

AMRITSAR GROUP OF COLLEGES

Autonomous Status Conferred by UGC | NAAC – A Grade

Artificial Intelligence Project Report on “Rock Paper Scissor”

Submitted in the partial fulfilment of the requirement of the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

Batch

(2020-2024)



Submitted to

Er. Anamika Jain
Assistant Professor
Department of CSE

Submitted by

Shivam Raj (2000200)
Siddharth Kumar Sonu (2000206)
Sumit Kumar Giri (2000213)
Vishal Kumar (2000233)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ABSTRACT

As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the process of converting speech into text. This is commonly used in voice assistants like Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. In the current scenario, advancement in technologies is such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, we realized that the concept of AI in every field is decreasing human effort and saving time.

ACKNOWLEDGEMENT

It is our proud privilege to release the feelings of our gratitude to several persons who helped us directly or indirectly to conduct this **Rock, paper & Scissor**. We express our heart full thanks and owe a deep sense of gratitude to our teacher **Er. Anamika Jain, Assistant Professor, Amritsar Group of Colleges**, for her guidance and inspiration in completing this project.

We are extremely thankful to the **Dr. Sandeep Kad, Head of Department** and all faculty members of CSE Department at **Amritsar Group of Colleges** for their co-ordination and co-operation and for their kind guidance and encouragement.

We also thank all our friends who have more or less contributed to the preparation of this Project Report, we will be always indebted to them.

The study has indeed helped us explore more knowledge avenues related to Artificial Intelligence Project and we are sure it will help us in future too.

DECLARATION

We **Shivam Raj, Sidharth Kumar Sonu, Sumit Kumar Giri** and **Vishal Kumar** hereby as a team declare that the project work entitled “**Rock Paper & Scissor**” is an authentic record of our own work carried out as per the requirements of **Artificial Intelligence Lab (ACCS-16702)** for the award of degree of the **B.Tech (CSE), Amritsar Group of Colleges, Amritsar**, under the guidance of **Dr. Sandeep Kad** (Head of the Department).

(Signature of students)

Shivam Raj (2000200)

Sidharth Kumar Sonu (2000206)

Sumit Kumar Giri (2000213)

Vishal Kumar (2000233)

TABLE OF CONTENTS

SR. NO.	CONTENT	PAGE NO.
1.	Abstract	ii
2.	Acknowledgement	iii
3.	Declaration	iv
4.	List of Figures	vi
5.	Introduction to Jarvis	1
6.	Objective	2
7.	Formulation of Problem	3
8.	Hardware and Software Requirements	4
9.	Technologies Used	5-6
10.	Tools Used	7
11.	Python Libraries	8
12.	Implementation Code of project	9-17
13.	Snapshots	18-19
14.	Conclusion	20
15.	Future Scope	21
16.	Bibliography	22

LIST OF FIGURES

FIG NO.	FIGURE NAME	PAGE NO.
1.	Imported Modules	5
11.	Task 1	18
13.	Task 2	18
15.	Task 3	19
17.	Closing Window	19

INTRODUCTION

The Rock, Paper, Scissors computer project is an interactive digital implementation of the classic hand game, designed to be played against a computer opponent. This project combines programming logic, user interaction, and basic artificial intelligence to create an engaging gaming experience. In Rock, Paper, Scissors, players simultaneously form one of three shapes (rock, paper, or scissors) with their hands, and the winner is determined based on the rules: rock defeats scissors, scissors defeat paper, and paper defeats rock. This project aims to replicate this traditional game in a digital format, enabling users to test their wits against a computerized opponent.

The primary objective of this project is to develop a Python-based Rock, Paper, Scissors game with a computer opponent that simulates human-like decision-making. The program will allow users to input their choice, generate a random choice for the computer, and then determine the winner based on the established game rules. This project serves as an opportunity to practice programming concepts such as conditional statements, loops, and basic artificial intelligence. The classic game of Rock, Paper, Scissors is a simple yet timeless pastime that has been enjoyed by people of all ages for generations. In a digital age, the scope of a Rock, Paper, Scissors project has evolved to offer a wide array of possibilities, combining the traditional fun of the game with the capabilities of technology. This project's scope can range from a basic implementation to more advanced and innovative features, catering to diverse audiences and purposes.

The scope of a Rock, Paper, Scissors project encompasses the design, development, and potential expansion of a digital version of the game. While the core concept remains straightforward—players choose between three hand gestures representing rock, paper, and scissors—the project can be enriched and customized to meet various objectives and user preferences.

OBJECTIVES

The key objectives of Rock, paper & Scissor are as follows:

Entertainment: The primary purpose of the game is to entertain players. It offers a fun and engaging way to pass the time.

Simplicity: The game is intentionally simple and easy to understand. It doesn't require any special equipment or extensive rules, making it accessible to a wide range of players.

