DTS311TC FINAL YEAR PROJECT

Player-Aware Intelligent Monitoring and Operations Navigator

Proposal Report

In Partial Fulfillment
of the Requirements for the Degree of
Bachelor of Engineering

Student Name	:	Taimingwang Liu
Student ID	:	2037690
Supervisor	:	Xihan Bian

School of AI and Advanced Computing
Xi'an Jiaotong-Liverpool University

November 2025

Abstract

Apply the font of Times New Roman to the paragraphs of the abstract using font size of 12. An abstract is usually one to three paragraphs long with a length of 150 to 350 words.

Contents

1 Introduction

The demand for **interactive and companion experiences** in real-time entertainment is on the rise. Players are no longer satisfied with simple automation or static overlays; they seek dynamic, engaging partners that offer meaningful interaction. As gaming experiences evolve, the focus is shifting towards assistants that not only provide practical help but also enrich the player's journey through companionship and engagement.

Games like AI2U: "With You 'Til The End" are already capitalizing on the demand for interactive experiences, offering players the novelty of engaging with generative AI characters [1]. At the same time, major companies such as NVIDIA, with its Avatar Cloud Engine (ACE), and Ubisoft, with its "NEO NPCs", are advancing foundational technologies to create autonomous agents that enhance gameplay by offering more than just conversation [2], [3]. These developments demonstrate the growing commercial viability of AI-driven experiences, supported by increased player engagement and media attention [4].

The rise of AI-driven virtual streamers, especially the Neuro-sama phenomenon, highlights a significant shift in both technology and community-driven commercialization. Neuro-sama, an AI-powered VTuber, engages in real-time conversations and dynamic gameplay, capturing the attention of a wide audience [5]–[7]. Although Neuro-sama remains closed-source, its success has sparked a vibrant open-source ecosystem, with developers working to replicate or expand upon its capabilities [8]–[10]. This technological shift has been met with strong commercial traction on platforms like Twitch, underscoring the growing market demand for AI that combines both utility and companionship [11].

The success of these AI-driven personalities validates the market's desire for this combination of utility and companionship; this project builds upon this principle, seeking to translate this proven engagement from a broadcast-entertainment model into a player-centric, companion-style assistant. This assistant is designed to augment, not automate, the player's agency. Its role is distinct from broadcast-focused AI-Vtubers and player-replacing automation bots; it is scoped as a persistent, in-game partner that uses a **full voice-loop** to provide support that is both relational and functional. The system's core capabilities are threefold: offering **proactive, contextual companionship** by using in-character awareness to identify opportunities (e.g., spotting missed items); engaging in **collaborative problem-solving** to understand player goals and access external knowledge (e.g., researching online strategies); and supporting **on-demand task delegation**, allowing the player to hand over control for specific, well-defined tasks (e.g., 'explore this area while I'm away').

The technical scope to achieve these capabilities involves three key techniques. First, the system will use **a Unified Action Interface via GUI**, a human-homomorphic interface with a screen-in, keyboard/mouse-out paradigm. This approach eliminates the need for game-specific APIs and ensures cross-platform adaptability [12], [13]. Second, actions will be managed through **Constrained Action Generation with Structured Output**. This method, which for-

mats commands from a predefined set of valid actions (e.g., 'move forward') into structured output (e.g., JSON), reduces errors like hallucinations and ensures that delegated tasks are legal and reproducible [14]. Third, the system will employ **a Low-Coupling Orchestration** (i.e., modularization by MCP-style). This plug-and-play approach is crucial, ensuring both scalability for integrating new skills like problem-solving and task-delegation, and testability which enables systematic ablation studies to evaluate each component's contribution [15].

With a clear understanding of market demand and technological feasibility, this project aims to develop a companion-style assistant that seamlessly integrates into gameplay. By demonstrating a system capable of this deeper, functional partnership—blending both companionship and shared agency—this work will enhance the overall player experience through dynamic, interactive assistance.

2 Literature Review

This section reviews the foundational research on AI agents for complex, interactive tasks, which underpins the development of the proposed game companion. It first establishes the technical feasibility of using human-homomorphic interfaces for game-agnostic control. Following this, it examines the evaluation protocols and performance metrics critical for benchmarking such systems. The review then explores the core **agentic chain**—a synthesis from recent literature comprising **planning, skills, reflection, and memory**—necessary for maintaining stability in long-horizon tasks. Subsequently, it details the modern learning paradigms required for robust agent training, before concluding with a discussion of persistent challenges, such as latency and safety, and a synthesis of the key literature gaps this project aims to address.

