আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
الجامعة الإسلامية العالمية شيتاغونغ
International Islamic University Chittagong

LAB REPORT

Course Code: CSE-3636
Course Title: Artificial Intelligence

Submitted To: Sara Karim
Adjunct Lecturer
Dept. of CSE, IIUC

Submitted By:
Name: Rahenuma Raiyan Haider
ID: C221295
Name: Nowshine Sharmili Piuli
ID: C223285
Semester: 6th
Section: C
Date of submission: 09/07/2025

Remarks:

Project Report: Rock-Paper-Scissors Game

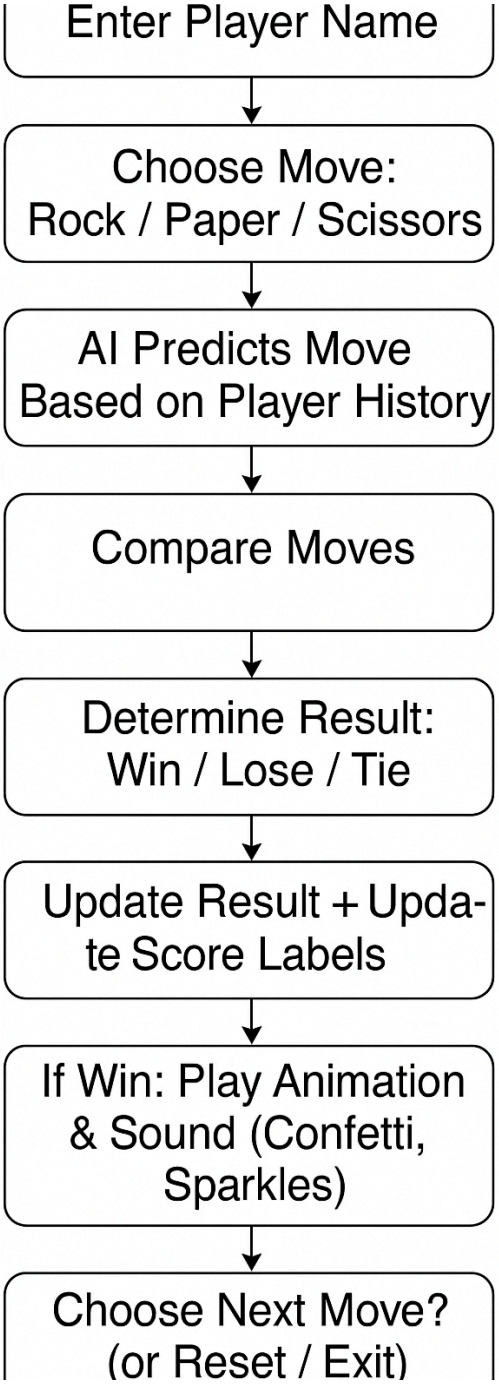
Introduction:

This Rock-Paper-Scissors game is a Python desktop application built with Tkinter, featuring an AI opponent that learns from the player's choices. It includes real-time score tracking, personalized player stats, a dynamic leaderboard, and fun win effects like confetti and sparkles. The game offers a smooth user experience with a clean interface, making it both engaging and visually appealing.

Objectives:

The objective of this project is to develop an interactive Rock-Paper-Scissors game using Python and Tkinter that not only allows players to compete against an AI with predictive behavior but also enhances the gaming experience through real-time score tracking, personalized player statistics, leaderboard ranking, and visual effects such as confetti and sparkles. The project aims to demonstrate the integration of GUI design, basic AI logic, and user engagement features within a simple yet entertaining desktop application.

Flowchart:



