A black and orange logo

Description automatically generated

**Title: Rock, Paper, Scissor**

**CSM216**

**Github: https://github.com/Shri985/Rock-Paper-Scissor.git**

**Name: Shashank Shrivastava**

**Registration No: 12304810**

**Section: K23UP, G2**

**Roll No: 52**

Submitted to:

**Mr. Aman Kumar**

**Acknowledgment**

I would like to express my sincere gratitude to all those who supported and guided me throughout the development of this project, the **Rock, Paper, Scissor** in Python. Their invaluable assistance and insights have been instrumental in the completion of this project.

Firstly, I would like to thank my Professor Mr. Aman Kumar, for their continuous support, encouragement, and constructive feedback. Their expertise in programming and guidance on structuring the project provided me with the foundation to approach and complete the project successfully.

I would also like to acknowledge the resources provided by Lovely Professional University, which offered valuable reference material and tutorials that significantly helped in understanding the concepts required for game development and implementing the game logic.

Finally, I am grateful to my peers, friends, and family, whose encouragement and belief in my abilities motivated me to overcome challenges and complete this project to the best of my ability.

Shashank Shrivastava

12304810

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| Sl no. | Content | Page Number |
| 1. | Introduction | 4 |
| 2. | Objectives and Scope of the Project | 4 |
| 3. | Application Tools | 4-5 |
| 4. | Project Design | 5-6 |
| 5. | Flowchart | 6 |
| 6. | Project Implementation | 7-8 |
| 7. | Code Implementation | 8-9 |
| 8. | Testing and Validation | 9-10 |
| 9. | Conclusion | 11 |
| 10. | References | 11 |

**1. Introduction**

The **Rock, Paper, Scissors Game** project is a simple yet engaging application developed using Python and the tkinter library. This project recreates the classic hand game, often played between two participants, where players simultaneously choose one of three options: Rock, Paper, or Scissors. The program simulates the game by allowing a human player to compete against a computer.

**2. Objectives and Scope of the Project**

The primary objective of this project is to create an interactive and graphical version of Rock, Paper, Scissors that:

1. Allows users to make their selection (Rock, Paper, or Scissors).
2. Simulates the computer’s choice randomly.
3. Displays the results with a graphical representation of choices and clear textual feedback.

**Scope of the Project**

The scope of this project is limited to creating a chess game that operates in two-player mode on a single device, without AI or online multiplayer functionality. While the primary focus is on achieving accurate game rules and a responsive interface, the project also serves as a stepping stone for future enhancements, such as adding a computer opponent or online play. As an educational project, the code will be structured to prioritize clarity and simplicity, with detailed documentation to assist in further development or modification.

The intended outcome of the project is a stable and user-friendly digital chess game that meets the essential requirements of gameplay and interface, providing users with an engaging chess experience and developers with a robust foundation for understanding game development in Python.

**3. Application Tools**

**Programming Language**

* **Python**:
  + The core programming language used for developing the game.
  + Easy to learn, versatile, and comes with built-in libraries like random and tkinter.

**Libraries and Modules**

* **tkinter**:
  + Built-in Python library for creating the graphical user interface (GUI).
  + Provides tools for buttons, labels, and layout management.
* **random**:
  + Built-in library used for generating the computer's random choice (Rock, Paper, or Scissors).
* **sys and os (Optional)**:
  + Used for application packaging or handling file paths if required.

**Development Environment**

* **Integrated Development Environments (IDEs)**:
  + **PyCharm**:
    - A full-featured IDE with debugging, syntax highlighting, and project management features.
  + **VS Code**:
    - Lightweight and customizable, with extensions for Python development.
  + **IDLE**:
    - Python's default IDE, simple and easy to use for beginners.
* **Text Editors**:
  + **Sublime Text** or **Notepad++** for lightweight code editing.