2.1 Feasibility of Human-Homomorphic Interfaces

Achieving the proposed unified action interface requires grounding the agent in **human-homomorphic interfaces**, a paradigm often referred to as General Computer Control (GCC). The feasibility of this approach is strongly demonstrated by Tan et al. in the **CRADLE** framework, which introduces the "Screen as input, keyboard/mouse as output" paradigm. This proves that a unified agent can perform complex, long-horizon tasks across different software, including games, without relying on specialized APIs [12]. This pipeline, proceeding from **screenshot to structured action**, is further validated by Gu et al. in the **UI-Venus** framework, who use Reinforcement Fine-Tuning (RFT) to successfully train agents on GUI tasks [13]. This is also supported by Wang et al. (2025) in the **Ponder & Press** framework, who argue that splitting GUI tasks into "Ponder" (planning) and "Press" (grounding) improves reliability [16].

A key component of this paradigm is advanced planning. The **Agent S** framework, introduced by Agashe et al. (2024), provides another strong validation of the GCC paradigm for complex tasks. Its key contribution is "Experience-Augmented Hierarchical Planning," where the agent learns from both external knowledge and its own internal "narrative" and "episodic" memory to decompose and solve long-horizon goals [17].

While feasible, this approach requires mechanisms to ensure stability. The **ORAK** benchmark, introduced by Park et al. (2025), highlights the importance of using a restricted valid-action set, or **Legal Move Constraints**. This feature, which greatly constrains the model's output space, is a critical optimization that can reduce action errors by 15–20% [15]. Representation of both state and action is also critical. The ORAK benchmark emphasizes that standardized **State Representation Formats** (e.g., agent position, task progress) are essential for providing rich context [15]. For **Move Representation Formats**, the **CodeAct** framework from Liang et al. (2024) proposes a powerful solution by using **executable Python code** as the agent's output. This is more robust than static JSON, as it can handle complex logic and allows the agent to **self-debug** by receiving and correcting its own error tracebacks [18].

On the input side, the ORAK benchmark is explicitly designed to evaluate agents using three different **input modalities**: Text-only, Image-only, and Text + Image [15]. This is a critical area of research, as the **V-MAGE** benchmark from Zheng et al. (2025) demonstrates that **vision-only inputs** pose significant generalization challenges [19], though other work supports a vision-only (no DOM/HTML) approach for improving generalizability [16].

2.2 Evaluation Protocols & Performance Metrics

To rigorously measure the performance of these agents, a new generation of evaluation frameworks has been established. Proper benchmarking is essential, with systems like ORAK and **LMGame-Bench** providing comprehensive frameworks for ensuring reliable results across diverse game types [15], [20]. A primary outcome from these evaluations is the poor performance of current models on out-of-distribution (OOD) tasks. The study by Guruprasad et al. (2025) notes this as poor OOD contextual generalization [14], a finding confirmed by other benchmarks and surveys [19], [21], [22].

Within these evaluations, researchers rely on a suite of metrics. General metrics such as `pass@k` are commonly used to measure success over repeated trials. More specific to GUI agents, metrics like `Invalid%` track the proportion of invalid actions, a critical diagnostic for models struggling to produce valid outputs [14], [22]. To properly evaluate agents on long-horizon tasks, benchmarks like ORAK also employ a mix of reward types, including **Dense Rewards** (for continuous feedback) and **Auxiliary Rewards** (e.g., for "correct format output") to guide and assess complex behaviors [15].

Beyond simple success rates, recent work has introduced more sophisticated systems for comparative and granular assessment. To address the difficulty of comparing models across different games, Zheng et al. introduce a **Dynamic ELO system** to standardize agent performance [19]. The same framework also moves beyond a single score by using "Unit Tests for Core Visual Abilities." This method provides a granular assessment of MLLM failures, identifying specific failure modes such as errors in **positioning**, **direction**, **identification** (e.g., seeing paths as obstacles), and a **conflict** between correct reasoning and incorrect execution [19]. Finally, recent papers argue for a **Human-Centered Evaluation Framework,** stating that metrics for a practical agent must also include **latency, safety, privacy, robustness, and error-recovery mechanisms** (like rollbacks) [23].

2.3 Agentic Modules and Stability in Long-Horizon Tasks

For an assistant to be a reliable, long-term partner, it must employ a set of **agentic modules** to manage complex tasks and mitigate error accumulation. Recent literature, including surveys by Hu et al. (2024) and Xu et al. (2024), has synthesized this into an "agentic chain" of planning, skills, reflection, and memory [24], [25]. The core architecture for this is a hierarchical multi-agent framework, as proposed by Zhang et al. (2025) in **AgentOrchestra**. This "conductor"

model features a central "planning agent" that decomposes complex objectives and delegates sub-tasks to "specialized agents" [26], [27]. For this "plug-and-play" architecture to function, a standardized "Model Context Protocol" (MCP) is necessary, which ORAK and other frameworks use to allow the central conductor to "seamlessly interact" with the specialized agentic modules.

