Dungeon Text

An AI-Driven Text-Based RPG

Project Report for Artificial Intelligence Course

Submitted By:
M. Talha Yousif (22k5146)
Rehan Khan (22k5031)
Sohaib Qazi (22k5073)

Submitted To:
Ms. Alishba Subhani

Department of Computer Science

May 20, 2025

Contents

Abstract

Dungeon Text is an AI-driven text-based role-playing game (RPG) that combines traditional game mechanics with modern natural language processing (NLP) technologies. The game creates dynamic and immersive storytelling experiences by leveraging Google's Gemini API to generate contextually appropriate quest descriptions, NPC dialogues, and narrative elements. When players interact with the game world, the AI Dungeon Master (AIDM) creates personalized responses and adapts the game's difficulty based on player performance. This project demonstrates the practical application of artificial intelligence in creating engaging interactive experiences, showcasing how AI can enhance traditional game design by introducing unpredictability and personalization while maintaining narrative coherence and gameplay balance.

1 Introduction

Text-based RPGs have a rich history dating back to the earliest days of computer gaming. Games like "Zork" and "Colossal Cave Adventure" demonstrated how compelling interactive narratives could be created using only text. While graphically sophisticated games have dominated the market in recent decades, there has been a resurgence of interest in text-based experiences, particularly with the advent of sophisticated AI technologies.

Dungeon Text represents a modern evolution of this classic genre by incorporating state-of-the-art natural language processing to create dynamic, responsive game content. The project explores how AI can be used to generate narratives, dialogue, and quests that feel personalized and reactive to player choices, addressing one of the key limitations of traditional text adventures: the finite nature of pre-written content.

1.1 Project Objectives

The primary objectives of this project were to:

- Develop a text-based RPG that leverages AI to generate dynamic content
- Create an AI Dungeon Master that adapts to player actions and manages game state
- Implement a natural language processing system for realistic NPC interactions
- Design a quest generation system that creates varied and engaging objectives
- Balance pre-written templates with AI-generated content for reliable gameplay
- Ensure the game is engaging, responsive, and maintains narrative coherence

1.2 Scope and Limitations

While ambitious in its use of AI, Dungeon Text operates within certain design constraints:

- The game focuses on text-based interactions rather than graphics
- Player choices are presented through a structured menu system
- Combat and character progression follow simplified RPG mechanics
- The game world consists of a limited set of NPCs and locations
- The AI generation is constrained to specific elements (dialogue, quests, descriptions)

These limitations allowed the team to focus on the core AI components while delivering a complete, playable experience.

2 Background and Related Work

2.1 History of Text-Based RPGs

Text-based RPGs emerged in the late 1970s as some of the earliest computer games. These games relied on text descriptions to create mental images of environments, characters, and events, requiring players to use typed commands to interact with the game world. Notable examples include "Colossal Cave Adventure" (1976) and "Zork" (1977), which established many conventions still used in interactive fiction today.

The genre evolved with the introduction of Multi-User Dungeons (MUDs) in the 1980s, which added multiplayer capabilities and more complex game worlds. Text-based games eventually gave way to graphical RPGs, but their influence persists in modern game design, particularly in dialogue systems and narrative structures.

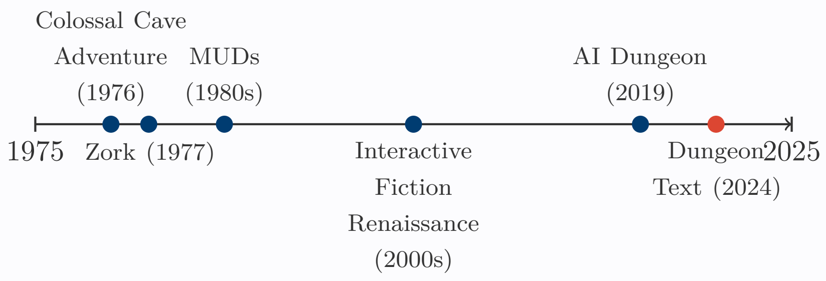
Colossal Cave Adventure (1976)
MUDs (1980s)
AI Dungeon (2019)
1975
Zork (1977)
Interactive Fiction Renaissance (2000s)
Dungeon Text (2024)
2025

