PROJECT REPORT ON

Rock Paper Scissors Game (Python game)

(UNDER THE PARTIAL FULFILLMENT OF THE UNIVERSITY

FOR COURSE OF T.Y.BSC COMPUTER SCIENCE)

SUBMITTED BY:

MR.SHUBHAM VARTAK

GUIDED BY:

PROF. MS. MONIKA BORATE

DEPARTMENT OF COMPUTER SCIENCE

PARLE TILAK VIDYALAYA ASSOCIATION’S

MULUND COLLEGE OF COMMERCE S.N.ROAD,

MULUND (WEST), MUMBAI-80

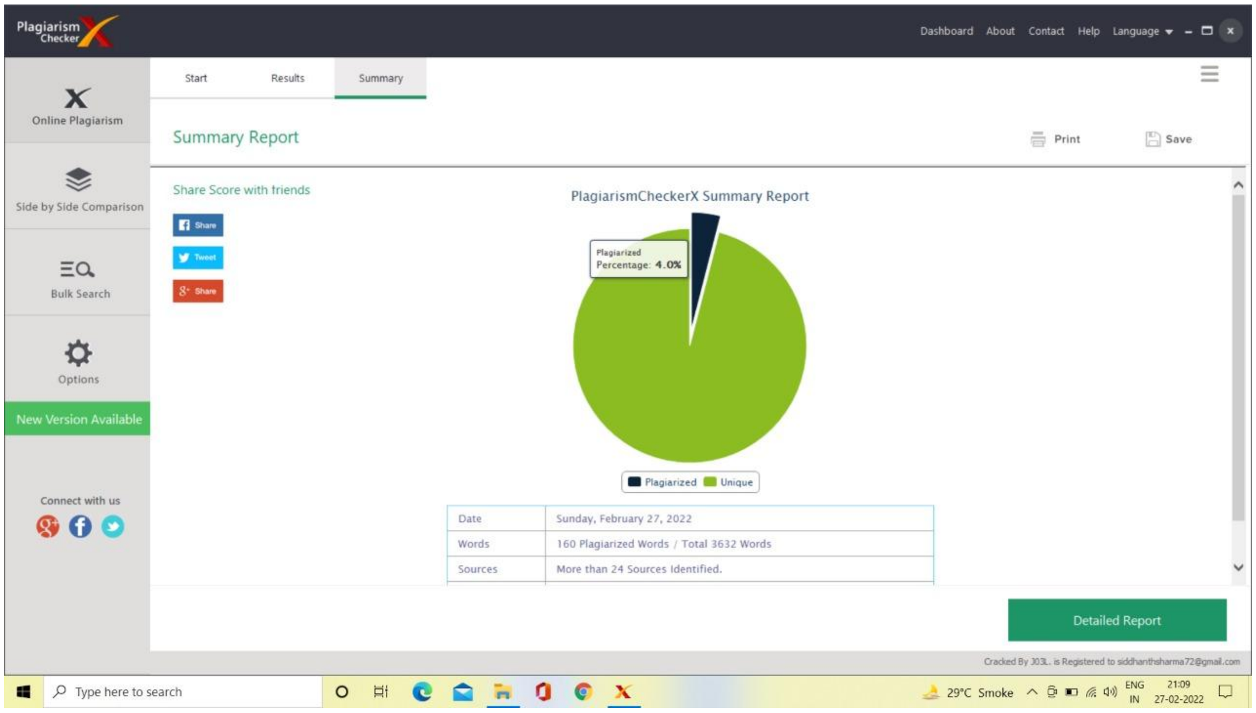
UNIVERSITY OF MUMBAI 2021-2022

PLAGIARISM REPORT

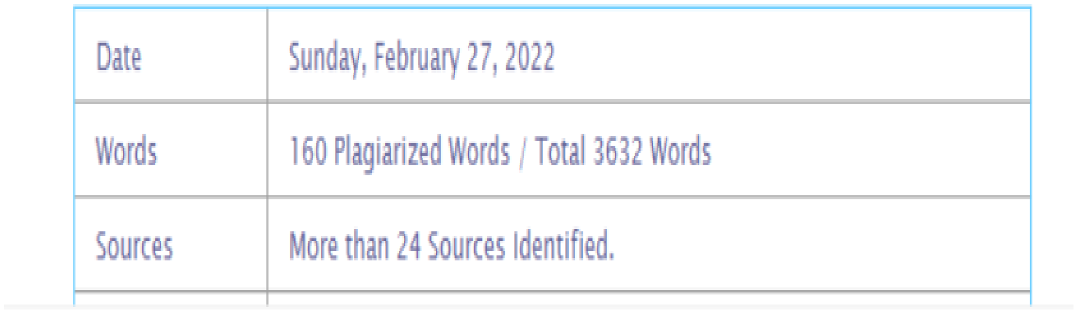
Plagiarism Checker X Originality Report

Similarity Found: 4%

Date: Sunday, February 27, 2022
Statistics: 160 words Plagiarized / 3632 Total words
Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.



Date	Sunday, February 27, 2022
Words	160 Plagiarized Words / Total 3632 Words
Sources	More than 24 Sources Identified.



Date	Sunday, February 27, 2022
Words	160 Plagiarized Words / Total 3632 Words
Sources	More than 24 Sources Identified.

ACKNOWLEDGEMENT

I like to extend my gratitude to **Dr. Sonali Pednekar**, our Principal and all staff of Mulund College of Commerce for providing us moral support, conducivework environment and the much needed inspiration to complete this project on time.

I also take this opportunity to thank our Course Coordinator **Prof. Reena Shah** and all the faculties of Department of Computer Science for giving us the most needed guidance and continuous encouragementthroughout the duration of the Programme.

I wish to extend my deepest gratitude and special thanks to my project guide **Prof. MS. Monika Borate**, for giving their generous support, necessary inputs and companionship during my project work.

I would like to convey my special thanks to the Management and all the staff of the college for providing the required infrastructure and resource to enable the completion and enrichment of my project.

I am extremely grateful to the University of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally I thank all my fellow friends who have directly or indirectly helped me in completing my project.

Shubham Dattatray Vartak

1.Title

Title of Project:

Rock Paper Scissors (Game)

Type of Project:

PYTHON GAME

Developed by:

Shubham Dattatray Vartak

2.Introduction

Rock paper scissors is a game constructed in python language using tkinter library. The is game is GUI. The game contains sign up page /login page, welcome page, main game page, result page, leaderboard page. Login /Sign up page and leaderboard is connected to database to store data of players. In Leaderboard page there is option to switch theme (Light and Dark) also leaderboard will show names of all the players (Signed up players), matches played, matches won, matches lose, matches tied.

2.1 Objective of the Project:

Objective of this project (Rock paper scissors game) is to create a challenging and exciting experience for the user which will future lead to allow them to compete from crossing their number of matches won and number of matches played. The game is build for a single player that plays with computer, anywhere, and anytime.

2.2 Description of current game:

Rock Paper scissors game is a hand game, usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are “Rock”, “Paper”, “Scissors”.

2.3 Limitations of current game:

Limitations of current game is that is played between two players, hence we cannot play this game along, we have to wait for one to play with us, also we cannot play it anywhere as it is played physically.

2.4 Description of Proposed game:

In proposed game, Game is played between Computer(System) and user, so it can also be played when you are along (no need to wait for anyone) and also there will be leaderboard which will keeps you challenging by showing number of matches won, number of matches losed. Also it can be played anywhere to relax and refresh mind by just opening in computer system.

2.5 Advantages of Proposed system:

The game is for enterainment purposed. But still it helps in reducing excess stress, also improves response time, improves thinking and concentration power.

3. Requirement Specification.

3.1 Software Requirement:

- Windows operating system
- Application Software – Front end:- Python GUI
  Back end:- SQLite3
- Platform – Python 3

3.2 Hardware Requirement:

- Server with minimum 2 GB space
- 4 GB ram

3.3 Data Requirement:

- Username
- Password
- Gender

3.4 Fact Finding Questions:

- What is game about ?
- How is the game played ?
- What are winning rules for the game ?
- How many rounds will be there in game ?
- How is data maintained ?
- Is it single player or multiplayer game ?
- I still have some more queries, where do I contact?

4. System Design Details

4.1 Event Table:

No.	Event	Trigger	Source	Activity	Response	Destination
1.	Sign Up	Button	User	Creates a new account	Adds new account to db	server
2.	Log In	Button	User	Login to user	Access to game for user	Server
3.	Start Now	Button	User	To start the game	Starts the game Screen	Game page
4.	Rock,Paper,Scissors	Image	User	To select Rock,Paper, Scissors	Selects the anyone (Rock,Paper ,Scissors)	Game page
5.	Result page	Page	User	To view result	It shows the result of game	Result page
6.	Leaderboard	Button	User	To view Leaderboard	Shows Leaderboar d of players.	Leaderboard page
7.	Play again	Button	User	To play again	It starts the game again without logging in.	Game page
8.	Exit	Button	User	To exit game	Exits Game	Home page

