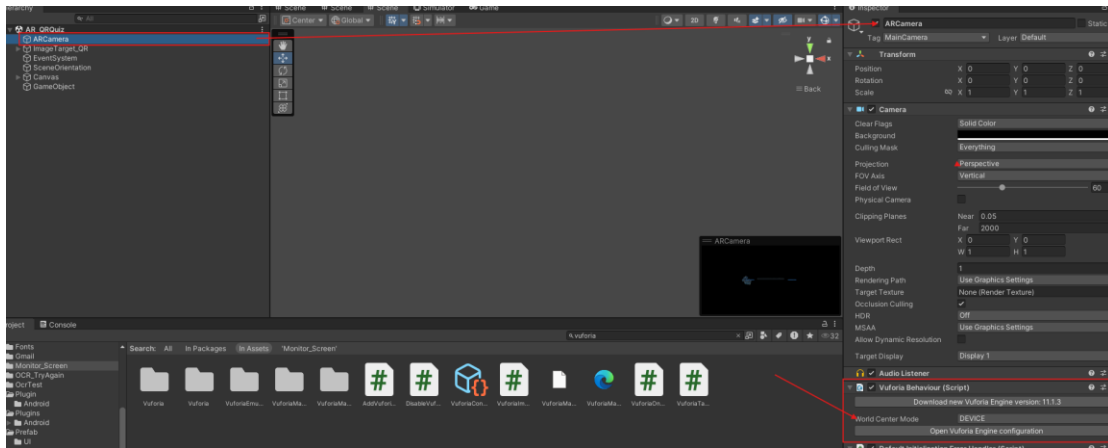


Figure 5.18 Open Vuforia Engine Configuration

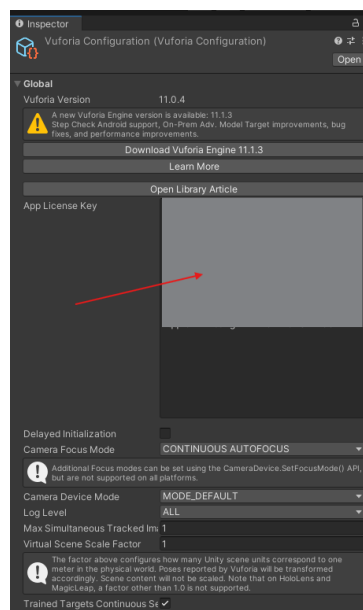


Figure 5.19 Paste License Key

Once activated, the engine was able to recognize image targets defined in the database. The database created earlier in the Vuforia Developer Portal was imported into Unity and linked with the project. The developer then created an Image Target object under the Vuforia Engine. As shown in Figure 5.20, the Type option was set to “From Database”, the imported database was selected, and the desired image target name was chosen. A 3D model or other content could then be placed as a child object of the Image

Target. When the user pointed their device camera at the image target, the corresponding model was spawned in the AR scene.

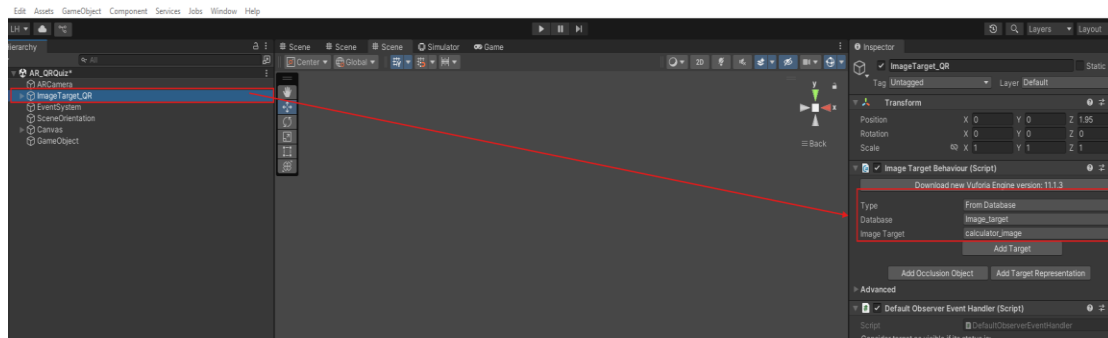


Figure 5.20 Image Target Behaviour Configuration

5.3.2 ARCore & AR Foundation Configuration

For markerless AR development, the AR Foundation and ARCore XR Plugin package were configured in Unity. First, the developer enabled the XR Plugin Management by navigating to Edit > Project Settings > XR Plug-in Management. As shown in Figure 5.21, the Android build platform was selected, and Google ARCore was enabled as the active provider.

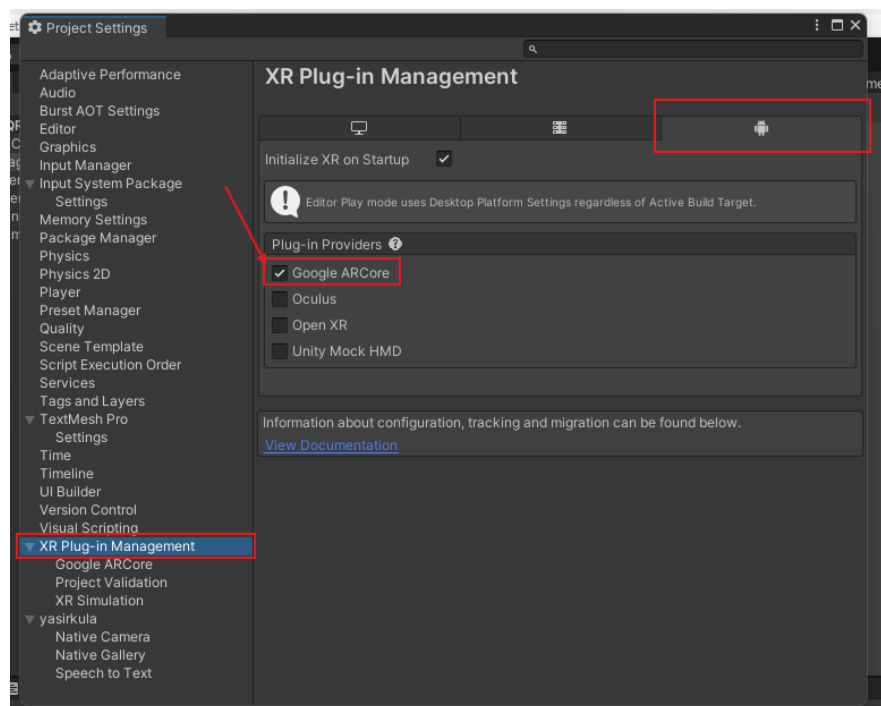


Figure 5.21 Activated Google ARCore

The AR Foundation package served as the cross-platform framework, providing a unified API for plane detection, anchors, and hit testing. Meanwhile, the ARCore XR

Plugin acted as the backend for Android devices, handling the actual motion tracking and environmental understanding.

After configuration, Unity's **AR Session** and **XR Origin (Mobile AR)** prefabs were added to the scene, as shown in Figure 5.22. The **AR Session** controlled the lifecycle of the AR experience, while the **XR Origin (Mobile AR)** defined the point of reference for placing virtual objects into the real-world environment.

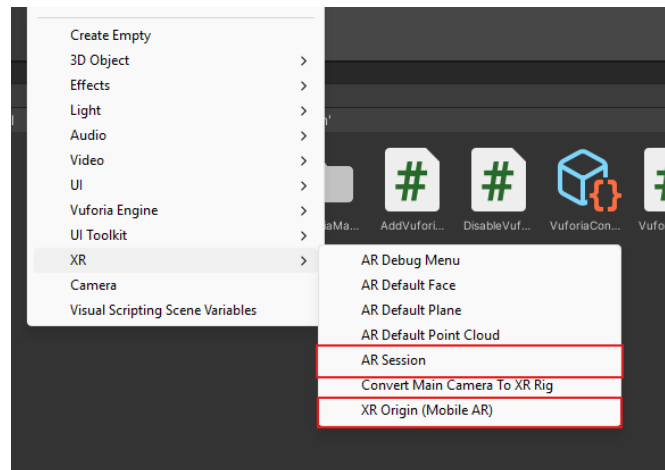


Figure 5.22 AR Session and XR Origin (Mobile AR) Prefabs

After the configuration, it allowed TechTutor application to support markerless AR functions, particularly for the AR Guide tutorials and AR Models feature, where 3D objects could be anchored onto the detected surfaces.

5.3.3 OpenAI API Configuration

The OpenAI API was configured to enable the AI-driven features in the TechTutor application. An API key was first obtained from the OpenAI Developer Portal by clicking Create new secret key. The developer was then prompted to assign a name to the key and create a project. As shown in Figure 5.23, the project “OpenAI” was created.

After the project was created, an information panel appeared displaying the newly generated secret key as shown in Figure 5.24. This key is only shown once and cannot be retrieved again; therefore, it had to be copied and stored securely for later use in Unity.

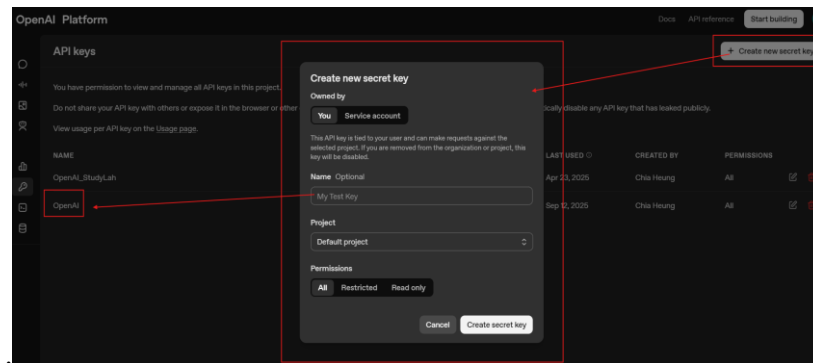


Figure 5.23 Creating OpenAI Secret Key

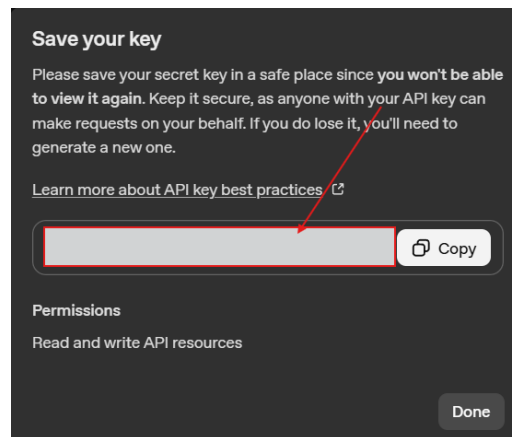


Figure 5.24 Generated OpenAI Secret Key

After securely storing the secret key, the developer integrated it into Unity through C# scripts. As shown in Figure 5.25, the `UnityWebRequest` class was used to send HTTP POST requests to the OpenAI endpoint. The request included the authorization header with the API key and JSON-formatted body parameters. Upon receiving a response, the result was parsed using Unity's `JsonUtility` to extract the generated output for use within the application.

```

using (var req = new UnityWebRequest(url, "POST"))
{
    req.uploadHandler = new UploadHandlerRaw(Encoding.UTF8.GetBytes(jsonBody));
    req.downloadHandler = new DownloadHandlerBuffer();
    req.SetRequestHeader("Authorization", "Bearer " + config.apiKey);
    req.SetRequestHeader("Content-Type", "application/json");

    yield return req.SendWebRequest();

    if (req.result != UnityWebRequest.Result.Success)
    {
        var msg = $"HTTP {req.responseCode} {req.error}\n{req.downloadHandler.text}";
        Debug.LogError("[AIQuizGenerator] " + msg);
        OnError?.Invoke(msg);
        yield break;
    }

    var resp = JsonUtility.FromJson<ChatResp>(req.downloadHandler.text);
    string content = resp != null && resp.choices != null && resp.choices.Length > 0
        ? resp.choices[0].message.content
        : null;

    if (string.IsNullOrEmpty(content))
    {
        Debug.LogError("[AIQuizGenerator] Empty content.");
        OnError?.Invoke("Empty content");
        yield break;
    }
    if (logResponse) Debug.Log("[AIQuizGenerator] JSON:\n" + content);
}

```

Figure 5.25 OpenAI API Integration using UnityWebRequest

5.3.4 EasyOCR Configuration

The **EasyOCR** library was integrated through a Flask backend server to provide text recognition functionality for the TechTutor application. This setup enabled the application to process uploaded or captured images and highlight relevant keywords in the AI Tutor module.

Before running the server, the required Python dependencies were installed, including:

- flask – for hosting the backend server
- flask-cors – to allow cross-origin requests from Unity
- pillow – for image processing
- numpy – for handling image data arrays
- easyocr – for performing optical character recognition

These libraries were installed using pip and stored in a requirements.txt file for environment setup.

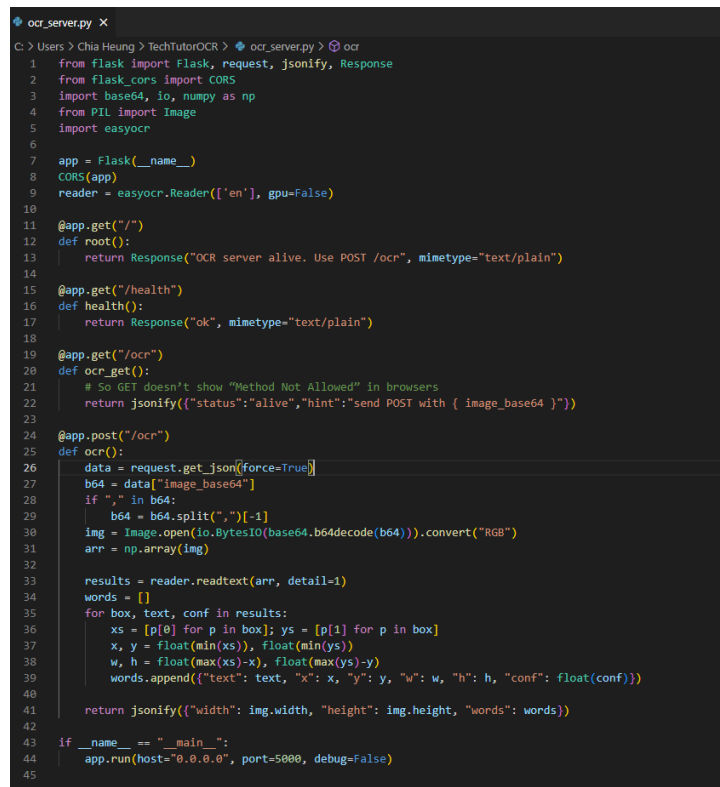
Next, a Flask server was developed in Python with EasyOCR installed as the recognition library. As shown in Figure 5.26, the server code (ocr_server.py) defined

endpoints to receive images in Base64 format, process them with EasyOCR, and return detected text along with bounding box coordinates in JSON format. This allowed Unity to know not only the detected words but also their positions in the image.

Then, server was launched in a Python virtual environment. As illustrated in Figure 5.27, the Flask server was successfully started and ran on `http://127.0.0.1:5000` (local) and on the host machine's IP address for network access. But, when it deployed to the mobile phone, it needs to change to the personal ip address and the mobile phone and the server should have the same network configuration in order to do the ocr.

In Unity, captured images or files selected from the device gallery were sent to this backend using **UnityWebRequest**. The server responded with recognized words and coordinates, which Unity then used to highlight keywords in Canvas Mode by overlaying semi-transparent boxes and directional arrows on the detected areas.

This configuration enabled the AI Tutor to extend beyond textual responses and provide visual guidance, making the learning process more interactive and user-friendly.

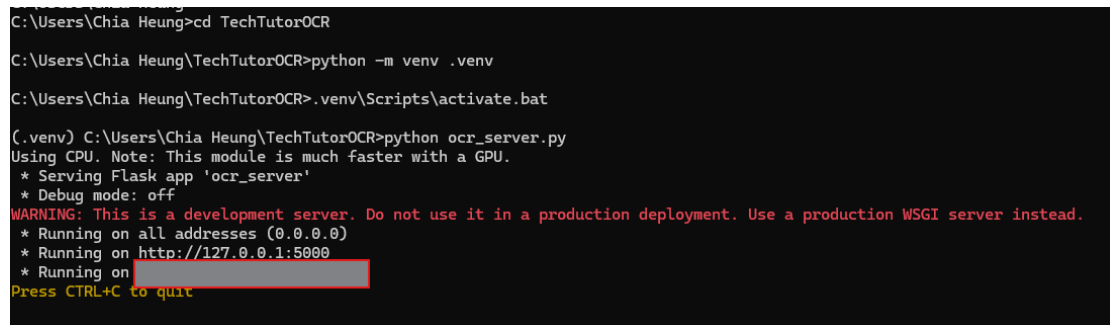


```

ocr_server.py X
C:\Users\Chia Heung > TechTutorOCR > ocr_server.py > ocr
1  from flask import Flask, request, jsonify, Response
2  from flask_cors import CORS
3  import base64, io, numpy as np
4  from PIL import Image
5  import easyocr
6
7  app = Flask(__name__)
8  CORS(app)
9  reader = easyocr.Reader(['en'], gpu=False)
10
11 @app.get("/")
12 def root():
13     return Response("OCR server alive. Use POST /ocr", mimetype="text/plain")
14
15 @app.get("/health")
16 def health():
17     return Response("ok", mimetype="text/plain")
18
19 @app.get("/ocr")
20 def ocr_get():
21     # So GET doesn't show "Method Not Allowed" in browsers
22     return jsonify({"status": "alive", "hint": "send POST with { image_base64 }"})
23
24 @app.post("/ocr")
25 def ocr():
26     data = request.get_json(force=True)
27     b64 = data["image_base64"]
28     if "," in b64:
29         b64 = b64.split(",")[-1]
30     img = Image.open(io.BytesIO(base64.b64decode(b64))).convert("RGB")
31     arr = np.array(img)
32
33     results = reader.readtext(arr, detail=1)
34     words = []
35     for box, text, conf in results:
36         xs = [p[0] for p in box]; ys = [p[1] for p in box]
37         x, y = float(min(xs)), float(min(ys))
38         w, h = float(max(xs)-x), float(max(ys)-y)
39         words.append({"text": text, "x": x, "y": y, "w": w, "h": h, "conf": float(conf)})
40
41     return jsonify({"width": img.width, "height": img.height, "words": words})
42
43 if __name__ == "__main__":
44     app.run(host="0.0.0.0", port=5000, debug=False)
45
46

```

Figure 5.26 Flask Server Code



```

C:\Users\Chia Heung>cd TechTutorOCR
C:\Users\Chia Heung\TechTutorOCR>python -m venv .venv
C:\Users\Chia Heung\TechTutorOCR>.venv\Scripts\activate.bat
(.venv) C:\Users\Chia Heung\TechTutorOCR>python ocr_server.py
Using CPU. Note: This module is much faster with a GPU.
* Serving Flask app 'ocr_server'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on
Press CTRL+C to quit

```

Figure 5.27 Flask Server Running

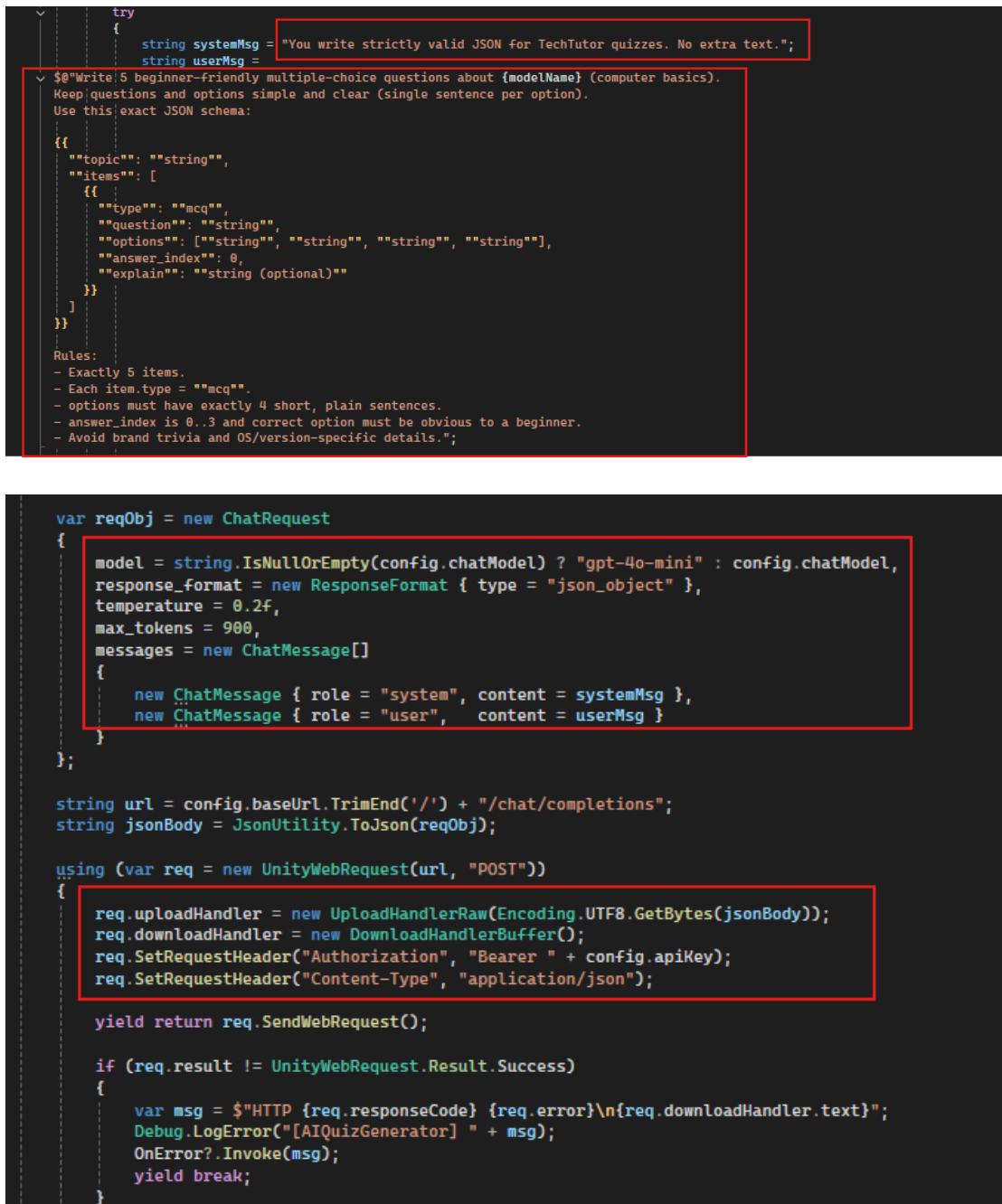
5.3.5 System Prompt Engineering for OpenAI

To ensure consistent and task-appropriate outputs from the OpenAI API, different system prompts were engineered for each module. The prompts enforce strict formats, restrict unnecessary text, and guide the model to return responses that are directly usable by TechTutor.

AR Quiz (MCQ) Generator

The AI Quiz module was implemented using the OpenAI Chat Completions API to automatically generate beginner-friendly multiple-choice questions. The system prompt was carefully engineered to instruct the model to return strictly valid JSON responses that follow a fixed schema, ensuring compatibility with Unity's JsonUtility parser.

The user prompt requested exactly **five multiple-choice questions** with **four options** each, where the correct answer must be obvious for beginners. For consistency and to minimize randomness in outputs, a low **temperature value** of **0.2** was applied in the API request, which reduced unnecessary variation across quiz generations. The generated JSON was then deserialized into a QuizPayload object, saved locally through the AIQuizLocalStore, and linked with the corresponding 3D model selected by the user. This approach ensured that each quiz session was reproducible, structured, and easily integrated into the AR environment.



```

try
{
    string systemMsg = "You write strictly valid JSON for TechTutor quizzes. No extra text.";
    string userMsg =
    $@"Write 5 beginner-friendly multiple-choice questions about {modelName} (computer basics).
    Keep questions and options simple and clear (single sentence per option).
    Use this exact JSON schema:

    {{
        ""topic"": ""string"",
        ""items"": [
            {{
                ""type"": ""mcq"",
                ""question"": ""string"",
                ""options"": [""string"", ""string"", ""string"", ""string""],
                ""answer_index"": 0,
                ""explain"": ""string (optional)""
            }}
        ]
    }}

    Rules:
    - Exactly 5 items.
    - Each item.type = ""mcq"".
    - options must have exactly 4 short, plain sentences.
    - answer_index is 0..3 and correct option must be obvious to a beginner.
    - Avoid brand trivia and OS/version-specific details.";

    var reqObj = new ChatRequest
    {
        model = string.IsNullOrEmpty(config.chatModel) ? "gpt-4o-mini" : config.chatModel,
        response_format = new ResponseFormat { type = "json_object" },
        temperature = 0.2f,
        max_tokens = 900,
        messages = new ChatMessage[]
        {
            new ChatMessage { role = "system", content = systemMsg },
            new ChatMessage { role = "user", content = userMsg }
        }
    };

    string url = config.baseUrl.TrimEnd('/') + "/chat/completions";
    string jsonBody = JsonUtility.ToJson(reqObj);

    using (var req = new UnityWebRequest(url, "POST"))
    {
        req.uploadHandler = new UploadHandlerRaw(Encoding.UTF8.GetBytes(jsonBody));
        req.downloadHandler = new DownloadHandlerBuffer();
        req.SetRequestHeader("Authorization", "Bearer " + config.apiKey);
        req.SetRequestHeader("Content-Type", "application/json");

        yield return req.SendWebRequest();

        if (req.result != UnityWebRequest.Result.Success)
        {
            var msg = $"HTTP {req.responseCode} {req.error}\n{req.downloadHandler.text}";
            Debug.LogError("[AIQuizGenerator] " + msg);
            OnError?.Invoke(msg);
            yield break;
        }
    }
}

```

Figure 5.28 AR Quiz Configuration Code Snippet

AI Tutor (Text/Image Step-by-Step Guidance)

The AI Tutor module was configured to provide step-by-step guidance in plain text, with optional **keyword highlighting** when a screenshot was attached. A carefully engineered system prompt instructed the model to always respond in a strict format, which is **numbered steps** with a **short action**, a **brief explanation**, and a ‘**Keywords**’, which is a line containing **one to three exact UI tokens** from the **image** (e.g., menu

labels or button names). If no tokens were available, the model was required to return Keywords: (none). To ensure consistency, additional rules prohibited greetings, summaries, markdown formatting, or code fences.

When a screenshot was included, the request was sent to **gpt-4o** with **both text and Base64-encoded image data**. For **text-only queries**, the lightweight **gpt-4o-mini** was used. In both cases, the API call was configured with **temperature = 0.2** to reduce variability and **max_tokens = 900** to prevent truncation across 4–6 steps. The request was built in JSON format and submitted via **UnityWebRequest**, with the API key passed securely in the authorization header.

The returned response was parsed, sanitized, and displayed in the chat panel. The extracted steps were also transformed into an internal guide object, which supported ‘**Canvas Mode**’ by overlaying **highlights** on **detected keywords** using **EasyOCR**. All interactions were saved to the chat history for later review.

```
string systemPrompt =
    "You are TechTutor, an AI assistant that explains computer tasks step-by-step.\n\n" +
    "Always respond in this STRICT plain-text format ONLY (no markdown, no code fences):\n\n" +
    "Step 1: <short action>\n\n" +
    "  <very brief explanation>\n\n" +
    "Keywords: <1-3 exact UI tokens from the image, comma-separated>\n\n" +
    "Step 2: <short action>\n\n" +
    "  <very brief explanation>\n\n" +
    "Keywords: <1-3 exact UI tokens from the image, comma-separated>\n\n" +
    "- Include 4-6 steps if needed\n\n" +
    "- Use simple words\n\n" +
    "- Keywords MUST be short, verbatim labels visible in the IMAGE (menu text, button labels, tab names, shortcuts like Ctrl+C)\n\n" +
    "- Do NOT invent tokens; if none are visible for a step, write: Keywords: (none)\n\n" +
    "- Do NOT add greetings, summaries, tips, or anything else\n\n" +
    "- Do NOT use ### or any markdown\n\n" +
    "- Do NOT wrap your answer in JSON or code fences\n\n";
```

```
string model = includeImage ? "gpt-4o" : "gpt-4o-mini";

if (includeImage)
{
    base64Image = EncodeImageToBase64(selectedImage);
    json = @"{
        ""model"": "" + model + @""",
        ""temperature"": 0.2,
        ""messages"": [
            { ""role"": ""system"", ""content"": "" + EscapeJson(systemPrompt) + @"" },
            {
                ""role"": ""user"",
                ""content"": [
                    { ""type"": ""text"", ""text"": "" + EscapeJson(userMessage) + @"" },
                    { ""type"": ""image_url"", ""image_url"": { ""url"": ""data:image/jpeg;base64," + base64Image + @"" } }
                ]
            }
        ],
        ""max_tokens"": 900
    }";
}
else
{
    json = @"{
        ""model"": "" + model + @""",
        ""temperature"": 0.2,
        ""messages"": [
            { ""role"": ""system"", ""content"": "" + EscapeJson(systemPrompt) + @"" },
            { ""role"": ""user"", ""content"": "" + EscapeJson(userMessage) + @"" }
        ],
        ""max_tokens"": 900
    }";
}
```

Uploaded with Image

Uploaded with Text Only

```

UnityWebRequest request = new UnityWebRequest(apiUrl, "POST");
byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(json);
request.uploadHandler = new UploadHandlerRaw(bodyRaw);
request.downloadHandler = new DownloadHandlerBuffer();

request.SetRequestHeader("Content-Type", "application/json");
request.SetRequestHeader("Authorization", "Bearer " + openAIKey);

// Show the user message immediately
responseText.text += $"\\n\\nYou: {userMessage}\\n\\nTechTutor:\\n";
Canvas.ForceUpdateCanvases();
chatScrollRect.verticalNormalizedPosition = 0f;

yield return request.SendWebRequest();

if (request.result == UnityWebRequest.Result.Success)
{
    string jsonResponse = request.downloadHandler.text;
    string reply = ExtractReply(jsonResponse);

    // / sanitize before displaying
    reply = StripDisclaimers(reply);
    reply = SanitizeForTMP(reply);

    lastBotReply = reply;

    // still build guide so AR can run
    lastGuide = BuildGuideFromPlainSteps(reply);

    string displayReply = StripKeywordLines(reply);
    StartCoroutine(TypeText(responseText.text, displayReply));
}

```

Figure 5.29 AI Tutor Configuration Code Snippet

AI Identify (Computer Component Description)

The AI Identify module was designed to analyze images of computer components and provide concise educational descriptions. When the user captured an image through the AR interface, the application first performed a “shutter” effect and then obtained a full-screen capture from the AR camera. The captured texture was downscaled and compressed into JPEG format to optimize network transfer, while a preview copy of the image was retained for display in the UI.

The processed image was then encoded into Base64 and sent to the OpenAI API with a specialized prompt. The system prompt instructed the model to identify the component (e.g., mouse, keyboard, monitor, speaker, laptop) and to respond in a strict format:

- This is a _____.
- [TITLE]: _____
- [INFO]: _____

The [TITLE] had to match the **component name**, while the [INFO] section was limited to one or two short paragraphs written in a beginner-friendly style. The prompt explicitly forbade describing the photo itself (e.g., “This image shows...”), ensuring that the output remained educational rather than observational.

The request was then sent using `UnityWebRequest` with the Base64-encoded image attached, as shown in Figure 5.31. Upon receiving a successful response, the text was parsed to extract the summary, title, and info sections. These outputs were displayed in the info panel, converted into speech using TTS, and saved together with the preview image into the history log for later review.

This configuration enabled the AI Identify module to transform simple camera captures into interactive learning content, bridging AR recognition with AI-driven explanations.

```
// 2) Downscale & compare (also generate preview copy)
Texture2D resized = Downscale(full, maxUploadWidth);
byte[] jpegBytes = resized.EncodeToJPG(jpegQuality);

// keep a preview texture to show in info panel (use the resized one)
if (lastPreviewTexture) Destroy(lastPreviewTexture);
lastPreviewTexture = resized; // keep for UI
Destroy(full); // free big texture

// 3) Build request
string base64Image = Convert.ToBase64String(jpegBytes);
string prompt =
    "You are helping to build an AR educational tutorial titled 'Identify Computer Components'.\\n" +
    "Your job is to describe what computer component is shown in the image.\\n" +
    "Examples of components include: mouse, keyboard, laptop, monitor, speaker, etc.\\n" +
    "Always reply in this exact format:\\n" +
    "This is a _____.\\n" +
    "[TITLE]: _____.\\n" +
    "[INFO]: _____.\\n" +
    "Replace the blanks with the name of the component.\\n" +
    "Make sure the INFO is written in a beginner-friendly way using 1 to 2 short paragraphs.\\n" +
    "Do not describe the image (e.g., 'This image shows...'). Instead, provide educational content about the component.\\n" +
    "The TITLE must be the same as the component name.";

string json =
    "{\\\"model\\\": \\\"gpt-4o\\\", \" +
    \"\\\"messages\\\": [{\" +
    \"\\\"role\\\": \\\"user\\\", \" +
    \"\\\"content\\\": [\" +
    \"\\\"type\\\": \\\"text\\\", \\\"text\\\": \\\"\" + prompt + "\\\"\", \" +
    \"\\\"type\\\": \\\"image_url\\\", \\\"image_url\\\": {\\\"url\\\": \\\"data:image/jpeg;base64, \" + base64Image + "\\\"}} \" +
    \"]\" +
    "} ]\"}";
```

Figure 5.30 AI Identify System Prompt Section

```
// 4) Send
UnityWebRequest request = new UnityWebRequest("https://api.openai.com/v1/chat/completions", "POST");
byte[] bodyRaw = System.Text.Encoding.UTF8.GetBytes(json);
request.uploadHandler = new UploadHandlerRaw(bodyRaw);
request.downloadHandler = new DownloadHandlerBuffer();
request.SetRequestHeader("Content-Type", "application/json");
request.SetRequestHeader("Authorization", "Bearer " + openAIApiKey);
```

```
// Extract parts for history & info
ExtractTitleAndInfo(reply, out string title, out string content);

// Save preview image to disk so it persists in history
string savedPath = SavePreviewTextureToDisk(lastPreviewTexture);
var entry = AddHistory(summary, title, content, savedPath);
// Do NOT persist here (it plays). Let SpeakSummaryOnce() create+save+play.
SpeakSummaryOnce();
```

Figure 5.31 AI Identify Configuration Code Snippet

5.4 System Operation

5.4.1 Main Menu Scene

Figure 5.32 shows the main menu screen displayed when the user first logs into the system. It consists of the TechTutor logo and three buttons for users to select, which are AR Learn, AR Guide, and AI Tutor.

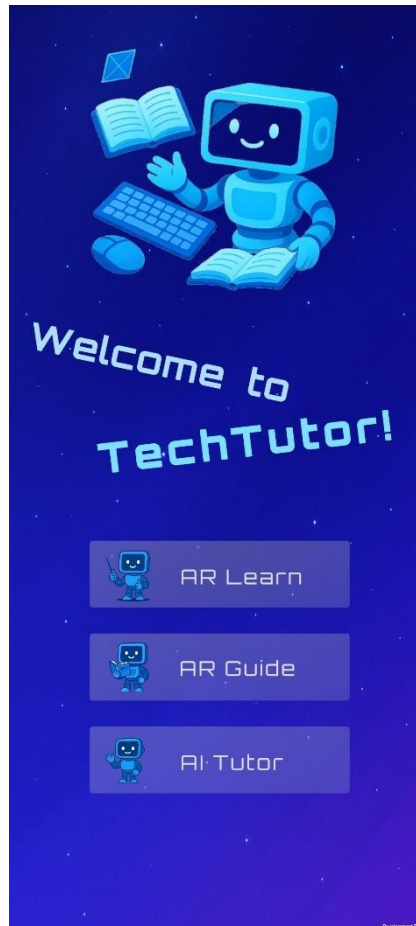


Figure 5.32 Main Menu Screen

5.4.2 AR Learn

Figure 5.33 shows the main screen of the AR Learn module. It consists of three buttons which are **AI Identify**, **AR Modules**, and **AR Quiz**.



Figure 5.33 AR Learn Module Main Screen

AI Identify

Figure 5.34 shows the AI Identify UI screen. The “Back” button (←) allows the user to return to the AR Learn module main screen. The “Capture” button enables users to point at any computer component and perform the real-time capturing function. The navigation section, represented by a robot character, displays instructions to guide users, and also includes a “More Info” button that provides additional details about the identified computer component. Lastly, the “History List” button allows users to review previously captured images along with the corresponding information.

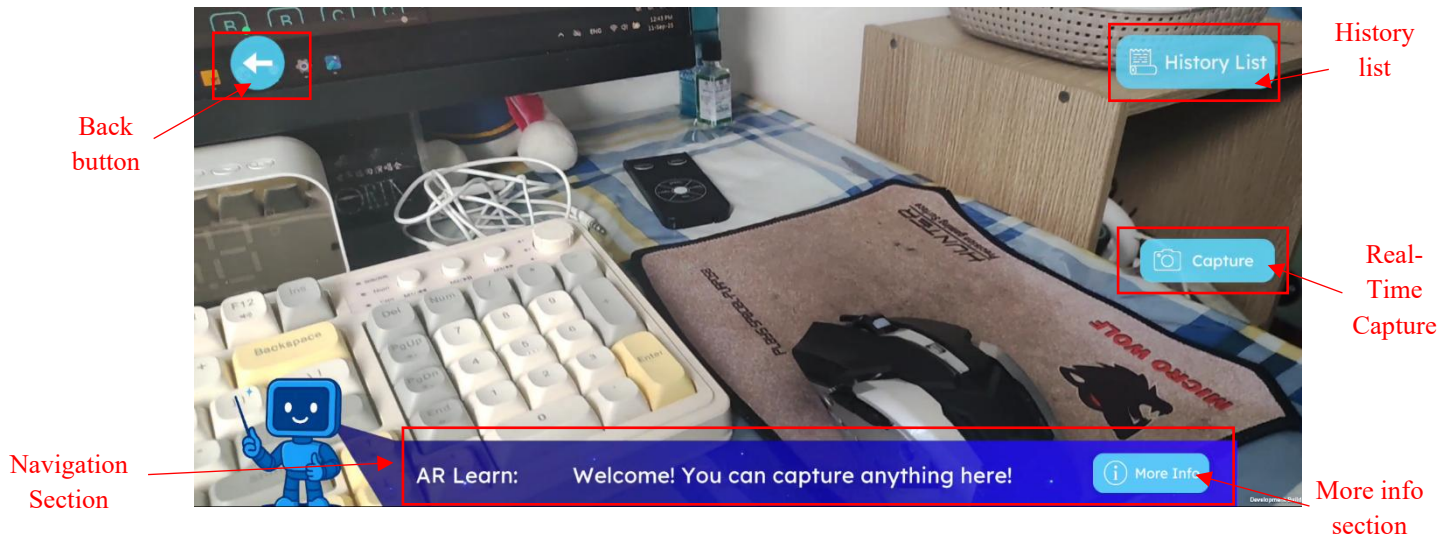


Figure 5.34 AI Identify Main Screen

Figure 5.35 shows the process when a user points at a computer component, in this case, a mouse, and clicks the “Capture” button. An information message, “Analyzing... Please wait...”, is displayed while the system processes the image. During this stage, the captured image is sent to OpenAI for object detection.

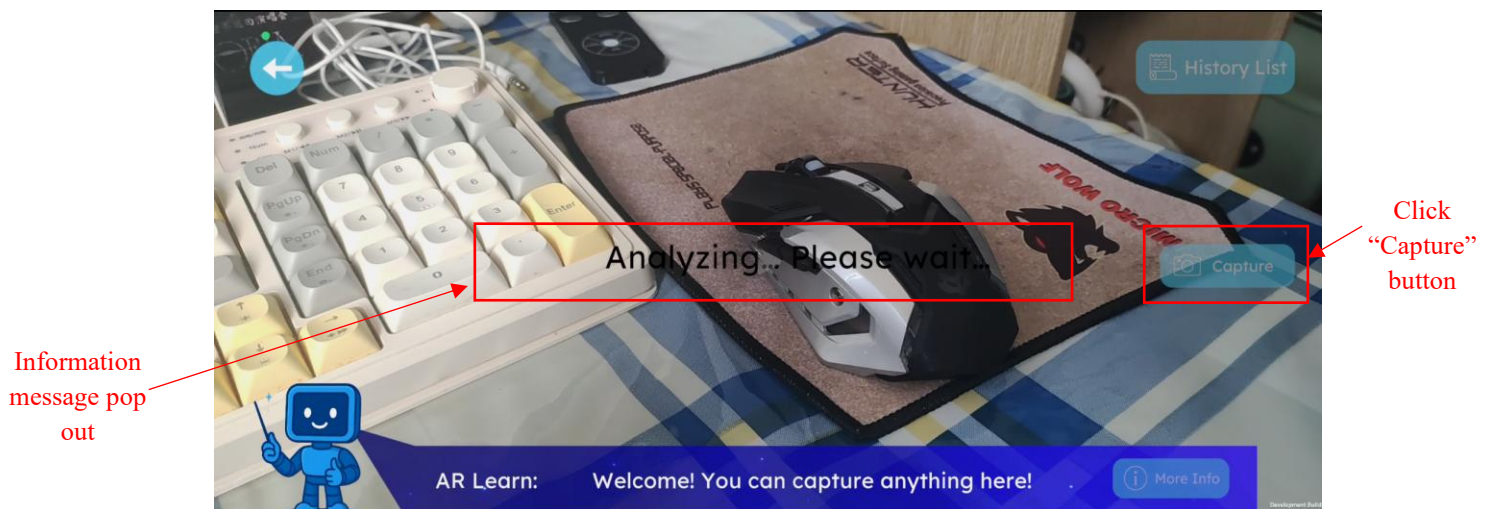


Figure 5.35 Capture Scene

Figure 5.36 shows the result provided by OpenAI. After the analysis, the content in the navigation area changes from “Welcome! You can capture anything here” to a message stating “This is a ____.” In this case, since the identified computer component is a mouse, the system displays the message “This is a mouse.” At the

same time, a voice response is also generated, informing the user with the spoken message “This is a mouse.”



Figure 5.36 OpenAI Response

Figure 5.37 shows the information panel that appears when the user clicks the More Info button. More detailed information about the computer component is displayed, accompanied by speech generated through OpenAI’s Text-to-Speech (TTS). The panel also includes a Voice Out button, which allows the user to replay the speech if they need to listen to the explanation again.

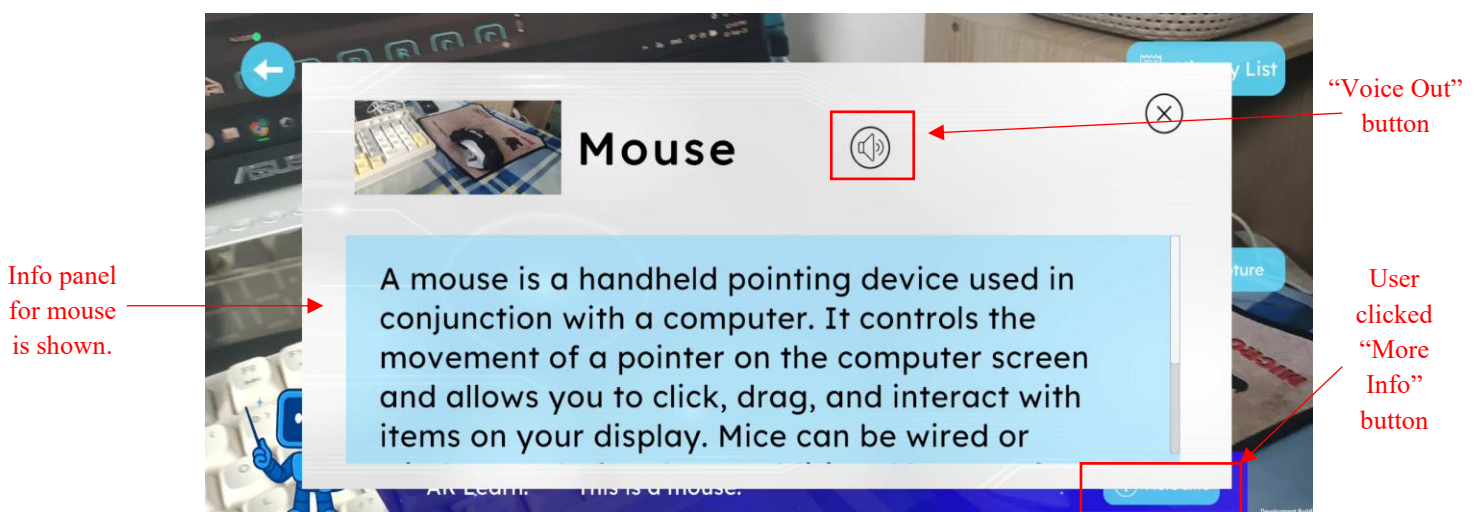


Figure 5.37 Mouse Info Panel

From Figures 5.38 and 5.39, after the user finishes reading and understanding the information about the mouse, they can continue to capture other computer components such as a monitor or laptop. In this case, a keyboard is captured, as shown in Figure 5.38. Subsequently, the navigation content changes from “This is a mouse” to “This is a keyboard,” accompanied by the corresponding voice output, as shown in Figure 5.39.

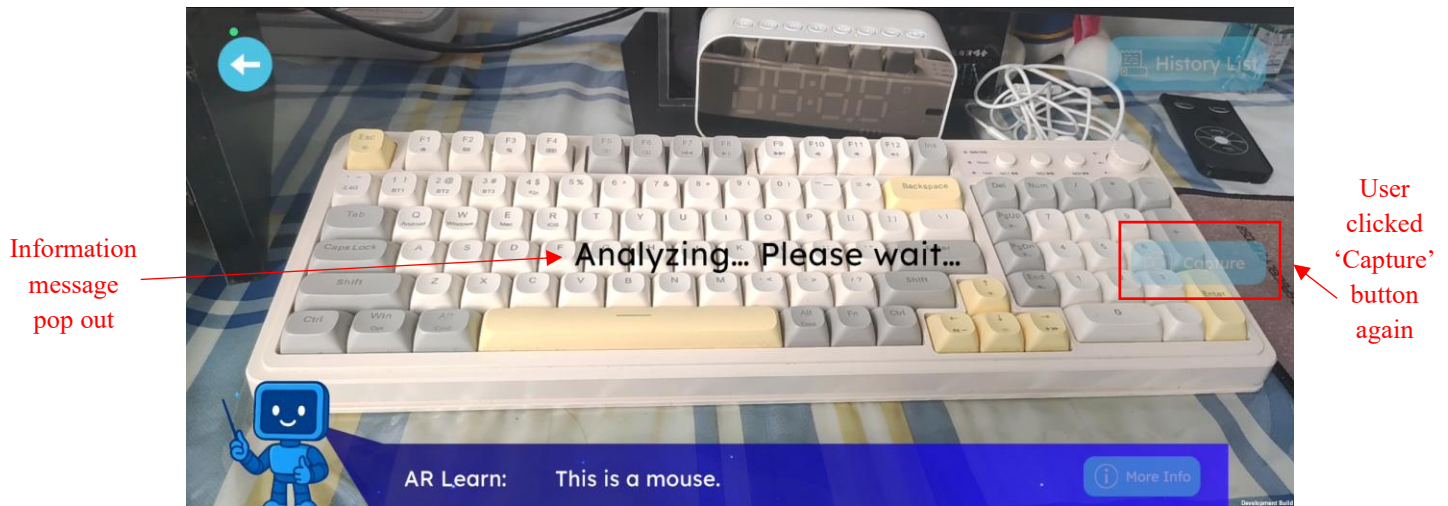


Figure 5.38 Capture Scene for Keyboard



Figure 5.39 Updated Navigation Content

Figure 5.40 illustrates the action when the user clicks the “History List” button. The system then displays a list containing all previously captured images along with their descriptions. As shown in Figure 5.41, each entry in the list includes a two-line description of the computer component, the date and time of capture, and a “View” button located at the end of the panel.