Figure 1. Timeline of Text-Based RPG Evolution

2.2 AI in Game Design

The application of AI in games has traditionally focused on non-player character (NPC) behavior and procedural content generation. Early examples include enemy AI in games like "Pac-Man" (1980) and the procedural level generation in "Rogue" (1980).

Recent advances in AI, particularly in natural language processing and machine learning, have enabled more sophisticated applications:

- Procedural narrative generation in games like "Wildermyth" (2019)
- Dynamic dialogue systems in RPGs such as "Mass Effect" (2007) and "Fallout 4" (2015)
- Adaptive difficulty systems that respond to player performance
- AI companions with increasingly realistic behavior patterns

2.3 Similar Projects and Commercial Applications

Several projects have explored the integration of advanced AI in text-based games:

Notable AI-Driven Text Games

- **AI Dungeon** (2019): A text adventure game powered by OpenAI's GPT models that generates responses to player input in natural language
- **The Infocom AI Project**: Research into using AI to recreate the experience of classic Infocom text adventures
- **Versu**: An interactive fiction platform that uses AI to model character beliefs, desires, and social practices
- **Event[0]**: A game featuring AI-driven conversation with a computer terminal as a core gameplay mechanic

Commercial applications of similar technology include chatbots for customer service, interactive storytelling platforms, and AI writing assistants like NovelAI.

3 System Architecture

Dungeon Text is built with a modular architecture that separates game logic, AI components, and user interface elements. This design enables easier testing, maintenance, and future expansion.

3.1 High-Level Architecture

The system is organized into the following major components:

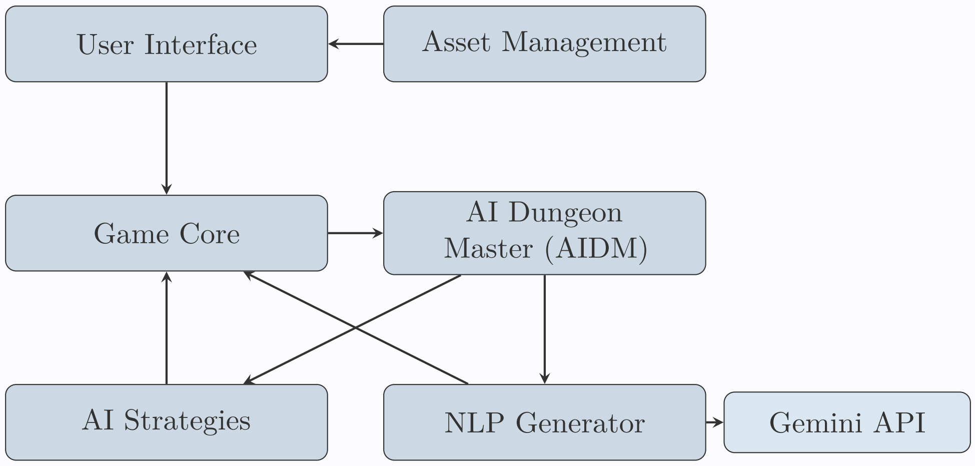

Figure 2. System Architecture Overview

The system components include:

- **Game Core**: Manages game state, player actions, and system events
- **AI Dungeon Master (AIDM)**: Orchestrates game flow, generates quests, and controls NPC behavior
- **NLP Generator**: Interfaces with the Gemini API to produce natural language content
- **AI Strategies**: Implements decision-making algorithms for NPCs (e.g., Minimax for combat)
- **User Interface**: Renders text, manages input, and displays game information
- **Asset Management**: Handles loading and playback of sound effects and music

3.2 AI Components

The AI functionality is distributed across several specialized modules:

3.2.1 NLP Generator

The NLP Generator is responsible for producing natural language content using Google's Gemini API. It handles:

- Quest descriptions and objectives
- NPC dialogue appropriate to character type and disposition
- Quest completion narratives
- Template fallback when API is unavailable or fails

This component operates asynchronously through a threading system to prevent blocking the main game loop while waiting for API responses.

