Mystic AI - a Text-based Mystery Game

Saurav Raj Pandey
University of North Carolina at Chapel Hill
srpandey@cs.unc.edu

Sagar Manjunath
University of North Carolina at Chapel Hill
sagarbm@cs.unc.edu

Abstract

We present a text-based murder mystery game powered by large language models (LLMs), where the player takes on the role of a detective tasked with solving a murder. The LLM dynamically creates the mystery, defines the characters, and role-plays each suspect in real time during interrogation. Unlike traditional games with scripted plots, our system offers an open-ended narrative experience driven by natural language interaction. The player can freely question characters, gather clues, and ultimately accuse one of the suspects. Our project explores how LLMs can support immersive, interactive storytelling and highlights the opportunities and limitations of using LLMs for consistent role-playing, world-building, and narrative logic.

1 Introduction

1.1 Why Build This Software?

Games are a compelling medium for exploring human-computer interaction, and narrative-driven games in particular benefit from advances in natural language understanding and generation. We wanted to explore how generative AI can enable richer, more personalized interactive stories — going beyond traditional hardcoded branching dialogues. The idea of a murder mystery game was especially appealing: it provides a natural structure with roles (suspects, detective), objectives (solve the mystery), and interactivity (interrogation and deduction) that aligns well with the strengths of LLMs. By building this software, we aimed to experiment with the creative and technical challenges of using language models to generate and sustain a coherent fictional world in real time.

1.2 Why Use LLMs?

Large language models are uniquely suited for dynamic dialogue generation, contextual memory, and creative writing—making them ideal candidates for powering a narrative game. Unlike rule-based or template-driven systems, LLMs can respond flexibly to arbitrary player input, maintain character personas, and adapt the story flow based on emerging gameplay. In our game, the LLM is used both at setup (to create the mystery and define the suspects) and during gameplay (to generate character responses and maintain narrative coherence). This dual use leverages the model's strengths in few-shot prompting, improvisational dialogue, and semantic understanding. LLMs thus serve not just as a backend logic engine, but as the primary storyteller and actor ensemble driving the gameplay experience.

Preprint. Under review.

2 Related works

Text-based games and puzzles are not a new concept. Games like Zork, Stein's gate and Planetfall use text-based interactions to navigate through the game. Websites like solvethemurder.com provide text-based murder mystery games. However, these approaches typically involve puzzles generated by humans and limited text options for the users. Given that LLMs are a recent development, there is a lot of room for modernizing these text-based games.

3 Methods

3.1 System Design

Our system consists of a FastAPI backend and a JavaScript-based frontend web application. Both components were developed with the assistance of OpenAI's GPT-4o model. The backend exposes three core APIs:

- **Setup API**: Initializes a new game by generating the setting, characters, and narrative context based on user-specified inputs.
- **Ground Truth API**: Constructs the underlying solution to the murder mystery, including the identity of the perpetrator, their motive, the method of the crime, and the distribution of relevant clues.
- **Conversation API**: Enables dynamic, in-character conversations between the player and the suspects.

The frontend is responsible for rendering the generated narrative and character profiles, managing user interactions, and enforcing a constraint on the number of questions a player can ask before making an accusation. This restriction encourages thoughtful interrogation and prevents infinite loops of questioning.

3.2 Game Logic

At the start of a game, the user is prompted to select three parameters:

1. A **difficulty level** on a scale from 1 to 10, which controls the subtlety of clues and complexity of the motive.
2. A **setting**, such as a supermarket, theater, swimming pool, or even a broader environment like a city.
3. A **murder method**, such as stabbing or poisoning.

Once submitted, the backend invokes the Setup API to generate a narrative scenario that satisfies the user's criteria. The frontend immediately renders this story to minimize user-perceived latency. Simultaneously, the Ground Truth API is called asynchronously to produce the underlying structure of the mystery: the killer's identity, their motive, how the murder was committed, and a set of clues designed to support the player's reasoning process.

These clues are critical for game solvability. In early iterations, suspects uniformly denied involvement, making it nearly impossible to deduce the perpetrator. By explicitly embedding subtle but consistent clues into the story and suspect responses, the game now rewards careful reading and logical deduction.

The player may interrogate suspects in any order. Each interaction is mediated by the Conversation API, which ensures that suspect responses remain consistent with the ground truth and with the evolving dialogue history. Once the player makes an accusation, the frontend reveals the ground truth and indicates whether the player's guess was correct.

Figures 1 and 2 illustrate the user interface corresponding to the gameplay pipeline described above. As shown in Figure 1, players begin by selecting the mystery parameters, which trigger story generation and backend setup. Figure 2 captures the main gameplay state, where players interact with suspects and analyze the unfolding narrative. After an accusation is made, the interface reveals the

ground truth, including the killer's identity, motive, and supporting clues, aligning visual feedback with the reasoning-driven structure enforced by the underlying APIs

