Requirements Specification Document

Team 3: AI Quest

Project Title: AI-Generated Text-Based Game
Date: 02/24/2025
Team Name: AI Quest
Course: CS399 - Generative AI in Software Development
Team Members: Tristen, Avnish, Aidan
Version: 1.0

Acceptance Statement:
Accepted as baseline requirements for the project:

For the team: _____ (Team Lead Signature)
Date: 24th February 2025

Table of Contents:

7. Risks and Limitations

---

1. Introduction

This document defines the requirements for a semester-long project in CS399, where we aim to create an AI-generated text-based game. The goal is to allow users to define certain aspects of the game, such as the setting and difficulty level, and dynamically generate content using a generative AI model. The project will explore the potential of generative AI in interactive storytelling and game mechanics while maintaining a realistic development timeline suitable for three students with other academic commitments.

Text-based games have a long history in the gaming industry, dating back to early interactive fiction and MUDs (Multi-User Dungeons). These games rely heavily on storytelling, player choices, and procedural generation. With modern AI capabilities, it is now possible to enhance these games by dynamically generating game content, creating a more immersive and responsive experience for players.

This project will utilize either an AI API (such as OpenAI's GPT-4 or similar) or a local large language model (LLM) like LLaMA to generate text-based game scenarios. Users will be able to input certain preferences, and the AI will craft a custom game narrative based on those parameters. The game will feature adaptive difficulty and a branching storyline to create a unique experience for each player.

---

2. Problem Statement

Traditional text-based games require extensive manual content creation, which can limit scalability and replayability. Developers must write numerous branching narratives, which is time-consuming and often results in static, predictable gameplay. Additionally, players seeking a unique and personalized experience may find existing games lacking in variety.

The main challenges include:

- **Lack of Dynamic Content:** Pre-written narratives do not adapt to user preferences.
- **Limited Replayability:** Static stories lead to predictable gameplay over multiple sessions.
- **High Development Effort:** Crafting multiple branching paths manually requires significant effort from developers.
- **Player Engagement:** Without AI-generated content, it is difficult to create an experience that feels truly interactive and adaptive.

The proposed project will address these challenges by leveraging generative AI to create game content dynamically, ensuring a more personalized and engaging experience for users.

---

3. Solution Vision

Our solution is a text-based game that uses generative AI to create unique and dynamic storylines based on user-defined preferences. The AI will generate:

- **Thematic Game Worlds:** Players can choose from predefined themes such as fantasy, sci-fi, horror, or historical settings.
- **Difficulty Levels:** AI will adjust the complexity of challenges and puzzles based on user selection.
- **Interactive Storylines:** The game will use AI-generated text to dynamically respond to player choices, creating unique branching narratives.
- **Adaptive Gameplay:** AI will modify the story and challenges based on the player's actions and decisions.

The game will be accessible via a simple web or command-line interface and will be designed to run efficiently within our semester-long timeframe.

---

4. Project Scope and Objectives

Scope

- Develop a functioning prototype of a text-based AI-generated game.
- Implement user input for selecting themes and difficulty levels.
- Integrate a generative AI model for story creation.
- Ensure a reasonable degree of replayability and variation in generated content.
- Design an intuitive interface for interacting with the game.

Objectives

- Deliver a playable game by the end of the semester.
- Ensure AI-generated content remains coherent and engaging.
- Maintain a feasible development scope, given time and resource constraints.
- Prioritize core functionalities over additional features to ensure completion.

---

5. Stakeholders and Users

Primary Users

- **General Players:** Individuals interested in interactive storytelling and text-based games.
- **Game Developers:** Those looking to explore generative AI in game development.
- **Academics & Researchers:** Those interested in studying AI-generated narratives.

Project Stakeholders

- **CS399 Course Instructors:** Evaluating project progress and learning outcomes.
- **Project Team Members:** Responsible for development, documentation, and testing.

---

6. Constraints and Assumptions

Constraints

- **Limited Development Time:** The project must be completed within a single semester.
- **Team Size:** Three students with existing coursework obligations.
- **Technology Selection:** AI integration must be feasible with available APIs or local LLM models.
- **Resource Availability:** Hardware constraints for running local AI models.
- **Realism:** Game scope should remain achievable within time constraints.