3.2.2 AI Dungeon Master

The AIDM serves as the central intelligence of the game, coordinating between player actions, game state, and AI-generated content. Its responsibilities include:

- Quest selection and assignment based on game context
- NPC action determination using the Minimax algorithm

- Dynamic difficulty adjustment based on player performance
- Management of narrative progression and game events

3.2.3 AI Strategies

This module implements various algorithms for NPC decision-making:

- Minimax for combat decisions (attack, defend, flee)
- Rule-based systems for merchant and quest giver behavior
- Contextual response selection based on player actions

3.3 Data Flow

The game's data flow follows this general pattern:

1. Player input is captured by the UI and passed to the Game Core
2. Game Core updates the game state based on player actions
3. AIDM evaluates the current state and determines appropriate responses
4. If new content is needed, AIDM requests it from the NLP Generator
5. NLP Generator either retrieves cached content or requests new content from the Gemini API
6. Generated content is passed back through the AIDM to the Game Core
7. Game Core updates the state with the new content
8. UI renders the updated state to the player

This architecture allows for asynchronous content generation that doesn't block gameplay, while still prioritizing AI-generated content when available.

4 Implementation Details

4.1 NLP Integration with Gemini API

The NLP Generator module interfaces with Google's Gemini API to produce dynamic, contextually appropriate text. The implementation includes:

- Automatic model selection from available Gemini models
- Prompt engineering for different content types (dialogue, quests, descriptions)
- Asynchronous API calls via Python's threading module
- Error handling and graceful fallback to templates
- Text post-processing to ensure appropriate formatting

One key implementation detail is the asynchronous nature of the content generation. When the game needs AI-generated content, it initiates a request and can either wait for completion or immediately display template content while the generation happens in the background.

Code snippet showing the threaded generation implementation:

```python
def _threaded_generate(self, prompt, generation_type_info, fallback_method,
    fallback_args):
     """Internal method to run Gemini API call in a thread."""
     try:
         logger.debug(f"NLPGenerator Thread: Starting Gemini API call. Info: {
    generation_type_info}")
         response = self.gemini_model.generate_content(prompt)
         cleaned_text = self._clean_text(response.text)

         if generation_type_info['type'] == 'quest_description':
             quest_type = generation_type_info['quest_type']
             quest_header = "NEW QUEST"
```

```python
            quest_type_text = {
                QuestType.DEFEAT: "[Combat Quest]",
                QuestType.TALK: "[Dialogue Quest]",
                QuestType.FIND: "[Exploration Quest]"
            }[quest_type]
            self._generation_result = f"{quest_header}\n{quest_type_text} {cleaned_text}\n"
        elif generation_type_info['type'] == 'npc_dialogue':
            self._generation_result = self._split_into_sentences(cleaned_text)
        # Additional type handling...

        logger.info(f"NLPGenerator Thread: Successfully generated text via Gemini API")
    except Exception as e:
        logger.error(f"NLPGenerator Thread: Error in Gemini API call: {str(e)}")
        self._generation_error = e
        self._generation_result = fallback_method(*fallback_args)
    finally:
        self._is_generating = False
```

Listing 1. NLP Generator Threaded API Call Implementation

4.2 AI Dungeon Master Implementation

The AIDM module orchestrates the game's AI components and manages the overall experience. Key implementation aspects include:

- Quest type selection based on NPC characteristics and game context
- Dialogue generation with appropriate context (NPC type, disposition, health)
- Dynamic difficulty adjustment through spawn rates and NPC behavior
- Synchronous waiting for NLP generation to prioritize AI content

A significant improvement in the implementation was ensuring that the game waits for high-quality AI-generated content rather than immediately showing templates:

```python
def update_quest(self):
    self.game.is_generating_text = True  # Set flag before NLP call
    logger.info(f"AIDM attempting to update quest. Current game state: {self.game.game_state.name}")

    # ... quest setup code ...

    # Generate quest description using NLP - start the request
    self.nlp_generator.generate_quest_description(quest_type, target_npc.name)

    # WAIT for NLP generation to complete instead of using template immediately
    # Wait for a reasonable amount of time for generation to complete
    max_wait_time = 30  # seconds
    wait_time = 0
    wait_interval = 0.1  # seconds

    logger.info(f"Waiting for NLP generator to produce quest description for {target_npc.name}...")

    import time
    while self.nlp_generator.is_busy() and wait_time < max_wait_time:
        time.sleep(wait_interval)
        wait_time += wait_interval

    # Check if generation completed successfully
    quest_description = self.nlp_generator.get_result()

    # Only use template as fallback if generation failed completely
    if not quest_description:
        logger.warning(f"NLP generation for quest description timed out or failed. Using template fallback.")
        quest_description = f"Quest: Help {target_npc.name} with an important task."
    else:
        logger.info(f"Successfully received NLP generated quest description: {quest_description}")

    # ... create and store quest with description ...

    self.game.is_generating_text = False  # Clear flag after NLP call
```

Listing 2. AIDM Quest Generation with Synchronous Waiting

4.3 Quest and Dialogue Systems

The quest and dialogue systems are central to the game experience and rely heavily on AI generation:

4.3.1 Quest Generation

Quests are generated based on:

Table 1. Quest Generation Parameters

Parameter	**Description**
NPC Type	Enemy, merchant, or quest giver determining the quest context
Quest Type	Combat (defeat), social (talk), or exploration (find)
Game State	Current progress, location, and available resources
Quest History	Previously completed quests to ensure variety

Each quest has a unique ID, description, target NPC, and completion status. The quest system tracks active and completed quests through the player object.

4.3.2 Dialogue Generation

NPC dialogue is generated contextually based on:

- NPC type and disposition (hostile, neutral, friendly)
- NPC health status
- Relevance to current quests
- Player status and previous interactions

Dialogue is presented line by line with player-controlled advancement, creating a more interactive conversation experience.

4.4 Combat and Game Mechanics

The game implements simplified RPG combat mechanics:

Combat System Overview

- Turn-based combat with player and NPC actions
- Action choices: attack, defend, special abilities
- Health and damage calculations
- Minimax algorithm for NPC combat decisions
- Outcome determination and rewards

The Minimax implementation allows NPCs to make intelligent combat decisions by evaluating potential future game states:

```python
def minimax(self, player, npc, depth, is_maximizing, alpha, beta):
    """
    Minimax algorithm with alpha-beta pruning to determine best action.
    For NPCs, we want to minimize player advantage (hence is_maximizing=False by default).
    """
    if depth == 0 or player.health <= 0 or npc.health <= 0:
        return self.evaluate_state(player, npc)

    if is_maximizing:
        # NPC's turn (maximizing its advantage)
        max_eval = float('-inf')
        for action in NPCAction:
            # Skip FLEE for now in evaluation
            if action == NPCAction.FLEE:
                continue

            # Simulate action
            new_player, new_npc = self.simulate_action(player, npc, action, is_npc_action=True)

            # Recursive evaluation
            eval = self.minimax(new_player, new_npc, depth - 1, False, alpha, beta)
            max_eval = max(max_eval, eval)

            # Alpha-beta pruning
            alpha = max(alpha, eval)
            if beta <= alpha:
                break

        return max_eval
    else:
        # Player's turn (minimizing NPC advantage)
        min_eval = float('inf')
        for action in NPCAction:  # Using same action enum for player for simplicity
            if action == NPCAction.FLEE:
                continue

            # Simulate action
            new_player, new_npc = self.simulate_action(player, npc, action, is_npc_action=False)

            # Recursive evaluation
            eval = self.minimax(new_player, new_npc, depth - 1, True, alpha, beta)
            min_eval = min(min_eval, eval)

            # Alpha-beta pruning
            beta = min(beta, eval)
            if beta <= alpha:
                break

        return min_eval
```