Figure 5.40 Accessing the History List

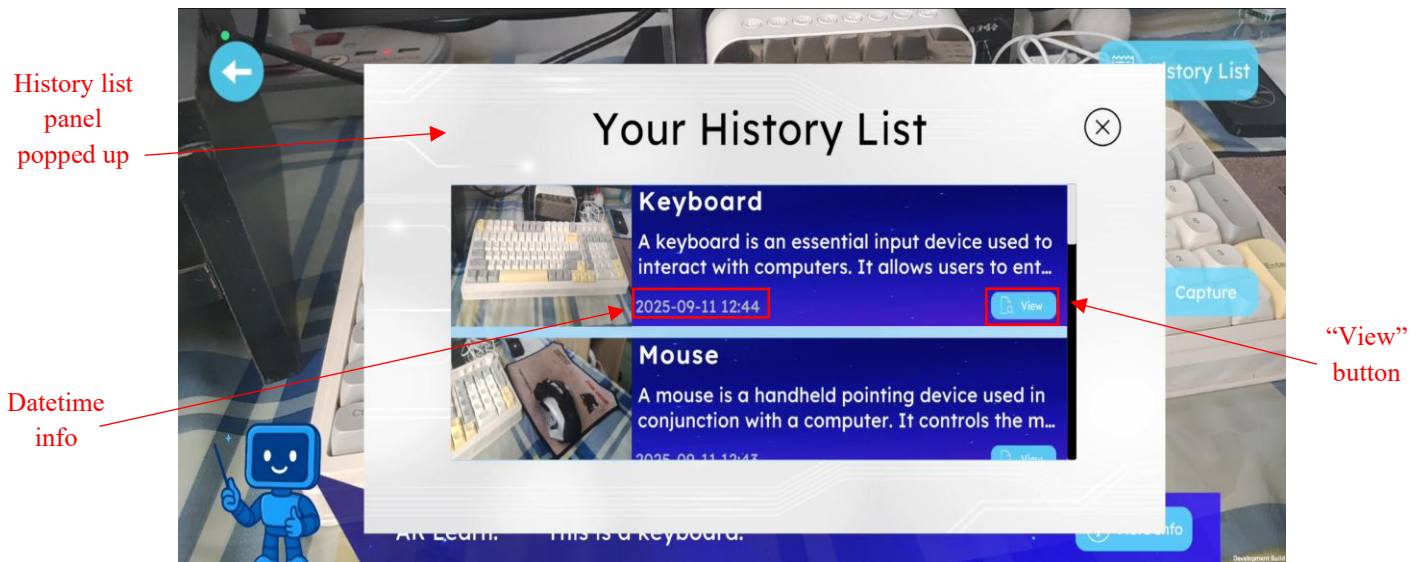


Figure 5.41 History List Interface

After the user clicks the “View” button, the corresponding information panel is displayed again, and the voice speech is played automatically. This allows the user to review the same information for that particular computer component, as shown in Figure 5.42.

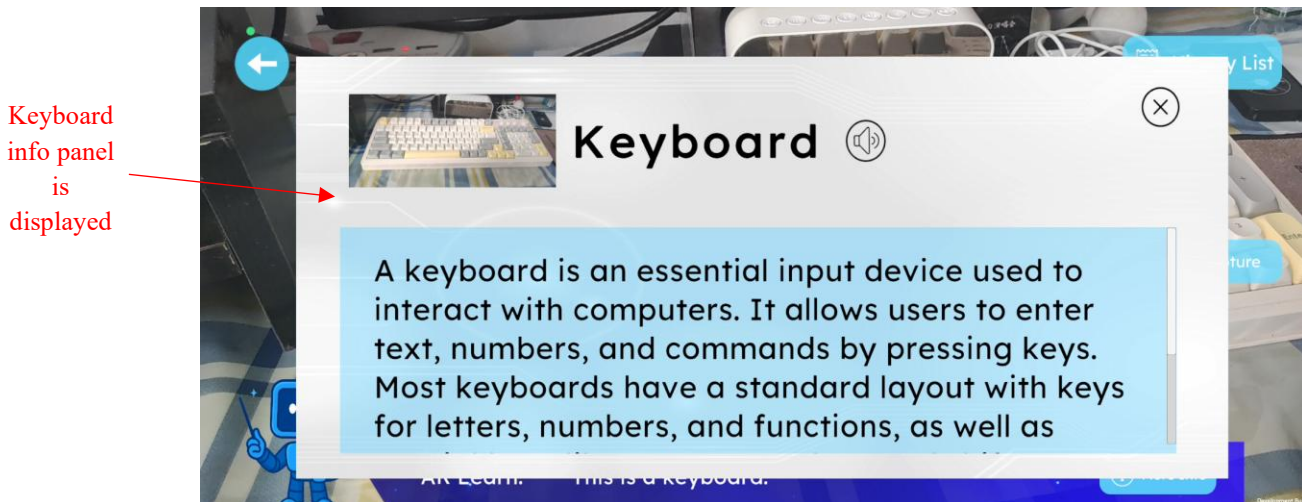


Figure 5.42 Keyboard Information Panel

AR Models

Figure 5.43 shows the main screen of the AR Models module. A “Back” button (←) is in the top-left corner, while a “Model Selection” button is provided in the top-right corner, allowing users to choose a model. In the bottom-left corner, a “Text-to-Speech” (TTS) icon enables users to trigger the tutorial without selecting a model manually. In the bottom-right corner, there are five control buttons. The first side is the movement control, which allows users to move the model left, right, up, and down. A “Reset” button is also available to return the model to its original position in the right side.

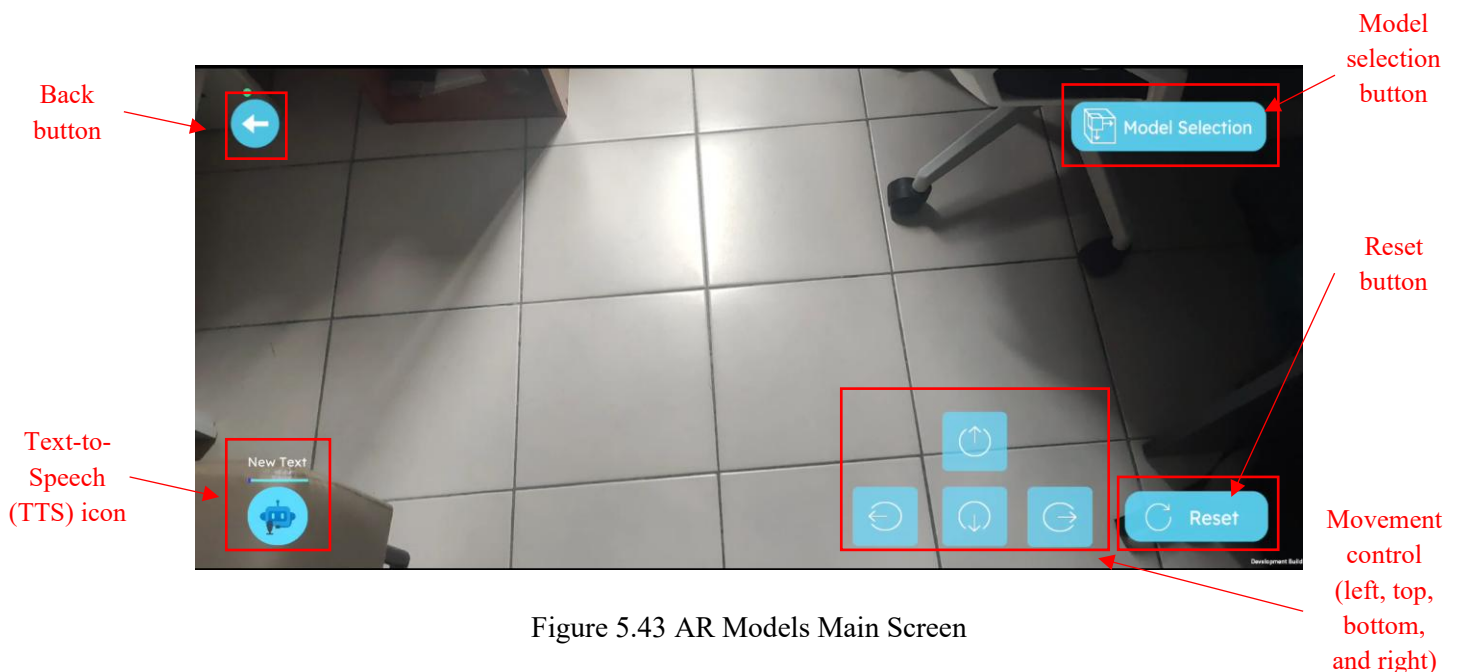


Figure 5.43 AR Models Main Screen

Before overlaying a model, the user needs to perform plane detection by scanning a horizontal surface. The surface should have visible texture to allow AR Foundation to capture it more effectively. Figure 4.44 shows the detected plane.



Figure 5.44 Plane Detection

After the user completes plane detection, they may click on the detected plane to lock it. Once the plane is locked, plane detection will no longer proceed. As shown in Figure 5.45, the model selection interface will then appear, consisting of five main models for users to choose from. They are keyboard, monitor, speaker, mouse, and laptop.

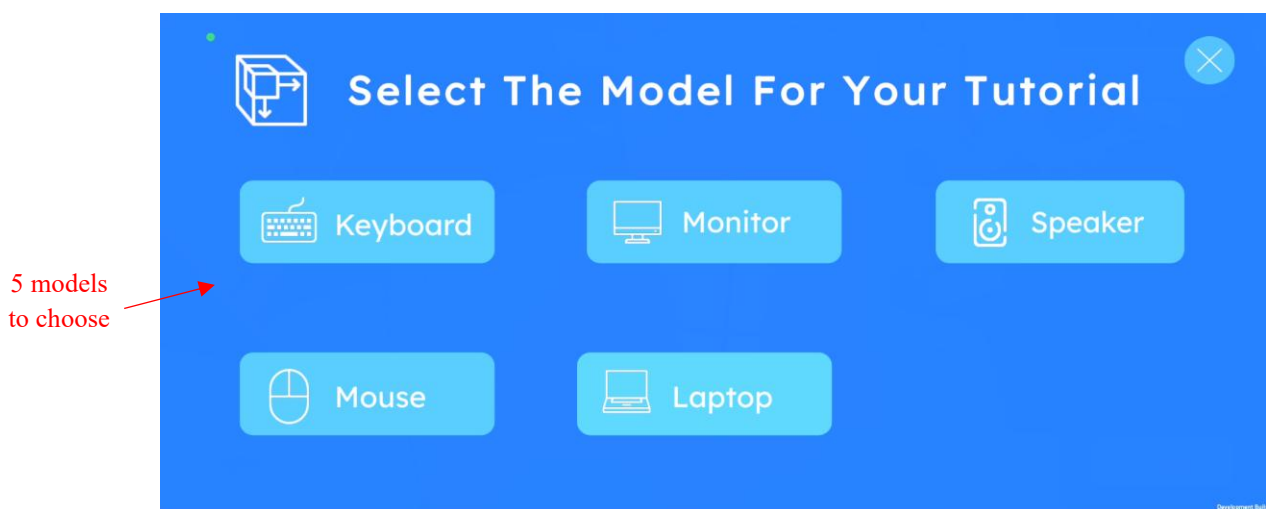


Figure 5.45 Model Selection Interface

Figure 5.46 shows that when the user selects the keyboard model in Figure 5.45, a preview of the keyboard is overlaid on the plane. The user may reposition it within the plane or to a preferred location, and then click the “Confirm Model” button to place the model in that position.

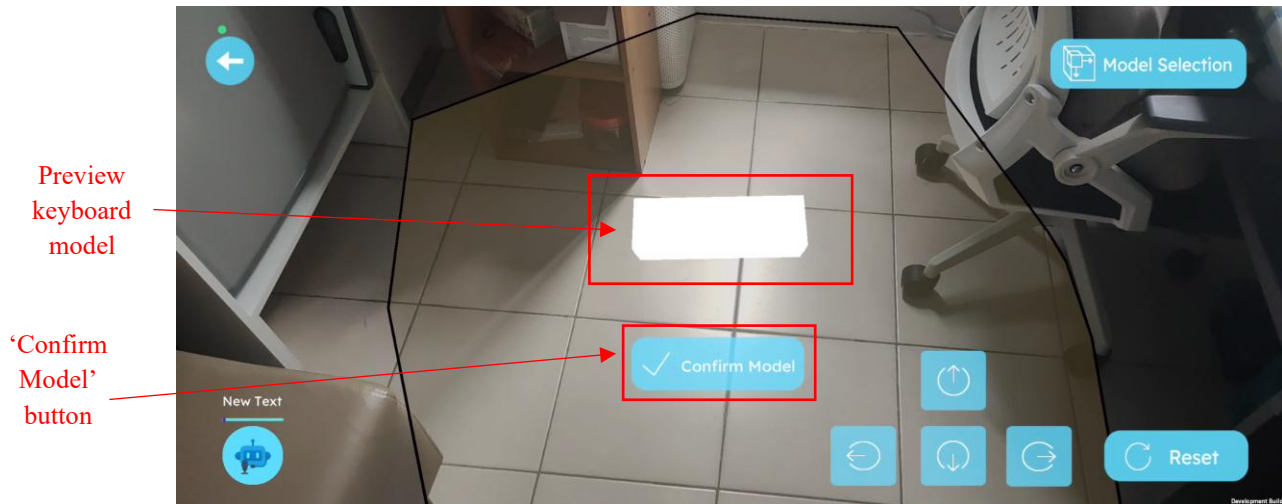


Figure 5.46 Keyboard Model Selected

Figure 5.47 shows that before confirming the model, the user may use two fingers to enlarge it, making the model more visible and adjusting it to a suitable size.

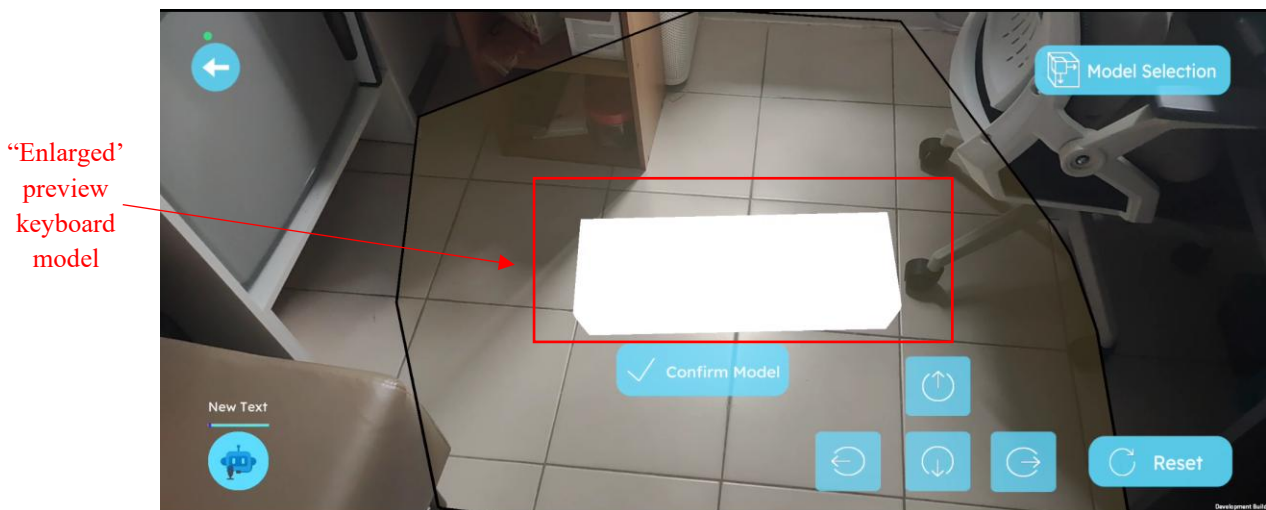


Figure 5.47 Enlarged Preview of Keyboard Model

Figure 5.48 shows that after the user clicks the “Confirm Model” button, the keyboard model is placed according to the position and size previously adjusted by the user.

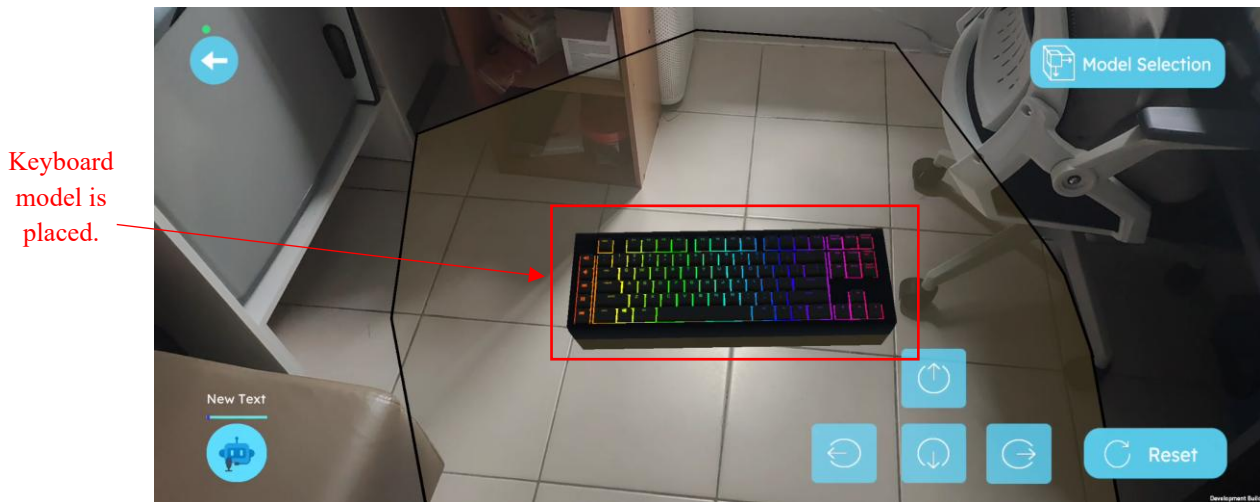


Figure 5.48 Keyboard Model Placed

Figure 5.49 shows that after the user clicks the model, then the information panel is displayed, and the corresponding speech is played simultaneously. The panel also includes an “Explore More” button, which the user may click to proceed to the next step and view more detailed information about the keyboard model.

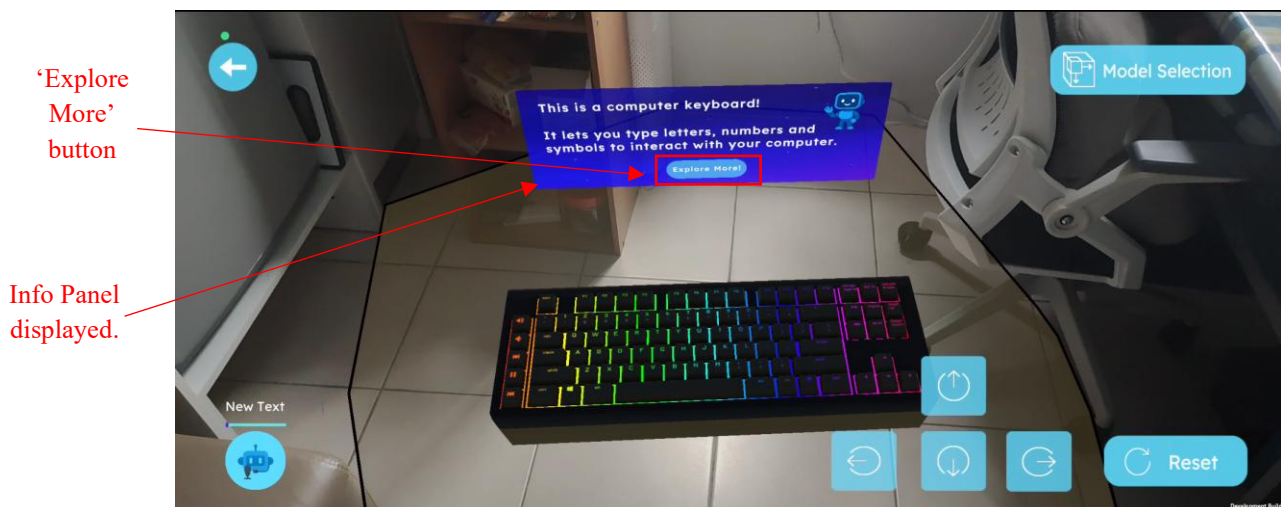


Figure 5.49 Information Panel Displayed

Figure 5.50 shows that after the user clicks the “Explore More” button, labelled text is displayed to highlight important keys on the keyboard, such as the ‘Windows’ Key, ‘Arrow’ Keys, ‘Delete’ Key, ‘Control’ Key, ‘Caps Lock’ Key, and the ‘Volume Up’ and ‘Volume Down’ Keys.

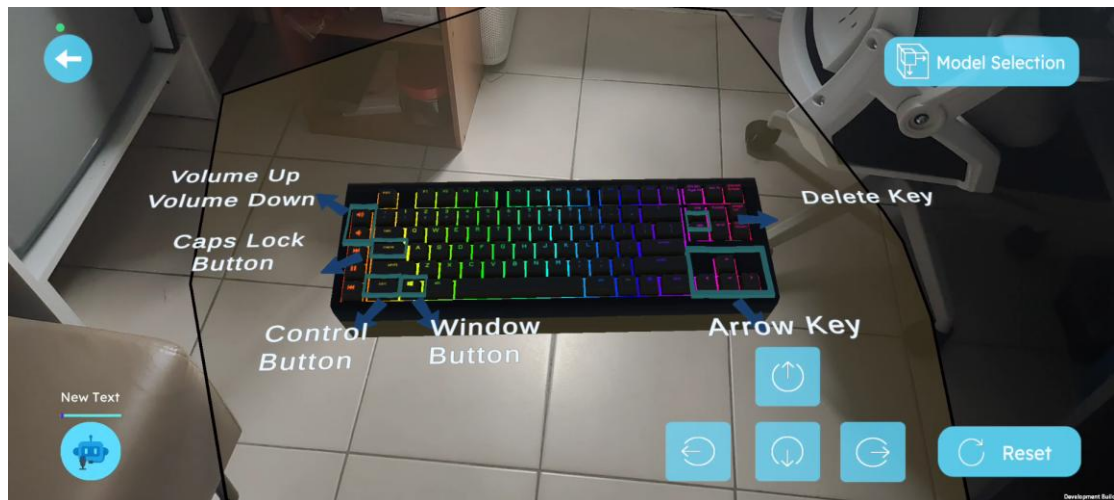


Figure 5.50 Key Labels Overlay

Figure 5.51 shows that when the user clicks the ‘Left’ button in the movement control section, the keyboard model shifts to the left.



Figure 5.51 Movement of Keyboard Model

Figure 5.52 shows that when the user clicks the “Reset” button, the keyboard model returns to its initial position and scale.

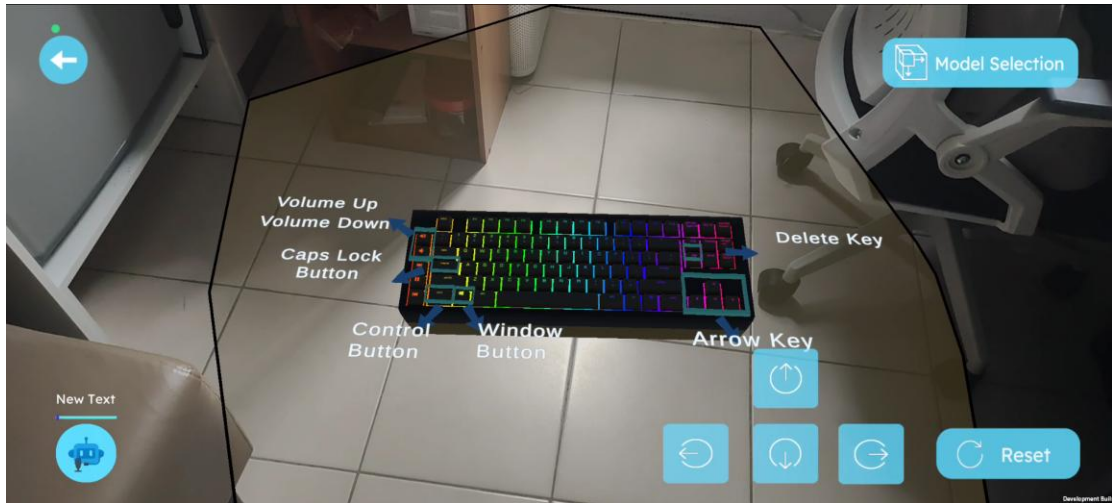


Figure 5.52 Position Reset

Figure 5.53 shows the information panel for the ‘Arrow’ Keys. When the user clicks the ‘Arrow’ key in the keyboard model, then this panel is displayed, providing detailed information about the keys, including their position, functions, and common usage.

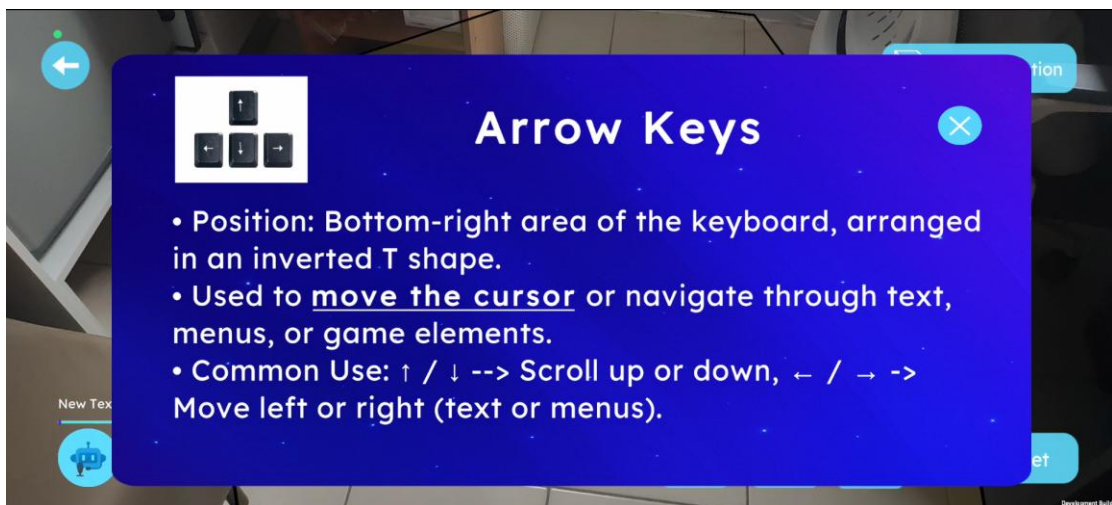


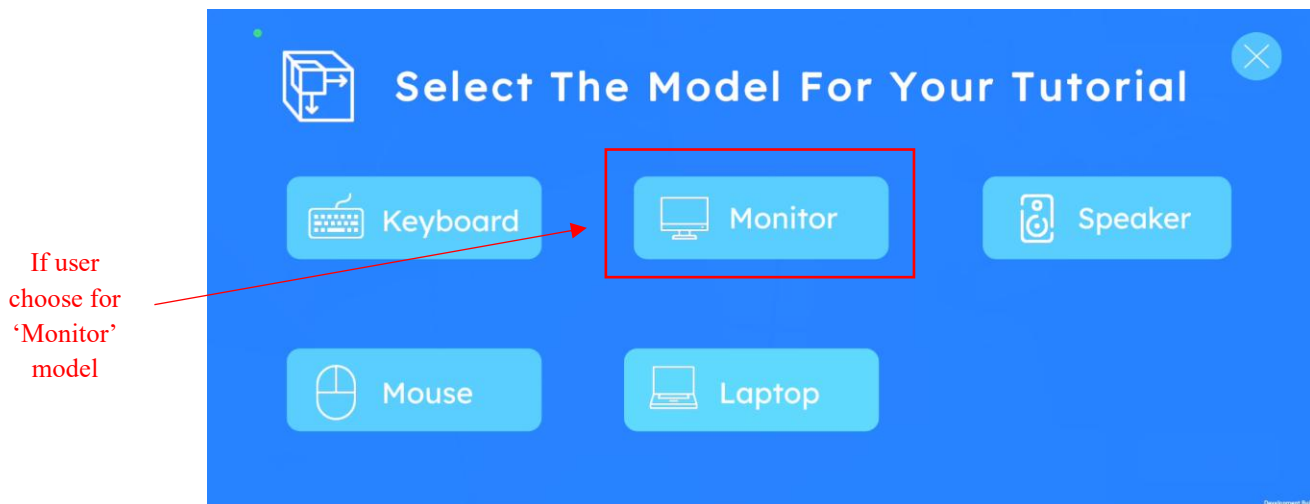
Figure 5.53 Arrow Keys Information Panel

Figure 5.54 shows the information panel for the remaining keys, including the ‘Volume’ Keys, ‘Caps Lock’ Key, ‘Control’ Keys, ‘Windows’ Key, and ‘Delete’ Key.



Figure 5.54 Information Panel for Other Keys

Figure 5.55 shows that when the user clicks the “Model Selection” button again and chooses the monitor model, the keyboard model is removed, and a preview of the monitor appears at the starting point.



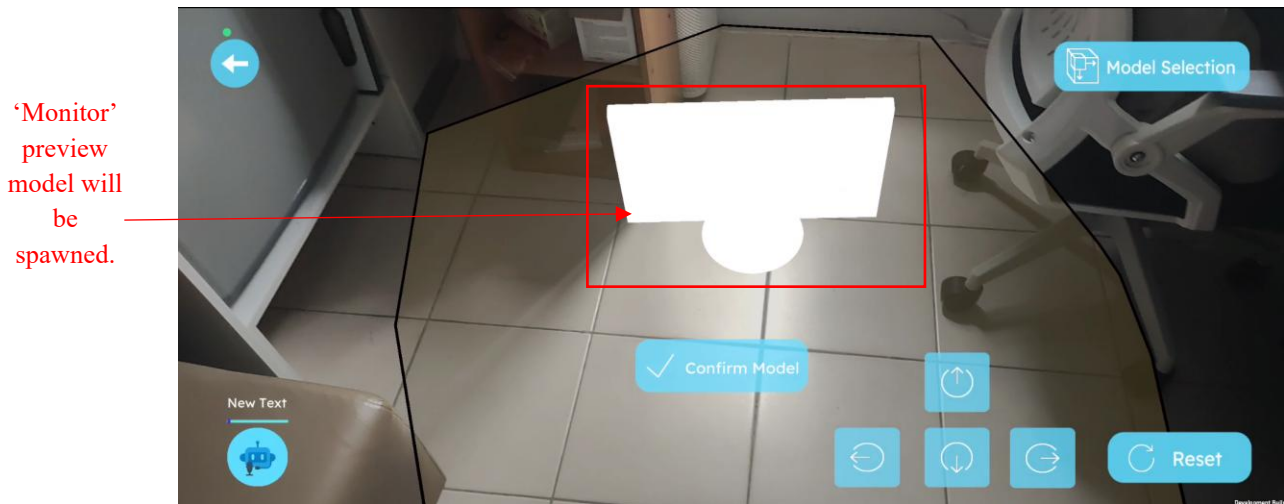


Figure 5.55 Selection Another Model

Figure 5.56 shows that when the user clicks the monitor model, the information panel is displayed. When the user clicks 'Explore More', labels appear for key parts of the monitor, including the Power Plug, HDMI Port, HDMI Label, Brand Label, and Power Button. Since the Power Plug and HDMI Port are located at the back of the monitor, the user can rotate the model with a finger gesture to view the labels clearly. Movement to the left and right is also supported.

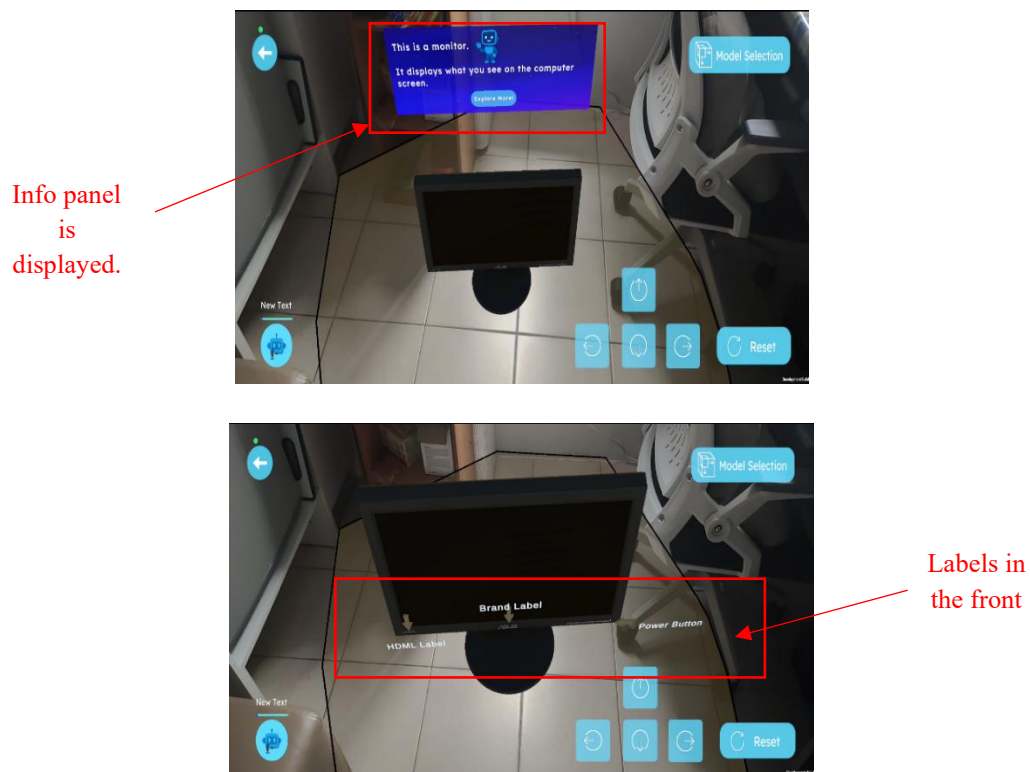


Figure 5.56 Information Panel and Labels for Monitor Model



Figure 5.56 Information Panel and Labels for Monitor Model (Cont.)

Figure 5.57 shows the information panel for all components of the monitor model, including the brand label, HDML label, Power Button, HDMI Port, and Power plug.



Figure 5.57 Information Panel for All Components (Monitor)

Figure 5.58 shows that when the user clicks the mouse model, the information panel is displayed. When the user clicks “Explore More” button, labels appear for key parts of the mouse, including the Scroll Wheel, Left Click, and Right Click.

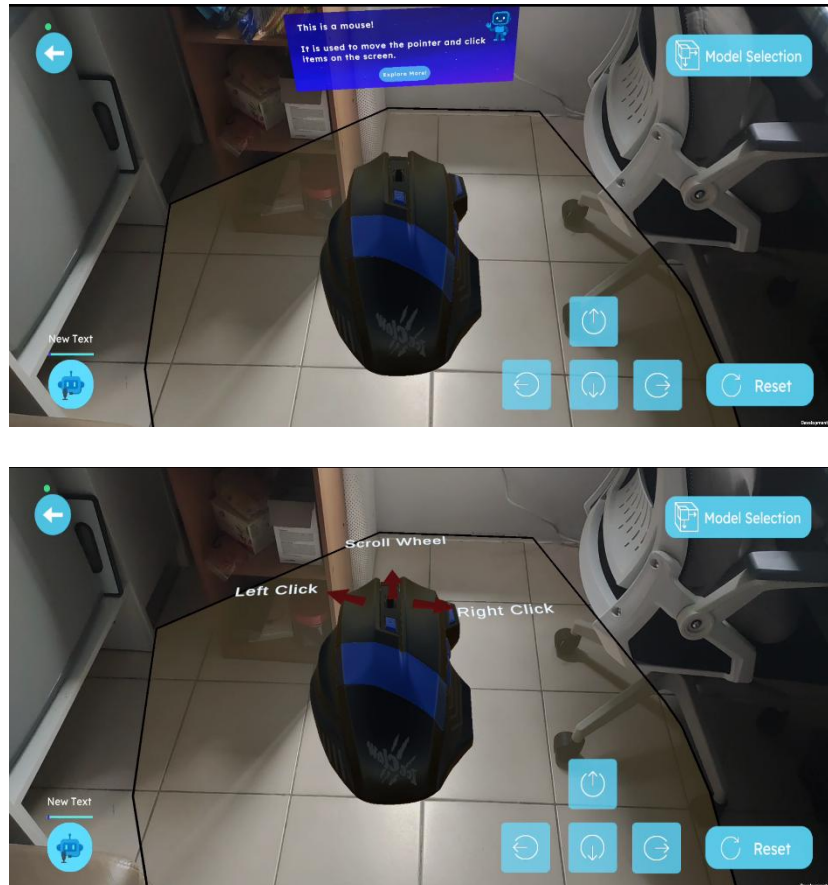


Figure 5.58 Information Panel and Labels for Mouse Model

Figure 5.59 shows the information panel for all components of the mouse model, including the Left Click, Right Click, and Scroll Wheel.

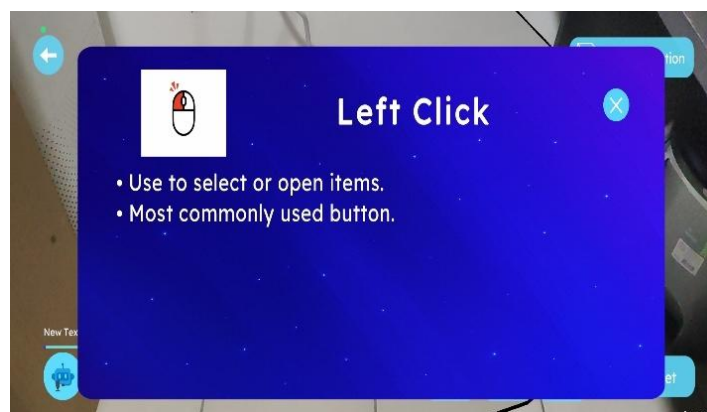


Figure 5.59 Information Panel for All Components (Mouse)

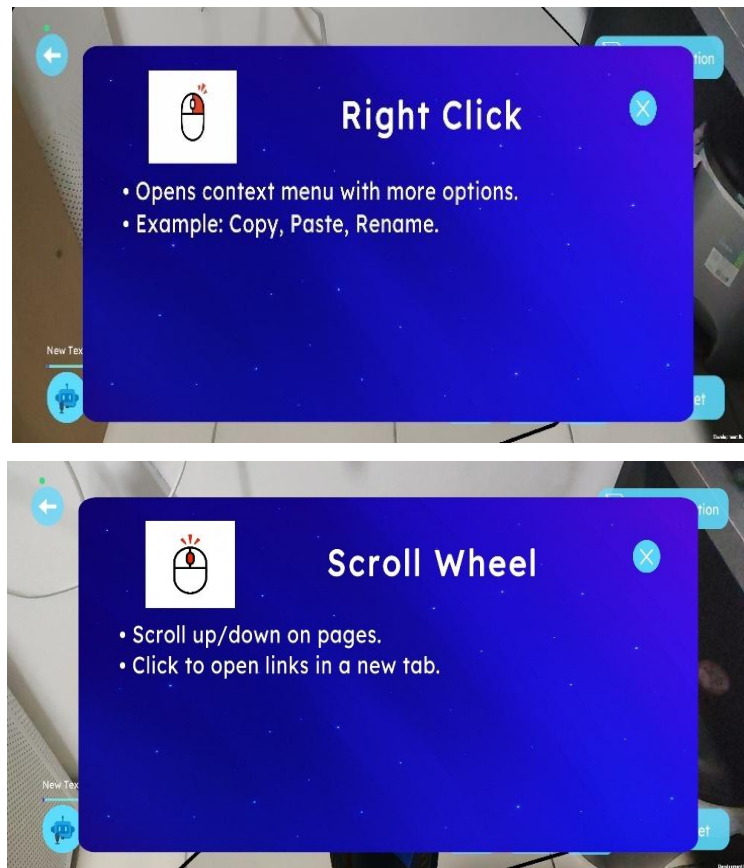


Figure 5.59 Information Panel for All Components (Mouse) (Cont.)

Figure 5.60 shows that when the user clicks the speaker model, the information panel is displayed. No additional labels appear for the speaker.

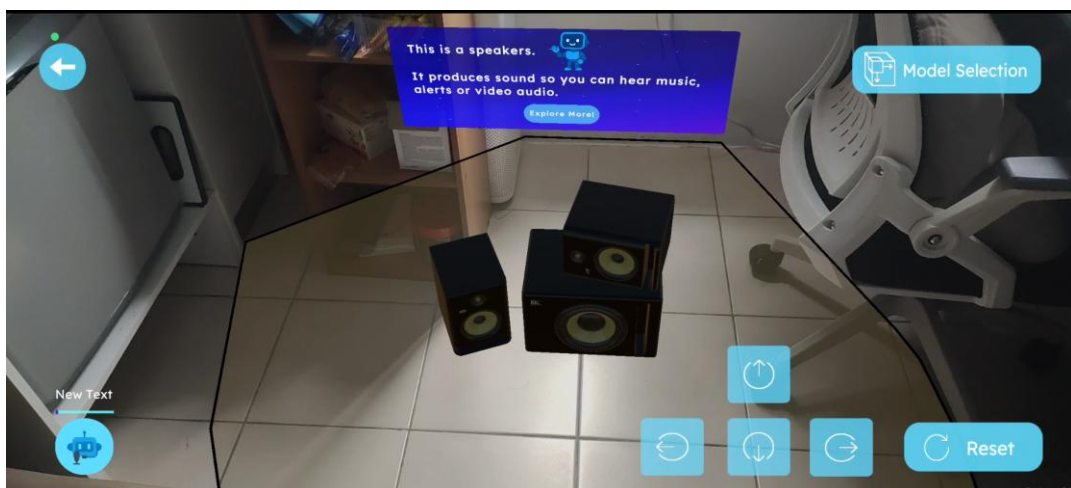


Figure 5.60 Information Panel for Speaker Model

Figure 5.61 shows the Speech-to-Text (STT) function in action. When the user clicks the button, the instruction changes from “New Text” to “Listening,” indicating that the system has started listening to the user’s speech. For example, if the user says, “laptop tutorial”, the action to switch the model is triggered. The existing model is removed, and the laptop preview model is spawned, as shown in Figure 5.61.

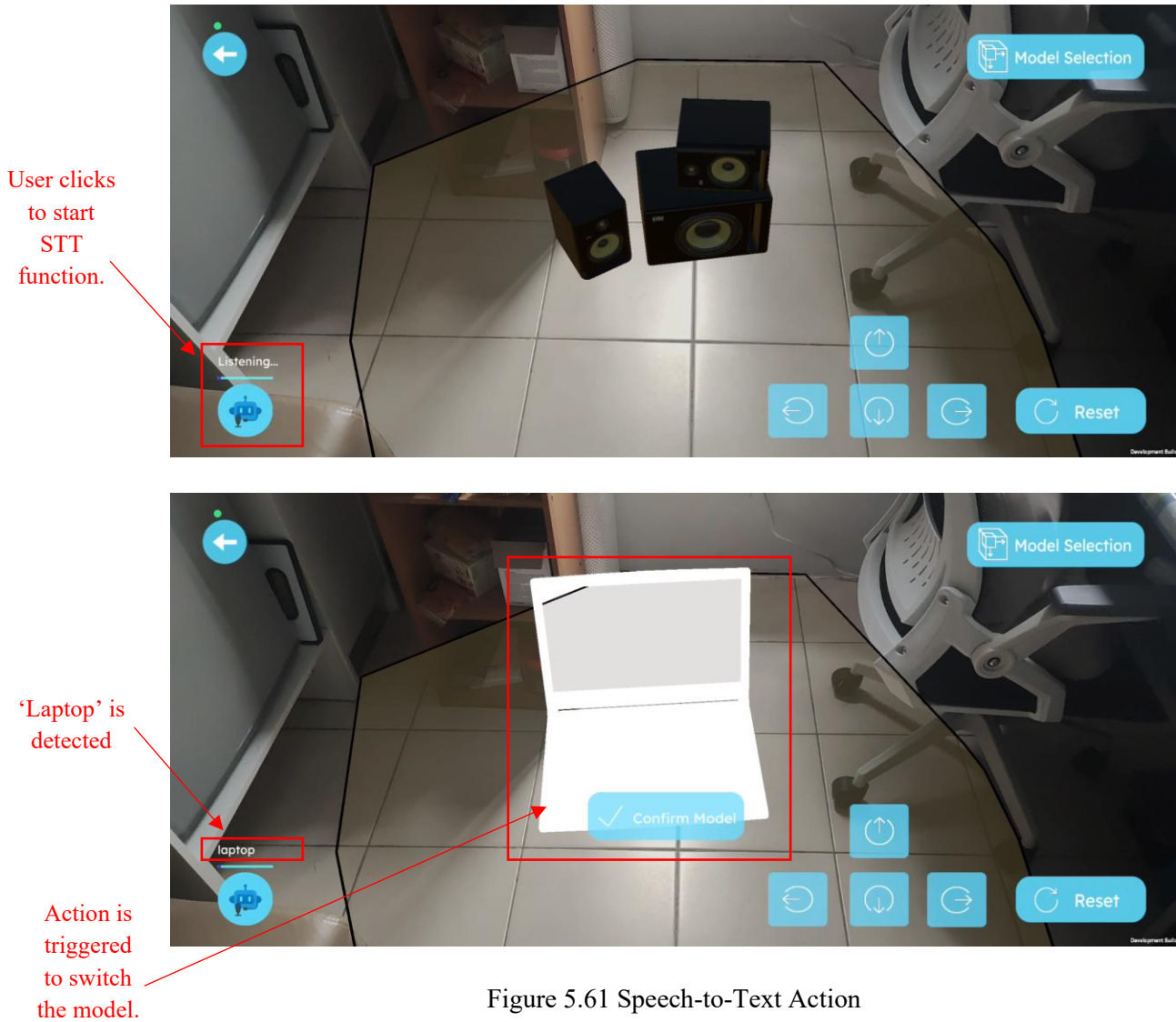


Figure 5.61 Speech-to-Text Action

Figure 5.62 shows that when the user clicks the laptop model, the information panel is displayed. When the user clicks the “Explore More” button, labels appear for key parts of the laptop, including the webcam, charging port, internal keyboard, touchpad, power button, and USB port. Similar to the monitor model, the laptop can be rotated to view different positions and examine the labels in more detail.

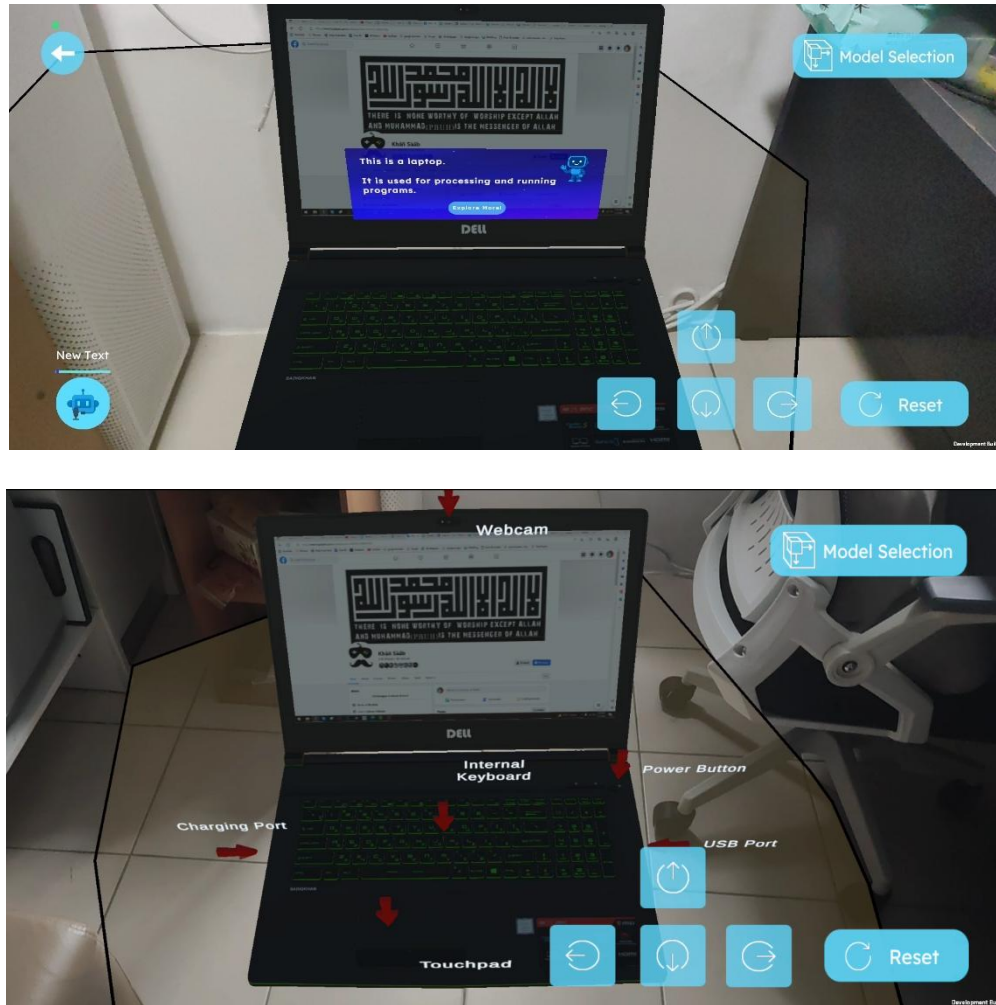


Figure 5.62 Information Panel for Laptop Model (Laptop)

Figure 5.63 shows the information panel for all the components for the monitor model, including the touchpad, internal keyboard, charging port, power button (laptop), webcam, and USB Port.