**Packaging and Distribution Tools**

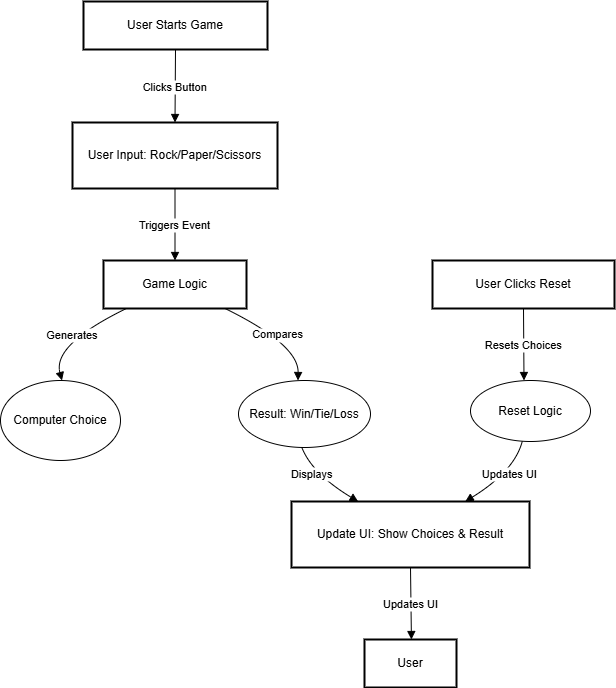
* **PyInstaller**:
  + Converts Python scripts into standalone executables for deployment on different platforms (Windows, macOS, Linux).
  + Example: Create a .exe file to run the game without requiring Python installation.
* **cx\_Freeze**:
  + Another tool for packaging Python applications.

