PYTHON GAME BOX

-Developed with Python

GitHub Respository Link:

https://github.com/Madhullika23/python-game-box

Team Members:

1. *Manya Agarwal (SRN: PES2UG25CS298)*
2. *Kritika Singh (SRN: PES2UG25AM135)*
3. *M Suraj (SRN: PES2UG25AM147)*
4. *Madhullika Ravi (SRN: PES2UG25EC072*)

TABLE OF CONTENTS

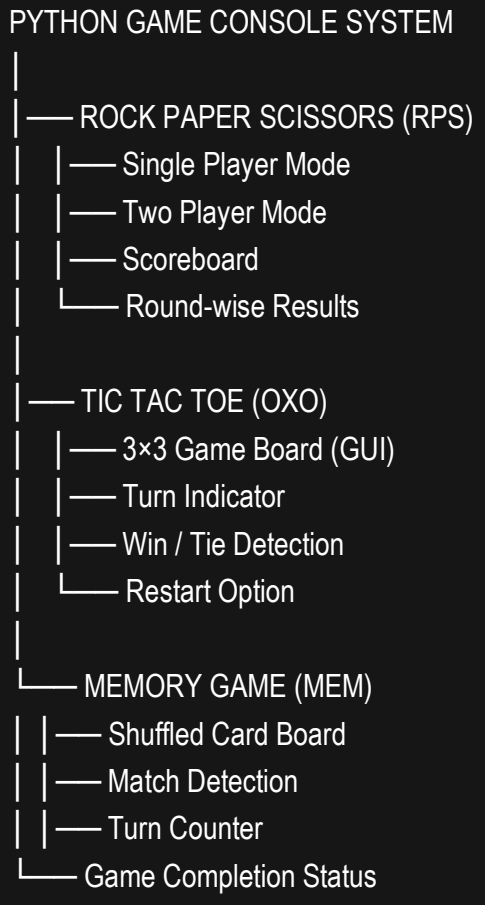
1. PROJECT OVERVIEW

This project is a Python-based multi-game console that allows users to play three different games within a single program. The games include Rock Paper Scissors (RPS), Tic Tac Toe (OXO), and a Memory Matching Game (MEM). The program combines both console-based and GUI-based interactions to offer a simple, engaging, and user-friendly gaming experience. It is designed using core Python concepts such as functions, loops, conditionals, Tkinter GUI, and randomization. The aim of the project is to understand game logic, user interaction handling, and multi-module integration in Python

2. PROBLEM STATEMENT

To design and develop a Python program that allows the user to choose between multiple games—Rock Paper Scissors, Tic Tac Toe, and Memory Match—within a single menu-driven application. The system must run smoothly, validate user inputs, maintain gameplay logic, and provide an enjoyable interactive experience.

3. DETAILED BLOCK DIAGRAM



4. APPROACH USED

1. Rock Paper Scissors (RPS)

- User chooses 1-player or 2-player mode.
- Player(s) give input as rock/scissor/paper.
- Computer uses the random module for its move.
- Winner is determined using conditional logic.
- Scoreboard shown after all rounds.

2. Tic Tac Toe (OXO)

- Implemented using Tkinter GUI.
- A 3×3 grid of buttons forms the game board.
- After each move, the game checks:
- Row, column, and diagonal wins
- Tie condition
- Winning cells are highlighted.
- Contains a restart button.

3. Memory Match Game (MEM)

- A shuffled list of paired symbols acts as the hidden board.
- User selects two card positions per turn.
- Correct matches stay revealed; mismatches are hidden again.
- Game uses loops and conditions until all pairs are found.

5. USER INSTRUCTIONS

Starting the Game

1. Run the Python file.
2. Enter your name when prompted.
3. Select a game from the menu: RPS/OXO/MEM

Rock Paper Scissors Instructions

- Choose : 1-player or 2-player mode.
- Select your move: Rock → 1 Scissor → 2 Paper → 3
- View round results and scores.
- Decide whether to continue playing.

Tic Tac Toe Instructions

- The Tic Tac Toe window will open.
- Player X begins the game.
- Click on any empty tile to place your symbol.
- Continue until someone wins or the board ties.
- Use the Restart button to begin a new game.

Memory Game Instructions

- The board shows numbers (hidden cards).
- Select two positions (1–12).
- If the symbols match, they stay revealed.
- If not, they hide again.
- Continue until all pairs are found.

