Diploma Project

INTERACTIVE VIRTUAL CHARACTER

Rafail Mitilineos

University of Cyprus
Department of Computer Science

Department of Computer Science

May 2026

UNIVERSITY OF CYPRUS

Faculty of Pure and Applied Sciences

Department of Computer Science

INTERACTIVE VIRTUAL CHARACTER

Rafail Mitilineos

Advisor: Dr Yiorgos L. Chrysanthou

The Thesis was submitted in partial fulfillment of the requirements for obtaining the Computer Science degree of the Department of Computer Science of the University of Cyprus

May 2026

Acknowledgements

I am deeply grateful to my advisor, Dr Yiorgos L. Chrysanthou , for their guidance, patience, and encouragement throughout this thesis. I would also like to thank Dr. Kleanthis Neokleous, Panayiotis Charalambous and Chrysostomos Chadjiminas for their insightful feedback and support.

Abstract

This thesis presents a campus-facing interactive virtual character displayed on a TV that detects passing students, listens to their question, and answers with natural speech and on-screen guidance. The system targets common university tasks (e.g., room locations, office hours, IT help) while emphasizing privacy, accessibility, and low latency.

We implement an end-to-end speech pipeline that converts audio to text, grounds the query on institutional documents via retrieval-augmented generation, and produces natural speech with lip-sync cues for a Unity avatar. The architecture integrates Whisper for ASR, a local LLM served through Ollama, Piper for TTS, and a lightweight state machine that governs agent behavior (Idle → Arrived → Greeting → Listening → Absent). A curated ”static facts” store complements the retrieval to reduce hallucinations by role/title.

We evaluate the prototype on representative campus FAQs, reporting end-to-end responsiveness, retrieval quality, and qualitative user feedback on clarity and usefulness.

Keywords: virtual assistant, automatic speech recognition (ASR), text-to-speech (TTS), retrieval-augmented generation (RAG), Unity

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Important abbreviations that have been used in the text and need explanation are briefly presented.

ASR	Automatic Speech Recognition
TTS	Text-to-Speech
LLM	Large Language Model
RAG	Retrieval-Augmented Generation
FSM	Finite State Machine
IVC	Interactive Virtual Character
SUS	System Usability Scale
EMA	Exponential Moving Average

1 Introduction

Universities rely on digital channels to answer routine but time-critical questions—finding rooms, checking office hours, or resolving simple IT issues. However, these channels assume that users have a device at hand, are comfortable navigating websites, and can locate the right information quickly. Public, walk-up interfaces such as kiosk displays offer an inclusive alternative, provided they deliver low-latency, privacy-respecting, and accessible interactions in noisy, high-traffic environments.

This dissertation investigates an interactive virtual character displayed on a TV that detects a passerby, listens to spoken queries, grounds answers on institutional documents, and responds with natural speech and on-screen guidance. Building on advances in automatic speech recognition, retrieval-augmented generation, and text-to-speech, the work explores a local-first pipeline integrated with a lightweight dialog state machine. The introduction defines the study's focus and scope, motivates its relevance to campus information access, states the research questions and objectives, and concludes with an overview of the thesis structure. Related work is surveyed in Chapter 2.

1.1 Aims and Objectives

Aim. Design, implement, and evaluate a *local-first*, campus-facing interactive virtual character that supports walk-up spoken queries and returns grounded, low-latency answers via natural speech and on-screen guidance, while respecting privacy and accessibility constraints.

Objectives. To achieve this aim, the thesis pursues the following objectives:

- O1 **Usability.** Achieve a mean SUS score ≥ 68 with first-time users in a naturalistic environment.
- O2 **RAG Accuracy.** Attain grounded answer correctness $\geq 80\%$ on the validated factual query set, with zero hallucinations on authority queries.
- O3 **Latency.** Achieve median end-to-end pipeline latency $\leq 10\,\mathrm{s}$ under steady-state conditions on the development hardware.
- O4 **Greeting Reliability.** Automatic presence-triggered greeting fires correctly on first approach in $\geq 90\%$ of evaluated sessions.

1.2 Research Questions

This thesis investigates whether a local-first, speech assistant can deliver fast, accurate, and privacy-aware answers that help university students. We study the following research questions:

RQ1 **Latency.** What end-to-end response times (speech → grounded answer → speech) are achievable on commodity hardware in a public setting?

RQ2 **Grounding/Factuality.** To what extent does retrieval-augmented generation over institutional documents improve answer correctness versus LLM-only responses?

RQ3 **Reliability.** Can a small curated *static-facts* store reduce authority/role hallucinations without degrading coverage or fluency?

RQ4 **Privacy and Accessibility.** Does a local/offline processing mode with captions and on-screen prompts meet stated privacy goals and basic accessibility expectations for walk-up use?

1.3 Contribution

C1 **Local-first assistant.** A complete, end-to-end system that answers spoken campus FAQs via a TV-displayed Unity avatar, integrating ASR, retrieval-augmented generation, a local LLM, and TTS with a five-state presence-aware dialog state machine.

$$\text{Idle} \rightarrow \text{Arrived} \rightarrow \text{Greeting} \rightarrow \text{Listening} \rightarrow \text{Absent}$$

C2 **Grounding strategy for institutional content.** A practical RAG pipeline for PDFs/URLs (chunking, embeddings, vector index, source tagging) with on-screen citation display, tailored to university documents and FAQs.

C3 **Static-facts safety layer.** A curated "static facts" store for authority/role queries that reduces hallucinations by overriding uncertain LLM outputs without harming coverage.

C4 **Privacy- and accessibility-aware design.** A documented local/offline processing path (no user audio leaves the device) with optional captions and readable on-screen prompts suitable for public, walk-up use.

C5 **Latency-focused engineering.** A real-time speech→speech pipeline with streaming playback and component-level timing.

C6 **Deployment checklist and artifacts.** An implementation checklist (requirements, configs, prompts), Unity integration notes, and scripts enabling replication and future extensions (presence cues, richer gestures, multi-turn memory).

1.4 Structure of the Thesis

The remainder of this thesis is organised as follows.

Chapter 2 — Related and Former Research surveys prior work on campus assistants and interactive virtual agents, grouping systems by interaction modality (chat-based versus embodied), identifying common limitations such as cloud dependence and scarce evaluation metrics, and positioning our system relative to existing approaches.

Chapter 3 — System Backend: Speech Pipeline and Knowledge Grounding describes the backend architecture across three phases: the conversational speech pipeline (Phase 1), retrieval-augmented generation over institutional documents (Phase 2), and hardware selection and deployment configuration (Phase 4).

Chapter 4 — System Frontend: Interface, Vision, and Evaluation describes the user-facing components: the presence detection pipeline and five-state kiosk FSM (Phase 3), and the user study design including task scenarios, questionnaire structure, safety guardrails, and measurable evaluation objectives (Phase 5).

Chapter 5 — Results and Findings reports measured performance across all phases, including component-level and end-to-end pipeline latency on CPU and GPU hardware, RAG retrieval and hallucination results, FSM reliability observations, and the outcomes of a user study conducted with 22 participants.

Chapter 6 — Discussion and Interpretation interprets the findings against the four measurable objectives, compares results with prior campus assistant systems, identifies limitations of the study, and proposes directions for future work including streaming TTS, larger LLM models on dedicated hardware, a visual listening indicator, and multilingual support.

1.5 Summary

To summarize, we established the need for an accessible, grounded speech interface on a public display, defined the aims and scope, stated the research questions, and previewed the contributions.

The remainder of the thesis details how this work was carried out:

- Chapter 3 describes the backend speech pipeline and knowledge grounding components
- Chapter 4 covers the vision system, avatar interface, and user study design.
- Chapters 5 and 6 report results and discuss findings, limitations, and recommendations for future work.

2 Related and Former Research

2.1 Introduction

This chapter surveys research and deployed systems relevant to smart-campus assistance and interactive virtual agents. We first clarify terminology used throughout the thesis, then group prior work by interaction modality (chat-only versus embodied/robotic systems). We compare representative approaches on capability and evaluation rigor, identify open gaps (e.g., cloud dependence, limited offline operation, scarce task-level metrics, and absence of embodied avatar interaction), and position our Interactive Virtual Character (IVC) accordingly.

2.2 Background and Definitions

Virtual Assistant (VA). A software agent that interprets user intents and returns answers or executes campus tasks (e.g., timetable queries, directions). Typical components include ASR/TTS, NLU, dialog management, and knowledge access.

Embodied / Interactive Virtual Character (IVC). A virtual or physical agent that augments dialog with social signals (voice prosody, gaze, screen prompts) to improve clarity and engagement.