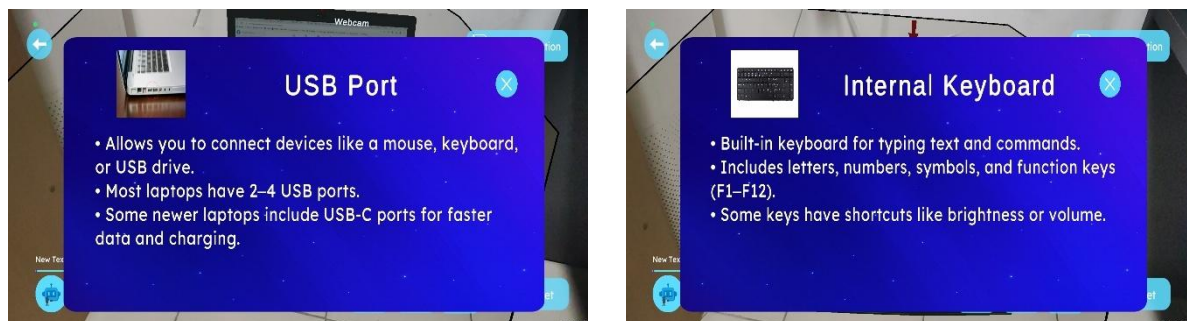


Figure 5.63 Information Panel for All Components (Laptop)

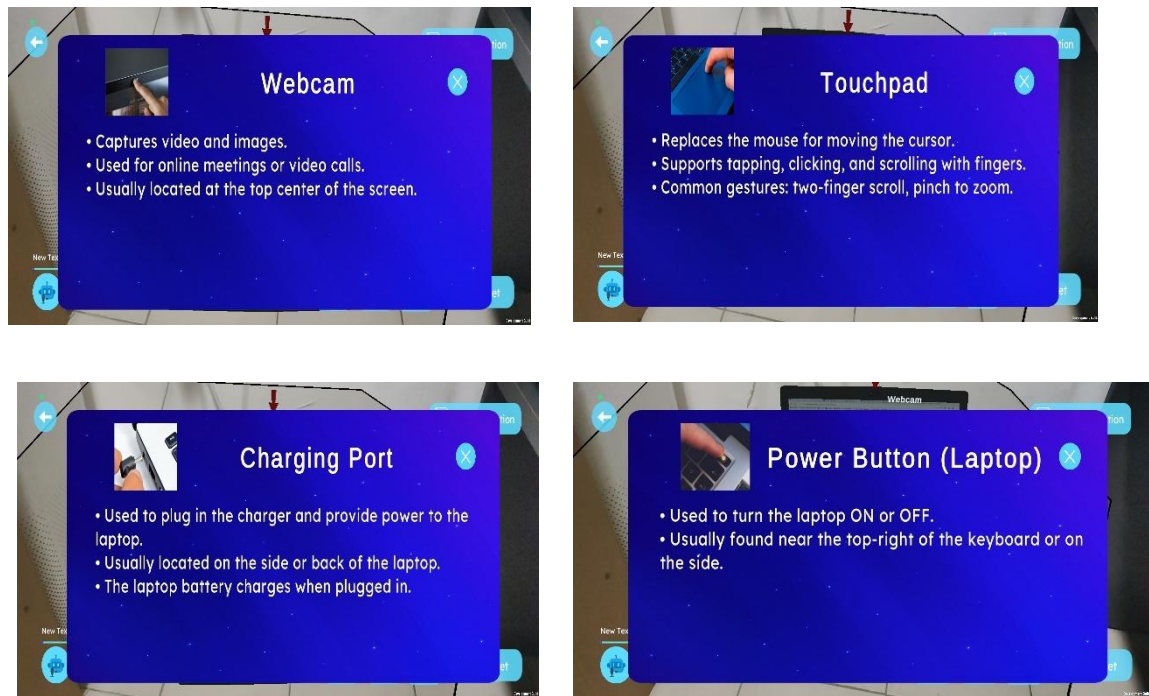


Figure 5.63 Information Panel for All Components (Laptop) (Cont.)

AR Quiz

Figure 5.64 shows the marker-based image used for the AR Quiz. When the user points the phone at this marker, the AR tutorial begins. All relevant models and panels are spawned.



Figure 5.64 AR Quiz Marker

Figure 5.65 shows the main screen of the AR Quiz module. It consists of a “Back” button (←) (top left), “Reset Position” button (bottom left), “Reset Scale” button (bottom left), and “History List” button (top right).



Figure 5.65 AR Quiz Main Screen

Figure 5.66 shows that when the user points the device at a marker, the model selection interface appears. Above the model are the “Choose Model” and “Unchoose Model” buttons. On either side of the model are “left (←)” and “right (→)” arrows, which the user can click to switch between models or swipe with a finger to move left and right. Beside the AR model robot, a text label “Model Selection: ____” is also displayed, which automatically updates whenever the user changes the selected model.

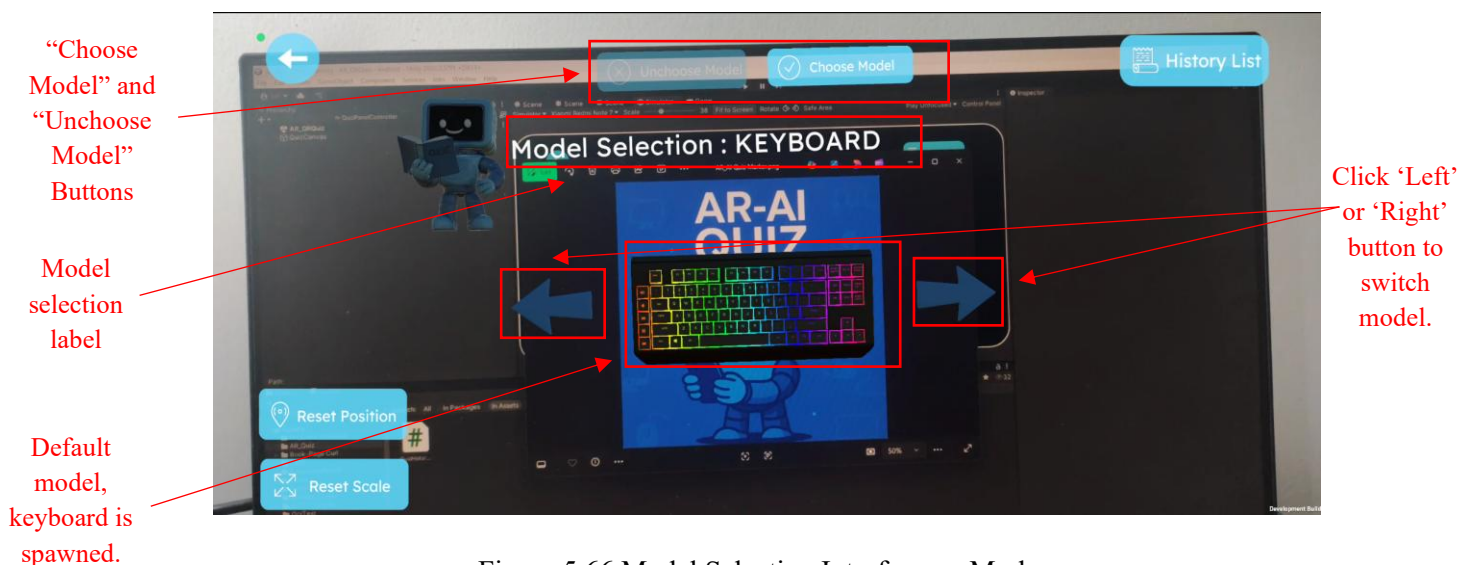


Figure 5.66 Model Selection Interface on Marker

Figure 5.67 shows the remaining model options, including the mouse, speaker, monitor, and laptop. When switching between models, the label updates accordingly. At the same time, a sound effect is played to announce the selected model (e.g., “mouse,” “keyboard,” “laptop”). In addition, swiping sound effect is included during the switching process to provide an immersive and engaging experience for users.

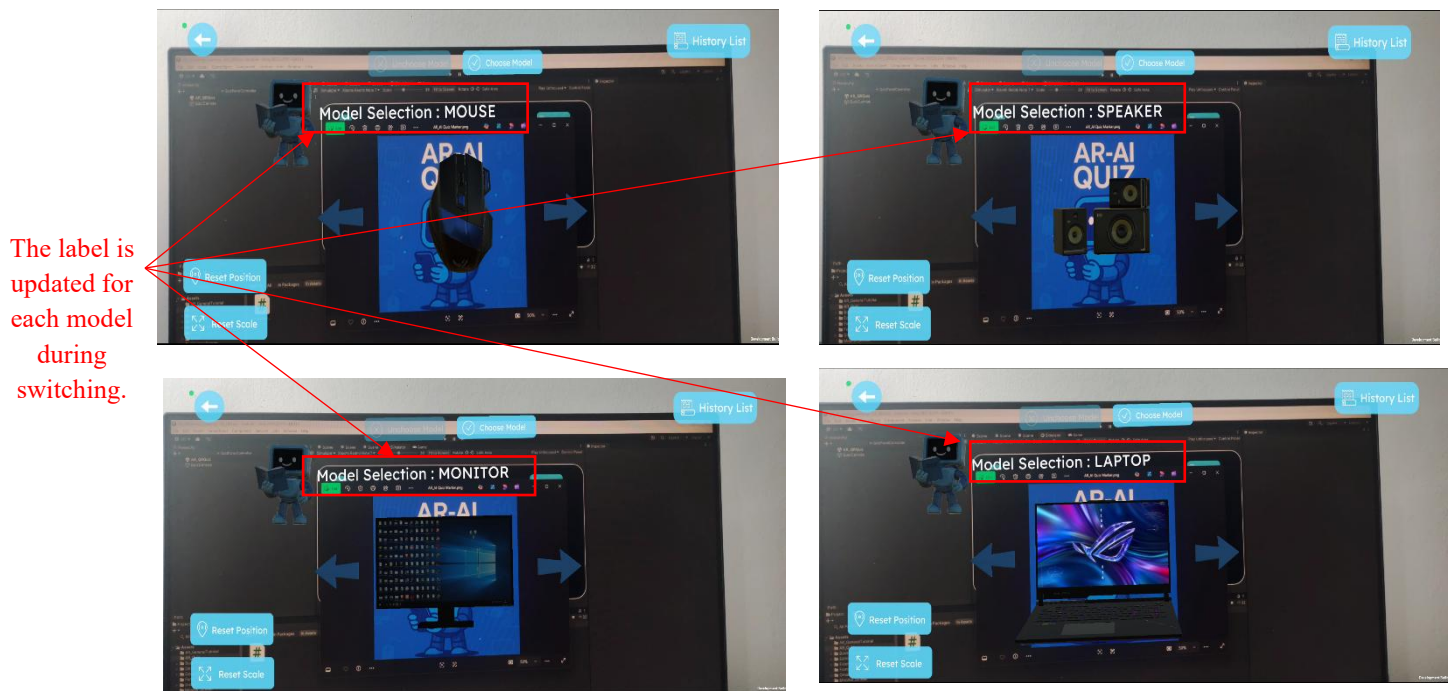


Figure 5.67 Model Switching

Figure 5.68 shows that when the user clicks the “Choose Model” button, the current model is locked, and the button becomes inactive (disabled). At this point, the quiz panel is spawned. The panel presents a set of default AI-generated questions for the user to answer. For the keyboard model, the first question is fixed and requires the user to locate specific keys on the keyboard. For example, as shown in Figure 5.68, the user is asked to click the ‘Windows’ key by identifying its position on the model. If the model appears too small, the user may enlarge it using a two-finger gesture or rotate it with one finger. The Reset Position and Reset Scale buttons can also be used to return the model to its original orientation and size if necessary.

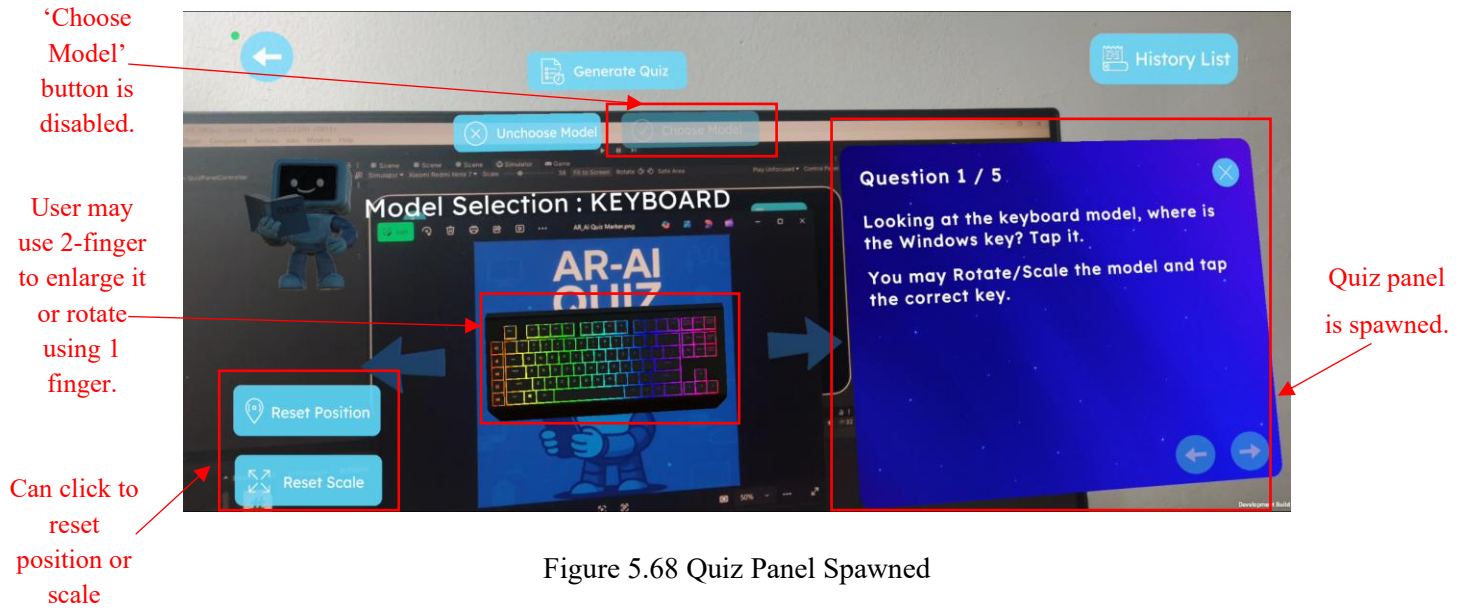


Figure 5.68 Quiz Panel Spawned

Figure 5.69 shows that when the user answers a quiz question correctly or incorrectly, a feedback response is provided. A correct answer is indicated with the label 'Correct' in green, accompanied by a positive sound effect, while an incorrect answer is shown with the label 'Try Again' in red, along with an error sound effect.

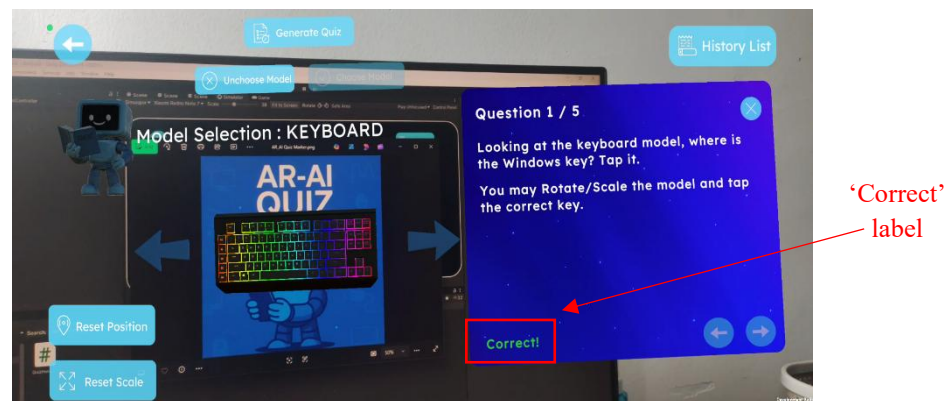


Figure 5.69 Correct and Wrong Label Feedback

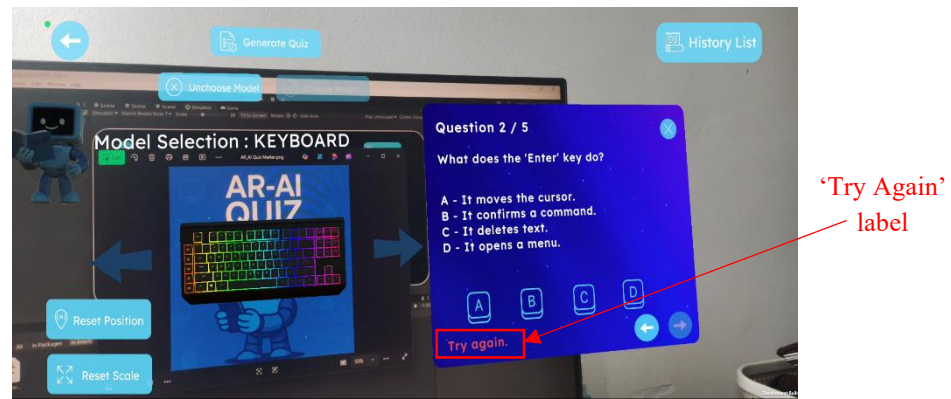


Figure 5.69 Correct and Wrong Label Feedback (Cont.)

Figure 5.70 shows that when the quiz is completed, the message ‘Quiz Complete!’ is displayed in the panel.

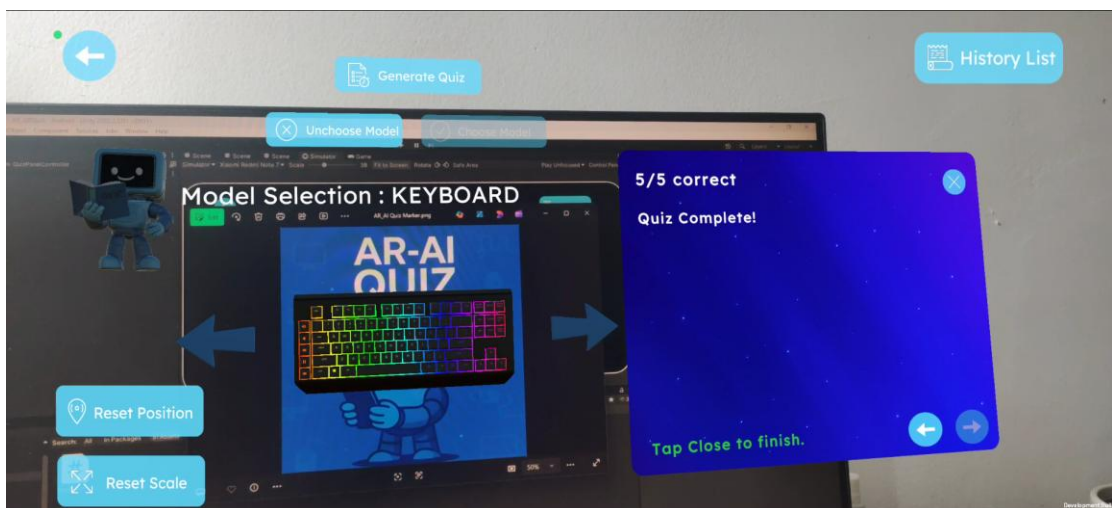


Figure 5.70 Finished Quiz Message

Figure 5.71 shows that when the user clicks the “Unchoose Model” button, the button becomes inactive while the “Choose Model” button is reactivated. The generated quiz and its panel also disappear. At this stage, the user may select a new model to begin a new quiz.

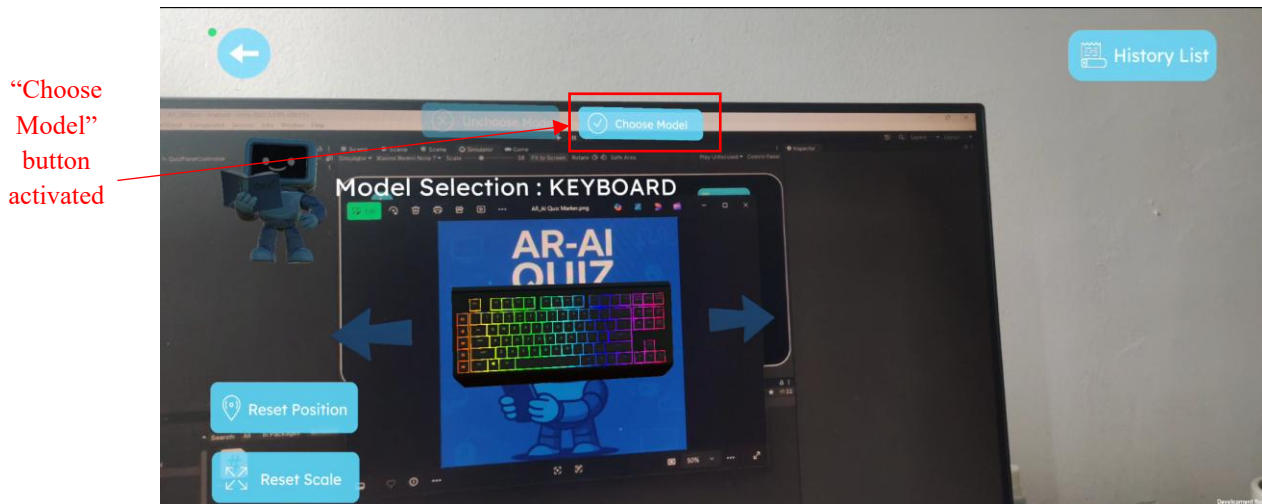


Figure 5.71 Unchoose Model Action

Figure 5.72 shows that when the user selects a new model (in this case, the laptop model) and clicks the “Generate Quiz” button, the request is sent to OpenAI to generate quiz questions along with the corresponding speech output through the Text-to-Speech (TTS) function. When the user clicks the button, a label “Generating quiz...” is displayed to inform the user that OpenAI is processing the request and preparing the quiz.

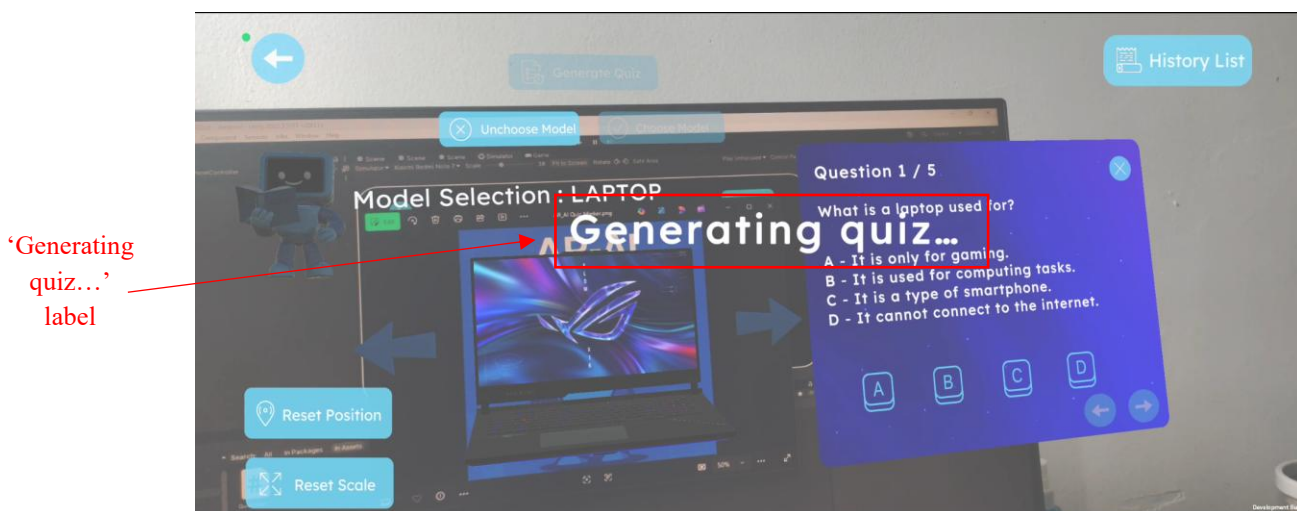


Figure 5.72 Triggering the Generate Quiz Button

Figure 5.73 shows that after the quiz generation process begins, a progress label “Preparing voice ... 0/5 → 5/5” is displayed, functioning like a breadcrumb indicator.

Once the preparation is complete, the label “Quiz Generated !!!” appears to confirm that the quiz is ready.

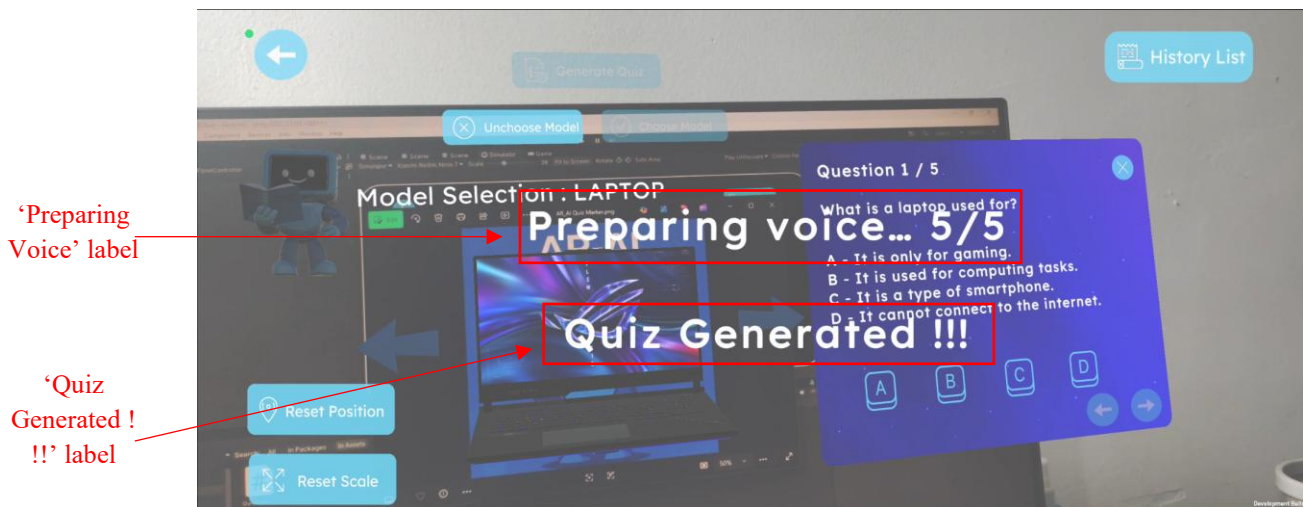


Figure 5.73 Second Feedback Label

Figure 5.74 shows that the panel updates to display newly generated questions provided by OpenAI. Along with this, the voice output is played simultaneously when the new questions are displayed, allowing the user to have a more immersive learning experience while answering the quiz.

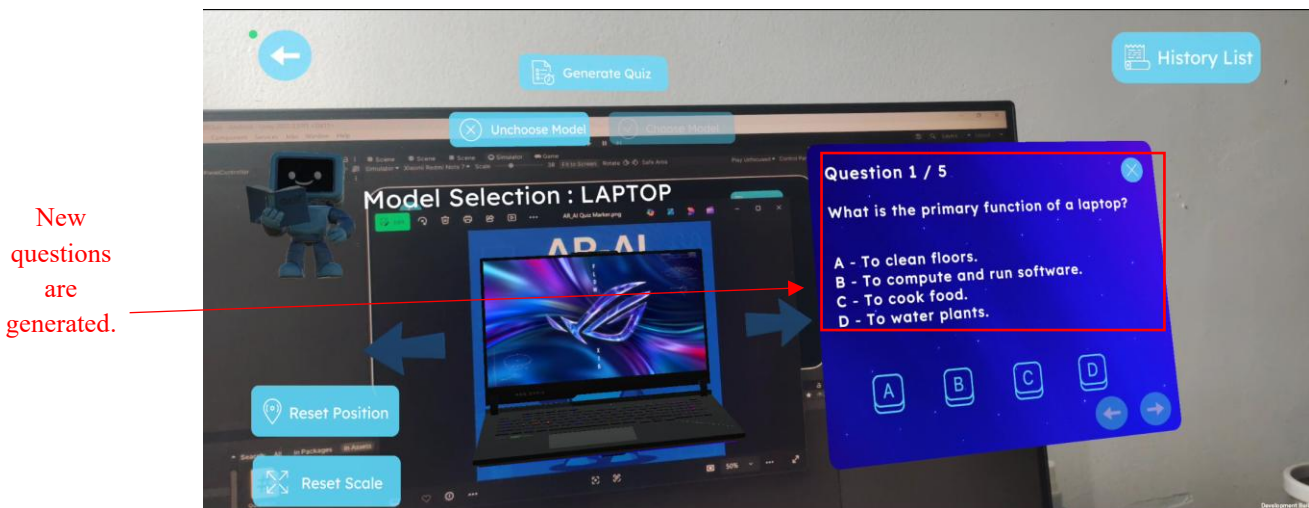


Figure 5.74 New Question Generated by OpenAI

Figure 5.75 shows the additional four questions generated by OpenAI.

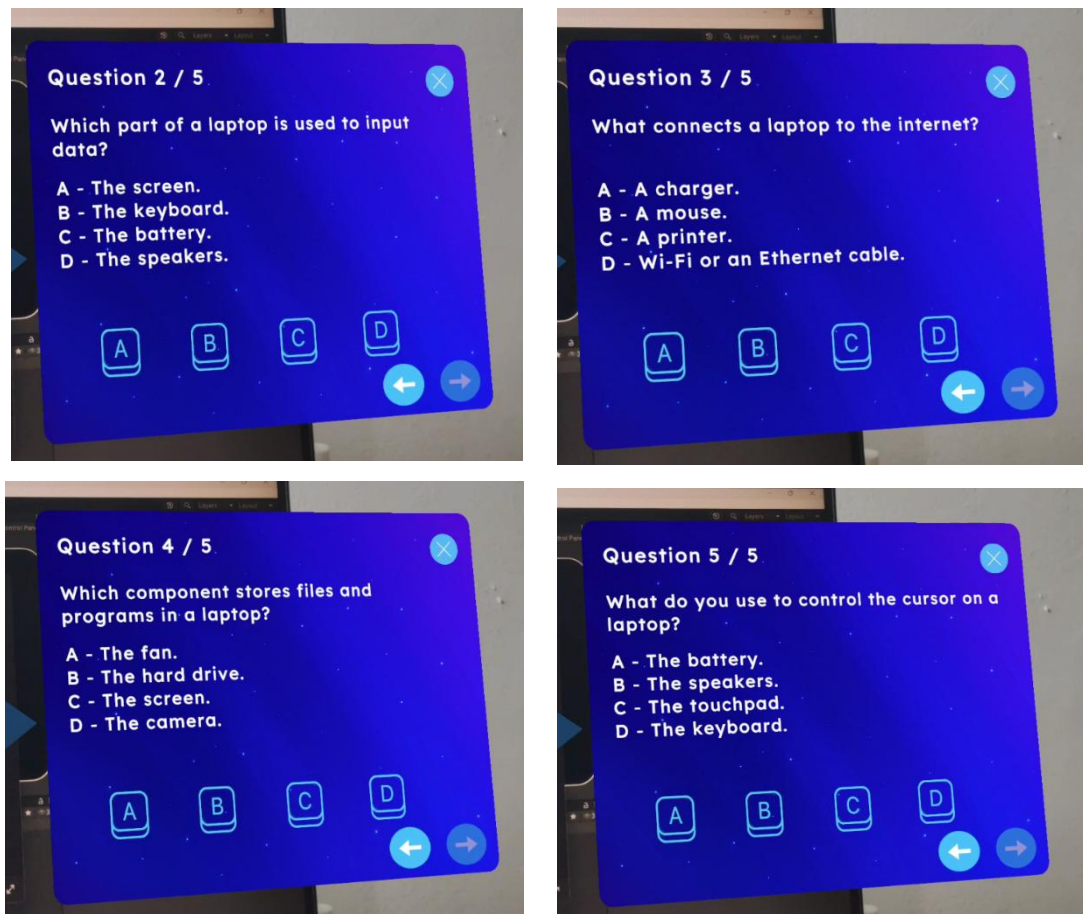


Figure 5.75 Other Questions Generated by OpenAI

Figure 5.76 shows the quiz history list, which appears when the user clicks the “Quiz History” button in the top-right corner. After clicking it, the user can view the quizzes previously answered, such as the keyboard and laptop quizzes. The history is arranged with the most recent quiz at the top. Each entry is displayed with its timestamp, model name, and the total number of correctly answered questions.

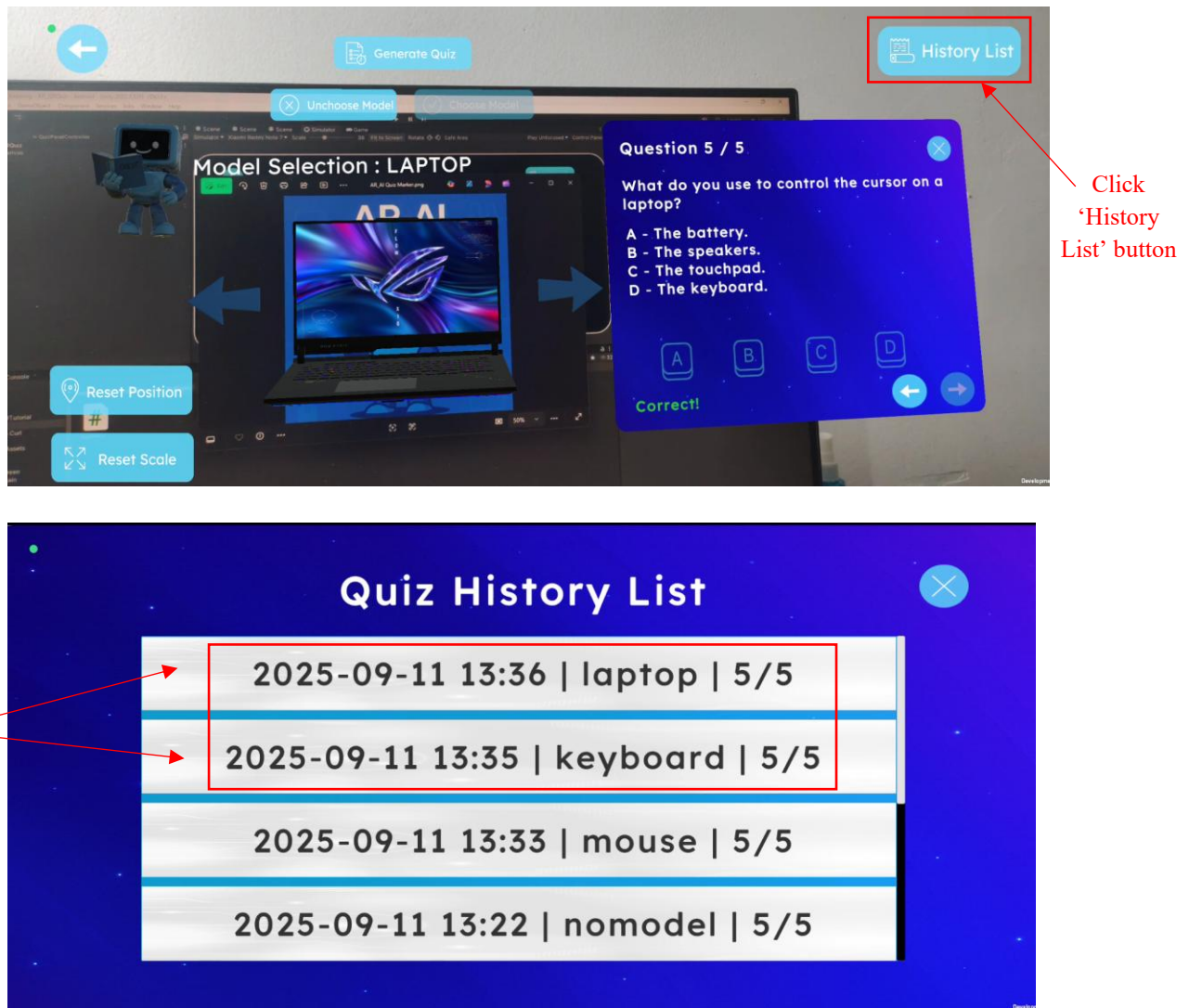


Figure 5.76 Quiz History List

Figure 5.77 shows that when the user clicks an item in the history list, the quiz detail panel is displayed. In this panel, the user can review all the quizzes previously answered, including the user's response, the correct answer, and the result.

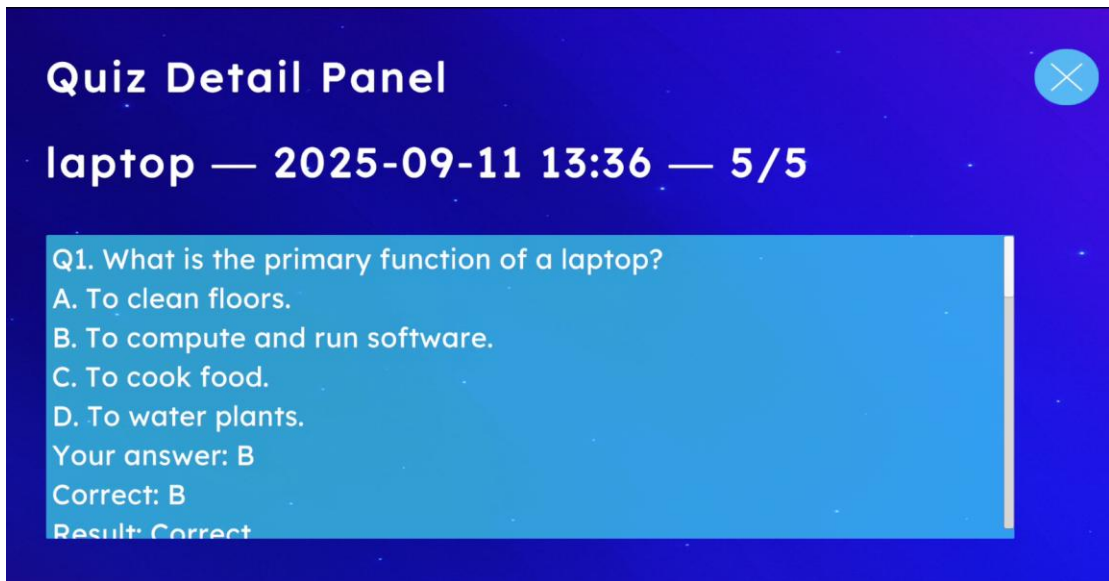


Figure 5.77 Quiz Detail Panel

5.4.3 AR Guide

Microsoft Word Tutorial, Snipping Tool Tutorial, Send Email Tutorial

As their operation flows are identical, starting from detection, displaying step-by-step instructions, and synchronizing text, voice, and visual cues, so the explanation is presented together in this section. The subsequent descriptions and screenshots therefore cover the operation of both types of tutorials in a combined manner.

In order to provide a complete view of the desk simulation in a consistent manner, all three tutorial scenes are set to a fixed portrait orientation. As shown in Figure 5.78, the interface includes a “Back” button (←) positioned at the top-left corner, while the bottom-left area contains the Text-to-Speech (TTS) icon and the “Change Mode” button.



Figure 5.78 AR Guide Main Scene

Figure 5.79 shows that user pointing the camera to scan the environment. Once a plane is successfully detected, a highlighted grey surface is displayed to indicate the area where the AR content can be placed.

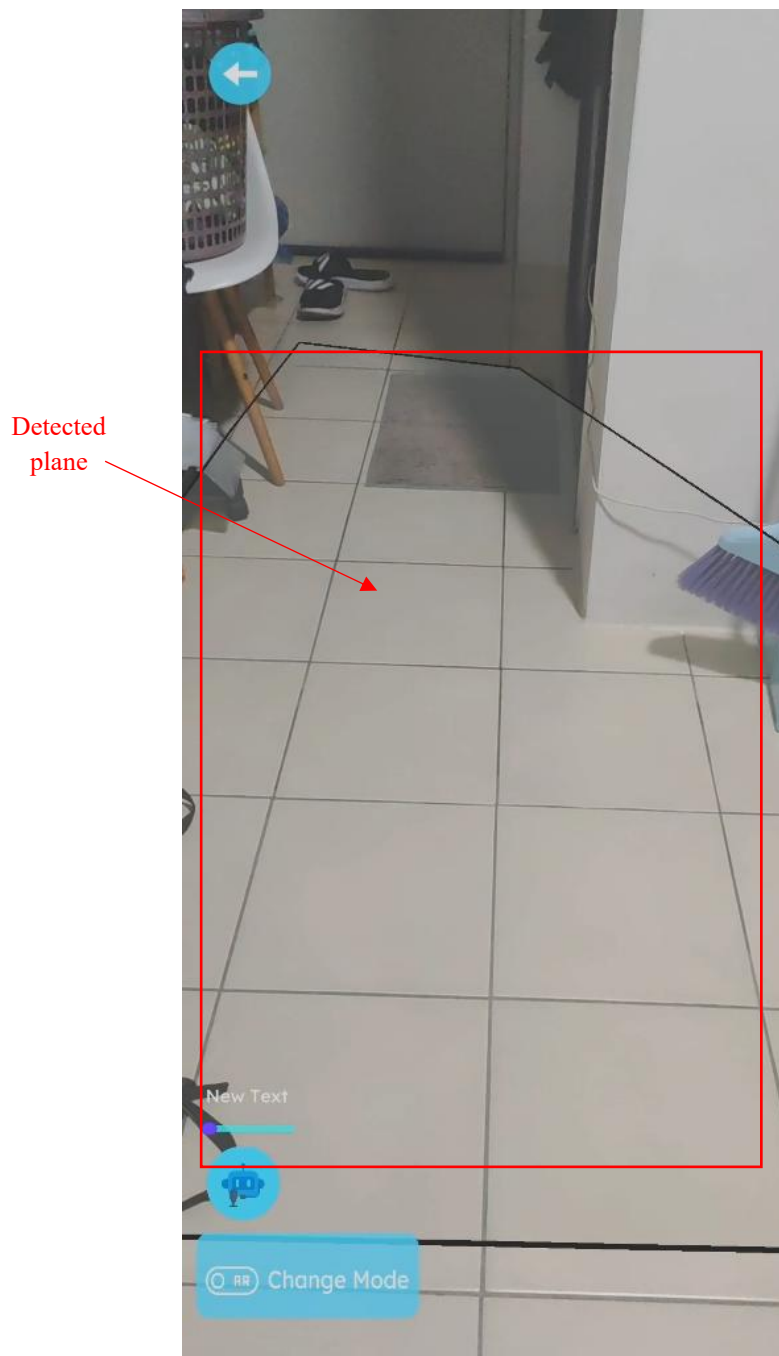


Figure 5.79 Plane Detection in AR Guide

Figure 5.80 shows that after the user clicks on the detected plane, the desk simulation model appears. The simulation includes a virtual monitor, keyboard, and mouse, along with an information panel and key highlight labels that display instructions or visual cues. At the same time, Text-to-Speech (TTS) narration is automatically played to guide the user through the tutorial steps.

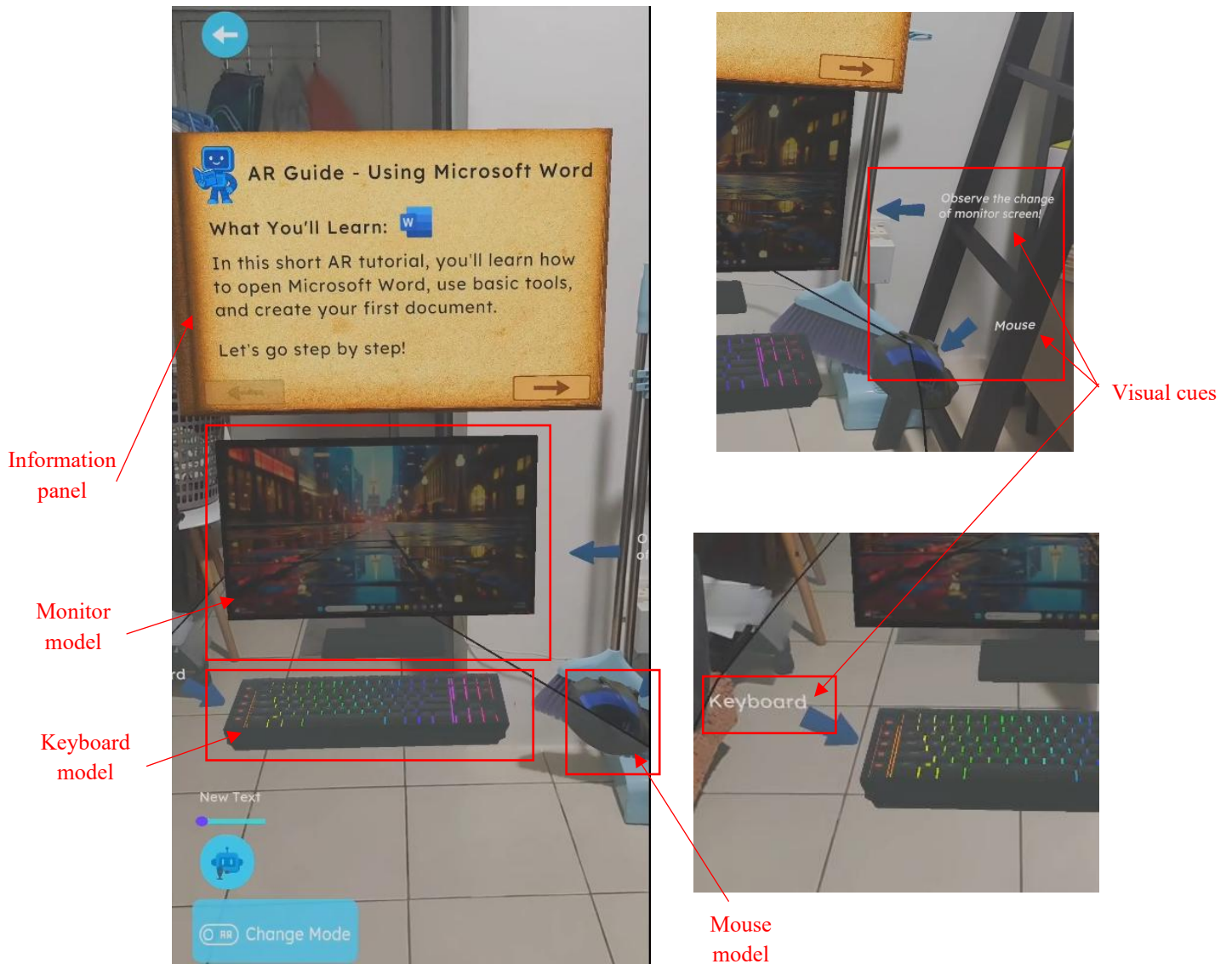


Figure 5.80 Desk Simulation Model Placement

Figure 5.81 shows that when the user clicks on the information panel, a control panel appears at the lower-right corner. This panel allows the user to adjust the position of the canvas by moving it left, right, up, or down. If the user wishes to return the canvas to its original placement, then the “Reset Position” button can be clicked to restore it instantly. These controls ensures that the canvas can be repositioned to achieve the best possible viewing experience for the user.

