Our project seeks to use artificial intelligence to generate interactive narratives that contain a plot twist. These narratives will be displayed to the user using the sandbox game system, Camelot. This document seeks to describe our proposed architecture as we move forward with this project.

We are planning to use a 4-tier architecture, and the sections are AI, prompt generator, translator, and Camelot.

AI
Prompt Generator
Translation
Camelot

Each layer will only be able to communicate with the layers directly above and below it, and interactions with the user and content from the AI will be passed back and forth through the two middle layers. The prompt generator will send seeds and user input to the AI, which returns text output that will match the narrative we are telling. This text output is then sent to the translator, which converts it to instructions that Camelot can understand. When the player takes action based on the new Camelot content, these actions will be converted back to language that the prompt generator can use by the translation level. The prompt generator then sends these prompts to the AI and the process repeats.

For the AI layer, we are currently planning to use GPT. Having a strong AI base will give us the best output. We hope to fine tune an available model to make its output more compatible with our project. Ideally, the output will be very specific and take small steps in the story rather than taking large jumps ahead, to keep better pacing for the user. If fine tuning the model does not help improve the output for our purposes, we might need to find or create a new AI model.

The prompt generator is where we will create novelty for this project. Its purpose is the direct the AI output based on user input. We want to consistently generate narratives that have plot twists within them. Creating consistent, usable prompts that guide the narrative into a plot twist event is the goal of this part of the project.

The intricacy of the translation layer is going to depend heavily on the AI output that we are able to achieve. Using GPT as is results in text-based narrative, written in plain English. Obviously, the Camelot software cannot use English as commands, and the prompt generator will most likely not be able to make good prompts off of Camelot commands. The translator is what will convert commands to English and English to commands. This process will involve some amount of NLP, which none of our group members are very confident with. One of the goals in fine-tuning the AI and of the prompt generator is to make the job of translation as easy as possible. An example of this is using the prompt generator to limit AI output to situations that are compatible with Camelot. The most basic form of translation is to take AI output and simply display it as is using in-game narration. While we might use this as a catch-all in case anything comes along that is unable to be handled by our translator, our goal will be to use this method as sparingly as possible.

The Camelot layer is the one that needs the least amount of work from us. A lot of last semester was spent familiarizing our group members with its features and limitations, and the translation layer outputting directly to Camelot commands is what allows this layer to work. This layer is what interacts with the user, and passes user input back up the chain. Things such as character generation will probably be done by randomizing attributes not mentioned by the AI output, and the prompt generator will limit locations and objects the AI uses to those that are present in Camelot.