5. Decide whether to continue playing

6. SAMPLE INPUT/OUTPUT

INPUT EXAMPLE:

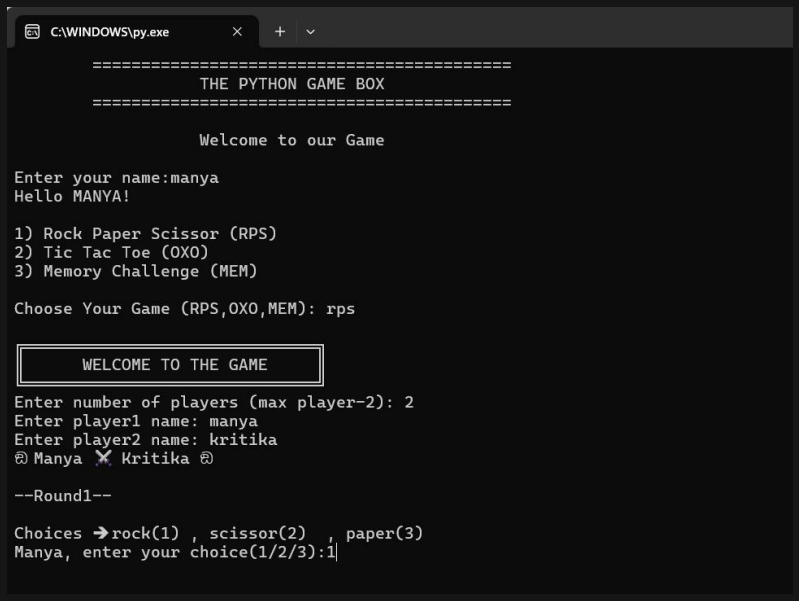
Enter your name: Manya
Choose Your Game: RPS
Enter number of players: 1
Choose your choice: 2
Play Again: Y

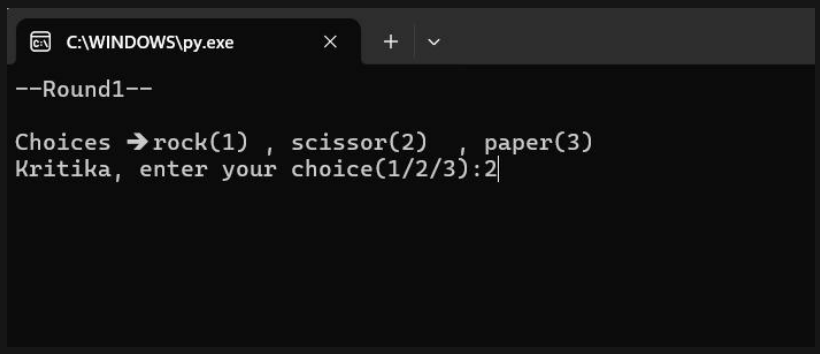
OUTPUT EXAMPLE:

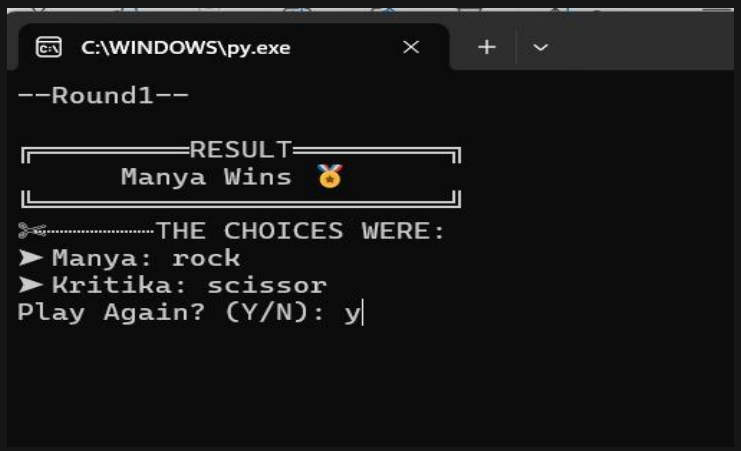
WELCOME TO THE GAME
Manya Wins 🏆
Rounds Played: 2
Final Score displayed

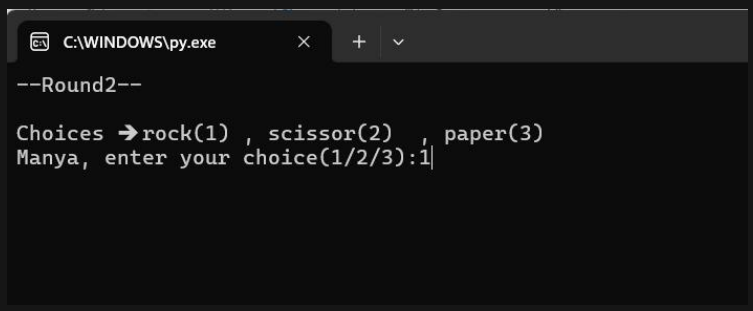
7. SCREENSHOTS

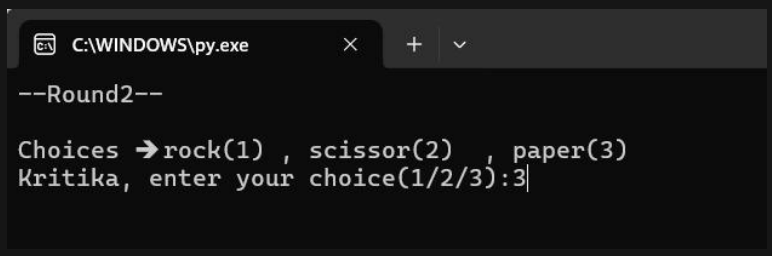
1. Rock-Paper-Scissor (RPS)