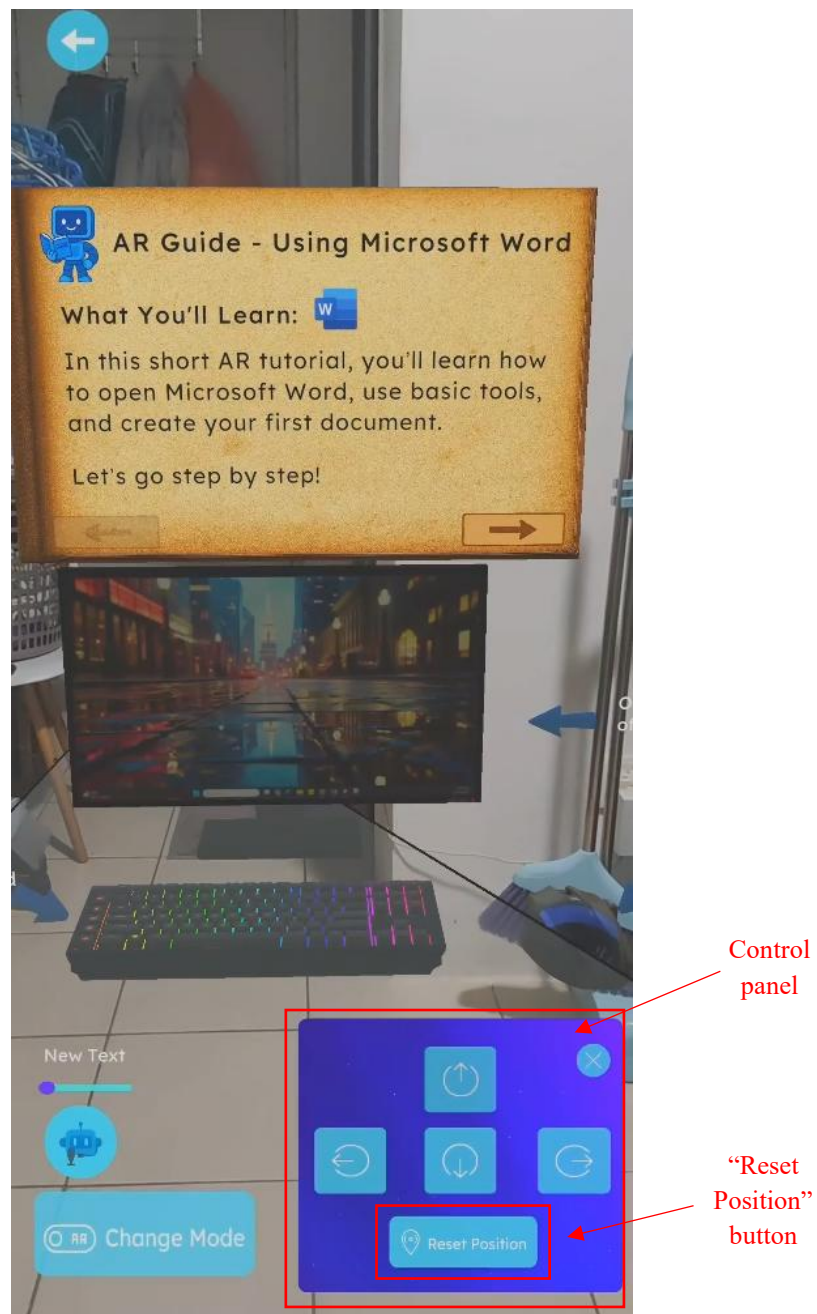


Figure 5.81 Information Panel Control Options

Figure 5.82 shows another control panel, which appears when the user clicks on the entire desk simulation model. Similar to the canvas control, this panel allows the user to move the model left, right, up, or down. In addition, it provides options to scale larger or scale smaller to improve the visibility of the overall model. The panel also includes

“Reset Position” and “Reset Scale” buttons, enabling the user to quickly restore the model to its original placement and size.

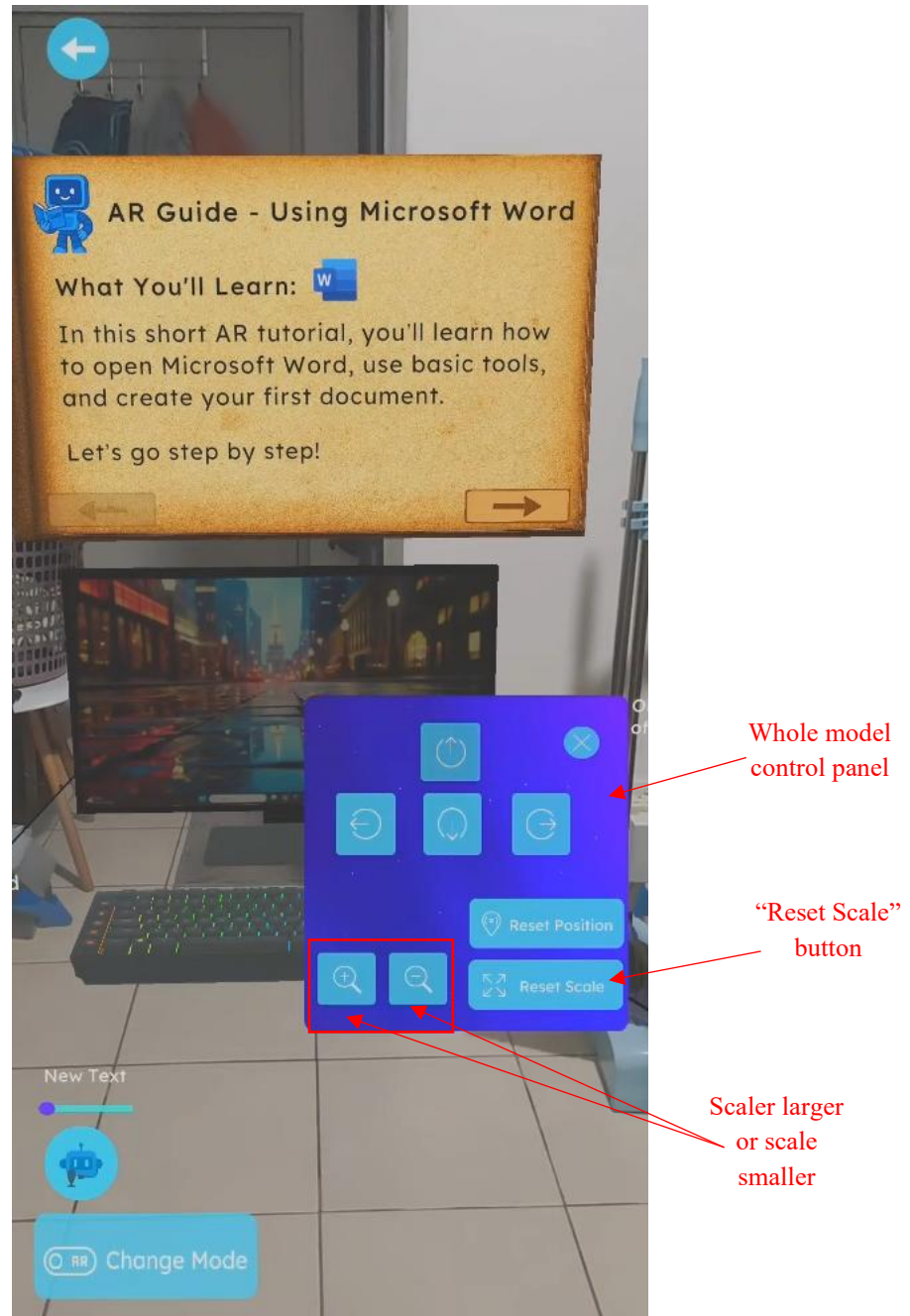


Figure 5.82 Whole Model Control Panel

Figure 5.83 shows that after adjusting the model, the user can continue with the tutorial. By clicking the “Right” (→) button, the next information panel appears together with the synchronized Text-to-Speech (TTS) narration. Similarly, clicking the “Prev” (←)

button returns to the previous step, with both the text and narration updated accordingly. In addition, users may click the STT icon and issue voice commands such as “next” or “back” to navigate through the tutorials hands-free.

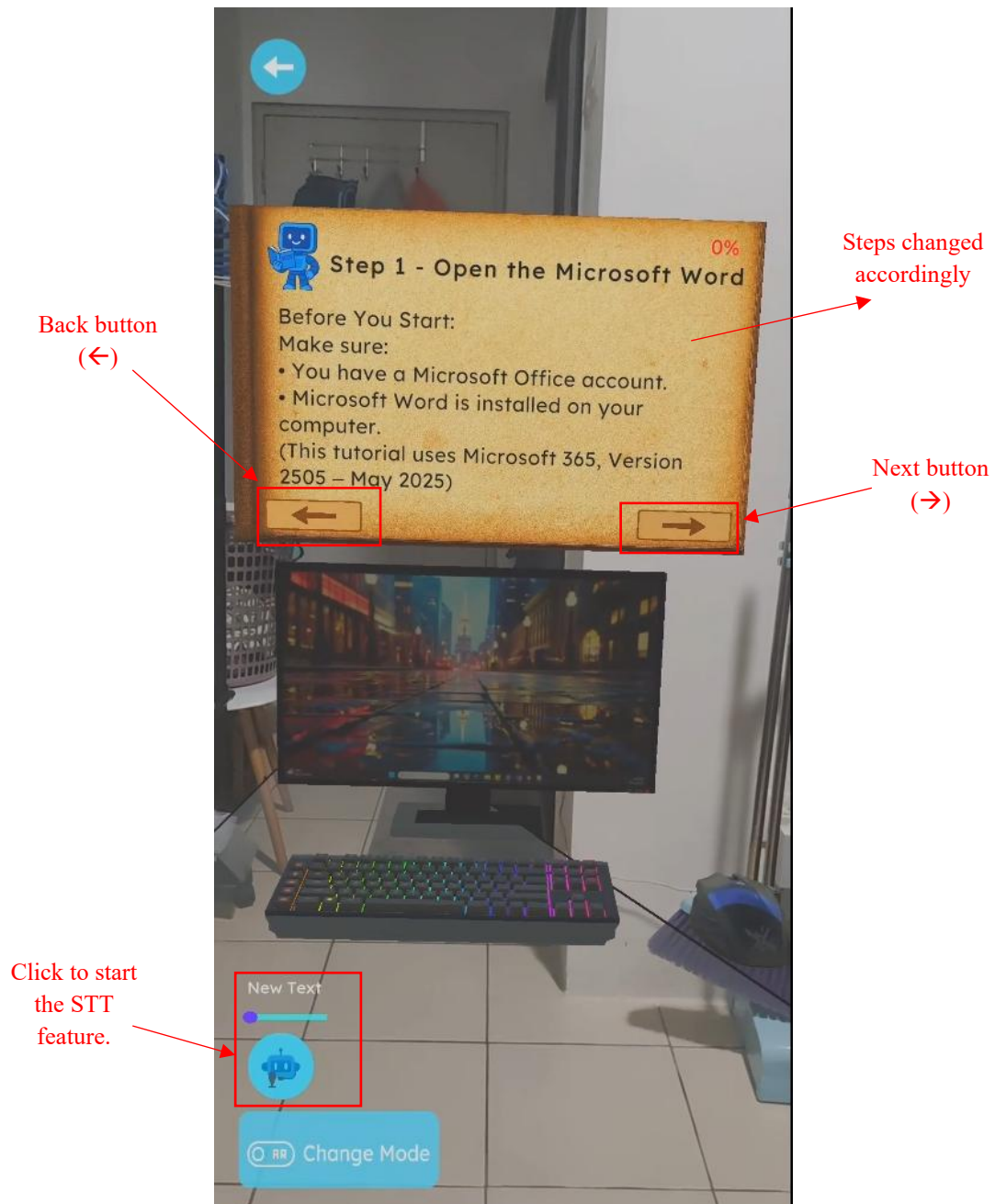


Figure 5.83 Tutorial Progression with Navigation Buttons

Figure 5.84 shows that the virtual monitor screen updates according to the current tutorial step. At the same time, visual cues are displayed when the step involves specific interactions such as moving the mouse, performing a search, or typing keys on the

keyboard. These synchronized updates help users follow the tutorial more intuitively by combining on-screen changes with corresponding physical actions.



Figure 5.84 Monitor Screen and Visual Cues

Figure 5.85 shows that when the user clicks the “Change Mode” button, the system switches to Canvas Mode. In this mode, the desk simulation model disappears, and the

user can freely move the camera and tap on the screen to spawn the canvas information panel. The spawned canvas remains synchronized with the current tutorial step from the desk simulation model, ensuring continuity of the instructions. At the same time, a “Reset Panel” button appears. If the user wishes to reset the position, clicking this button once will trigger the function, and the user can then tap the screen again to spawn the panel in a new position.

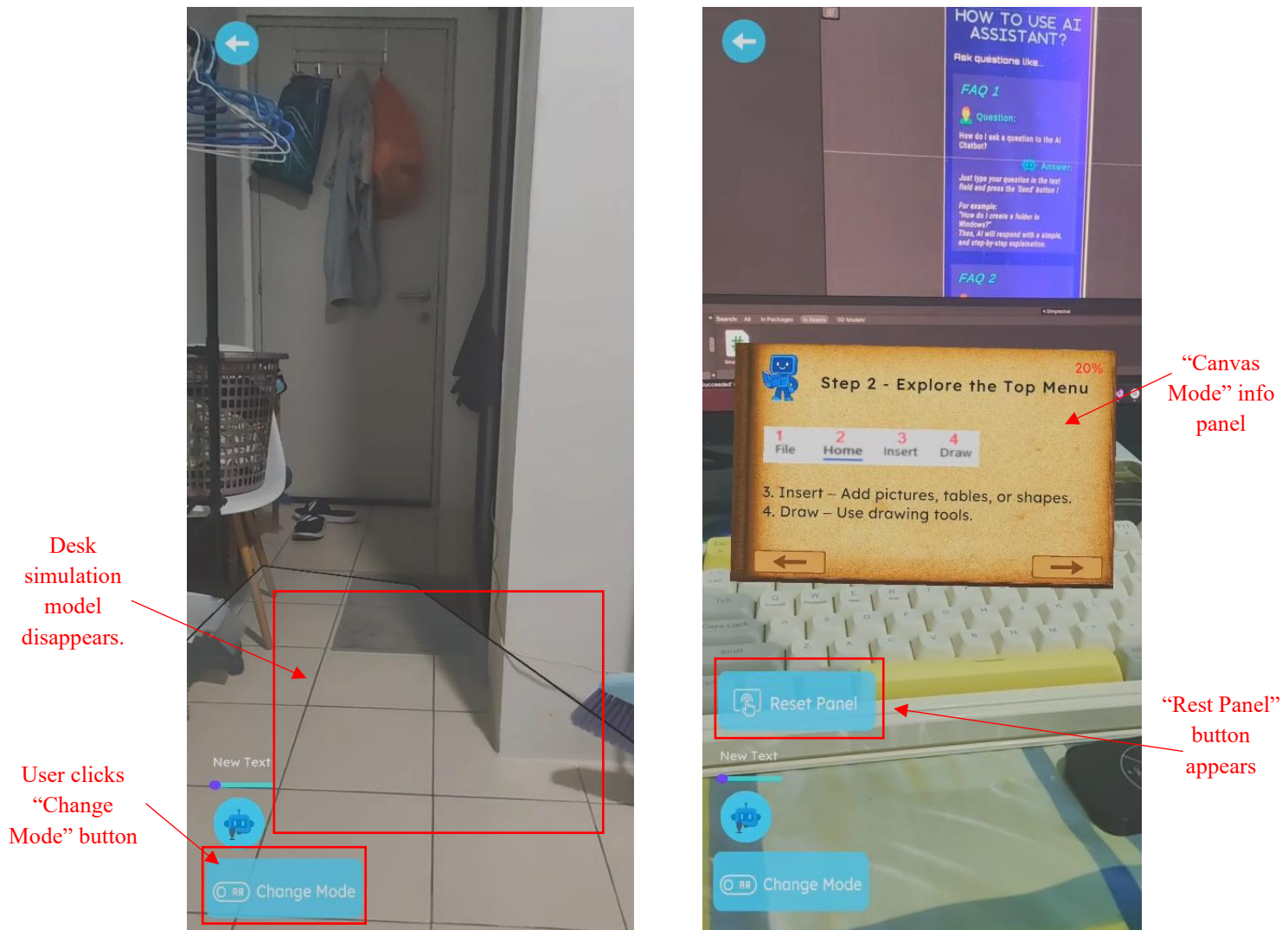


Figure 5.85 Change Mode and Canvas Synchronization (Canvas Mode)

Figure 5.86 shows that if the user wants to switch back to the 3D desk simulation model, they can simply click the “Change Mode” button again. Once activated, the Canvas Mode ends, and the desk simulation model respawns at the same location on the previously detected plane.

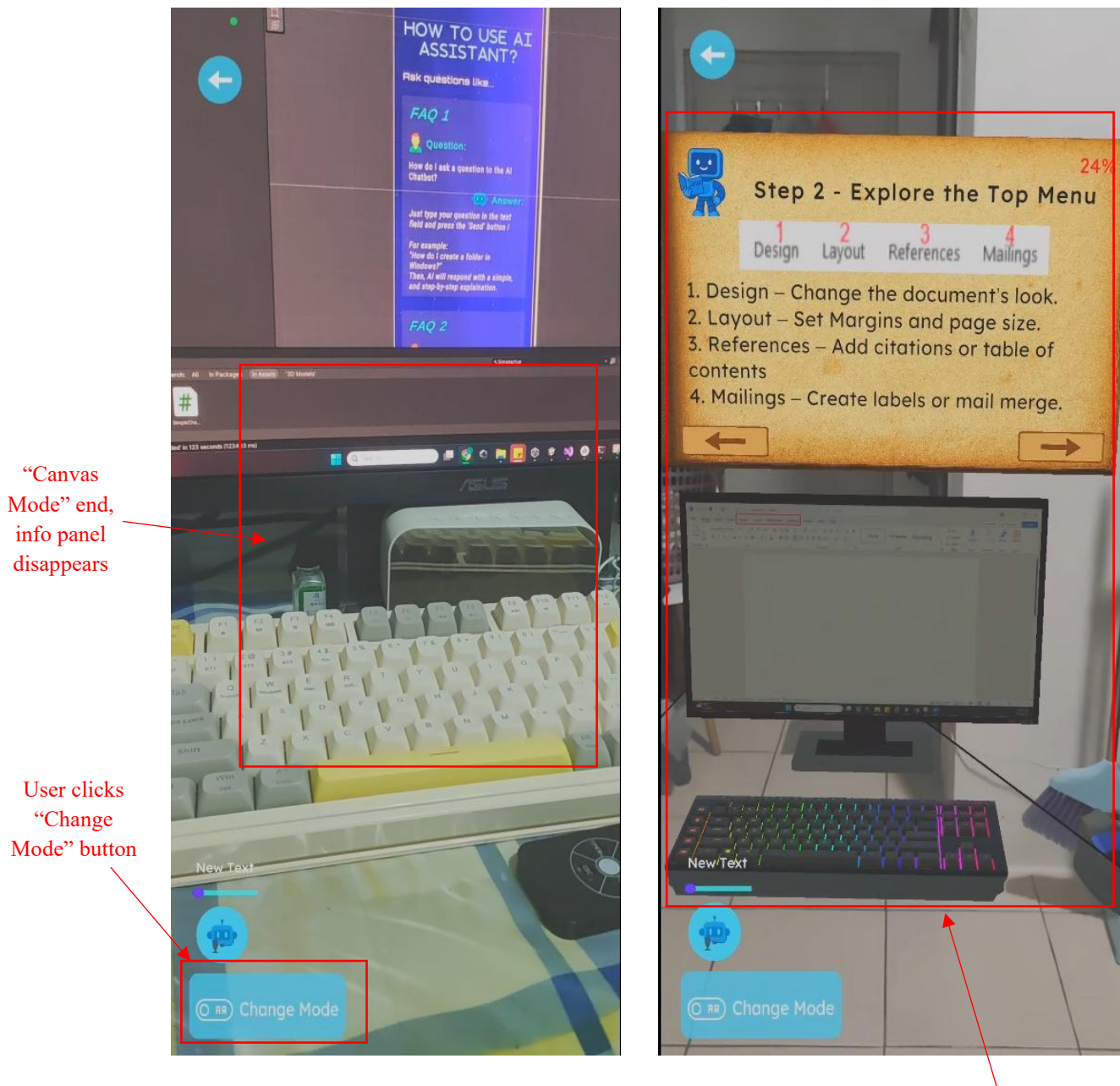


Figure 5.86 Switching Back to Desk Simulation Mode

Desk simulation appears in the same position.

Afterwards, the user may proceed step by step until the tutorial reaches completion. This describes the overall operation of the AR Guide module, which consists of 3 submodules, **Microsoft Word Tutorial**, **Sending Email Tutorial**, and **Snipping Tool Tutorial**. The total number of steps for each tutorial, including the final “tutorial completed” message, is as stated below:

- Microsoft Word: **27 steps**
- Sending Email: **20 steps**
- Snipping Tool: **16 steps**

For the full list of **completed tutorial steps** and the **corresponding monitor screen updates**, please refer to the Appendix section:

- Microsoft Word: **Appendix C**
- Send Email: **Appendix D**
- Snipping Tool: **Appendix E**

Calculator Tutorial

The next part describes the marker-based tutorial, which focuses on the operation of the Calculator Tutorial. In this case, the tutorial begins when the user points the camera at the real calculator displayed on the laptop. Once the calculator marker is detected, the system activates the tutorial and displays the first instructional panel.

Figure 5.87 shows the main screen of the Calculator Tutorial. A “Back” button (←) is placed at the top-left corner, and the screen is set to a fixed orientation to ensure a consistent and smooth user experience. At the bottom-left corner, there is an STT icon that enables voice navigation. The figure also shows the information panel spawning when the user points the camera at the calculator in the laptop screen. Once the calculator marker is detected, the information panel appears, and the user can click the “Next” button to proceed with the tutorial steps.

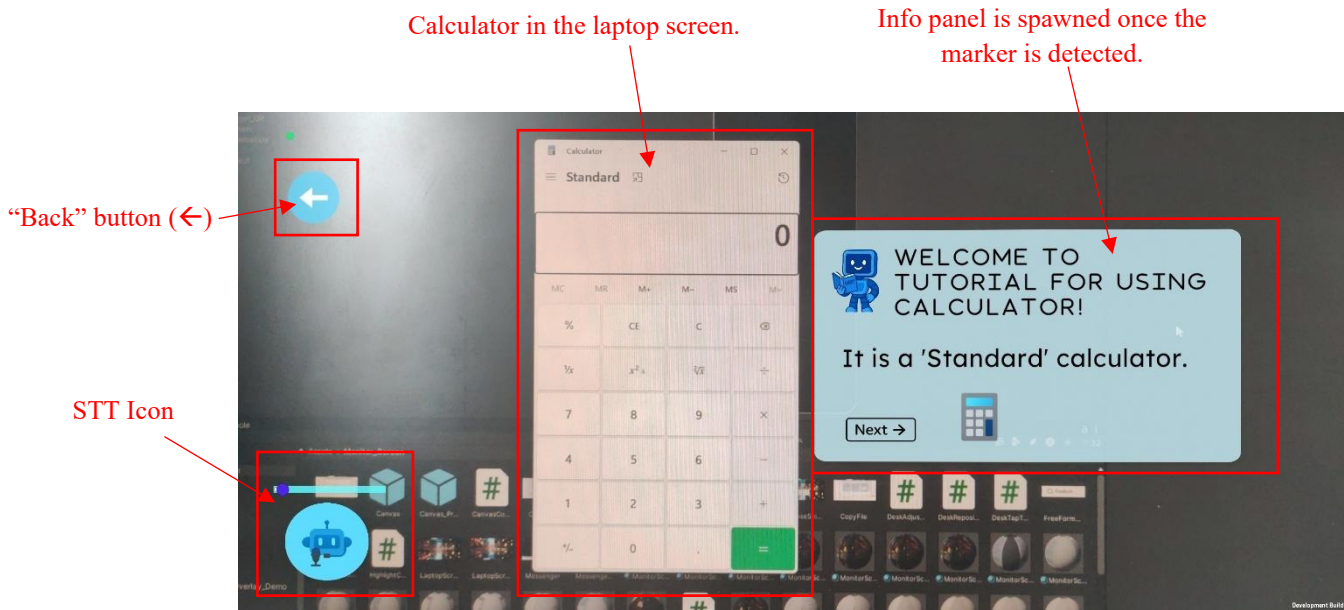


Figure 5.87 Main Screen and Info Panel of Calculator Tutorial

Figure 5.88 illustrates the sequence of steps contained in the Calculator Tutorial. Similar to the previous tutorials, the interface includes “Back” and “Prev” buttons that allow the user to navigate between steps. Navigation can be triggered either by clicking these buttons directly or by using the STT icon to issue voice commands such as “next” or “back.”

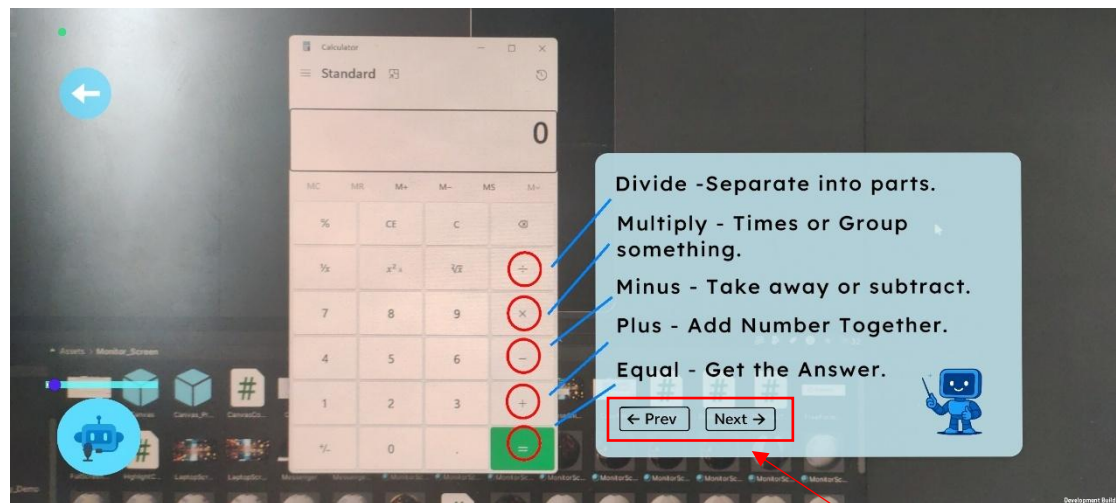


Figure 5.88 Tutorial Steps in Calculator Tutorial

“Prev” and “Next” buttons

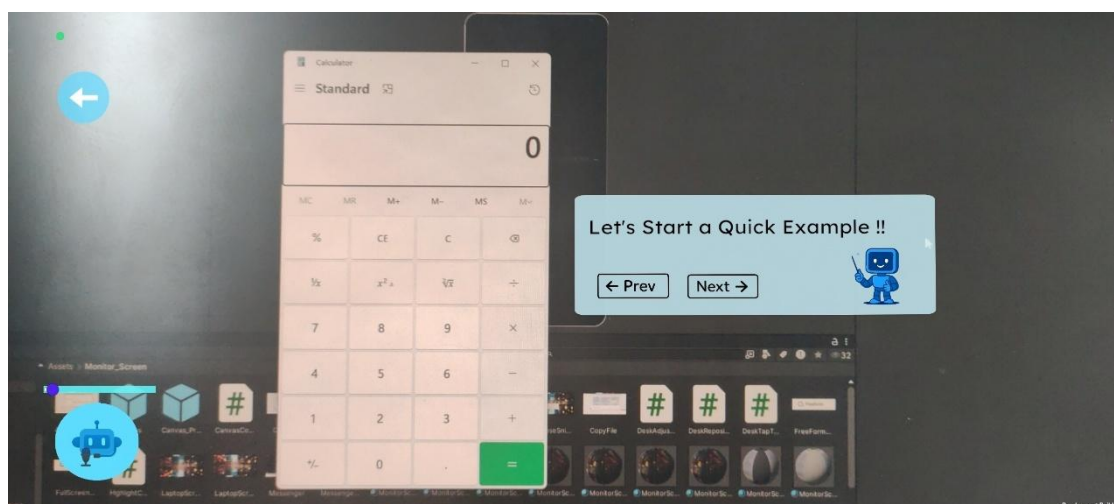
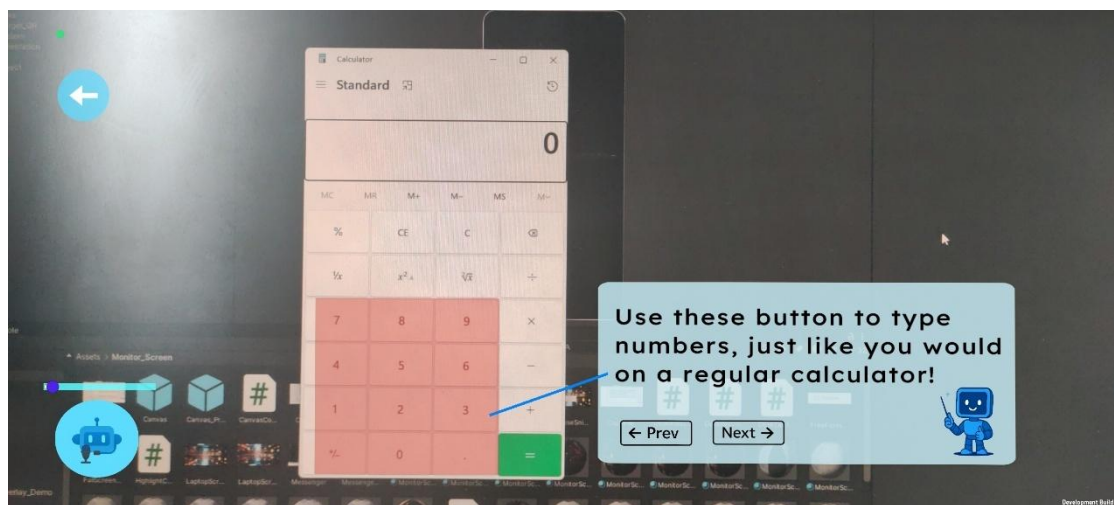
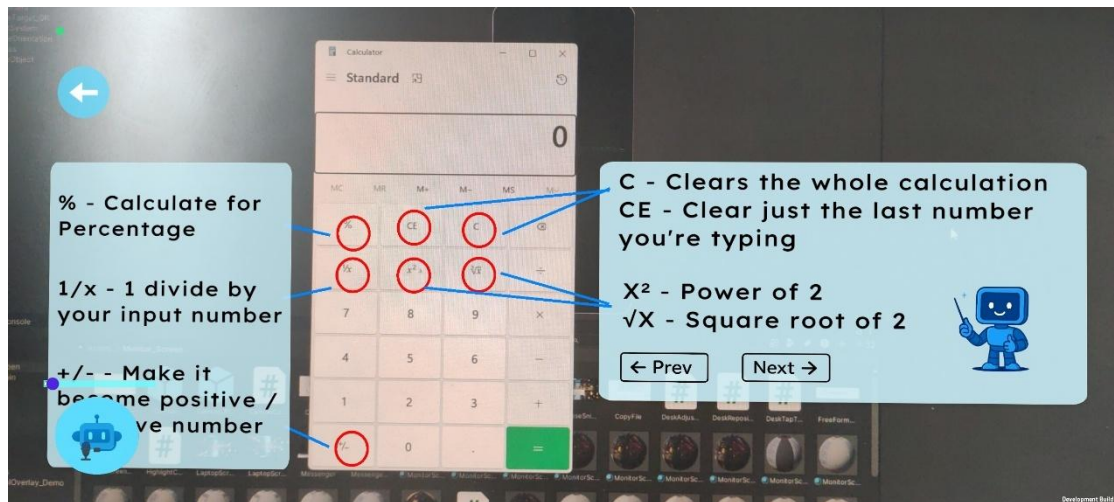


Figure 5.88 Tutorial Steps in Calculator Tutorial (Cont.)

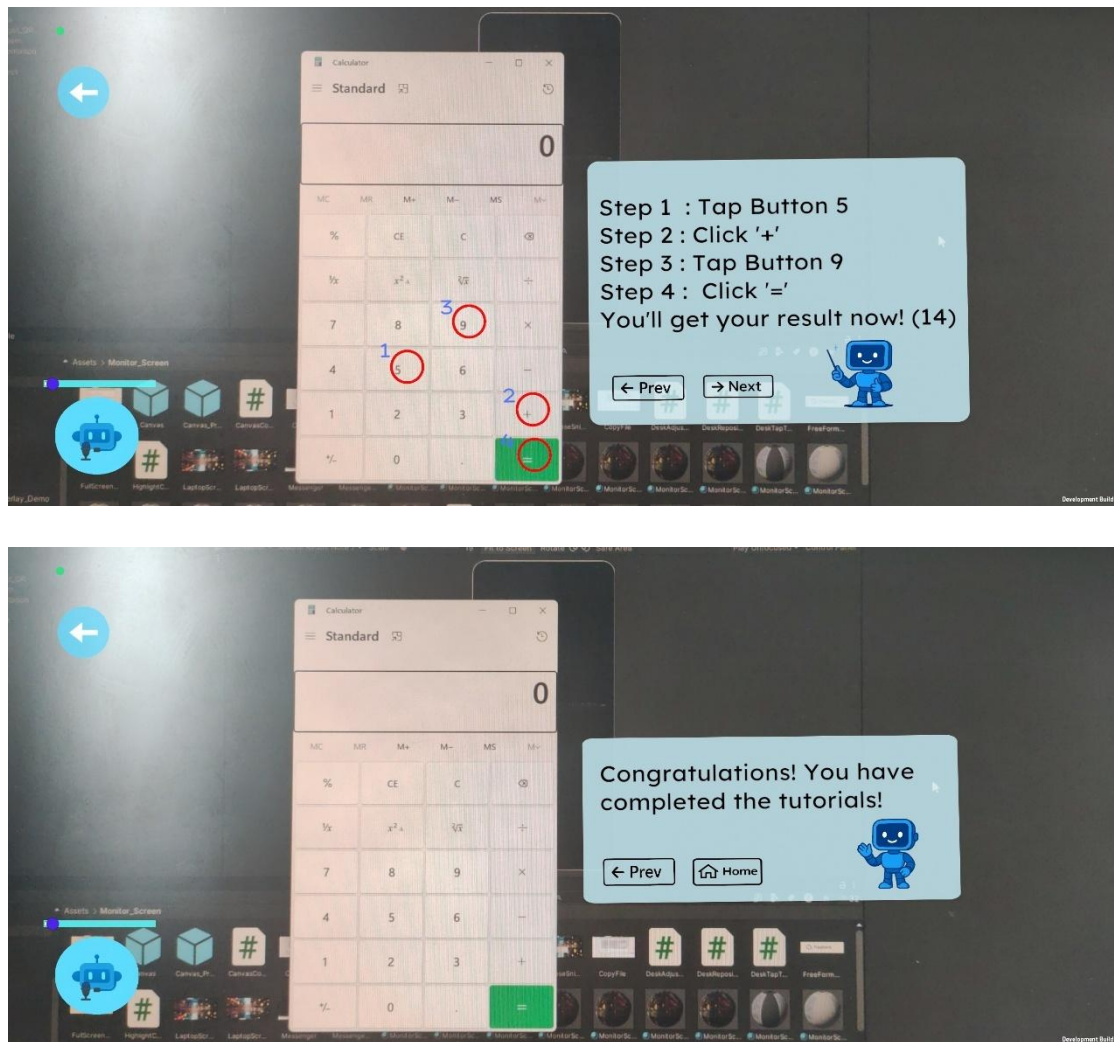


Figure 5.88 Tutorial Steps in Calculator Tutorial (Cont.)

5.4.4 AI Tutor

Figure 5.89 shows the main screen of the AR Learn module. It consists of two buttons. They are **AI Assistance** and **Frequently Asked Questions (FAQs)**.

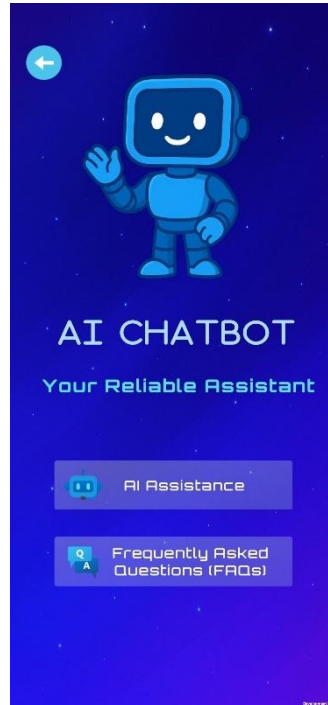


Figure 5.89 AR Learn Module Main Screen

AI Assistance

Figure 5.90 shows the main screen of the AI Assistance module. In the top-left corner, there is a “Back” (←) button, and in the top-right corner, a “History List” button. The first section allows the user to type a question, and below it is three buttons. They are “Sent”, “Gallery”, and “Take Photo”. The next section displays the AI chatbot’s responses. Below this, there is a “Canvas” Mode button.

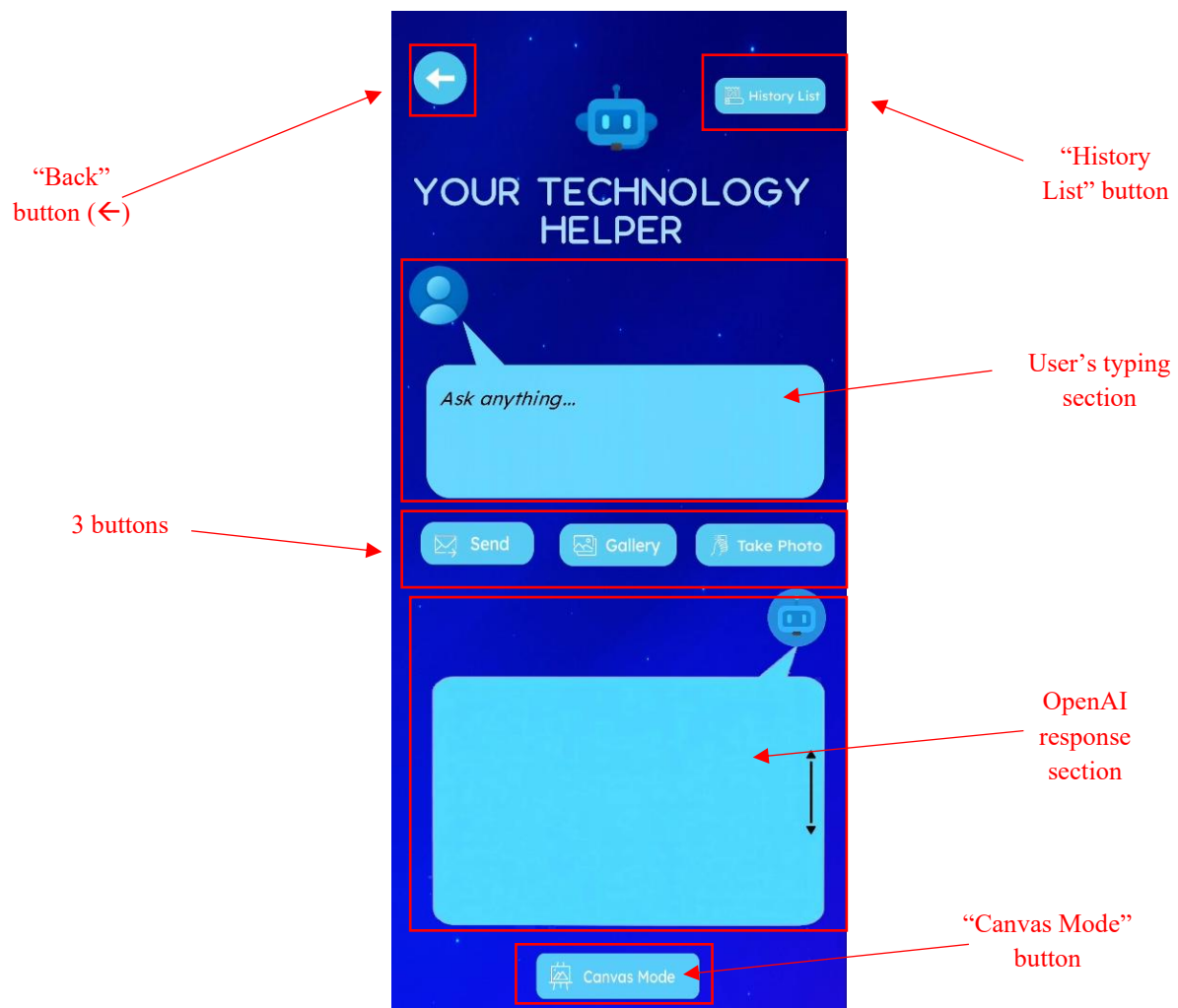


Figure 5.90 AI Assistance Main Screen

Figure 5.91 shows that when the user clicks the “Gallery” button, an option menu interface appears, allowing the user to choose whether to select an image from the gallery, file manager, or photos.

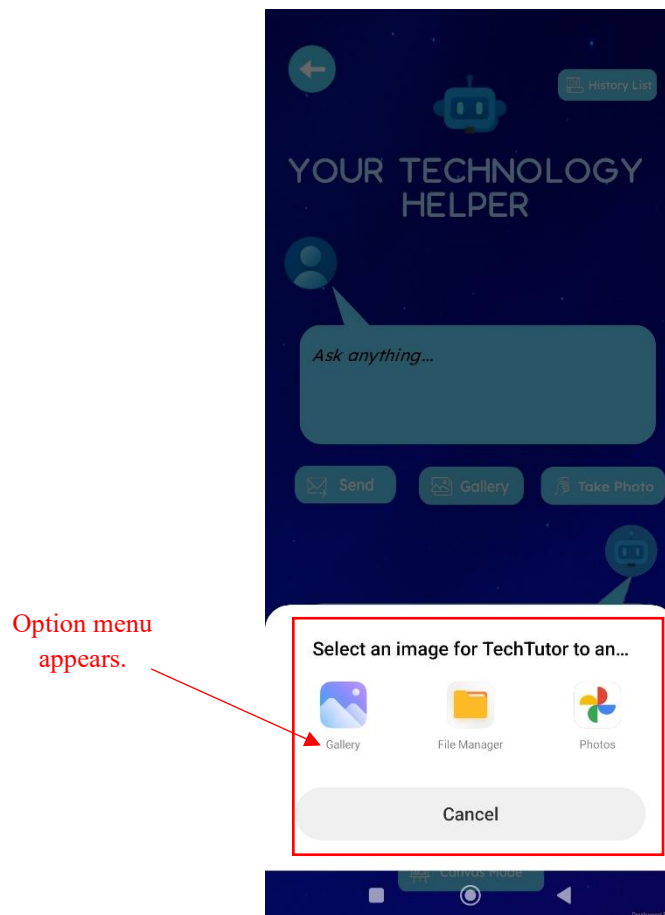


Figure 5.91 Click the Gallery Button

Figure 5.92 shows that when the user selects “Gallery” from the option menu, the system allows the user to choose an image from the device’s gallery / album.

User's
device
gallery

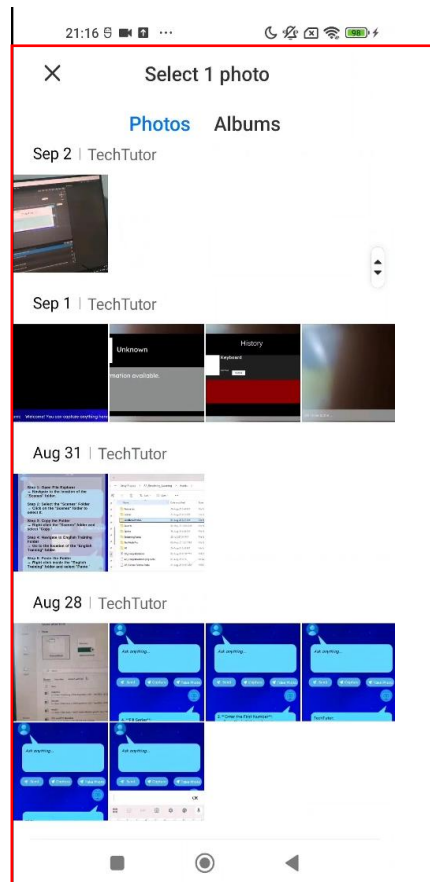


Figure 5.92 Select Photo from the Gallery

Figure 5.93 shows that after the user selects an image from the gallery, the image preview is displayed in the area above the AI response section. An “X” button in the top-right corner allows the user to delete the image and select a new one if needed.

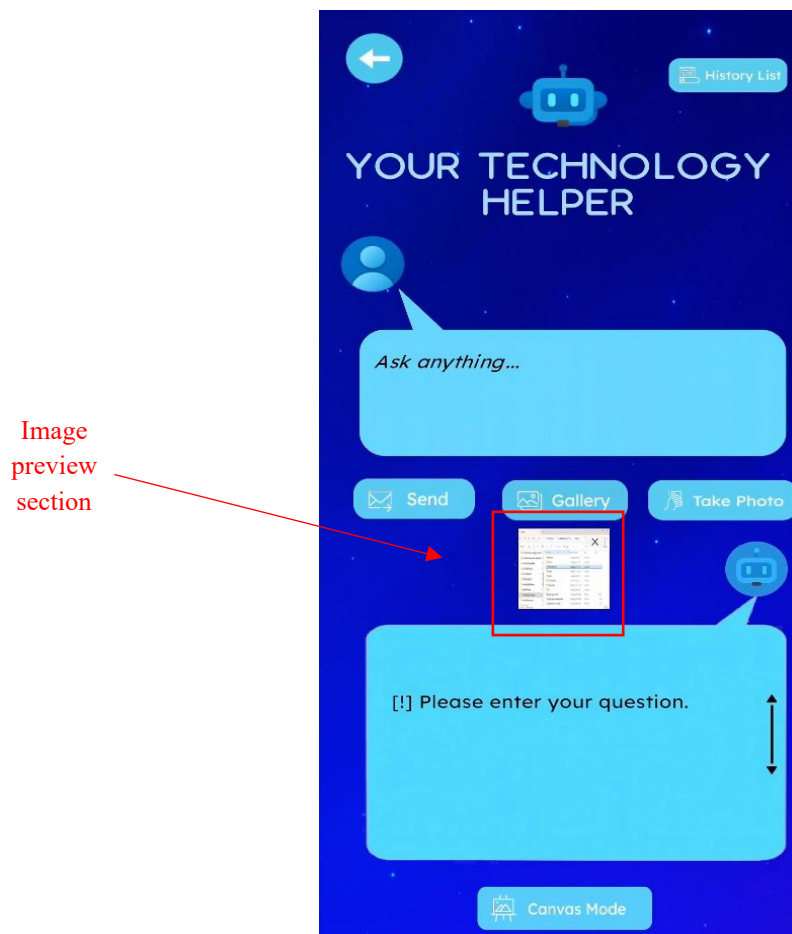


Figure 5.93 Selected Image from the Gallery

Figure 5.94 shows that the user may click the image to view it in a larger size and with more detail.

