School of Computing  
CA326 Year 3 Project Proposal Form

**SECTION A**

Project Title: Dungeons And Crawlers

Student 1 Name: Caolan Cochrane ID Number : 22490802

Student 2 Name Ethan Doyle ID Number \_\_\_\_\_\_\_\_\_\_\_

*(A third team member is exceptional and requires detailed justification.)*

Staff Member Consulted: Mark Humphrys

**Note:** We had written a project proposal not to this spec, there was a lot of detail so I decided to add it all here. But the project description is split into necessary and extra context so it is not necessary to read this entire document. One section is pasted to the start just for consistency in reading. Thanks!

**Division of Work** - Outlines how the work is envisaged to be split equally among the team members.

50 – 50: Design and Programming shall be done in as equal a measure as possible. Caolan has more technical experience, but Ethan is a quick learner. We will play to our strengths and have discussions over implementation and divide work accordingly. Work effort will be communicated honestly.

1. **Programming language(s)** - Javascript
2. **Programming tool(s)** - WebSockets, Three.js library, Falcon LLM (undecided LLM or model, currently a placeholder), yaml for LLM interaction.
3. **Learning Challenges** - Replication and client-side prediction → How do we decide if the player has gone out of bounds, when is rollback criteria met, at what frequency are update batches sent to the server.
4. **Hardware / software platform** - Linux backend, express js routing and API management
5. **Special hardware / software requirements** - Describe any special requirements. → Dedicated Server Architecture. We can host, but it should also be hostable on the school of computing. No special requreiments.

Project Description (1-2 pages): To fit for two pages, I moved some of the info from the original document up, and I’ve split this document into two parts: The original project description and then an expanded one that has more context.

# **Project Proposal: AI-Driven, Procedurally Generated Multiplayer Game**

## **Overview**

Our project is focused on creating a procedurally generated, multiplayer web-based game driven by AI-generated storytelling and dialogue. The end goal is to allow players to define the style and theme of the game world with a prompt like, “A Roman Empire gladiator game, underwater, with vampires.” From this prompt (or eventually, other parameters), the AI would generate relevant story elements, dialogue, and possibly even environments and quests that match the theme.

Each game world will be generated using a unique seed, providing consistent replayability. In the long term, we aim to tie the world generation to both the player’s prompt and the seed, making the experience even more personalised. However, for the initial implementation, our focus is on building a strong foundation of AI-driven dialogue and procedural systems that we can expand over time.

**What are we solving?**

Procedurally generated games, thus far and for the most part, have remained fairly coupled to their procedural nature. It’s difficult to build on a story, and have in-depth immersive experiences and memorable moments that are directly outside of the players control. The enjoyment stems mostly from the environment, and how you interact with it. But that environment doesn’t focus on stories or role-playing.

We are enhancing replayability, but also giving the player the reigns to allow them to go as far as they please. We’re focusing on building from a good foundation, and building a framework to integrate LLM capabilities into a block game. This could, if expanded upon enough, lead to a world that lives and breathes, where npc’s have alignments and political views and character traits and communities can fall and thrive depending on major events. We could have news, trade-lines and economies between communities. This is all very far in the future, and we will establish the foundations first. But afterwards the possibilities are really exciting.

## **Expected Outcomes and Long-Term Vision**

The final product will be a flexible, AI-driven game that lets players shape their world based on their creativity and a unique seed. With AI-driven dialogue and procedural generation, we aim to offer a replayable experience, where players can return to the same world with the same prompt and seed, but discover new angles in each session.

By the end of the project, we plan to deliver:

**AI-Generated, Replayable Storytelling**: A consistent, deterministic dialogue and quest system tied to a unique prompt and seed.

We think that whether the AI’s output should be deterministic could possibly be a player-decided option. This allows us to make everyone happy, allowing the players to go as far as they want, without us needing to take the responsibility of ensuring it works well. We would have a suggested experience that is configured optimally, but allow players to explore any and all possibilities.

**Real-Time Multiplayer**: A stable, optimised multiplayer experience that scales smoothly with the number of players.

We want the multiplayer option to be as seamless for the user as possible. Auth is something that we haven’t really considered. We want to quickly recover from disconnects for player experience. We’re thinking cookies signed server-side with a TTL of maybe an hour could be a good way to go about this, but we definitely aren’t anywhere near sure as to how we will implement this. It could wind up being as simple as a room code per session, but I doubt this.