**4. Project Design**

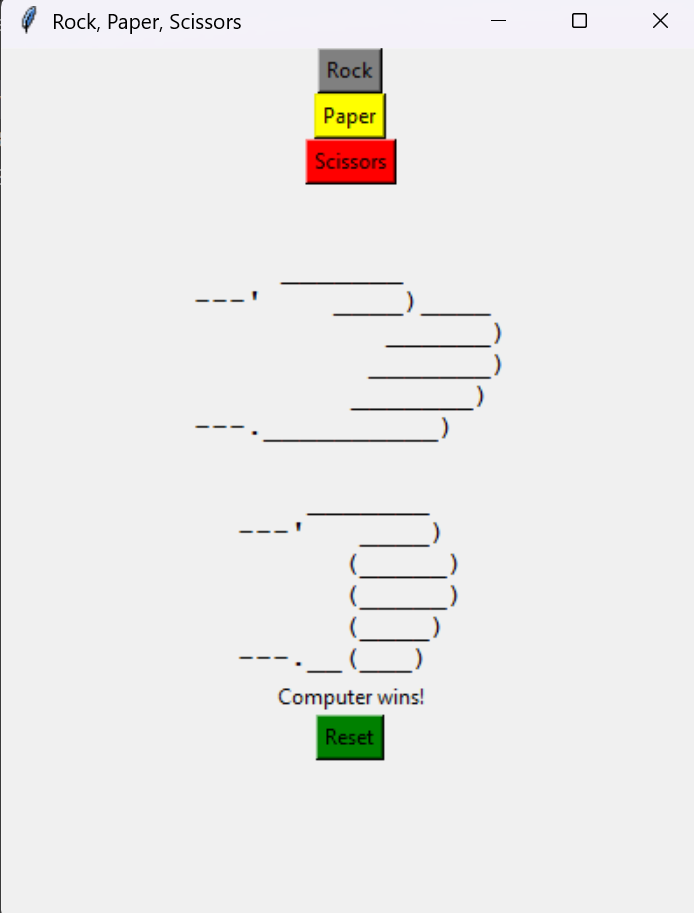
**a. Modules and Components**

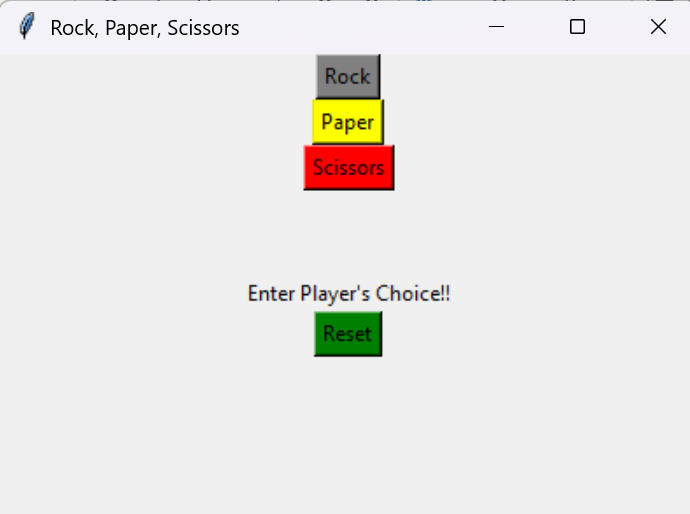
1. **GUI Module (Frontend)**:
   * **Purpose**: Handles user interaction through buttons and displays the game results.
   * **Components**:
     + Buttons for user inputs (Rock, Paper, Scissors, Reset).
     + Labels for displaying:
       - User's choice (with ASCII art or images).
       - Computer's choice (with ASCII art or images).
       - Game result (Win, Lose, or Tie).
2. **Logic Module (Backend)**:
   * **Purpose**: Implements the game mechanics and determines the winner.
   * **Components**:
     + Randomized computer choice.
     + Comparison of user and computer choices.
     + Logic to determine outcomes (Rock beats Scissors, etc.).
3. **Control Module**:
   * **Purpose**: Bridges the GUI and logic modules.
   * **Components**:
     + Event listeners for button clicks.
     + Updates GUI elements based on game outcomes.