4.2 Class Diagram:

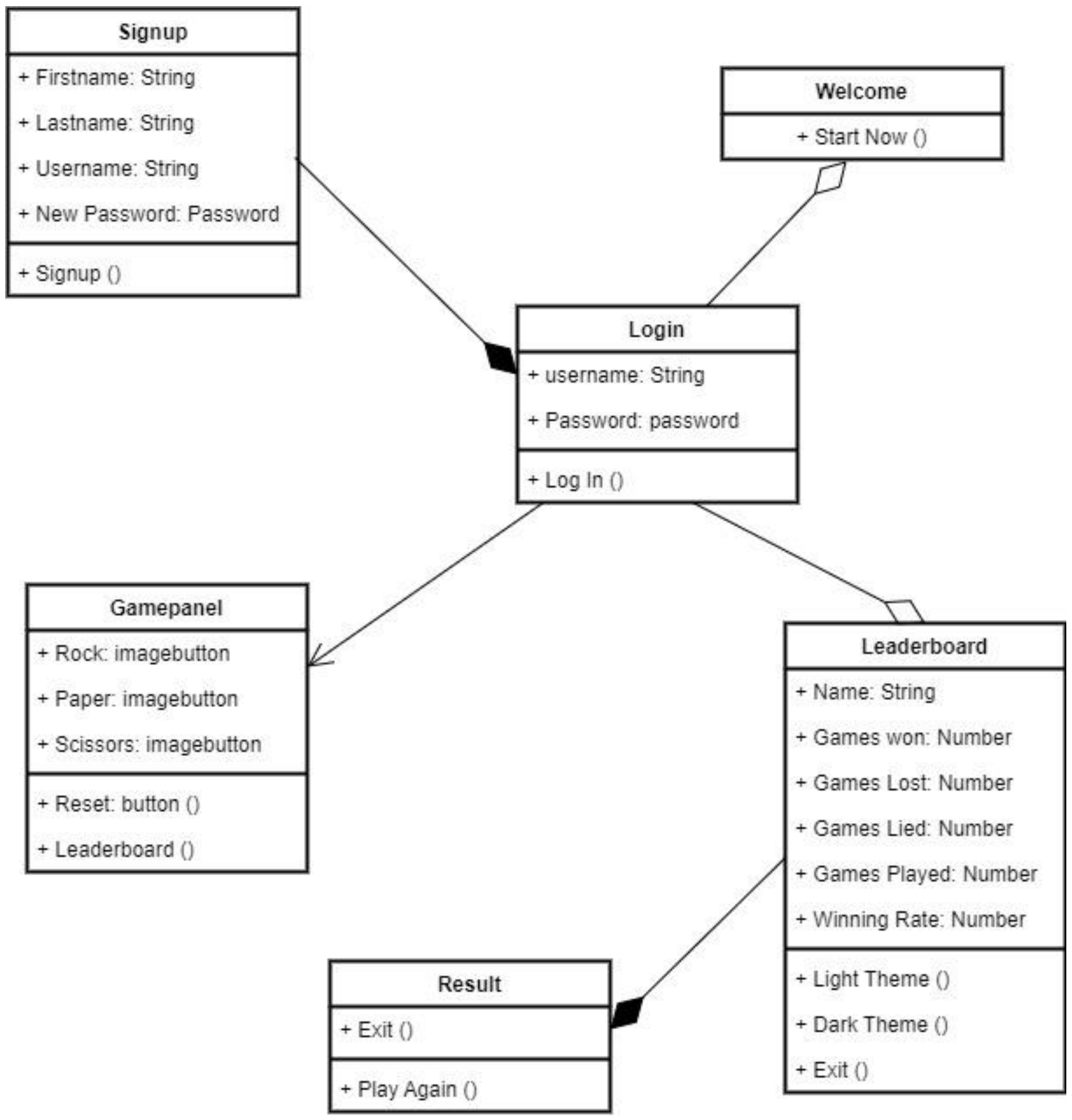


Figure 4.2: Class Diagram

4.3 Use Case Diagram:

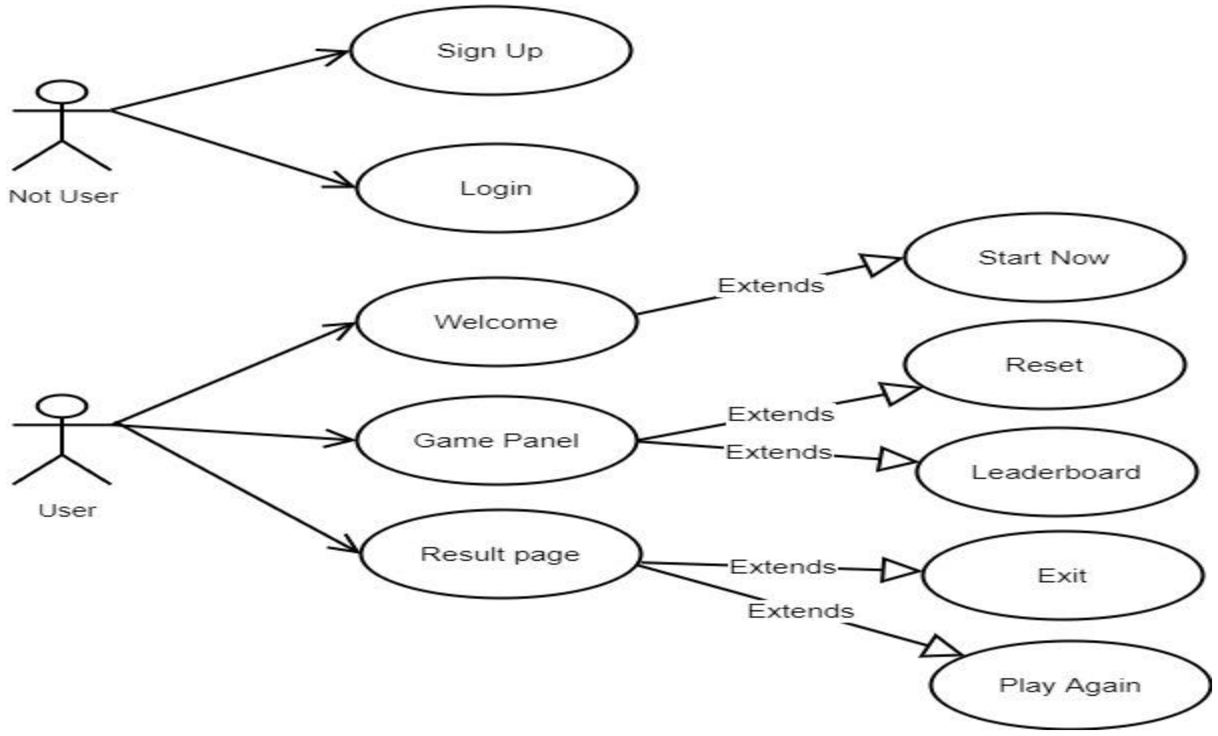


Figure 4.3.1: Use Case Diagram (Game Scenario)

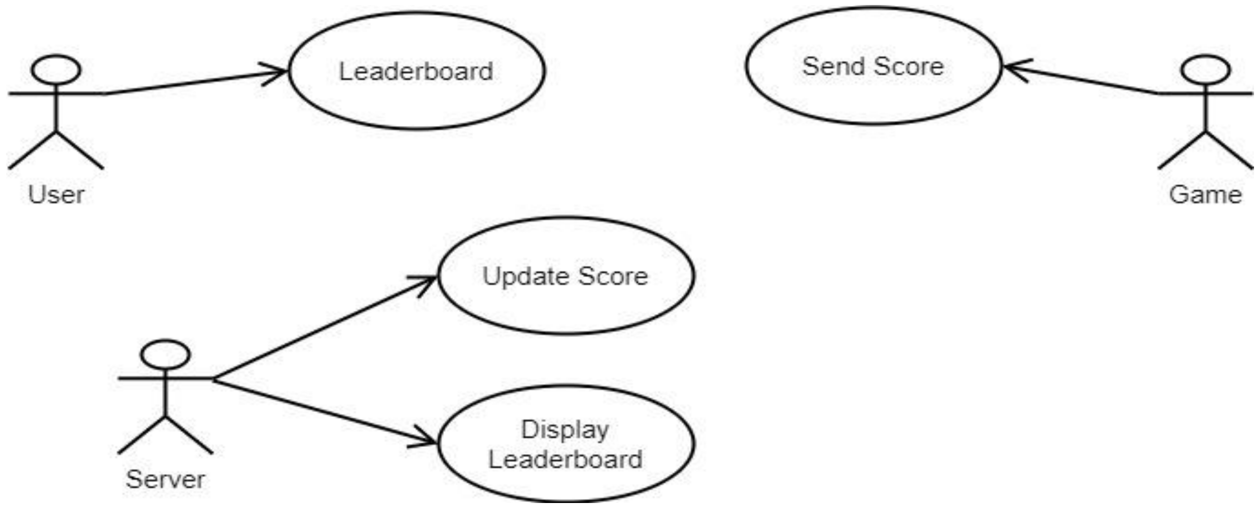


Figure 4.3.2: Use Case Diagram (Leaderboard Scenario)