Assumptions

- Users will have basic familiarity with text-based games.
- A working internet connection is available for AI API usage.
- The AI model can generate coherent and engaging text within acceptable latency.
- Development tools and libraries are accessible to all team members.

---

7. Risks and Limitations

Potential Risks

- **AI Model Limitations:** The model may generate incoherent or repetitive text, reducing game quality.
- **Implementation Challenges:** Integrating generative AI with a text-based interface may introduce technical complexities.
- **User Engagement Issues:** The generated narratives might lack depth compared to hand-written content.
- **Scalability:** Limited control over AI-generated content may make debugging difficult.
- **Ethical Considerations:** AI-generated content should avoid inappropriate or offensive material.

Risk Mitigation Strategies

- **Content Filtering:** Implement basic content moderation to ensure AI-generated text remains appropriate.

- **Pre-Defined Templates:** Use structured templates to guide AI-generated narratives for coherence.
- **Testing & Feedback:** Regular playtesting sessions to refine generated content.
- **Scoped Development:** Prioritize core features to ensure a functional MVP (Minimum Viable Product).

Conclusion

This requirements document outlines the foundational aspects of our AI-generated text-based game project for CS399. The game aims to leverage generative AI to provide users with personalized interactive storytelling experiences. With a clear scope, defined objectives, and an awareness of potential risks, our team is committed to developing a playable and engaging prototype within the semester's constraints. Our focus remains on delivering a practical and functional implementation that demonstrates the capabilities of generative AI in gaming while keeping our workload realistic.

Functional Requirements

This section details the functional requirements for the AI-Generated Text-Based Game project. The following requirements have been developed with the constraints of a three-student team and a limited semester timeline in mind. They provide a structured hierarchy of high-level and low-level functionalities that the final system must implement. The goal is to outline realistic, achievable features that demonstrate the power of generative AI in interactive storytelling, while remaining within our capacity and time constraints.

---

1. System Overview

The system is designed to deliver a text-based interactive game in which players define key parameters—such as theme, setting, and difficulty level—and receive dynamically generated narratives based on their choices. The core functionality is powered by either an external generative AI API (e.g., GPT-4) or a local large language model (LLM) such as LLaMA. The game is intended to be accessible through either a command-line

interface or a simple web interface. Overall, the system is composed of several functional modules that collaborate to offer a seamless and engaging user experience.

---

2. System Initialization and Configuration

2.1. Startup Routine

- **Boot-Up Process:**
  On startup, the system initializes necessary modules, including the user interface, configuration manager, and AI integration module. This process will verify that all dependencies are correctly loaded and that any required connections (e.g., to an AI API or local LLM) are successfully established.
- **Configuration Loading:**
  The application will load default configurations from a file or embedded settings. These configurations include default themes, difficulty settings, API keys or paths to local model files, and system parameters such as timeouts and error handling protocols.
- **Fallback and Recovery:**
  In case the AI service is unavailable or misconfigured, the system will revert to a fallback mode that offers a limited static narrative, ensuring that the game remains playable even when dynamic content generation is interrupted.

2.2. Environment and Dependency Checks

- **Internet and Local Resource Verification:**
  For API-based operations, the system verifies that a stable internet connection is available. For local LLM usage, the system checks for sufficient computational resources (e.g., memory, processing power) to run the model.
- **User Role Initialization:**
  Although the game is intended for general players, the system may include a hidden administrative mode for debugging and testing, ensuring that detailed logs and diagnostic data are available to assist in troubleshooting.

---

3. User Interface (UI) and User Experience (UX)

3.1. Main Menu and Navigation

- **Welcome Screen:**
  The game launches with a welcome screen that introduces the game's premise and displays basic instructions. The UI design will be minimalistic yet informative, reflecting the text-based nature of the game.
- **Navigation Options:**
  The main menu will include options such as "Start Game," "Configure Game Settings," "View Help/Instructions," and "Exit." Navigation is achieved via keyboard input (for CLI) or clickable elements (if a web interface is used).

