# Technical Report – Advanced AI : Dungeon Master

## **Introduction:**

Problem Statement:  
Interactive storytelling has always relied heavily on human creativity, limiting scalability and requiring manual effort. Our project addresses the challenge of automating a Dungeon Master (DM) experience using artificial intelligence to dynamically generate narratives, control world logic, and interact meaningfully with players in real time.

### Motivation:

We both share an interest in D&D and gaming, so we thought choosing this subject for the project would be interesting and fun at the same time. We wanted to create an AI that has knows the players stats and equipment and can change these accordingly.

### Approach:

We developed a Python-based game that uses Gemini-2.0-flash to handle narrative generation, while maintaining structured game logic for stats, equipment, and combat. The AI generates story outputs and game state updates in JSON format, allowing for consistent interaction between the player and the game world.

### Link to github repository:

<https://github.com/ihno999/avanced_ai_semester_project>

## **Data:**

Description:  
Our "data" consists of dictionaries for items, equipment slots, character stats, and spell definitions. Instead of traditional datasets, we built rule-based systems to constrain and guide the model’s behavior.

Preprocessing:  
We created:

* A stat-boosting item dictionary (item\_stats.py)
* A structured spell list with mana costs and effects

Challenges:  
The biggest challenge was preventing hallucinations from the LLM, such as it inventing non-existent items or misformatting JSON. This required iterative prompt engineering and runtime validation logic to catch and reject bad responses.

## **Model & Methods**

### Techniques Used:

* **Gemini-2.0-flash** for natural language generation
* **Prompt Engineering** to embed game logic (stat calculations, JSON formatting, valid equipment slots)
* **Stat System** using RPG mechanics (e.g., strength affects attack, defense reduces damage)

### Key Logic Features:

* Damage = base\_damage + (strength × 0.5)
* Damage Taken = incoming × (1 - defense × 0.03)
* JSON state updates handled via custom <META>{...}</META> tags
* Equipment limited to 8 slots, all validated against item\_stats.py

## **Results & Evaluation:**

### Performance:

* Generated coherent, interactive narratives 90–95% of the time
* Successfully maintained a consistent game state across dozens of interactions
* High engagement due to dynamic responses and meaningful stat impact

## Insights:

* Strict JSON formatting reduced AI freedom but increased system stability
* Prompt tuning was essential for balancing creativity and control

## **Contributions:**

## **Challenges & Future Work**

### Challenges:

* Ensuring Gemini output was always JSON-compliant
* Preventing cheating or unrealistic actions ("I win instantly")
* Avoiding logic loops or repeated narrative patterns

### Future Improvements:

* Add turn-based combat with enemy AI
* Expand spellcasting system with cooldowns
* Implement multiplayer support
* Integrate visuals or map-based exploration