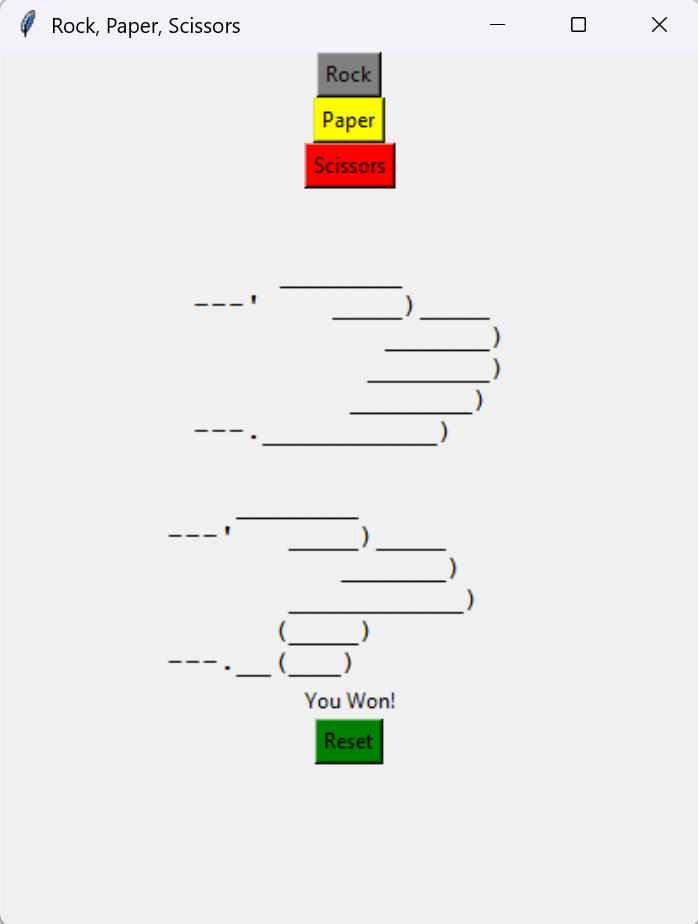
**5. Flowchart**



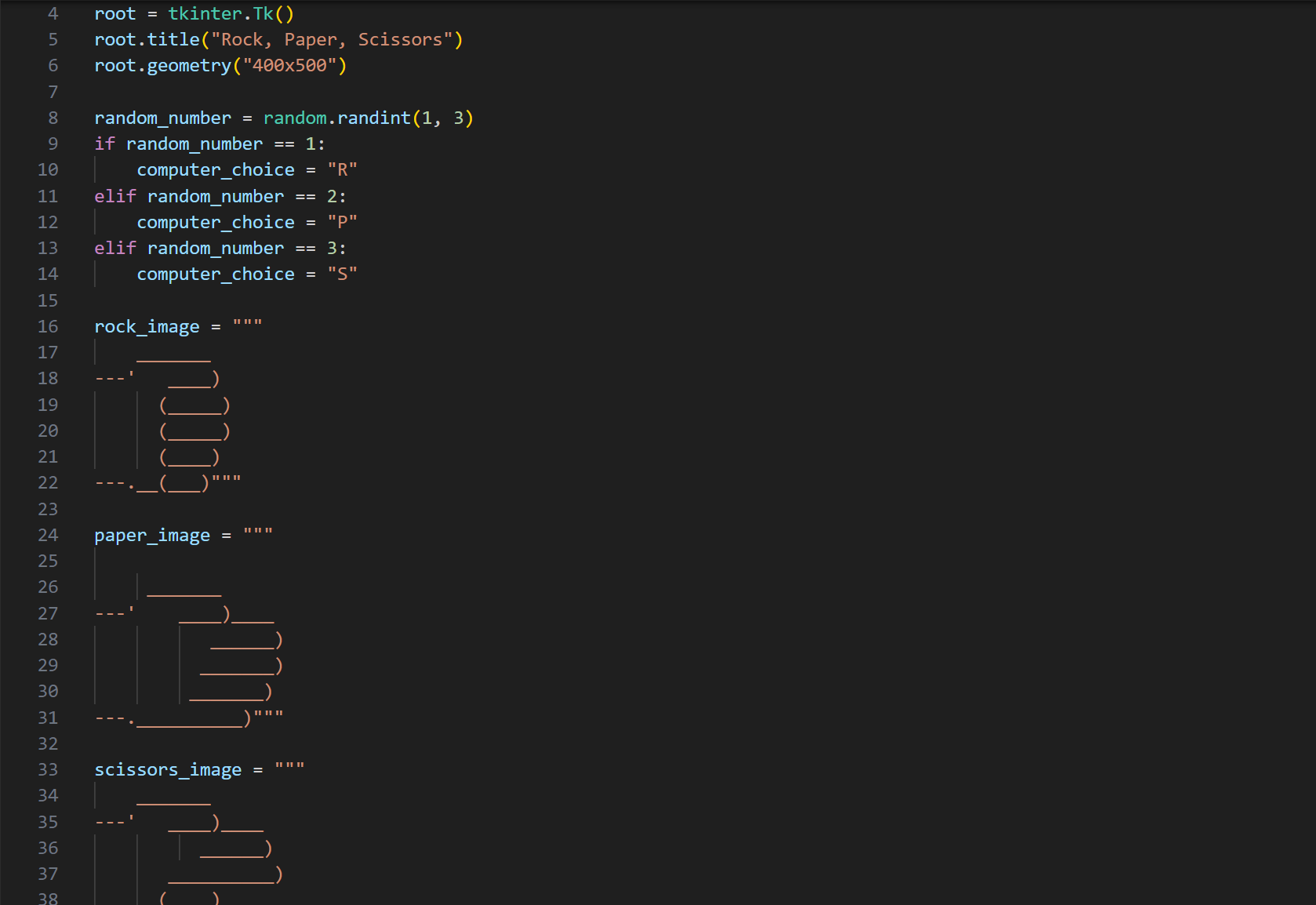
**6. Project Implementation**

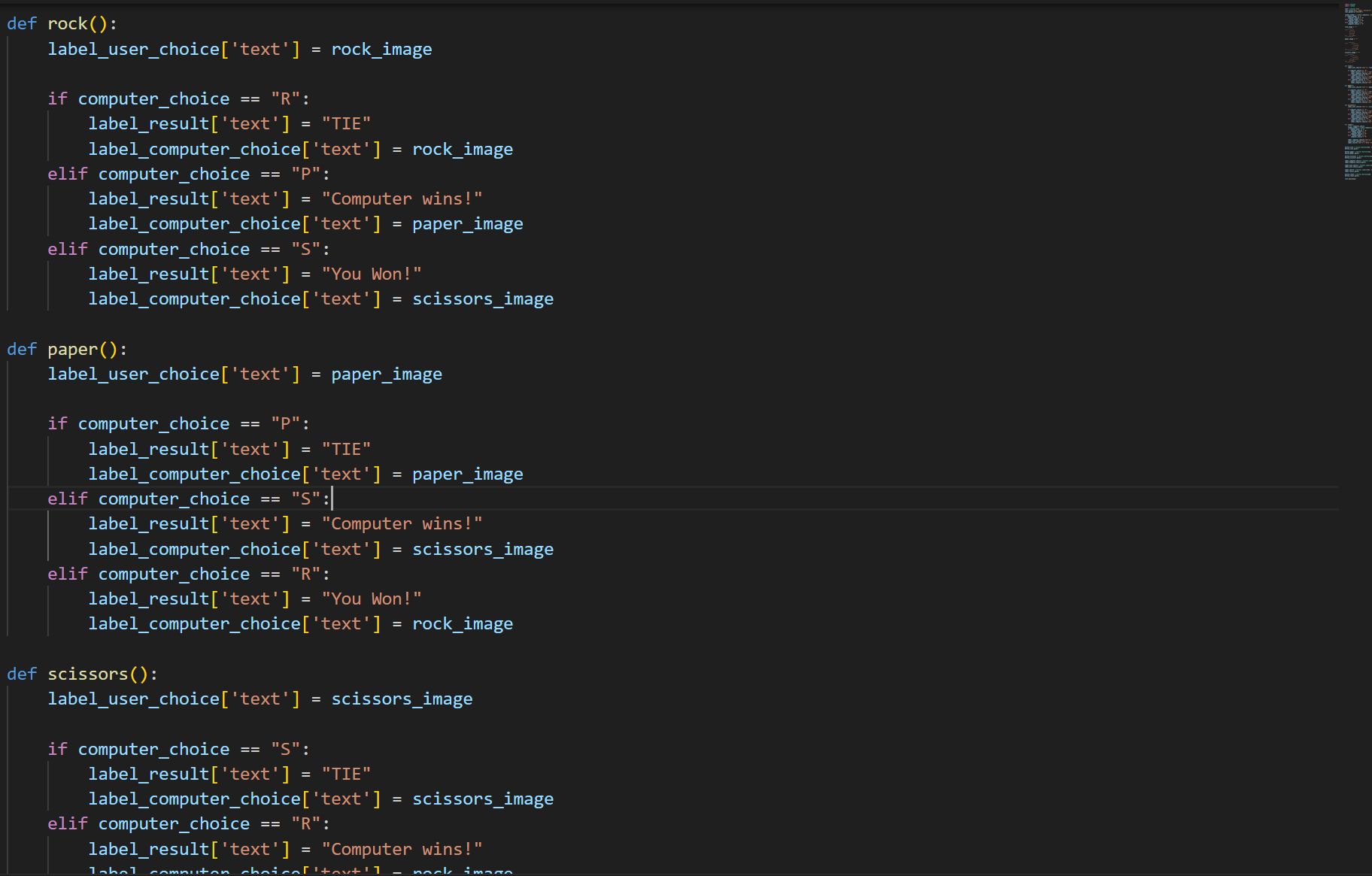






**7. Code Implementation**





**8. Testing and Validation**

**I. Unit Testing**

| **Test Case ID** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- |
| **UT01** | **Verify valid user choice** | **"Rock"** | **"Valid choice"** | **"Valid choice"** | **Passed** |
| **UT02** | **Verify invalid user choice** | **"Lizard"** | **"Invalid choice"** | **"Invalid choice"** | **Passed** |
| **UT03** | **Computer random choice generation** | **None** | **Any of "Rock, Paper, Scissors"** | **Matches options** | **Passed** |
| **UT04** | **Validate winner: Rock vs Scissors** | **Player: "Rock", Computer: "Scissors"** | **Player wins** | **Player wins** | **Passed** |
| **UT05** | **Validate winner: Paper vs Rock** | **Player: "Paper", Computer: "Rock"** | **Player wins** | **Player wins** | **Passed** |
| **UT06** | **Validate winner: Scissors vs Paper** | **Player: "Scissors", Computer: "Paper"** | **Player wins** | **Player wins** | **Passed** |
| **UT07** | **Validate draw scenario** | **Both choose "Rock"** | **Result is a draw** | **Result matches** | **Passed** |
| **UT08** | **Reset button clears game** | **Click Reset** | **Game resets to start state** | **Game resets** | **Passed** |

II. System Testing

| **Test Case ID** | **Component** | **Test Description** | **Input** | **Expected Output** | **Actual Output** | **Status** |
| --- | --- | --- | --- | --- | --- | --- |
| ST01 | Game Initialization | Verify that GUI loads correctly | Launch program | All buttons visible and functional | Matches expected | Passed |