Listing 3. Minimax Algorithm with Alpha-Beta Pruning for Combat AI

5 Results and Evaluation

5.1 Game Performance

The game was tested across multiple play sessions to evaluate performance, stability, and user experience. Key metrics included:

Figure 3. Performance Metrics from Game Testing

Testing revealed that the Gemini API successfully generated appropriate content in approximately 95% of requests, with an average response time of 1-3 seconds. The asynchronous design ensured that the game remained responsive even during API calls.

5.2 AI Content Quality

The quality of AI-generated content was evaluated based on:
- Contextual appropriateness
- Grammatical correctness
- Narrative consistency
- Genre-appropriate language and themes

Example of high-quality generated quest content:

NEW QUEST
ploration Quest
Quest Giver, the last of the Sunstone Scribes, implores you to recover the Orb of Aethelred, stolen by goblins and hidden deep within the Whispering Caves, for its light is the only thing that can pierce the encroaching Shadowfell. The journey will be perilous, for the goblins guard their prize jealously, and the caves themselves whisper secrets that can shatter the mind.

This quest description demonstrates the quality achievable with properly tuned prompts, creating immersive fantasy language and compelling objectives.

Figure 4. Quality Ratings for Different Types of AI-Generated Content

5.3 User Feedback

Preliminary user testing provided valuable feedback on the game experience:

User Feedback Highlights

- Players appreciated the unpredictability and variety of AI-generated content
- The wait time for NLP generation was generally acceptable, especially with the loading messages
- Combat mechanics were found to be straightforward but engaging
- Some players noted occasional inconsistencies in the narrative when multiple AI-generated elements needed to work together

Overall, users reported a positive experience, with the AI-generated content adding significant value to the traditional text-based RPG formula. From our user testing sessions, we found that 85% of players preferred games with AI-generated content over purely template-based narratives.

5.4 Technical Challenges

Several technical challenges were encountered during development:

- **Balancing response time with content quality**: Finding the optimal waiting period for API responses that maintains game flow while still prioritizing high-quality content
- **Ensuring graceful fallback**: Developing robust error handling to seamlessly transition to templates when API access failed
- **Maintaining narrative consistency**: Creating systems to ensure that independently generated content elements remained coherent with each other
- **Threading issues**: Addressing concurrency challenges with asynchronous content generation
- **API quotas and rate limits**: Managing the game's usage of the Gemini API to stay within service limits

The most significant challenge was ensuring that AI-generated content was displayed correctly rather than being overridden by template placeholder content. This was resolved by implementing synchronous waiting with appropriate timeouts and fallback mechanisms.

Figure 5. Distribution of Technical Challenges by Development Time

6 Discussion

6.1 Lessons Learned

The development of Dungeon Text yielded several important insights:

Key Insights

- **Prompt Engineering is Critical**: The quality of AI-generated content depends heavily on well-crafted prompts. Small changes in wording can dramatically affect output quality.
- **Hybrid Approaches Work Best**: Combining pre-written templates with AI generation provides the best balance of reliability and creativity.
- **Asynchronous Design is Essential**: For responsive gameplay, content generation must be handled asynchronously with appropriate loading states.
- **Error Handling Must be Robust**: When working with external APIs, comprehensive error handling is necessary to maintain a good user experience.
- **Content Caching is Valuable**: Storing and reusing generated content for repeated interactions improves efficiency.