Decision-Making: The game encourages players to make quick decisions based on the available options (Rock, Paper, Scissors). This can help improve decision-making skills.

Randomness and Chance: The random nature of the computer's choice introduces an element of unpredictability, adding excitement and anticipation to each round.

Fair Competition: The game is designed to be fair, as both the player and the computer have an equal chance of winning, losing, or tying. The outcome is not influenced by external factors

Strategy (Optional): While the game is inherently simple, players can choose to implement strategies based on their observations of the opponent's choices.

Learning Programming (For Developers): For those learning programming, creating a Rock, Paper, Scissors game provides an opportunity to practice fundamental concepts such as conditional statements, user input, and randomization.

Basic AI (Optional): Implementing a computer opponent with basic decision-making logic can add an extra layer of challenge for players.

Memory and Pattern Recognition (Optional): Provide entertainment options, such as music and video playback, jokes, and games, to enhance user experience during downtime.

Social Interaction (Multiplayer Versions): In multiplayer versions, the game can serve as a social activity, allowing players to compete against each other and interact in a light-hearted manner.

FORMULATION OF PROBLEM

Objective: Develop a program that simulates the Rock, Paper, Scissors game. The program should allow a user to play against a computer opponent. The game will follow the traditional rules: Rock beats Scissors, Scissors beat Paper, and Paper beats Rock.

Input: User's choice (Rock, Paper, or Scissors)

Output: Result of the round (Win, Lose, or Tie)

Game Rules: Rock beats Scissors, Scissors beat Paper, Paper beats Rock

Player vs. Computer Interaction: The user will input their choice through a graphical user interface (GUI) or command-line interface (CLI) The computer will generate a random choice.

Game Logic: The program will determine the winner based on the user's choice and the computer's choice. If both choices are the same, the round is a tie. Otherwise, the winner is determined by the game rules.

Randomization: The computer's choice will be generated randomly using a random number generator.

Error Handling: The program should handle invalid inputs by providing appropriate feedback to the user and allowing them to re-enter a valid choice.

Additional Features (Optional): Implement a scoring system to keep track of the user's wins, losses, and ties. Create a graphical user interface (GUI) to enhance user experience. Add a basic AI to the computer opponent for more challenging gameplay. Allow the user to play multiple rounds and display the overall winner

Constraints: The program should be user-friendly and intuitive, ensuring that players can easily input their choice and understand the game outcome.

Testing: Conduct thorough testing to ensure the program accurately follows the game rules and provides correct outcomes for different inputs.

Deployment (Optional): If desired, the program can be deployed as a standalone application, web-based game, or integrated into a larger software system. This would involve considerations for platform compatibility and deployment strategies

HARDWARE & SOFTWARE REQUIREMENTS

Hardware Requirements

Minimum Hardware Requirements:

- **Processor:** Dual-core processor (e.g., Intel Core i3 or AMD equivalent)
- **Memory (RAM):** 2 GB
- **Storage:** 64 GB HDD
- **Graphics:** Basic integrated graphics card

Preferred Hardware Requirements:

- **Processor:** Quad-core processor (e.g., Intel Core i5 or AMD equivalent)
- **Memory (RAM):** 4 GB or higher • **Storage:** 128 GB SSD or HDD
- **Graphics:** Integrated graphics card with DirectX 11 support

Software Requirements

- **Operating System:** Windows/macOS/Linux
- **Code Editor:** Visual Studio Code/Sublime Text
- **Web Browser:** Chrome/Firefox
- **API:** Times of India API

TECHNOLOGIES USED

PYTHON



Python is a versatile and widely-used programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python's popularity has grown steadily over the years, making it a favourite choice for beginners and experienced developers alike.

Here are some key features of Python:

- 1.Easy to Read and Write:** Python's clean and straightforward syntax emphasizes readability and reduces the cost of program maintenance. This makes it an excellent choice for those new to programming.
- 2.High-Level Language:** Python is a high-level language, which means it abstracts low-level operations, making it more user-friendly and efficient for developers.
- 3.Interpreted Language:** Python is an interpreted language, which means you can execute code directly without the need for compilation. This results in faster development cycles.
- 4.Dynamic Typing:** Python uses dynamic typing, allowing variables to change types as needed, making it more flexible and adaptable.
- 5.Rich Standard Library:** Python's standard library provides a vast collection of modules and functions that simplify various tasks, from file handling to web development.

Python's adaptability and rich ecosystem make it suitable for a wide range of applications, from web development to scientific research. Its user-friendly syntax and strong community support continue to make it a top choice for developers worldwide.

PYTHON TKINTER

Tkinter: A Python Library for GUI Development

Tkinter is a standard GUI (Graphical User Interface) library for Python. It provides a set of tools and widgets that allow developers to create desktop applications with graphical interfaces. Tkinter is simple to use and comes bundled with most Python installations, making it one of the most accessible choices for building GUI applications.