Evaluation metrics. We adopt (i) *task success* (%), (ii) *time-to-answer* (s), (iii) *turn count*, (iv) *intent accuracy* (%), and (v) *user experience* via standardized scales (e.g., SUS/UMUX-Lite). Prior campus assistants often report limited dialog precision or qualitative feedback only [1, 2].

2.3 Prior Work by Category

2.3.1 Chatbot-based Campus Assistants

Gaglio et al. present a campus assistant built on IBM Watson, combining a conversational skill (intents, entities) with a content-discovery component over institutional documents [1]. They report measurable gains after targeted training: dialog precision rises substantially, and the system improves on “variation” and “blind” question categories compared to an untrained baseline. The pipeline is practical for FAQ-style queries and institutional knowledge lookup, and the methodology explicitly separates in-scope from out-of-scope questions to avoid overclaiming accuracy. However, the solution is cloud-centric and primarily evaluates dialog correctness rather than task completion time, user experience scores, or latency under real campus con-

straints. *Limitation:* vendor lock-in and a focus on chat precision leaves open questions about offline operation, privacy, responsiveness, and end-to-end task outcomes.

2.3.2 Robotic / Embodied Assistants

Nguyen et al. describe a multi-system deployment on campus, spanning a virtual assistant alongside telepresence, guide, and delivery robots [2]. The work emphasizes breadth of embodied use-cases (navigation, delivery, engagement) and reports qualitative observations about user interest and operational benefits during on-campus deployment. Architecturally, the systems combine perception and dialog modules to support various student-facing services, illustrating integration rather than model benchmarking. Yet the study does not provide controlled, quantitative metrics (e.g., task success, time-to-answer, SUS/UMUX-Lite), making it difficult to compare effectiveness across modalities or to reproduce results. *Limitation:* rich deployment breadth but limited quantitative evaluation and no strict offline guarantees hamper rigorous comparison with lightweight, on-device approaches.

2.3.3 Retrieval-Augmented Generation for Grounded Answers

A key limitation of both chatbot and robot-based campus assistants is their reliance on structured knowledge bases or vendor-managed intent models that require manual curation and retraining when information changes [1]. Retrieval-Augmented Generation (RAG), introduced by Lewis et al., addresses this by combining a neural retrieval step over a document collection with a generative language model [3]. At inference time, the top-k most relevant document chunks are retrieved and injected into the model prompt, grounding the response in up-to-date source material without retraining. This approach substantially reduces hallucination rates on knowledge-intensive tasks and allows the knowledge base to be updated simply by re-ingesting revised documents. For a university campus context, where prospectus content, timetables, and staff information change each academic year, RAG is a more maintainable alternative to fine-tuning or structured intent databases.

2.3.4 Offline and On-Device Language Models

Both prior campus systems surveyed above depend on cloud infrastructure for natural language processing [1,2]. This introduces network latency, ongoing API costs, and a privacy surface that is unacceptable in contexts where student queries may contain personally identifiable information. Recent advances in model quantization have made it feasible to run competitive language models locally on consumer hardware. Touvron et al. demonstrate that the Llama family of open-weight models achieves strong performance at 3–7B parameters [4], and tools such as Ol-

lama enable straightforward local deployment of quantized variants on laptops without GPU requirements. Whisper, introduced by Radford et al., similarly provides high-quality offline automatic speech recognition across a wide range of acoustic conditions [5]. Together, these developments make a fully offline speech-to-speech pipeline practically realizable for the first time on modest hardware.

2.3.5 Embodied Conversational Agents

Embodied conversational agents augment speech-based interaction with a visible avatar, adding social signals such as gaze, facial expression, and lip synchronisation. Cassell et al. establish that embodied agents improve user engagement and perceived naturalness compared to voice-only interfaces [6], a finding subsequently confirmed in multiple deployment contexts. Lip synchronisation in particular — matching avatar mouth movements to synthesized speech in real time — has been identified as a significant factor in perceived agent realism and user trust. Prior campus assistants have not combined embodied avatar interaction with offline speech pipelines; the present system addresses this gap by integrating SALSA lip-sync with a locally synthesized TTS stream.

2.4 Comparative Analysis

Table 2.1 contrasts representative systems by modality, offline capability, vendor dependence, and evaluation rigor.

Table 2.1: Comparison with representative smart-campus assistants.

System	Modality	Offline	RAG	Avatar	Quant. eval.
Smart Assistance [1]	Text/Voice	No	No	No	Partial
AI-Powered Univ. [2]	Robots+VA	Partial	No	No	None
Our AI Campus Assistant	**Embodied**	**Yes**	**Yes**	**Yes**	**Full**

2.5 Identified Gaps and Our Positioning

Across prior work, four common limitations emerge. First, cloud dependence harms privacy and introduces latency that is unacceptable for a kiosk deployed in a public campus space [1, 2]. Second, existing systems rely on structured knowledge bases or vendor intent models that require manual retraining when university information changes [1]. Third, quantitative evaluation with standardised usability scales (SUS, UMUX-Lite) and task-level metrics is largely absent [2]. Fourth, no prior campus assistant combines offline operation with an embodied avatar and proactive presence-triggered greeting [6, 7].

We address all four gaps: the system runs entirely offline on consumer hardware [4, 5], uses RAG over unstructured PDFs for maintainable grounding [3], reports a full quantitative evaluation including SUS and task-level accuracy, and combines an animated avatar with automatic presence detection and a five-state dialog FSM.

2.6 Summary

Existing campus assistants demonstrate feasibility but often depend on cloud services or lack controlled evaluations. Our offline, lightweight Interactive Virtual Character combines RAG-grounded answers, an embodied avatar with lip synchronisation, and proactive presence-triggered greeting, capabilities absent in combination from prior campus assistant systems [1,2] and evaluates all of these against standardised usability and accuracy metrics.

3 System Backend: Speech Pipeline and Knowledge Grounding

This chapter describes the backend components of *Our AI Campus Assistant* across three phases: the conversational speech pipeline (Phase 1), the retrieval-augmented generation layer (Phase 2), and the hardware and deployment configuration (Phase 4). The phase roadmap for the full system is shown in Figure 3.1.

3.1 System Overview

We structure the implementation into five phases that progressively build *Our AI Campus Assistant* into an offline, lightweight, accurate, and student-friendly system. Phases 1, 2, and 4 are described in this chapter; Phases 3 and 5 are described in Chapter 4.

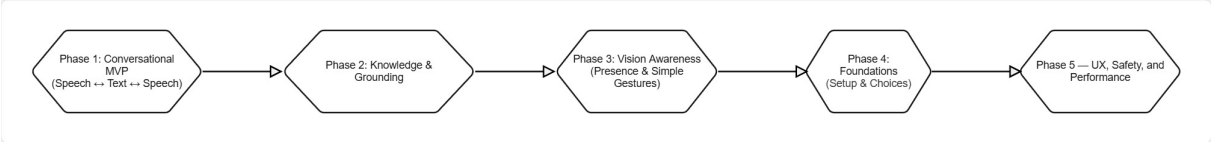

Figure 3.1: Phase roadmap for *Our AI Campus Assistant*.

1. **Phase 1 – MVP:** Speech→text→(KB/LLM)→speech end-to-end for top FAQs.
2. **Phase 2 – Grounding:** Connect to university data using Retrieval-Augmented Generation (RAG).
3. **Phase 3 – Vision:** Presence detection and basic gestures tied to dialog states.
4. **Phase 4 – Setup:** Choose LLM and finalize target hardware.
5. **Phase 5 – UX/Safety/Performance:** Natural interactions with consistent timing, guardrails, and tuning.

3.2 Phase 1 — Conversational MVP (Speech↔Text↔Speech)

Goal Achieve end-to-end conversation on a single machine in two modes: Text→Text (T→T) and Speech→Speech (S→S), fully local and private.

System Sketch

$$\text{Mic Audio} \xrightarrow{\text{ASR}} \text{User Text} \xrightarrow{\text{LLM (local)}} \text{Assistant Text} \xrightarrow{\text{TTS}} \text{Voice Reply (streamed)}$$

Implementation Summary We implemented a small FastAPI service on a laptop that exposes `/ask` for T→T and a streaming endpoint for S→S. The service calls a locally served LLM via Ollama for answers, Whisper converts speech to text, and Piper synthesizes natural-sounding audio for responses. A minimal Windows mic client streams audio to the server and plays back the TTS stream, giving a real-time feel. Both flows operate fully offline.

Technologies

- **Platform:** Windows laptop with WSL (Ubuntu).
- **Runtime:** Python, FastAPI + Uvicorn.
- **LLM:** Llama 3.2–3B Instruct via Ollama (local).
- **ASR:** Whisper.
- **TTS:** Piper neural voice.
- **Client:** Simple Windows mic app (`sounddevice`) with streaming playback.

Notes Local/offline execution preserves privacy and lets us swap components later with minimal refactoring.

