**Industrial Internship Report on**

**”Project Name”**

**Prepared by**

**PATEL SRILAXMI**

|  |
| --- |
| *Executive Summary* |
| This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).  This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.  My project was Rock Paper Scissors  This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship. |

**TABLE OF CONTENTS**

[1 Preface 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)

[2.3 Objective 9](#_Toc139702810)

[2.4 Reference 9](#_Toc139702811)

[2.5 Glossary 10](#_Toc139702812)

[3 Problem Statement 11](#_Toc139702813)

[4 Existing and Proposed solution 12](#_Toc139702814)

[5 Proposed Design/ Model 13](#_Toc139702815)

[5.1 High Level Diagram (if applicable) 13](#_Toc139702816)

[5.2 Low Level Diagram (if applicable) 13](#_Toc139702817)

[5.3 Interfaces (if applicable) 13](#_Toc139702818)

[6 Performance Test 14](#_Toc139702819)

[6.1 Test Plan/ Test Cases 14](#_Toc139702820)

[6.2 Test Procedure 14](#_Toc139702821)

[6.3 Performance Outcome 14](#_Toc139702822)

[7 My learnings 15](#_Toc139702823)

[8 Future work scope 16](#_Toc139702824)

# Preface

# Summary of 6 weeks Reports:

Week 1: Requirements and Planning

Defined game rules, identified necessary components, and planned the user interface.

Week 2: Game Structure Design

Created a flowchart to visualize different game states and planned functions and data structures.

Week 3: Core Gameplay Implementation

Implemented player input, random computer selection, and game outcome determination.

Week 4: User Experience Enhancement

Added features such as score display, instructions, and input validation.

Week 5: Advanced Features

Incorporated a scoring system and options for different game modes or difficulty levels.

Week 6: Testing and Refinement

Conducted thorough testing, gathered feedback, and made necessary refinements for a polished game.

# About need of relevant Internship in career development :

Relevant internships are crucial for career development as they provide practical experience and exposure to real-world work environments. They allow individuals to apply their knowledge and skills in a professional setting, bridging the gap between academia and industry requirements. Internships offer the opportunity to explore and understand a chosen field or industry in greater depth, learning from established professionals and gaining insights into industry practices and trends.

# Brief about My project/problem statement:

The project "Rock Paper Scissors" in Python aims to create a game that allows users to play the classic Rock Paper Scissors game against the computer. The problem statement involves designing and implementing the game logic, user interface, and interaction flow.

Opportunity given by USC/UCT:

The Upskill campus and Uniconverge Technologies gave me a great opportunity to show case my special talents . I would like to thank them for providing such a great internship.

How Program was planned



Thanks to all , who have helped you directly or indirectly.

I would like to suggest you guys to focus on your career rather than other activities.

# Introduction

## About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various**Cutting Edge Technologies e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end**etc.