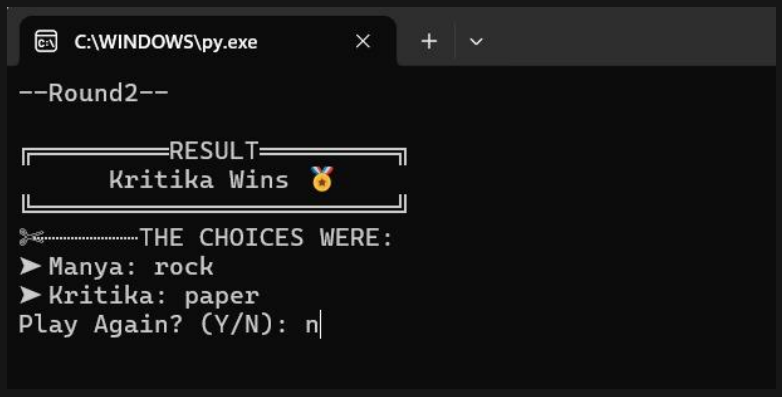






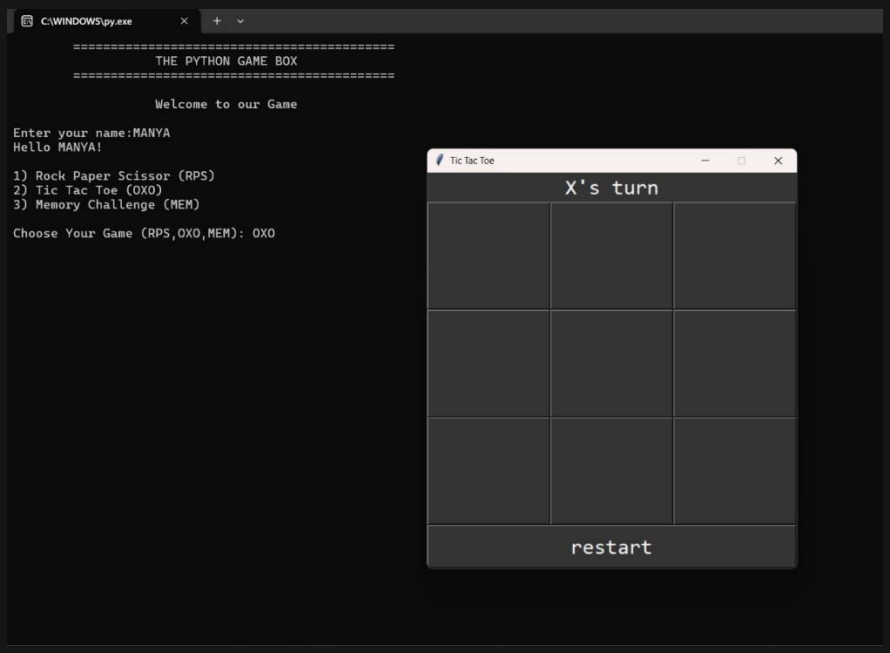

C:\WINDOWS\py.exe
--Round2--
Choices ➔rock(1) , scissor(2) , paper(3)
Manya, enter your choice(1/2/3):1