The "conductor" agent's internal logic is an **agentic chain**, as demonstrated in the CRADLE framework, which includes modules for "Info Gathering," "Self-Reflection," "Skill Curation," and "Action Planning" [12]. **Memory** is a critical module in this chain. The Agent S framework provides a powerful model for this, using "Experience-Augmented Hierarchical Planning" that retrieves from long-term "narrative memory" (past tasks) and "episodic memory" (past steps) [17], [20], [28]. To make long-term planning tractable, agents also rely on a curated set of **atomic skills** (or "Macros") [29], which the CRADLE framework implements with its "Skill Curation" module [12].

To prevent error accumulation and **task drift**, a dedicated safety module based on the **BacktrackAgent** framework from Yuan et al. (2025) is required. This module uses a "Verifier" (for simple failures), a "Judger" (to identify semantic errors, e.g., "Game Over" screen), and a "Reflector" (to backtrack to a known good state and replan) [20], [28], [30]. Finally, a key new direction is **Self-Evolution**, which the UI-Venus framework implements with its "Self-Evolving Trajectory History Alignment." This method acts as automated reflection, allowing the agent to refine its own reasoning and "thought-action pairs" for more "coherent planning" [13], [24].

2.4 Learning Paradigms for Robust Agent Training

A stable agent architecture must be supported by a robust training paradigm. While many foundational models are trained via supervised imitation, modern agent systems are increasingly applying **Reinforcement Fine-Tuning (RFT)**. The **UI-AGILE** framework from Lian et al. (2025), for example, provides a SOTA (State-of-the-Art) training methodology, proposing novel reward-shaping techniques like a "Continuous Grounding Reward" (to incentivize precision) and a "Simple Thinking" Reward (to balance speed and reasoning) [13], [31], [32]. To overcome sparse rewards, systems also employ a combination of **dense rewards** (feedback at each timestep) and **auxiliary rewards** (feedback for sub-goals or correct formatting) [32], [33].

As an alternative to online RFT, **offline learning paradigms** like the **Decision Transformer** (DT) (Chen et al., 2021) frame RL as a sequence modeling problem, allowing agents to learn from static datasets of expert behavior, as explored by frameworks like `R2-Play` [34]. A key paradigm for solving latency is the **Hybrid Offline Generation** approach from the **PORTAL** framework (Wang et al., 2025). It uses the LLM *offline* to generate a game-playing policy as a **Behavior Tree (BT)**, which is then executed at **zero-latency** by a simple runtime interpreter [35].

Finally, reliability is enhanced through **structured action generation**. The CodeAct framework proposes a highly robust method: using **executable Python code** as the action output. This allows the agent to **self-debug** by receiving an error traceback from the interpreter and correcting its own code [18], [36]. This is often combined with **Grounded task execution**, which explicitly ties an agent's actions to the UI or game state to reduce ambiguity [29].

2.5 Challenges and Future Directions

Despite rapid progress, several persistent challenges must be addressed. The most immediate is **latency**, as studies in real-time human-AI coordination by Liu et al. (2023) have empirically demonstrated that latency beyond approximately 100 ms degrades fluid cooperation [37]. The PORTAL framework's "zero-latency" compiled Behavior Tree approach is a key proposed solution to this [23], [35].

Second, **safety and robustness** are critical [23], [28], [38]. The BacktrackAgent framework provides a concrete solution, with a 3-module architecture ("Verifier," "Judger," "Reflector") to detect, assess, and recover from semantic errors [30]. This is vital because **out-of-distribution (OOD) generalization** is a critical failure point for most agents [14], [19], [21]. The PORTAL framework offers a breakthrough solution here as well, demonstrating unprecedented cross-game generalization by compiling policies that were successfully applied to thousands of different 3D games [35].

Recent surveys also identify that current agents still struggle with **long-horizon tasks suffering from error accumulation** and **exhibiting low fine-grained precision** [22], [28], [39]. These challenges collectively define the **future directions** for the field: enhancing **multimodal capabilities**, improving **accuracy** in fine-grained control, and solving the "prohibitive inference latency" of current systems [12], [30], [35].