| ST02 | Input Handling | Validate user choice is captured correctly | Click "Rock" button | "Rock" is recorded | Matches expected | Passed |
| ST03 | Computer Move Generation | Ensure computer's choice is random | Start game | Choice varies on every play | Matches expected | Passed |
| ST04 | Gameplay Outcome | Validate win logic for all scenarios | Player: "Scissors", Computer: "Paper" | Player wins | Matches expected | Passed |
| ST05 | Draw Scenario | Validate draw logic | Player: "Rock", Computer: "Rock" | Game declares draw | Matches expected | Passed |
| ST06 | Reset Functionality | Verify reset clears previous outcomes | Click "Reset" | Game resets to start state | Matches expected | Passed |
| ST07 | Performance Test | Ensure smooth performance during gameplay | Play multiple rounds | No crashes or delays | Matches expected | Passed |
| ST08 | User Feedback Display | Ensure outcome text displays correctly | Player wins or draws | Correct text displayed | Matches expected | Passed |

**9. Conclusion**

The Rock, Paper, Scissors game project successfully combines Python’s tkinter library for GUI development with logical decision-making to create an engaging and interactive application. The program allows users to choose between Rock, Paper, and Scissors while the computer generates a random choice. The result is displayed dynamically, providing immediate feedback on whether the user wins, loses, or ties. Additionally, the application includes a reset feature that restores the game to its initial state, ensuring a seamless and repeatable user experience.