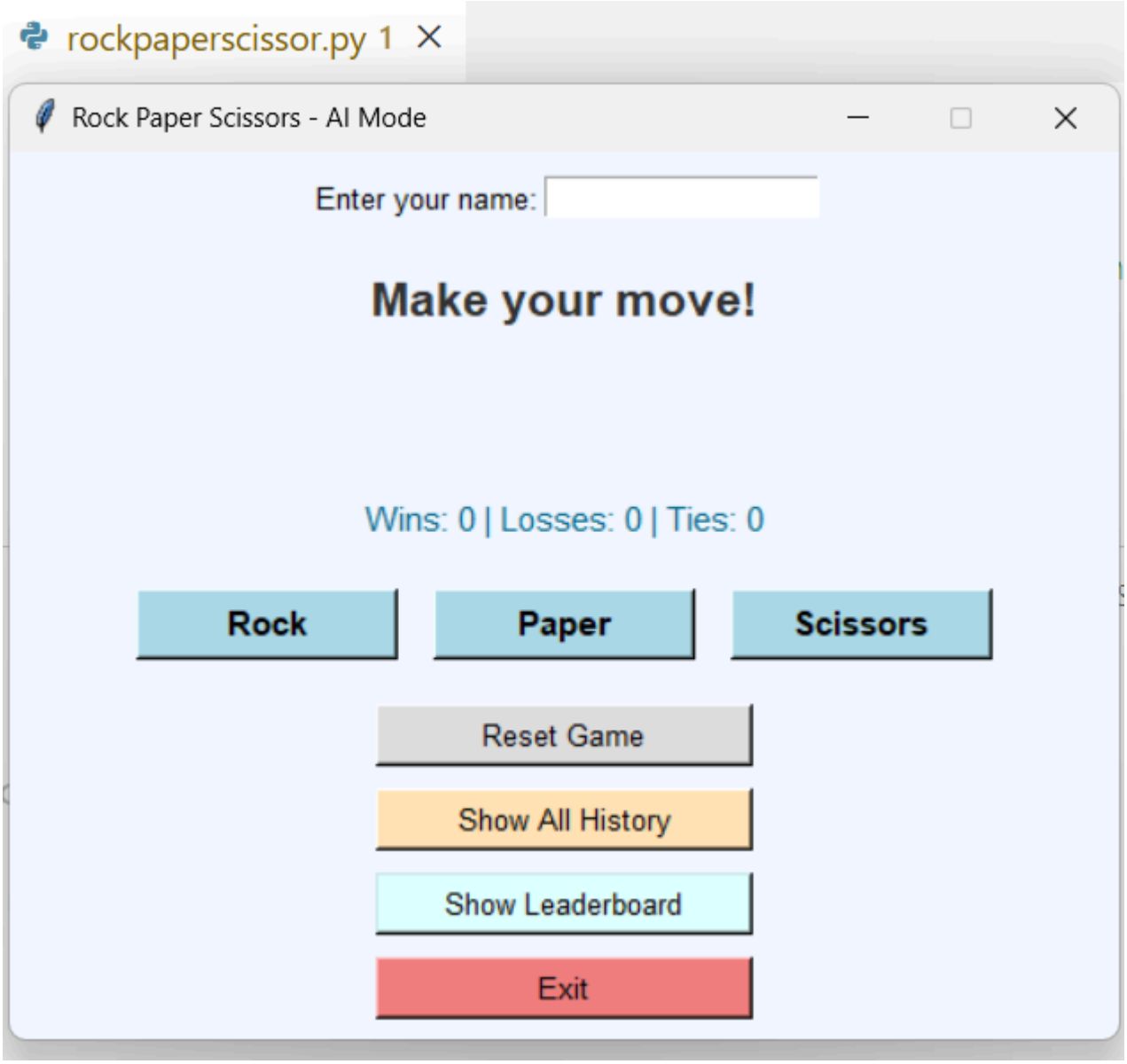
Code Implementation:

Although this is a console-based application and not a web or mobile app, the concepts of frontend and backend still apply in a simplified way.

Frontend:

In a web or desktop GUI application, the frontend is the graphical interface the user interacts with. But in this CLI (Command Line Interface) game, the frontend consists of the text-based input and output:

- Prompting the user for a move
- Displaying choices and results
- Asking to continue or exit
- Show all history
- Reset game
- Show leaderboard



Backend:

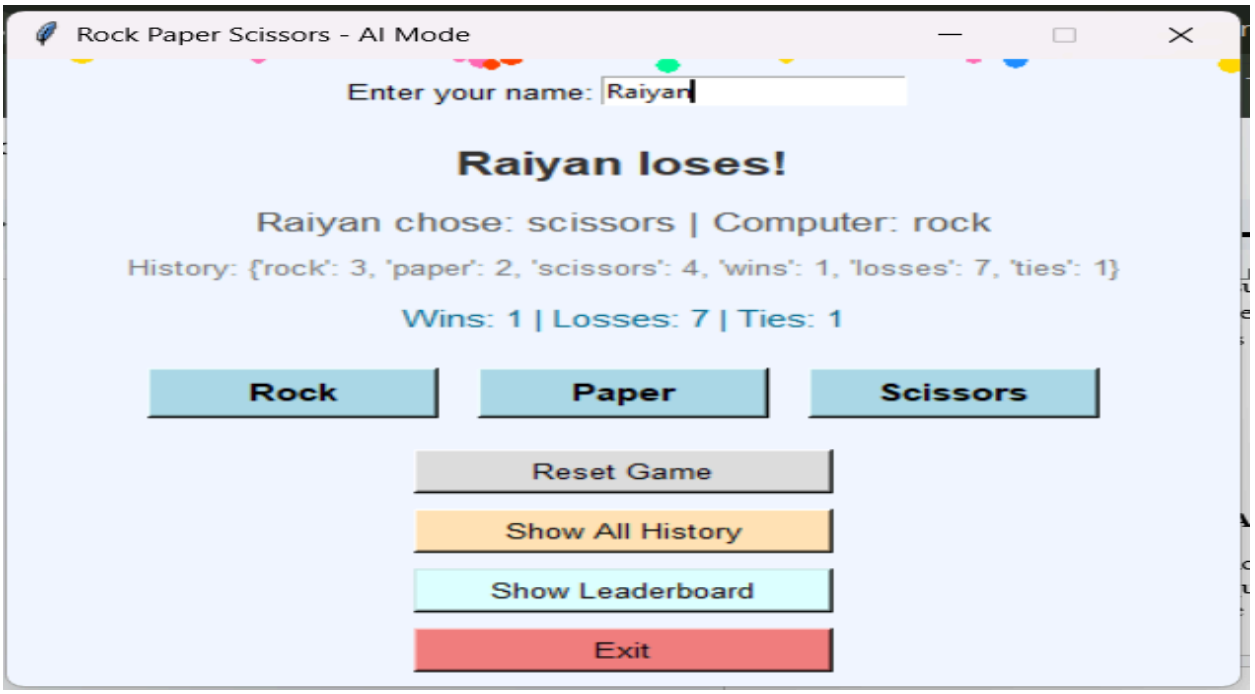
The backend refers to the core logic and functionality that drives the game:

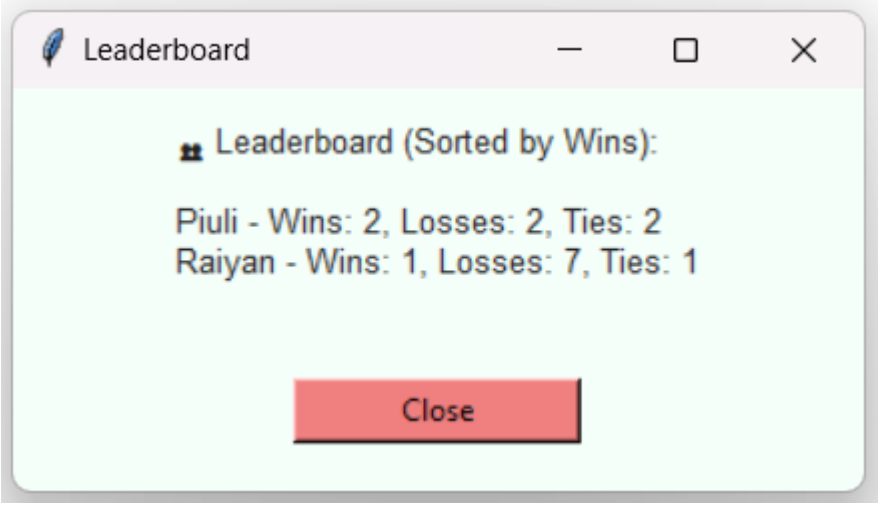
- The random.choice() logic to simulate computer behavior
- The if-elif-else conditions to decide the winner
- Loop control to keep or end the game
- Data validation (ensuring valid player input)

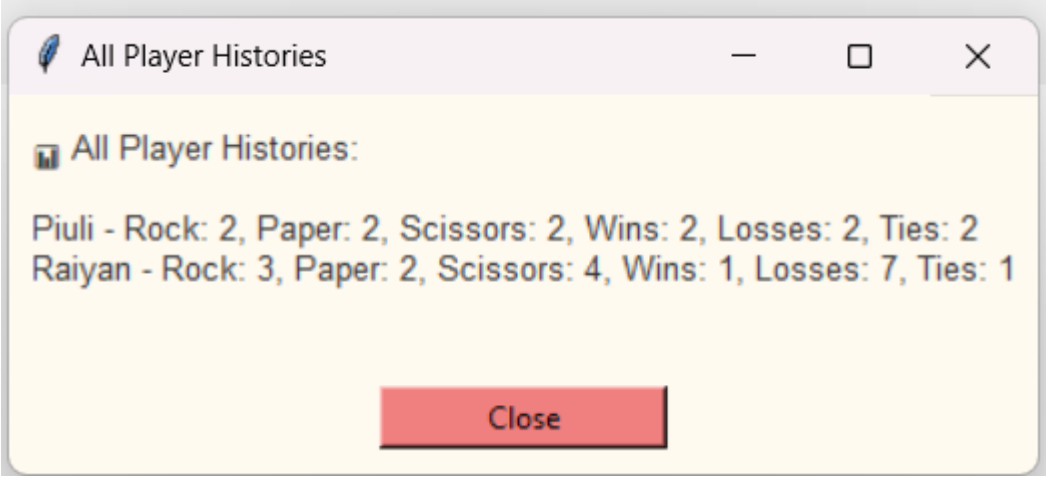
```python
import tkinter as tk
import random
import winsound  # For playing win sound on Windows

# --- Setup main window ---
root = tk.Tk()
root.title("Rock Paper Scissors - AI Mode")
root.geometry("500x400")
root.resizable(False, False)
root.configure(bg="#f0f8ff")

options = ("rock", "paper", "scissors")
player_stats = {}  # Stores each player's history

# --- Player name input ---
name_frame = tk.Frame(root, bg="#f0f8ff")
name_frame.pack(pady=10)

name_label = tk.Label(name_frame, text="Enter your name:", font=("Arial", 10), bg="#f0f8ff")
name_label.pack(side="left")

player_name_var = tk.StringVar()
name_entry = tk.Entry(name_frame, textvariable=player_name_var)
name_entry.pack(side="left")

# --- Labels for result display ---
result_label = tk.Label(root, text="Make your move!", font=("Arial", 16, "bold"), bg="#f0f8ff", fg="#333")
result_label.pack(pady=10)

choice_label = tk.Label(root, text="", font=("Arial", 13), bg="#f0f8ff", fg="#555")
```

