VIT®

BHOPAL

Memory Match Game – Project Report

Submitted by : SUPRIYA SUMAN

Course: B.Tech – (AI&ML)

Registration No. – 25BAI10171

1. Introduction

This project is a desktop-based Memory Match Game built using Python and Tkinter. The game helps users improve focus, memory retention, and responsiveness. It offers a clean interface, timer, move counter, and a scoring system.

2. Problem Statement

People often need short mental breaks to refresh focus during long study or work sessions. Existing games are either too complex or lack simple functionality. This project aims to provide a lightweight, offline, brain-refreshing memory game.

3. Functional Requirements

- Display a 4x4 memory card grid

- Allow card reveal on click

- Match detection for identical emoji pairs

- Track moves, time, and matched pairs

- Store best score in a local JSON file

- Provide reset/shuffle functionality

- Show completion popup with performance stats

4. Non-functional Requirements

- Easy-to-use interface

- Quick response time

- Lightweight and portable

- Minimal dependencies (Tkinter only)

- Runs offline

5. System Architecture

The system follows a simple event-driven structure.

Components:

- UI Layer (Tkinter buttons & labels)

- Logic Layer (match detection, timer updates)

- Storage Layer (JSON-based best score saving)

6. Design Diagrams (Text Descriptions)

○ Use Case Diagram:

- Actor: User

- Use Cases: Start game, reveal card, match cards, reset board.

○ Workflow Diagram:

User clicks → Card reveals → Two cards open → Match/No Match → Update score → Continue.

○ Sequence Diagram:

User → Button Click → Reveal Function → Logic Check → Update UI → Save Best Score.

○ Class/Component Diagram:

- Game UI Component

- Game Logic Component

- Storage Handler

○ ER Diagram (JSON Storage):

- Entity: BestScore { moves, time }

7. Design Decisions & Rationale

- Tkinter chosen for simplicity and built-in availability.

- JSON chosen due to lightweight nature and easy read/write.

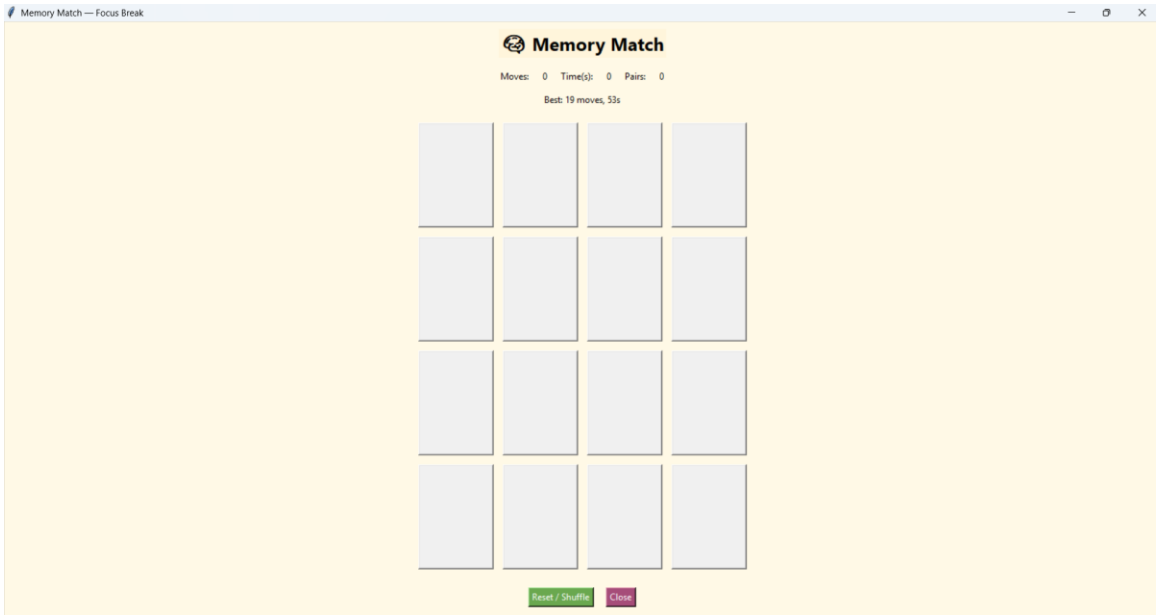
- Emojis used instead of images for universal compatibility.