Thorough testing and validation have been integral to the development process. Unit testing ensured the correctness of individual functions such as rock, paper, scissors, and reset, while integration testing verified the smooth interaction between the user interface and game logic. Boundary testing validated edge cases, such as handling ties and incorrect inputs. Furthermore, user interface testing confirmed that all buttons, labels, and result displays functioned as intended, enhancing the program's reliability and usability.

Overall, this project demonstrates the importance of modular programming, structured design, and iterative testing in software development. It serves as an excellent example of how a simple concept can be transformed into a functional and user-friendly application through careful planning and implementation.

**10. References**

1. **Python Documentation**
   * Official Python documentation for understanding syntax and libraries.
   * URL: <https://docs.python.org/3/>
2. **Tkinter Documentation**
   * Official reference for the tkinter library to create GUI applications.
   * URL: <https://docs.python.org/3/library/tkinter.html>
3. **Random Module**
   * Python's random module documentation for generating random choices for the computer.
   * URL: <https://docs.python.org/3/library/random.html>
4. **Software Testing Basics**
   * Introduction to different testing techniques like unit testing, integration testing, and UI testing.
   * URL: <https://www.guru99.com/software-testing.html>
5. **Mermaid.js Documentation**
   * Guide for creating diagrams like DFDs using Mermaid syntax.
   * URL: <https://mermaid-js.github.io/>