4.4 Sequence Diagram:

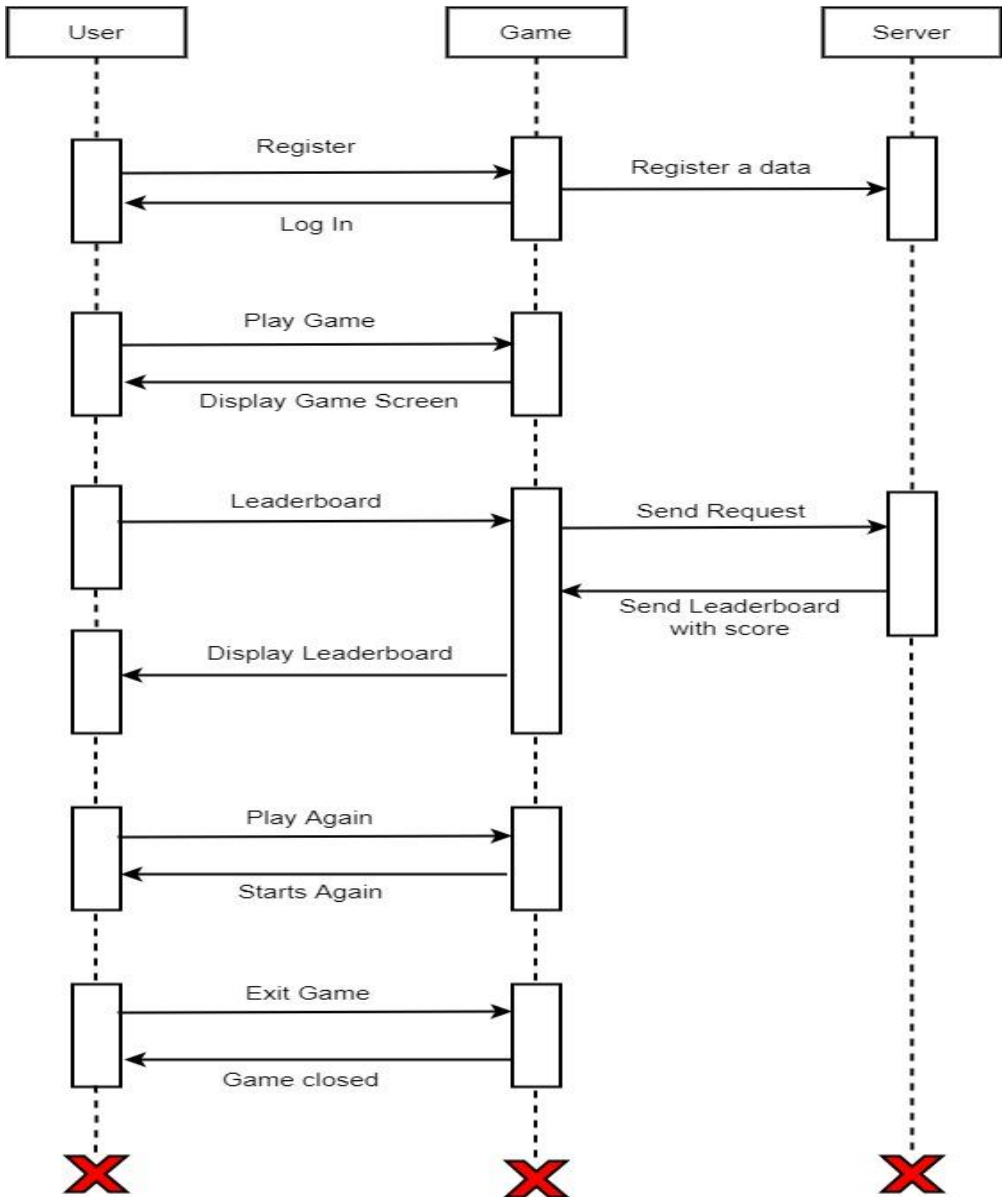


Figure 4.4: Sequence Diagram

4.5 Activity Diagram:

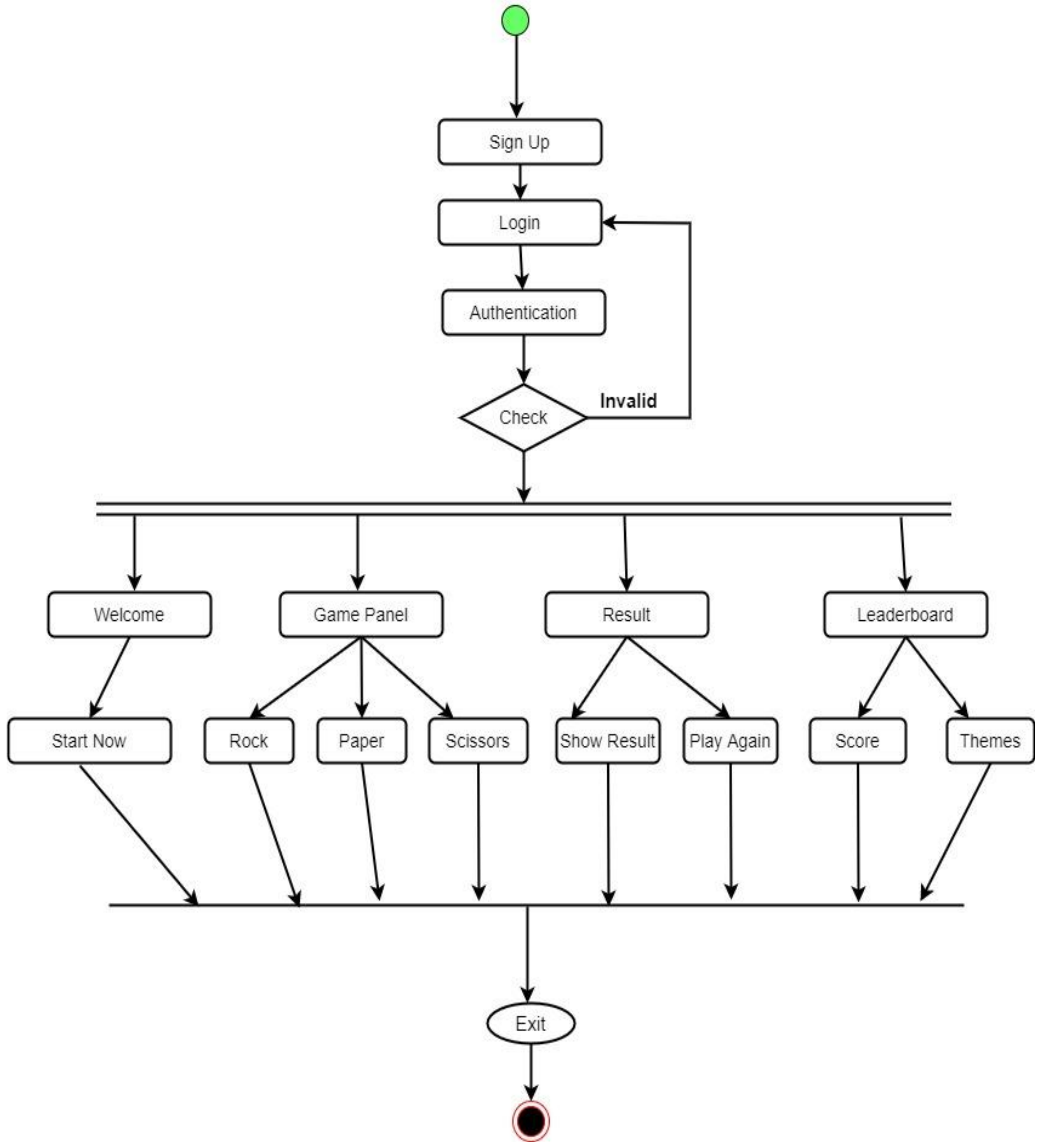


Figure 4.5: Activity Diagram

4.6 State Diagram:

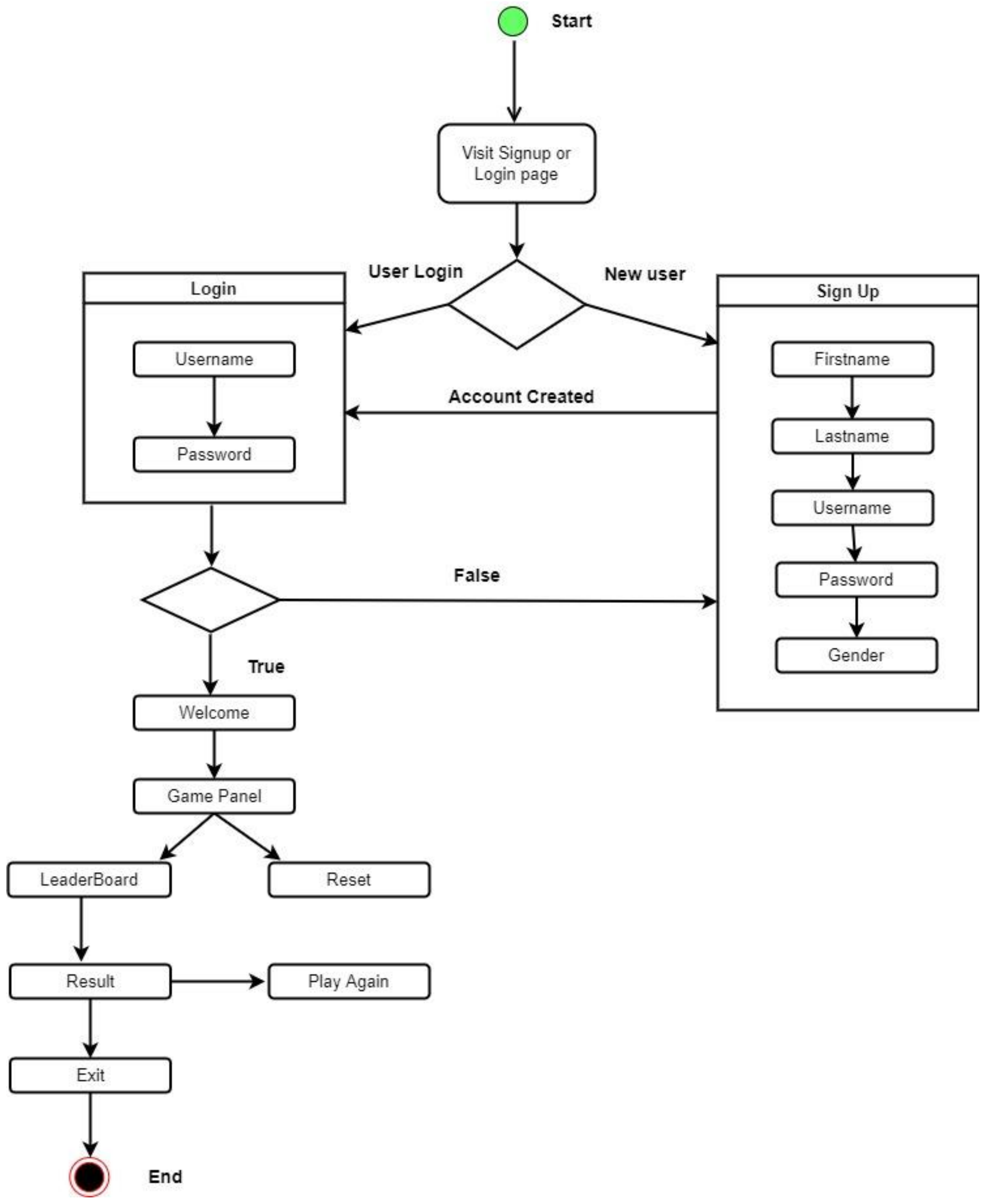


Figure 4.6: State Diagram

4.7 Package Diagram:

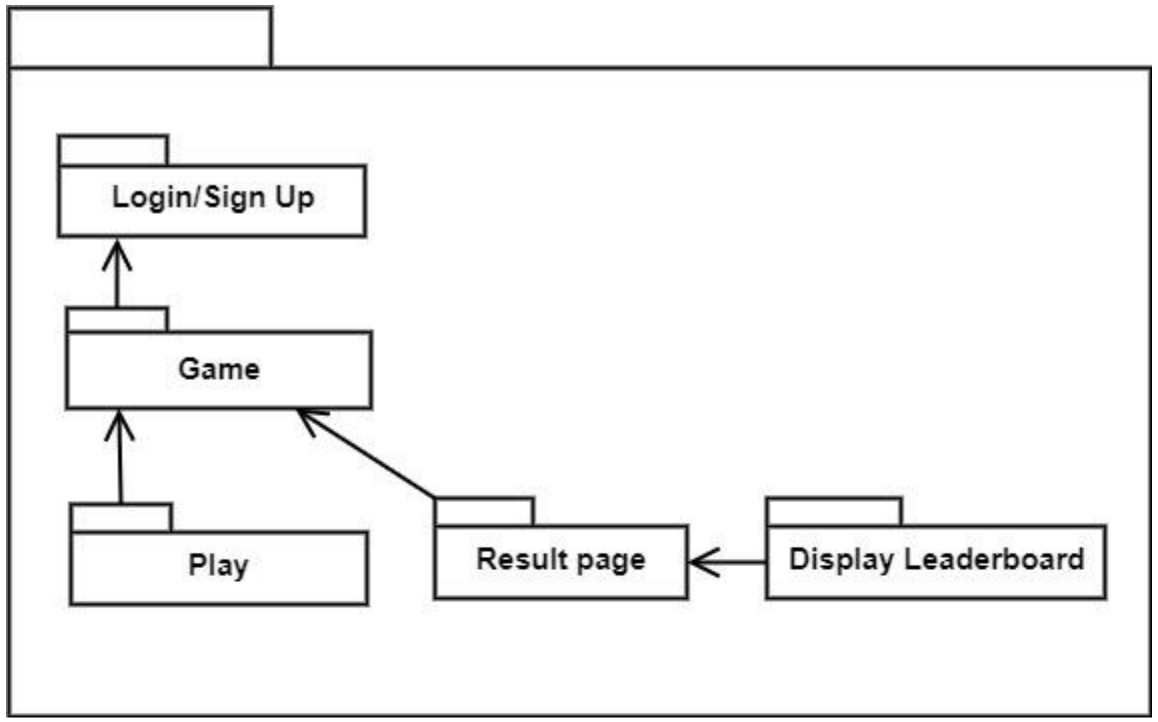


Figure 4.7: Package Diagram

4.8 Component Diagram:

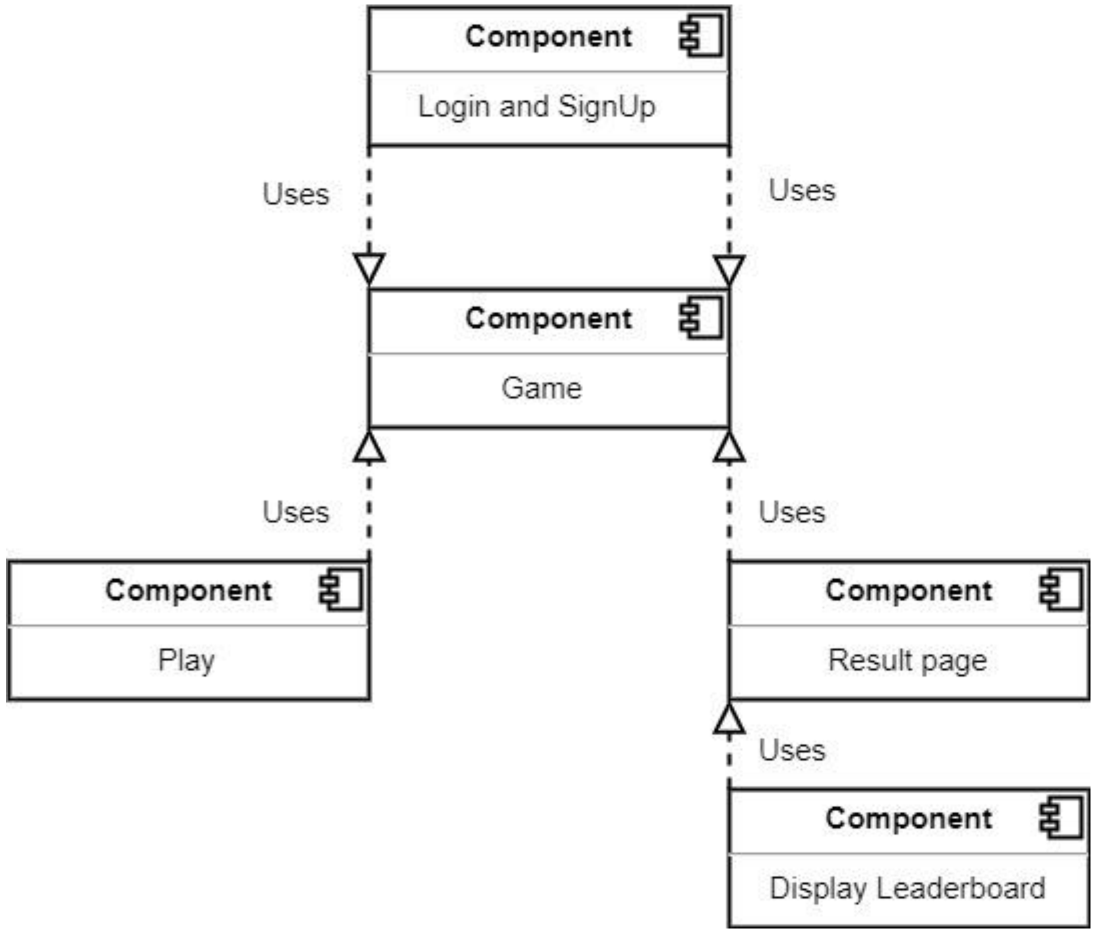


Figure 4.8: Component Diagram

4.9 Deployment Diagram:

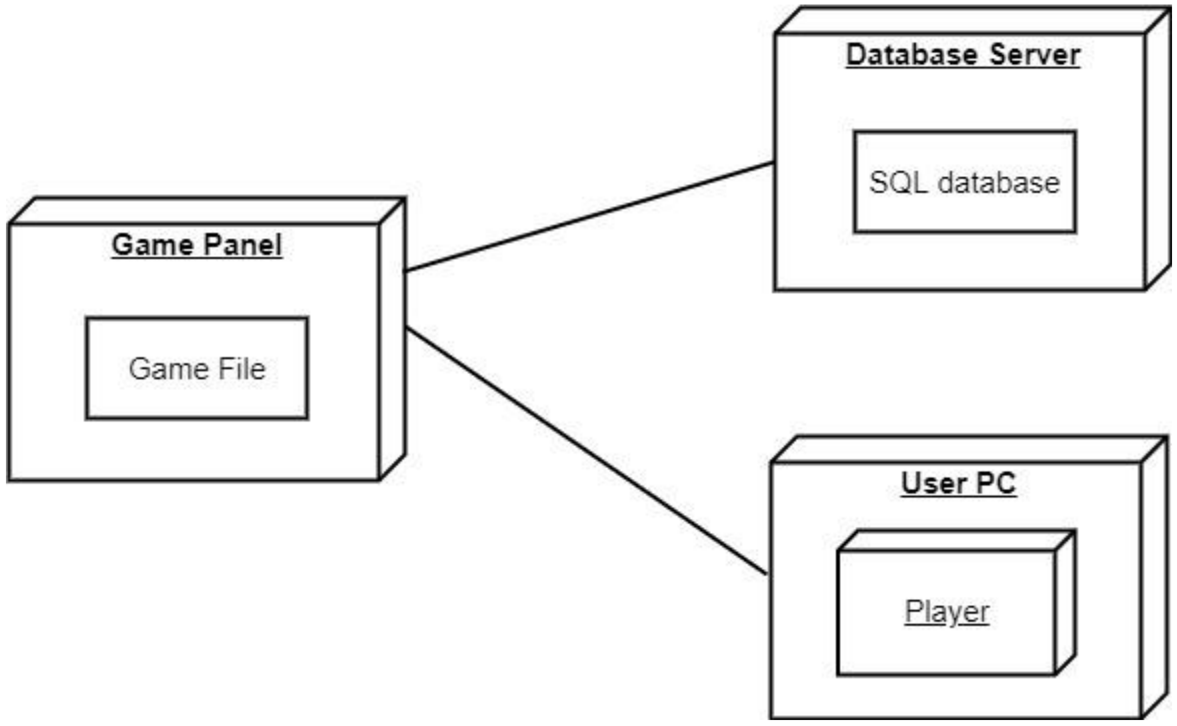