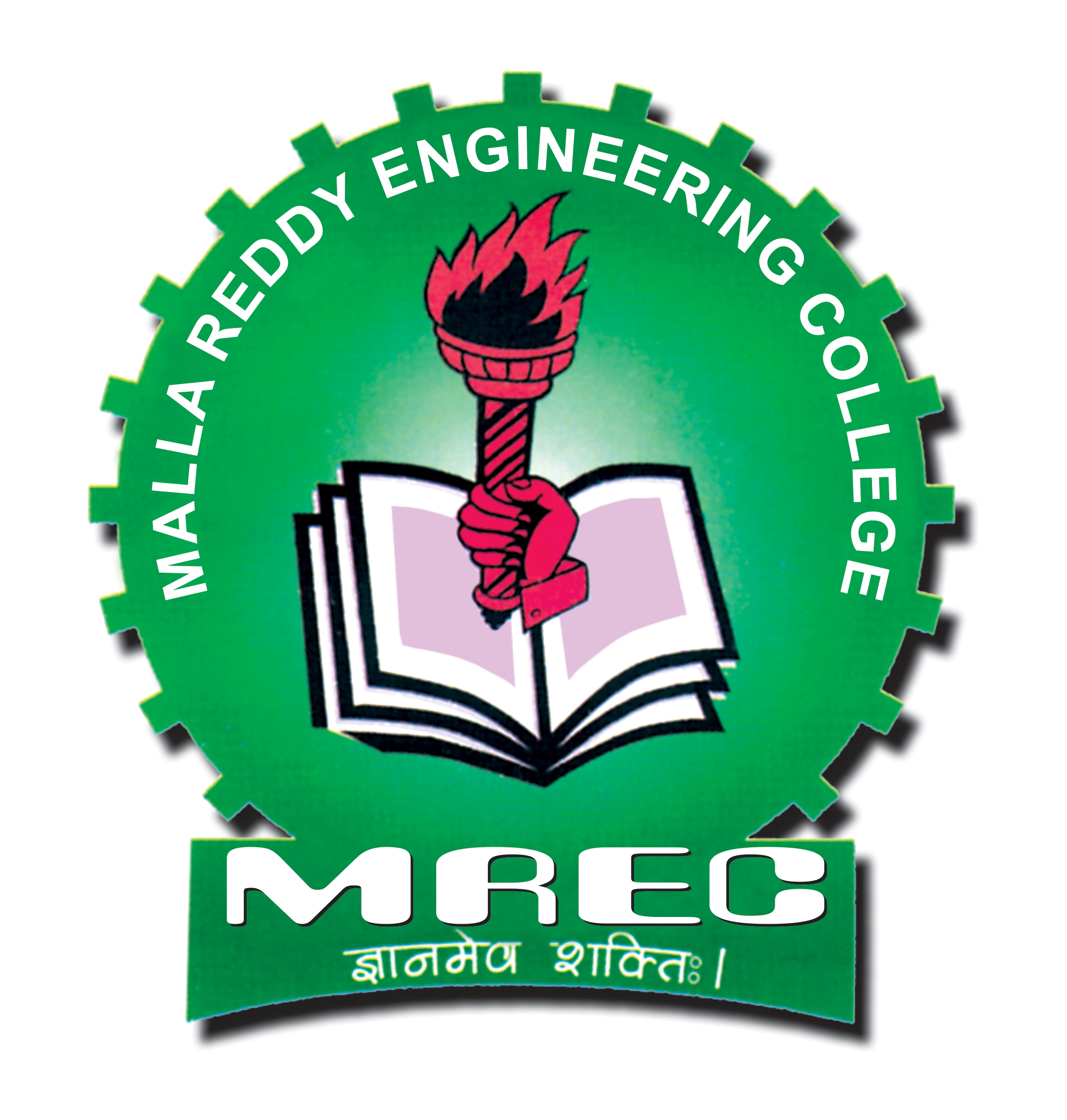
# UCT IoT Platform **(****)**

**UCT Insight** is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

* It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
* It supports both cloud and on-premises deployments.

It has features to  
• Build Your own dashboard  
• Analytics and Reporting  
• Alert and Notification  
• Integration with third party application(Power BI, SAP, ERP)  
• Rule Engine



# Smart Factory Platform ()

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

* with a scalable solution for their Production and asset monitoring
* OEE and predictive maintenance solution scaling up to digital twin for your assets.
* to unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
* A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

# What is LoRaWAN? - Yeastar Workplace Help based Solution

UCT is one of the early adopters of LoRAWAN teschnology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

# Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



## About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

<https://www.upskillcampus.com/>

upSkill Campus aiming to upskill 1 million learners in next 5 year



## The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

## Objectives of this Internship program

The objective for this internship program was to

 ☛ get practical experience of working in the industry.

 ☛ to solve real world problems.

 ☛ to have improved job prospects.

 ☛ to have Improved understanding of our field and its applications.

 ☛ to have Personal growth like better communication and problem solving.

## Reference

[1]Python official Documentation (https://docs.python.org.comx )

[2] Python package index (https://py.pi.org.com)

[3] Stack overflow (https://stackoverflow.com)

## Glossary

|  |  |
| --- | --- |
| Terms | Acronym |
| Rock Paper Scissors | GUI |
| User Interface | CLI |
| Random number generation | API |
| Game outcome Determination | MVP |
| Conditional statements | IDE |

# Problem Statement

The problem statement for the "Rock Paper Scissors" Python project is to create a program that allows users to play the classic game of Rock Paper Scissors against the computer. The program should have the following functionalities:

* + - 1. User Input: The program should prompt the user to input their choice of either rock, paper, or scissors.