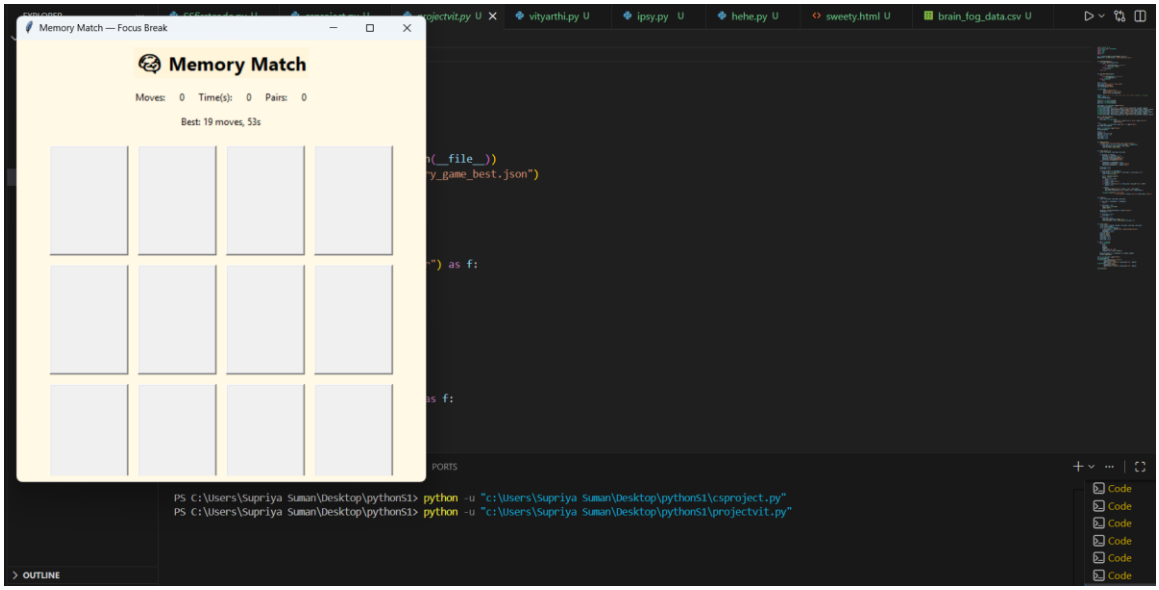
- 4x4 grid selected for balanced difficulty.