Here are some of its features:

Cross-Platform Compatibility: Tkinter works on various platforms, including Windows, macOS, and Linux, ensuring that your GUI applications can be used across different operating systems.

Standard Library Inclusion: Tkinter is included as part of the Python standard library, so there's no need to install additional packages or dependencies.

User-Friendly Widgets: Tkinter provides a wide range of built-in widgets, such as buttons, labels, text entry fields, radio buttons, and more. These widgets simplify the process of creating user interfaces.

Customization: You can customize the appearance of widgets and windows by specifying attributes like colors, fonts, and sizes. This flexibility allows you to create visually appealing and unique interfaces.

Event Handling: Tkinter supports event-driven programming, which means you can define how your application responds to user interactions like button clicks or keyboard input.

Geometry Management: Tkinter uses a grid-based geometry manager that makes it easy to arrange widgets within windows, ensuring they display correctly.

Dialog Boxes: Tkinter provides pre-built dialog boxes for common tasks like file selection, message confirmation, and input.

Integration with Other Python Libraries: You can combine Tkinter with other Python libraries like Matplotlib for data visualization or NumPy for scientific applications.

Theming: You can create and apply themes to your Tkinter applications for a consistent and professional look.

TOOLS USED

VISUAL STUDIO CODE



Visual Studio Code (VS Code) is a free, open-source, and lightweight code editor developed by Microsoft. It is one of the most popular code editors among developers and is available on Windows, Mac, and Linux. VS Code has gained popularity due to its versatility, a wide range of features, and extensive support for programming languages and frameworks. In this article, we will explore some of the key features of VS Code and how it can improve your coding experience. VS Code has a simple, clean, and intuitive user interface. It has a file explorer pane on the left, a text editor pane in the middle, and a sidebar on the right that displays different types of information, such as debugging information, version control information, and extensions.

Here are the key features of Visual Studio Code :

Cross-Platform: VS Code is available on Windows, macOS, and Linux, ensuring a consistent experience across different operating systems.

Free and Open Source: It's free to use and open-source, allowing the community to contribute and create extensions.

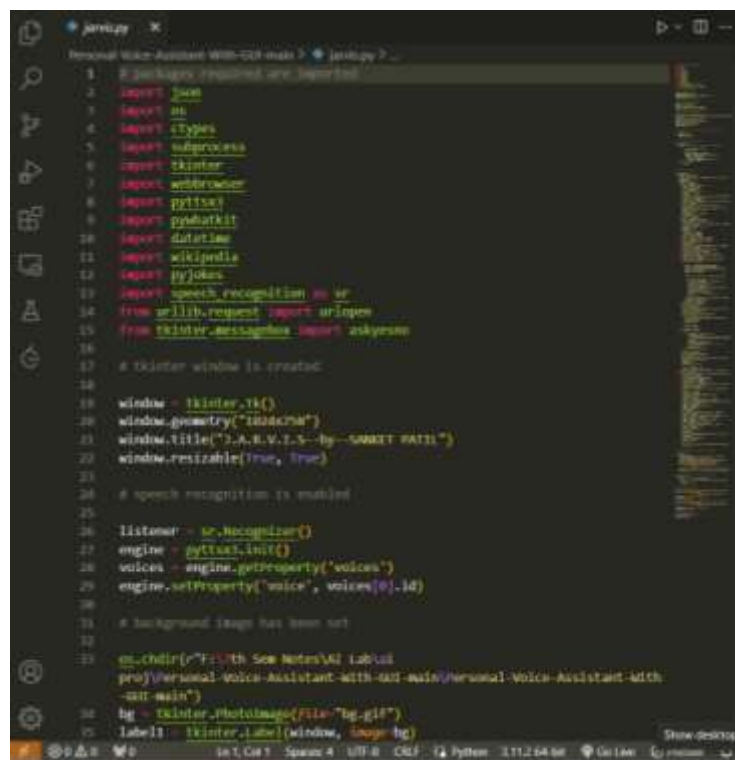
Intelligent Code Completion: VS Code provides intelligent code suggestions and completions, improving coding speed and accuracy.

Versatile Language Support: It supports a wide range of programming languages and offers syntax highlighting, autocompletion, and debugging capabilities for many of them.

Extensions and Marketplace: A vast library of extensions is available through the Visual Studio Code Marketplace, allowing you to customize and extend its functionality.