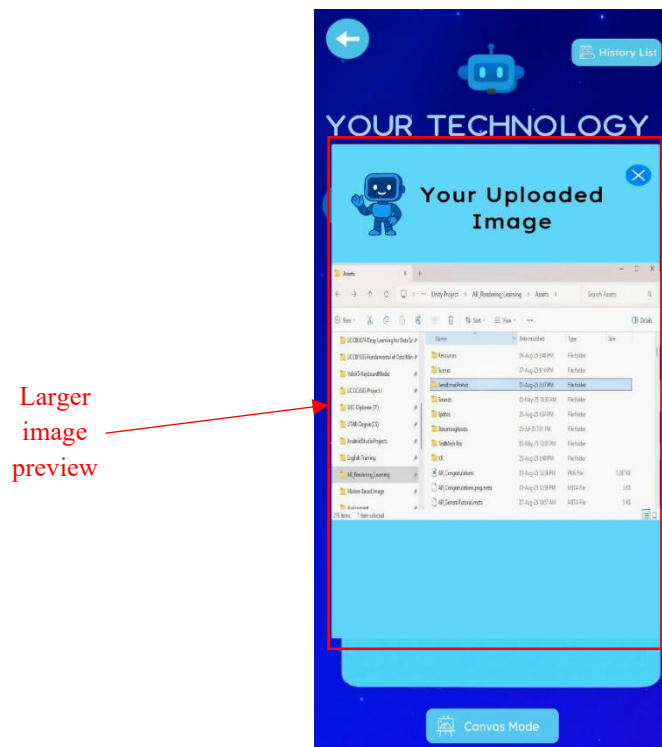


Figure 5.94 Clicking to Enlarge the Image

Figure 5.95 shows that after the user inserts an image, they can click the “Send” button to submit the question to OpenAI.



Figure 5.95 User Prompt with Question and Send Action

Figure 5.96 shows the response generated by OpenAI. The full step-by-step instructions are as follows:

- Step 1: Select the “Scenes” folder.
 - Click on the “Scenes” folder to select it.
- Step 2: Cut the folder.
 - Right-click the “Scenes” folder and choose Cut from the menu.
- Step 3: Navigate to the “English Training” folder.
 - Find and click on the “English Training” folder in the left panel.
- Step 4: Paste the folder.
 - Right-click inside the “English Training” folder and choose Paste from the menu.

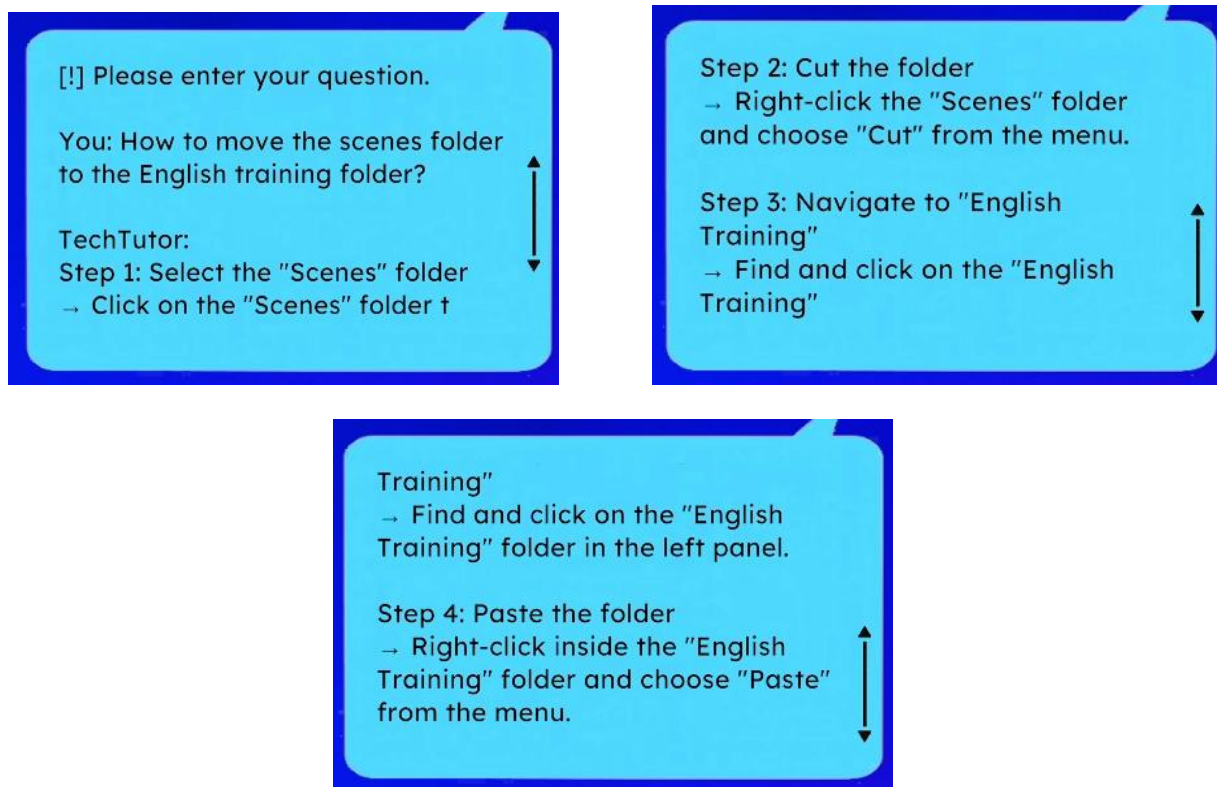


Figure 5.96 OpenAI Response

Figure 5.97 shows that after the user clicks the Canvas Mode button, the interface switches to ‘Canvas Mode’ in a new scene. The selected image is then sent to EasyOCR for processing. The user may need to wait for OCR to respond and highlight the detected text.

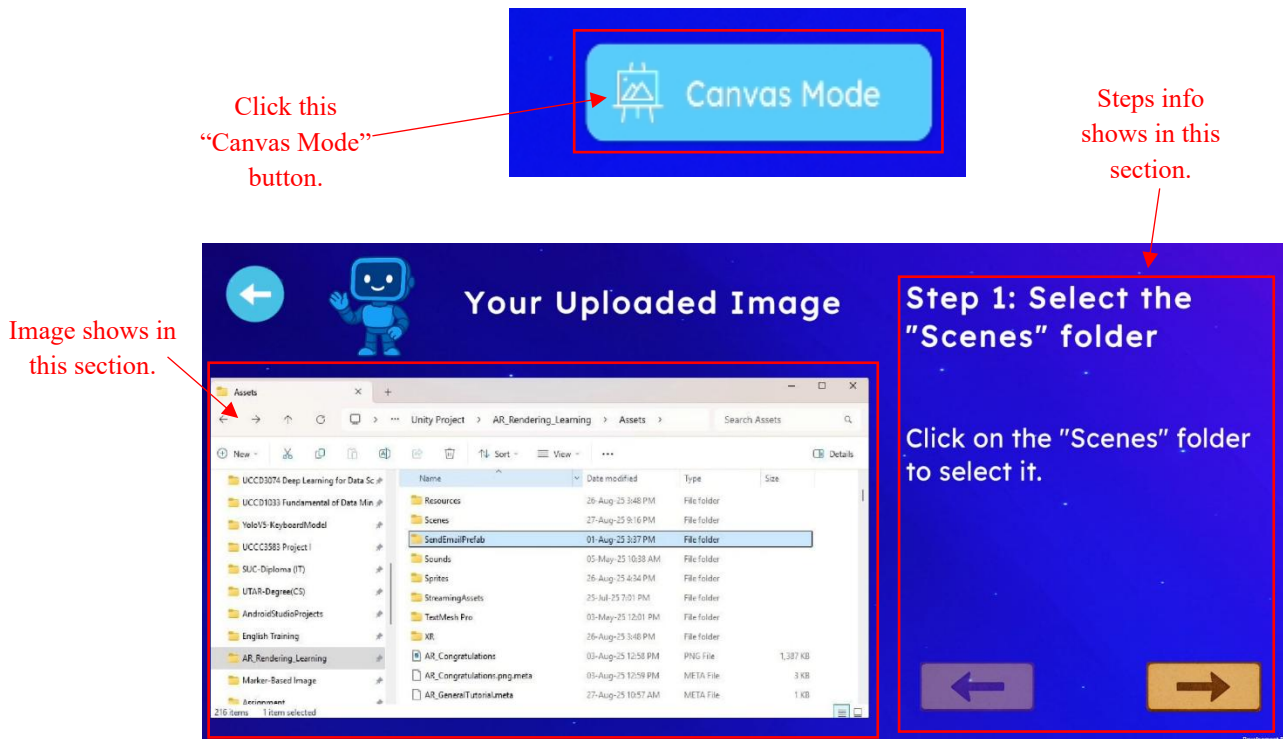


Figure 5.97 Canvas Mode

Figure 5.98 shows that after EasyOCR completes its detection, the recognized text is highlighted in red with reduced opacity. Above the highlighted text, a red arrow is displayed to provide a more visually obvious cue for the user to follow

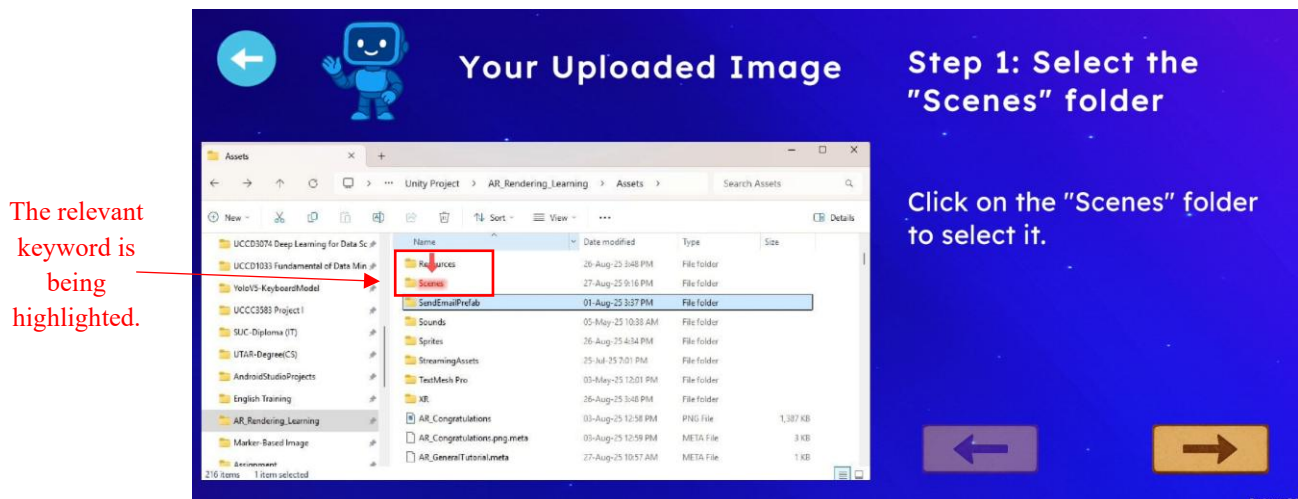



Figure 5.98 Highlight Text using EasyOCR

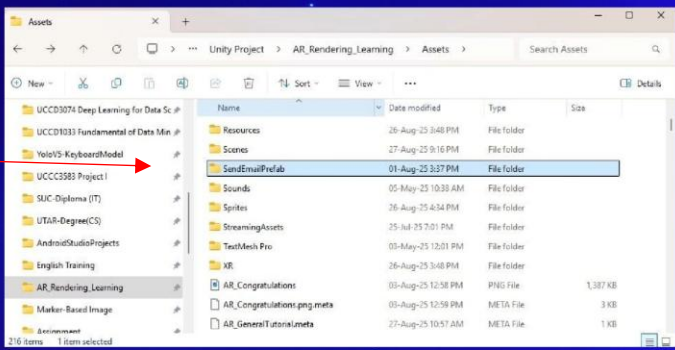
Figure 5.99 shows the remaining steps in 'Canvas Mode'. If a keyword is detected, it is highlighted in the corresponding step. If no keyword is found, no text is highlighted.

No relevant keyword, no keyword is highlighted.






Your Uploaded Image

Step 2: Cut the folder



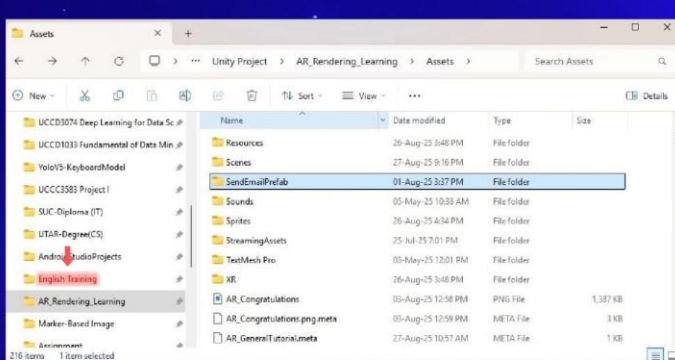
Right-click the "Scenes" folder and choose "Cut" from the menu.






Your Uploaded Image

Step 3: Navigate to "English Training"



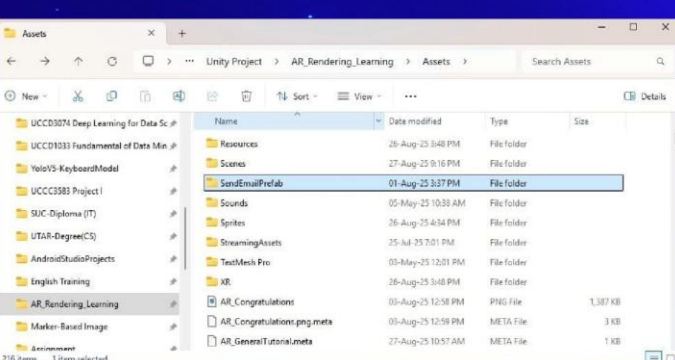
Find and click on the "English Training" folder in the left panel.



Your Uploaded Image

Step 4: Paste the folder



Right-click inside the "English Training" folder and choose "Paste" from the menu.


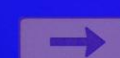



Figure 5.99 Remaining Steps with EasyOCR

Figure 5.100 shows that when the user clicks the "History List" button, the chat history list is displayed along with the associated image, timestamp, and a "View" button.

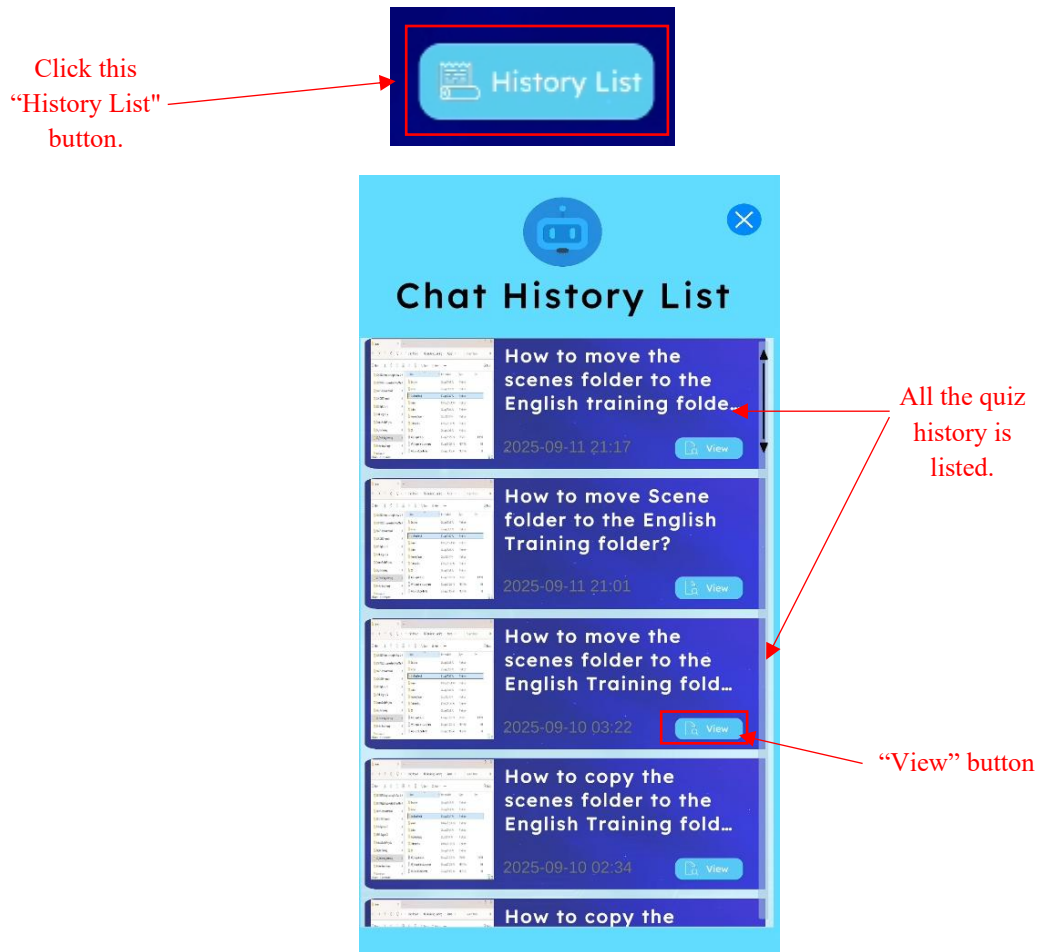


Figure 5.100 Chat History List

Figure 5.101 shows that when the user clicks the View button, the detail panel is displayed. This allows the user to review the response to the question that was previously asked.

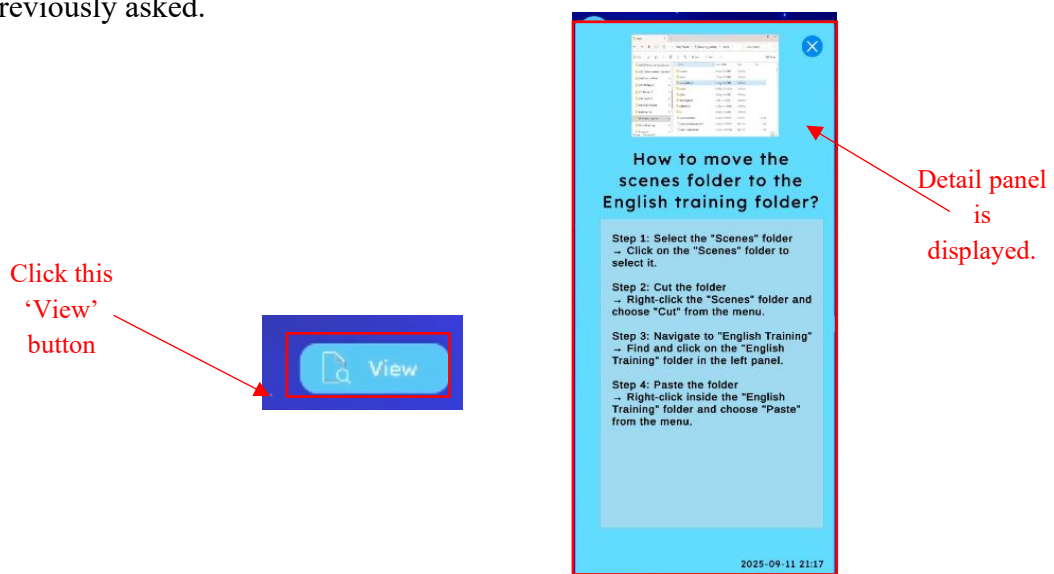


Figure 5.101 Chat History Detail Panel

Frequently Asked Questions (FAQs)

Figure 5.102 shows the list of Frequently Asked Questions (FAQs) provided for users, giving them a basic understanding of how to use the chatbot and EasyOCR. A total of six FAQ questions is available, along with two sample questions that serve as examples to guide users on how to ask questions to the chatbot.

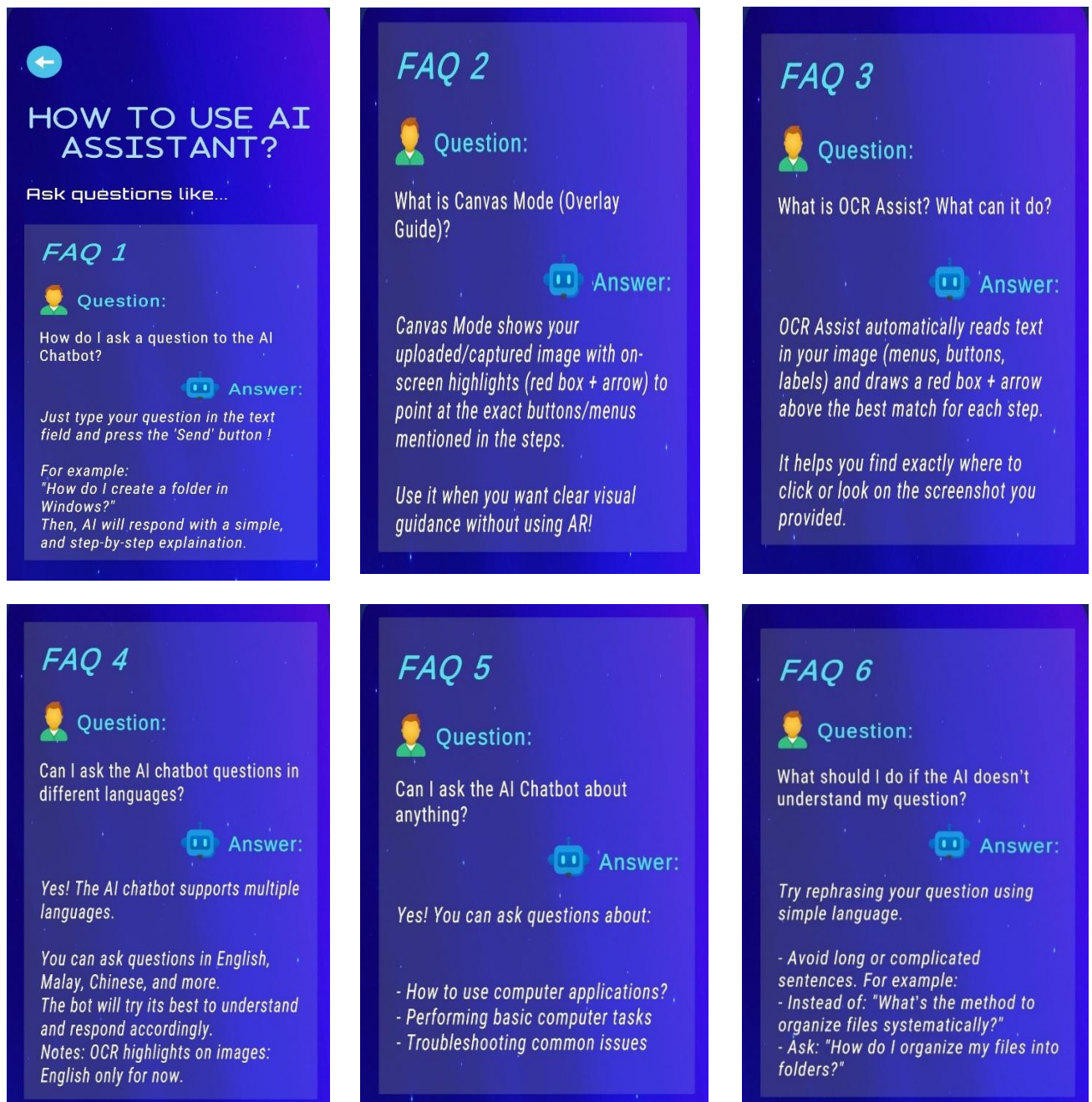


Figure 5.102 Lists of All FAQ Questions

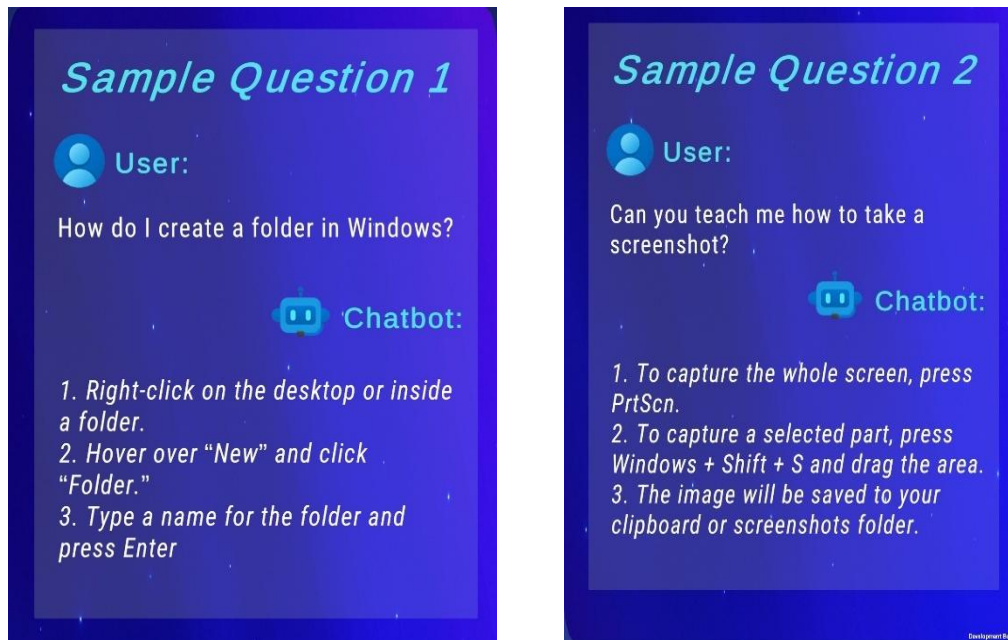


Figure 5.102 Lists of All FAQ Questions (Cont.)

5.5 Implementation Issues and Challenges

During the development and deployment of TechTutor, several implementation challenges were encountered. These challenges were largely related to system performance, device constraints, and real-world deployment conditions.

Performance Lag and Confidence Value in Detection

Initially, the system explored training a custom YOLOv5 model for detecting computer components. However, the detection confidence values typically ranged only between 0.5 and 0.7 during real-time use. This not only reduced the reliability of recognition but also caused noticeable lag on mobile devices, which disrupted the smoothness of the user experience. Therefore, after evaluating the trade-offs, the project transitioned to using OpenAI-based identification, which provided more consistent and user-friendly results.

High Batter Consumption and Device Heating

Running AR applications with continuous camera usage and plane detection is resource intensive. Extended use of the AR modules led to rapid batter depletion and noticeable heating of the mobile device. This challenge highlighted the limitations of mobile

hardware for long AR sessions, requiring the project to emphasize shorter, focused tutorials to balance usability with device constraints.

Misalignment of AR Models in Real-world Environments

Although AR objects and overlays appeared stable within the Unity environment, real-world testing revealed occasional misalignment. In some cases, 3D models would drift, remain stuck in incorrect positions, or fail to anchor properly due to variations in lighting, surface textures, or environmental conditions. This required extensive testing across multiple scenarios to refine placement logic and ensure overlays were as accurate and stable as possible.

Time-intensive Testing and Calibration

As AR interactions are highly dependent on the user's physical environment, extensive interactive testing was required. For example, issues that did not appear during Unity simulation scene often surfaced in the real-world usage, such as the poor detection in low-light conditions. Considerable time was needed to be spent on testing, calibration, and re-adjusting the placement and scaling of AR elements to improve the robustness of the system.

5.6 Concluding Remark

This project has successfully implemented the TechTutor application by combining AR and AI technologies into a single mobile learning platform. This implementation covered the integration of AR Learn, AR Guide, and AI Tutor modules, supported by Unity, Vuforia, ARCore, and OpenAI API. Each module was configured and tested to ensure that the intended features, such as AR-based tutorials, computer components identification, interactive models, AR quizzes, and AI-driven assistance were fully operational.

Despite challenges such as performance lag, high battery consumption, and occasional misalignment of AR overlays, the system was refined to achieve a stable and functional prototype. Overall, this project has demonstrated that the intended features have been fully implemented, and the TechTutor application is now operational as designed.

Chapter 6

System Evaluation and Discussion

This chapter summarize the evaluation of the TechTutor application based on the testing and analysis. It begins with the testing approach and setup, followed by the results obtained from the system implementation. The chapter then further discusses the challenges encountered during development, evaluates the extend to which the project objectives have been achieved, and concludes with final remarks on the overall system performance.

6.1 System Testing Approach

To evaluate the TechTutor application, **functional testing** was conducted using a **black-box approach**. This approach was selected because it focuses on validating whether each function of the system behaves according to the specified requirements, without the need to examine the internal implementation details. In practice, different inputs and user actions were provided to the system, and the outputs were observed to check if they matched the expected results.

The testing covered all three main modules of the system, which are **AR Learn**, **AR Guide**, and **AI Tutor**. Each module was tested based on its intended functionality, such as displaying AR-based tutorials, identifying computer components, generating quizzes, and responding to user queries.

The evaluation placed emphasis on the **correctness of system behaviour** and the **consistency of results** across multiple trials. The detailed functional test cases and their outcomes are presented in the following subsection.

6.2 Testing Setup and Result

The testing of the TechTutor application was performed on an Android mobile device (Mi 11 Lite, 8GB RAM, Android 13) to simulate a real usage environment. The device was used in indoor conditions under normal lighting, with stable internet connectivity for features requiring access to the OpenAI API. Each module of the system, **AR Learn**,

AR Guide, and AI Tutor was tested to verify that the functionalities worked as intended.

A **black-box functional testing** approach was adopted, where user actions were carried out and the system responses were observed. The expected outputs were compared against the actual results to determine whether the test cases passed or failed.

AR Learn Module

Table 6.1 AI Identify Test Case

Module 1 - AR Learn Module					
Test Case Module: AI Identify					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Identify Keyboard	Point Camera to 'Keyboard' object in real world and click 'Capture' button.	System display text "This is a keyboard" in the navigation section and plays TTS narration.	As Expected.	Pass
TC02	Identify Mouse	Point Camera to 'Mouse' object in real world and click 'Capture' button.	System displays text "This is a mouse" in the navigation section and plays TTS narration.	As Expected.	Pass
TC03	Identify Laptop	Point Camera to 'Laptop' object in real world and click 'Capture' button.	System displays text "This is a laptop" in the navigation section and plays TTS narration.	As Expected.	Pass
TC04	Identify Monitor	Point Camera to 'Monitor' object in real world and click 'Capture' button.	System displays text "This is a monitor" in the navigation section and plays TTS narration.	As Expected.	Pass

TC05	Object Capture and Analysis	Point camera to any object, click the 'Capture' button, and send request to OpenAI.	System displays "Analyzing ... Please wait ..." message, then shows the detected object text and plays TTS narration successfully.	As Expected.	Pass
TC06	Open More Info Panel	After an object is analyzed, click the 'More Info' button.	System opens the information panel with extended details of the detected object, and TTS narration begins automatically.	As Expected.	Pass
TC07	Replay Sound in Info Panel	Click the 'Sound Out' button in the info panel.	System replays the TTS narration for the detected object.	As Expected.	Pass
TC08	Open History List	Click the 'History List' button.	System displays the history list containing past identifications.	As Expected.	Pass
TC09	Scroll History List	Scroll through the history list panel.	System allows vertical scrolling, showing all previous history entries.	As Expected.	Pass
TC10	View History Detail	Click the 'View' button on a history entry.	System opens the detailed information panel of that entry and automatically plays TTS narration.	As Expected.	Pass
TC11	Back to AR Learn Main Screen	Click the 'Back arrow' button at the top-left corner.	System navigates back to the AR Learn main scene.	As Expected.	Pass

TC12	Close Info Panel	Click the 'Close' button (X) on the info panel.	The info panel closes.	As Expected.	Pass
TC13	Close History Detail Panel	Click the 'Close' button (X) on the history detail panel.	The detail panel closes.	As Expected.	Pass

Table 6.2 AR Models Test Case

Module 1 - AR Learn Module					
Test Case Module: AR Models					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Plane Detection	Point camera to a surface with textures.	System detects and shows highlighted plane on the surface.	As Expected.	Pass
TC02	Lock Plane	Click on detected plane.	Selected plane is locked, other planes disappear, and model selection panel appears.	As Expected.	Pass
TC03	Model Selection: Keyboard	From selection panel, choose "Keyboard".	Preview model of "Keyboard" appears in the middle of locked plane.	As Expected.	Pass
TC04	Model Selection: Mouse	Choose "Mouse" model.	Preview model of "Mouse" appears in middle of plane.	As Expected.	Pass
TC05	Model Selection: Monitor	Choose "Monitor" model.	Preview model of "Monitor" appears in middle of plane.	As Expected.	Pass

TC06	Model Selection: Laptop	Choose “Laptop” model.	Preview model of “Laptop” appears in middle of plane.	As Expected.	Pass
TC07	Model Selection: Speaker	Choose “Speaker” model.	Preview model of “Speaker” appears in middle of plane.	As Expected.	Pass
TC08	STT: Keyboard	User says, “Keyboard tutorial”.	Existing model disappears, “Keyboard” preview model spawns in middle of plane.	As Expected.	Pass
TC09	STT: Mouse	User says, “Mouse tutorial”.	“Mouse” preview model spawns in middle of plane.	As Expected.	Pass
TC10	STT: Monitor	User says, “Monitor tutorial”.	“Monitor” preview model spawns in middle of plane.	As Expected.	Pass
TC11	STT: Laptop	User says, “Laptop tutorial”.	“Laptop” preview model spawns in middle of plane.	As Expected.	Pass
TC12	STT: Speaker	User says, “Speaker tutorial”.	“Speaker” preview model spawns in middle of plane.	As Expected.	Pass
TC13	Move Model Before Confirm	Drag preview model before pressing Confirm.	Model can be moved freely within plane.	As Expected.	Pass
TC14	Scale Model	Use pinch gesture to scale preview model.	Model scales smoothly without distortion.	As Expected.	Pass
TC15	Confirm Placement	Click “Confirm Model” button.	Model fixed in place at selected position.	As Expected.	Pass
TC16	Click Model to Show Info Panel	Tap on confirmed model.	Info panel appears with details and TTS narration.	As Expected.	Pass

TC17	Explore More (Show Labels)	Click “Explore More” button in info panel.	Info panel closes and part labels are overlaid on the model.	As Expected.	Pass
TC18	Click Label for Detail	Tap a part label on the model.	Detailed info panel for that part appears.	As Expected.	Pass
TC19	Close Detail Panel	Click Close (X) button on detail panel.	Detail panel closes, returning to labelled model.	As Expected.	Pass
TC20	Rotate Laptop Model	Rotate Laptop model using gesture.	Laptop rotates smoothly on plane.	As Expected.	Pass
TC21	Rotate Monitor Model	Rotate Monitor model using gesture.	Monitor rotates smoothly on plane.	As Expected.	Pass
TC22	Reset Position	Click “Reset” button.	Model resets to original position and rotation.	As Expected.	Pass
TC23	Move Model with Buttons	Use “Up”, “Down”, “Left”, “Right” buttons.	Model moves correctly in specified direction.	As Expected.	Pass

Table 6.3 AR Quiz Test Case

Module 1 - AR Learn Module					
Test Case Module: AR Quiz					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Marker detection	Point camera to marker image.	Model, quiz panel, and buttons are spawned.	As Expected.	Pass
TC02	Play model sound + update label	Select a model (Keyboard, Mouse,	Corresponding sound plays. Label	As Expected.	Pass

		Speaker, Laptop, Monitor).	updates to “Model Selection: ____”.		
TC03	Switch model with arrows	Click “Left” / “Right” arrows.	Different models are displayed.	As Expected.	Pass
TC04	Switch model with hands (swipe left and right)	Swipe left / right on screen.	Model changes according to swipe direction.	As Expected.	Pass
TC05	Choose model for quiz	Click “Choose Model” button.	Quiz panel appears with AI-generated question. Model is locked (can’t be switch) and button states update, which “Choose Model” button become unactive. “Unchoose Model” button become active.	As Expected.	Pass
TC06	Answer quiz question	Select one of 4 answers.	Correct Answer → shows “Correct” label and correct sound effect. Wrong Answer → shows “Try again” label and error sound.	As Expected.	Pass
TC07	Interactive quiz with keyboard model	For Keyboard model Q1, click key (e.g., Windows, Caps Lock).	System detects key input and evaluates answer.	As Expected.	Pass
TC08	Generate new quiz	Click “Generate Quiz” button.	Request sent to OpenAI and new	As Expected.	Pass

			quiz is generated and shown in the quiz panel.		
TC09	Display new question	After AI Question generation completes.	Quiz panel shows new question and sound narration plays automatically.	As Expected.	Pass
TC10	Unchoose model	Click “Unchoose Model” button.	Model unlocked; and “Choose Model” button re-activated. Model now can be switched.	As Expected.	Pass
TC11	Model manipulation (rotate/scale)	Use gestures on model.	Model can be rotated and scaled smoothly.	As Expected.	Pass
TC12	Reset model	Click Reset button.	Model position and scale reset to default.	As Expected.	Pass
TC13	Open history list	Click History List button.	Quiz history list panel displayed and it is scrollable.	As Expected.	Pass
TC14	View quiz detail	Click entry in history list.	Quiz detail panel opens with full question/answer info.	As Expected.	Pass
TC15	Back to AR Learn main	Click Back (arrow) button	System navigates back to AR Learn main screen.	As Expected.	Pass

AR Guide Module

Table 6.4 Simple Calculator Test Case

Module 2 - AR Guide Module					
Test Case Module: Simple Calculator					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Marker-based Detection	Point the camera at the calculator marker.	The tutorial panel is spawned automatically when the marker is detected.	As Expected.	Pass
TC02	Tutorial Panel Alignment	Detect the calculator marker in real-world space.	The tutorial step panel spawns in the correct position above the marker with no misalignment or drifting.	As Expected.	Pass
TC03	Next Button Navigation	Click the “Next” button during the tutorial.	The tutorial proceeds to the next instructional step, and the panel updates accordingly.	As Expected.	Pass
TC04	Previous Button Navigation	Click the “Prev” button during the tutorial.	The tutorial returns to the previous step, with the content updated correctly.	As Expected.	Pass
TC05	Tutorial Labels	Observe the labels displayed in each tutorial step.	All labels are clearly displayed and properly aligned with the step panel, ensuring	As Expected.	Pass

			they can be seen without obstruction.		
TC06	STT Navigation	Say “Next” or “Prev” using voice commands.	The tutorial proceeds to the next or previous step based on the spoken command.	As Expected.	Pass
TC07	Back Button Navigation	Click the “Back” button at any stage of the tutorial.	The tutorial closes and the system returns to the AR Guide main screen without error.	As Expected.	Pass

Table 6.5 Microsoft Word Test Case

Module 2 - AR Guide Module					
Test Case Module: Microsoft Word					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Plane Detection	Point camera to a surface with textures.	System detects and highlights the plane surface.	As Expected.	Pass
TC02	Lock Plane and Spawn Model	Click to lock detected plane.	Other planes disappear. Desk Simulation model spawns with tutorial panel and TTS narration plays automatically.	As Expected.	Pass
TC03	STT Navigation	Say “Next” or “Prev” using STT button.	Tutorial proceeds to the next or previous step according to voice command.	As Expected.	Pass

TC04	Move Desk Simulation Model	Drag Desk Simulation model within the plane.	Model moves smoothly within plane boundaries.	As Expected.	Pass
TC05	Key Highlight Labels	Observe highlighted keys during tutorial steps.	Key highlight labels are clearly shown on the Desk Simulation model.	As Expected.	Pass
TC06	Info Canvas Controls	Click the info canvas.	Canvas control panel appears. Arrows (top, left, bottom, up) are clickable for movement.	As Expected.	Pass
TC07	Reset Info Canvas Position	Click the reset position button.	Info canvas returns to default position.	As Expected.	Pass
TC08	Model Control Panel	Click Desk Simulation model.	Model control panel appears. User can move model (top, left, bottom, right) and scale larger or smaller.	As Expected.	Pass
TC09	Reset Model Position and Scale	Click reset buttons in model control panel.	Model resets to default position and scale.	As Expected.	Pass
TC10	Next and Previous Buttons	Click Next or Prev button on tutorial panel.	Tutorial navigates correctly to next or previous step.	As Expected.	Pass
TC11	Trigger Canvas Mode	Switch to Canvas Mode by clicking the 'Canvas' mode button.	Desk Simulation model disappears. Only canvas info panel appears in front of camera when clicked.	As Expected.	Pass
TC12	Reset Canvas Panel	Click Reset Panel button in Canvas Mode.	Info panel resets and can be	As Expected.	Pass

			respawned in front of camera.		
TC13	Synchronization Between Modes	Ensure canvas panel syncs with Desk Simulation mode.	Both panels display the same tutorial step consistently.	As Expected.	Pass
TC14	Return to Desk Simulation Mode	Switch back from Canvas Mode to Desk Simulation mode.	Desk Simulation model reappears in the same locked plane position.	As Expected.	Pass
TC15	Monitor Screen Simulation	Navigate through each tutorial step.	The monitor model updates its screen display to match the current step, simulating the correct action or interface for that step.	As Expected.	Pass

The functional test cases for **Send Email** and **Snipping Tool** tutorials are identical to those of **Microsoft Word** tutorial, as they share the same AR Guide interaction flow and system functions. The only difference lies in the tutorial step content, monitor screen, which does not affect the test procedure.

Table 6.6 Send Email Test Case

Module 2 - AR Guide Module					
Test Case Module: Send Email					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Plane Detection	Point camera to a surface with textures.	System detects and highlights the plane surface.	As Expected.	Pass
TC02	Lock Plane and Spawn Model	Click to lock detected plane.	Other planes disappear. Desk	As Expected.	Pass

			Simulation model spawns with tutorial panel and TTS narration plays automatically.		
TC03	STT Navigation	Say “Next” or “Prev” using STT button.	Tutorial proceeds to the next or previous step according to voice command.	As Expected.	Pass
TC04	Move Desk Simulation Model	Drag Desk Simulation model within the plane.	Model moves smoothly within plane boundaries.	As Expected.	Pass
TC05	Key Highlight Labels	Observe highlighted keys during tutorial steps.	Key highlight labels are clearly shown on the Desk Simulation model.	As Expected.	Pass
TC06	Info Canvas Controls	Click the info canvas.	Canvas control panel appears. Arrows (top, left, bottom, up) are clickable for movement.	As Expected.	Pass
TC07	Reset Info Canvas Position	Click the reset position button.	Info canvas returns to default position.	As Expected.	Pass
TC08	Model Control Panel	Click Desk Simulation model.	Model control panel appears. User can move model (top, left, bottom, right) and scale larger or smaller.	As Expected.	Pass
TC09	Reset Model Position and Scale	Click reset buttons in model control panel.	Model resets to default position and scale.	As Expected.	Pass
TC10	Next and Previous Buttons	Click Next or Prev button on tutorial panel.	Tutorial navigates correctly to next or previous step.	As Expected.	Pass

TC11	Trigger Canvas Mode	Switch to Canvas Mode by clicking the 'Canvas' mode button.	Desk Simulation model disappears. Only canvas info panel appears in front of camera when clicked.	As Expected.	Pass
TC12	Reset Canvas Panel	Click Reset Panel button in Canvas Mode.	Info panel resets and can be respawned in front of camera.	As Expected.	Pass
TC13	Synchronization Between Modes	Ensure canvas panel syncs with Desk Simulation mode.	Both panels display the same tutorial step consistently.	As Expected.	Pass
TC14	Return to Desk Simulation Mode	Switch back from Canvas Mode to Desk Simulation mode.	Desk Simulation model reappears in the same locked plane position.	As Expected.	Pass
TC15	Monitor Screen Simulation	Navigate through each tutorial step.	The monitor model updates its screen display to match the current step, simulating the correct action or interface for that step.	As Expected.	Pass

Table 6.7 Snipping Tool Test Case

Module 2 - AR Guide Module					
Test Case Module: Snipping Tool					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Plane Detection	Point camera to a surface with textures.	System detects and highlights the plane surface.	As Expected.	Pass
TC02	Lock Plane and Spawn Model	Click to lock detected plane.	Other planes disappear. Desk Simulation model spawns with tutorial panel and TTS narration plays automatically.	As Expected.	Pass
TC03	STT Navigation	Say “Next” or “Prev” using STT button.	Tutorial proceeds to the next or previous step according to voice command.	As Expected.	Pass
TC04	Move Desk Simulation Model	Drag Desk Simulation model within the plane.	Model moves smoothly within plane boundaries.	As Expected.	Pass
TC05	Key Highlight Labels	Observe highlighted keys during tutorial steps.	Key highlight labels are clearly shown on the Desk Simulation model.	As Expected.	Pass
TC06	Info Canvas Controls	Click the info canvas.	Canvas control panel appears. Arrows (top, left, bottom, up) are clickable for movement.	As Expected.	Pass
TC07	Reset Info Canvas Position	Click the reset position button.	Info canvas returns to default position.	As Expected.	Pass

TC08	Model Control Panel	Click Desk Simulation model.	Model control panel appears. User can move model (top, left, bottom, right) and scale larger or smaller.	As Expected.	Pass
TC09	Reset Model Position and Scale	Click reset buttons in model control panel.	Model resets to default position and scale.	As Expected.	Pass
TC10	Next and Previous Buttons	Click Next or Prev button on tutorial panel.	Tutorial navigates correctly to next or previous step.	As Expected.	Pass
TC11	Trigger Canvas Mode	Switch to Canvas Mode by clicking the 'Canvas' mode button.	Desk Simulation model disappears. Only canvas info panel appears in front of camera when clicked.	As Expected.	Pass
TC12	Reset Canvas Panel	Click Reset Panel button in Canvas Mode.	Info panel resets and can be respawned in front of camera.	As Expected.	Pass
TC13	Synchronization Between Modes	Ensure canvas panel syncs with Desk Simulation mode.	Both panels display the same tutorial step consistently.	As Expected.	Pass
TC14	Return to Desk Simulation Mode	Switch back from Canvas Mode to Desk Simulation mode.	Desk Simulation model reappears in the same locked plane position.	As Expected.	Pass

AI Tutor Module

Table 6.8 AI Assistance Test Case

Module 3 – AI Tutor Module					
Test Case Module: Snipping Tool					
Test Case ID	Test Case Description	Input / Action	Expected Output	Actual Results	Result
TC01	Type Question	Click on the user asking section and type a question using device keyboard.	Text can be entered into the input field.	As Expected.	Pass
TC02	Upload from Gallery	Click “Gallery” button to select an image from device storage.	System allows image selection from gallery.	As Expected.	Pass
TC03	Take Photo	Click “Take Photo” button to capture an image.	Camera opens and user can take a photo in real time.	As Expected.	Pass
TC04	Send Text Query	Type a text-only question and click “Send”.	OpenAI responds with a concise 3–5 step explanation.	As Expected.	Pass
TC05	Display Image Preview	Upload an image via Gallery or Take Photo.	Preview of the uploaded image appears below the “Gallery” button.	As Expected.	Pass
TC06	View Larger Image	Click on the image preview.	Image opens in larger view.	As Expected.	Pass
TC07	Delete Preview Image	Click “X” on the image preview.	Preview image is removed from the panel.	As Expected.	Pass
TC08	Switch to Canvas Mode	Click “Canvas Mode” button.	Application switches to a new scene (Canvas Mode).	As Expected.	Pass

TC09	Keyword Detection in Canvas Mode	Upload an image and switch to Canvas Mode.	Text in the image is detected, and relevant keywords are highlighted.	As Expected.	Pass
TC10	Correct Keyword Highlight Position	Observe highlighted keywords.	Highlighted keywords appear at the correct positions in the image.	As Expected.	Pass
TC11	Next Step Navigation	Click “Next” (→) button in Canvas Mode.	System navigates to the next step.	As Expected.	Pass
TC12	Previous Step Navigation	Click “Prev” (←) button in Canvas Mode.	System navigates back to the previous step.	As Expected.	Pass
TC13	Back to AI Assistance	Click “Back” (arrow) button in Canvas Mode.	Application returns to AI Assistance main scene.	As Expected.	Pass
TC14	Open History List	Click “History List” button.	Chat history list panel is displayed.	As Expected.	Pass
TC15	Scroll History List	Scroll within the history list panel.	History entries can be scrolled vertically.	As Expected.	Pass
TC16	View History Detail	Click “View” button on a history entry.	Detailed panel of the selected entry is displayed.	As Expected.	Pass
TC17	Close History Panel	Click the “Close” button on the chat history detail panel.	Panel closes and returns to history list.	As Expected.	Pass

6.3 Project Challenges

During the evaluation stage of TechTutor application, several challenges were encountered that affected the testing process and overall assessment of the system. Unlike the implementation issues discussed in Chapter 5, these challenges were primarily related to the conditions under which the system was tested and the behaviour of the application in real-world usage.

One of the major challenges was the **dependence of AR features on the testing environment**. Lighting conditions, surface textures, and background clutter had an obvious impact on plane detection and marker stability. In well-lit areas, the system performed smoothly, but in uneven lighting, AR overlays occasionally drifted or misaligned. Similarly, reflective or glossy surfaces sometimes interfered with plane detection, requiring multiple attempts to achieve stable placement.