2.6 Synthesis and Gaps in the Literature

This review reveals that while significant progress has been made in functional agent control, a clear and actionable gap exists in the literature regarding unified, companion-style assistants. The primary literature gap is identified by the **Four-Quadrant Taxonomy** (Sun & Wu, 2025), which separates AI companions on an **(Emotional vs. Functional)** axis. Current research is bifurcated, leaving the blend of these quadrants as a key gap, with frameworks like `EmotionAWARE` highlighting the complexity of the "Emotional" component [40], [41].

This gap is evident in SOTA functional architectures like AgentOrchestra and UI-Venus, which are exclusively focused on task completion (the "Functional" axis) and do not include or address the relational component [13], [26]. The HCI study, **Cooperation Between Player and AI** (Lundström, John, 2024), suggests that a functional agent with higher autonomy is key to fostering a strong relational bond, as players tend to reject companions that increase their workload [42].

This project is therefore positioned to fill this gap. In doing so, it will also address two other persistent challenges: **real-time adaptability**, a known gap that benchmarks like LMGame-Bench are designed to test [20], and **cross-game transferability**. The difficulty of creating **game-agnostic agents** is a key hurdle [12], [15], and the PORTAL framework's success in this area highlights it as a critical SOTA research direction [35]. By focusing on a human-homomorphic interface and feedback-driven learning, this project will directly contribute to these unsolved areas, demonstrating a system capable of the deeper, functional partnership that current literature lacks.

3 Methodology

This project is timely. While the market demand is clear, the technological feasibility for a **reproducible, non-API-dependent** assistant has only recently emerged. Foundational research is now converging on the key methodologies required for productization. This includes the development of **unified evaluation protocols** and **modular ablation frameworks**, which are essential for ensuring that experiments are reproducible and comparable [15], [20]. Furthermore, recent work has demonstrated the viability of **GCC** pathways (i.e., screen-in, keyboard/mouse-out), confirming that capable agents can operate without relying on game-specific APIs [12], [13].

The proposed methodology is a **novel, hierarchical architecture** designed to fill the functional-relational gap identified in the literature. The system is architected as a central "Orchestrator" managing several specialized, "plug-and-play" sub-modules. This design is inspired by the modularity of SOTA (State-of-the-Art) frameworks like *AgentOrchestra* and will be integrated via a **Model Context Protocol (MCP)**, which provides a standardized bus for robust, low-coupling communication [15], [26].

3.1 Core Design Principles

The system's architecture is guided by four key principles:

1. **Unified Input (GCC):** The agent will operate on the "screen-in, keyboard/mouse-out" **GCC** paradigm. This human-homomorphic interface is foundational, ensuring the agent is game-agnostic and does not rely on specialized APIs, a concept validated by the *CRADLE* framework [12].

2. **Lightweight, Structured Perception:** To avoid the bottleneck of feeding raw video to a large VLM, a **Lightweight Perception Module** (e.g., a fine-tuned YOLO model) will first process the screen. This module will locate objects of interest (e.g., health bars, minimaps, enemies) and output a **structured text description** of the game state for the VLM to consume.

3. **Constrained Action Generation:** The agent's output will be **Constrained JSON** from a pre-defined **Skill Set**. The LLM's role is to select a valid skill, not to generate raw coordinates. This aligns with the reliability principles of "Legal Move Constraints" and structured outputs [15], [18], [36].

4. **Proactive Companionship:** The system is designed to be a proactive partner, not just a reactive tool. This is achieved by having the Relational Core monitor game-state events (e.g., "danger status" identified by the Orchestrator) and initiate interaction, fulfilling the "Relational-Functional Loop" identified in the literature.

3.2 Proposed System Architecture

The system's data flow is a continuous loop managed by a central Orchestrator, which is comprised of several system-level modules (e.g., LLM communication, voice loop, GCC I/O).

1. **Perception:** The **Lightweight Perception Module** scans the screen to produce structured text data (see Listing 1).

Listing 1: Structured game-state JSON

```
{
    "description": "The player is in combat with slimes in a forest
   area. Health is low.",
    "num_entities_nearby": 3,
    "is_player_in_combat": true,
    "is_player_in_danger": true,
    ...
}
```

2. **Orchestration (The "Conductor"):** The central **Orchestrator Module** (a VLM) receives this structured game-state data and the user's voice/text input. Its primary job is to analyze the complete context, understand the player's *intent*, and identify *proactive triggers* (e.g., "player health is critical").