* + - 1. Computer Selection: The program should generate a random selection for the computer opponent.

* + - 1. Outcome Determination: Based on the choices made by the user and the computer opponent, the program should determine the outcome of the round according to the rules of the game: rock beats scissors, scissors beat paper, and paper beats rock.

* + - 1. Score Tracking: The program should keep track of the number of wins, losses, and ties for the user against the computer opponent.

* + - 1. Repeat Gameplay: The program should allow the user to play multiple rounds of the game until they choose to quit.

* + - 1. Error Handling: The program should handle invalid user inputs, such as incorrect spellings or choices outside the given options and provide appropriate error messages or instructions.

* + - 1. The goal of the project is to create an interactive and functional game of Rock Paper Scissors that provides an enjoyable experience for the user while implementing the necessary logic and functionalities to handle the game rules and outcomes accurately.



# Existing and Proposed solution

Existing solution :

The existing solution for the "Rock Paper Scissors" game might involve a manual implementation of the game logic using conditional statements and user input through the command line. This solution would require writing code to handle user input validation, generating random computer selections, determining the outcome, and keeping track of the score. However, the specific implementation details would depend on the developer's approach.

## Proposed Solution:

A proposed solution for the "Rock Paper Scissors" Python project could involve the following steps:

Design the Game Structure: Plan the overall structure of the game, including the user interface and flow of interactions.

Implement User Input: Prompt the user to input their choice (rock, paper, or scissors) using a command-line interface or a graphical user interface (GUI).

Generate Computer Selection: Use a random number generator or random choice function to generate a random selection for the computer opponent.

Determine the Outcome: Compare the choices made by the user and the computer opponent using conditional statements to determine the outcome (win, loss, or tie) based on the game rules.

Track the Score: Implement a scoring mechanism to keep track of the user's wins, losses, and ties against the computer opponent.

Allow Multiple Rounds and Quit Option: Create a loop that allows the user to play multiple rounds of the game until they choose to quit. Provide an option for the user to exit the game at any time.

Handle Error Cases: Implement error handling mechanisms to validate user input, handle unexpected situations, and provide appropriate error messages or instructions.

It is important to note that the specific implementation details and code structure may vary based on the developer's preference and the chosen user interface.

## Code submission (Github link)

* 1. Code submission link : <https://github.com/sonyyy15/rps-code/blob/08e188da0110aa7955f4b180d1b4b3f6f93c46ad/rps.txt>
  2. Code:

import random

def get\_user\_choice():

while True:

choice = input("Enter your choice (rock/paper/scissors): ").lower()

if choice in ['rock', 'paper', 'scissors']:

return choice

else:

print("Invalid choice. Please try again.")

def determine\_winner(user\_choice, computer\_choice):

if user\_choice == computer\_choice:

return "It's a tie!"

elif (

(user\_choice == 'rock' and computer\_choice == 'scissors') or

(user\_choice == 'paper' and computer\_choice == 'rock') or

(user\_choice == 'scissors' and computer\_choice == 'paper')

):

return "You win!"

else:

return "Computer wins!"

def play\_game():

print("Let's play Rock-Paper-Scissors!")

user\_choice = get\_user\_choice()

computer\_choice = random.choice(['rock', 'paper', 'scissors'])

print("You chose:", user\_choice)

print("Computer chose:", computer\_choice)

print(determine\_winner(user\_choice, computer\_choice))

play\_game()

# Proposed Design/ Model :

Here is the model :

Game Class: Create a class to represent the game. This class will have methods to handle game functionalities.

User Input: Implement a method to prompt the user for their choice of rock, paper, or scissors and validate the input.

Computer Selection: Create a method to randomly generate the computer opponent's selection.

Outcome Determination: Implement a method to compare the user's choice and the computer's selection to determine the outcome of each round based on the game rules.

Score Tracking: Add attributes to the Game class to keep track of the user's wins, losses, and ties against the computer opponent.