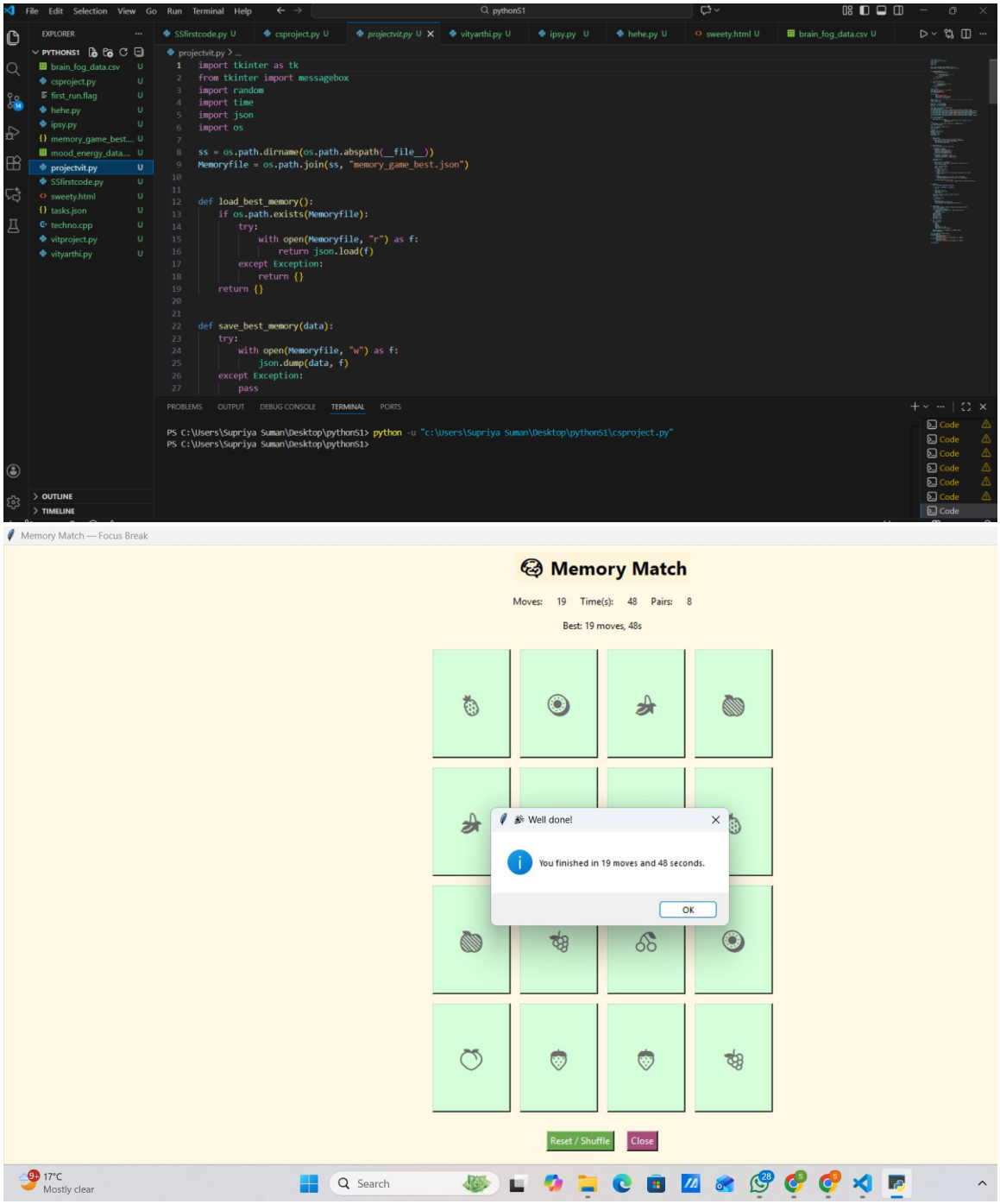
8. Implementation Details

The game uses Tkinter for UI and Python lists to store card state. A timer updates every 500 ms, and best scores are saved to JSON. Buttons represent cards, and event callbacks handle game logic.

9. Screenshots / Result







10. Testing Approach

- Manual testing: button clicks, card matching, reset functionality.

- Timer correctness testing.

- JSON file read/write verification.

- Edge case: double-click fast actions, mismatched pairs.

11. Challenges Faced

- Managing reveal lock to prevent multiple fast clicks.

- Ensuring JSON file works across different systems.

- Resetting button default colors for different operating systems.

12. Learnings & Key Takeaways

- Deep understanding of Tkinter event-driven programming.

- Gained experience with JSON handling.

- Improved debugging skills.

- Understood UI state management.

13. Future Enhancements

- Difficulty levels (3x3, 4x4, 6x6).

- Sound effects.

- Animated card flips.

- Player profile and score tracking.

- Web version using React/Pygame.

14. References

https://docs.python.org/3/

https://docs.python.org/3/library/tkinter.html

https://docs.python.org/3/library/random.html

https://docs.python.org/3/library/time.html

https://docs.python.org/3/library/json.html

https://www.geeksforgeeks.org/python-gui-tkinter/

https://www.tutorialspoint.com/python/python_gui_programming.htm