Result:

The developed Rock-Paper-Scissors game successfully allows players to compete against an AI that adapts based on their move history. It provides real-time score updates, personalized player statistics, a sorted leaderboard, and engaging visual effects like confetti and sparkles upon winning. The user interface is interactive and easy to navigate, and the game handles multiple players' histories within a single session. Overall, the project achieves its goal of creating an enjoyable and intelligent desktop game using Python and Tkinter.







Algorithm Used:

1. Frequency Analysis for AI Prediction

The AI tracks how often each player chooses rock, paper, or scissors. It then predicts the player's next move based on the most frequently selected option and counters it with the winning move. This makes the AI behave in a smarter, more adaptive way.

2. Decision Logic for Game Outcome

After both player and AI make their choices, the game checks the rules to determine the outcome. It compares the moves and decides whether it's a win, loss, or tie using simple conditional statements.

3. Real-Time Score and History Update

The player's win, loss, tie count, and move history are updated instantly after each round. This allows accurate tracking of performance and enables features like score display and leaderboard ranking.

Error that faced during this project:

1.Confetti Layer Not Visible
→ Solved by using confetti_canvas.lift() and lower() to manage widget stacking.

2.Sound Not Playing
→ Occurred on non-Windows systems due to use of winsound; resolved by noting OS compatibility.

3.Animation Overlap or Flicker
→ Sparkles or confetti animations clashed or didn't clear properly; fixed with timed deletion and after() usage.

4.AI Prediction Not Updating Correctly
→ Caused by uninitialized player stats; handled by checking and initializing player data properly.

6.Score Not Refreshing Immediately
→ Needed manual root.update_idletasks() to reflect UI changes instantly.

7.Empty Player Name Caused Ambiguity
→ Solved by assigning a default name like "Player" when input is blank.

8.Multiple Players Overwriting Data
→ Prevented by uniquely identifying players via name input and managing stats in a dictionary.

Future Consideration:

1. Add Machine Learning for Smarter AI
Instead of using frequency analysis, implement machine learning models like decision trees or reinforcement learning to detect deeper patterns in player behavior and adapt dynamically.

2. Multiplayer Support (Local or Online)
Introduce a two-player mode—either locally on the same device or online via socket programming or web integration—so players can compete with friends.

3. Persistent Storage for Player Data
Use a database or local file system to store player stats and leaderboard data permanently, allowing players to resume progress across sessions.

4. Custom Avatars and Themes
Allow players to select avatars and apply custom color themes or background images for a more personalized experience.

6. Advanced Analytics Dashboard
Add visual charts or summaries of player performance over time to help users understand and improve their strategies.

Limitation:

1. The AI uses only basic frequency analysis for predictions.
2. Player data is not saved after closing the application.
3. The game supports only single-player mode.
4. Sound effects work only on Windows systems.
5. No input validation for player names.
6. The AI cannot handle non-repetitive or bluffing strategies.
7. The interface is not optimized for mobile or web platforms.

Conclusion:

This Rock-Paper-Scissors game project demonstrates how a simple Python GUI application can be enhanced with basic AI logic, real-time stats tracking, and engaging visual effects to create an interactive user experience. While it effectively showcases fundamental concepts like event-driven programming, decision-making, and UI design, it also opens up possibilities for future enhancements such as smarter AI, multiplayer support, and persistent data storage to make it more dynamic and scalable.