**Expandable and Modifiable Systems**: A modular game foundation that can accommodate new AI features, modding capabilities, and complex game elements.

We want to expose a good modding API, mainly for our own use for now. If we design our APIs and frameworks for building parts of the game we can standardise it for both ourselves and anyone who might want to contribute. This means systems like world generation, items, blocks, quests etc.

Our approach is to start with a solid foundation in procedural generation and AI storytelling, making it robust enough to expand on in the future. Ultimately, we’re building a game world that can grow in complexity and depth over time, with a focus on giving players a unique, replayable experience.

Of course the reality is, now that we’re implementing dynamic content generation into the project, the possibilities really can be endless. With a simple way to visualise the world as a 3d array on top of using LLM's in interesting ways we can have whole communities interacting in the game world, with migration systems, weather, farming, news and other systems. What we’re focusing on is creating a base from which we can build these systems into the game world. The possibilities are endless so long as we implement everything carefully.

**THIS IS THE END OF THE PROJECT DESCRIPTION → ANYTHING FURTHER IS EXTRA CONTEXT**

## **Key Objectives**

**AI-Driven Storytelling and Dialogue Generation**

**Deterministic Dialogue**: Our primary use for AI will be generating dialogue and quest narratives that adapt to the player’s chosen theme. By anchoring dialogue to both the seed and prompt, we aim to deliver a coherent experience that can be replayed and feel fresh.

**Experimenting with Local AI Models**: We’ll explore different local AI models, such as Falcon7B, to identify one that efficiently generates relevant, engaging content without significant performance overhead.

**Procedural World Generation and Replayability**

**Seed-Based World Building**: The world will be generated using a unique seed, with the possibility of adding prompt-based elements over time. The goal is for players to have a consistent, replayable experience, where the same prompt and seed will produce the same world layout and story framework.

**Modular System for World Expansion**: We’ll establish a flexible base for procedural landscapes that allows us to add future features like AI-generated buildings, political systems, races, or puzzles. For now, we’ll focus on a scalable foundation that could accommodate these systems later.

**Multi-User Real-Time Gameplay with WebSockets**

**Real-Time Interaction**: We’ll use WebSockets to enable real-time multiplayer gameplay, allowing players to interact within the AI-generated world.

**Optimised Network Management**: To keep the experience smooth, we’ll implement differential updates and client-side prediction, ensuring efficient data flow even as more players join.

**Persistent Sessions with Cookies**

**Session Continuity**: Cookies will store player preferences and world states, making it easy to resume previous sessions. While cookies will be used mainly for convenience, we’ll manage any associated security considerations if we end up hosting on a custom server.

**Run-Time Validation for AI-Generated Code**

**UI-Driven Content Customization**: Players will have options to regenerate specific AI-generated content (like dialogue) if it doesn’t meet their expectations, adding a level of customization.

**Code Handling and Error Management**: To maintain stability, we’ll validate and manage any code generated by the AI in real-time, ensuring it integrates smoothly with the game.

## **Project Stages**

**Stage 1: Minimum Viable Product (MVP)**

**Procedural World and Multiplayer Setup**: Build a basic procedural world using Perlin noise and Three.js for rendering, and set up WebSocket-based multiplayer.

**Initial AI Prompt System**: Allow players to input open-ended prompts, which the AI will interpret to create initial world elements and themes.

**Basic UI and Interactivity**: Set up UI elements to control world aspects and allow players to interact with regenerable content.

**Stage 2: Large Language Model (LLM) Integration**

**Implement LLM for Dialogue and Quests**: Research and integrate a locally installed LLM (e.g., Falcon7B) that can generate storyline dialogue and quests based on the user prompt and seed. We can tie quest objectives to procedurally generated structures, add lore, drama, consequences.

**Early Experiments in World Expansion**: Begin exploring the possibility of using AI to generate elements like structures or basic political systems and assess the model’s overhead.

**Stage 3: Optimization and System Enhancements**

**Network Optimization**: Focus on efficient data transmission and client-side prediction to minimize latency and enhance multiplayer scalability.

**Server-Side Optimization**: Improve server-side memory usage and data handling to prepare for increased player count.

**Stage 4: Expanding Core Systems and Advanced Features**

**Enhanced World-Building**: Integrate features like economy, tribes, and other procedural elements for deeper interaction within the game world.