3. **Delegation (via MCP):** The Orchestrator delegates the identified task via the MCP bus to a specialized sub-module:

   - **The Functional Core (The "Agent"):** Receives functional commands (e.g., "move to quest marker"). It selects the appropriate skill from the **Skill Set** and outputs the **Constrained JSON** to the **GCC I/O Module** for keyboard/mouse execution.
   - **The Relational Core (The "Companion"):** Receives conversational cues or proactive triggers (e.g., the "danger status" event). It generates an "in-character" response, which is sent to the **Voice Loop Module** and the **Frontend Module**. This frontend will use **Live2D**—a 2D animation technology standard in the VTuber industry—to provide a floating character with dynamic expressions and motions that match the agent's dialogue.

4. **Verification (The Feedback Loop):** After the Functional Core executes an action, a **Verifier Module** closes the loop. This module is inspired by the principles of error-checking frameworks like *BacktrackAgent* [30]. It performs a quick check on the *new* screen state to confirm the action was successful (e.g., "Is the inventory *now* on screen?"). If it fails, it reports the error back to the Orchestrator for replanning.

4 Project Plan

This section details the execution and validation strategy for the proposed methodology. It outlines the project's core objectives, the final deliverables, a two-phased development and evaluation plan, the project timeline, and a risk analysis.

4.1 Project Objectives

This project is defined by two primary, high-level objectives:

1. **Engineering Objective:** To design, build, and deploy a novel, real-time, unified AI companion prototype. This system will be built on a modular, **MCP-style** architecture that successfully integrates a functional "Agent" core and a proactive, "Relational" core.

2. **Validation Objective:** To execute a reproducible, phased evaluation plan to first validate the system's functional competence in controlled environments, and then to prove the core hypothesis: that the unified agent provides a measurably superior user experience in complex, dynamic environments.

4.2 Expected Deliverables

The final deliverables for this project will consist of four main components:

- **The System Prototype:** A functional source code implementation of the unified AI companion, including the central Orchestrator, all sub-modules (Functional, Relational, Safety), the perception pipeline, and the Live2D frontend.

- **Documentation:** Detailed documentation of the source code, system design, and module interactions, providing an understanding of the system's architecture and implementation.

- **The Evaluation Suite:** All scripts, configuration files, and collected data used for system validation, including the functional test suite and the qualitative user study surveys and (anonymized) results.

- **The Video Demonstration:** A recorded demo video showcasing the prototype's key functional and relational capabilities, highlighting the system's performance and user interaction in real-time.

4.3 Development Phases and Evaluation

The project will be executed in two distinct phases to manage complexity and validate the system incrementally.

Phase 1: Functional Core Validation (Simple & Logic-Based Games) The first phase will focus on building and validating the core technical stack (Perception, Orchestrator, Functional Core, Safety). This will be tested in controlled, low-rhythm, or logic-based games, such as **2048** as a unit test, or **Stardew Valley** as a more complex scenario, to isolate and debug the agent's competence. A test suite of standardized, functional tasks (e.g., "Achieve the 1024 tile," "Maps to the General Store," "Plant 10 seeds") will be created. **Success in this phase** will be measured using SOTA (State-of-the-Art) metrics from the literature, including a high **Task Success Rate (TSR)** and a low `Invalid%` of actions.

Phase 2: Unified System Validation (Complex & High-DOF Games) Once the core agent is functionally competent, the second phase will integrate the **Relational Core** (including the Live2D frontend) and transfer the full system to more complex, high-degree-of-freedom (DOF) environments like **Minecraft**. This will test the agent's ability to handle long-horizon planning and real-time threats. The core hypothesis will then be tested via an **A/B user study** with a small sample size (5-10 users)[1]. **Group A (Control)** will play with the **Functional Core only** (a "silent tool"). **Group B (Test)** will play with the **Full Unified System**. Success will be measured using a qualitative survey to assess "companionship," "proactivity," and "cognitive workload" [23], [42]. A stretch goal for this phase, if time permits, is to test the OOD transferability of the agent to a highly complex RPG like *Genshin Impact*.

4.4 Project Timeline

The project is scheduled for 18 active work units, from late November 2025 to mid-May 2026. The timeline is structured around three main development blocks: 4 units in Semester 1 (late Nov - Dec), **at least 4 units during the Winter Holiday** (Jan - Feb), and 10 units in Semester 2 (Mar - May), detailed in Figure 1. The development plan is not sequential but parallel, centered on a core "brain" (the Orchestrator) with other modules being developed, integrated, and refined concurrently. To manage this iterative process, modules are versioned (e.g., v1, v2): a 'v1' module represents an initial, stubbed-out implementation, while a 'v2' module represents a more refined, fully-integrated, and tested version.