Gameplay Loop: Create a loop that allows the user to play multiple rounds until they choose to quit. Display the result of each round and update the score accordingly.

Error Handling: Implement error handling mechanisms to handle invalid user input or unexpected scenarios gracefully. Display appropriate error messages or instructions to guide the user.

User Interface: Depending on the chosen user interface, implement methods to display prompts, results, and score updates. This can be through a command-line interface (CLI) or a graphical user interface (GUI) using libraries like Tkinter.

Main Function: Created a main function to initialize the game, handle the gameplay loop, and provide an option for the user to quit.

By designing the project in this manner, I can encapsulate the game functionalities within a Game class, making the code modular and easier to maintain. This design allows for flexibility in the choice of user interface and facilitates the separation of concerns, making the code more organized and readable.

## High Level Diagram :

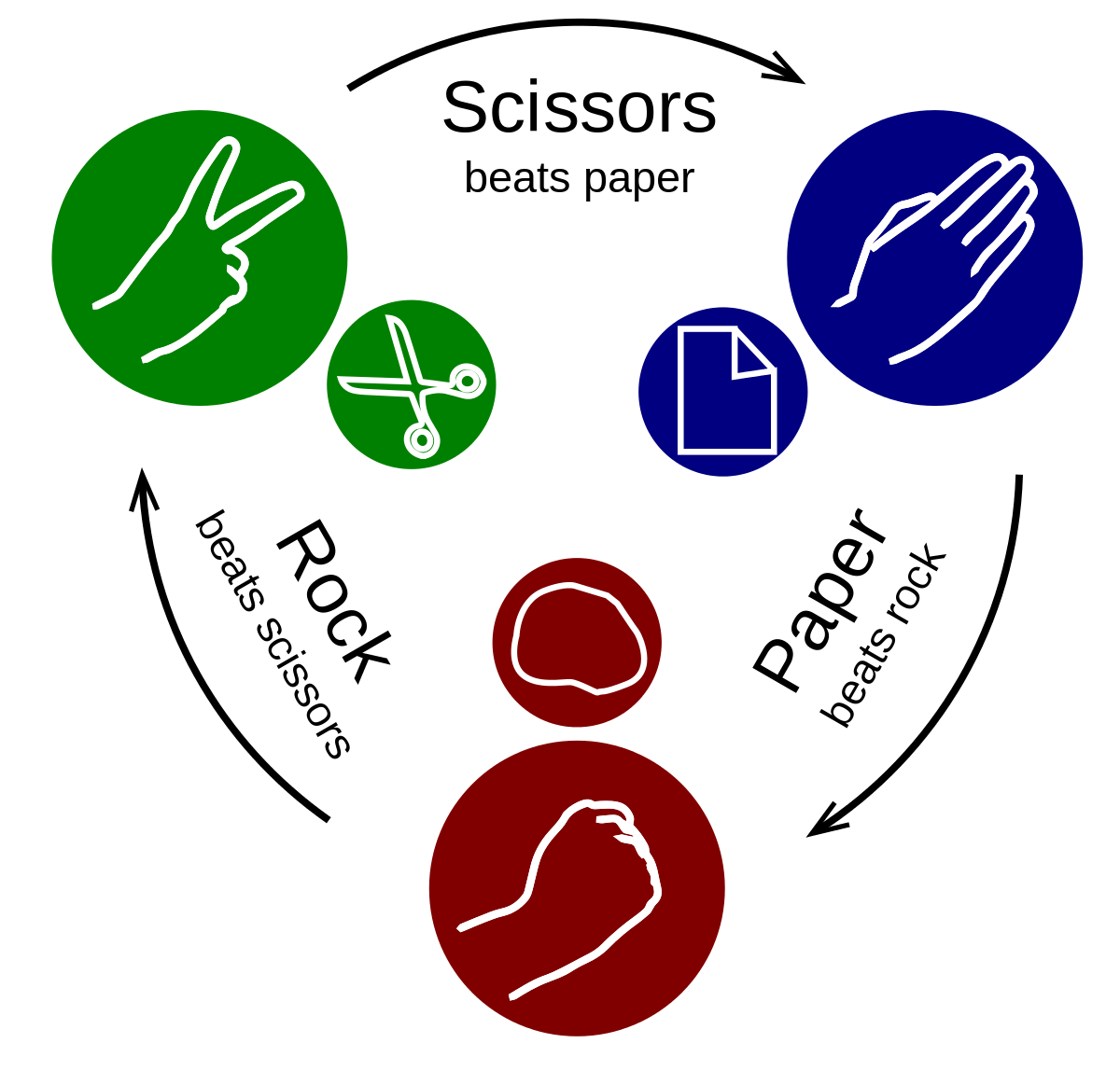
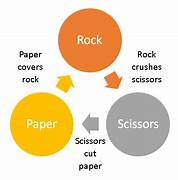