3.2. Game Configuration Interface

- **Theme Selection:**
  Users will be presented with a list of predefined themes (e.g., Fantasy, Sci-Fi, Horror, Historical) from which to choose. Each theme selection will guide the AI in generating a setting-specific narrative. A brief description accompanies each theme to help the user decide.
- **Difficulty Selection:**
  Users can choose among preset difficulty levels (Easy, Medium, Hard). The selected difficulty will influence the complexity of challenges, puzzles, and the narrative's tone. This parameter allows the AI to tailor content—such as the intricacy of puzzles or the ambiguity of challenges—to match player preferences.
- **Additional Customization Options:**
  The system will allow users to optionally input additional preferences, such as narrative style or pacing. These inputs serve as supplementary hints for the AI model without significantly complicating the UI.

3.3. In-Game Display and Interaction

- **Dynamic Text Display:**
  The core game area is dedicated to displaying AI-generated text. The narrative updates in real time based on user choices, with clear demarcation of different story segments to help maintain continuity.
- **Input and Command Processing:**
  A dedicated input area will capture player responses or commands. The interface is designed to validate user entries in real time, ensuring that input adheres to expected formats (e.g., commands, yes/no answers, menu selections).
- **Feedback Mechanisms:**
  The interface will provide immediate feedback on invalid inputs or errors, guiding users to correct mistakes. Additionally, occasional tips and clarifications may appear to help new players acclimate to the text-based game environment.

---

4. Game Configuration and Setup

4.1. Theme and Setting Configuration

- **Predefined Themes:**
  The system will offer several curated themes that serve as high-level directives for narrative generation. Each theme will have associated keywords and narrative templates that the AI leverages to maintain consistency.
- **Customization Parameters:**
  Users can specify certain aspects of the game world, such as the era (modern, medieval, futuristic), the type of environment (urban, wilderness, fantastical realms), and any specific narrative elements they would like to see included. These inputs are parsed and sent as parameters to the generative model.

4.2. Difficulty and Challenge Customization

- **Difficulty Settings:**
  The game will support three primary difficulty settings. These settings adjust:
  - The complexity and number of puzzles.
  - The frequency and ambiguity of choices.
  - The likelihood of encountering unexpected narrative twists.
- **Adaptive Gameplay:**
  The AI module will consider user-selected difficulty and real-time gameplay data (e.g., time taken to respond, frequency of decision changes) to adapt the narrative complexity and pacing dynamically.

4.3. Initial Game Setup

- **Player Profile Initialization:**
  On starting a new game, the system creates a temporary profile that includes the selected configurations. This profile tracks the player's choices, narrative progress, and any session-specific variables required for maintaining story coherence.
- **Session Parameters:**
  The initial configuration also sets session-level parameters, such as the starting narrative context, initial challenges, and any randomized elements that will contribute to replayability.

---

5. AI Integration and Narrative Generation

5.1. Integration with Generative AI

- **API/Model Initialization:**
  The system will include an abstraction layer to interface with either an external AI API or a locally hosted LLM. This layer handles authentication, model initialization, and parameter configuration, ensuring that the generative process is modular and adaptable.
- **Query Construction:**
  Based on user inputs and game state, the system constructs queries for the AI. These queries include details such as the selected theme, difficulty level, current narrative context, and any recent player choices.
- **Response Parsing and Validation:**
  Upon receiving the AI-generated text, the system validates the output for coherence and relevance. If the generated text is incomplete or incoherent, a fallback routine either reissues the query or provides a default narrative fragment.

5.2. Dynamic Narrative Generation

- **Initial Narrative Generation:**
  At game startup, the AI generates an introductory narrative that sets the scene according to the user's theme and difficulty preferences. This narrative establishes the initial context and hints at potential story arcs.
- **Branching Storylines:**
  As the game progresses, the AI will generate multiple narrative branches based on player decisions. Each decision point is communicated as a prompt to the AI, which then produces corresponding narrative segments that maintain logical continuity with previous events.
- **Adaptive Story Progression:**
  The narrative generation process is iterative. After each major decision, the system updates the game state and passes the updated context back to the AI. This iterative process ensures that the game narrative remains dynamic and responsive to player choices, while also incorporating adaptive difficulty elements.
- **Content Randomization:**
  To enhance replayability, the system incorporates controlled randomization in the narrative generation process. This randomness can affect minor story details, dialogue options, or the order of events, ensuring that no two playthroughs are identical.

