**Part 1: Initial Project Ideas**

1. **Project Idea 1**: Adventure Game
   * **Description**: Create a text-based game where the player navigates rooms or locations, encounters enemies, and makes choices. Each choice leads to new scenarios or consequences. The objective is typically to survive or achieve a goal (e.g., find treasure, or defeat a boss).
   * **Rule-Based Approach**: Gives the player choices and based on the choice text will be displayed giving them another choice and using rules will determine the consequences. Like if the player encounters an enemy and they choose to fight if they have a weapon they will defeat the enemy and if they do not they will take damage.
2. **Project Idea 2**: Recommendation System
   * **Description**: A rule-based recommendation system suggests items (movies, books, products, etc.) based on predefined criteria or user profiles. Unlike machine learning recommender systems, these rules are manually crafted.
   * **Rule-Based Approach**: The user defines their preferences, such as genre or demographics. For each recommendation, there will be conditions that have to be met and partial matches will have a lower score than full matches. It will filter out items that don't meet the minimum score and share the rest.
3. **Project Idea 3**: Trivia Quiz System
   * **Description**: A trivia quiz system asks questions, checks answers, and manages scoring based on correct or incorrect responses. You can add difficulty levels and a rule-based structure for how questions are chosen or how hints are given.
   * **Rule-Based Approach**: This uses rules to check players' answers and award points if the answer is correct. It would also deduct points if the answer is incorrect.

**Chosen Idea**:   Adventure Game

**Justification**: I chose this project because it seems fun to create and interact with. I get to manually craft the story and what the player goes through.

**Part 2: Rules/Logic for the Chosen System**

The Adventure Game system will follow these rules:

1. **Look or Inspect Rules**:
   * **If** the user types look, show the room description and items in the room.
2. **Item Interaction Rules**:
   * Command: take <item> or pick up <item>
     1. **If** yes, remove it from rooms[current\_room]["items"] and add to player\_state["inventory"].
     2. **If** no, respond “You can’t find <item> here.”
   * Command: use <item> [on <target>
     1. Check if <item> is in the player’s inventory.
     2. **If** <target> is specified, check if it exists in the room or is relevant. For example, use the key on the door.
        1. **If** the rules say “key opens the door,” then unlock the door or remove a barrier.
        2. **If** the rules say “potion heals 5 HP,” then increase the player’s health by 5.
     3. **If** no matching rule is found, respond “You can’t use <item> here.”
3. **Combat/Enemy Encounter Rules**:
   * Check **if** the player enters a fight.
     1. Execute the rule for that event: If the player has a sword, deal damage to the goblin. Otherwise, the player takes damage.
4. **Response/Feedback Rules**:
   * Give players clear feedback about the result of their actions. For instance:
   * After the player moves, display the new room’s description.
   * If the player’s health changes, display it.
   * If the room is dark and the player doesn’t have a light source, mention they can’t see anything.
5. **Progression & End Conditions**:
   * Victory Condition:  “If the player reaches the main entrance with the key, print ‘You win!’ and end the game.”
   * Defeat Condition: “If the player’s health falls to 0 or below, print ‘You have died.’ and end the game.”
   * Story Milestones: “Once the player picks up the sword the statue comes to life.”

**Part 5: Reflection**

My project was designing an interesting decision making game. My system used a series of if-then statements to check what the player is trying to do against the area the player is in, the player itself then respond appropriately.

**Challenges**: Some challenges I had was when prompting the AI to fix some errors I had if I wasn’t specific enough the AI would suggest to change code that was not part of the problem. So, I had give the AI specific feedback on how I wanted it to work or what needed to be fixed. For example at one point the player would get flee feedback for the basement when entering a hallway, and it ended up being a indentation error. The AI is also not very creative when it came to designing the program so to get a starting point I had to give it every room, interaction, and item in said rooms and how to get to those rooms or the rooms would exist with no way for the player to get to them.