Figure 1: HIGH LEVEL DIAGRAM OF THE SYSTEM

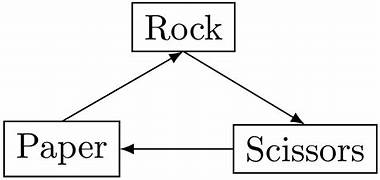
* 1. Low Level Diagram:



## Interfaces (if applicable)

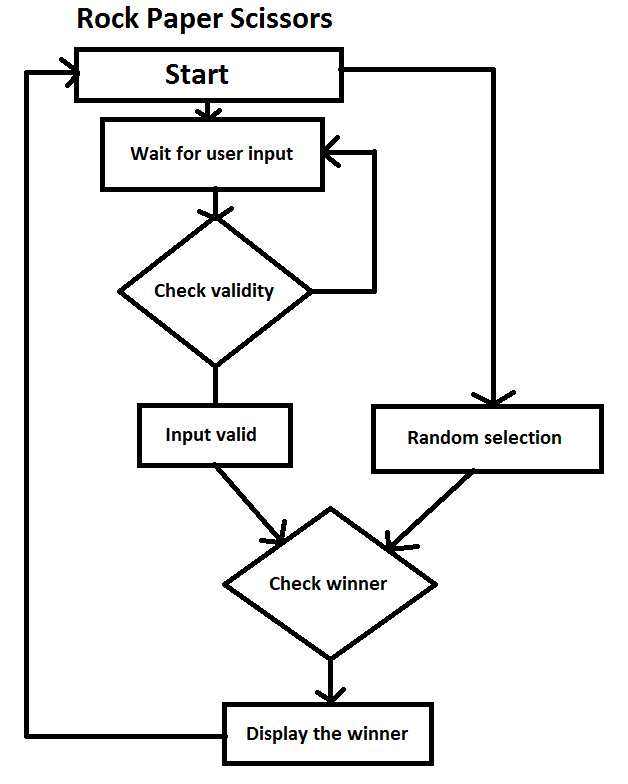
Update with Block Diagrams, Data flow, protocols, FLOW Charts, State Machines, Memory Buffer Management.

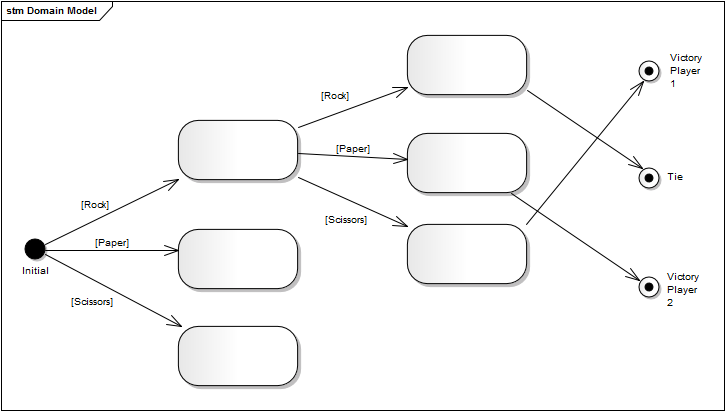
# Block Diagram:



# Flow Chart :

# 





# Memory Buffer Management System:

In this , memory buffer management may not be a significant concern. The game typically involves real-time interaction and does not require extensive memory allocation or management. However, here are a few considerations related to memory buffer management that can be taken into account.