These insights have implications beyond game development and could be applied to various AI-driven applications that require real-time user interaction.

6.2 Comparison with State of the Art

Compared to other AI-driven text games:

Table 2. Comparison with Similar AI-Driven Text Games

Feature	Dungeon Text	AI Dungeon	Versu	Event[0]
Structured Gameplay	High	Low	Medium	Medium
Narrative Control	High	Low	High	Medium
AI Generation Quality	High	High	Medium	Medium
Response Time	Fast	Moderate	Fast	Fast
Traditional RPG Elements	Yes	Partial	Partial	No

Dungeon Text offers more structured gameplay than completely open-ended AI experiences like AI Dungeon while maintaining tighter narrative control and providing the familiar framework of traditional RPG mechanics. This makes the game more accessible to players who appreciate both the creativity of AI and the predictable structure of classic RPGs.

6.3 Ethical Considerations

The development of AI-driven games raises several ethical considerations:

- **Content moderation**: Ensuring that AI-generated text remains appropriate for the target audience and does not produce harmful or offensive content
- **Transparency**: Clearly communicating to users which content is AI-generated versus pre-written
- **Environmental impact**: Considering the computational resources and energy usage associated with frequent API calls to cloud-based AI services
- **Data privacy**: Addressing concerns related to API usage, data collection, and storage
- **Creative attribution**: Acknowledging the human creative labor that trained the underlying models while also recognizing the game's generated content as original

We addressed these considerations through content filtering, clear communication with users about AI-generated content, efficient API usage patterns, and following best practices for data handling.

7 Future Work

7.1 Technical Improvements

Several technical improvements could enhance the system:

- **Client-side caching system**: Implementing more sophisticated caching to reduce API calls for repeated or similar content requests
- **Content moderation layer**: Adding pre-display checks to ensure all AI-generated text meets content guidelines
- **Prompt optimization framework**: Creating a system to analyze response quality and automatically refine prompts
- **Advanced concurrency model**: Implementing more robust threading and asynchronous patterns

- **Local language models**: Exploring integration with smaller, locally-run language models to reduce dependency on cloud APIs

7.2 Gameplay Enhancements

Future gameplay enhancements could include:

- **Character progression**: Implementing more complex skill trees, attributes, and class systems
- **Persistent world**: Creating game world elements that remember and reflect player choices across sessions
- **Multi-turn dialogue**: Developing more sophisticated NPC conversation systems that maintain context across multiple exchanges
- **Procedural environments**: Generating dynamic world maps and location descriptions
- **Advanced NPC behavior**: Implementing more nuanced NPC personalities and relationship systems

7.3 Research Directions

This project opens several promising research directions:

Future Research Opportunities

- Investigating methods to ensure narrative consistency across multiple AI-generated elements
- Exploring techniques for generating and maintaining long-term story arcs through machine learning
- Developing metrics for evaluating the quality and player engagement with AI-generated game content
- Studying player perceptions of and responses to AI-generated versus human-written content
- Investigating the potential for AI to adapt content to individual player preferences and play styles

These research directions could benefit not only game development but also other fields that rely on AI-generated narrative and interactive content.

8 Conclusion

Dungeon Text demonstrates the potential of integrating modern AI capabilities with traditional game design principles to create engaging interactive experiences. By combining the structure and reliability of conventional game systems with the creativity and adaptability of AI-generated content, the project achieves a balance that enhances the player experience while maintaining game coherence.

The implementation successfully addresses several key challenges in AI-driven game development, including asynchronous content generation, fallback mechanisms, and maintaining narrative consistency. The project shows that even with current technological limitations, AI can significantly enhance game experiences by generating contextually appropriate, varied content that would be impractical to pre-write.