PYTHON LIBRARIES

- ❖ In Rock Paper & Scissor following python libraries were used:
- ❖ CV2: it is a Python Library which provides a wide range of tools and functions for tasks like image and video processing, object detection, facial recognition, and more.
- ❖ Random: The random library in Python is a standard library module that provides functions for generating random numbers.
- ❖ Time: The time library in Python is a standard module that provides various functions for working with time-related operations. It allows you to measure time intervals, handle time conversions, and deal with time-related calculations.
- ❖ OS: The OS library in Python provides functions to interact with the operating system, allowing tasks like file operations and system commands



```
1 # Packages Required are Imported
2 import json
3 import os
4 import ctypes
5 import subprocess
6 import tkinter
7 import webbrowser
8 import pyttsx3
9 import pywhatkit
10 import datetime
11 import wikipedia
12 import pyjokes
13 import speech_recognition as sr
14 from urllib.request import urlopen
15 from tkinter.messagebox import askyesno
16
17 # A tkinter window is created
18
19 window = tkinter.Tk()
20 window.geometry("1024x768")
21 window.title("D.A.R.V.I.S--by-SMART PATIL")
22 window.resizable(True, True)
23
24 # A speech recognition is enabled
25
26 listener = sr.Recognizer()
27 engine = pyttsx3.init()
28 voices = engine.getProperty('voices')
29 engine.setProperty('voice', voices[0].id)
30
31 # A background image has been set
32
33 os.chdir(r"C:\Python\Notes\AI Lab\ai
34 proj\Personal-Voice-Assistant-with-GAI-main\Personal-Voice-Assistant-with-
35 -GAI-main")
36 bg = tkinter.PhotoImage(file="bg.gif")
37 label1 = tkinter.Label(window, image=bg)
```

Fig. No.1: Imported Modules

IMPLEMENTATION

CODE

Main.py

```
import cv2
import time
import os
import HandTrackingModule as htm
import random

wCam, hCam = 640, 480
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)

detector = htm.handDetector(detectionCon=0.75)

tipIds = [4, 8, 12, 16, 20]

timeUp = 5

folderPath = "ExitAndStartImages"
myListExitAndStart = os.listdir(folderPath)
ExitAndStartImagesList = []
for imPath in myListExitAndStart:
    image = cv2.imread(f'{folderPath}/{imPath}')
    ExitAndStartImagesList.append(image)
hExit, wExit, cExit = ExitAndStartImagesList[0].shape
hStart, wStart, cStart = ExitAndStartImagesList[1].shape

folderPath = "PRSIImages"
myListPRS = os.listdir(folderPath)

PRSIImagesList = []
for imPath in myListPRS:
```

```

image = cv2.imread(f'{folderPath}/{imPath}')
PRSIImagesList.append(image)
hPaper, wPaper, cPaper = PRSIImagesList[0].shape
hRock, wRock, cRock = PRSIImagesList[1].shape
hScissors, wScissors, cScissors = PRSIImagesList[2].shape
hSpace, wSpace = 50, 30

"""
Captures a frame from the camera, and returns this frame and the hands' landmarks that's in this frame
"""

def captureLandmarksInFrame():
    success, img = cap.read() # read the frame
    img = cv2.flip(img, 1) # mirror the frame
    img, num_hands_detected = detector.findHands(img, draw=False) # find the hands
    landmarksList = detector.findPosition(img, draw=False) # create the hands' landmarks
    return img, num_hands_detected, landmarksList

"""
When the user puts his hand in front of the camera, analyses his choice:
paper = 1
rock = 2
scissors = 3
none of the above = 0
"""

def analyseUserChoice(landmarksList):
    fingers = []
    # Thumb
    if landmarksList[tipIds[0]][1] < landmarksList[tipIds[4] - 1][1]: # right hand
        if landmarksList[tipIds[0]][1] < landmarksList[tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)
    else: # left hand
        if landmarksList[tipIds[0]][1] > landmarksList[tipIds[0] - 1][1]:
            fingers.append(1)

```



```

    else:
        fingers.append(0)
# 4 Fingers
for id in range(1, 5):
    if landmarksList[tipIds[id]][2] < landmarksList[tipIds[id] - 2][2]:
        fingers.append(1)
    else:
        fingers.append(0)
totalFingers = fingers.count(1) # the number of fingers

if totalFingers == 5: # paper
    return 1
elif totalFingers == 0: # rock
    return 2
elif totalFingers == 2 and fingers[1] == 1 and fingers[2] == 1: # scissors
    return 3
return 0

"""
Gets the user's choice and shows the suitable image on the upper left corner of the screen
"""
def showUserChoice(img, userChoice):
    if userChoice == 1: # paper
        cv2.putText(img, "USER", (wSpace + 30, hSpace - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255,
193, 43), 2)
        img[hSpace:hSpace + hPaper, wSpace:wSpace + wPaper] = PRSImagesList[0]
    elif userChoice == 2: # rock
        cv2.putText(img, "USER", (wSpace + 30, hSpace - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (186,
108, 255), 2)
        img[hSpace:hSpace + hRock, wSpace:wSpace + wRock] = PRSImagesList[1]
    elif userChoice == 3: # scissors
        cv2.putText(img, "USER", (wSpace + 30, hSpace - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (57,
229, 255), 2)
        img[hSpace:hSpace + hScissors, wSpace:wSpace + wScissors] = PRSImagesList[2]

"""