Long-Running Development Tasks (Units 2-15) Several core components will be in development for the majority of the project. The **Orchestrator & MCP Bus** will be the first component built (v1) and will be continuously updated (v2, v3) as new modules are plugged in. The **Functional Core (RFT/PORTAL)** and the **Perception Module (YOLO/OCR)** will be developed and trained for Phase 1 games (e.g., *Stardew Valley*) during the holiday, and then undergo transfer-learning and refinement for the Phase 2 game (*Minecraft*) in Semester 2.

[1] An A/B user study involves comparing two or more versions of a product or feature to evaluate user preferences or performance. A sample size of 5-10 users is typically used in early-stage pilot studies to gather preliminary feedback before larger-scale testing.

Block 1: Core Framework (Units 1-4) The initial 4 units will be dedicated to building the "brain". This includes finalizing the **MCP API design** and implementing the initial **Orchestrator (v1)**. In parallel, a simple **Live2D Frontend (v1 stub)** and the **Voice Loop Module** will be connected to validate the core relational message-passing loop.

Block 2: Functional Core (Units 6-9) The 4-unit Winter Holiday block will be focused on the first major bottleneck: the "eyes" and "hands". The **Perception Module (v1)** for *Stardew Valley* and the **GCC I/O Module** will be built. This block is primarily dedicated to the intensive **RFT training** (or 'PORTAL' BT-generation) of the **Functional Core (v1)**.

Block 3: Integration & Validation (Units 11-20) The final 10-unit block is for integration, validation, and finalization. The **Safety Core** (inspired by 'BacktrackAgent') will be built and plugged in. The **Phase 1 (Functional) Evaluation** will be run. Concurrently, the **Perception (v2)** and **Functional (v2)** modules will be adapted for the high-DOF *Minecraft* environment. The **Relational Core (v2)** will be fully integrated with its proactive triggers. The final weeks will be dedicated to running the **Phase 2 (A/B User Study)**, followed by final code cleanup, documentation, and demo video preparation.

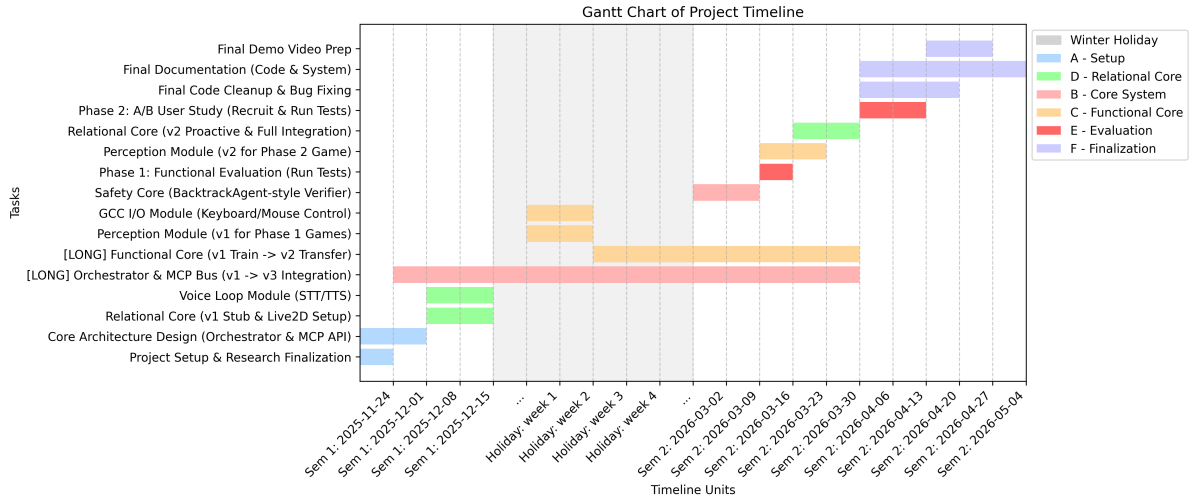

Figure 1: Gantt Chart of Project Timeline

4.5 Risk and Ethics Analysis

This plan identifies three primary technical risks and one ethical consideration:

Risk 1: Latency Real-time LLM inference (>100 ms) in Phase 2's combat scenarios will break immersion [37]. The mitigation is architectural: the system will use a hybrid model inspired by `PORTAL`, where high-frequency actions (like combat dodges) are handled by a zero-latency compiled Behavior Tree, reserving the LLM for high-level planning [35].

Risk 2: Safety & Error Accumulation The agent may get stuck in loops or fail long-horizon tasks, especially in complex 3D environments. The mitigation is a dedicated Safety Core, inspired by `BacktrackAgent`, which includes a "Verifier" and "Judger" to detect and recover from semantic errors (e.g., "player is stuck," "mob approaching"), triggering a replan [30].