---

6. Game Progression and Branching Narrative

6.1. Story Flow Management

- **Narrative Segmentation:**
  The game narrative is divided into discrete segments or “chapters.” Each segment concludes with a set of player choices that determine the direction of the story. The segmentation approach facilitates both narrative coherence and ease of debugging.
- **Decision Points:**
  At key junctures, the system presents the player with a limited set of choices. Each choice is associated with specific parameters that are relayed to the AI for the subsequent narrative generation. This controlled branching ensures that the narrative remains manageable and logically consistent.

6.2. Context Maintenance and State Tracking

- **Game State Persistence:**
  The system maintains a persistent record of the game state, which includes the current narrative context, the choices made by the player, and any variables that influence subsequent story generation. This record is essential for ensuring that the narrative remains coherent across branching storylines.
- **Contextual Memory:**
  A lightweight contextual memory mechanism is employed to track recent narrative events. This memory is periodically summarized and passed to the AI, providing the necessary context for generating subsequent text without overwhelming the model with the entire game history.

6.3. Looping and Replayability

- **Replay Mechanisms:**
  Upon completing a narrative arc, players are given the option to restart or branch off from previous decision points. The system supports multiple playthroughs with varying narratives, emphasizing the game’s replayability.
- **Progressive Difficulty Adjustment:**
  As players advance, the game monitors performance and decision patterns to subtly adjust narrative complexity. This adaptive mechanism ensures that the game remains challenging and engaging, regardless of the initial difficulty setting.

---

7. User Interaction and Input Processing

7.1. Input Handling

- **Command Parsing:**
  The system is designed to handle various types of user inputs—whether free-form text commands or predefined selections from a menu. An input parser validates and categorizes these inputs, ensuring that they match expected patterns.
- **Error Correction:**
  When the system encounters an unrecognized or ambiguous input, it will prompt the user for clarification. For example, if a player enters an unexpected command, the system provides a list of acceptable commands or suggests re-entering the input.

7.2. Decision-Making Interface

- **Choice Prompts:**
  At each decision point, the interface clearly outlines available options. The design prioritizes clarity by numbering options or using clearly distinguishable labels, ensuring that players can easily comprehend and select their desired path.
- **Feedback Loops:**
  The game immediately acknowledges each valid input, confirming the player's choice and indicating that the system is processing the next narrative segment. This immediate feedback is vital for maintaining user engagement and ensuring that players feel in control of the narrative.
- **User Guidance:**
  Help commands and tooltips are integrated throughout the interface. Should a player require assistance, they can invoke these guides to understand gameplay mechanics, input formats, or narrative implications of their decisions.

---

8. Error Handling, Logging, and Recovery

8.1. Robust Error Handling

- **Input Validation Errors:**
  The system must detect and gracefully handle invalid inputs by prompting the user with corrective instructions. This includes detecting typos, unrecognized commands, or contextually inappropriate choices.
- **System Exceptions:**
  Any runtime errors or exceptions (e.g., API timeouts, connection failures) will be caught by the system's error-handling routines. The player is notified of the issue in simple, non-technical language while the system logs detailed error information for later debugging.

---

9. Optional and Future Enhancements

Given the tight schedule and resource constraints, the following optional features are identified as stretch goals. These features are not guaranteed for the initial prototype but provide a roadmap for potential enhancements if time permits:

9.1. Save and Load Game Functionality

- **Game Session Persistence:**
  Allow players to save their current game state and resume later. This includes storing the narrative context, decision history, and any custom configurations.
- **User-Friendly Interface:**
  Implement intuitive save and load commands within the interface, ensuring that players can manage their sessions without confusion.

9.2. Custom Scenario Creation

- **Advanced Configuration Options:**
  For users who desire a more tailored experience, include an advanced mode where they can define custom narrative parameters or provide a brief storyline prompt that the AI will expand upon.
- **User-Defined Templates:**
  Provide a framework for players to upload or select narrative templates. The AI would then incorporate these templates into its generative process, offering a semi-customized narrative experience.

9.3. Multi-Session Narrative Continuity

- **Extended Story Arcs:**
  Explore the possibility of maintaining narrative continuity across multiple sessions. While this feature is ambitious given the timeline, a modular design will allow for its incorporation in later iterations.
- **Contextual Merging:**
  Develop algorithms to merge narrative context from previous sessions with current inputs, enabling a coherent and extended storytelling experience.