```

Ruffles the computer's choice:

```
paper = 1
rock = 2
scissors = 3
```

And shows the suitable image on the upper right corner of the screen

```
"""
def ruffleAndShowComputerChoice(img):
    computerChoice = random.randint(1, 3)

    if computerChoice == 1: # paper
        cv2.putText(img, "COMPUTER", (wCam - wPaper - wSpace - 5, hSpace - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.9,
            (255, 193, 43), 2)
        img[hSpace:hSpace + hPaper, wCam - wSpace - wPaper:wCam - wSpace] = PRSImagesList[0]
    elif computerChoice == 2: # rock
        cv2.putText(img, "COMPUTER", (wCam - wRock - wSpace - 5, hSpace - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.9,
            (186, 108, 255), 2)
        img[hSpace:hSpace + hRock, wCam - wSpace - wRock:wCam - wSpace] = PRSImagesList[1]
    elif computerChoice == 3: # scissors
        cv2.putText(img, "COMPUTER", (wCam - wScissors - wSpace - 5, hSpace - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.9,
            (57, 229, 255), 2)
        img[hSpace:hSpace + hScissors, wCam - wSpace - wScissors:wCam - wSpace] = PRSImagesList[2]

    return computerChoice

"""
```

Given the user's and the computer's choices, the function finds the winner and shows it on the screen

The winning-losing options are:

```
-----
WINNER   |  LOSER
-----+-----
paper = 1 | rock = 2
rock = 2  | scissors = 3
scissors = 3 | paper = 1
```

Therefore the next equation holds: $(\text{WINNER} + 1) \% 3 == \text{LOSER} \% 3$

"""

```
def findAndShowWinner(img, userChoice, computerChoice):
```

```
    winner = "T"
```

```
    if (computerChoice + 1) % 3 == userChoice % 3:
```

```
        winner = "C"
```

```
    elif (userChoice + 1) % 3 == computerChoice % 3:
```

```
        winner = "U"
```

```
    if winner == "C":
```

```
        cv2.putText(img, "You Lose!", (203, 100), cv2.FONT_HERSHEY_PLAIN, 3, (14, 2, 203), 5)
```

```
    elif winner == "U":
```

```
        cv2.putText(img, "You Win!", (211, 100), cv2.FONT_HERSHEY_PLAIN, 3, (22, 166, 0), 5)
```

```
    else:
```

```
        cv2.putText(img, "It's a Tie!", (201, 100), cv2.FONT_HERSHEY_PLAIN, 3, (31, 198, 255), 5)
```

```
def main(rec=0):
```

```
    onePaperSignFrameCounter = 0
```

```
    twoHandsFrameCounter = 0
```

```
    while True:
```

```
        img, num_hands_detected, landmarksList = captureLandmarksInFrame()
```

```
        if rec == 0:
```

```
            cv2.putText(img, "WELCOME:", (140, 100), cv2.FONT_HERSHEY_SIMPLEX, 2, (31, 198, 255),
```

```
5)    5)
```

```
            cv2.putText(img, "WELCOME:", (150, 110), cv2.FONT_HERSHEY_SIMPLEX, 2, (134, 68, 6), 5)
```

```
        else:
```

```
            cv2.putText(img, "Try Again", (144, 104), cv2.FONT_HERSHEY_SIMPLEX, 2, (31, 198, 255), 5)
```

```
            cv2.putText(img, "Try Again", (150, 110), cv2.FONT_HERSHEY_SIMPLEX, 2, (134, 68, 6), 5)
```

```
        img[hCam - hExit:hCam, wCam - wExit:wCam] = ExitAndStartImagesList[0]
```

```
        img[hCam - hStart:hCam, 0:wStart] = ExitAndStartImagesList[1]
```

```
        if num_hands_detected == 2:
```

```
            onePaperSignFrameCounter = 0
```

```
            twoHandsFrameCounter += 1
```

```
            if twoHandsFrameCounter >= 20:
```

```

        time.sleep(1)
    return
elif num_hands_detected == 1:
    twoHandsFrameCounter = 0
    userSign = analyseUserChoice(landmarksList)
    if userSign == 1:
        onePaperSignFrameCounter += 1
        if onePaperSignFrameCounter >= 10:
            break
    else:
        onePaperSignFrameCounter = 0
else:
    onePaperSignFrameCounter = 0
    twoHandsFrameCounter = 0
cv2.imshow("Rock Paper Scissors Game", img)
cv2.waitKey(1)

# A game starts - counting down till the decision time:
t_now = time.time()
t_end = t_now + timeUp
while t_now < t_end:
    img, num_hands_detected, landmarksList = captureLandmarksInFrame()
    if num_hands_detected == 2:
        cv2.putText(img, "Show only ONE hand", (wSpace + 30, hSpace - 5),
                    cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 0), 2)
    elif num_hands_detected == 1:
        userChoice = analyseUserChoice(landmarksList)
        if userChoice == 0:
            cv2.putText(img, "Make your choice!", (wSpace + 30, hSpace - 5),
                        cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 0), 2)
        else:
            showUserChoice(img, userChoice)
    else:
        cv2.putText(img, "Make your choice!", (wSpace + 30, hSpace - 5),
                    cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 0), 2)