Risk 3: OOD Generalization The agent may fail to transfer from Phase 1 (e.g., **Stardew Valley**) to Phase 2 (e.g., **Minecraft** or *Genshin Impact*), a known SOTA challenge [14]. This transfer is a core part of the research, and the risk is mitigated by using a game-agnostic `GCC` interface and a `PORTAL`-style policy generator, which has shown SOTA cross-game generalization [12], [35].

Ethics: Privacy The system requires capturing the user's screen and voice. The mitigation is to process all data **locally on-device**. No PII (Personally Identifiable Information) will be stored or transmitted.

References

[1] AlterStaff, *Ai2u: With you 'til the end*, https://store.steampowered.com/app/2880730/AI2U_With_You_Til_The_End/, Accessed: 2025-10-10, 2025.

[2] NVIDIA. "Nvidia ace for games - autonomous game characters." Accessed: 2025-11-05, NVIDIA Developer. [Online]. Available: https://developer.nvidia.com/ace-for-games.

[3] Ubisoft. "How ubisoft's new generative ai prototype 'neo npcs' changes the narrative." Accessed: 2025-11-05, Ubisoft News. [Online]. Available: https://news.ubisoft.com/en-gb/article/5qXdxhshJBXoanFZApdG3L/how-ubisofts-new-generative-ai-prototype-changes-the-narrative-for-npcs.

[4] J. Kim. "Bringing personality to pixels, inworld levels up game characters using generative ai," NVIDIA Blog. [Online]. Available: https://blogs.nvidia.com/blog/generative-ai-npcs/.

[5] Vedal and Neuro-sama, *Neuro-sama official youtube channel*, https://www.youtube.com/@Neurosama, Accessed: 2025-10-10, 2022.

[6] StreamElements. "State of the stream: Twitch 2024 year in review." Accessed: 2025-10-11. [Online]. Available: https://blog.streamelements.com/state-of-the-stream-twitch-2024-year-in-review-ef4d739e9be9.

[7] "Q4 2024 global live streaming landscape." Accessed: 2025-10-11, Streams Charts. [Online]. Available: https://streamscharts.com/news/q4-2024-global-livestreaming-landscape.

[8] O.-L.-V. contributors, *Open-llm-vtuber: An open-source ai vtuber framework*, https://github.com/Open-LLM-VTuber/Open-LLM-VTuber, Accessed: 2025-10-10, 2025.

[9] moeru-ai, *Airi: Ai waifu / virtual character container inspired by neuro-sama*, https://github.com/moeru-ai/airi, Accessed: 2025-10-10, 2025.

[10] kimjammer, *Neuro: A local-model recreation of neuro-sama*, https://github.com/kimjammer/Neuro, Accessed: 2025-10-10, 2025.

[11] "Vedal's ai vtuber neuro-sama sets new twitch hype train world record." Accessed: 2025-10-11, Streams Charts. [Online]. Available: https://streamscharts.com/news/vedals-ai-vtuber-neuro-twitch-hype-train-record.

[12] W. Tan *et al.*, "Cradle: Empowering foundation agents towards general computer control," *arXiv preprint arXiv:2403.03186*, 2024.

[13] Z. Gu *et al.*, "Ui-venus technical report: Building high-performance ui agents with rft," *arXiv preprint arXiv:2508.10833*, 2025.

[14] P. Guruprasad, Y. Wang, S. Chowdhury, H. Sikka, and P. P. Liang, "Benchmarking vision, language, & action models in procedurally generated, open ended action environments," *arXiv preprint arXiv:2505.05540*, 2025.

[15] D. Park *et al.*, "Orak: A foundational benchmark for training and evaluating llm agents on diverse video games," 2025, arXiv:2506.03610. arXiv: 2506.03610.

[16] Y. Wang, H. Zhang, J. Tian, and Y. Tang, "Ponder & press: Advancing visual gui agent towards general computer control," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 1461–1473.

[17] S. Agashe, J. Han, S. Gan, J. Yang, A. Li, and X. E. Wang, "Agent s: An open agentic framework that uses computers like a human," *arXiv preprint arXiv:2410.08164*, 2024.

[18] X. Wang *et al.*, "Executable code actions elicit better llm agents," in *Forty-first International Conference on Machine Learning*, 2024.

[19] X. Zheng *et al.*, "V-mage: A game evaluation framework for assessing vision-centric capabilities in multimodal large language models," *arXiv preprint arXiv:2504.06148*, 2025.