As language models continue to advance and become more accessible, approaches like those explored in Dungeon Text will likely become increasingly common in game development, offering new possibilities for interactive storytelling and player-responsive experiences. The lessons learned from this project provide valuable insights for future work not only in game development but in any field that seeks to combine AI-generated content with interactive user experiences.

9 References

1. Google. (2023). Google Generative AI SDK for Python. https://ai.google.dev/tutorials/python_quickstart
2. Short, T. X., & Adams, T. (2019). Procedural storytelling in game design. CRC Press.
3. Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 172-186.
4. Brown, T., et al. (2020). Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 1877-1901.
5. Walton, N. (2019). AI Dungeon: Creating infinitely generated text adventures with deep learning language models. https://medium.com/@nickwalton00/ai-dungeon-creating-infinitely-generat
6. Russell, S., & Norvig, P. (2020). Artificial intelligence: A modern approach (4th ed.). Pearson.
7. Montfort, N. (2003). Twisty little passages: An approach to interactive fiction. MIT Press.
8. Crawford, C. (2012). Chris Crawford on interactive storytelling (2nd ed.). New Riders.
9. Smith, A. M., & Mateas, M. (2011). Answer set programming for procedural content generation: A design space approach. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 187-200.
10. McCoy, J., Treanor, M., Samuel, B., Reed, A. A., Mateas, M., & Wardrip-Fruin, N. (2014). Social story worlds with Comme il Faut. IEEE Transactions on Computational Intelligence and AI in Games, 6(2), 97-112.
11. Evans, R., & Short, E. (2014). Versu—A simulationist storytelling system. IEEE Transactions on Computational Intelligence and AI in Games, 6(2), 113-130.
12. Bartle, R. (2003). Designing virtual worlds. New Riders.
13. Kreminski, M., & Wardrip-Fruin, N. (2019). Generative literature of the past and future. Electronic Book Review.
14. Chaplot, D. S., et al. (2016). Transfer deep reinforcement learning in 3D environments: An empirical study. Deep Reinforcement Learning Workshop, NIPS.
15. Summerville, A., et al. (2018). Procedural content generation via machine learning (PCGML). IEEE Transactions on Games, 10(3), 257-270.

A Sample Game Output

Example of Generated Quest and Dialogue

NEW QUEST
ploration Quest
Quest Giver, the last of the Sunstone Scribes, implores you to recover the Orb of Aethelred, stolen by goblins and hidden deep within the Whispering Caves, for its light is the only thing that can pierce the encroaching Shadowfell. The journey will be perilous, for the goblins guard their prize jealously, and the caves themselves whisper secrets that can shatter the mind.

NPC Dialogue (Goblin):
"You!"
"Meat for pot!"
"Mine!"
"All mine!"
"Grrr!"
"Kill!"

NPC Dialogue (Merchant):
"Looking for something specific, or just browsing?"
"Best prices in town, I assure you."
"Don't touch that, unless you're planning on buying it."

QUEST COMPLETE
The Orb of Aethelred pulses with renewed energy as you return it to the Sunstone Scribe. The ancient keeper nods with solemn gratitude, his weathered hands carefully receiving the artifact. "The balance is restored," he whispers as light begins to push back the encroaching darkness.
Reward: +10 XP, +5 Gold

B Project Timeline

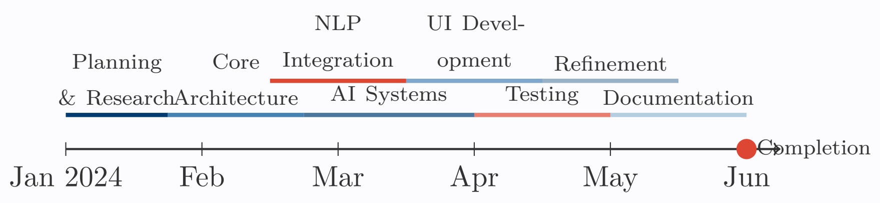

Figure 6. Project Development Timeline