```

```

    rgb = (random.randint(0, 150), random.randint(0, 150), random.randint(0, 150))
    cv2.putText(img, str(int(t_end - t_now) + 1), (wCam - 150, 150), cv2.FONT_HERSHEY_SIMPLEX,
4, rgb, 12)
    cv2.imshow("Rock Paper Scissors Game", img)
    cv2.waitKey(1)
    t_now = time.time()

# Time's Up! Analyse results:
img, num_hands_detected, landmarksList = captureLandmarksInFrame()
if num_hands_detected >= 2:
    cv2.putText(img, "Too Many Hands!", (125, 100), cv2.FONT_HERSHEY_PLAIN, 3, (134, 68, 6), 5)
elif num_hands_detected == 1:
    userChoice = analyseUserChoice(landmarksList)
    if userChoice == 0:
        cv2.putText(img, "Rock Paper or Scissors ONLY", (70, 100), cv2.FONT_HERSHEY_SIMPLEX, 1,
(134, 68, 6), 3)
    else:
        showUserChoice(img, userChoice)
        computerChoice = ruffleAndShowComputerChoice(img)
        findAndShowWinner(img, userChoice, computerChoice)
else:
    cv2.putText(img, "No Hands.....!", (155, 100), cv2.FONT_HERSHEY_PLAIN, 3, (134, 68, 6), 5)

cv2.imshow("Rock Paper Scissors Game", img)
cv2.waitKey(1)
time.sleep(3)
main(rec+1)

if __name__ == "__main__":
    main()

```

HandTrackingModule.py

```
import cv2
import mediapipe as mp
import time

class handDetector():
    def __init__(self, mode=False, maxHands=2, complexity=1, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.complexity = complexity
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.complexity,
                                         self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        num_hands_detected = 0
        if self.results.multi_hand_landmarks:
            num_hands_detected = len(self.results.multi_hand_landmarks)
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                                self.mpHands.HAND_CONNECTIONS)
        return img, num_hands_detected

    def findPosition(self, img, handNo=0, draw=True):
        lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
```

```
for id, lm in enumerate(myHand.landmark):
    # print(id, lm)
    h, w, c = img.shape
    cx, cy = int(lm.x * w), int(lm.y * h)
    # print(id, cx, cy)
    lmList.append([id, cx, cy])
    if draw:
        cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

return lmList
```