---

10. Summary and Development Considerations

10.1. Development Roadmap

- **Prioritization:**
  The primary focus is on delivering a functional prototype that covers core functionalities: user configuration, AI-powered narrative generation, real-time interaction, and error handling. Optional features serve as enhancements if time and resources permit.
- **Incremental Development:**
  The project will follow an iterative development cycle, with each iteration focusing on a distinct functional module. Regular playtesting and feedback loops will guide the refinement of narrative generation and user interaction features.

10.2. Realism and Scope

- **Feasibility:**
  The functional requirements have been scoped to ensure that a working prototype can be delivered within the semester, considering the availability of three team members. Every requirement has been vetted for feasibility, and the system is designed to gracefully handle limitations (e.g., fallback modes for AI failures).
- **Team Collaboration:**
  Each functional module is modular and can be developed concurrently by different team members. Clear interfaces and well-documented code will facilitate integration and allow for parallel progress across UI design, AI integration, and game state management.
- **Evaluation and Acceptance:**
  The project will be evaluated based on the completeness and stability of the core functionalities outlined above. The acceptance of these functional requirements

serves as the baseline for subsequent development, testing, and refinement of the prototype.

By adhering to the functional requirements described in this document, our team commits to delivering a text-based game that is both engaging and technically sound. The project leverages generative AI to create dynamic narratives, adapts to user input in real time, and ensures a balanced workload for a small, busy team. The detailed breakdown of functions—ranging from system initialization to narrative branching and error recovery—lays a solid foundation for a successful semester-long project in CS399, demonstrating both the potential and the practical challenges of using AI in software development.

Performance (Non-Functional) Requirements

This section defines measurable performance benchmarks that the AI-generated text-based game must meet. The criteria outlined below cover response times, accuracy, usability, and system resource usage. These requirements are based on assumed hardware and software environments typical of a college setting, ensuring that our project remains realistic and verifiable within our semester timeline.

---

1. Response Time

1.1. AI-Generated Content

- **Initial Narrative Generation:**
  When the game is launched, the AI should generate and display the introductory narrative within **15 seconds** under a standard broadband connection (approx. 25 Mbps) or on local hardware configured with at least 32GB RAM and a dedicated GPU for LLM operations.
- **Dynamic Narrative Updates:**
  For subsequent story segments triggered by user choices, the generative AI should produce text within **3 to 5 seconds**. This limit ensures that gameplay remains smooth and engaging, preventing noticeable delays during interaction.

1.2. User Interface (UI) Responsiveness

- **Input Acknowledgment:**
  All user inputs—whether keyboard commands in a CLI or clicks in a web interface—should be acknowledged by the system within **0.5 seconds**. If further processing (such as AI query generation) is needed, a progress indicator must be displayed to inform the user.
- **Error Recovery:**
  In the event of an error (e.g., API timeout or invalid input), the system will resolve the issue and provide appropriate feedback within **15 seconds**.

---

2. Accuracy and Coherence of AI Output

- **Coherence Benchmark:**
  At least **90%** of AI-generated narrative segments must be contextually coherent with preceding game content. This target will be verified via user testing and surveys.

---

3. Usability and Learnability

- **Novice User Training:**
  New users should be able to grasp basic game functionalities—such as theme selection and command input—within **5 minutes** using the in-built help and tutorial system. This expectation will be confirmed through structured user testing sessions.
- **Advanced User Efficiency:**
  Users familiar with the game should experience an average command execution time of less than **1 second** after initial setup, ensuring a fluid and responsive interaction.

---

4. System Throughput and Scalability

- **Resource Utilization:**
  The application must not exceed **70%** of available CPU and memory resources

on our assumed development hardware during peak usage. This target helps maintain overall system responsiveness and stability.

---

5. Assumptions on Hardware and Software Environment

- **Hardware:**
  - For API-based operations: A standard college laptop/desktop with **at least 16GB RAM** is assumed.
  - For local LLM operations: A system with **32GB+ RAM** and a dedicated GPU is expected.
- **Network:**
  - AI API operations assume a stable broadband connection (approximately **25 Mbps** or better).
  - Offline mode performance assumes optimized configurations for local model execution.
- **Software:**
  - The system will utilize lightweight, asynchronous programming languages and frameworks to ensure that performance benchmarks are met.