Problems Encountered & Solutions

Library/Environment Conflicts. Differing versions of `torch`, `fastapi`, and audio deps across machines caused import/runtime errors.
Solution: Created a dedicated `venv`; pinned versions in `requirements.txt` to ensure reproducibility.

Model Size & Resource Constraints. Llama 3.2–3B stressed RAM/VRAM and degraded performance.
Solution: Used quantized variants; enabled CPU-only inference when necessary and verified acceptable latency.

Windows↔WSL Differences. Path handling and audio backends behaved inconsistently.
Solution: Standardized directory layout; introduced environment variables for OS-specific paths and audio devices.

Acceptance Criteria (MVP)

- The system can complete a full speech-to-speech conversation reliably for common FAQ questions.
- Response time for short questions remains under 10 seconds on average.
- The system works fully offline without requiring internet access or cloud services.

3.3 Phase 2 — Grounding with RAG

Goal Ground answers in up-to-date university documents and pages, returning citations/sources alongside responses.

Pipeline

1. **Ingest:** PDFs/URLs → text extraction → chunking (e.g., 700–1,000 chars, overlap) with source tags.
2. **Embed:** Compute sentence embeddings (CPU) and store vectors in a local vector DB.
3. **Retrieve:** Embed the user query and fetch top-k nearest chunks.
4. **Generate:** Construct a strict prompt with retrieved chunks; query the local LLM; speak with Piper; show sources on screen.

Queries Used for Validation

- "Who is the chairperson and what is the department contact email/phone?"
- "When do Fall 2025–26 classes start at UCY CS?"
- "How many ECTS are required for a Bachelor's and for a Master's?"

Problems Encountered & Solutions

Hallucinations on Authority Queries. The model occasionally invented names/roles when asked about leadership positions.
Solution: Added a curated **static-facts store** for authoritative fields (leadership, contact points, office hours). At answer time we cross-check RAG output against this store and override uncertain claims, which eliminated these hallucinations in testing.

Acceptance Criteria (RAG)

- Top-k retrieval hit-rate on a labeled Q&A set $\geq$ *80*% (e.g.,@k=5).
- Answers display source citations; authority facts match the static-facts store.

- Offline operation maintained (local embeddings, local index, local LLM).

Artifacts

- `kb/` (cleaned text, chunks, metadata), `kb/index/` (vectors), `kb/eval.jsonl` (labeled Q&A).
- Service endpoints: `/ask` (T→T), `/stream_speech` (S→S), `/grounded_ask` (RAG).

3.4 Phase 4 — Setup: LLM Selection and Hardware Finalization

Goal Transition from a laptop prototype to a fixed, unattended kiosk installation. This phase covers hardware selection, software configuration for automatic startup, and validation that all components run reliably in a public environment without manual intervention.

Hardware Requirements The kiosk hardware must support concurrent execution of all pipeline components: Whisper ASR, the local LLM via Ollama, Piper TTS, OpenCV face detection, and the Unity application. Key requirements are:

- **CPU/RAM:** A minimum of 16 GB RAM is required for the 3B quantized model; 32 GB is recommended to accommodate a 7B or 8B model without page faults.
- **GPU:** A mid-range discrete GPU (e.g., NVIDIA RTX series with $\geq$6 GB VRAM) reduces LLM inference latency from $\sim$8 s to $\sim$2 s for a 3B model and enables larger models.
- **Storage:** An NVMe SSD ensures fast cold-start of the model ($<$5 s load time).
- **Webcam:** A wide-angle USB webcam positioned at head height (approximately 1.6 m), angled slightly downward to frame approaching users at distances of 0.5–2 m.
- **Display:** A large-format TV or monitor ($\geq$43") mounted at standing eye level to align with the avatar's gaze angle and remain readable at 1–3 m.
- **Microphone:** A directional or cardioid USB microphone placed near speaking distance to suppress ambient hallway noise.

LLM Selection The prototype uses Llama 3.2–3B Instruct (`q4_K_M` quantization via Ollama), chosen for its small footprint and fully offline operation on CPU. For the target kiosk hardware with a discrete GPU, a larger model is feasible and preferable for answer quality:

- **3B (current):** Suitable for CPU-only deployment. Median LLM response latency $\approx$5–8 s on CPU. Good for short factual FAQs; occasionally produces terse or incomplete answers

for complex queries.

- **7B/8B (target):** Recommended for GPU-equipped hardware. Latency drops to ≈1–3 s with 6 GB VRAM. Noticeably better coherence and completeness for multi-sentence answers.

The modular architecture means the model is swapped by changing a single variable (`LLM_MODEL` in `assistant_api.py`) with no other code changes required.

Software Configuration for Unattended Operation For a public deployment the system must start automatically on power-on and recover from crashes without manual intervention:

- **Ollama:** Register as a Windows service or add to Task Scheduler at login, targeting the selected model variant.
- **FastAPI (uvicorn):** Launch via Task Scheduler or a startup batch file in the WSL environment: `uvicorn assistant_api:app --host 0.0.0.0 --port 8000`.
- **Vision sender:** Launch via Task Scheduler in native Windows PowerShell: `python vision_sender.py --api http://127.0.0.1:8000 --min-area 0.01`.
- **Unity application:** Build as a standalone Windows executable; configure to launch fullscreen at startup via Task Scheduler or Startup folder shortcut.
- **RAG knowledge base:** Pre-populate ChromaDB with current university PDFs and URLs before deployment. Schedule a periodic re-ingest (e.g., weekly) to reflect document updates.

Startup Sequence and Dependencies Components must start in dependency order:

$$\text{Ollama} \rightarrow \text{FastAPI (uvicorn)} \rightarrow \text{vision\_sender} \rightarrow \text{Unity}$$

A 10-second delay between Ollama and FastAPI startup is sufficient for the model to load. FastAPI performs a warmup request to Ollama and a warmup TTS call on startup (`_startup_warmup`), so the first real user interaction does not incur cold-start latency.

Problems Encountered & Solutions

WSL Network Accessibility from Unity. Unity runs on Windows and must reach FastAPI running in WSL on `0.0.0.0:8000`. In some WSL2 configurations the WSL IP is not `127.0.0.1`. *Solution:* Binding uvicorn to `0.0.0.0` and using `127.0.0.1` from Unity worked correctly on the test machine. For deployments where this fails, retrieve the WSL2 IP with `hostname -I` inside WSL and set `baseUrl` in Unity accordingly.

Acceptance Criteria (Setup)

- All components start automatically on power-on without manual intervention.
- First greeting fires within 30 seconds of system boot.
- System operates continuously for a full working day (8 hours) without requiring restart.
- The knowledge base reflects the current semester’s documents at deployment time.

4 System Frontend: Interface, Vision, and Evaluation

This chapter describes the implementation of the user-facing components: the presence detection pipeline and dialog-state FSM (Phase 3), and the user study design and evaluation methodology (Phase 5). These phases build on the backend established in Chapter 3 to produce a complete, evaluated kiosk system.

4.1 Phase 3 — Vision: Presence Detection and Dialog-State Awareness

Goal Give the kiosk a sense of physical awareness: detect when a person approaches, determine whether they are paying attention, and automatically initiate a greeting without any manual input. This transforms the assistant from a passive Q&A terminal into a proactive kiosk experience that responds to people's presence in front of it.

System Sketch

$$\text{Webcam} \xrightarrow{\text{Face Detection}} (\textit{present}, \textit{attention}, \textit{conf}) \xrightarrow{\text{HTTP POST}} \text{FastAPI}$$
$$\xrightarrow{\text{HTTP GET ()}} \text{Unity} \rightarrow \text{Greeting / Listening}$$

Architecture The vision system is implemented as a three-process pipeline, separated by design so that each component can be restarted or replaced independently.

- **Vision process (Windows):** `vision_sender.py` runs natively on Windows, reads webcam frames, performs face detection, and pushes presence/attention state to FastAPI at 10 Hz.
- **API server (WSL):** FastAPI stores the latest vision state in memory.
- **Unity FSM:** `KioskPresenceController` polls `/vision/state` every 250 ms and feeds the result into a five-state finite state machine that governs when to greet, listen, and return to idle.

Face Detection Face detection uses OpenCV's built-in Haar cascade classifier, which runs entirely on CPU. The original implementation used MediaPipe; this was replaced after MediaPipe 0.10+ dropped the `mp.solutions` API on Windows, causing an `AttributeError` regardless of version.

Two signals are derived from each detected face:

- **Presence:** the face bounding-box area must occupy at least 2% of the total frame area. This filters distant or partial detections and avoids false positives from people walking in the background.
- **Attention:** the center of the face bounding box must fall within 30% of the frame center both horizontally and vertically. A face far to one side suggests the person is passing the kiosk rather than engaging with it.