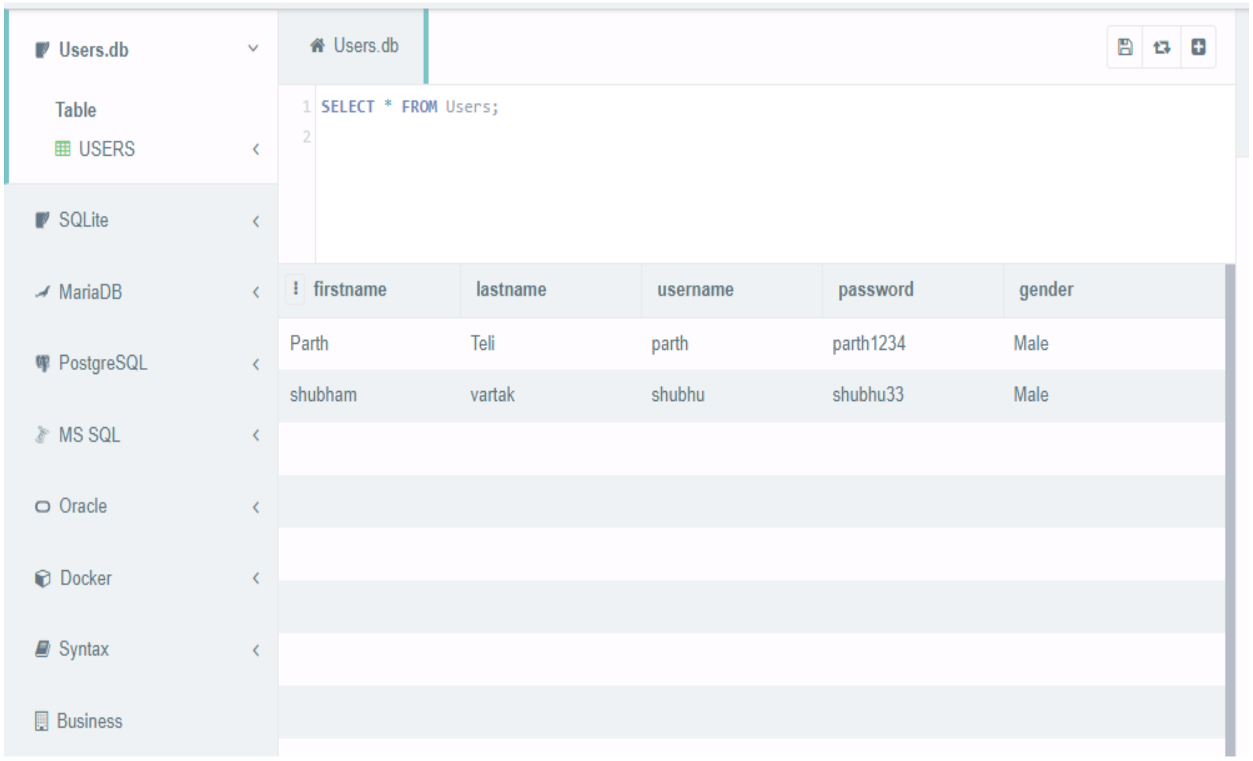


Figure 4.9: Deployment Diagram

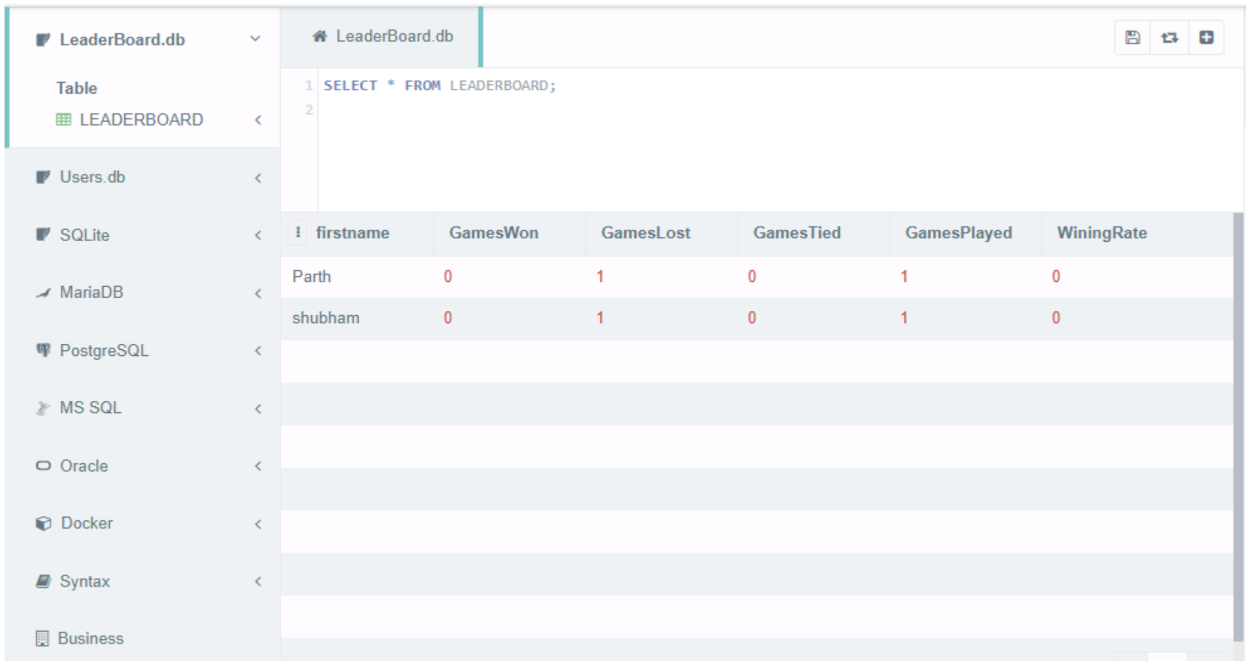
4.10 Database Design:



```
SELECT * FROM Users;
```

firstname	lastname	username	password	gender
Parth	Teli	parth	parth1234	Male
shubham	vartak	shubhu	shubhu33	Male

Figure 4.10.1: Database table for users.



```
SELECT * FROM LEADERBOARD;
```

firstname	GamesWon	GamesLost	GamesTied	GamesPlayed	WiningRate
Parth	0	1	0	1	0
shubham	0	1	0	1	0

Figure 4.10.2: Database table for Leaderboard.