Another challenge was the **variation in device performance**. The system was primarily tested on a mid-range device (Mi 11 Lite), which managed to run the application with acceptable performance. However, continuous AR usage led to device heating and faster battery drain, limiting the duration of testing sessions. This raised concerns about how the application might perform on lower-end devices with less processing power and memory.

User interaction also presented difficulties, particularly with the Speech-to-Text (STT) feature. While STT worked well in quiet environments, noisy surroundings reduced its accuracy, requiring users to repeat commands. Another challenge was the **limited project timeline**. The entire system, which involved integrating AR features, AI functionalities, and multiple interactive modules, had to be designed, developed, and tested within approximately nine months. This relatively short duration placed constraints on extensive experimentation and further optimization. As a result, the focus was placed on ensuring core functionalities were fully implemented and stable, while some advanced features and refinements were left as potential areas for future work.

Finally, the **AI response consistency** posed another challenge. Although OpenAI generally produced relevant and accurate answers, the phrasing of responses varied across different trials for the same query. This made it more difficult to evaluate the reliability of AI responses in a standardized way during testing.

Overall, these challenges highlighted the sensitivity of AR and AI systems to environmental conditions, device capabilities, and user interactions. While they did not prevent successful testing, they provided valuable insights into the limitations and potential improvements for future iterations of the application.

6.4 Objectives Evaluation

This subsection evaluates the extent to which the objectives stated in Chapter 1 were achieved. Each objective is assessed based on the implemented features and tested functionalities of the TechTutor application.

- **To develop an AR-based Learning Modules, AR Learn and AR Guide.**

This objective is achieved. Both AR Learn and AR Guide modules were successfully implemented, allowing users to recognize computer components, follow interactive tutorials, and attempt quizzes for self-assessment. For example, the AR Learn module consists of three submodules: AI Identify, AR Models, and AR Quiz. These submodules enable users to identify real-world computer components through AI, interact with 3D virtual models, and answer automatically generated quizzes in marker-based AR. On the other hand, the AR Guide module consists of four submodules: Microsoft Word, Snipping Tool, Send Email, and Calculator tutorials. These provide step-by-step instructions with synchronized narration, visual highlights, and the option to switch between 3D AR mode and Canvas mode for flexibility. Therefore, it can be concluded that this objective has been fully achieved.

- **To develop an AI Tutor Module.**

The objective is achieved. The AI Tutor module was successfully implemented to provide real-time support and personalized guidance through both text and image-based queries. Users are able to type questions, upload images from the gallery, or capture photos in real time to seek assistance. With the integration of EasyOCR, keywords from screenshots can be detected and highlighted in Canvas Mode, guiding the users to the relevant part of the interface. Additional features such as history list and image preview were also included to improve the usability of the system.

Hence, the AI Tutor module has met the intended objective of providing contextual, on-demand support.

- **To implement adaptive learning features.**

This objective is achieved. The system incorporated adaptive features that enhance inclusivity and support different learning speeds and styles. For example, narration review is available through Text-to-Speech (TTS), while Speech-to-Text (STT) enables hands-free navigation for tutorials and quizzes. The quiz history list and chat history in AI Tutor allow users to revisit past learning sessions at their own pace. These features make the application more user-friendly for diverse groups, particularly older adults and beginners who may require additional support. Therefore, it can be concluded that the adaptive learning objective was successfully achieved.

6.5 Concluding Remark

This chapter has presented the evaluation of the TechTutor application through functional testing, project challenges, and objectives assessment. Functional test cases were carried out for each module, AR Learn, AR Guide, and AI Tutor, to verify that the implemented features operated according to the specified requirements. The results confirmed that the application was stable, and all intended functionalities were successfully achieved.

Several challenges were identified during the evaluation stage, including environmental dependence of AR features, device performance limitations, Speech-to-Text (STT) accuracy in noisy surroundings, AI response variation, and the constraints of a nine-month project timeline. These challenges, however, did not prevent the system from being tested successfully, and they provided important insights into areas for future improvement.

Finally, the objectives stated at the beginning of the project were revisited and evaluated. All objectives in the development of AR-based learning modules, the AI Tutor module, and adaptive learning features were successfully achieved. This demonstrates that the system is able to provide an interactive, AR-integrated, and AI-assisted platform for learning essential computer and technology skills.

Chapter 7

Conclusion and Recommendations

This chapter concludes the project by summarizing the overall work done, highlighting the key outcomes achieved, and reflecting on how the objectives were met. It also presents recommendations for future enhancement and directions to further improve and expand the TechTutor application.

7.1 Conclusion

This project set out to design and develop TechTutor, an Augmented Reality (AR)-integrated and AI assisted mobile applications which aimed at improving digital literacy, particularly among older adults and beginners. The system was developed to address the challenges identified in the problem statement, which are digital literacy gap, difficulties in learning basic computer skills, and the limitations of existing passive learning methods. By combining immersive AR experiences with adaptive AI-driven support, TechTutor provides an intuitive, engaging, and accessible platform for technology learning.

The system was successfully implemented with 3 main components, which are **AR Learn**, **AR Guide**, and **AI Tutor**. The AR Learn module enables users to recognize computer components, interact with 3D models, and take AI-generated quizzes, supported by Speech-to-Text (STT) and Text-to-Speech (TTS) functionalities. The AR Guide module provides step-by-step tutorials for essential tasks such as Microsoft Word, sending emails, snipping tools and also operating calculators, with synchronized narration and visual cues in both AR and Canvas modes. The AI Tutor module offers real-time support through a chatbot, image-based queries, and OCR-powered keyword highlighting, ensuring context-aware guidance beyond the tutorials.

Through these features, the project achieved its objectives of creating an interactive, adaptive and inclusive learning platform. The system demonstrated its ability to support different learning speeds, provide hands-on guidance, and enhance user engagement,

particularly for older adults with limited exposure to the latest technology. Furthermore, the application shows potential for classroom use, where it can complement traditional teaching by offering interactive, experiential learning opportunities, especially for learning technology-related skills.

In conclusion, TechTutor successfully addressed the gaps in current digital literacy tools by integrating AR, AI, and adaptive features into a unified platform. The project contributes to bridging the digital divide by making computer learning more approachable, less overwhelming, and better suited to diverse user needs.

7.2 Recommendations

While TechTutor has achieved its objectives, there remain opportunities to further enhance its functionality, scalability, and adaptability. The following recommendations are proposed for future development:

1. Expand Tutorial Coverage

Additional tutorials can be developed to cover more computer-related tasks, such as Microsoft Excel, PowerPoint, and online communication tools. This would broaden the scope of the application and support a wider range of digital literacy.

2. AI-Adapted Tutorial Generation

The integration of fully AI-adapted tutorials could allow dynamic generation of step-by-step instructions without the need for manual documentation. However, this approach would require extensive training on computer applications to ensure that the AI can accurately detect interface elements, understand workflows, and adapt guidance to user progress. Although not yet feasible with current technology, future advancements may enable scalable and personalized tutorial generation.

3. AI-Generated Visual Content

Recently, AI-generated images are not yet mature enough to consistently produce accurate and step-consistent computer interface screens. Nevertheless, as generative technologies advance, they may be incorporated to automatically generate realistic and context-aware visuals for tutorials, reducing reliance on pre-prepared assets and enabling fully AI-driven AR tutorial content.

4. Enhanced Offline Functionality

As the current system relies on internet connectivity for OpenAI services, future development should consider integrating offline support through lightweight local models or pre-downloaded AI responses. This would ensure accessibility in environments with limited or unstable connectivity.

5. Improved AR Stability and Accessibility

Optimizations can be introduced to improve AR performance on lower-end devices, ensuring stable tracking and smoother operation. Accessibility features such as adjustable text size, high-contrast display options, and simplified navigation should also be incorporated to better support older adults and users with special needs.

6. Multi-Language Support and Analytics

To broaden accessibility, the system can be extended to support multiple languages. Furthermore, analytics features such as learning progress tracking and personalized feedback could provide users with insights into their performance, encouraging continuous improvement and self-paced learning.

7.3 Closing Remark

In summary, TechTutor demonstrates the potential of integrating AR and AI technologies to enhance the digital literacy and support diverse learners. With further improvements and advancements in AI-driven tutorial generation and content creation, the system can evolve into a fully adaptive and scalable platform, which can contribute more meaningfully to bridging the digital divide in both individual and classroom learning contexts.

REFERENCES

- [1] A. Smith, “Older Adults and Technology Use,” Pew Research Center: Internet, Science & Tech. <https://www.pewresearch.org/internet/2014/04/03/older-adults-and-technology-use/> (accessed: Aug. 6, 2024)
- [2] L. Wang, P.-L. P. Rau, and G. Salvendy, “Older Adults’ Acceptance of Information Technology,” *Educational Gerontology*, vol. 37, no. 12, pp. 1081–1099, Dec. 2011, doi: <https://doi.org/10.1080/03601277.2010.500588>.
- [3] O. Campbell, “Designing For The Elderly: Ways Older People Use Digital Technology Differently — Smashing Magazine,” Smashing Magazine. <https://www.smashingmagazine.com/2015/02/designing-digital-technology-for-the-elderly/> (accessed: Aug. 6, 2024)
- [4] Y. Sorunlari and Dergisi, “A qualitative study on skills of elders to use digital technology products from digital divide perspective,” *YSAD) Elderly Issues Research Journal (EIRJ)*, vol. 13, no. 2, pp. 107–122, 2020, doi: <https://doi.org/10.46414/yasad.788412>.
- [5] N. Charness, C. L. Kelley, E. A. Bosman, and M. Mottram, “Word-processing training and retraining: Effects of adult age, experience, and interface.,” *Psychology and Aging*, vol. 16, no. 1, pp. 110–127, 2001, doi: <https://doi.org/10.1037/0882-7974.16.1.110>.
- [6] N. Charness and W. R. Boot, “Aging and information technology use: potential and barriers,” *Current Directions in Psychological Science*, vol. 18, no. 5, pp. 253–258, Oct. 2009, doi: <https://doi.org/10.1111/j.1467-8721.2009.01647.x>.
- [7] S. Almeida-Ferreira, A. I. Veloso, and O. Mealha, “Older Adults and Email Use: The challenges facing interface co-design,” *Networking Knowledge: Journal of the MeCCSA Postgraduate Network*, vol. 10, no. 1, pp. 44–63, Mar. 2017, doi: <https://doi.org/10.31165/nk.2017.101.496>.
- [8] T. N. Friemel, “The digital divide has grown old: Determinants of a digital divide among seniors,” *New Media & Society*, vol. 18, no. 2, pp. 313–331, Jun. 2016, doi: <https://doi.org/10.1177/1461444814538648>.

- [9] M.-B. Ibáñez and C. Delgado-Kloos, “Augmented reality for STEM learning: A systematic review,” *Computers & Education*, vol. 123, pp. 109–123, Aug. 2018, doi: <https://doi.org/10.1016/j.compedu.2018.05.002>.
- [10] M. Akçayır and G. Akçayır, “Advantages and challenges associated with augmented reality for education: A systematic review of the literature,” *Educational Research Review*, vol. 20, no. 1, pp. 1–11, Feb. 2017, doi: <https://doi.org/10.1016/j.edurev.2016.11.002>.
- [11] R. Winkler and M. Soellner, “Unleashing the Potential of Chatbots in Education: A State-Of-The-Art Analysis,” *Academy of Management Proceedings*, vol. 2018, no. 1, p. 15903, Aug. 2018, doi: <https://doi.org/10.5465/ambpp.2018.15903abstract>.
- [12] American Library Association, “Digital Literacy,” Welcome to ALA’s Literacy Clearinghouse. <https://literacy.ala.org/digital-literacy/> (accessed: Apr. 15, 2025)
- [13] N. W. A. Majid et al., “The Effectiveness of Using Assemblr Edu Learning Media to Help Student Learning at School,” *Jurnal Penelitian Pendidikan IPA*, vol. 9, no. 11, pp. 9243–9249, Nov. 2023, doi: <https://doi.org/10.29303/jppipa.v9i11.5388>.
- [14] Apple, “ARKit 3 - Augmented Reality - Apple Developer,” Apple Developer. <https://developer.apple.com/augmented-reality/arkit/> (accessed: Apr. 20, 2025)
- [15] R. Palmarini, J. A. Erkoyuncu, R. Roy, and H. Torabmostaedi, “A systematic review of augmented reality applications in maintenance,” *Robotics and Computer-Integrated Manufacturing*, vol. 49, pp. 215–228, Feb. 2018, doi: <https://doi.org/10.1016/j.rcim.2017.06.002>.
- [16] F. Arena, M. Collotta, G. Pau, and F. Termine, “An Overview of Augmented Reality,” *Computers*, vol. 11, no. 2, p. 28, Feb. 2022, doi: <https://doi.org/10.3390/computers11020028>.
- [17] A. Benassi et al., “Augmented reality and intelligent systems in Industry 4.0,” *HAL Archives Ouvertes*. <https://hal.archives-ouvertes.fr/hal-03018976/> (accessed: Aug. 19, 2024)

- [18] A. Z. A. Halim, “Applications of augmented reality for inspection and maintenance process in automotive industry,” *Journal of Fundamental and Applied Sciences*, vol. 10, no. 3S, pp. 412–421, May 2018, doi: <https://doi.org/10.4314/jfas.v10i3S.35>.
- [19] Z. Oufqir, A. El Abderrahmani, and K. Satori, “ARKit and ARCore in serve to augmented reality,” *IEEE Xplore*, Jun. 01, 2020. <https://ieeexplore.ieee.org/abstract/document/9204243> [12]
- [20] P. Nowacki and M. Woda, “Capabilities of ARCore and ARKit Platforms for AR/VR Applications,” *Advances in Intelligent Systems and Computing*, pp. 358–370, May 2019, doi: https://doi.org/10.1007/978-3-030-19501-4_36. [13]
- [21] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, “Mobile Augmented Reality Survey: From Where We Are to Where We Go,” *IEEE Access*, vol. 5, pp. 6917–6950, 2017, doi: <https://doi.org/10.1109/access.2017.2698164>. [14]
- [22] “AR Foundation | AR Foundation | 6.1.0,” Unity3d.com. <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.1/manual/index.html> (accessed: Apr. 23, 2025) [15]
- [23] P. Rawat, “Augmented Reality Manual - An AR Instructional User Guide Revolutionizing How We Learn & Work,” Argenie.ai. <https://www.argenie.ai/blogs/augmented-reality-manual---an-ar-instructional-user-guide-revolutionizing-how-we-learn-work> (accessed Aug. 29, 2024).
- [24] “How to Develop an AR/MR Manual to Use in Business,” Program-Ace. <https://program-ace.com/blog/how-to-develop-ar-manual-best-practices/> (accessed Aug. 29, 2024)
- [25] Quivervision, “QuiverVision,” Quivervision.com, 2025. <https://quivervision.com> (accessed Aug. 29, 2024).
- [26] Quiver Augmented Reality, “*Quiver Education Starter*,” YouTube. (Jan. 05, 2020). Accessed: Aug. 25, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=BOOh4-OpfOI>


- [27] Quiver Augmented Reality, “*How to use the Quiver App?*,” YouTube. (Mar. 03, 2022). Accessed: Aug. 25, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=vae2TBJIPww>
- [28] M. K. Mokhtar, F. Mohamed, M. S. Sunar, M. A. M. Arshad, and M. K. Mohd Sidik, “Development of Mobile-Based Augmented Reality Colouring for Preschool Learning,” *IEEE Xplore*, Nov. 01, 2018. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8632639> (accessed: Aug. 24, 2024).
- [29] Visible Body, “Visible Body | Use AR for Interactive Muscle Action Labs,” YouTube. (Mar. 17, 2023). Accessed: Aug. 25, 2024. [Online Video]. Available: <https://www.youtube.com/watch?v=1FxRXgIYZuE>
- [30] Visible Body, “Augmented Reality in Human Anatomy Atlas 2018 for Mobile | Visible Body,” YouTube. (Sep. 18, 2017). Accessed: Aug. 25, 2025. [Online Video]. <https://www.youtube.com/watch?v=TsGaDUWLhAI>
- [31] V. Body, “Visible Body Augmented Reality for iOS and Android,” www.visiblebody.com. <https://www.visiblebody.com/ar> (accessed Aug. 29, 2024).
- [32] M. Spencer, “ARChem,” App Store, Jun. 25, 2023. <https://apps.apple.com/lv/app/archem/id6450583945?platform=ipad> (accessed Aug. 29, 2024).
- [33] M. R. L. Y. Menikrama, et.al., “ARChem: Augmented Reality Chemistry Lab,” *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 0276–0280, Oct. 2021, doi: <https://doi.org/10.1109/iemcon53756.2021.9623121>.
- [34] Furkan Dinc, A. De, A. Goins, Tansel Halic, M. Massey, and F. Yarberry, “ARChem: Augmented Reality Based Chemistry LAB Simulation for Teaching and Assessment,” Nov. 2021, doi: <https://doi.org/10.1109/ithet50392.2021.9759587>.
- [35] JigSpace, “JigSpace: Interactive 3D AR Presentations for Measurable Business Impact,” Jig.com, 2024. <https://www.jig.com/> (accessed Apr. 25, 2025).

- [36] JigSpace, “JigSpace demo: 3D presentations in augmented reality,” YouTube. (Jul. 22, 2021). Accessed: Apr. 25, 2025. [Online Video]. <https://www.youtube.com/watch?v=eSsjQ7wZOtw>
- [37] Assemblr Studio, “Assemblr Studio - Build AR in Bigger Canvas,” itch.io, 2021. <https://assemblr.itch.io/assemblr-studio> (accessed Apr. 25, 2025).
- [38] U. Technologies, “Unity - Manual: Getting started with AR development in Unity,” docs.unity3d.com. <https://docs.unity3d.com/Manual/AROverview.html> (accessed Apr. 18, 2025).
- [39] Assemblr World, “Assemblr AR Markers,” YouTube. (Nov. 01, 2021). Accessed: Apr. 25, 2025. [Online Video]. <https://www.youtube.com/watch?v=YpwOecJytGk>
- [40] Assemblr | EDU, “Assemblr EDU - Augmented Reality (AR) for Classrooms,” edu.assemblrworld.com. <https://edu.assemblrworld.com/> (accessed Apr. 18, 2025).
- [41] S. Gray, “What is the Agile Methodology in Software Development?,” Medium, Jul. 29, 2020. <https://serenagray2451.medium.com/what-is-the-agile-methodology-in-software-development-c93023a7eb85> (accessed Apr. 20, 2025)
- [42] S. Timotheou et al., “Impacts of digital technologies on education and factors influencing schools’ digital capacity and transformation: A literature review,” *Education and Information Technologies*, vol. 28, no. 28, pp. 6695–6726, Nov. 2022, doi: <https://doi.org/10.1007/s10639-022-11431-8>.
- [43] A. Singh, K. Bacchuwar, and A. Bhasin, “A Survey of OCR Applications,” *International Journal of Machine Learning and Computing*, pp. 314–318, 2012, doi: <https://doi.org/10.7763/ijmlc.2012.v2.137>.
- [44] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character Region Awareness for Text Detection,” *arXiv (Cornell University)*, Apr. 2019, doi: <https://doi.org/10.48550/arxiv.1904.01941>.
- [45] B. Shi, X. Bai, and C. Yao, “An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*,


- vol. 39, no. 11, pp. 2298–2304, Nov. 2017, doi: <https://doi.org/10.1109/tpami.2016.2646371>.
- [46] D. R. Vedhaviyassh, R. Sudhan, G. Saranya, M. Safa, and D. Arun, “Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using Deep Learning Algorithm,” *IEEE Xplore*, Dec. 01, 2022. doi: <https://doi.org/10.1109/ICECA55336.2022.10009215>.

APPENDICES



Appendix A Poster




TechTutor: AR-Integrated Learning Application for Computer and Technology Use




INTRODUCTION




TechTutor is a mobile application that integrates Augmented Reality (AR) and Artificial Intelligence (AI) to help beginners and older adults learn basic computer skills. It provides interactive tutorials, quizzes, and AI assistance, making learning more engaging and accessible.




PROBLEM STATEMENTS




- **Digital Literacy Gap:** Older adults face difficulties using technology due to lack of simple guidance.
- **Basic Skills Challenge:** Tasks like Email and Microsoft Word remain confusing without step-by-step help.
- **Existing Methods:** Traditional teaching is passive, lacking interactivity and engagement.



PROJECT OBJECTIVES







- To develop AR-based learning modules.
- To develop an AI Tutor module.
- To implement adaptive learning features.







MAIN MODULES




AR Learn:
AI Identify, AR Models, AR Quiz




AR Guide:
Step-by-Step tutorials, TTS narration, Visual highlights, 3D AR Mode & Canvas Mode




AI Tutor:
Image & Text input, EasyOCR keyword detection






METHODOLOGY




- **Tools:** Unity, Vuforia, ARCore, OpenAI API, EasyOCR
- **Techniques:** AR Plane Detection, Marker-based AR, TTS/STT
- **Approach:** Agile Software Development Methodology

CONCLUSION



TechTutor demonstrates that how AR and AI can make learning basic computer skills more interactive, accessible, and engaging. The system helps reduce the digital literacy gap, especially for beginners and older adults.



Universiti Tunku Abdul Rahman
Bachelor of Computer Science (Honours)
Faculty of Information and Communication Technology

Supervisor: Dr Ng Hui Fuang
Prepared By: Loh Chia Heung

Appendix B Credits for 3D Models

1. 3D Keyboard Model



Model Name	Gaming keyboard
Source (Link)	https://sketchfab.com/3d-models/gaming-keyboard-c1e5e07153d84978a1ad7256d720220a
Author	ZxTulz
License Type	CC Attribution

2. 3D Mouse Model



Model Name	Ice Claw mouse
Source (Link)	https://sketchfab.com/3d-models/ice-claw-mouse-43de4d030ae94667bd8b8a478dd371ac
Author	dmitriy7776661111
License Type	CC Attribution

3. 3D Monitor Model



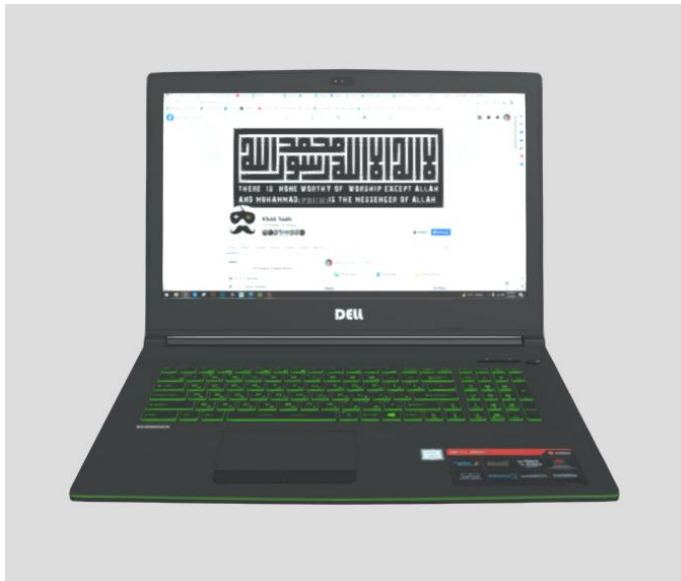
Model Name	Monitor
Source (Link)	https://sketchfab.com/3d-models/monitor-9f6f9018f14a4dbea1ad1aea0ce89e7c
Author	portgl16
License Type	CC Attribution

4. 2nd 3D Monitor Model



Model Name	PC Monitor
Source (Link)	https://sketchfab.com/3d-models/pc-monitor-qb1DsRh6GJU2intFpbqCjRQMTD5
Author	3DHaupt
License Type	CC Attribution - Noncommercial

5. 3D Laptop Model



Model Name	Laptop
Source (Link)	https://sketchfab.com/3d-models/laptop-75b0086c55044de59aa0126f4f30933f
Author	KhanSaab
License Type	CC Attribution

6. 2nd 3D Laptop Model



Model Name	Asus ROG Strix Scar 17 (2023) G733 Gaming Laptop
Source (Link)	https://sketchfab.com/3d-models/asus-rog-strix-scar-17-2023-g733-gaming-laptop-51eca7b2e5884c4087f3499e523d5184
Author	Ranaha Creative Studio
License Type	CC Attribution

7. 3D Speaker Model



Model Name	KRK Rokit 5 Speakers
Source (Link)	https://sketchfab.com/3d-models/krk-rokit-5-speakers-6448444ed5d14b3eb8fe44a30d0d8712
Author	Ryan_Nein
License Type	CC Attribution

Appendix C Completed Tutorial Steps for Microsoft Word

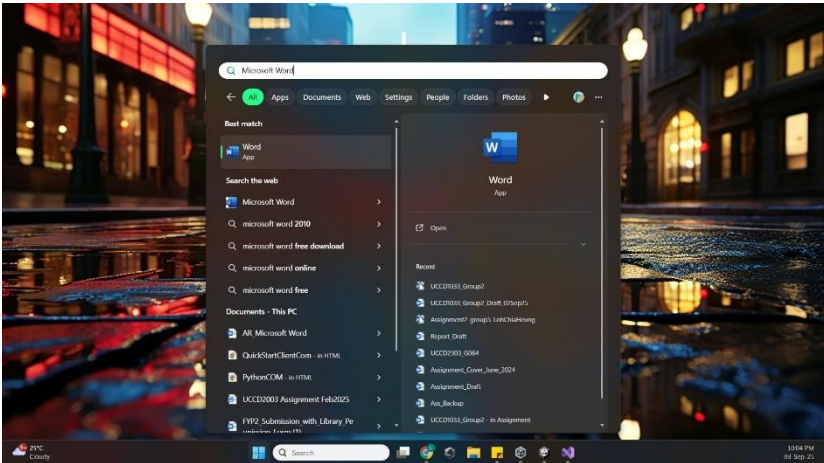
Step	Tutorial Panel
<p>Step 0 – AR Guide – Using Microsoft Word</p> <p>(1 Step)</p>	<div data-bbox="592 353 1275 786"> </div> <p style="text-align: center;">Steps Panel</p> <div data-bbox="523 857 1347 1350"> </div> <p style="text-align: center;">Monitor Model Screen</p>
<p>Step 1 – Open the Microsoft Word</p> <p>(3 steps)</p>	<div data-bbox="572 1453 1294 1917"> </div> <p style="text-align: center;">Steps Panel</p>



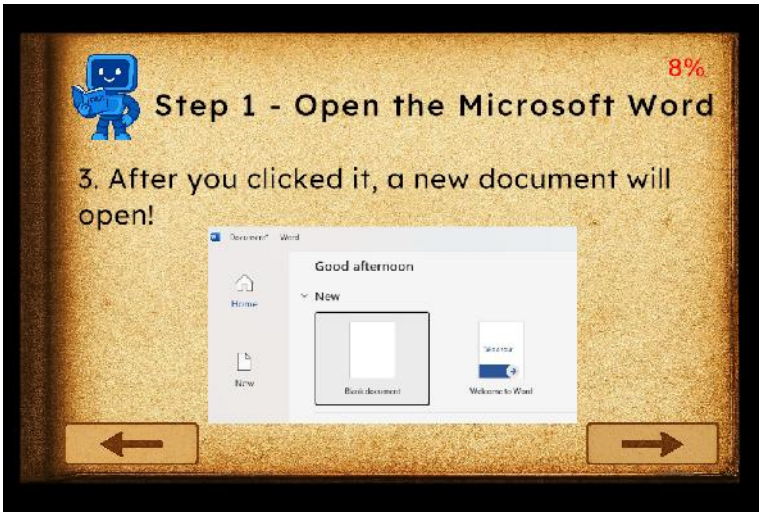
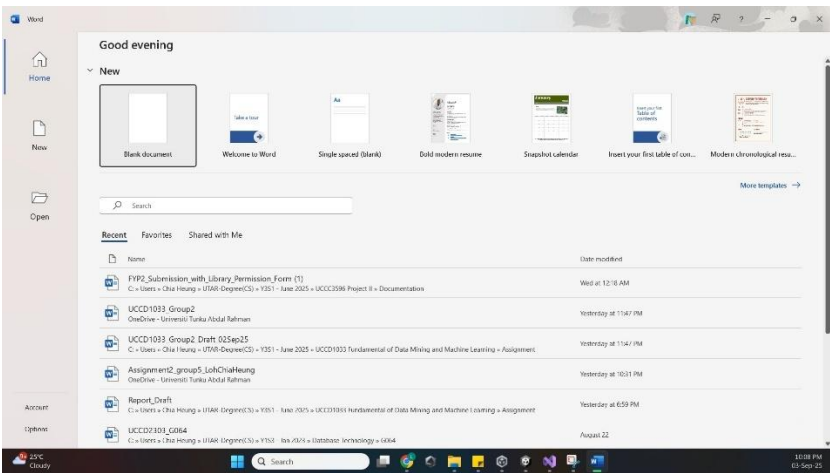
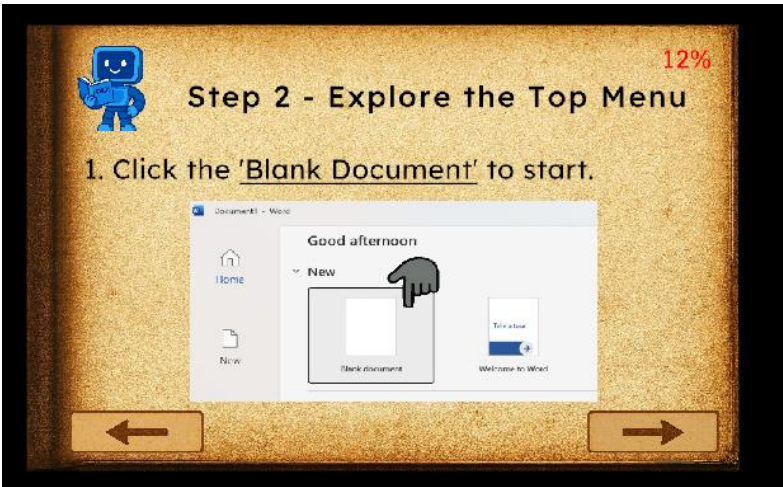
Monitor Model Screen

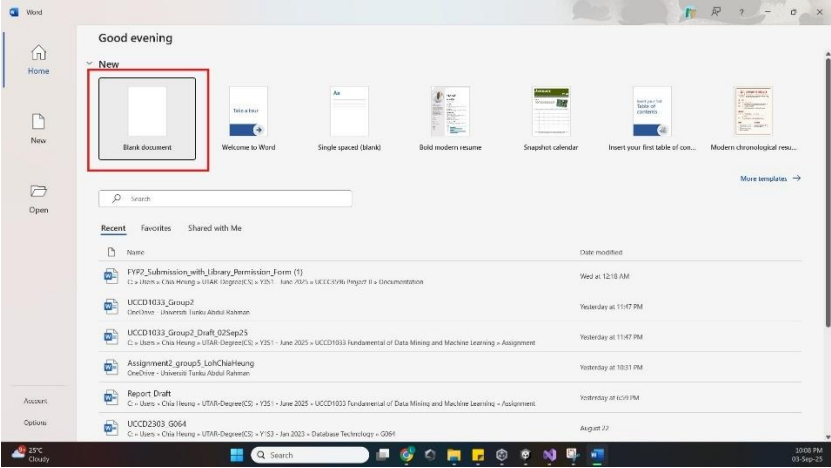

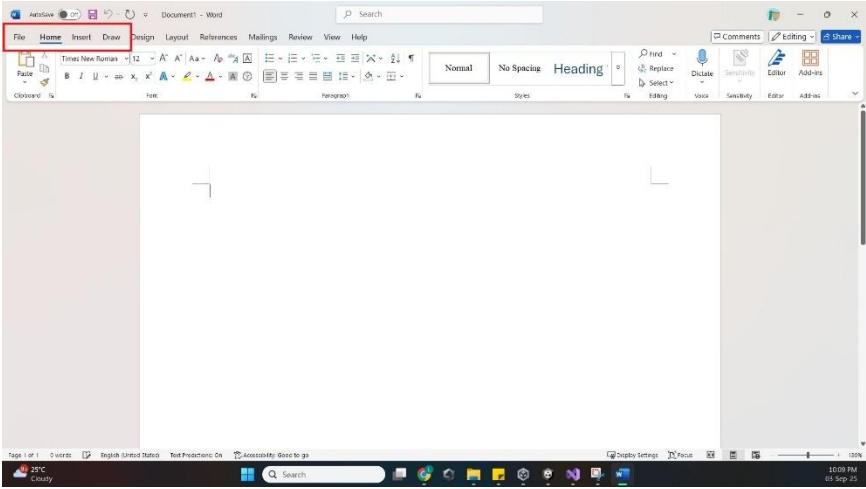


Steps Panel



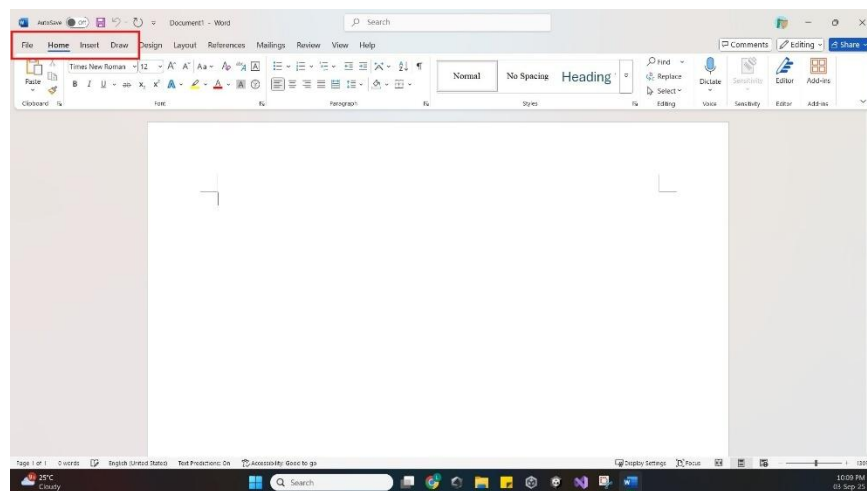
Monitor Model Screen

	<div data-bbox="552 221 1313 730">  </div> <div data-bbox="853 748 1013 790" data-label="Caption"> <p>Steps Panel</p> </div> <div data-bbox="517 799 1350 1270">  </div> <div data-bbox="777 1288 1090 1330" data-label="Caption"> <p>Monitor Model Screen</p> </div>
<p>Step 2 – Explore the Top Menu</p> <p>(5 steps)</p>	<div data-bbox="541 1375 1326 1859">  </div> <div data-bbox="853 1877 1013 1917" data-label="Caption"> <p>Steps Panel</p> </div>

	<div></div> <div>Monitor Model Screen</div>
	<div></div> <div>Steps Panel</div>
	<div></div> <div>Monitor Model Screen</div>

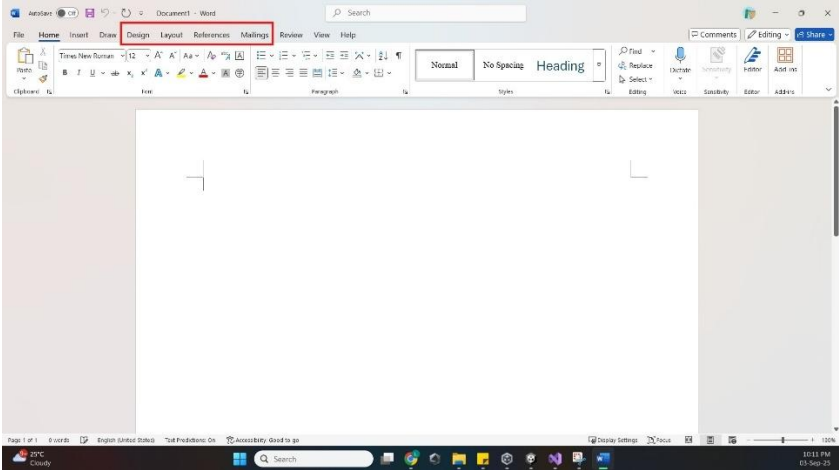

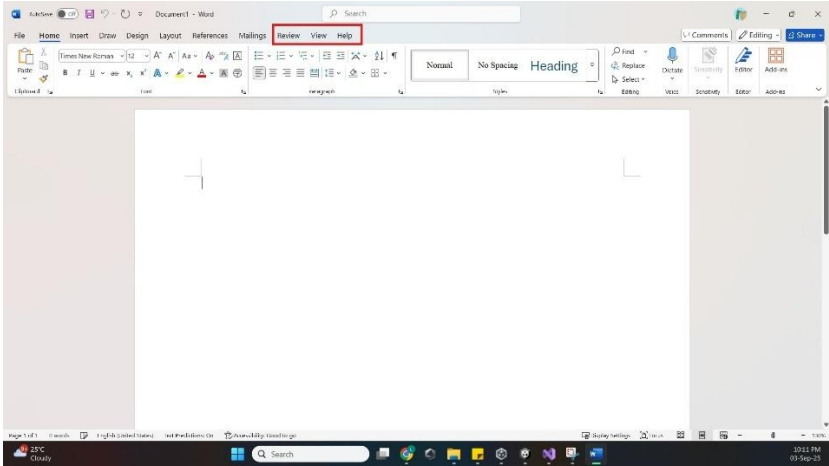


Steps Panel

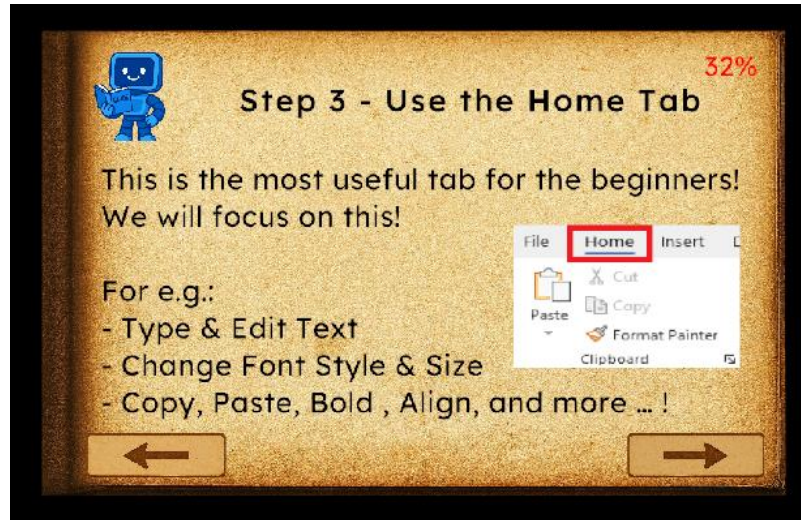


Monitor Model Screen

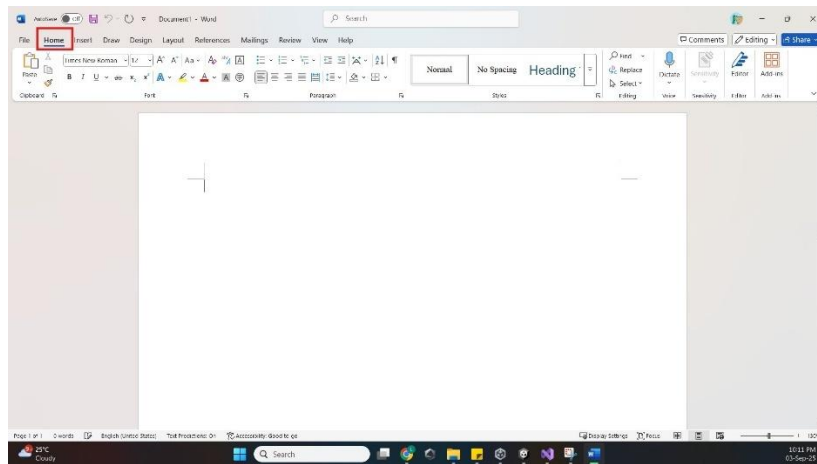


	<div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>
	<div></div> <div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>

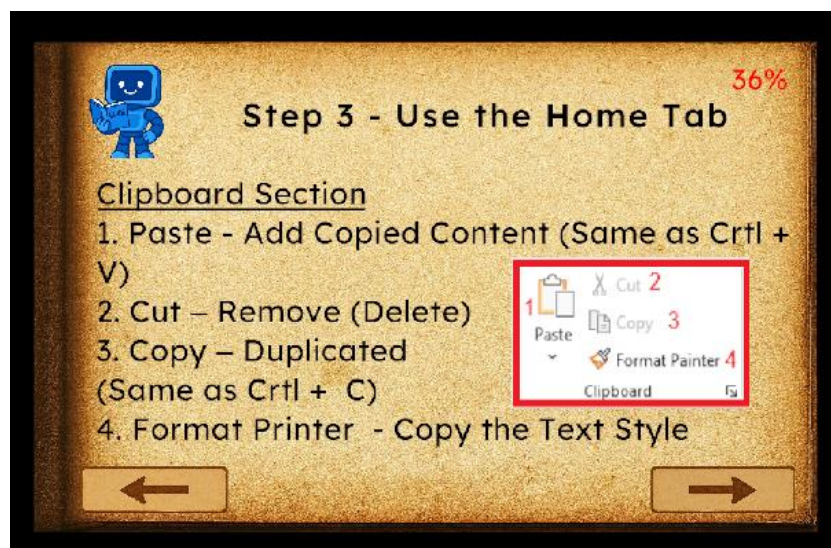
**Step 3 –
Use the
Home Tab
(5 Steps)**



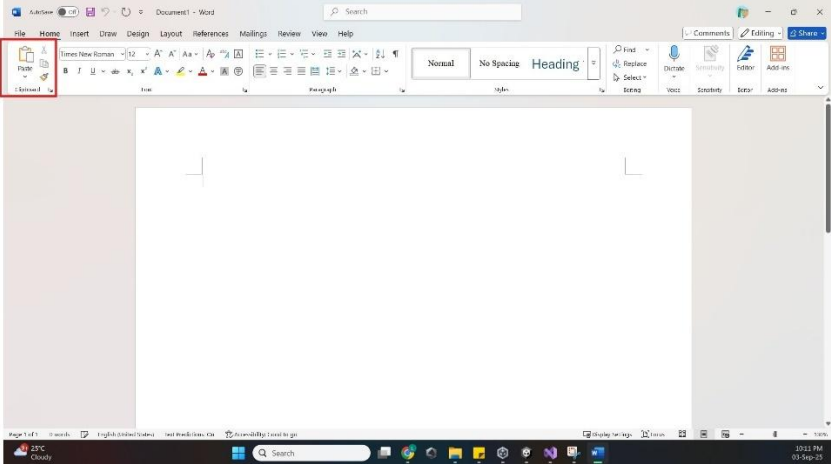
Step Panel



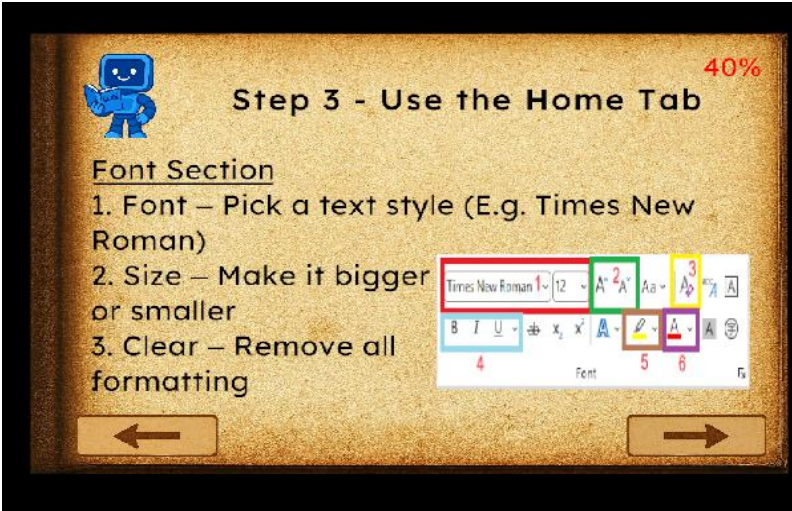
Monitor Model Screen



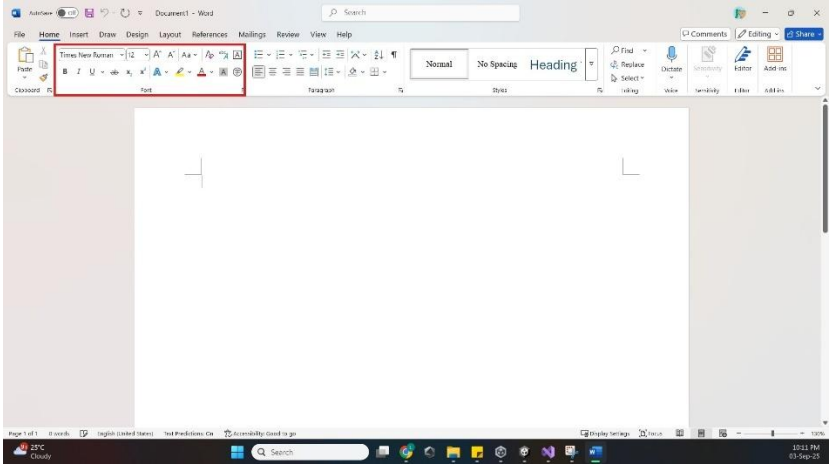
Steps Panel



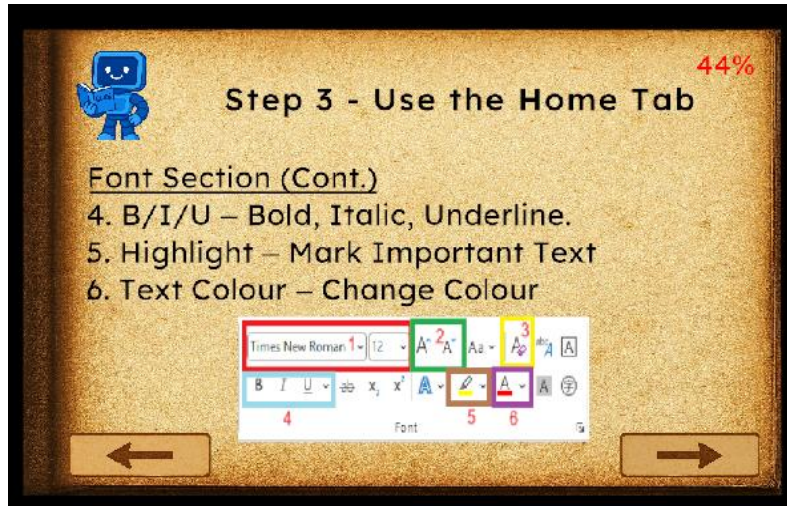
Monitor Model Screen



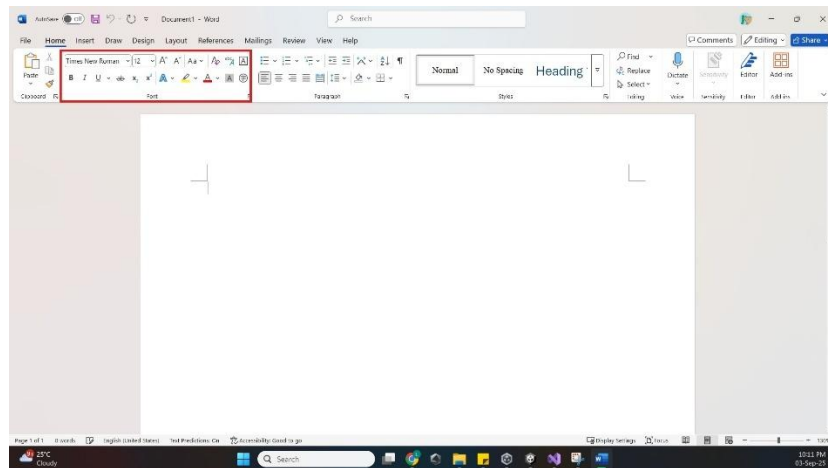
Steps Panel



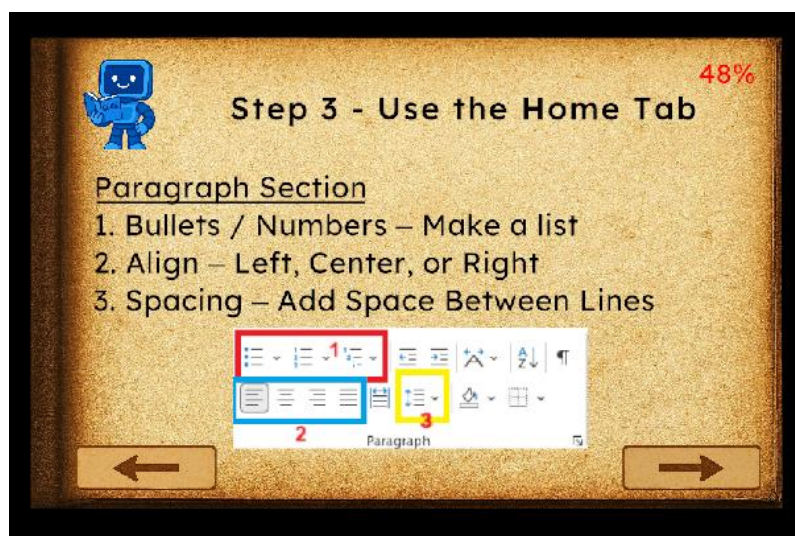
Monitor Model Screen



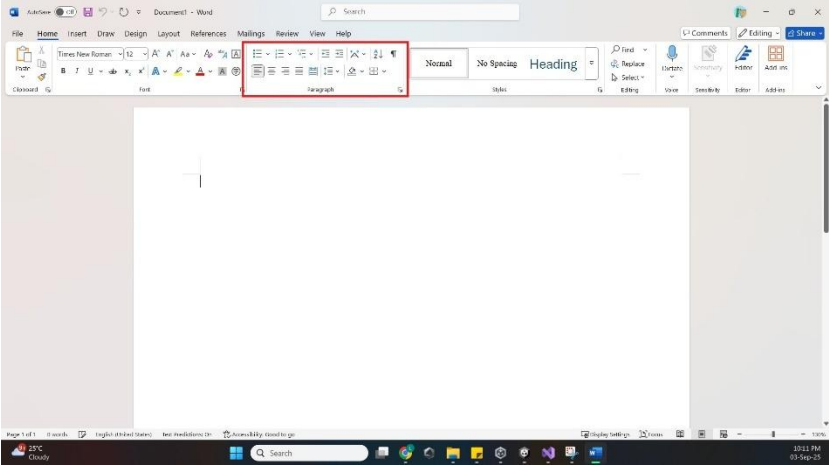

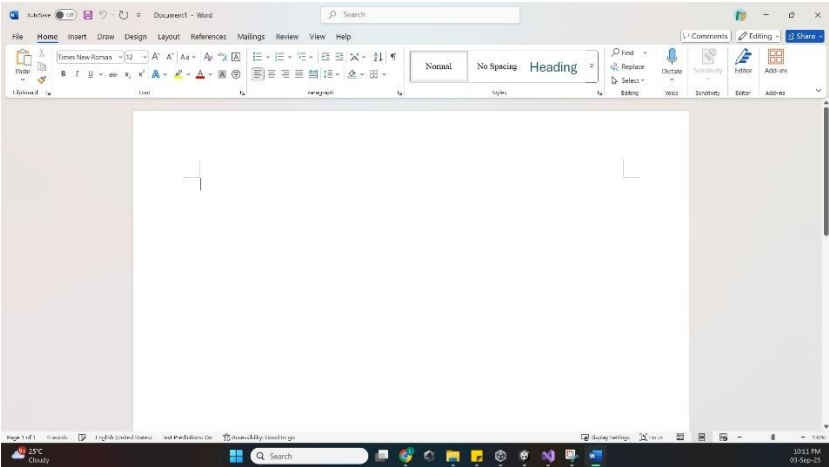
Steps Panel



Monitor Model Screen

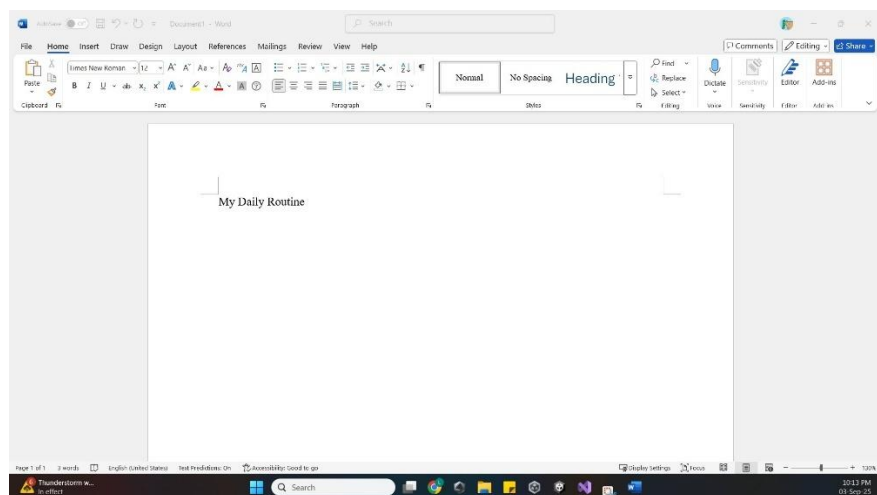


Steps Panel

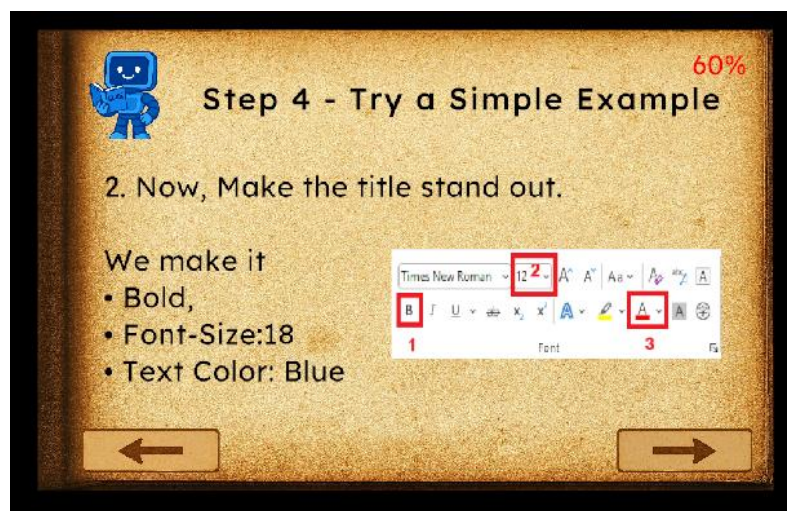
	<div><p>Monitor Model Screen</p></div>
<div><p>Step 4 – Try a Simple Example (11 Steps)</p></div>	<div><p>Steps Panel</p><p>Monitor Model Screen</p></div>