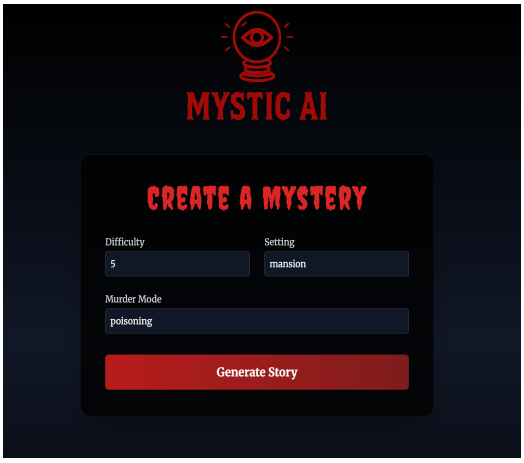

Figure 1: Mystic AI Home Screen

Figure 2: Mystic AI Gameplay Screen

Figure 3: Mystic AI Example Game Run

3.3 Prompts and LLMs Used

We used Google's Gemini family of models for all generation tasks. Specifically:

- **Gemini 2.0 Flash** was used for generating the initial setup and handling suspect conversations. We found that this model was fast and responsive, and it generally maintained coherence across interactions. However, it struggled to insert meaningful clues into character dialogue or descriptions.
- **Gemini 2.5 Flash** was therefore used for ground truth generation, where clue quality and logical consistency were essential. This model demonstrated a better grasp of narrative structure and produced clues that aligned well with both the motive and method of the murder.

Prompt engineering was a key part of our development process. We used few-shot examples and structured templates to ensure that the LLMs followed consistent formats and respected constraints. For character conversations, prompts included background context, personality traits, and a reminder to embed or reference clues consistent with the ground truth. The Setup and Ground Truth APIs required the LLM to generate text in JSON format.

4 Evaluation

Evaluating a narrative-based game poses unique challenges due to the . To systematically assess puzzle solvability and difficulty scaling, we designed an automated evaluation setup where a large language model (LLM) plays the role of the detective.

4.1 Automated Evaluation Framework

We evaluated four different LLMs — `gemini-2.0-flash`, `gemini-2.0-flash-lite`, `gemini-1.5-flash`, and `gemini-1.5-flash-8b`—across five difficulty levels: 1, 2, 5, 8, and 10. For each combination of model and difficulty, we ran four trials using newly generated puzzles each time. We evaluated two modes of play:

- **0-question mode:** The LLM was given only the setup and character descriptions, and asked to immediately identify the murderer.

- **6-question mode:** The LLM was allowed to interrogate suspects up to six times via the Conversation API before making an accusation.

This grid of conditions allowed us to assess how each model handled reasoning under different levels of uncertainty and clue availability.

4.2 Results and Analysis

We summarize performance in Figure 6, where each cell reflects the percentage of trials in which the LLM correctly identified the culprit. Our key findings were:

1. **Accuracy drops with difficulty:** As expected, all models exhibited a decline in performance as the puzzle difficulty increased, in both interaction modes.
2. **Zero-shot reasoning is effective at low difficulties:** Even without asking questions, LLMs were often able to solve easier puzzles using descriptive clues embedded in the setup.
3. **Interrogation improves performance:** Allowing the LLM to ask six questions significantly increased success rates, especially on mid- and high-difficulty puzzles.

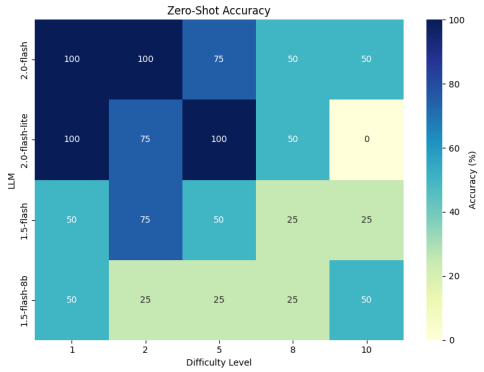

Figure 4: (a) Accuracy with 0 questions

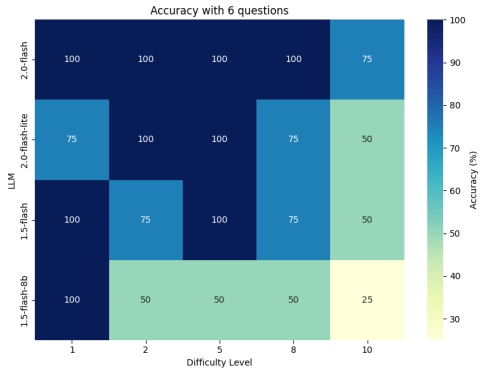

Figure 5: (b) Accuracy with 6 questions

Figure 6: LLM detective performance across difficulty levels and model types.

5 Conclusion

In this project, we explored the use of large language models (LLMs) to generate and facilitate an interactive, text-based murder mystery game. Our system leverages LLMs not only to craft coherent and solvable mystery narratives, but also to embody suspect characters during player interrogation. We designed the game to be modular, with a FastAPI backend and a lightweight frontend, and used different LLMs for generation and interaction to balance creativity and control.

Our evaluation, conducted by having LLMs act as detective agents, demonstrated that the generated puzzles are solvable, with accuracy scaling predictably with difficulty and available interaction. We found that even without questioning, LLMs were able to solve many puzzles at low difficulty levels. When allowed to interrogate suspects, LLMs significantly improved their performance—showcasing the importance of interactive reasoning in narrative-based tasks.

This project highlights the potential of LLMs in dynamic storytelling and game design, and serves as an example of how such models can be used to create engaging and logically consistent interactive experiences. In future work, we hope to explore more sophisticated narrative structures, incorporate player feedback for puzzle adaptation, and evaluate human player engagement at scale.