5. System Implementation.

```
from tkinter import *

import sqlite3

from PIL import ImageTk, Image

from tkinter import messagebox

import random

sqlite3.paramstyle = 'named'

root = Tk()

root.title("Login/Signup Form")

root.geometry("400x400")

root.resizable(width=False, height=False)

lframe = LabelFrame(root, padx = 15, pady = 15, borderwidth=5)

sframe = LabelFrame(root, padx=15, pady=15, borderwidth=5)

USER = ""

UserGender = ""

userWin = 0

compWin = 0

counter = 0

turns = 3  # After how many turns the game will end

rHandButton = ''

pHandButton = ''

sHandButton = ''

# Function creating the whole ROCK-PAPER-SCISSORS Game

def play():
    global rHandButton, pHandButton, sHandButton, userWin, compWin, Scoreboard,
resetButton, rockLabel, paperLabel, scissorLabel, buttonHolder, LeaderBoardBtn

    rockLabel = Label(root, text='Rock', bg='#238f02', fg='white', width=35, padx=10, pady=10)

    paperLabel = Label(root, text='Paper', bg='#de9a03', fg='white', width=35, padx=10, pady=10)

    scissorLabel = Label(root, text='Scissors', bg='#c20c0c', fg='white', width=35, padx=10,
pady=10)

    rockLabel.grid(row=0, column=0, padx=5, pady=7)

    paperLabel.grid(row=0, column=1, padx=5, pady=5)

    scissorLabel.grid(row=0, column=2, padx=5, pady=5)

    rHandButton = Button(root, image=rHandPhoto, command=lambda: youPick('rock'))

    pHandButton = Button(root, image=pHandPhoto, command=lambda: youPick('paper'))

    sHandButton = Button(root, image=sHandPhoto, command=lambda: youPick('scissors'))

    rHandButton.grid(row=1, column=0)

    pHandButton.grid(row=1, column=1)

    sHandButton.grid(row=1, column=2)

    Scoreboard = Label(root, text="SCORE \n\n    " + USER.upper() + " - " + str(userWin) +
"\t\tCOMPUTER - " + str(compWin), bg='orange',

                       fg='white', padx=10, pady=20)

    Scoreboard.config(font=("Times", 15))

    Scoreboard.grid(row=2, column=0, columnspan=2, sticky=W + E, padx=10, pady=10)

    buttonHolder = Frame(root)

    buttonHolder.grid(row=2, column=2)

    resetButton = Button(buttonHolder, text='RESET', fg='white', command=lambda:
reset_frame(), bg='green', width=30,pady=10)

    resetButton.pack(pady=5)

    LeaderBoardBtn = Button(buttonHolder, text='Leader Board', fg='black', command=lambda:
getLeaderboard(), bg='cyan', width=30, pady=10)
```

```
LeaderBoardBtn.pack(pady=5)

# Computer randomly picks a choice

def computerPick():

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

    return choice

# Function to play the game again after it is finished once

def playAgain():

    global counter, userWin, compWin, rHandButton, pHandButton, sHandButton

    counter = 0

    userWin = 0

    compWin = 0

    # top.quit()

    start()

    return

# Function containing the whole logic of won-lose-tie in the game. Decision maker :)

def youPick(yourChoice):

    global click, userWin, compWin, Scoreboard, rockImage, tieImage, paperImage, loseImage,
scissorImage, winImage, compPick, rockLabel, paperLabel, scissorLabel, counter, turns, top

    compPick = computerPick()

    if click:

        counter += 1

        if yourChoice == 'rock':

            rHandButton.configure(image=rockImage)

            rockLabel.configure(text='Rock')

            if compPick == 'rock':

                pHandButton.configure(image=rockImage)
                sHandButton.configure(image=tieImage)
                paperLabel.configure(text='Rock')
                scissorLabel.configure(text='Tie')
                click = False
            elif compPick == 'paper':
                pHandButton.configure(image=paperImage)
                sHandButton.configure(image=loseImage)
                paperLabel.configure(text='Paper')
                scissorLabel.configure(text='Lose')
                compWin += 1
                click = False
            else:
                pHandButton.configure(image=scissorImage)
                sHandButton.configure(image=winImage)
                paperLabel.configure(text='Scissors')
                scissorLabel.configure(text='Win')
                userWin += 1
                click = False
        elif yourChoice == 'paper':
            rHandButton.configure(image=paperImage)
            rockLabel.configure(text='Paper')
            if compPick == 'rock':
                pHandButton.configure(image=rockImage)
                sHandButton.configure(image=winImage)
                paperLabel.configure(text='Rock')
                scissorLabel.configure(text='Win')
                userWin += 1
                click = False
            elif compPick == 'paper':
                pHandButton.configure(image=paperImage)
                sHandButton.configure(image=tieImage)
                paperLabel.configure(text='Paper')
                scissorLabel.configure(text='Tie')
                click = False
            else:
                pHandButton.configure(image=scissorImage)
                sHandButton.configure(image=loseImage)
                paperLabel.configure(text='Scissors')
                scissorLabel.configure(text='Lose')
                compWin += 1
                click = False
        else:
            rHandButton.configure(image=scissorImage)
            rockLabel.configure(text='Scissors')
            if compPick == 'rock':
                pHandButton.configure(image=rockImage)
                sHandButton.configure(image=loseImage)
                paperLabel.configure(text='Rock')
                scissorLabel.configure(text='Lose')
                compWin += 1
                click = False

            elif compPick == 'paper':

                pHandButton.configure(image=paperImage)

                sHandButton.configure(image=winImage)

                paperLabel.configure(text='Paper')

                scissorLabel.configure(text='Win')

                userWin += 1

                click = False

            else:

                pHandButton.configure(image=scissorImage)

                sHandButton.configure(image=tieImage)

                paperLabel.configure(text='Scissors')

                scissorLabel.configure(text='Tie')

                click = False

          else:

        if yourChoice == 'rock' or yourChoice == 'paper' or yourChoice == 'scissors':

            rHandButton.configure(image=rHandPhoto)

            pHandButton.configure(image=pHandPhoto)

            sHandButton.configure(image=sHandPhoto)

            rockLabel.configure(text='Rock')

            paperLabel.configure(text='Paper')

            scissorLabel.configure(text='Scissors')

            click = True

          Scoreboard = Label(root, text="SCORE \n\n    "+ USER.upper() +" - " + str(userWin) +
        "\t\tCOMPUTER - " + str(compWin), bg='orange',

                      fg='white', padx=10, pady=20)
    Scoreboard.config(font=("Times", 15))
    Scoreboard.grid(row=2, column=0, columnspan=2, sticky=W + E, padx=10, pady=10)
    if counter == turns:
        message = ''
        if userWin > compWin:
            message = 'You Won!!'
        elif userWin < compWin:
            message = 'You Lose!!'
        else:
            message = 'You Tied!!'
        insertToLeaderBoard()
        top = Toplevel()
        top.title("Result")
        top.geometry('300x300')
        confetiImg = PhotoImage(file="RockPaperScissorsImages/confeti.gif")
        confetiLabel = Label(top, image=confetiImg)
        confetiLabel.image = confetiImg
        confetiLabel.grid(row=0, column=0)
        messageFrame = Frame(top)
        messageFrame.grid(row=0, column=0)
        message = Label(messageFrame, text=message)
        message.config(font=("Times", 30, 'bold'))
        message.pack()
        exitButton = Button(messageFrame, text="Exit", bg='red', fg='white', width=10, padx=3,
pady=3,command=root.quit)
        exitButton.config(font=("Times", 12))
        exitButton.pack(pady=3, padx=3)

        rHandButton.configure(state="disabled")

        pHandButton.configure(state="disabled")

        sHandButton.configure(state="disabled")

        playAgainBtn = Button(messageFrame, text="PlayAgain", bg='#8953ff', fg='white',
width=10, padx=3, pady=3,command=playAgain)

        playAgainBtn.config(font=("Times", 12))

        playAgainBtn.pack(pady=3, padx=3)

# Reseting the frame to original starting pictures

def reset_frame():

    global click

    rHandButton.configure(image=rHandPhoto)

    pHandButton.configure(image=pHandPhoto)

    sHandButton.configure(image=sHandPhoto)

    click = True

click = ''

# Function creating the GAME window, reading the images

def start():

    global root, click, rHandPhoto, pHandPhoto, sHandPhoto, userWin, compWin, rockImage,
paperImage, scissorImage, loseImage, winImage, tieImage

    root.destroy()

    root = Tk()

    root.title('Rock Paper Scissors Game')

    root.resizable(width=False, height=False)

    click = True

    userWin = 0
    compWin = 0
    # ----------------Image set----------------
    rHandPhoto = PhotoImage(file='RockPaperScissorsImages/rHand.png')
    pHandPhoto = PhotoImage(file='RockPaperScissorsImages/pHand.png')
    sHandPhoto = PhotoImage(file='RockPaperScissorsImages/sHand.png')
    rock = Image.open("RockPaperScissorsImages/Rockimg.jpg")
    rockImage = ImageTk.PhotoImage(rock)
    paper = Image.open("RockPaperScissorsImages/Paperimg.jpg")
    paperImage = ImageTk.PhotoImage(paper)
    scissors = Image.open("RockPaperScissorsImages/Scissorsimg.jpg")
    scissorImage = ImageTk.PhotoImage(scissors)
    win = Image.open("RockPaperScissorsImages/YouWin.jpg")
    winImage = ImageTk.PhotoImage(win)
    lose = Image.open("RockPaperScissorsImages/YouLose.jpg")
    loseImage = ImageTk.PhotoImage(lose)
    tie = Image.open("RockPaperScissorsImages/YouTie.jpg")
    tieImage = ImageTk.PhotoImage(tie)
    play()
    return
# Function creating the WELCOME User Page
def welcomeUserPage():
    global root
    root.destroy()
    root =  Tk()
    root.title("Welcome ^_^ ")
    root.geometry("400x400")

    root.resizable(width=False, height=False)

    name = "Welcome " + USER

    welcomeUser = Label(root, text = name, pady = 20, width= 25)

    welcomeUser.config(font=("Times", 20, "bold"))

    welcomeUser.grid(row = 0, column = 0, columnspan = 2)

    if UserGender == "Male":

        maleimg = ImageTk.PhotoImage(Image.open("RockPaperScissorsImages/Male.jpg"))

        maleLabel = Label(root ,image=maleimg)

        maleLabel.image = maleimg

        maleLabel.grid(row=1, column=0)

        compimg = PhotoImage(file = "RockPaperScissorsImages/computer.png")

        compLabel = Label(root, image=compimg)

        compLabel.image = compimg

        compLabel.grid(row=1, column=1)

    else:

        femaleimg = ImageTk.PhotoImage(Image.open("RockPaperScissorsImages/Female.jpg"))

        femaleLabel = Label(root, image = femaleimg)

        femaleLabel.image = femaleimg

        femaleLabel.grid(row=1, column = 0)

        compimg = PhotoImage(file="RockPaperScissorsImages/computer.png")

        compLabel = Label(root, image=compimg)

        compLabel.image = compimg

        compLabel.grid(row=1, column=1)
    StartBtn = Button(root, text="Start Now", pady=10, width=27, bg='green', fg='white',
command = start)

    StartBtn.grid(row=2, column=0, columnspan = 2, pady=(30,0))

    return

# Function to check if the username and password is present in the database and is correct

def check():

    global username, password, USER, UserGender

    conn = sqlite3.connect('Users.db')

    c = conn.cursor()

    c.execute("SELECT * from Users where username = :user and password = :pass",

            {

                'user': username.get(),

                'pass': password.get()

            }

            )

    data = c.fetchone()

    conn.commit()

    conn.close()

    if username.get() == "" or password.get() == "" or data == None:

        messagebox.showerror("Try Again", "Username or password is incorrect.")

        # clear the textboxes

        username.delete(0, END)

        password.delete(0, END)

    else:

        USER += data[0]

        UserGender = data[4]
        welcomeUserPage()
    return
# Function to create USERS TABLE
def create():
    conn = sqlite3.connect('Users.db')
    c = conn.cursor()
    c.execute("""
    CREATE TABLE USERS(
    firstname text,
    lastname text,
    username text,
    password text,
    gender text
    )
    """)
    conn.commit()
    conn.close()
    return
# create()
# Function to create LeaderBoard Table
def LeaderboardCreate():
    conn = sqlite3.connect('LeaderBoard.db')
    c = conn.cursor()
    c.execute("""
    CREATE TABLE LEADERBOARD(
    firstname text,
    GamesWon int,
    GamesLost int,
    GamesTied int,
    GamesPlayed int,
    WiningRate real
    )
    """)
    conn.commit()
    conn.close()
    return
# LeaderboardCreate()
# Function to INSERT data in the Leaderboard table
def insertToLeaderBoard():
    conn = sqlite3.connect('LeaderBoard.db')
    c = conn.cursor()
    c.execute("SELECT * FROM LEADERBOARD WHERE firstname = :USER", {
        'USER' : USER
    })
    data = c.fetchone()
    won = data[1]
    lost = data[2]
    tied = data[3]
    if userWin>compWin:
        won += 1
    elif userWin<compWin:
        lost += 1
    else :
        tied += 1
    played = data[4] + 1
    rate = round(won/played, 2)
    c.execute("""
        UPDATE LEADERBOARD SET
        GamesWon = ?,
        GamesLost = ?,
        GamesTied = ?,
        GamesPlayed = ?,
        WiningRate = ?
        WHERE firstname = ?;""",(won, lost, tied, played, rate, USER))
    conn.commit()
    conn.close()
    return
# Function to change to LIGHT theme in LeaderBoard
def changeLight():
    boardb.configure(bg='white')
    board.configure(bg='white')
    light.configure(bg='black', fg='white')
    dark.configure(bg='black', fg='white')
    return
```

```
# Function to change to DARK theme in LeaderBoard

def changeDark():

    boardb.configure(bg='black')

    board.configure(bg='black')

    light.configure(bg='white', fg='black')

    dark.configure(bg='white', fg='black')

    return

# Function to exit all the existing windows

def exitall():

    boardb.quit()

    root.quit()

    # Function to construct the Leader Board

def getLeaderboard():

    global boardb, board, light, dark

    boardb = Tk()

    boardb.title("Leader Board :)")

    boardb.configure(bg = 'white')

    board = Frame(boardb, bg  ='white', padx = 10, pady = 10)

    board.grid(row = 0, column = 0, columnspan = 3)

    conn = sqlite3.connect('LeaderBoard.db')

    c = conn.cursor()

    c.execute("SELECT * FROM LEADERBOARD ORDER BY GamesWon DESC")

    data = c.fetchall()

    rowno = 2

    Heading = Label(board, text = "Leaderboard", bg = 'brown', fg = 'white', width = 45, pady =
10, padx = 10)
```