SNAPSHOTS



Fig. No.2: Task 1

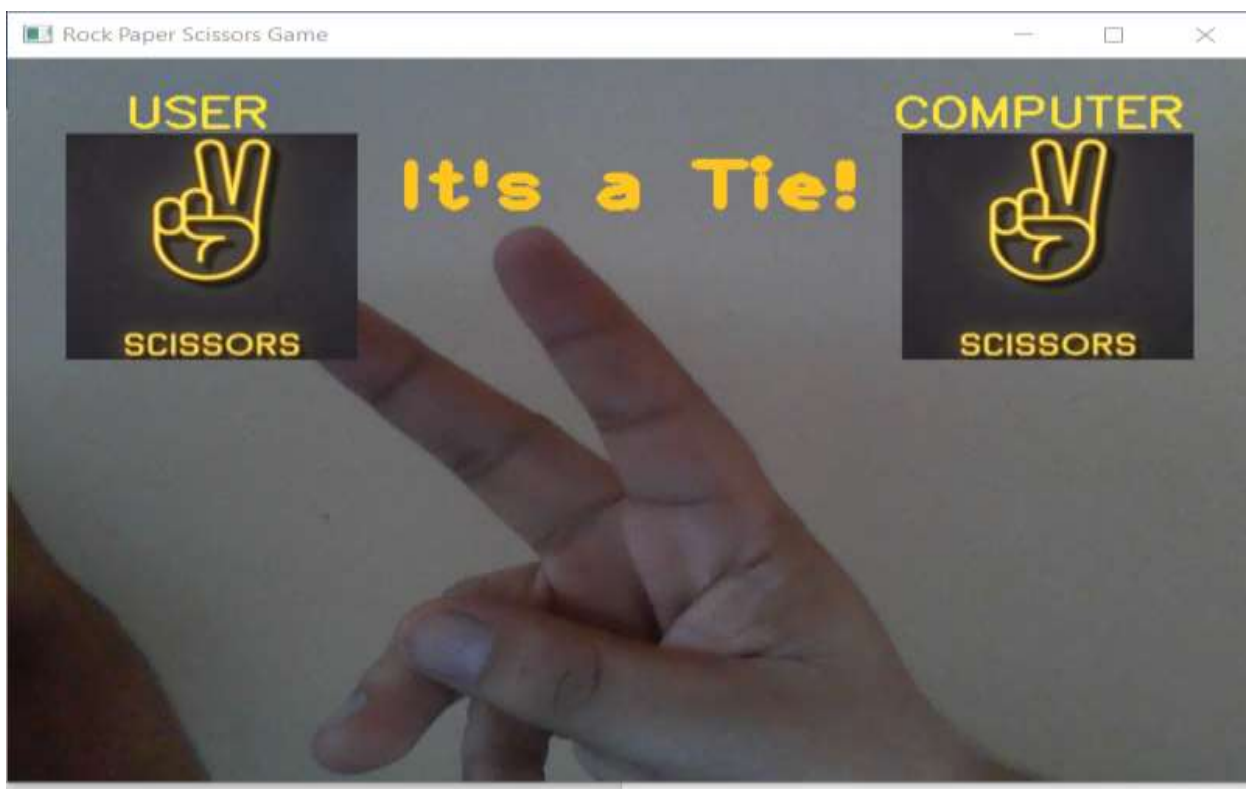


Fig. No.3: Task 2

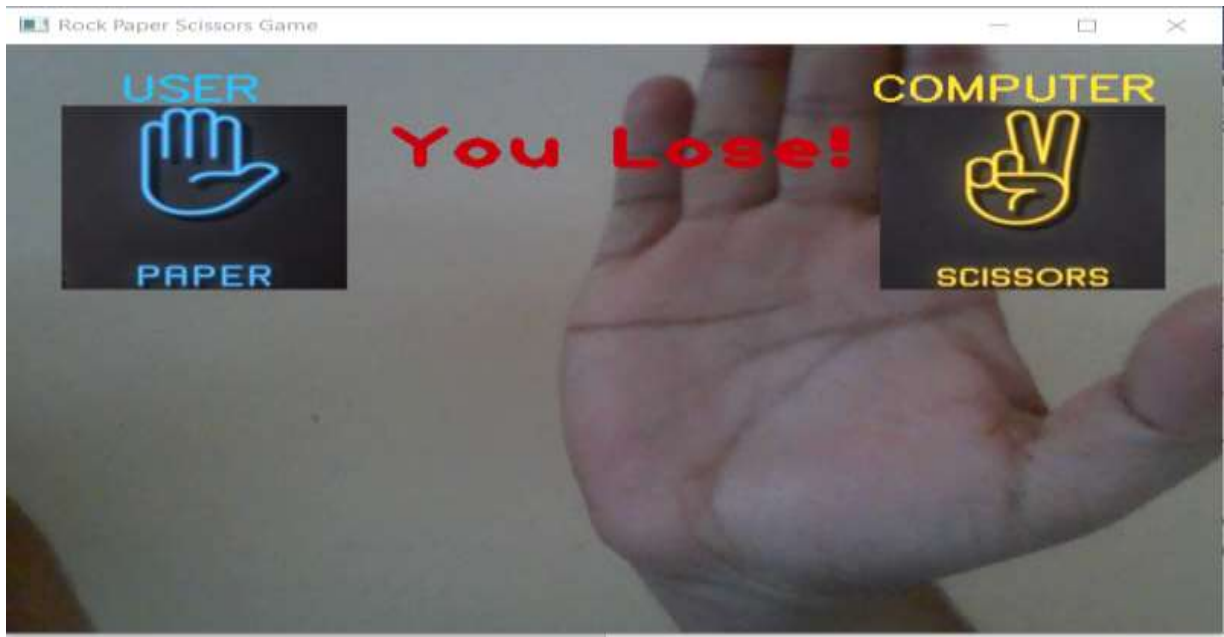


Fig. No. 4: Task 3

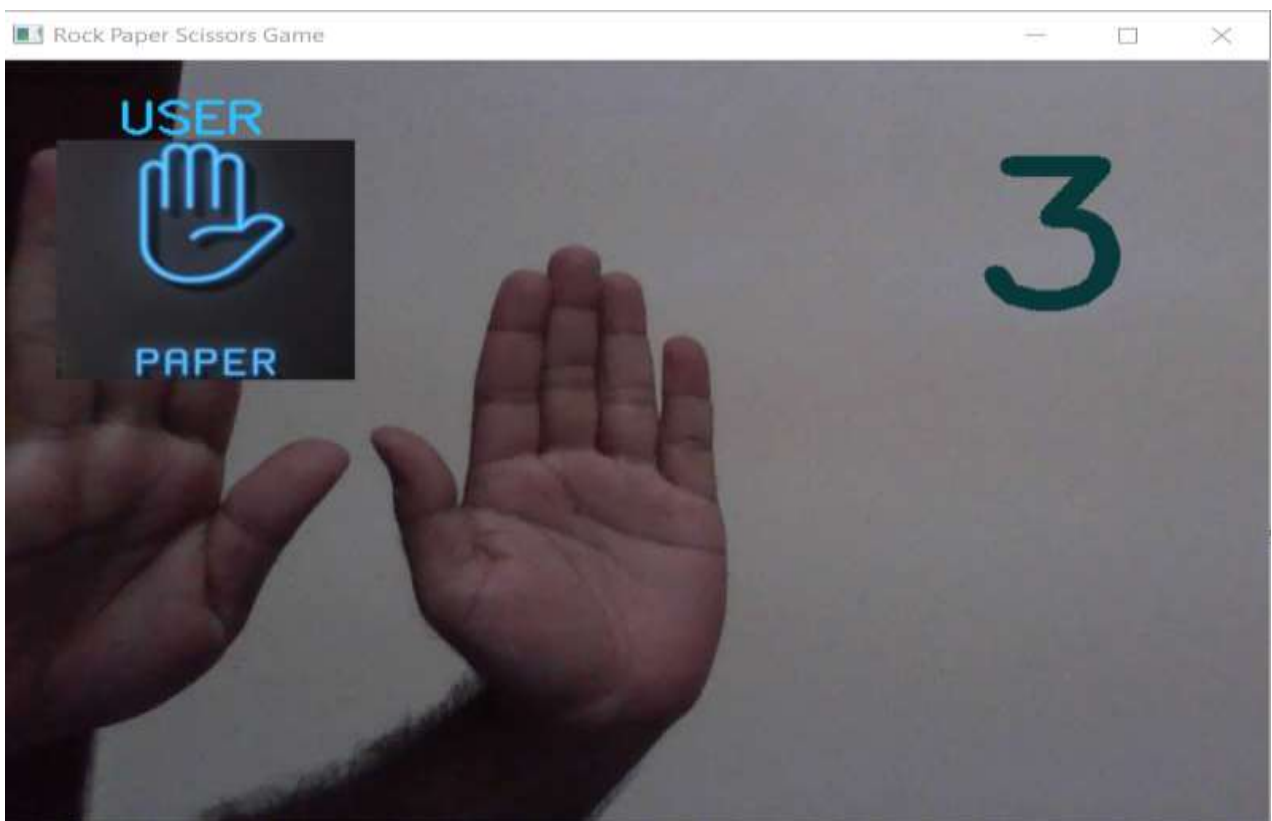


Fig No.5: Closing Window

CONCLUSION

The Jarvis AI desktop assistant project has been a resounding success, achieving its initial objectives and more. Throughout its development and deployment, the project garnered substantial user adoption, and the feedback received has been invaluable in shaping the assistant's capabilities and responsiveness.

Users have reported significantly enhanced productivity, with the assistant automating tasks and providing quick access to information, ultimately streamlining their daily routines. The project's unwavering commitment to user data security and privacy has been a hallmark of its success, and we have continuously improved the assistant's performance through regular updates and community contributions.

Challenges encountered along the way have served as valuable learning experiences, and we remain dedicated to addressing them in future iterations. The impact of the Jarvis AI assistant on users' lives and work has been profound, and we express our heartfelt gratitude to the development team, partners, and the broader community that supported this journey.

Looking ahead, we are excited about the future directions of the project, including expanding features, third-party integrations, and enhanced platform compatibility, all while maintaining our commitment to ethical AI development and compliance with relevant regulations. We will continue to provide support, updates, and resources to ensure users derive the maximum benefit from this innovative tool.

The Jarvis AI desktop assistant project is not just a milestone; it's a foundation upon which we will build a future where AI seamlessly integrates into our lives, making them more efficient and enjoyable.

Future Scope

The game of Rock, Paper, Scissors is a simple yet popular game that can be enhanced and expanded in various ways.

Here's the future scope for a Rock, Paper, Scissors project:

- Multiplayer and Online Play
- AI and Machine Learning
- Variations and Customization
- Tournaments and Leaderboards
- Mobile Apps and Cross-Platform Play
- Educational and Training Applications
- Social Media Integration
- AI Chatbot for Strategy Advice

BIBLIOGRAPHY

- ❖ <https://www.geeksforgeeks.org/>
- ❖ <https://openai.com/>
- ❖ <https://www.youtube.com/>
- ❖ <https://www.google.com>
- ❖ <https://chatgpt.com>