📌 Project Idea, Problem Statement & Goal

💡 Project Idea

The project is a maze racing game where a human player competes against an AI-controlled bot to reach a randomly placed goal inside a maze. The maze contains static obstacles that create random paths. The first player to reach the goal wins the round.

The AI opponent uses the A* pathfinding algorithm to calculate the shortest route to the goal and navigates the maze accordingly. The player controls their movement manually using the keyboard.

🛑 Problem Statement

Traditional maze games often lack real competition and dynamic movement, especially against basic AI. This reduces the challenge and limits player engagement. The challenge is to develop an AI that:

- Finds the shortest path efficiently using a known algorithm like A*.
- Provides a fair competition against a human-controlled player.
- Responds fast enough to give a sense of real-time gameplay.

🎯 Goal

The primary goal of this project is to create a competitive maze racing game between a human player and an AI bot using intelligent pathfinding. Specifically, the project aims to:

- Implement a bot that uses the A* algorithm to navigate a maze with random obstacles.
- Develop a turn-based or frame-based gameplay where both the player and bot race to the goal.
- Provide an interactive and visually clear experience using Pygame (Python).
- Allow multiple rounds with updated positions and scores.

🚀 Summary

This project delivers a competitive and educational game experience by combining maze navigation with AI-based pathfinding. Using A*, the bot becomes a strategic opponent to the player, encouraging engagement, planning, and fast thinking in a race to the goal.

📌 PEAS Model

- **Performance:**
    - Reaches the goal in the minimum time
    - Number of rounds won by each player
    - Smooth movement without collisions
    - Efficient pathfinding execution
- **Environment:**
    - 2D maze with static obstacles and a randomly placed goal
    - Human player using keyboard input
    - AI bot using A* algorithm
    - Visual interface using Pygame
- **Actuators:**
    - Player: Keyboard arrows
    - Bot: Movement commands (up, down, left, right) from A* path
    - On-screen visual updates
- **Sensors:**
    - Maze layout
    - Player and bot positions
    - Goal location
    - Goal reached or not

🎯 ODESA Model

- **Observe:** Bot observes the maze, its own position, and the goal location
- **Decide:** Calculates shortest path to goal using A* algorithm
- **Execute:** Moves one step along the path
- **Sense:** Checks if goal is reached or collision occurs
- **Act:** Updates score, triggers new round if needed