---

6. Verification and Testing

- **User Testing:**
  Scheduled playtesting sessions will assess the usability and learnability of the game, verifying that novice users can learn the system within the set time frame and that advanced users maintain efficient interactions.

By adhering to these performance requirements, our team commits to delivering an interactive text-based game that is responsive, accurate, and user-friendly—ensuring a quality experience even within the limitations of our semester timeline and available resources.

Environmental Requirements

1. **Platform Constraints:** The game must be playable on **Windows, macOS, and Linux**, with priority given to a desktop-based experience over mobile.

2. **AI Model Accessibility:**
    - If using an online API (e.g., OpenAI, Cohere), the system must integrate with a **well-documented API** that supports text generation.
    - If using a local model (e.g., LLaMA), the game must support **model downloading and local inference** without excessive setup.

3. **Internet Dependence:**
    - The system should function in an **offline mode** if using a local model.
    - If relying on an online API, a clear error message should appear when no internet connection is available.
4. **Programming Language & Tools:** The programming language and framework will be chosen based on feasibility, with considerations for **Python, JavaScript, or Kotlin**.

5. **Storage & Data Handling**: User preferences (e.g., game theme, difficulty) will be stored **locally in JSON files** or in a lightweight database (e.g., SQLite) to avoid cloud storage dependencies.

6. **Security Considerations:** The system should not collect or store any personally identifiable information (PII) and should adhere to standard **data protection practices**.

Potential Risks

1. **AI Output Quality Issues:**

    - Risk: The AI may generate incoherent or nonsensical text, leading to poor user experience.

    - Mitigation: Implement a **post-processing filter** to verify content coherence and offer regeneration if output quality is low.
2. **Performance Bottlenecks:**
    - Risk: The AI may take too long to generate responses, reducing gameplay fluidity.
    - Mitigation: Optimize requests by using **smaller models** or implementing **response caching** for repeated queries.
3. **Technical Integration Challenges:**
    - Risk: Difficulty in integrating the AI model/API with the game's logic.
    - Mitigation: Choose a well-documented API or framework, and allocate **at least two weeks** for integration testing.
4. **Scope Creep:**
    - Risk: Adding too many features could extend development beyond available time.
    - Mitigation: Stick to core functionality (game generation and basic interaction) and postpone extra features.
5. **Team Coordination Issues:**
    - Risk: Given the late start, collaboration challenges may arise.
    - Mitigation: Use a project management tool (e.g., Trello, GitHub Projects) to track progress.

Project Plan

The project will follow a **lean, milestone-based approach**, with the following key milestones:

1. **Week 7-8: Research & Setup**
    - Finalize technology stack (API vs. local LLM)
    - Set up the development environment and repository
2. **Week 9-10: Core Game Development**
    - Implement game state management
    - Integrate AI text generation
3. **Week 11: Basic UI & Interaction Implementation**
    - Develop the user input system for specifying game settings

    - Test AI response handling
4. **Week 12: Refinement & Optimization**
    - Improve response speed and UI elements
    - Ensure offline/local model functionality (if applicable)
5. **Week 13-14: Testing & Final Adjustments**
    - Conduct **end-to-end user testing**
    - Fix major bugs and polish UI
6. **Week 15: Documentation & Submission**
    - Compile final project documentation
    - Prepare for submission/presentation

Conclusion

This document outlines the key specifications for our generative AI-driven text-based game, focusing on performance expectations, environmental constraints, potential risks, and a structured project plan. Given our limited timeline and team availability, we have prioritized **core functionality and feasibility** over feature expansion. Our approach ensures a **playable, engaging experience** while keeping technical complexity within reasonable limits.

The next steps involve finalizing the AI integration approach and starting implementation, ensuring that we stay on track with our project milestones. By maintaining clear communication and structured progress tracking, we aim to successfully deliver a functional prototype by the semester's end.

Works Cited

Introduction, Problem Statement, Solution Vision, Project Scope, Stakeholders, Constraints, Risks: Chat log

Functional & Non-Functional Requirements: Chat Log
Logo: Chat Log

Environmental Requirements Project Plan, Conclusion: Chat Log