Bootcamp

Inter IIT Tech Meet 14.0

AI/ML PROBLEM STATEMENT

Team Size : Up to 2 members

Submission Deadline : Oct 10th, 12:00 PM

Presentations: Oct 11th

Machine Learning/AI

Text-based storytelling in tabletop role-playing games relies on one key figure — the Dungeon Master (DM) — who weaves the story, manages characters, and maintains continuity across adventures. A skilled DM ensures that every choice matters, every consequence persists, and the world evolves coherently over time.
However, large language models (LLMs), while capable of generating rich narratives, often struggle to sustain long-term coherence. They forget past events, contradict earlier scenes, or lose consistency across sessions.
This project aims to design an AI Dungeon Master that can deliver persistent, interactive storytelling through an intelligent memory architecture. The goal is to create a system that maintains context across turns, preserves player choices, and supports immersive, evolving gameplay experiences.

Objectives:

1. Vision and Design Goals:
    - Build an intelligent narrative system that combines creativity with consistency.
    - Enable the AI to behave like a true Dungeon Master — remembering events, characters, and evolving world states.
    - Enhance immersion and engagement by maintaining continuity across multiple sessions.
2. Core Challenge — Memory Architecture:
    - Working Memory (Short-Term): Retain recent scene details and player actions (≈5 turns).
    - Persistent Memory (Long-Term): Recall crucial past events (~30 turns) to ensure world and character consistency.
    - Implement efficient retrieval mechanisms such as Retrieval-Augmented Generation (RAG) or any creative memory design that balances accuracy and scalability.
3. Technical Focus:
    - Integrate an LLM engine (self-hosted via systems like llama.cpp or external free APIs like Groq).
    - Design a modular, interpretable, and scalable memory system.
    - Emphasize clarity, personalization, and stability over long-form interaction.
    - Ensure all code is clean, documented, and reproducible for evaluation.

Evaluation Criteria

Projects will be evaluated through live demonstrations, automated testing, code review, and presentations.

The total score is 100 points + up to 15 bonus points.

A. Gameplay & Robustness (25 Points)

- Gameplay Quality (15): Engagement, narrative coherence, and depth of interaction.
- Application Stability (10): Must sustain 30 turns without crashing or producing incoherent output.

B. Memory Management (40 Points)

- Short-Term Recall (15): Accuracy in tracking recent actions or dialogue.
- Long-Term Recall (25): Consistency in recalling past events introduced earlier in the story (~30 turns).

C. Architecture & Code Quality (15 Points)

- System Design (10): Efficiency, modularity, and clarity of architecture.
- Code Clarity (5): Readability, documentation, and maintainability.

D. Presentation & Reporting (20 Points)

- Presentation (10): Clear explanation of design, architecture, and live demo.
- Technical Report (10): Concise (2–4 pages), highlighting methodology, memory design, and outcomes.

Bonus (Up to 15 Points)

- Character Memory (+10): NPCs recall prior interactions and evolve accordingly.
- Dynamic Quest Log (+5): Automatically updates and references player progress and quests.

Deliverables

Each team must submit a GitHub repository containing:

- Source Code: Fully functional and modular implementation.
- README.md: Setup guide, system overview, and usage instructions.
- Recording: Short demonstration showing both short-term and long-term recall, as a link in the readme.
- Technical Report: 2–4 pages (PDF/Word/LaTeX) covering system design, memory approach, and evaluation.