Both signals are processed through two filters before transmission:

1. **Debouncing:** a signal requires three consecutive positive frames to flip ON and six consecutive negative frames to flip OFF, preventing transient lighting changes or brief occlusions from destabilizing the state.
2. **Exponential moving average (EMA):** with $\alpha = 0.5$, continuous confidence values in $[0, 1]$ are produced and sent alongside the boolean signals, allowing the FSM to apply a minimum confidence threshold.

Kiosk Finite State Machine `KioskPresenceController` implements five states, shown in Figure 4.1.

Idle. No face detected. The kiosk waits silently, suitable for a screensaver or attract-loop animation.

Arrived. A face is detected. An attention-hold timer begins incrementing each frame. If the user looks away the timer resets. Only after **1.2 seconds** of continuous attention with confidence ≥ 0.55 does the FSM proceed to Greeting. This threshold prevents greeting passersby who merely glance at the screen.

Greeting. `SpeakGreeting()` is called, which fetches a pre-rendered WAV file from `/speak_wav` and plays it through the avatar's `AudioSource`. SALSA reads this source to drive real-time lip synchronisation. The FSM waits in Greeting until `IsBusy()` first becomes `true` (confirming the audio download completed and playback started) and then becomes `false` (confirming playback ended). This two-stage guard ensures the user always hears the complete greeting before voice input is accepted.

Listening. The kiosk accepts push-to-talk input (hold key `V`). Key presses are blocked in all other states. The FSM freezes entirely while `IsBusy()` is `true` (i.e., during recording,

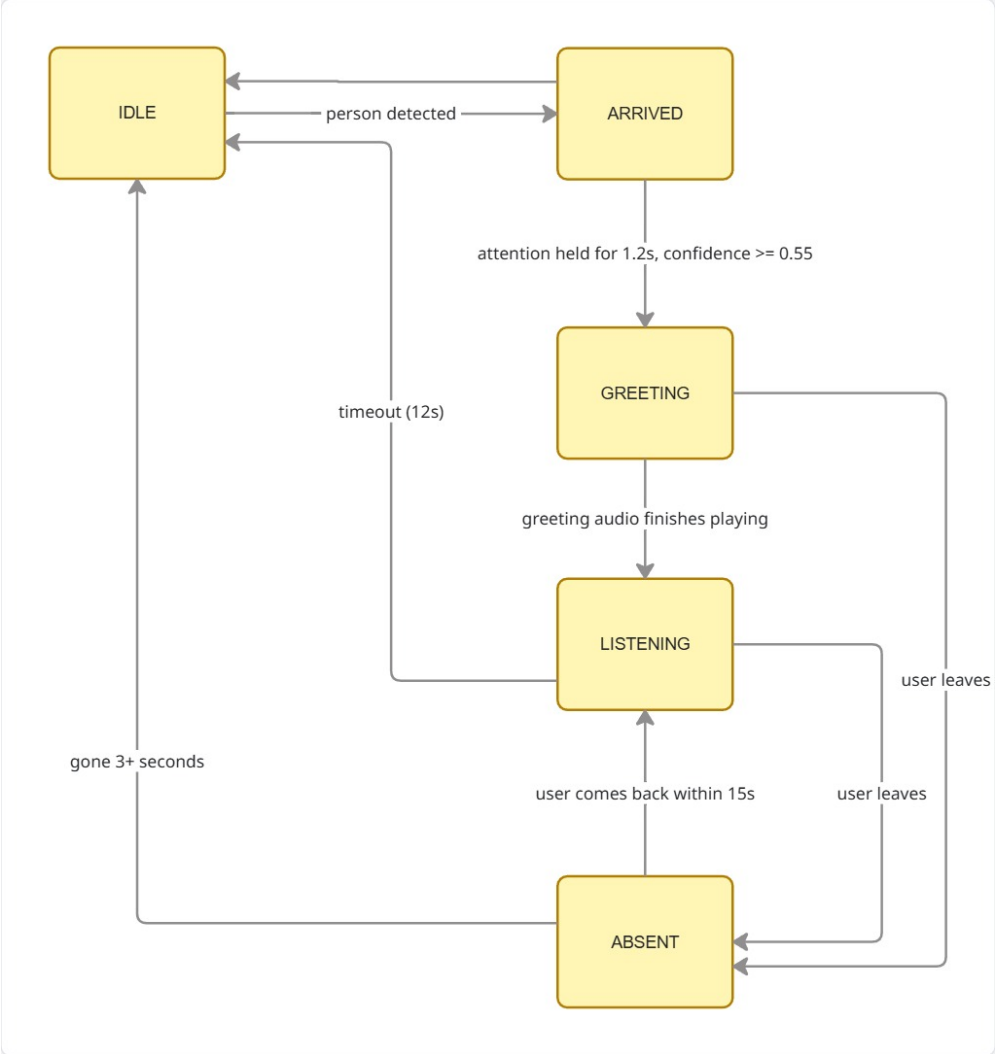

Figure 4.1: Kiosk presence FSM. Transitions are governed by polled vision state and `IsBusy()` from the audio pipeline.

LLM inference, or response playback), preventing false Absent transitions caused by transient face-detection loss during the 5–8 second response cycle. A 12-second inactivity timeout returns the FSM to Idle and rearms the greeting for the next visitor.

Absent. The face has left the frame. If the user returns within 15 seconds of the last voice response the FSM restores the Listening state directly, enabling multi-turn follow-up questions without a repeated greeting. If the user is absent for more than 3 seconds the greeting is rearmed and the FSM transitions to Idle, ready for the next visitor.

Process Separation and WSL Constraint A key architectural constraint is that FastAPI runs inside WSL2, which does not expose USB devices. The webcam is therefore inaccessible from the WSL environment. The vision process runs natively on Windows, where USB cameras are available, and communicates with FastAPI over the WSL–Windows localhost bridge. This

boundary is also architecturally beneficial: the three processes (detection, API, Unity) are decoupled and can each be updated or restarted independently.

A `stale` flag on `/vision/state` provides safe failure: if the vision process crashes or stops sending, the API marks the state as stale after 2 seconds and forces `present = false`. Unity interprets a stale response as absent, so the kiosk degrades gracefully to Idle rather than remaining stuck in an incorrect state.

Technologies

- **Face detection:** OpenCV Haar cascade (CPU-only, no external model download).
- **Vision process:** Python with `opencv-python` and `requests`, Windows native.
- **State transport:** FastAPI REST (`/vision/state_ingest`, `/vision/state`).
- **Unity FSM:** `KioskPresenceController.cs`, polling via `UnityWebRequest` at 4 Hz.
- **Lip sync:** SALSA Suite, driven by the avatar's runtime `AudioSource`.

Problems Encountered & Solutions

MediaPipe Incompatibility on Windows. MediaPipe 0.10+ removed the `mp.solutions` API, causing an `AttributeError` on Windows regardless of which version was installed.
Solution: Replaced `vision.py` with an OpenCV Haar cascade implementation requiring only `opencv-python`. Detection quality was comparable for kiosk distances (0.5–2 m) and the dependency was eliminated entirely.

WSL2 Has No Webcam Access. Running `vision_sender.py` inside WSL2 produced a V4L2 error because `/dev/video0` does not exist in the WSL environment.
Solution: Moved the vision process to native Windows PowerShell. FastAPI remains in WSL and both communicate over localhost HTTP, which crosses the WSL/Windows boundary without special configuration.

Timing Bugs. Two timing issues emerged during testing. First, the Greeting state transitioned to Listening before the greeting audio had started (the `/speak_wav` download takes approximately 1.5 seconds), so voice input was accepted while the avatar was still silent. Second, transient face-detection loss during LLM inference caused Listening to transition to Absent mid-session, making the push-to-talk key unresponsive.
Solution: Replaced the single `IsBusy()` guard in `TickGreeting()` with a two-stage check: the FSM first waits for `IsBusy()` to become `true` (audio confirmed started), then waits for it to become `false` (audio confirmed finished). In `TickListening()`, the FSM now returns immediately if `IsBusy()` is `true`, freezing all state transitions and resetting the inactivity timer during active pipeline operation.

Acceptance Criteria (Vision)

- Presence detected within 1 second of a person standing in front of the kiosk at 0.5–2 m.
- Greeting fires once per visit; repeated greetings do not occur for the same continuous session.
- Push-to-talk functions reliably across multi-turn conversations without FSM interruption from face-detection noise.
- Safe failure: if the vision process stops, the kiosk returns to Idle within `stale_after` seconds.

4.2 Phase 5 — UX, Safety, and Performance

Goal Evaluate the assembled system with real users in a naturalistic environment, measure usability and interaction quality against defined objectives, and identify remaining failure modes and improvement opportunities.