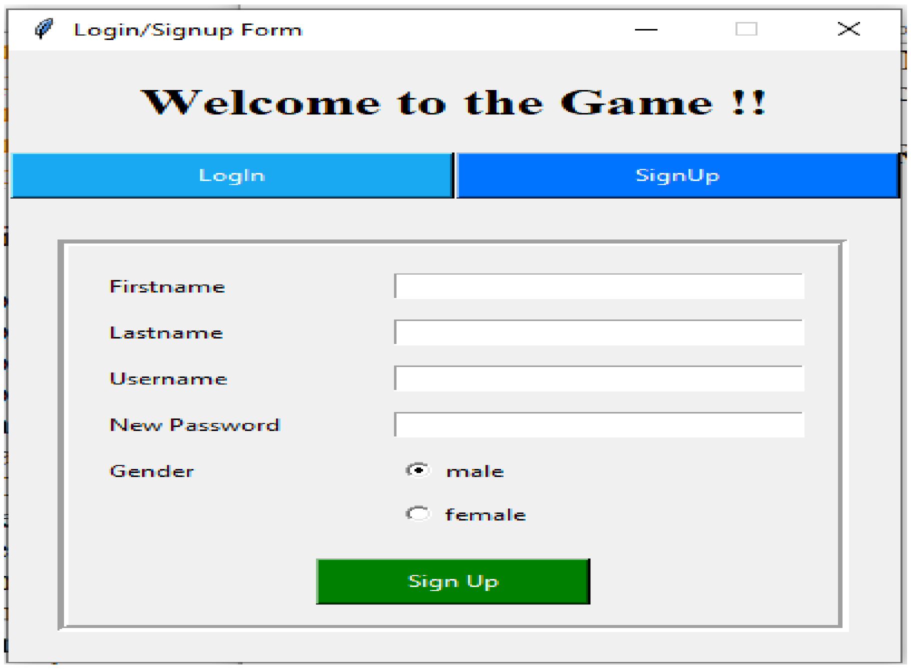
```
    Heading.config(font=("Times", 20, "bold"))

    Heading.grid(row = 0, column = 0, columnspan = 6, padx = 5, pady  = 5)

    name = Label(board, text="Name", width=15, bg='#0069b3', fg='white', padx=5, pady=5)

    name.grid(row=1, column=0, padx=2, pady=2)

    won = Label(board, text="Games Won", width=15, bg='#a617ff', fg='white', padx=5, pady=5)

    won.grid(row=1, column=1, padx=2, pady=2)

    lost = Label(board, text="Games Lost", width=15, bg='#e01717', fg='white', padx=5, pady=5)

    lost.grid(row=1, column=2, padx=2, pady=2)

    tied = Label(board, text="Games Tied", width=15, bg='#e77c00', fg='white', padx=5, pady=5)

    tied.grid(row=1, column=3, padx=2, pady=2)

    played = Label(board, text="Games Played", width=15, bg='#30b000', fg='white', padx=5,
pady=5)

    played.grid(row=1, column=4, padx=2, pady=2)

    rate = Label(board, text="Winning Rate", width=15, bg='#ff1f60', fg='white', padx=5,
pady=5)

    rate.grid(row=1, column=5, padx=2, pady=2)

for record in data:

        name = Label(board, text = record[0], width = 15, bg = '#3d7eac', fg = 'white', padx = 5,
pady = 5)

        name.grid(row = rowno, column = 0, padx = 2, pady = 3)

        won = Label(board, text=record[1], width = 15, bg = '#b846ff', fg = 'white', padx = 5, pady
= 5)

        won.grid(row = rowno, column=1, padx = 2, pady = 3)

        lost = Label(board, text=record[2], width = 15, bg = '#e33f3f', fg = 'white', padx = 5, pady =
5)

        lost.grid(row = rowno, column=2, padx = 2, pady = 3)

        tied = Label(board, text=record[3], width = 15, bg = '#e38d2a', fg = 'white', padx = 5, pady
= 5)
```

```
        tied.grid(row = rowno, column=3, padx = 2, pady = 3)

        played = Label(board, text=record[4], width = 15, bg = '#62be40', fg = 'white', padx = 5, pady = 5)

        played.grid(row = rowno, column=4, padx = 2, pady = 3)

        rate = Label(board, text=record[5], width = 15, bg = '#ff5385', fg = 'white', padx = 5, pady = 5)

        rate.grid(row = rowno, column=5, padx = 2, pady = 3)

        rowno += 1

    conn.commit()

    conn.close()

    light = Button(boardb, text = "Light Theme", pady = 7, command = changeLight, bg = 'black', fg = 'white', width = 30)

    light.grid(row = 1, column = 0, pady = (0, 20))

    dark = Button(boardb, text="Dark Theme", pady = 7, command = changeDark, bg = 'black', fg = 'white', width = 30)

    dark.grid(row=1, column=1, pady = (0, 20))

    exit = Button(boardb, text="Exit", pady=7, command=exitall, bg='red', fg='white', width=30)

    exit.grid(row=1, column=2, pady=(0, 20))

# To print the records on the GUI -> To check the records entered in the database

def printdata():

    global fname, lname, username, password

    conn = sqlite3.connect('Users.db')

    c = conn.cursor()

    c.execute("SELECT * FROM USERS")

    data = c.fetchall()

    records = Label(root, text = data)

    records.grid(row = 3, column = 0, columnspan = 2)
    conn.commit()

    conn.close()

# printdata()

# Function to insert data in the USERS database -> called from signup

def insert():

    global fname, lname, username, password, gender

    conn1 = sqlite3.connect('Users.db')

    c = conn1.cursor()

    c.execute("INSERT INTO USERS VALUES(:fname, :lname, :username, :password, 
:gender)",

              {

                  'fname':fname.get(),

                  'lname':lname.get(),

                  'username':username.get(),

                  'password':password.get(),

                  'gender' : gender.get()

              }

              )

    conn1.commit()

    conn1.close()

    conn2 = sqlite3.connect('LeaderBoard.db')

    c2 = conn2.cursor()

    c2.execute("""

            INSERT INTO LEADERBOARD VALUES(:name, 0, 0, 0, 0, 0)""",{'name' : 
fname.get()})

    conn2.commit()
    conn2.close()
    # clear the textboxes
    fname.delete(0,END)
    lname.delete(0,END)
    username.delete(0,END)
    password.delete(0,END)
    login()
    return
loginbool = False
signupbool = False
# Login page creation
def login():
    global signupbool, sframe, loginbool, LoginBtn, SignupBtn, username, password, root
    if signupbool == True:
        sframe.destroy()
        signupbool = False
    loginbool=True
    root.geometry("400x400")
    LoginBtn.configure(bg = '#0074ff')
    SignupBtn.configure(bg='#19a8f2')
    lframe = LabelFrame(root, padx = 15, pady = 15, borderwidth=5)
    lframe.grid(row = 2, column = 0, columnspan = 2,padx = 5, pady = (30,5))
    usernameLabel = Label(lframe , text = "Username", pady = 5, anchor=W, width = 10)
    usernameLabel.grid(row = 0, column = 0, sticky=W+E, padx= (0,20))
    passwordLabel = Label(lframe, text="Password", pady=5, anchor = W, width = 10)
    passwordLabel.grid(row=1, column=0, sticky=W+E, padx= (0,20))

    username = Entry(lframe, width = 30)

    username.grid(row = 0, column=1)

    password = Entry(lframe, width = 30)

    password.grid(row = 1, column=1)

    Login = Button(lframe, text = "Log In", width = 15, padx = 5, pady = 4, bg='green', fg='white',
command = check)

    Login.grid(row = 2, column = 0, columnspan = 2, padx = 10, pady = (10,0))

    return

# Sign Up Page creation

def signup():

    global sframe, loginbool, signupbool, SignupBtn, LoginBtn, fname, lname, username,
password, root, gender

    if loginbool==True:

        lframe.destroy()

        loginbool = False

    signupbool = True

    root.geometry("400x450")

    SignupBtn.configure(bg='#0074ff')

    LoginBtn.configure(bg='#19a8f2')

    sframe = LabelFrame(root, padx=15, pady=15, borderwidth=5)

    sframe.grid(row=2, column=0, columnspan=2, padx=5, pady=(30, 5))

    fnameLabel = Label(sframe, text = "Firstname", pady=5, anchor=W, width=15)

    fnameLabel.grid(row = 0, column = 0, sticky=W + E, padx=(0, 20), pady=(0,5))

    lnameLabel = Label(sframe, text="Lastname", pady=5, anchor=W, width=15)

    lnameLabel.grid(row=1, column=0, sticky=W + E, padx=(0, 20), pady=(0,5))
    usernameLabel = Label(sframe, text="Username", pady=5, anchor=W, width=15)

    usernameLabel.grid(row=2, column=0, sticky=W + E, padx=(0, 20), pady=(0,5))

    passwordLabel = Label(sframe, text="New Password", pady=5, anchor=W, width=15)

    passwordLabel.grid(row=3, column=0, sticky=W + E, padx=(0, 20), pady=(0,5))

    genderLabel = Label(sframe, text="Gender", pady=5, anchor=W, width=15)

    genderLabel.grid(row=4, column=0, sticky=W + E, padx=(0, 20), pady=(0, 5))

    fname = Entry(sframe, width=30)

    fname.grid(row = 0, column=1, pady=(0,5))

    lname = Entry(sframe, width=30)

    lname.grid(row=1, column=1, pady=(0,5))

    username = Entry(sframe, width=30)

    username.grid(row=2, column=1, pady=(0,5))

    password = Entry(sframe, width=30)

    password.grid(row=3, column=1, pady=(0,5))

    gender = StringVar()

    gender.set("Male")

    Radiobutton(sframe, text = "male", variable = gender, value = "Male", anchor = W).grid(row
= 4, column = 1, pady=(0,5), sticky = W+E)

    Radiobutton(sframe, text="female", variable=gender, value="Female", anchor =
W).grid(row=5, column=1, pady=(0, 5), sticky = W+E)

    Signup = Button(sframe, text="Sign Up", width=15, padx=5, pady=5, bg='green', fg='white',
command = insert)

    Signup.grid(row=6, column=0, columnspan=2, padx=10, pady=(15, 0))

    return

# WELCOME LABEL

WelcomeLabel = Label(root, text = "Welcome to the Game !!", pady = 20)
```

```
WelcomeLabel.config(font=("Times", 20, "bold"))

WelcomeLabel.grid(row=0, column=0, columnspan = 2)

# LOGIN BUTTON

LoginBtn = Button(root, text = "LogIn", command = login, pady = 5, width=27, bg='#19a8f2',
fg='white')

LoginBtn.grid(row = 1, column = 0, padx = 1)

# SIGN UP BUTTON

SignupBtn = Button(root, text = "SignUp", command = signup, pady = 5, width=27,
bg='#19a8f2', fg='white')

SignupBtn.grid(row = 1, column = 1)

root.mainloop()
```