Data Storage: If you choose to implement features like score tracking or game history, you may need to allocate memory to store this data. You can use data structures like lists or dictionaries to manage and store the necessary information.

Memory Optimization: While the game itself may not have significant memory requirements, it's always a good practice to optimize memory usage. Avoid unnecessary duplication of data and ensure efficient memory utilization by freeing up any unused memory resources when appropriate.

Resource Deallocation: If you allocate any resources during the game, such as opening files or establishing network connections, make sure to release those resources properly to avoid memory leaks and ensure efficient memory management.

Performance Considerations: While playing the game, monitor the memory usage and performance to identify any bottlenecks or excessive memory consumption. Optimize the code or data structures if needed to improve efficiency and minimize memory usage.

It's important to note that memory buffer management becomes more critical in complex applications with larger data sets or when dealing with resource-intensive operations. However, for the relatively simple and lightweight nature of the "Rock Paper Scissors" game, the memory management requirements are generally minimal.

# Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Scalability: Real industry projects often need to handle a large user base and high volumes of data. Performance testing helps identify any scalability issues, such as slow response times or resource limitations, ensuring that the system can handle the expected workload.

User Experience: In real industry projects, providing a smooth and responsive user experience is crucial. Performance testing helps uncover any bottlenecks or inefficiencies that could impact the user experience, enabling optimization and improvement of system responsiveness.

Production Environment: Real industry projects are deployed in production environments where multiple systems and components interact. Performance testing helps simulate and evaluate the system's behavior under realistic conditions, including various user loads, network connectivity, and hardware configurations.

Compliance and SLAs: Industries often have specific compliance requirements or service level agreements (SLAs) to meet. Performance testing helps ensure that the system can meet these requirements, such as response time thresholds or maximum downtime limits.

# Constraints :

Constraints can be e.g. memory, MIPS (speed, operations per second), accuracy, durability, power consumption etc.

* + - 1. System Resources : The game should be designed to utilize system resources efficiently, considering limitations such as CPU usage, memory usage, and disk I/O.

* + - 1. Response Time: The game should provide a responsive user experience with minimal delay between user input and system response. Performance testing can help identify any bottlenecks that may cause significant delays and optimize the system for faster response times.

* + - 1. Scalability: While the "Rock Paper Scissors" game may not have extensive scalability requirements, it should still be designed to handle multiple users concurrently without performance degradation. The system should be tested to ensure it can handle a reasonable number of simultaneous game sessions.

* + - 1. User Load: The system should be able to handle varying user loads, from low to peak usage periods. Performance testing can help determine the system's performance under different load scenarios and ensure it can handle increased traffic without issues.

* + - 1. Compatibility: The game should be compatible with different platforms, operating systems, and versions of Python. Testing should be performed to ensure smooth operation across various environments and configurations.

Considering these constraints and conducting performance testing can help ensure the "Rock Paper Scissors" game meets industry standards, delivers a seamless user experience, and performs optimally in real-world scenarios.

## Test Plan/ Test Cases:

User Input Testing: Validate user input for different scenarios, including valid choices (rock, paper, scissors) and invalid inputs. Verify that the system handles incorrect or unexpected input gracefully with appropriate error messages.

Computer Selection Testing: Verify that the computer opponent's selection is randomly generated and covers all possible choices (rock, paper, scissors).

Outcome Determination Testing: Test the game outcome determination logic by simulating various combinations of user and computer choices. Ensure that the game correctly identifies wins, losses, and ties based on the established rules.

Score Tracking Testing: Validate that the score tracking mechanism accurately updates the user's wins, losses, and ties as per the game outcomes. Test the score display functionality.

## Test Procedure:

### Input Testing:

a. Test valid input: Enter valid choices (rock, paper, scissors) and verify that the game accepts them without errors.

b. Test invalid input: Enter invalid choices or unexpected inputs (e.g., numbers, symbols) and verify that the game handles them with appropriate error messages.

### Computer Selection Testing:

a. Verify randomness: Play multiple rounds and observe that the computer opponent's selections are random and not predictable.

b. Test coverage: Ensure that the computer opponent selects all possible choices (rock, paper, scissors) over a significant number of rounds.