[20] L. Hu *et al.*, "Lmgame-bench: How good are llms at playing games?, 2025a," *URL https://arxiv. org/abs/2505.15146*,

[21] X. Liu *et al.*, "Visualagentbench: Towards large multimodal models as visual foundation agents," *arXiv preprint arXiv:2408.06327*, 2024.

[22] D. Nguyen *et al.*, "Gui agents: A survey," in *Findings of the Association for Computational Linguistics: ACL 2025*, 2025, pp. 22 522–22 538.

[23] C. Chen *et al.*, "Toward a human-centered evaluation framework for trustworthy llm-powered gui agents," *arXiv preprint arXiv:2504.17934*, 2025.

[24] S. Hu *et al.*, "A survey on large language model-based game agents," *arXiv preprint arXiv:2404.02039*, 2024.

[25] X. Xu *et al.*, "A survey on game playing agents and large models: Methods, applications, and challenges. arxiv pre-print," *arXiv preprint arXiv:2403.10249*, 2024.

[26] W. Zhang *et al.*, "Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving," *arXiv e-prints*, arXiv–2506, 2025.

[27] K.-T. Tran, D. Dao, M.-D. Nguyen, Q.-V. Pham, B. O'Sullivan, and H. D. Nguyen, "Multi-agent collaboration mechanisms: A survey of llms," *arXiv preprint arXiv:2501.06322*, 2025.

[28] X. Hu *et al.*, *Os agents: A survey on mllm-based agents for computer, phone and browser use*, 2024.

[29] Z. Durante *et al.*, "Agent ai: Surveying the horizons of multimodal interaction," *arXiv preprint arXiv:2401.03568*, 2024.

[30] Q. Wu, P. Gao, W. Liu, and J. Luan, "Backtrackagent: Enhancing gui agent with error detection and backtracking mechanism," *arXiv preprint arXiv:2505.20660*, 2025.

[31] S. Lian *et al.*, "Ui-agile: Advancing gui agents with effective reinforcement learning and precise inference-time grounding," *arXiv preprint arXiv:2507.22025*, 2025.

[32] Z. Xi *et al.*, "Agentgym-rl: Training llm agents for long-horizon decision making through multi-turn reinforcement learning," *arXiv preprint arXiv:2509.08755*, 2025.

[33] Z. Peng *et al.*, "Improving agent behaviors with rl fine-tuning for autonomous driving," in *European Conference on Computer Vision*, Springer, 2024, pp. 165–181.

[34] Y. Jin *et al.*, "Read to play (r2-play): Decision transformer with multimodal game instruction," *arXiv preprint arXiv:2402.04154*, 2024.

[35] Z. Xu *et al.*, "Agents play thousands of 3d video games," *arXiv preprint arXiv:2503.13356*, 2025.

[36] M. X. Liu *et al.*, """ we need structured output": Towards user-centered constraints on large language model output," in *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2024, pp. 1–9.

[37] J. Liu *et al.*, "Llm-powered hierarchical language agent for real-time human-ai coordination," *arXiv preprint arXiv:2312.15224*, 2023.

[38] M. Zubia, T. D. Simão, and N. Jansen, "Robustifying RL agents for safe transfer through action disturbances," in *Proceedings of the BNL Conference on Artificial Intelligence and Machine Learning (BNAIC/BeNeLearn 2024)*, 2024. [Online]. Available: https://bnaic2024.sites.uu.nl/wp-content/uploads/sites/986/2024/10/Robustifying-RL-Agents-for-Safe-Transfer-through-Action-Disturbances.pdf.

[39] F. Tang *et al.*, "A survey on (m) llm-based gui agents," *arXiv preprint arXiv:2504.13865*, 2025.

[40] E. Sun and Z. Wu, "Systematizing llm persona design: A four-quadrant technical taxonomy for ai companion applications," *arXiv preprint arXiv:2511.02979*, 2025.

[41] G. Gamage, D. De Silva, N. Mills, D. Alahakoon, and M. Manic, "Emotion aware: An artificial intelligence framework for adaptable, robust, explainable, and multi-granular emotion analysis," *Journal of Big Data*, vol. 11, no. 1, p. 93, 2024.

[42] J. Lundström, *Cooperation between player and ai: The effect of ai companions' level of autonomy on the player experience*, 2024.

Appendix A. Title of Appendix A

A.1 Appendix Heading 1

Text of the appendix goes here

A.2 Appendix Heading 2

Text of the appendix goes here

A.3 Appendix Table and Figure Captions

In appendices, table and figure caption labels and numbers are typed in manually (e.g., Table A1, Table A2, etc.). These do not get generated into the lists that appear after the Table of Contents.

Appendix B. Title of Appendix B

Text of the appendix goes here if there is only a single heading.