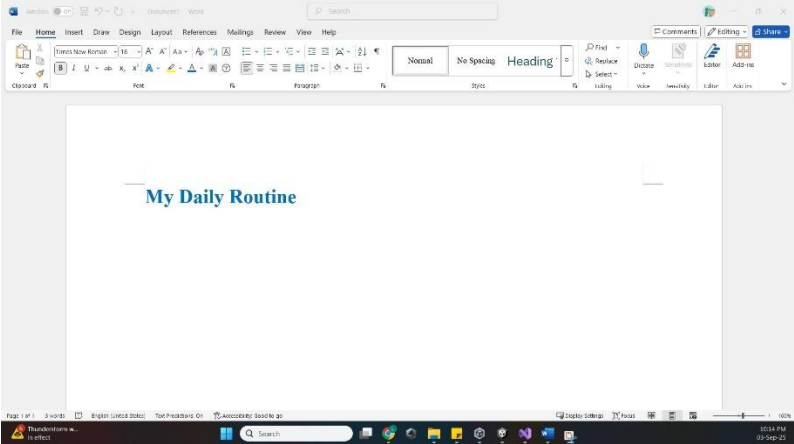
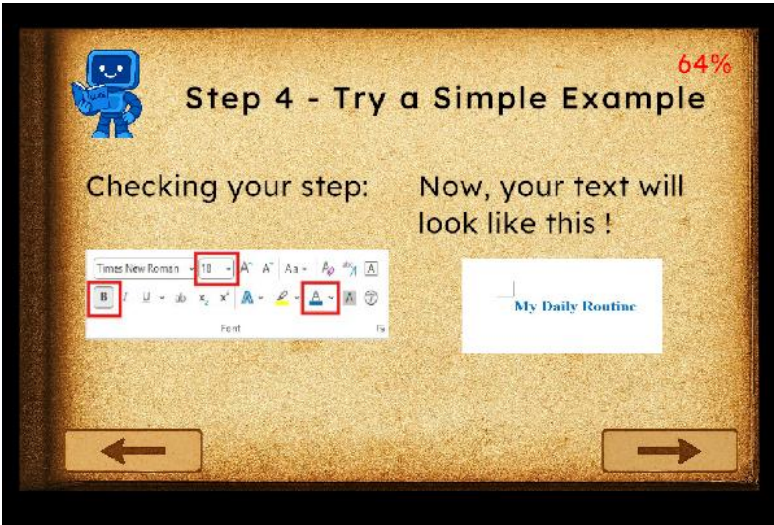
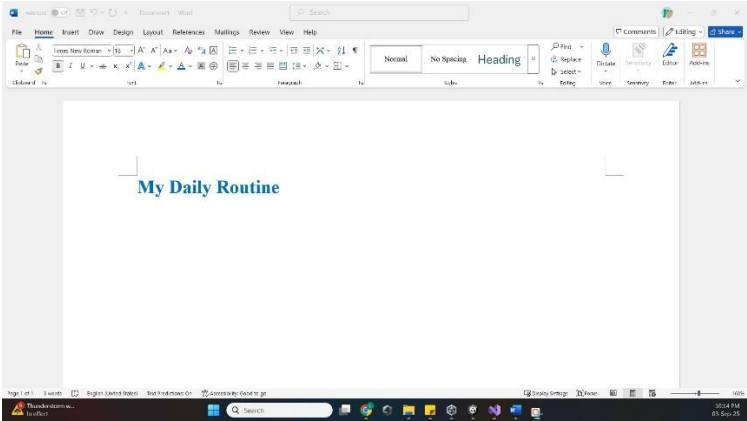
Steps Panel



Monitor Model Screen

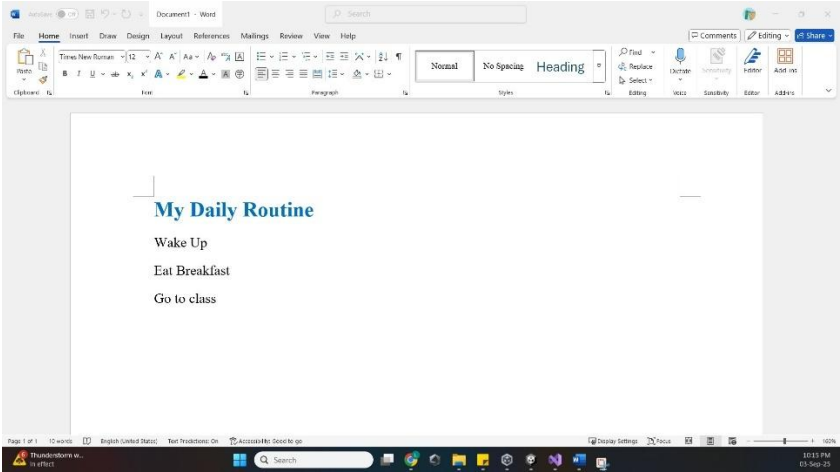


Steps Panel

	<div><p>Monitor Model Screen</p></div>
	<div><p>Steps Panel</p></div>
	<div><p>Monitor Model Screen</p></div>

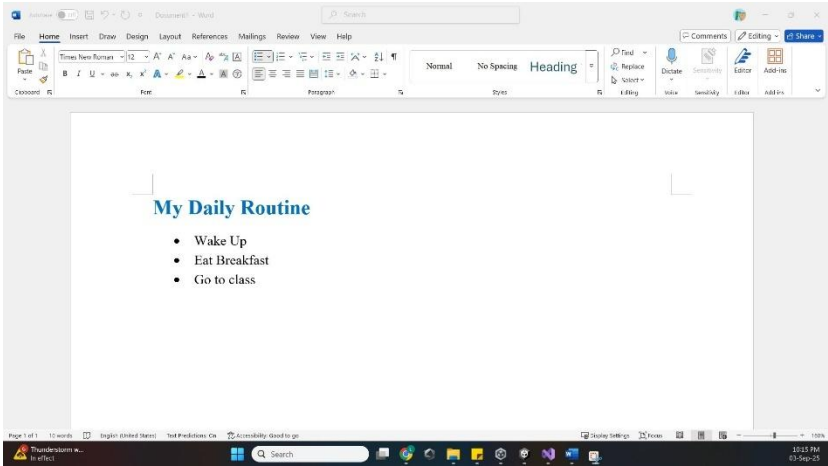

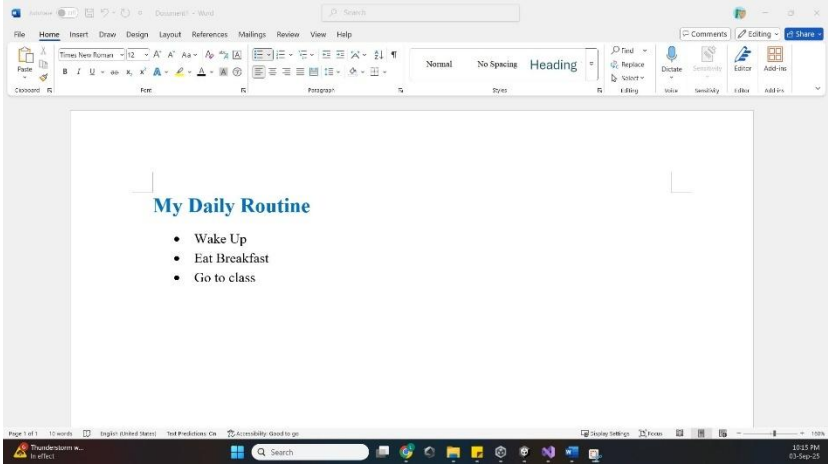


Steps Panel



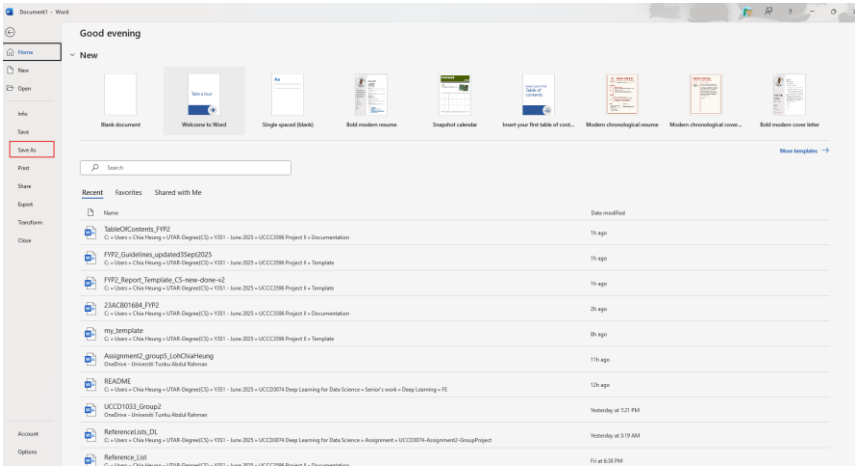
Monitor Model Screen



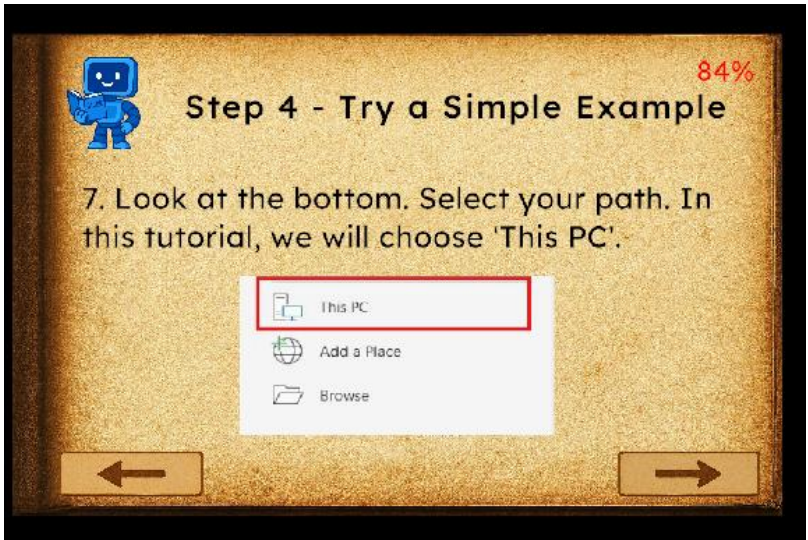
	<div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>
	<div></div> <div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>



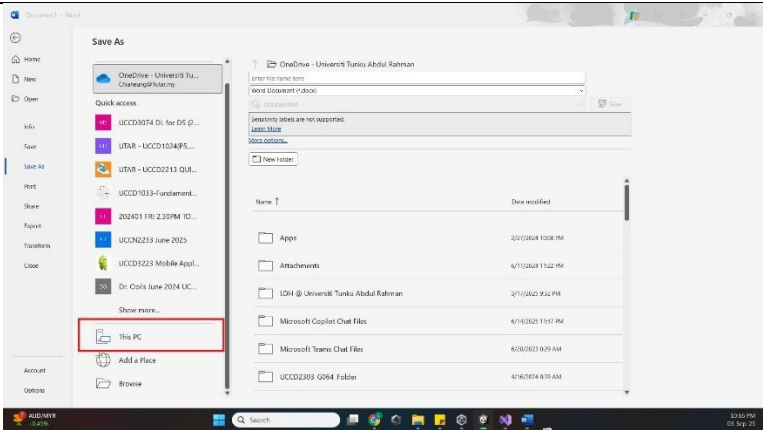
Steps Panel



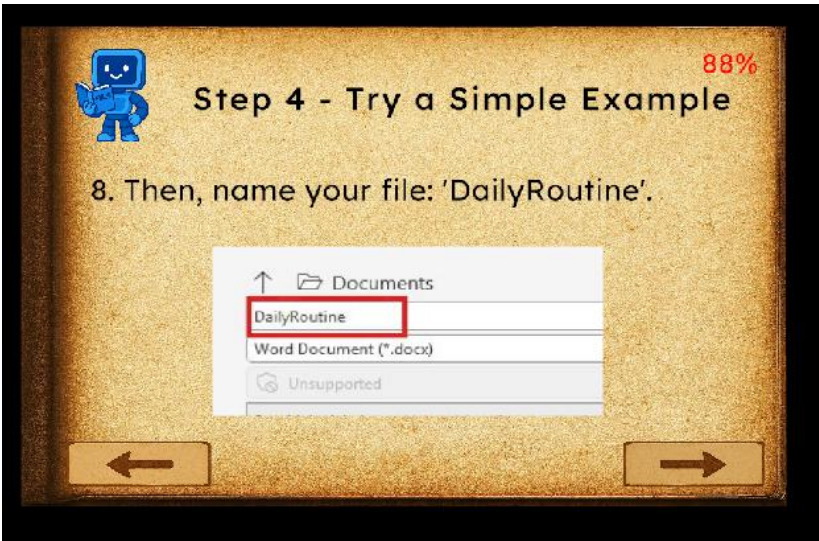
Monitor Model Screen



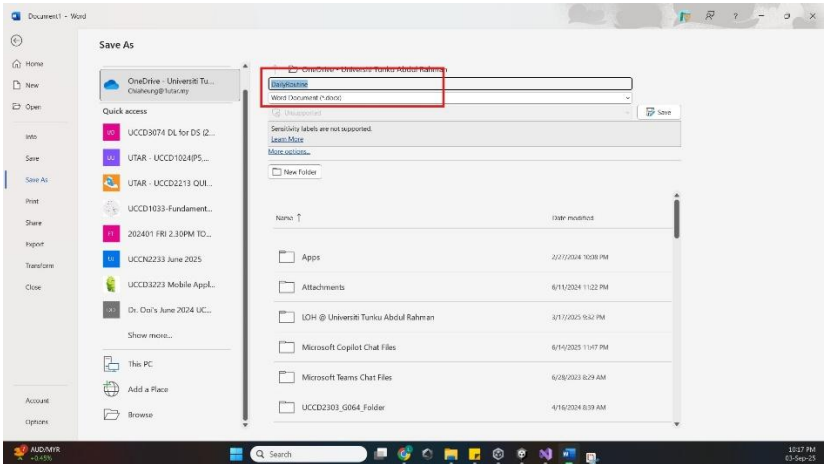
Steps Panel



Monitor Model Screen



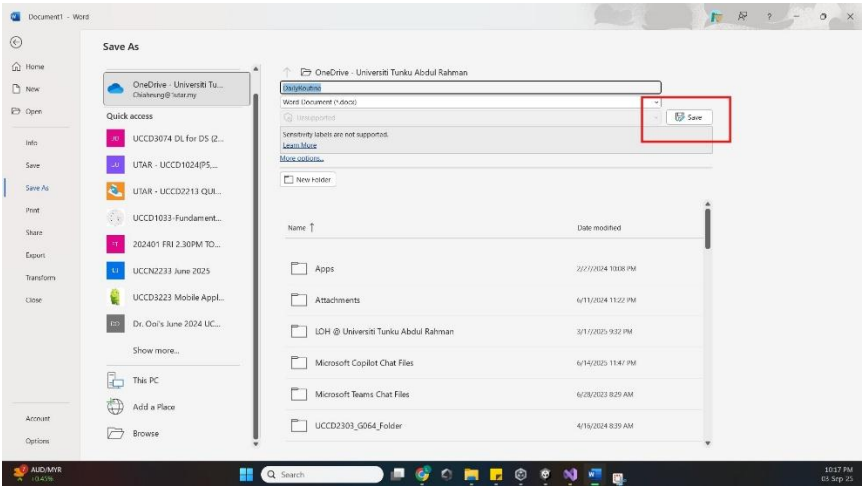
Steps Panel



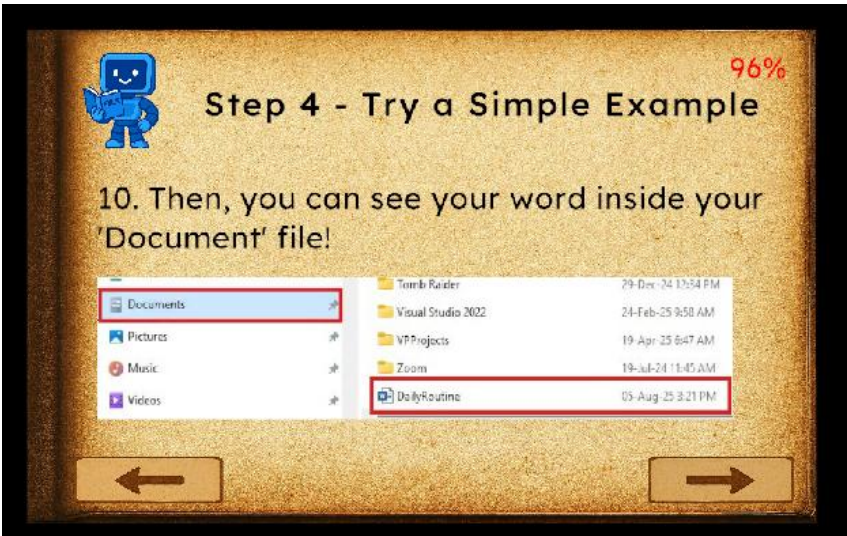
Monitor Model Screen

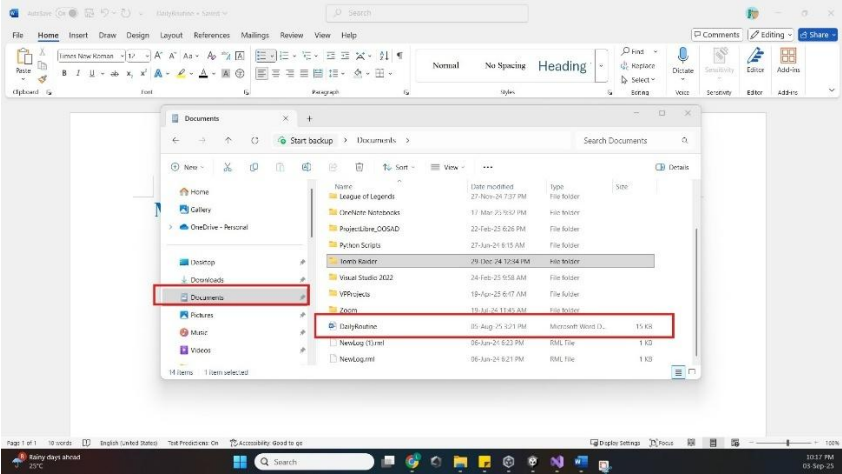




Steps Panel








Monitor Model Screen

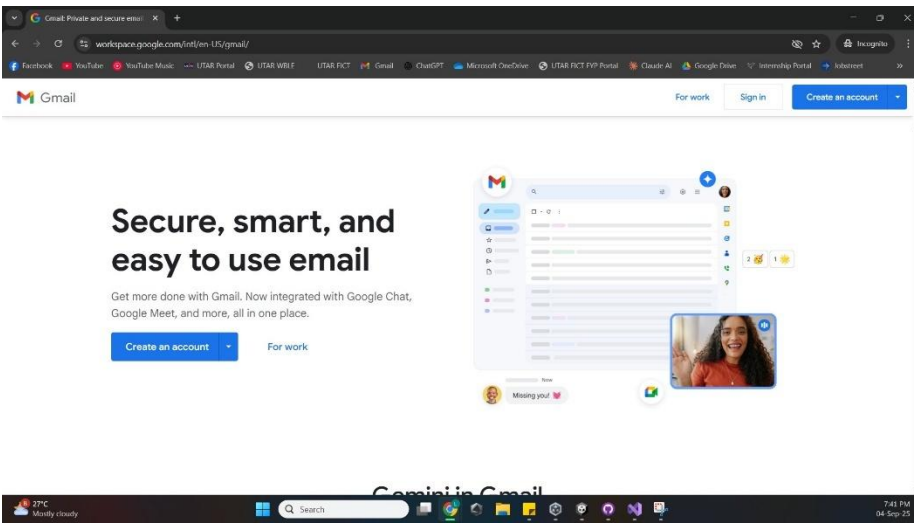


	<div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>
<div>Step 5 – Tutorial Completed (1 Step)</div>	<div></div> <div>Steps Panel</div> <div></div> <div>Monitor Model Screen</div>


Appendix D Complete Tutorial Steps for Sending Email (Gmail)

Step	Tutorial Panel
<p>Step 0 – AR Guide – Sending an Email</p> <p>(1 Step)</p>	<div data-bbox="555 353 1318 862">  </div> <p data-bbox="855 884 1015 920">Steps Panel</p> <div data-bbox="529 936 1340 1391">  </div> <p data-bbox="778 1408 1091 1444">Monitor Model Screen</p>
<p>Step 1 – Open and Login to Gmail</p> <p>(6 Steps)</p>	<div data-bbox="539 1494 1329 2004">  </div>

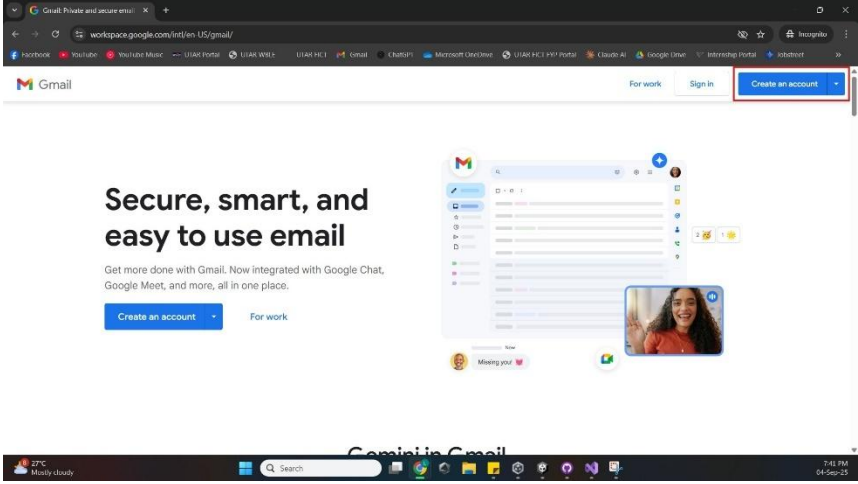
	<p style="text-align: center;">Steps Panel</p>  <p style="text-align: center;">Monitor Model Screen</p>
	 <p style="text-align: center;">Steps Panel</p>



Monitor Model Screen



Steps Panel

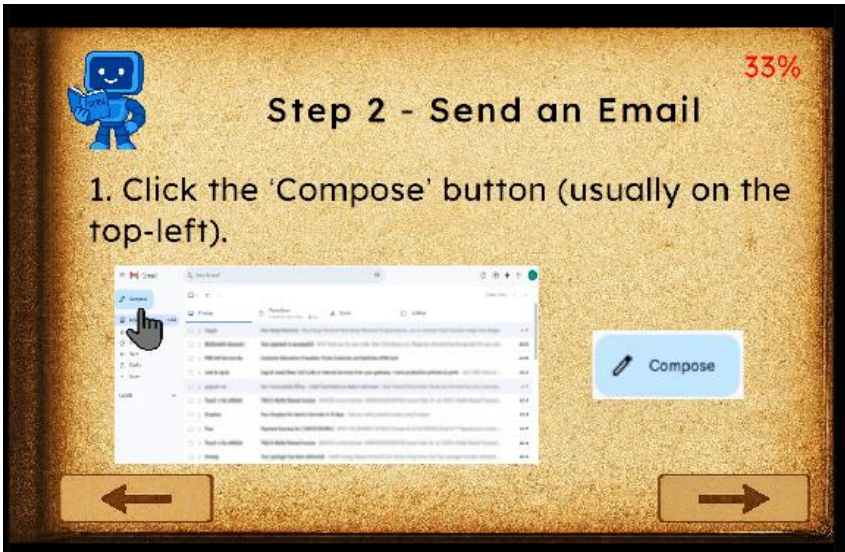
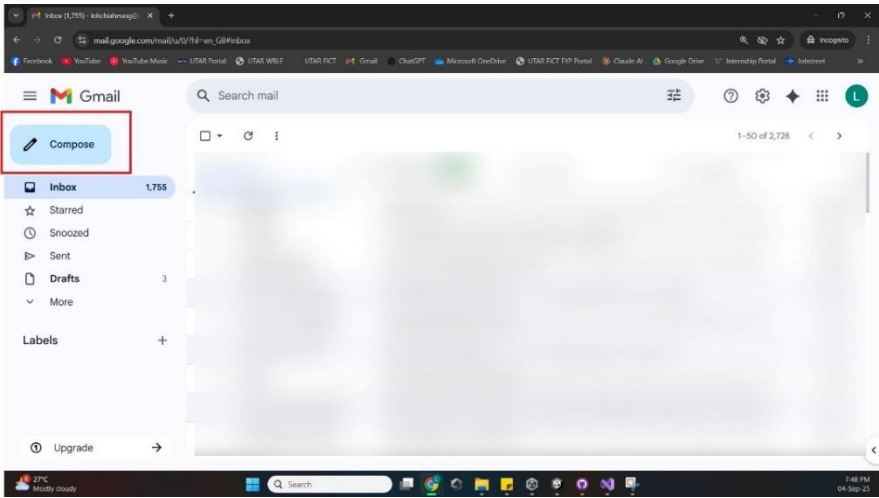


	<p style="text-align: center;">Monitor Model Screen</p> <div data-bbox="523 246 1340 777"> </div> <p style="text-align: center;">Steps Panel</p> <div data-bbox="504 880 1361 1357"> </div> <p style="text-align: center;">Monitor Model Screen</p> <div data-bbox="497 1460 1366 2016"> </div>
--	--

Steps Panel

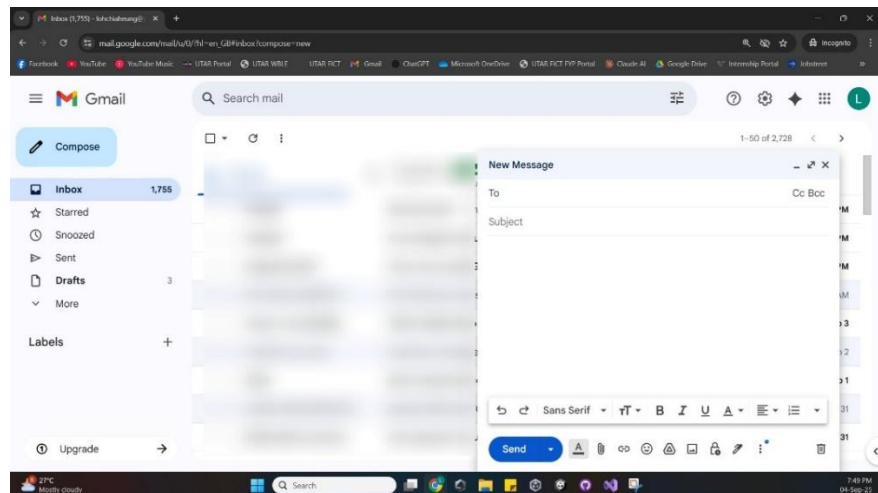
Monitor Model Screen

Steps Panel

	Monitor Model Screen
Step 2 – Send an Email (12 Steps)	<div><p>The image shows a monitor screen with a blue robot icon in the top-left corner. In the top-right corner, there is a red '33%' progress indicator. The main heading is 'Step 2 - Send an Email'. Below it, the instruction reads: '1. Click the 'Compose' button (usually on the top-left)'. A screenshot of the Gmail interface is shown, with a hand cursor pointing to the 'Compose' button in the top-left sidebar. To the right of the screenshot is a blue 'Compose' button with a pencil icon. At the bottom of the screen are two large, light-brown buttons with black arrows pointing left and right.</p></div> <p>Steps Panel</p> <div><p>The image shows a screenshot of the Gmail web interface. The 'Compose' button, located in the top-left sidebar, is highlighted with a red rectangular box. The sidebar also shows 'Inbox' with 1,755 emails, 'Starred', 'Snoozed', 'Sent', 'Drafts' with 3 emails, and 'Labels'. The main content area is blurred. The bottom of the screen shows a Windows taskbar with a search bar and various application icons. The system clock in the bottom-right corner shows '7:48 PM 04-Sep-25'.</p></div> <p>Monitor Model Screen</p>



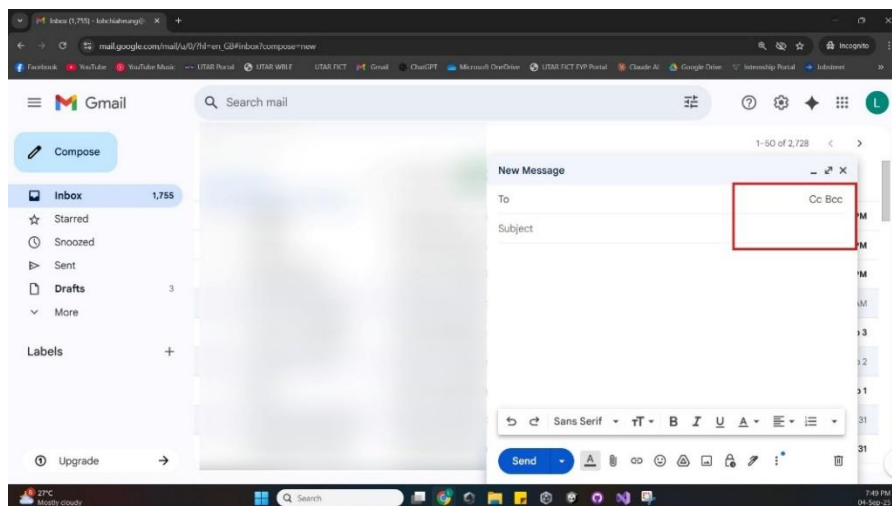
Steps Panel



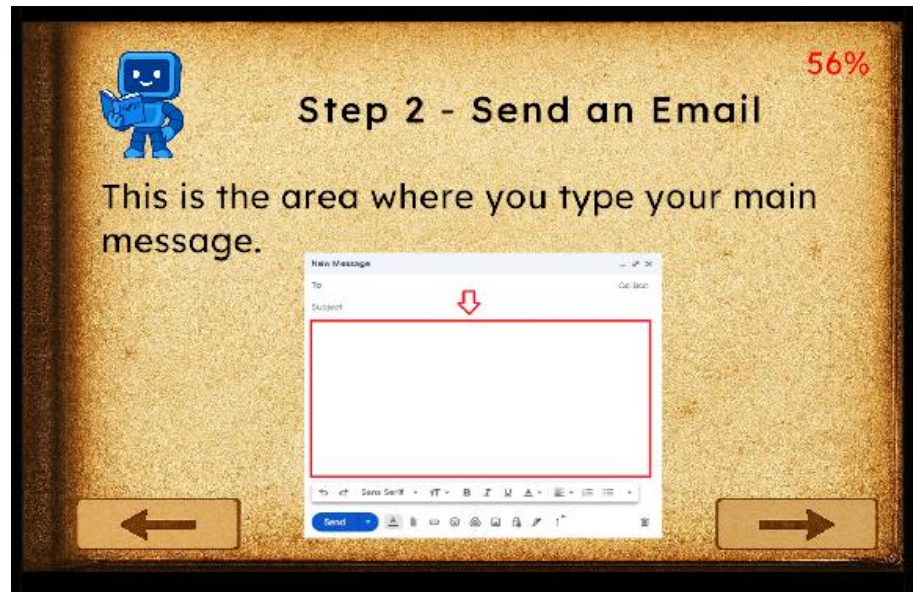
Monitor Model Screen



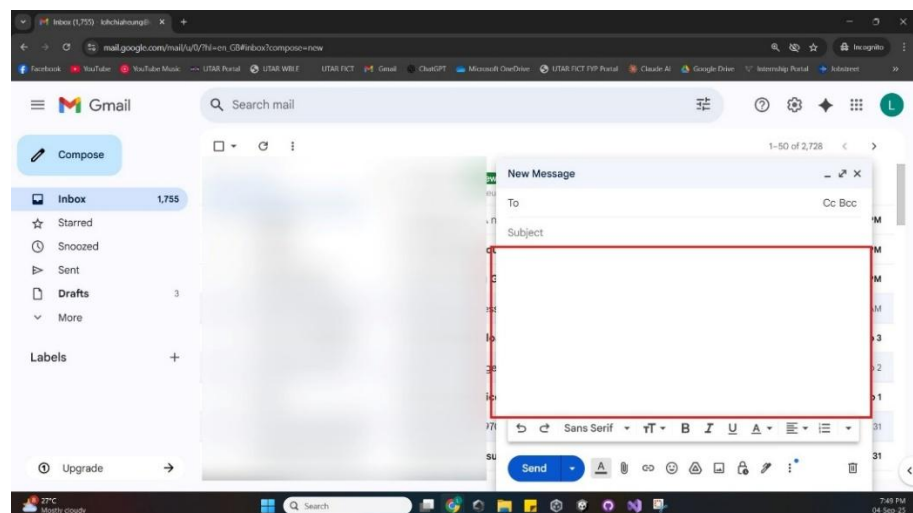
Steps Panel



Monitor Model Screen



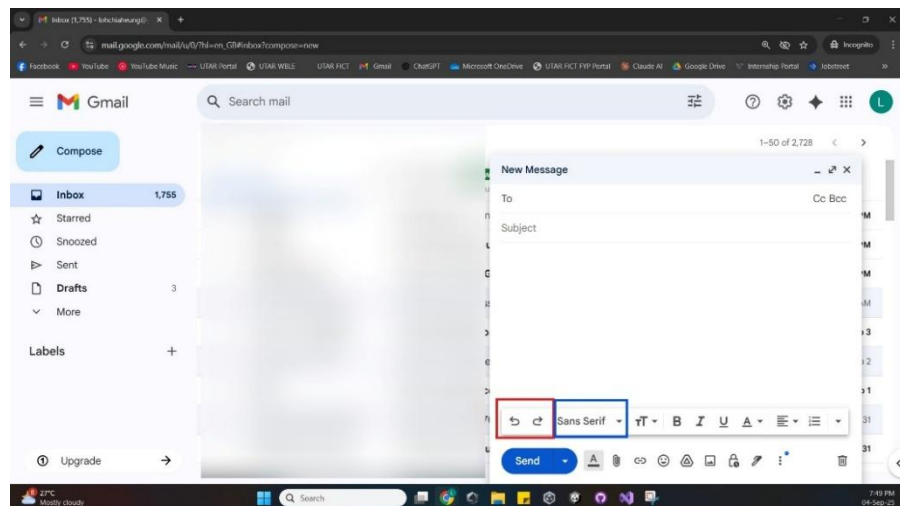
Steps Panel



Monitor Model Screen



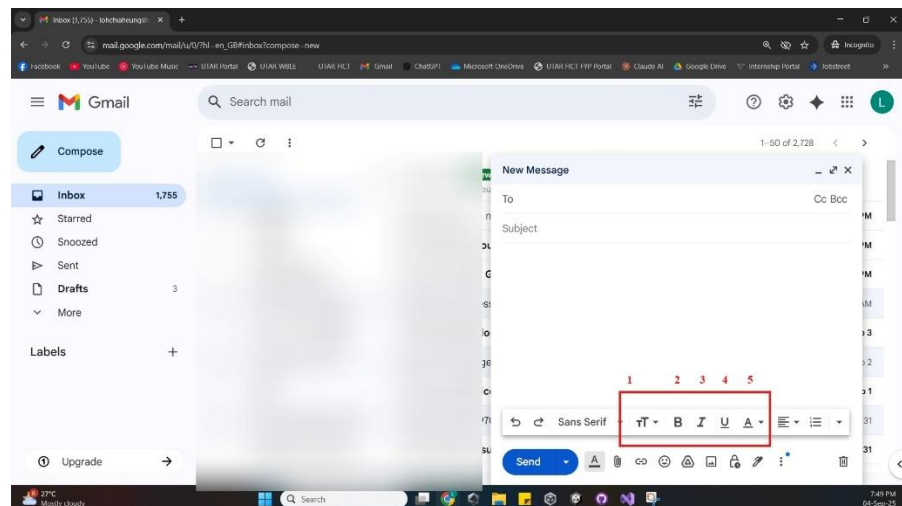
Steps Panel



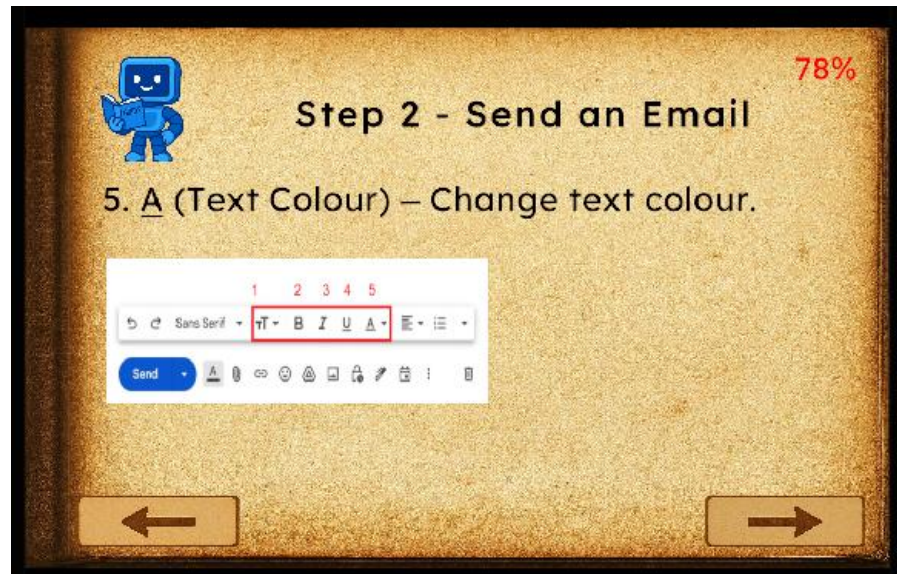
Monitor Model Screen



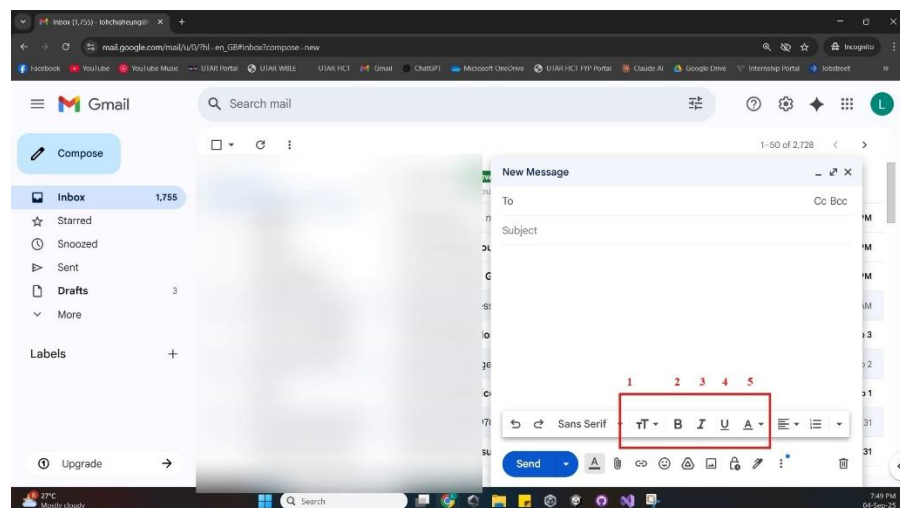
Steps Panel



Monitor Model Screen



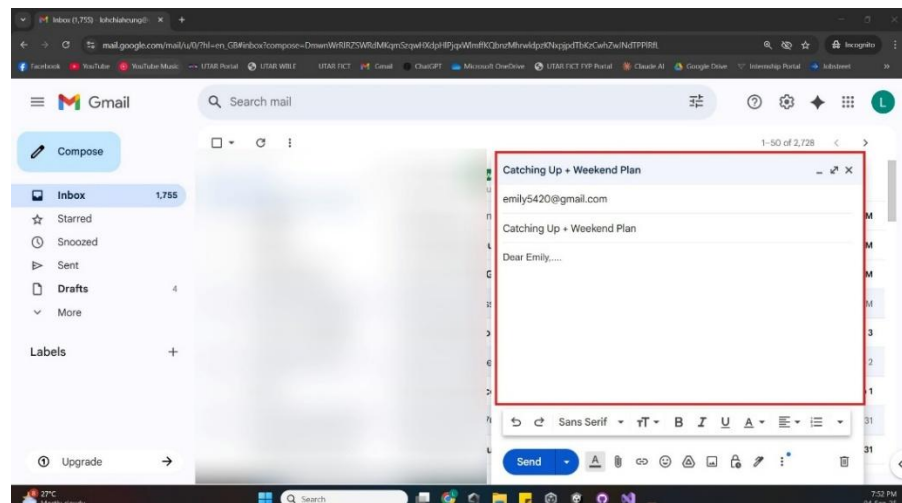
Steps Panel



Monitor Model Screen



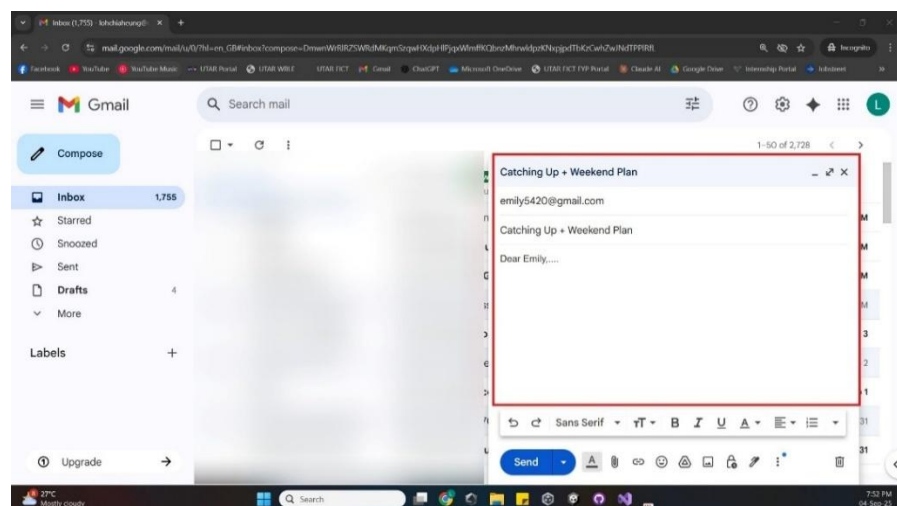
Steps Panel



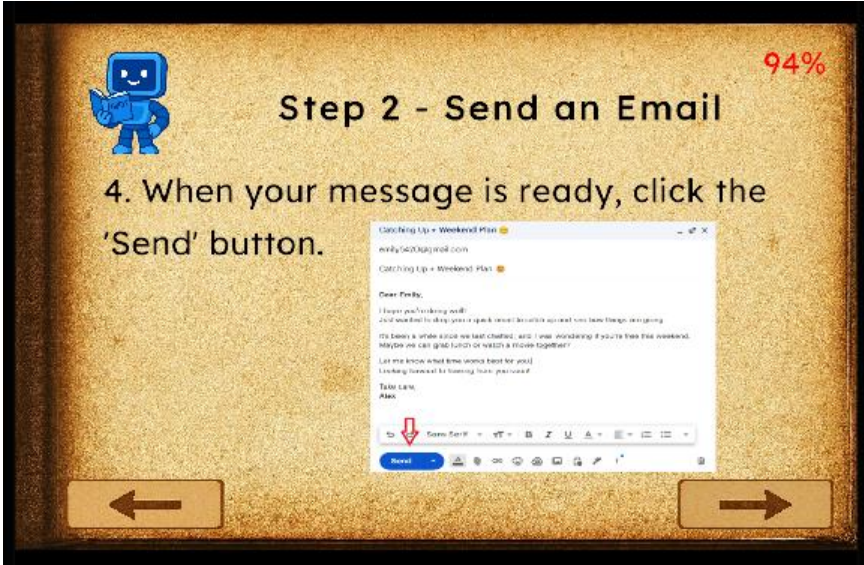
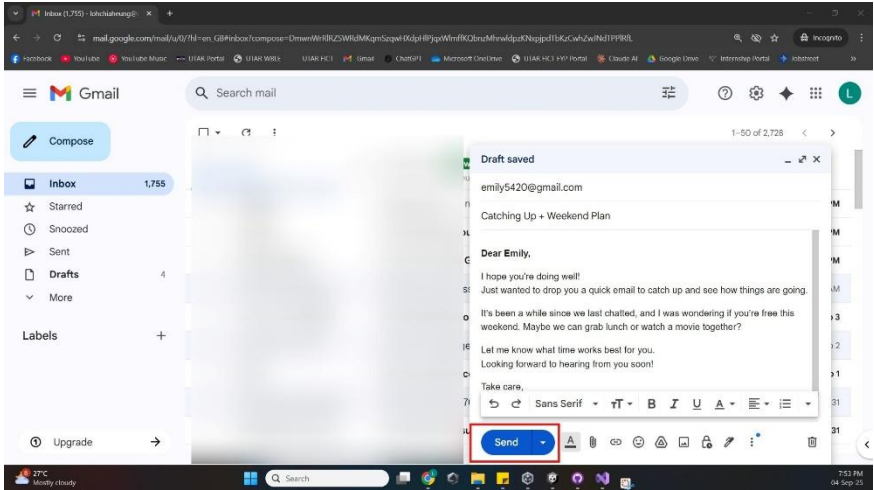