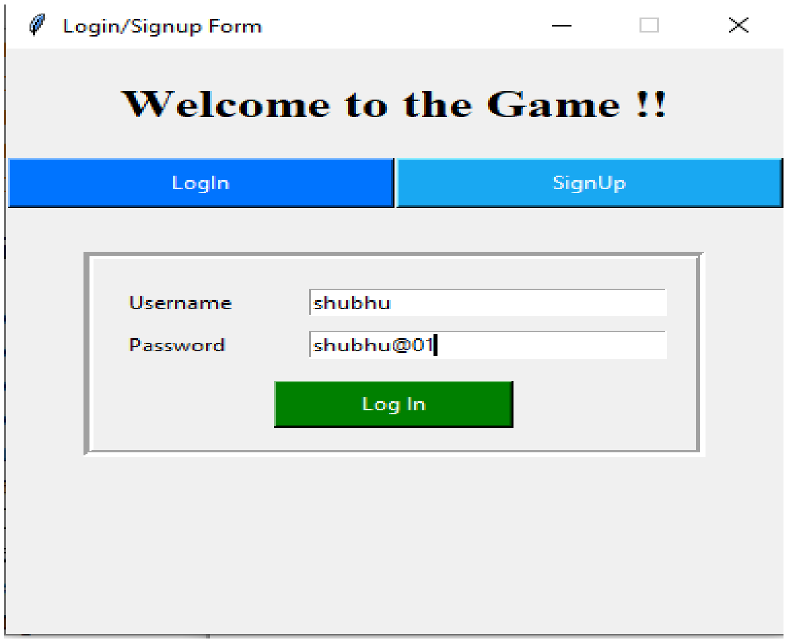
6. Results

6.1 Screenshots:



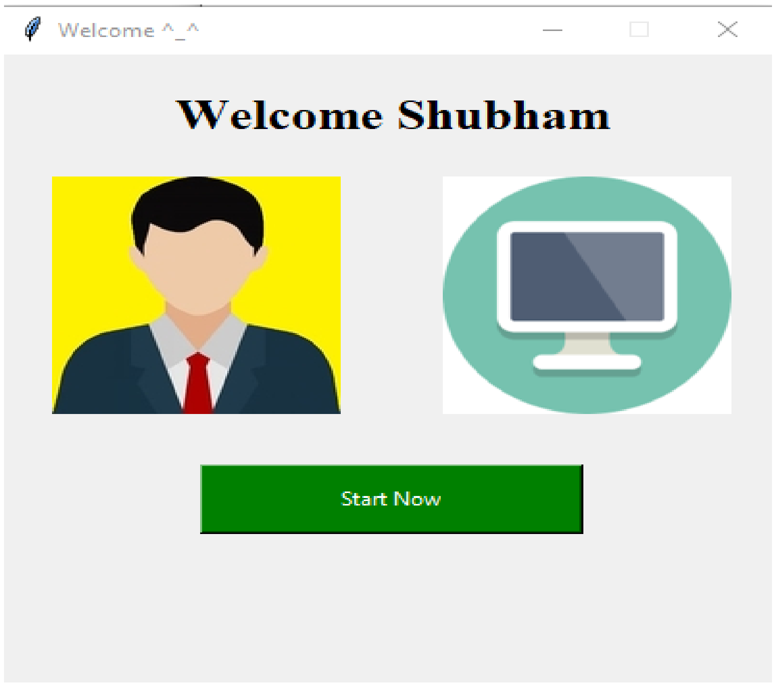
Firstname
Lastname
Username
New Password
Gender [✓ male] female