**Modding API for Customization**: Introduce a modding API that allows players to expand or modify game elements, fostering a more personalised experience.

**Dynamic Game Features**: Add adaptive difficulty, special events, or seasonal changes that respond to the player’s actions and game state.

**Tec** **hnologies**

**Entity-Component System (ECS) with Ecsy for Modularity**  
  - We’re using Ecsy to build the game in a modular way. By breaking down each part of the game (like movement, quests, and interactions) into components, we can add or change features without affecting the entire game. This approach keeps the game structure flexible and easy to update.

**Basic Quest System and State Replication**  
  - The server will use differential state replication to minimise data transfer, sending only changes instead of complete updates. This keeps the game responsive and allows players to interact in real time. Our quest system will start with simple tasks like item retrieval but will be designed to support more complex objectives in the future. Client-side prediction and server reconciliation will also be applied to reduce lag.

* Client-side prediction is incredibly important to player experience. It’s one of the most important systems that we will have to implement on the frontend.
* Other choices like how we serialise/transfer updates are more practical and quick -> Don’t do custom serialisation, but use something more efficient than json i.e binary data
* Then there are optimizations that we can consider as we are beginning to scale up the technical aspects of the game and want to optimise performance.

**Improvements over Similar Games and Challenges Addressed**

Our project addresses some common issues found in games like Minecraft, while also adding unique features:  
  
**1. Modular Architecture with ECS for Scalability**  
  - Minecraft’s codebase doesn’t use an ECS framework, which can make expanding it challenging without impacting performance. By using Ecsy, we’re building the game in a modular way, where each component operates independently. This makes it easy to add new features or update systems without slowing down the game, which is key as the project grows.  
 **2. Story-Driven Quests with Deterministic AI**  
  - Unlike Minecraft, which lacks story-based content, our game will use an LLM to create consistent dialogue and narrative quests. By connecting the LLM output to the world’s seed, each game session will have a unique storyline tied to the environment, making the game more engaging and replayable.  
  
**3. Streamlined Networking and Data Management**  
  - Our use of differential updates and client-side prediction reduces the amount of data being sent, making multiplayer interactions smoother. This approach avoids network lag and keeps the game responsive even as more players join. Minecraft servers often face issues with memory load and network lag as more entities are added, which our approach helps prevent.

## **Expected Outcomes and Long-Term Vision**

The final product will be a flexible, AI-driven game that lets players shape their world based on their creativity and a unique seed. With AI-driven dialogue and procedural generation, we aim to offer a replayable experience, where players can return to the same world with the same prompt and seed, but discover new angles in each session.

By the end of the project, we plan to deliver:

**AI-Generated, Replayable Storytelling**: A consistent, deterministic dialogue and quest system tied to a unique prompt and seed.

We think that whether the AI’s output should be deterministic could possibly be a player-decided option. This allows us to make everyone happy, allowing the players to go as far as they want, without us needing to take the responsibility of ensuring it works well. We would have a suggested experience that is configured optimally, but allow players to explore any and all possibilities.

**Real-Time Multiplayer**: A stable, optimised multiplayer experience that scales smoothly with the number of players.

We want the multiplayer option to be as seamless for the user as possible. Auth is something that we haven’t really considered. We want to quickly recover from disconnects for player experience. We’re thinking cookies signed server-side with a TTL of maybe an hour could be a good way to go about this, but we definitely aren’t anywhere near sure as to how we will implement this. It could wind up being as simple as a room code per session, but I doubt this.

**Expandable and Modifiable Systems**: A modular game foundation that can accommodate new AI features, modding capabilities, and complex game elements.

We want to expose a good modding API, mainly for our own use for now. If we design our APIs and frameworks for building parts of the game we can standardise it for both ourselves and anyone who might want to contribute. This means systems like world generation, items, blocks, quests etc.

Our approach is to start with a solid foundation in procedural generation and AI storytelling, making it robust enough to expand on in the future. Ultimately, we’re building a game world that can grow in complexity and depth over time, with a focus on giving players a unique, replayable experience.

Of course the reality is, now that we’re implementing dynamic content generation into the project, the possibilities really can be endless. With a simple way to visualise the world as a 3d array on top of using LLM's in interesting ways we can have whole communities interacting in the game world, with migration systems, weather, farming, news and other systems. What we’re focusing on is creating a base from which we can build these systems into the game world. The possibilities are endless so long as we implement everything carefully.