Monitor Model Screen

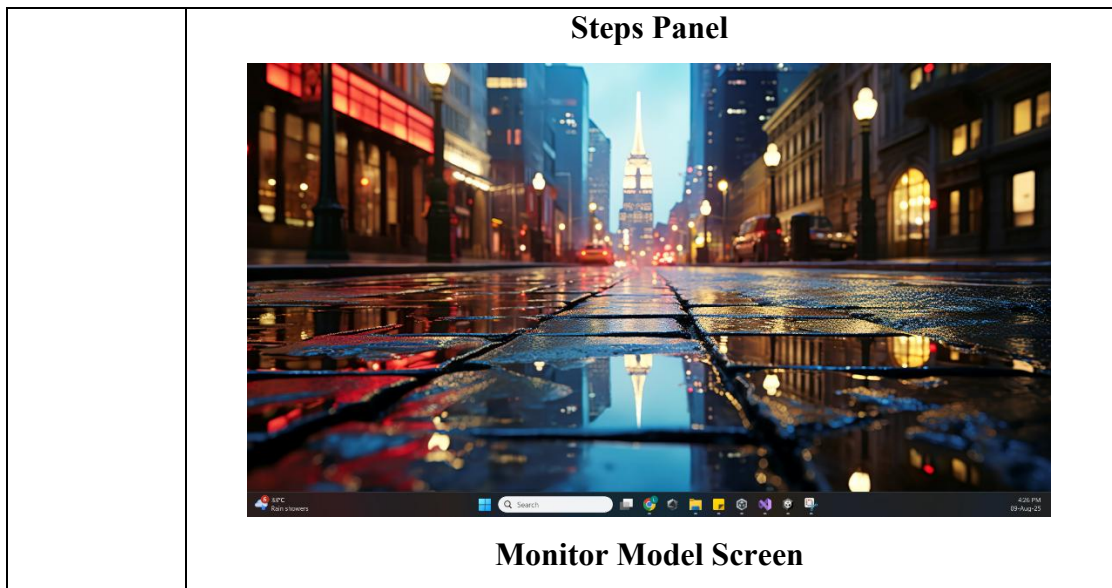


Steps Panel

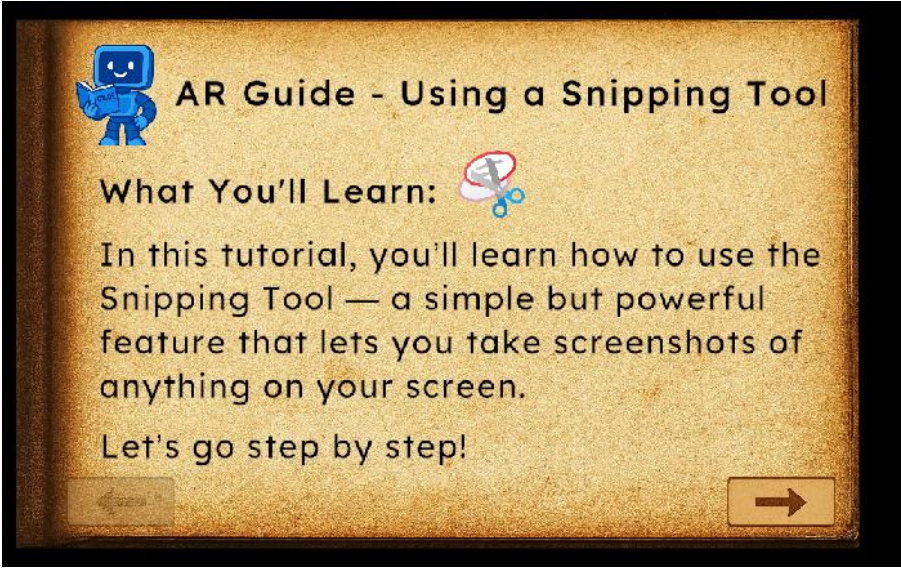



Monitor Model Screen

	 <p style="text-align: center;">Steps Panel</p>  <p style="text-align: center;">Monitor Model Screen</p>
<p>Step 3 – Tutorial Completed</p> <p>(1 Step)</p>	



Appendix E Complete Tutorial Steps for Snipping Tool

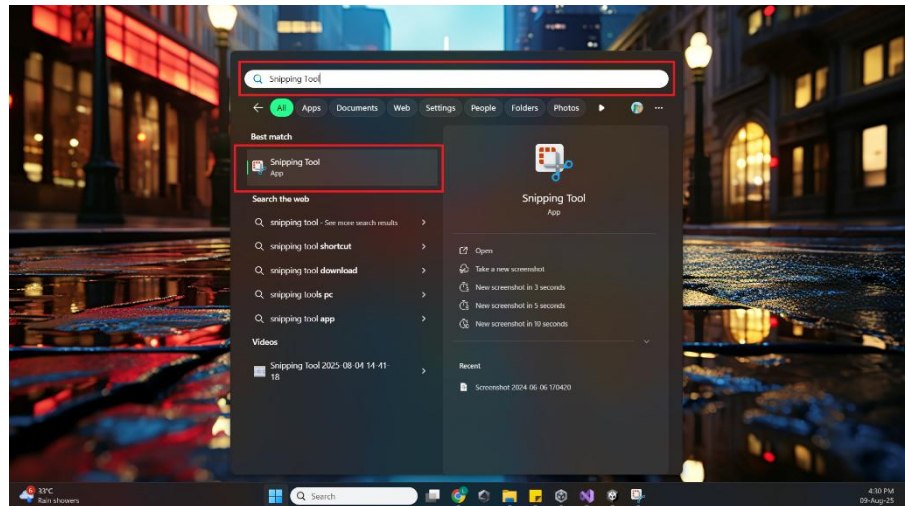
Step	Tutorial Panel
Step 0 – AR Guide – Using a Snipping Tool (1 Step)	<div><p>The AR Guide screen has a parchment-like background. At the top left is a blue robot icon holding a book. The title 'AR Guide - Using a Snipping Tool' is in the top right. Below the title, it says 'What You'll Learn:' followed by a scissors icon. The main text reads: 'In this tutorial, you'll learn how to use the Snipping Tool — a simple but powerful feature that lets you take screenshots of anything on your screen. Let's go step by step!'. At the bottom are two arrows: a left arrow on the left and a right arrow on the right.</p></div> <div><p>Steps Panel</p><p>The Steps Panel shows a night-time street scene with wet pavement reflecting city lights and buildings. The Windows taskbar is visible at the bottom of the image.</p></div> <div><p>Monitor Model Screen</p></div>

**Step 1 –
Open the
Snipping
Tool

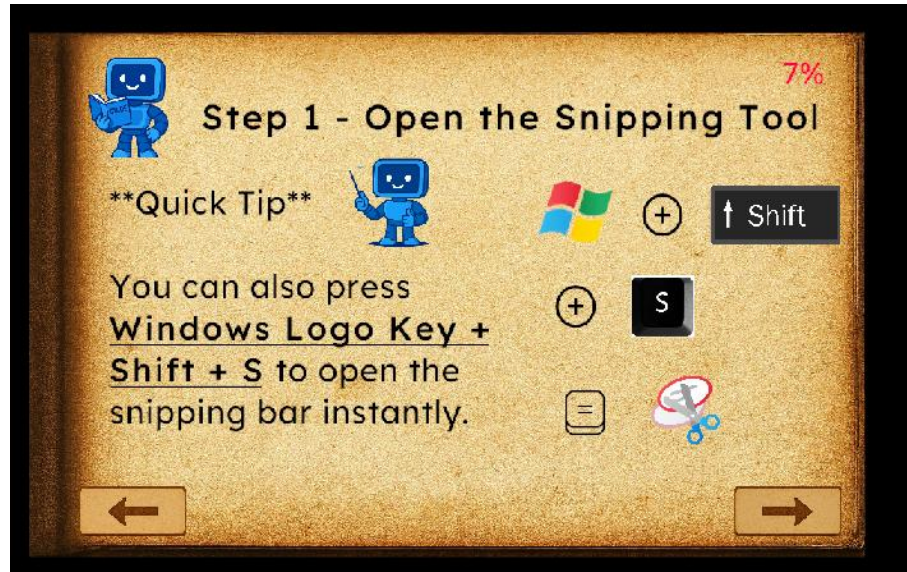
(2 Steps)**



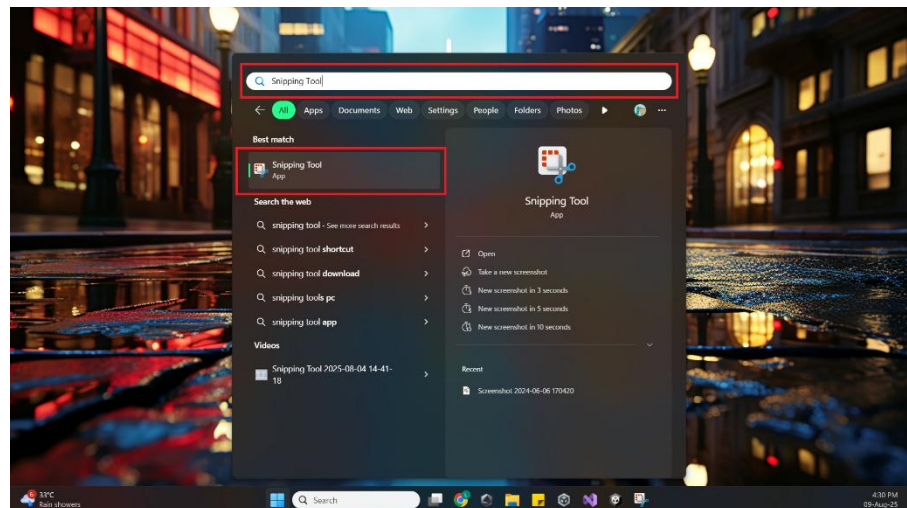
Steps Panel



Monitor Model Screen



Steps Panel



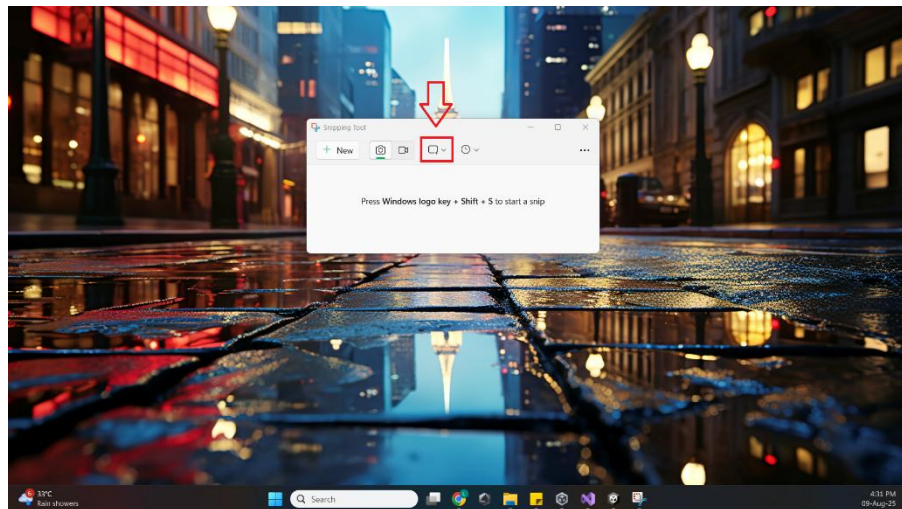
Monitor Model Screen

**Step 2 –
Choose
Your Snip
Type

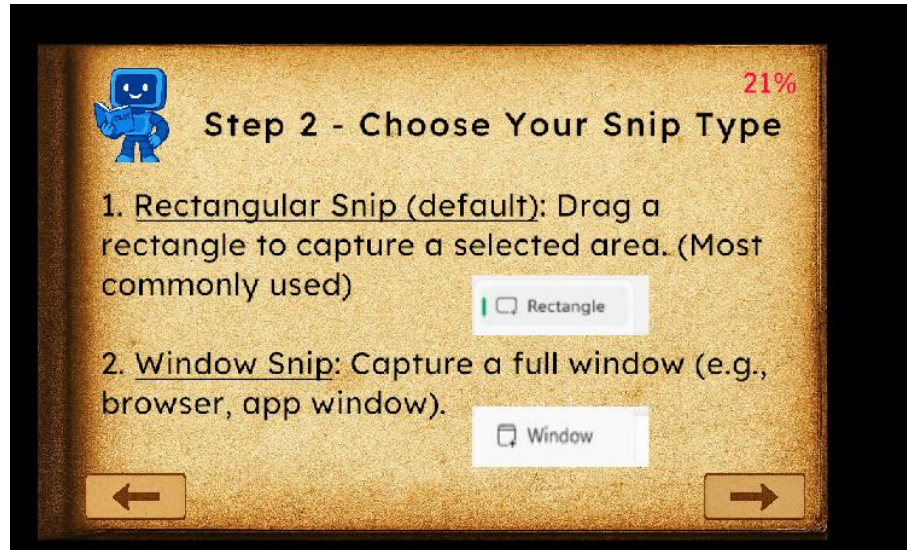
(3 Steps)**



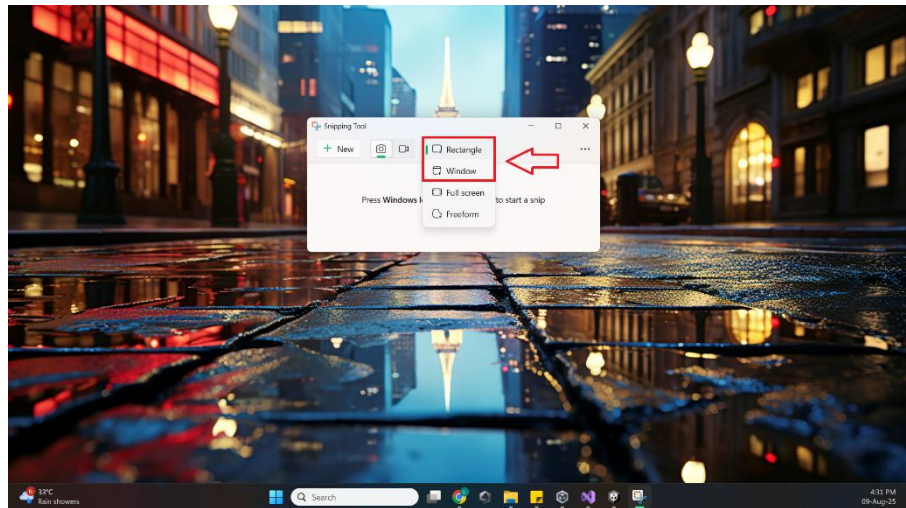
Steps Panel



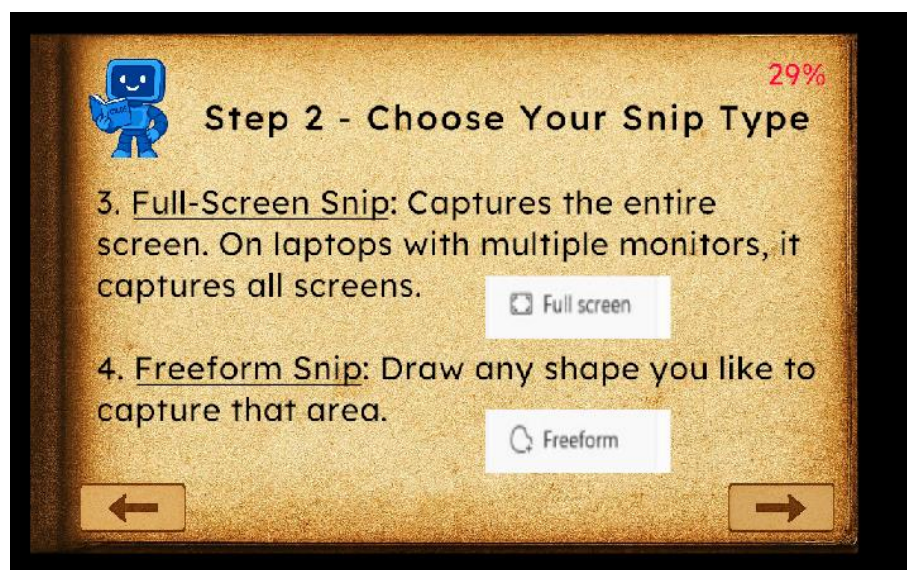
Monitor Model Screen

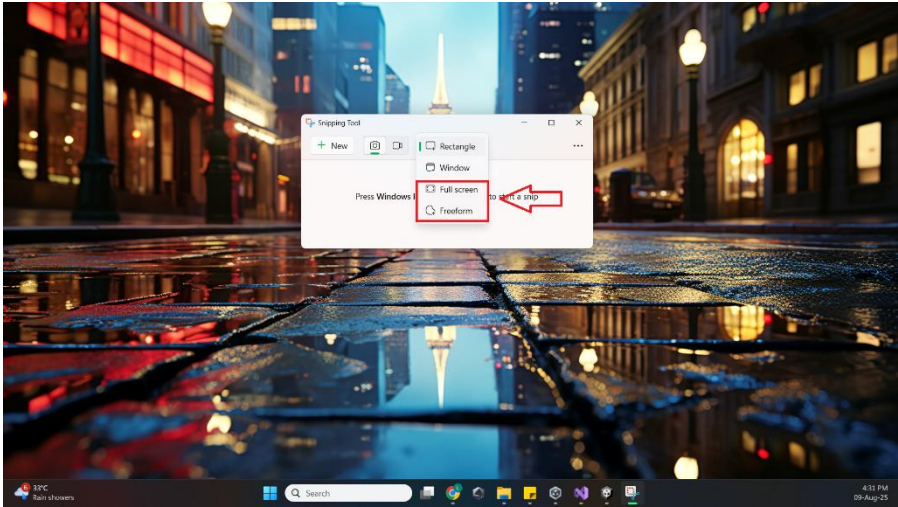
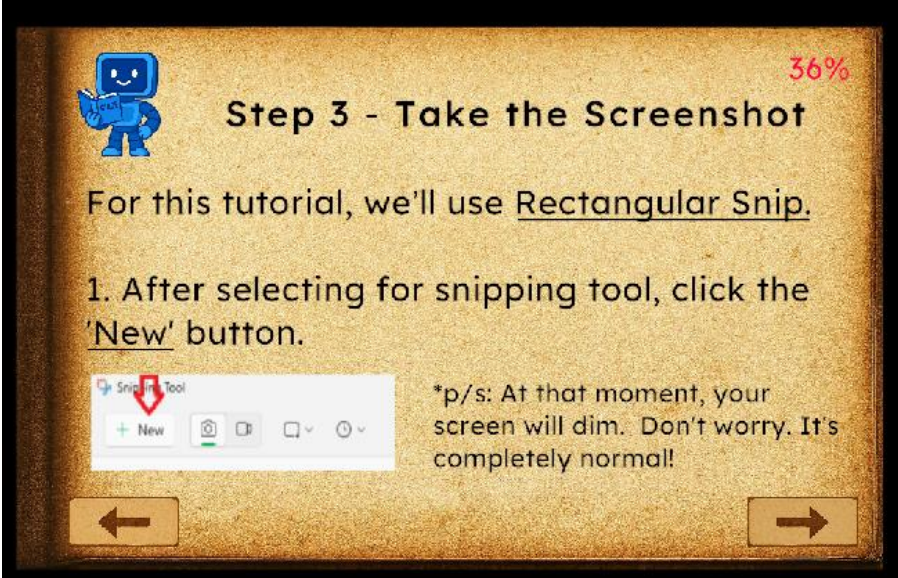
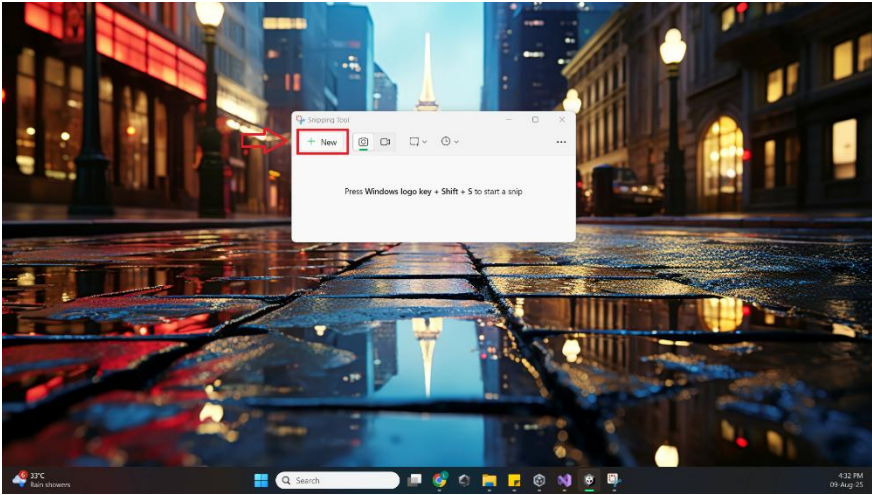


Steps Panel

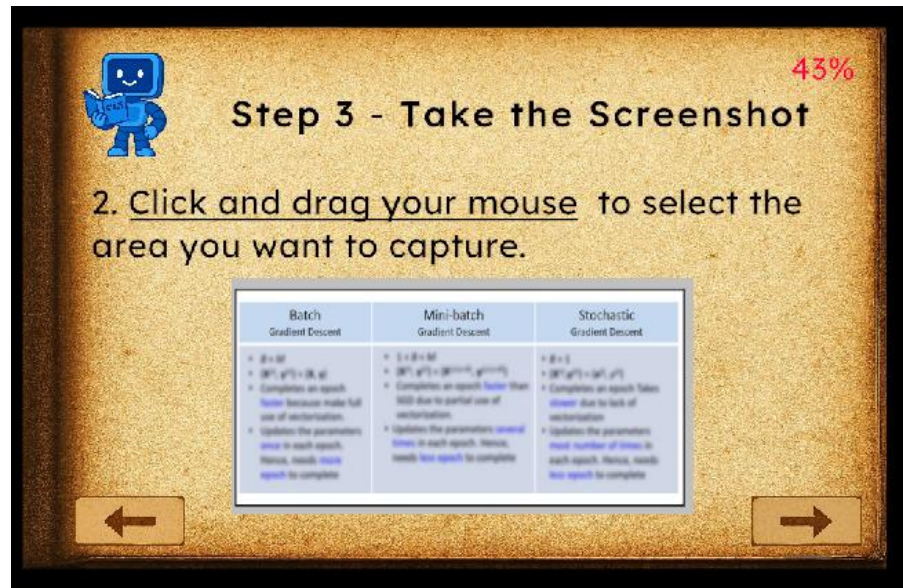


Monitor Model Screen

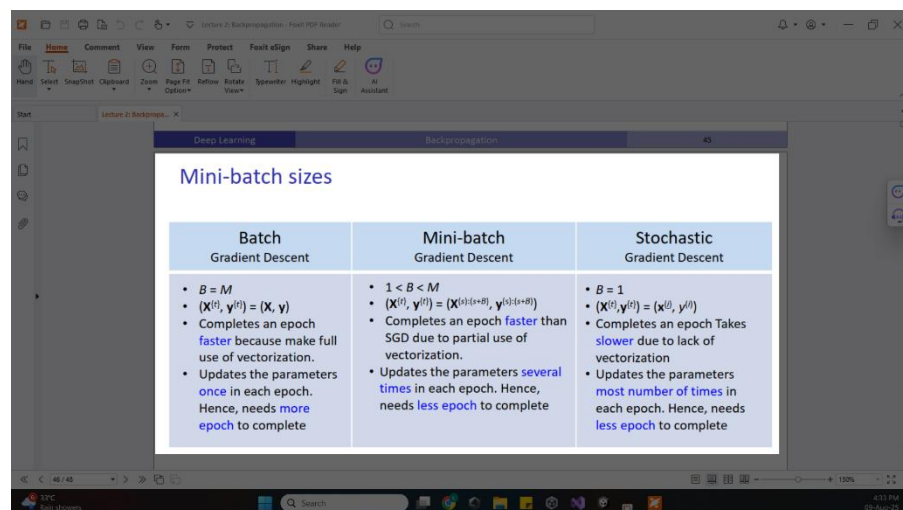


	<p style="text-align: center;">Steps Panel</p>  <p style="text-align: center;">Monitor Model Screen</p>
<p>Step 3 – Take the Screenshot</p> <p>(3 Steps)</p>	 <p style="text-align: center;">Steps Panel</p> 

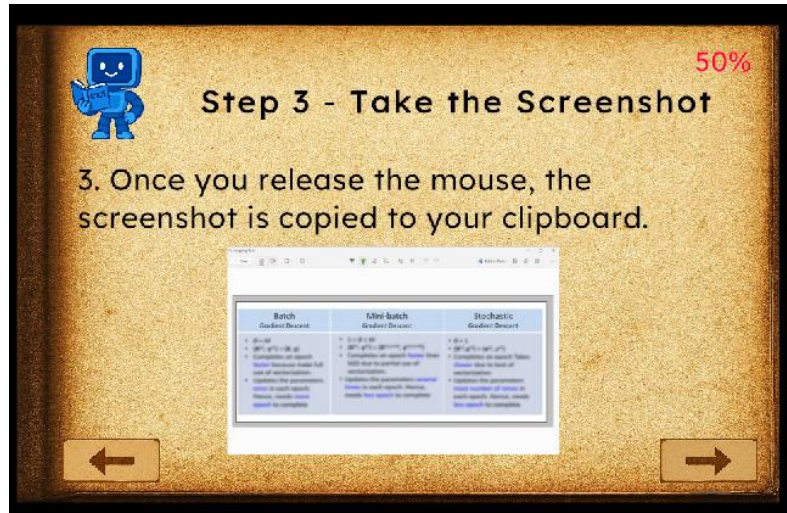
Monitor Model Screen



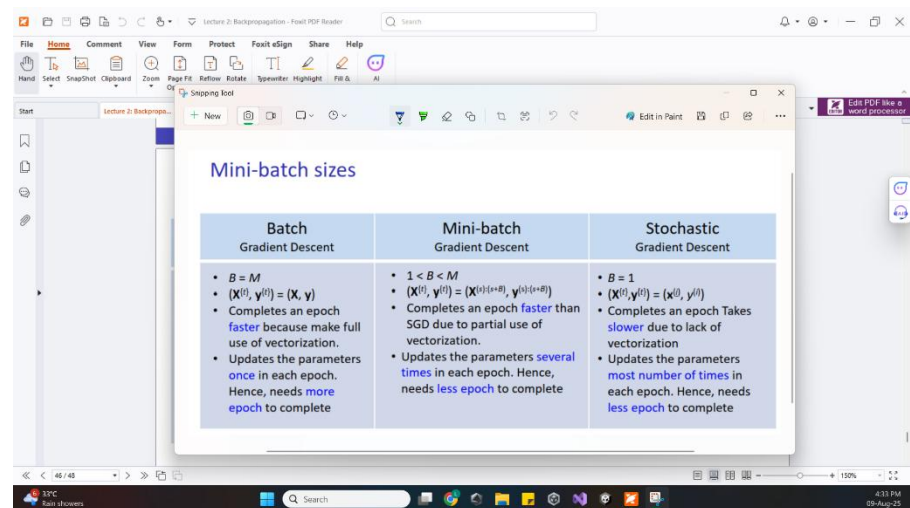
Steps Panel



Monitor Model Screen

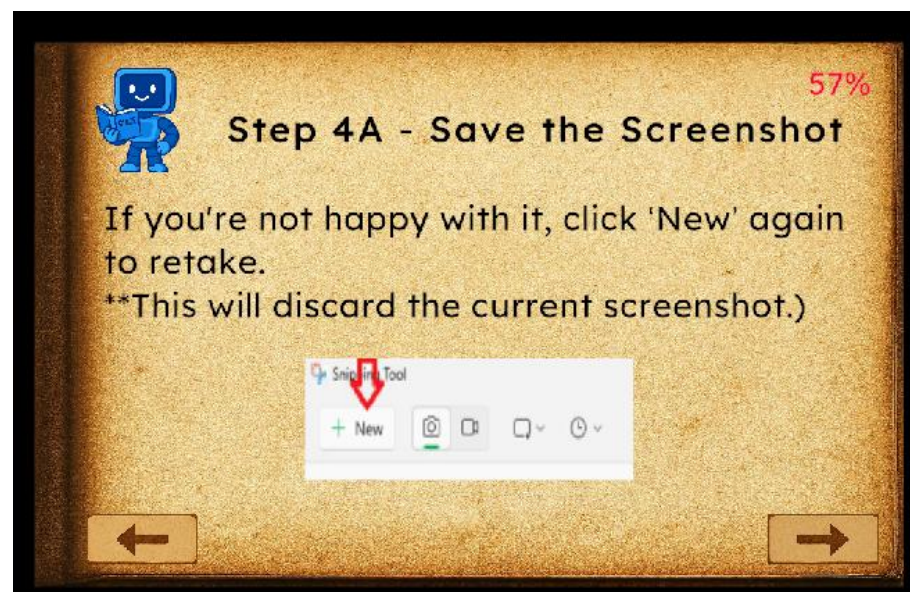


Steps Panel

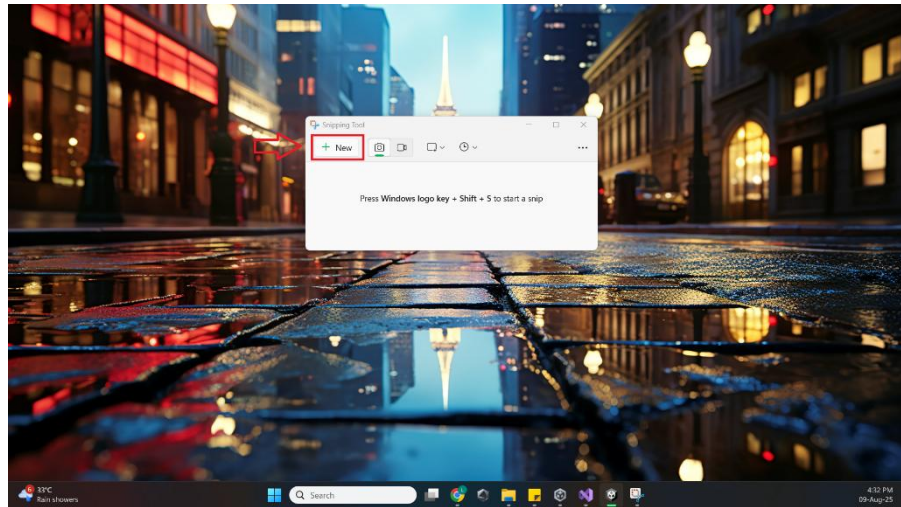


Monitor Model Screen

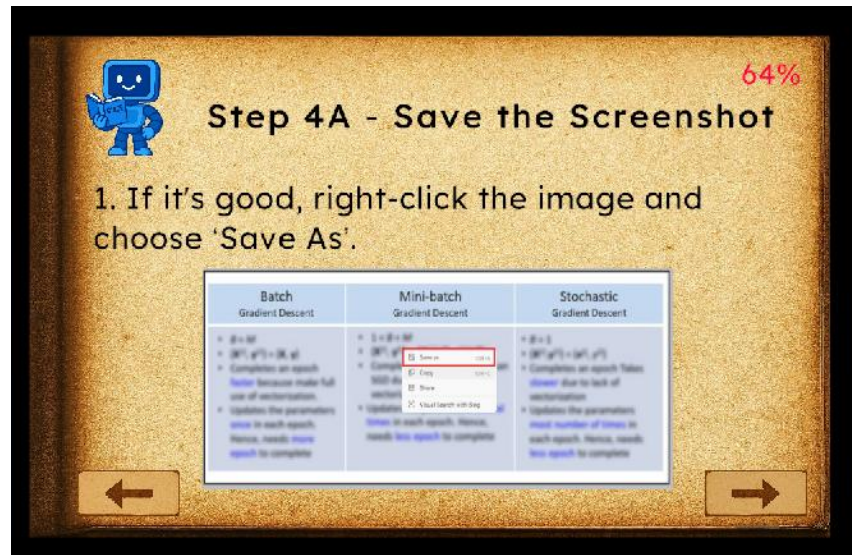
Step 4A – Save the Screenshot (4 Steps)



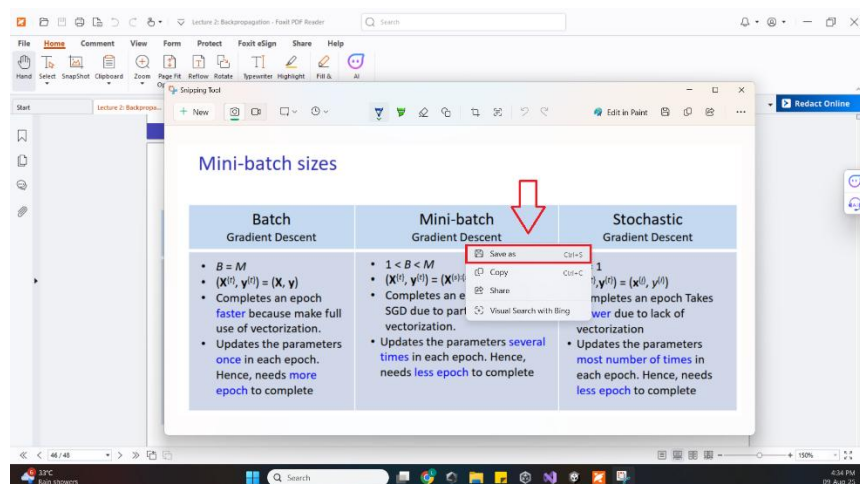
Steps Panel



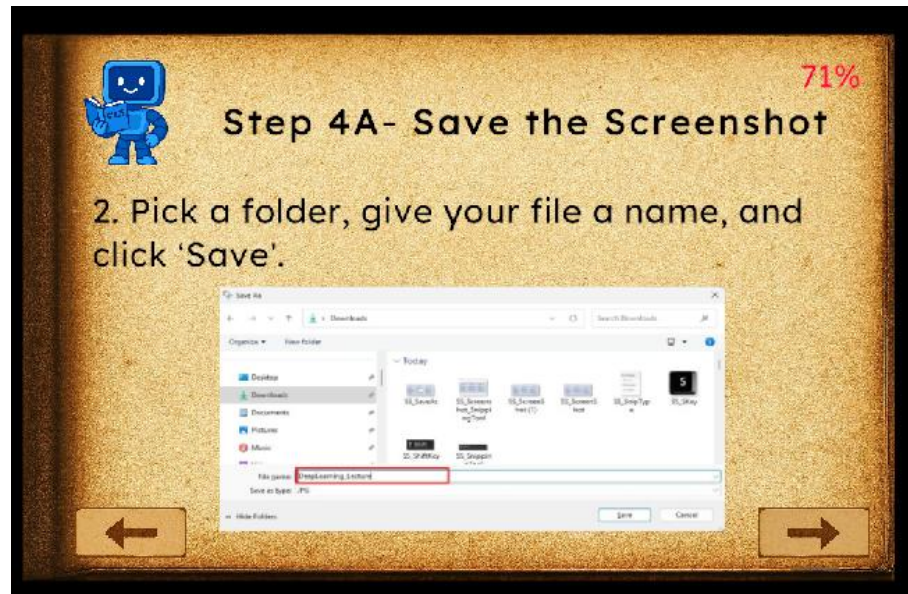
Monitor Model Screen



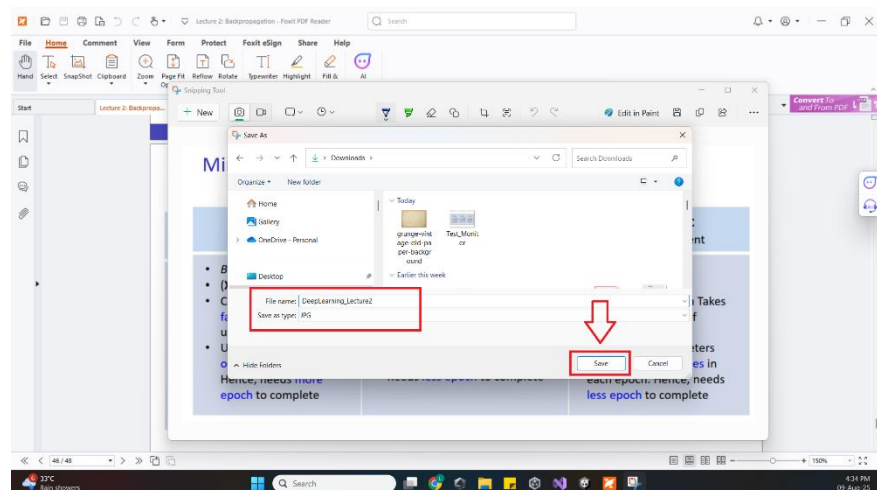
Steps Panel



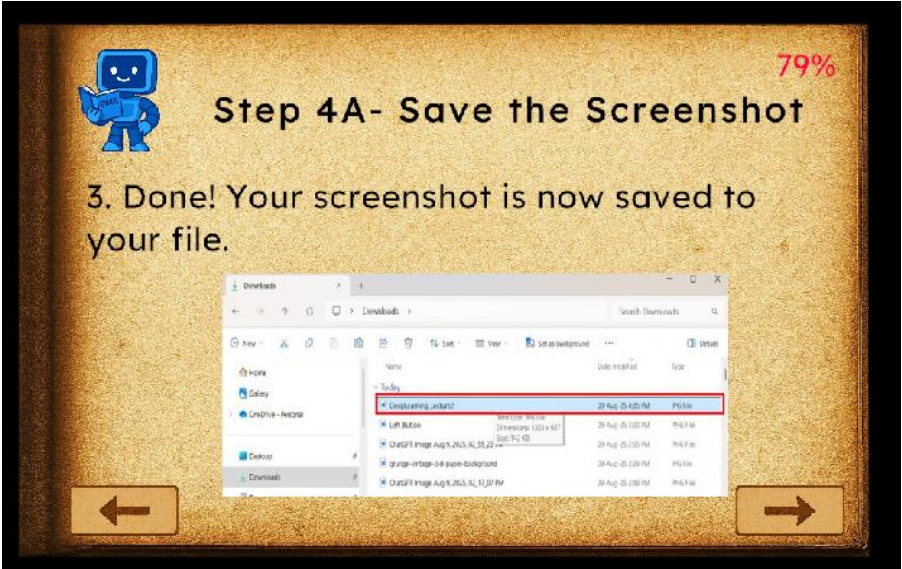
Monitor Model Screen



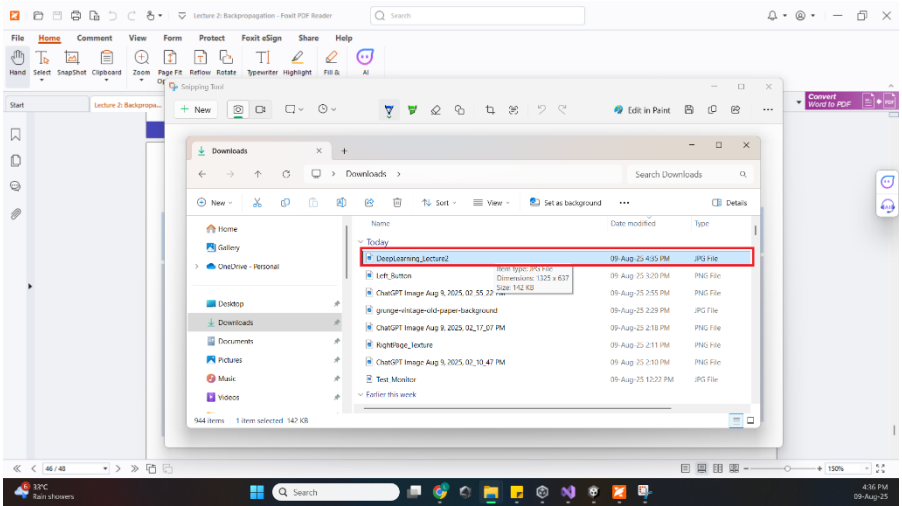
Steps Panel



Monitor Model Screen



Steps Panel



Monitor Model Screen

Step 4B – Copy and Share

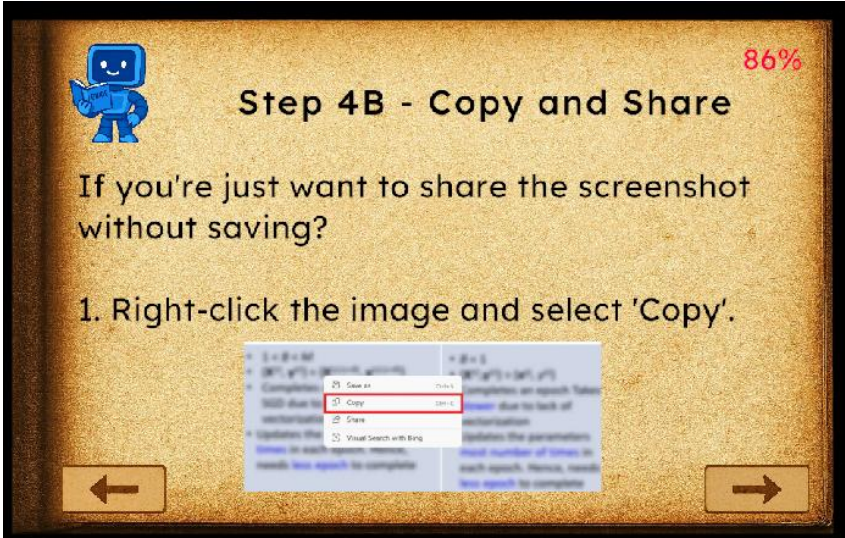
(2 Steps)

86%

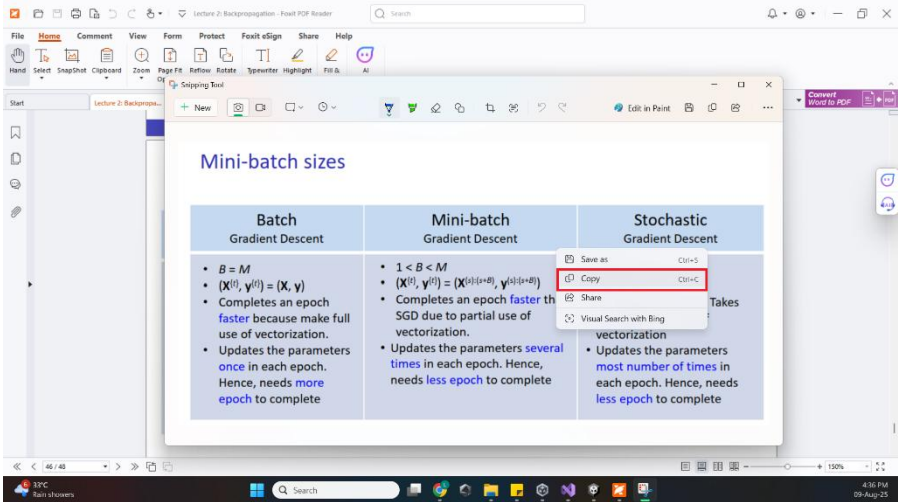
Step 4B - Copy and Share

If you're just want to share the screenshot without saving?

1. Right-click the image and select 'Copy'.



Steps Panel




Monitor Model Screen

93%

Step 4B - Copy and Share

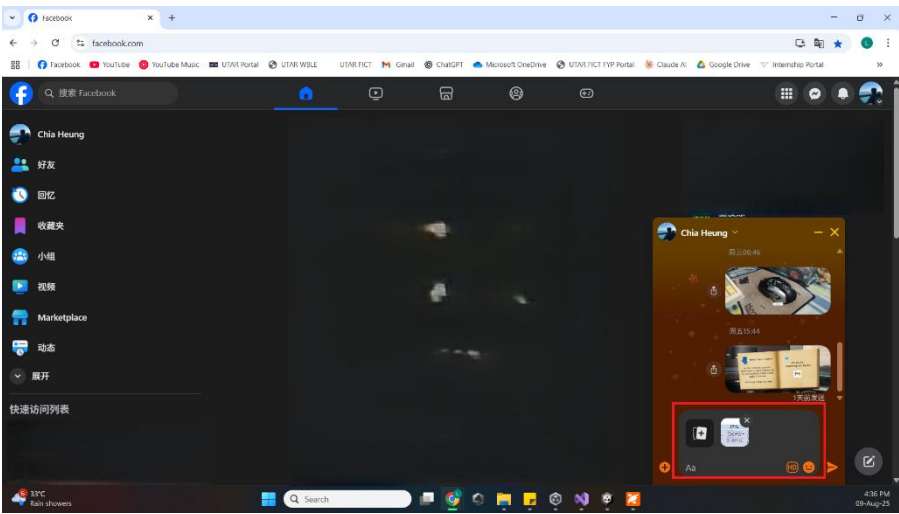

2. Then, paste it (Ctrl + V) into your destination. (E.g. Gmail, Messenger, WhatsApp, Microsoft Word,...)



Steps Panel

Bachelor of Computer Science (Honours)
Faculty of Information and Communication Technology (Kampar Campus), UTAR

E-12

	<div></div> <div>Monitor Model Screen</div>
<div>Step 5 – Tutorial Completed (1 Steps)</div>	<div></div> <div>Steps Panel</div>

