**Project Ideas**

* Device Troubling System - a system like the old-style help wizards: the user describes a problem with a device (e.g., a printer), and the system guides them through steps to fix it.
  + **Example**: Printer troubleshooting guide.
  + **Rules**:
    - *If the printer does not turn on, instruct user to check power cable.*
    - *If printers print blank pages and the ink level is low, instruct them to replace the ink cartridge.*
    - *If the printer is connected but not recognized, check drivers.*
* Game AI - a rule-based system for a small puzzle game or an advisor system that serves as an opponent movements in a board game (e.g., Tic-Tac-Toe, Checkers).
  + **Example**: Rule-based Tic-Tac-Toe engine.
  + **Rules**:
    - *If there is a move that leads to immediate win, make that move.*
    - *Otherwise, if the opponent can win the next move, block their move.*
    - *Otherwise, play in the center if available, else pick a corner.*
* Personal Finance Advisor - a system that takes a user’s financial situation (income, expenses, debts, savings goals) and provides advice based on a set of rules.
  + **Example**: Monthly budgeting tips generator
  + **Rules**:
    - *If monthly expenses exceed income, then advise "Cut discretionary spending."*
    - *If credit card debt is high and interest rates are high, recommend "Focus on paying off credit card debt first."*
    - *If a user has surplus, recommend "Invest in a high-interest savings account."*

I chose to go with the Game AI idea for a Tic-Tac-Toe game. I decided to go with this idea because I love Tic-Tac-Toe and I feel this would be a fun project to explore and build, keeping me engaged through completion.

**Tic-Tac-Toe Engine rules and logic:**

1. Representing the board:

3×3 board as a list (or array) of length 9, indexed as:

0 | 1 | 2

---+---+---

3 | 4 | 5

---+---+---

6 | 7 | 8

1. Defining winning lines to check:

Tic-Tac-Toe has 8 possible winning lines (3 rows, 3 columns, 2 diagonals). We can define them as tuples of indices then iterate over winning\_lines to see if there’s a line where we can win or need to block.

winning\_lines = [

(0, 1, 2), # Row 1

(3, 4, 5), # Row 2

(6, 7, 8), # Row 3

(0, 3, 6), # Column 1

(1, 4, 7), # Column 2

(2, 5, 8), # Column 3

(0, 4, 8), # Diagonal 1

(2, 4, 6) # Diagonal 2

]

1. Rules Priorities and Logic:

Rule 1: Win if possible - If there is a line where X appears twice and the third cell is empty, then place X there.

Rule 2: Block opponent’s win - If there is a line where O appears twice and the third cell is empty, then place X there to block.

Rule 3: Take the center if available - If the center (index 4) is empty, place X there.

Rule 4: Take any corner - If at least one of the corners is empty, place X in the first empty corner you find.

Rule 5: Take any side - If none of the higher-priority rules have applied, place X in the first empty side cell.

1. Handling Turn Order and Checking for Game End:

Game Flow:

* 1. Initialize empty board = [" "] \* 9.
  2. Keep track of whose turn it is, e.g., 'X' or 'O'.
  3. After each move:
  4. Check if the game is won by 'X' or 'O' by iterating over winning\_lines to see if any line is all 'X' or all 'O'.
  5. Check if the board is full (no ' ' left) => it’s a draw.
  6. Switch turns and keep playing until the game ends.

**Outcome**

Unless I intentionally placed a turn into a losing position, all matches ended in a tie.

**Reflection**

This system works by acting as an opponent in a game of Tic-Tac-Toe that follows a set of rules based on the users’ moves. AI prioritizes a set of rules to decide which move to make next with the ultimate objective of winning. The user gets to make the first move then the AI moves considering the users’ first move to avoid giving up a winning line to the user. Both the user and AI take turns until either a winning line is made, or the board is full resulting in a tie.

In collaborating with Chat GPT to design and code this project, I felt the experience was seamless. I found that being detailed and specific in my prompts helped provide accurate and effective responses. Chat GPT also did a great job with its explanations. Especially when it came to coding the program in Python. It explained each step in detail so I could understand the code logic, which helped me with adding comments in the code.