### Outcome Determination Testing:

a. Test win scenarios: Simulate different combinations of choices (rock vs. scissors, paper vs. rock, scissors vs. paper) and verify that the game correctly identifies the user as the winner.

b. Test loss scenarios: Simulate different combinations of choices and verify that the game correctly identifies the user as the loser.

c. Test tie scenarios: Simulate identical choices (rock vs. rock, paper vs. paper, scissors vs. scissors) and verify that the game correctly identifies a tie.

### Score Tracking Testing:

a. Verify score update: Play multiple rounds with different outcomes and ensure that the game accurately updates the user's wins, losses, and ties.

b. Test score display: Check that the game displays the correct score after each round and maintains the score throughout the gameplay.

### Game Loop Testing:

a. Test multiple rounds: Play several rounds in succession and verify that the game continues to prompt for user input and display results until the user chooses to quit.

b. Test exit option: Choose to quit the game during different stages and verify that the game exits gracefully without errors.

## Performance Outcome :

#### Responsiveness:

The game should exhibit quick response times, providing immediate feedback to the user after making their choice. A well-implemented project should have minimal lag or delay in displaying the results.

#### Scalability:

The game should be able to handle increasing user loads without significant performance degradation. Performance testing can help identify any bottlenecks or resource limitations that may impact scalability.

#### Resource Usage:

The game should utilize system resources efficiently, avoiding excessive CPU usage or memory consumption. A well-optimized project should operate without straining the system and ensure smooth gameplay.

#### Stability:

The game should be stable and not crash or encounter errors during normal gameplay. Thorough testing and error handling mechanisms can help ensure stability and minimize unexpected issues.

#### Compatibility:

The game should function consistently across different platforms and Python versions. Compatibility testing can help identify any compatibility issues and ensure smooth operation in various environments.

#### Network Performance:

If the game involves network communication, such as online multiplayer functionality, the performance outcome will depend on factors like network latency, bandwidth, and server response.

# My learnings:

In the process of working on the "Rock Paper Scissors" Python project, I have gained several valuable learnings. Here are some potential learnings from this project:

Game Logic Implementation: You have learned how to design and implement the game logic for a simple game like Rock Paper Scissors. This includes handling user input, generating random computer selections, determining the game outcome, and tracking the score.

User Interface Development: Depending on the chosen interface (command-line or graphical), you may have learned how to create user-friendly prompts, display game results, and handle user interactions effectively.

Error Handling and Validation: By incorporating error handling mechanisms, you have learned how to validate user input, handle unexpected scenarios gracefully, and provide informative error messages.

Testing and Debugging: Through the process of testing, you have gained experience in designing and executing test cases to ensure the correct functionality of the game.

# Future work scope

### Areas for future work and scope:

### GUI Implementation:

If the project was initially developed with a command-line interface (CLI), you could consider expanding it to include a graphical user interface (GUI) using libraries like Tkinter or Pygame. This would provide a more visually appealing and interactive experience for users.

### Multiplayer Functionality:

Extend the game to include multiplayer capabilities, allowing users to play against each other over a network. This could involve implementing networking protocols, setting up a server-client architecture, and enabling online matchmaking.

### AI Opponent:

Enhance the game by adding an AI opponent that can adapt its strategy based on the user's choices. Implementing different AI algorithms, such as rule-based or machine learning-based approaches, would provide a more challenging and dynamic gameplay experience.

### Game Variations:

Introduce additional variations of the Rock Paper Scissors game, such as Rock Paper Scissors Lizard Spock or other regional variations. This would add diversity and keep the game engaging for users.

### Leaderboard and Achievements:

Implement a leaderboard feature that tracks high scores and achievements. This would encourage competition among players and provide a sense of accomplishment for achieving specific milestones.

### Sound and Visual Effects:

Enhance the game's audio and visual experience by adding sound effects, animations, and graphics. This would further engage users and make the game more immersive.

### Mobile App Development:

Consider developing a mobile app version of the game for iOS and Android platforms. This would allow users to play the game on their smartphones and tablets, expanding the reach and accessibility of the project.