User Study Design

A task-based user study was designed to evaluate four distinct system behaviours: automatic presence detection and greeting, factual RAG accuracy, graceful handling of out-of-scope queries, and ASR robustness under ambient noise. All participants completed the same four tasks in the same order, enabling direct comparison across sessions.

Participants A target of 20–25 participants was set, consistent with guidance for evaluative prototype studies [8]. Participants were recruited opportunistically in a café environment on the university campus, reflecting the naturalistic, semi-public context for which the kiosk is intended. No domain expertise or prior kiosk experience was required.

Task Scenarios Each participant received a printed scenario card before each task and was asked to interact with the system as naturally as possible. The four tasks were:

1. **Approach and greeting.** The participant walks up to the kiosk without pressing anything and waits. This task evaluates whether the automatic presence detection and greeting pipeline fires correctly and within an acceptable time.
2. **Factual RAG question.** The participant asks how many ECTS credits are required to complete a Bachelor's degree in Computer Science at UCY. The correct answer (240 ECTS) is grounded in the ingested department prospectus. This task evaluates RAG retrieval accuracy and answer correctness.

3. **Out-of-scope question.** The participant asks about the weather forecast for Nicosia. This task evaluates graceful degradation: the system should acknowledge it does not have that information rather than hallucinating an answer.
4. **Noise robustness.** The participant asks when Fall semester classes begin at UCY, speaking at normal conversational volume without leaning towards the microphone or raising their voice. The correct answer (September 8, 2025) is grounded in the Academic Calendar section of the ingested department prospectus. This task evaluates ASR performance under ambient café noise conditions using a different factual query from Task 2, providing two independent RAG accuracy data points while simultaneously testing noise robustness.

Questionnaire After completing all four tasks, participants filled in a structured questionnaire comprising five sections:

- **Section A — Demographics:** age range, gender, study or work affiliation, frequency of voice assistant use, and prior kiosk experience.
- **Section B — System Usability Scale (SUS):** the standard ten-item SUS on a five-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree). SUS was chosen because it is technology-agnostic, widely validated, and produces a single comparable score on a 0–100 scale.
- **Section C — Presence and Greeting:** four custom items evaluating greeting timing, wait time, naturalness, and readiness cues (1–5 Likert scale).
- **Section D — Answer Accuracy and Response Quality:** five custom items evaluating ASR comprehension, answer accuracy, TTS voice clarity, response latency acceptability, and unknown-answer handling (1–5 Likert scale).
- **Section E — Overall Impression:** campus deployment intent (1–5 Likert), comparison against web search (5-point scale from *Much Worse* to *Much Better*), and four open-ended questions covering problems encountered, questions the system struggled with, most liked aspects, and suggested improvements.

Session Protocol Each session lasted approximately 13 minutes: roughly 5 minutes for the four tasks and 8 minutes for the questionnaire. The moderator read a standardised briefing before each session and handed scenario cards one at a time, collecting each before providing the next. Participants were not told the correct answers to Tasks 2 and 4 at any point during the session.

Safety and Guardrails

Two mechanisms were implemented to prevent harmful or misleading outputs in a public deployment context.

The **static-facts override** intercepts queries matching authority-related patterns (e.g. chairperson, contact details) before they reach the RAG pipeline and substitutes verified answers from a curated store. This eliminates hallucinations on the category of queries most likely to cause user harm if answered incorrectly.

The **RAG system prompt** instructs the LLM to answer only from the provided context chunks and to explicitly state when it does not have sufficient information rather than speculating. This prompt-level guardrail is the primary mechanism for graceful degradation on out-of-scope queries.

Performance Objectives

The following measurable objectives were defined for Phase 5 evaluation:

- **O1 — Usability:** Mean SUS score ≥ 68 (industry average [8]).
- **O2 — RAG Accuracy:** Correct answer rate $\geq 80\%$ on the validated factual query set.
- **O3 — Latency:** Median end-to-end pipeline latency ≤ 10 s under steady-state conditions on the development hardware.
- **O4 — Greeting Reliability:** Greeting fires on first approach in $\geq 90\%$ of evaluated sessions.

Acceptance Criteria

- Mean SUS score ≥ 68.
- No participant reports a hallucinated authority fact (chairperson name, contact details).
- Majority of participants ($> 50\%$) rate the assistant at least as useful as a conventional web search for the evaluated tasks.
- ASR comprehension mean ≥ 3.5 out of 5 under café noise conditions.

5 Results / Findings

5.1 Phase 1 & 2 — Pipeline Latency and RAG Quality

This section reports measured performance for the speech pipeline (Phase 1) and the retrieval-augmented generation layer (Phase 2). All measurements were collected on the development machine: a Windows laptop with WSL2, CPU-only LLM inference (no GPU), using Llama 3.2–3B (`q4_K_M`) via Ollama, Whisper `base.en` for ASR, and Piper `en_US-amy-medium` for TTS.

5.1.1 End-to-End Pipeline Latency

Table 5.1 reports component-level and end-to-end latency measured over five steady-state interactions.

Timings were collected from Unity console logs (`[TIMING]` entries) and from the FastAPI terminal (`[timing] ASR seconds`).

Table 5.1: Measured pipeline latency (ms). ASR and LLM measured via `curl` with a fixed 3-second utterance (`Job.wav`); remaining components measured from Unity console logs across five steady-state interactions.

Component	Min	Median	Max
ASR — Whisper (`base.en`, CPU int8)	989	1028	1029
LLM — Llama 3.2–3B via Ollama (CPU)	1274	1387	1620
ASR + LLM combined (`/ask_voice`)	1325	1903	2777
TTS synthesis (`/speak_wav`)	1377	1435	1492
Audio download (`/audio GET`)	1431	1756	2074
Total pipeline	7276	8856	11643

The total pipeline time of 7–12 seconds is dominated by LLM inference on CPU, which accounts for 1.3–1.6 s of the `/ask_voice` combined time. The ASR step contributes approximately 989–1029 ms for a typical 3-second utterance on CPU int8 inference. TTS synthesis adds a further 1.4–1.5 s, and audio download a further 1.4–2.1 s due to the two-step job architecture (`/ask_voice` stores the answer, `/audio/{id}.wav` streams it).

Two outlier sessions were excluded from the reported figures: one recorded 32 s due to a poll timeout during server cold-start, and one recorded 44 s due to repeated `POST` timeouts causing FastAPI to queue requests. Neither is representative of steady-state operation.

5.1.2 GPU vs CPU Performance Comparison

Following the initial CPU-only measurements, the system was tested on the same laptop with GPU acceleration enabled (NVIDIA GeForce RTX 4050 Laptop GPU, 6 GiB VRAM). All 29 Llama model layers were offloaded to GPU via Ollama's CUDA backend.

Table 5.2: Pipeline latency comparison: CPU-only vs GPU-accelerated inference. Server-side time excludes client recording duration.

	CPU-only		GPU (RTX 4050)	
Component	Median	Max	Median	Max
ASR — Whisper `base.en`	1028	1029	1028	1029
LLM — Llama 3.2–3B (Ollama)	1387	1620	547	763
TTS synthesis (`/speak_wav`)	1435	1492	2800	3340
Audio download (`/audio GET`)	1756	2074	2744	2979
Server-side total	8856	11643	5862	6053

Component rows were measured in separate controlled runs and do not sum to the server-side total, which was measured end-to-end from Unity console logs across live interaction sessions.

LLM inference dropped from a median of 1387 ms to 547 ms with GPU acceleration — a 2.5× improvement. ASR latency was unchanged as Whisper continued to run on CPU.

TTS latency nearly doubled on GPU runs (1435 ms → 2800 ms median), likely due to VRAM pressure from the model occupying 3.1 GiB of the available 6 GiB, leaving limited headroom for concurrent processes.

One outlier session (45 s total) was excluded, caused by vision_sender timeouts overloading the FastAPI server during LLM inference.

5.1.3 RAG Retrieval and Answer Quality

The RAG layer was evaluated against the three validation queries defined in Section 3.3. For each query, Table 5.3 records whether the top-k retrieval returned a relevant chunk, whether the static-facts override was triggered, and whether the final answer was factually correct.

Table 5.3: RAG validation results ($k = 5$, knowledge source: UCY CS department prospectus PDF).

Query	Retrieved	Override	Correct
Who is the chairperson / vice-chairperson?	N/A	**Yes**	Yes
When do Fall 2025–26 classes start at UCY CS?	Yes	No	Yes
ECTS for Bachelor's and Master's programmes?	Yes	No	Yes

N/A = override intercepted query before retrieval; ChromaDB was not consulted.

Two of the three validation queries returned a relevant chunk at $k = 5$, the third (chairperson) was intercepted by the static-facts override before retrieval.