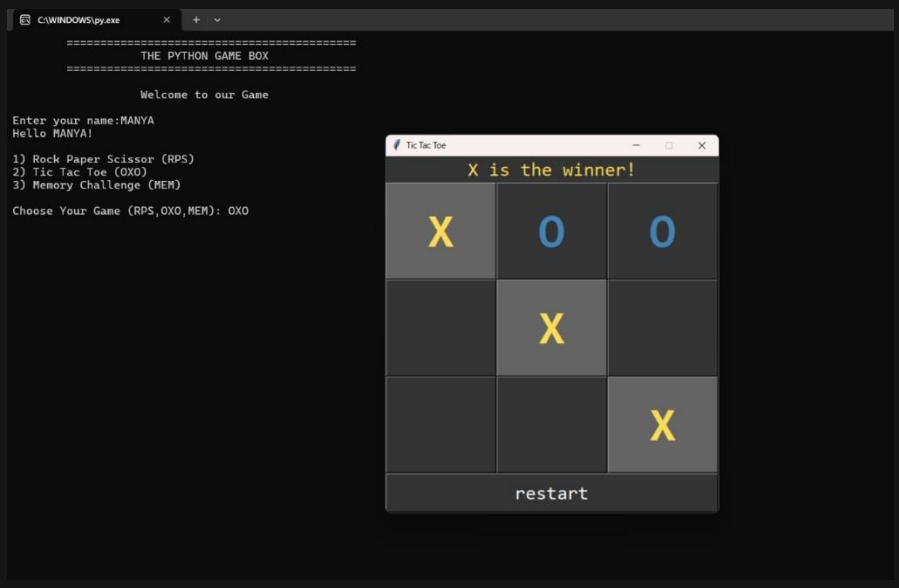

C:\WINDOWS\py.exe
--Round2--
Choices ➔rock(1) , scissor(2) , paper(3)
Kritika, enter your choice(1/2/3):3


C:\WINDOWS\py.exe
--Round2--
RESULT
Kritika Wins 🏅
THE CHOICES WERE:
➤ Manya: rock
➤ Kritika: paper
Play Again? (Y/N): n

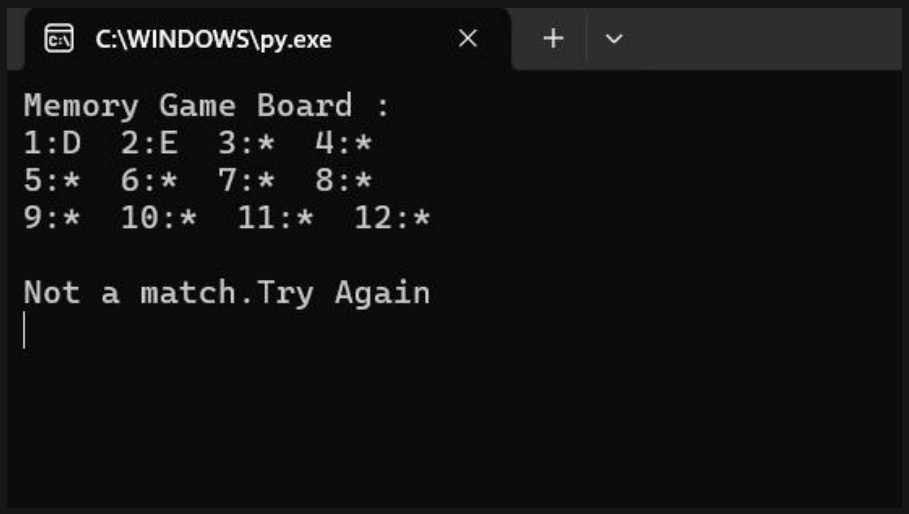

C:\WINDOWS\py.exe
SCORE BOARD
Rounds Played: 2
➤ Manya total score: 1
➤ Kritika total score: 1
OVERALL WINNER
⚑ TIE ⚑
Do you want to play again? (Y/N): n

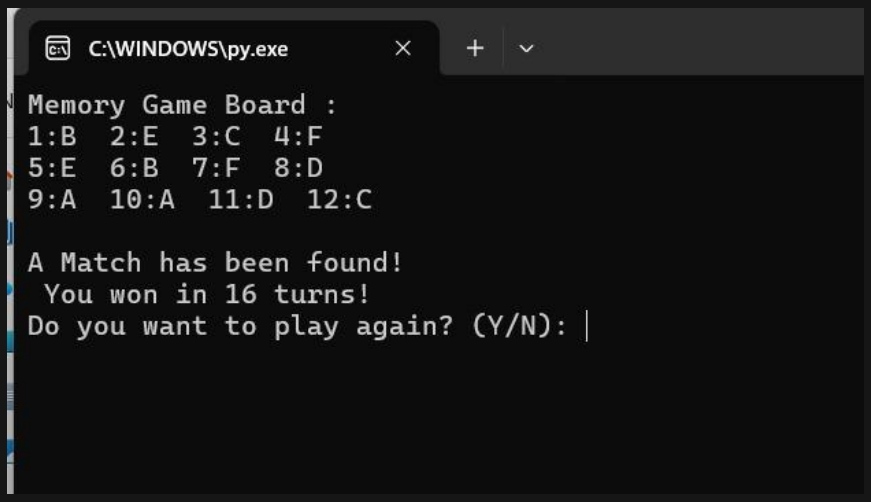
2. Tic Tac Toe (OXO)





3. Memory Game (MEM)





8. CHALLENGES FACED

1. Handling invalid inputs such as letters and out-of-range values.

2. Maintaining smooth game flow using loops and restart conditions.

3. Implementing correct winner logic in Tic Tac Toe.

4. Managing scores and variables across functions.

5. Designing and resetting the Tkinter interface properly.

6. Testing and debugging all games through repeated trials.