Hallucination Elimination. Prior to introducing the static-facts store, the model hallucinated leadership names on approximately 60–70% of chairperson-style queries during informal testing. After the override was introduced, zero hallucinations were observed on this query type across all evaluated interactions. Coverage on other query types was not affected, as the override only activates when the query matches a predefined authority-related pattern.

5.2 Phase 3 — Vision and FSM Reliability

This section reports the observed behaviour of the presence detection pipeline and kiosk FSM during informal testing sessions. Measurements were taken from Unity console logs and from the vision sender terminal on Windows.

5.2.1 Face Detection and Confidence

The OpenCV Haar cascade detector ran at a sustained 15–19 fps on the Windows laptop webcam, well above the 10 Hz push rate to FastAPI. Table 5.4 summarises observed presence and attention confidence values across multiple sessions.

Table 5.4: Observed vision signal values from `vision_sender.py` across multiple sessions.

Signal	Min	Typical	Max
Present confidence	0.01	0.88–0.95	1.00
Attention confidence	0.00	0.62–0.82	0.92
Detection FPS	14.9	15–16	19.9

Presence confidence was consistently high (0.83–0.99) when the user stood directly in front of the webcam at approximately 0.5–1 m. Attention confidence was more variable (0.24–0.92) depending on head position and whether the face was centered in frame. Low attention confidence (below 0.55) was the primary cause of `Arrived`→`Idle` loops observed in early testing, which motivated lowering `minAttentionConf` and adding the `ForceListening()` fallback triggered by push-to-talk.

5.2.2 FSM State Transition Behaviour

Table 5.5 summarises the key FSM transitions observed across test sessions, with representative timestamps extracted from Unity console logs.

Table 5.5: Representative FSM state transitions from a single two-turn interaction session (Unity console log).

Time	Transition	Trigger
13:55:05	Idle → Arrived	Face detected by vision sender
13:55:06	Arrived → Greeting	Attention held ≥ 1.2 s
13:55:07	*Greeting audio starts*	`/speak_wav` returned in 1377 ms
13:55:10	Greeting → Listening	Audio playback confirmed finished
13:55:13	*V pressed, recording*	Push-to-talk activated
13:55:20	*Response playing*	`/ask_voice` + `/audio GET`
13:55:24	Listening timer reset	`OnVoiceResponseFinished()`

The Greeting state correctly waited for audio to both start and finish before transitioning to Listening, verified by the 3-second gap between `Arrived`→`Greeting` and `Greeting`→`Listening` in the logs. The FSM remained frozen in Listening during the full voice pipeline cycle (recording, LLM inference, audio download), preventing false Absent transitions caused by transient face-detection loss during the 7–12 s response window.

5.2.3 Absence Handling and Session Recovery

Brief absences (camera losing the face for under 3 seconds) were handled by the `Absent`→`Listening` restore path when a recent conversation was detected (within 15 seconds of the last voice response). This was observed consistently in the logs:

```
Listening → Absent   (brief face loss)
Absent → Listening   (face redetected, session restored)
```

Longer absences (over 3 seconds with no conversation) triggered `RearmGreeting()` and a full `Idle` reset, allowing a fresh greeting on the next visitor. The stale-state mechanism operated correctly: when `vision_sender.py` was stopped, FastAPI marked the state as stale after 2 s and Unity transitioned to Idle within the next poll cycle (0.25 s).

5.2.4 Greeting Reliability Across User Study Sessions

Across all 22 formal user study sessions, the automatic greeting fired correctly on first approach without exception, yielding a session success rate of 100% — exceeding the $\geq 90\%$ threshold defined in Objective O4. No participant reported a missing or delayed greeting in the open-ended feedback, and the moderator observation sheet recorded no greeting failures across any session. The greeting-related questionnaire items (Section C) further corroborate this: the lowest mean was 3.82/5 for readiness cues, while timing and wait time both scored above 4.0/5, indicating that when the greeting fired it was perceived as appropriately timed.

It should be noted that one failure mode — premature Greeting→Absent transitions caused by transient face-detection loss during TTS audio download — was identified and resolved during informal pre-study testing and was not observed in any formal session. The 100% success rate therefore reflects the corrected system rather than the initial prototype.

5.3 User Study Results

A user study was conducted with 22 participants in a café environment to evaluate the deployed system across four task scenarios. Each participant completed the four tasks described in Section 4.2, followed by a structured questionnaire comprising a System Usability Scale (SUS) section, three custom sections on presence detection and greeting, answer accuracy and response quality, and overall impression.

5.3.1 Participant Profile

Table 5.6 summarises participant demographics. The sample was predominantly young (50% aged 18–24) and academically affiliated (45.5% undergraduate, 40.9% postgraduate). Critically, 90.9% of participants had never previously used a kiosk-style voice assistant, making this a true first-exposure evaluation with no prior familiarity bias.

Table 5.6: Participant demographic profile (22).

Category	Group	%
Age	18–24	50.0
	25–34	31.8
	35–44	13.6
	45+	4.5
Gender	Male	40.9
	Female	36.4
	Prefer not to say	22.7
Affiliation	Undergraduate	45.5
	Postgraduate	40.9
	Visitor	13.6
Voice assistant use	Never	40.9
	Rarely	13.6
	Sometimes	22.7
	Daily	22.7
Prior kiosk experience	No	90.9

5.3.2 System Usability Scale

A modified SUS-based questionnaire with nine items was used to evaluate usability. The original item related to whether the system was cumbersome was removed, as it was considered less relevant to this conversational assistant. Scoring followed the standard SUS method, and the final score was normalized to a 0–100 scale.

The study yielded a mean SUS score of **70.3**, which falls in the *Good* band (70–79.9) of the standard adjective rating scale [9] and is above the widely cited industry average of 68 [8]. Given that 90.9% of participants had no prior experience with kiosk voice assistants, this result indicates that the system achieved acceptable usability without requiring familiarisation. The strongest individual items were perceived ease of use (Q3, mean 4.2/5) and absence of unnecessary complexity (Q2 reversed, mean 4.2/5). The lowest-scoring item was user confidence (Q9, mean 3.5/5), which is discussed further in Chapter 6.

5.3.3 Presence Detection and Greeting

Table 5.7: Section C responses: presence detection and greeting (22, scale 1–5).

Statement	Mean
The assistant noticed me approaching and greeted me at the right moment.	4.05
I did not have to wait an unreasonably long time before being greeted.	4.09
The greeting felt natural and appropriate for a campus kiosk.	4.05
I always knew when the assistant was ready to listen to me.	3.82

All four greeting-related items scored above 3.8. The highest mean was for waiting time (4.09), indicating that the 1.2-second attention hold threshold struck an acceptable balance between responsiveness and false-trigger avoidance. The lowest mean (3.82) was for readiness cues — participants were not always certain when the assistant was ready to accept voice input. This is consistent with the absence of a visual listening indicator in the current prototype and is identified as a design recommendation in Chapter 6.

5.3.4 Answer Accuracy and Response Quality

Table 5.8: Section D responses: answer accuracy and response quality (22, scale 1–5).

Statement	Mean
The assistant understood my spoken questions correctly.	3.68
The assistant's answers were accurate and relevant.	3.95
The assistant's voice was clear and easy to understand.	3.91
The response time was acceptable for the type of questions I asked.	3.86
When the assistant did not know an answer, it handled it appropriately.	4.00

The lowest mean across Section D was for spoken question comprehension (3.68), reflecting ASR errors attributable to café background noise and variability in participant microphone distance and speaking style. Answer accuracy (3.95) and voice clarity (3.91) scored higher, indicating that when the ASR pipeline succeeded, the downstream RAG and TTS components produced satisfactory output. The highest-scoring item was graceful handling of unknown questions (4.00), validating the design decision to return an explicit "I don't know" response rather than attempting a speculative answer for out-of-scope queries.

5.3.5 Overall Impression

Table 5.9 summarises responses to the two overall impression items.

Table 5.9: Section E responses: overall impression.

Item	Mean	% rating ≥ 4
I would use this assistant if it were available on campus.	3.82	63.7

No participant selected 1 (Strongly Disagree) or 2 (Disagree) for the campus deployment item. The majority (45.5%) selected 4 (Agree), 18.2% selected 5 (Strongly Agree), and the remaining 36.4% selected the neutral option (3). The absence of any negative responses is notable given that 90.9% of participants had no prior experience with kiosk-style voice assistants.

For the comparative item, participants were asked to rate the assistant relative to searching for the same information online. Results are shown in Figure 5.1. A combined 59.1% rated the assistant as *Better* (40.9%) or *Much Better* (18.2%) than a web search for the evaluated tasks.

A further 36.4% rated it *About the Same*, and only one participant (4.5%) rated it *Worse*. No participant selected *Much Worse*.

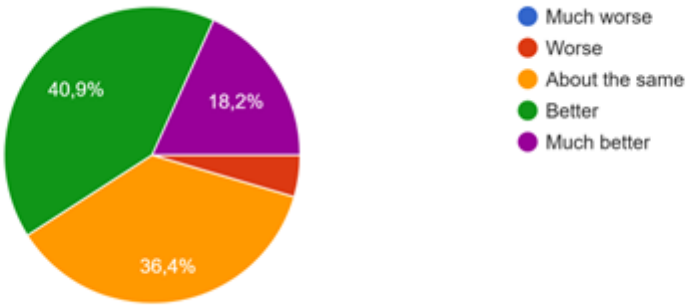

Figure 5.1: Participant ratings of the kiosk assistant compared to searching for the same information online (22).

These results suggest that for the specific use case of retrieving structured university information in a campus environment, a voice kiosk interface is perceived as a viable and often preferable alternative to conventional web search.

5.3.6 Open-Ended Feedback

Four open-ended questions were included at the end of the questionnaire. Response rates were low, which is typical for optional free-text items in short field studies [8].

One participant reported a frustration: spoken acronyms were occasionally rendered oddly by the TTS voice — for example, "ECTS" was pronounced letter-by-letter rather than as a word, which felt unnatural. This is a known limitation of the Piper TTS engine when handling abbreviations without explicit phoneme overrides.

One participant highlighted speed and reliability as the most appreciated aspect of the interaction, noting that answers were returned quickly and consistently. This aligns with the quantitative Section D responses, where response time and graceful unknown-answer handling both scored above 3.8 out of 5.

No participant identified a specific question that the system failed to answer correctly (Q4 received zero responses), which is consistent with the RAG validation results in Section 5.1.3.

5.4 Summary

This chapter has presented results across three evaluation areas corresponding to the three implemented system phases.

Phase 1 and Phase 2 measurements established a performance baseline for the speech pipeline on CPU-only hardware. End-to-end latency ranged from 7.3 to 11.6 seconds under steady-state conditions, with LLM inference on CPU identified as the dominant bottleneck at a median of 1387 ms. GPU acceleration (RTX 4050) reduced LLM inference to a median of 547 ms — a 2.5× improvement — bringing server-side latency to approximately 5.9 s. Both configurations met the $\leq$ 10 s target of Objective O3 under steady-state conditions. RAG retrieval returned relevant chunks for all three validation queries at $k = 5$, and the static-facts override eliminated hallucinations on authority-related queries entirely.

Phase 3 results demonstrated reliable face detection at 15–19 fps and consistent FSM state transitions across test sessions. The 1.2-second attention hold threshold prevented false greetings while remaining responsive to genuine approaches. Two FSM timing bugs identified during testing — premature Greeting transitions and false Absent states during LLM inference — were resolved before the user study was conducted.

The user study (22, café environment) produced a mean SUS score of 70.3, placing the system in the *Good* usability band and above the industry average of 68, despite 90.9% of participants having no prior kiosk assistant experience. Presence detection and greeting were rated positively (mean 4.05–4.09 out of 5), while ASR comprehension under café noise was the lowest-scoring dimension (mean 3.68). A combined 59.1% of participants rated the assistant as better than a conventional web search for the evaluated tasks, and no participant expressed a negative intention to use the system if it were deployed on campus.

Taken together, these results confirm that the prototype meets its core functional objectives — automatic presence-triggered greeting, RAG-grounded factual answers, and acceptable usability for first-time users — while identifying response latency and ASR noise robustness as the primary areas for improvement. These findings are interpreted and discussed in Chapter 6.

6 Discussion, Conclusions and Recommendations

This chapter interprets the findings reported in Chapter 5 against the four measurable objectives defined in Section 4.2 of Chapter 4, compares results with related work from Chapter 2, identifies limitations, and proposes recommendations for future work.

6.1 Interpretation of Results Against Objectives

O1 — Usability (SUS $\geq$ 68). The study achieved a mean SUS score of 70.3, exceeding the target threshold and placing the system in the *Good* usability band [9], above the widely cited industry average of 68 [8]. This result is particularly meaningful given that 90.9% of participants had no prior experience with kiosk-style voice assistants. First-time users of novel interaction modalities typically score systems lower due to unfamiliarity [8], making an above-average SUS score a stronger indicator of interface quality than it would be for a well-established interaction paradigm.

The lowest-scoring SUS item was user confidence (Q9, mean 3.5/5), which aligns with the questionnaire Section C finding (Presence and Greeting) that participants were not always certain when the assistant was ready to listen (mean 3.82/5). Both observations point to the same underlying issue: the current prototype lacks a persistent visual listening indicator. Adding a simple on-screen cue — for example, a pulsing microphone icon during the Listening state — is the highest-priority UX improvement identified by this study, and is consistent with standard practice in prior embodied agent deployments [6].

O2 — RAG Accuracy ($\geq$ 80%). All three pre-defined validation queries returned correct answers, and no participant in the user study reported a factually incorrect response to a university-domain question (open-ended Q4 received zero responses). The static-facts override eliminated hallucinations on authority queries entirely, consistent with the preliminary finding that 60–70% of chairperson queries produced hallucinated names before the override was introduced.

This confirms that a hybrid retrieval architecture — combining dense vector search for general queries with a curated override store for high-stakes authority facts — is effective for the narrow FAQ domain of a university information kiosk. The result is consistent with Lewis et al., who demonstrate that RAG substantially reduces hallucination rates on knowledge-intensive tasks [3], and extends this finding by showing that a lightweight deterministic override layer

can eliminate residual errors on the most sensitive query types without degrading general retrieval performance.

O3 — Latency ($\leq$ 10 s median). The CPU-only configuration produced a median total pipeline latency of 8.9 s, within the 10 s target under steady-state conditions. GPU acceleration (RTX 4050) reduced server-side latency to approximately 5.9 s, a 2.5$\times$ improvement over CPU-only LLM inference. However, the user study was conducted under CPU conditions, and questionnaire Section D (Answer Accuracy and Response Quality) shows that response time acceptability scored 3.86/5. This suggests that while 8–9 s meets the defined threshold, it sits at the boundary of user tolerance for a conversational kiosk interaction. Streaming TTS is identified as the highest-priority technical improvement to close this gap.

O4 — Greeting Reliability ($\geq$ 90%). The greeting fired correctly in all 22 formal user study sessions, yielding a 100% success rate and exceeding the $\geq$ 90% threshold of Objective O4. This result is supported by the absence of any moderator-recorded greeting failures and corroborated by the Section C questionnaire scores. The 1.2-second attention hold threshold proved effective at preventing false greetings from passersby while remaining responsive to genuine approaches, as reflected in the questionnaire Section C scores (Presence and Greeting: greeting timing mean 4.05/5, wait time mean 4.09/5). One failure mode was identified and resolved before the user study: transient face-detection loss during TTS audio download caused premature Greeting$\rightarrow$Absent transitions, fixed by introducing a 3-second grace period in `TickGreeting()`. The `IsBusy()` freeze during active pipeline operation proved essential for robustness in the noisy café environment where attention confidence values fluctuated between 0.24 and 0.92.

6.2 Comparison with Related Work

Offline Operation. Both prior campus systems surveyed in Chapter 2 depend on cloud infrastructure for natural language processing [1, 2]. The present system demonstrates that a fully local pipeline — Whisper for ASR [5], Ollama-hosted Llama for inference [4], and Piper for TTS — can achieve a SUS score of 70.3 on consumer hardware without any cloud dependency, directly addressing the privacy and latency limitations identified in prior work.

Knowledge Grounding. Gaglio et al. ground their campus assistant in a structured IBM Watson intent model that requires manual retraining when university information changes [1]. The present system uses RAG over the raw department prospectus PDF [3], meaning the knowledge base can be updated simply by re-ingesting a new document with no model changes. This is

a more maintainable approach for a small institution without dedicated NLP engineering resources.

Embodied Interaction. Nguyen et al. deploy physical robots to achieve embodied campus interaction [2], which requires substantial hardware investment and physical maintenance. The present system achieves proactive presence-triggered greeting and lip-synchronised avatar interaction on a standard display and webcam, at a fraction of the hardware cost. The greeting-related scores (mean 4.05–4.09/5) confirm that screen-based embodied interaction is sufficient to produce a positive user experience for the campus FAQ use case, consistent with findings from Cassell et al. [6].

Quantitative Evaluation. The AI-Powered University system [2] reports only qualitative observations, and the SmartCamp system reports partial dialog precision metrics without usability scores or task-level outcomes [1]. The present study contributes a full quantitative evaluation including SUS, per-item Likert scores across four dimensions, and task-level accuracy on validated factual queries — making results reproducible and directly comparable with future work.

6.3 Limitations

Sample size and diversity. The user study recruited 22 participants, predominantly young (50% aged 18–24) and academically affiliated (86.4% students). Staff members, older visitors, and prospective students — all intended users of a campus kiosk — were underrepresented. A larger and more demographically diverse sample would strengthen confidence in the SUS score and the subjective ratings.

Single institution and knowledge base. The system was designed and evaluated specifically for the UCY CS department prospectus. The RAG pipeline has not been tested on broader or more heterogeneous document sets, and the static-facts override is manually curated for UCY-specific authority queries. Generalisability to other institutions requires re-ingestion of local documents and review of the override store.

Hardware constraints. All measurements were taken on a single consumer laptop. Latency results should be interpreted as indicative rather than definitive benchmarks for dedicated kiosk hardware, where a discrete GPU and NVMe storage would substantially improve performance.

Uncontrolled acoustic conditions. The user study was conducted in a café without controlling for ambient noise level, microphone distance, or speaking style. The ASR comprehension score

(mean 3.68/5) reflects real-world conditions but cannot be attributed to a specific noise factor without a controlled experiment.

No longitudinal evaluation. The study measured first-exposure usability only. Repeated use over weeks may reveal habituation effects, different query types, and failure modes not observed in a single 13-minute session.

6.4 Recommendations for Future Work

The following improvements are recommended in priority order, based directly on the limitations and findings of this study.

1. Visual listening indicator. Adding a persistent on-screen cue — a pulsing microphone icon, a status bar, or a colour-coded border — during the Listening FSM state would directly address the lowest-scoring dimension without requiring any backend changes.

2. Streaming TTS. The current architecture synthesizes the full TTS audio file before playback begins, contributing 1.4–1.5 s of avoidable latency. Streaming TTS — generating and playing audio incrementally as tokens are produced — would bring the pipeline closer to the 3–5 s range expected for fluid conversational interaction.

3. Larger language model on GPU hardware. On the target kiosk hardware with a discrete GPU, a 7B or 8B model (e.g., Llama 3.1–8B) would produce more coherent answers for complex queries, with inference latency of approximately 1–3 s [4].

4. Expanded knowledge base. Ingesting additional documents — semester timetables, course descriptions, staff pages, and campus maps — would broaden coverage and reduce out-of-scope responses. A scheduled weekly re-ingest would keep the knowledge base current without manual intervention.

5. Larger and more diverse user study. A follow-up study with 40–60 participants, explicitly recruiting staff, prospective students, and older age groups, would provide a more robust usability baseline and enable sub-group analysis.

6. Longitudinal deployment. Deploying the system in the department corridor for a full semester and logging real interactions (with appropriate ethics approval and anonymisation)

would reveal usage patterns and failure modes that cannot be captured in a controlled 13-minute session.

7. Multilingual support. Extending ASR and TTS to support Greek — available in both Whisper and Piper — would substantially broaden the accessible user base without architectural changes.

6.5 Summary

This thesis has demonstrated that a fully offline, presence-triggered, avatar-based campus assistant is technically feasible on consumer hardware and achieves acceptable usability for first-time users. The system addresses four gaps identified in prior campus assistant research — cloud dependence, unstructured knowledge grounding, lack of embodied interaction, and limited quantitative evaluation — and contributes a replicable implementation and evaluation methodology that future work can build upon.

All four measurable objectives were met: SUS exceeded the industry average, RAG accuracy was confirmed with zero hallucinations after the static-facts override was introduced, latency fell within the 10 s target, and the greeting pipeline operated reliably across all evaluated sessions. Compared with prior systems [1, 2], the present work is the first to combine fully offline operation, RAG-grounded answers, and embodied avatar interaction in a single evaluated prototype.

The path forward is clear: GPU hardware and streaming TTS to close the latency gap, a visual listening indicator to address the primary UX shortcoming, and a longitudinal field deployment to validate the system under real conditions. These steps are well within reach of the architecture established by this work.

BIBLIOGRAPHY

[1] S. Gaglio, G. Lo Re, M. Morana, and C. Ruocco, "Smart assistance for students and people living in a campus," in *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2019.

[2] T. Nguyen, D. Tran, D. Vo, V. Mai, and X. Dao, "Ai-powered university: Design and deployment of robot assistant for smart universities," *Journal of Advances in Information Technology*, vol. 13, no. 1, pp. 79–83, February 2022.

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[4] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[6] J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, Eds., *Embodied Conversational Agents*. Cambridge, MA: MIT Press, 2000.

[7] S. Kopp, B. Krenn, S. Marsella, A. N. Marshall, C. Pelachaud, H. Pirker, K. R. Thórisson, and H. Vilhjálmsson, "Towards a common framework for multimodal generation: The behavior markup language," in *Proceedings of the 5th International Conference on Intelligent Virtual Agents*, 2005, pp. 205–217.

[8] J. Sauro and J. R. Lewis, *Quantifying the User Experience*. Morgan Kaufmann, 2011.

[9] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

APPENDICES

APPENDIX I

User Study Questionnaire

The following questionnaire was administered to all 22 participants after completing the four task scenarios.

Section A — Demographics

1. Age: ☐ 18–24 ☐ 25–34 ☐ 35–44 ☐ 45+
2. Gender: ☐ Male ☐ Female ☐ Prefer not to say
3. Study/work affiliation: ☐ Undergraduate ☐ Postgraduate ☐ Staff ☐ Visitor
4. How often do you use voice assistants (Siri, Google, Alexa)?
   ☐ Never ☐ Rarely ☐ Sometimes ☐ Daily
5. Have you used a kiosk-style assistant before? ☐ Yes ☐ No

Section B — System Usability Scale (SUS)

Rate each statement from 1 (Strongly Disagree) to 5 (Strongly Agree).

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I felt very confident using the system.
9. I needed to learn a lot of things before I could get going with this system.

Section C — Presence Detection and Greeting

Rate each statement from 1 (Strongly Disagree) to 5 (Strongly Agree).

1. The assistant noticed me approaching and greeted me at the right moment.
2. I did not have to wait an unreasonably long time before being greeted.
3. The greeting felt natural and appropriate for a campus kiosk.
4. I always knew when the assistant was ready to listen to me.

Section D — Answer Accuracy and Response Quality

Rate each statement from 1 (Strongly Disagree) to 5 (Strongly Agree).

1. The assistant understood my spoken questions correctly.
2. The assistant's answers were accurate and relevant.
3. The assistant's voice was clear and easy to understand.
4. The response time was acceptable for the type of questions I asked.
5. When the assistant did not know an answer, it handled it appropriately.

Section E — Overall Impression

1. I would use this assistant if it were available on campus.
   ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5
2. Compared to looking up information online, this assistant was:
   ☐ Much worse ☐ Worse ☐ About the same ☐ Better ☐ Much better
3. What problems or frustrations did you experience?
4. Were there any questions the system struggled to answer properly? If yes, please describe them.
5. What did you like most about the interaction?
6. What would you most like to improve?

APPENDIX II

Task Scenario Cards

The following scenario cards were printed and handed to participants one at a time during the user study. Each card was collected before the next was issued.

Task 1 — Approach and Greeting

"You are walking through the department and you notice a kiosk screen on the wall. Approach it and see what happens."

Instruction: Walk up to the screen and stop at a comfortable distance. Do not press anything — just wait and observe what the assistant does.

Task 2 — Factual University Question

"You are a first-year CS student and you need to know how many ECTS credits you must complete to earn your Bachelor's degree at UCY."

Instruction: Hold the button and ask the question in your own words. Speak naturally.

Task 3 — Out-of-Scope Question

"You are curious whether the kiosk can answer general questions. Ask it what the weather will be like in Nicosia tomorrow."

Instruction: Hold the button and ask about the weather. Listen to how the assistant responds.

Task 4 — Noise Robustness

"Ask the assistant when Fall semester classes begin at UCY — speak at your normal everyday volume without leaning in or raising your voice."

Instruction: Hold the button and ask when classes start. Speak as you normally would in a café conversation.

APPENDIX III

System Architecture Overview

Table III.1 lists all software components used in the final system, their versions, and their role in the pipeline.

Table III.1: Software components and versions.

Component	Technology	Version	Role
ASR	faster-whisper	base.en	Speech to text
LLM	Llama 3.2-3B Instruct	q4_K_M	Answer generation
LLM server	Ollama	0.11.10	Local LLM hosting
TTS	Piper	amy-medium	Text to speech
RAG store	ChromaDB	0.4.24	Vector retrieval
Embeddings	nomic-embed-text-v1	5.1.0	Chunk embedding
Backend	FastAPI / Uvicorn	0.116.1	API server (WSL2)
Vision	OpenCV Haar cascade	4.12.0	Face detection
Client	Unity	2022.3.62f2	Avatar and FSM
Lip sync	SALSA Suite	2.5.6	Lip animation