Sign Up Page

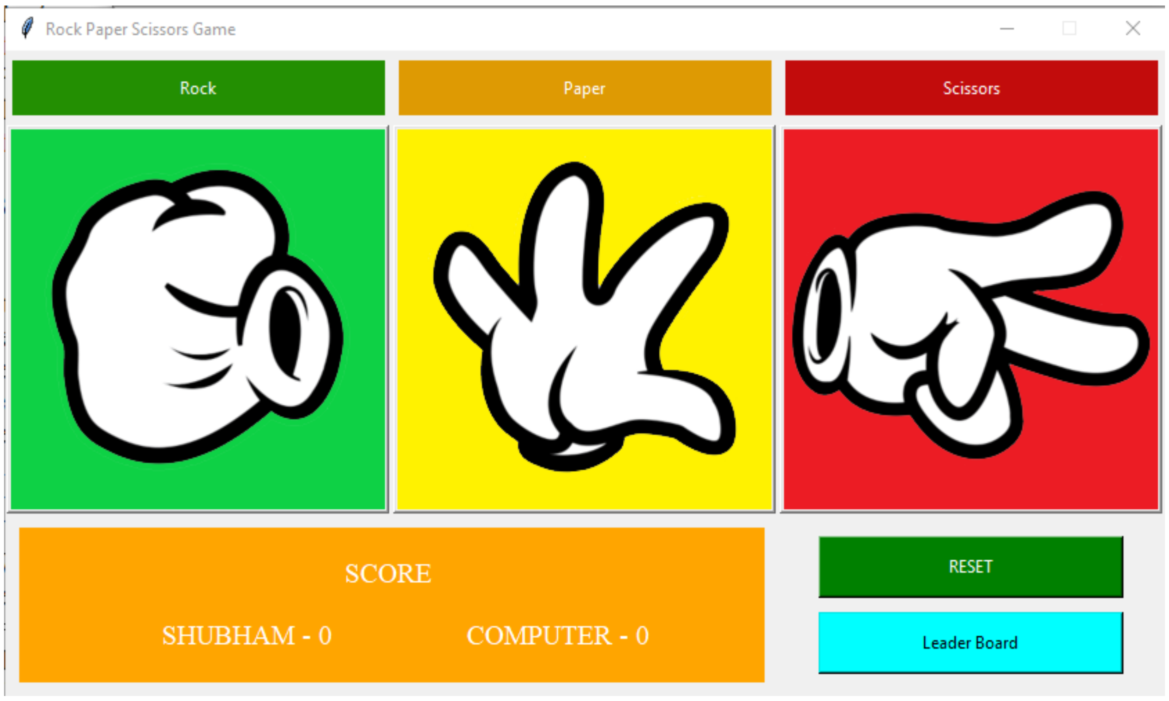


Username	shubhu
Password	shubhu@01

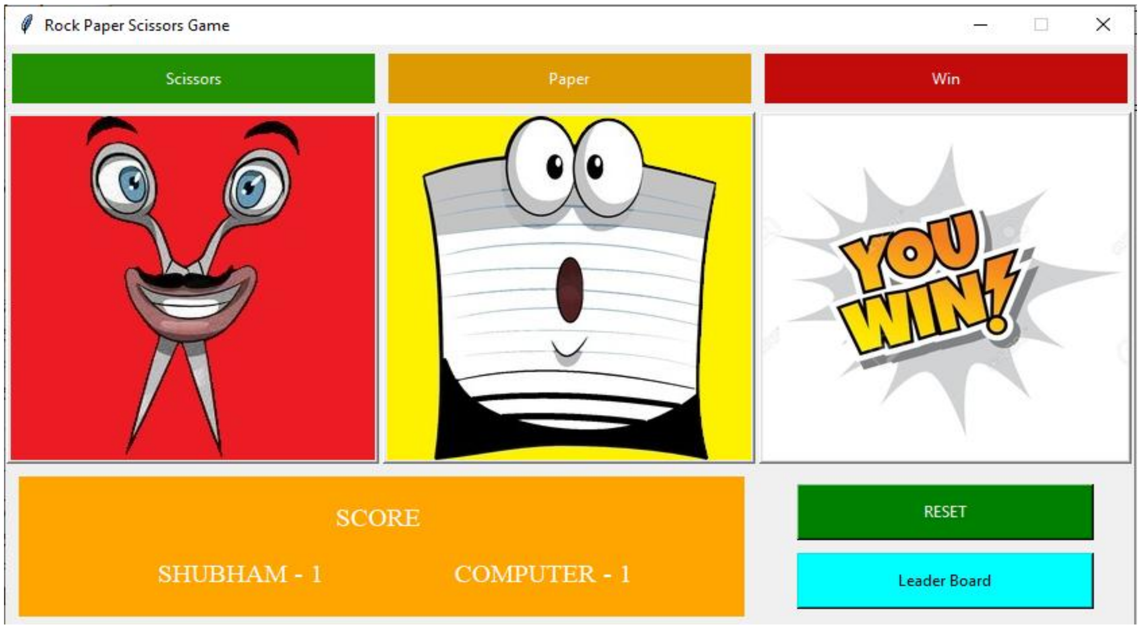
Login Page



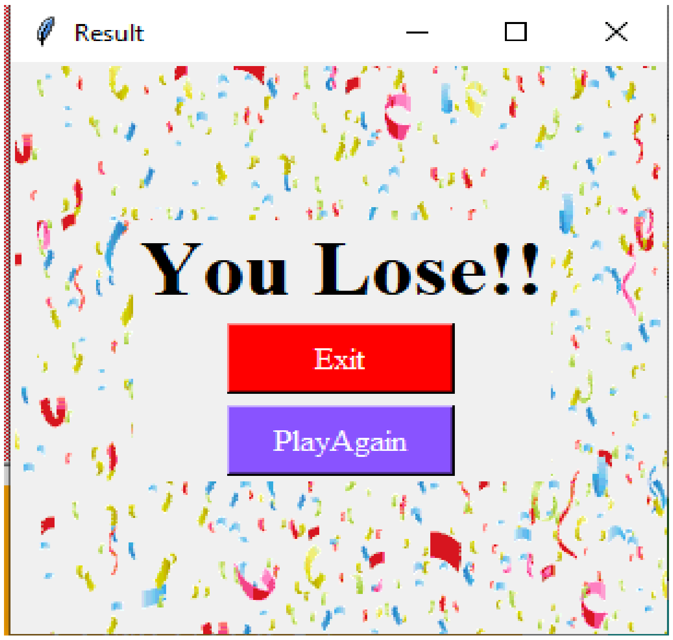
Welcome Page



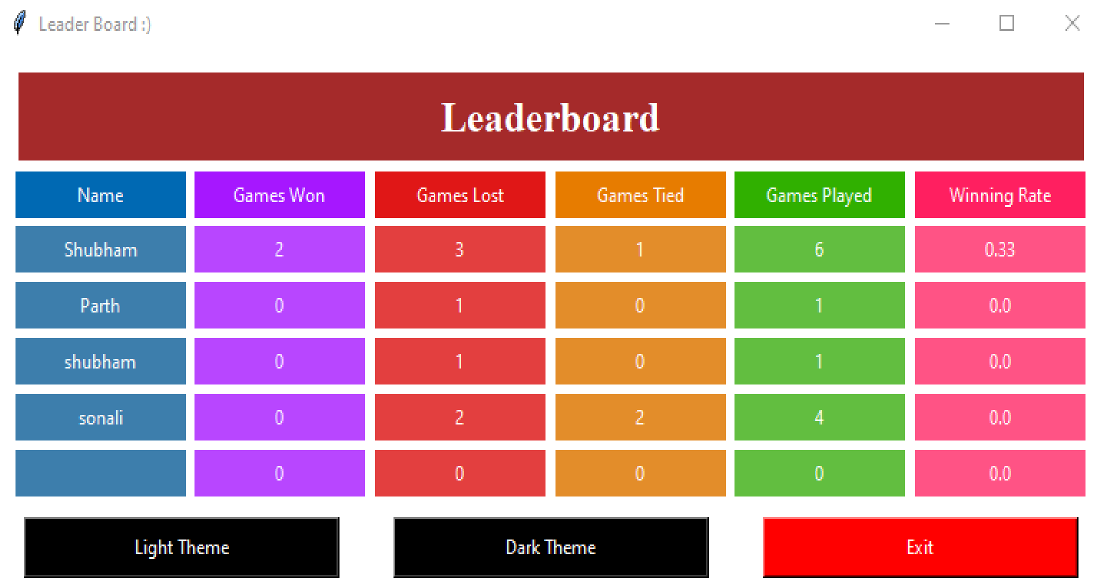
Game Page



Game Page (While palying)



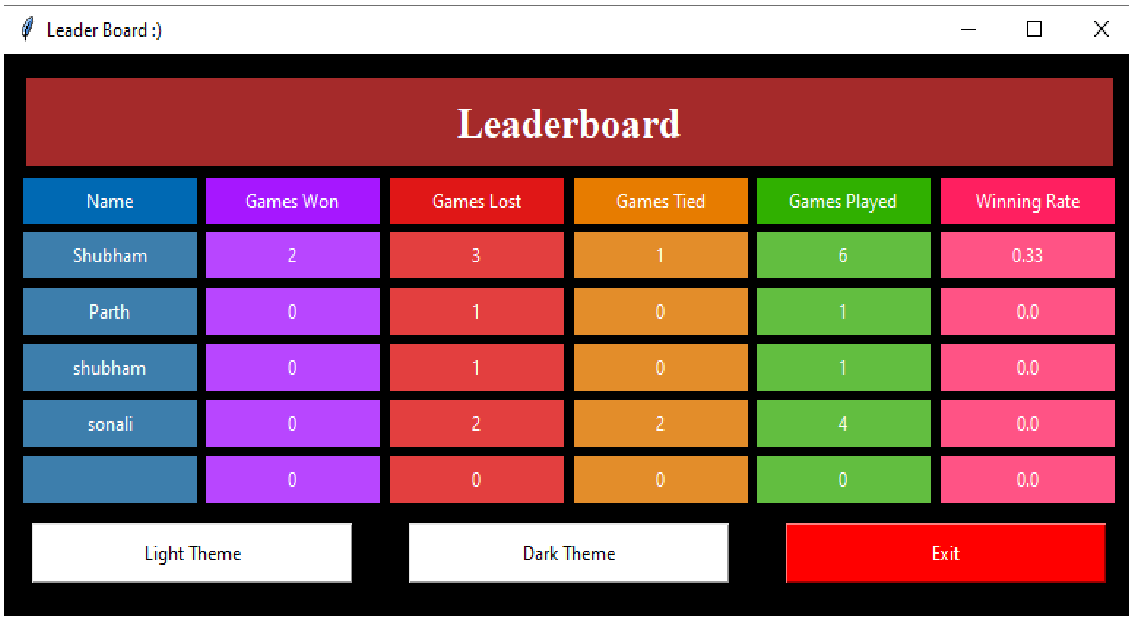
Result page



Leaderboard

Name	Games Won	Games Lost	Games Tied	Games Played	Winning Rate
Shubham	2	3	1	6	0.33
Parth	0	1	0	1	0.0
shubham	0	1	0	1	0.0
sonali	0	2	2	4	0.0
	0	0	0	0	0.0

Leaderboard page

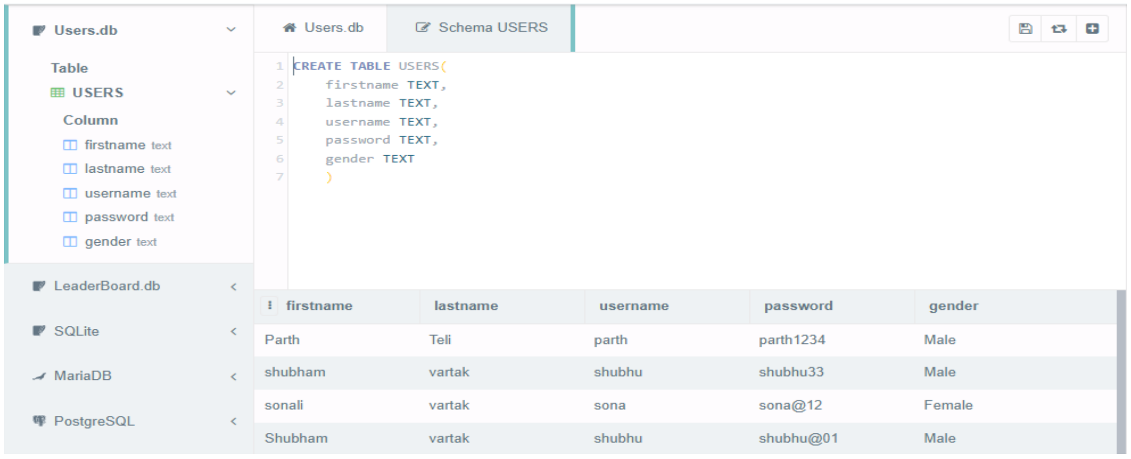


Leaderboard

Name	Games Won	Games Lost	Games Tied	Games Played	Winning Rate
Shubham	2	3	1	6	0.33
Parth	0	1	0	1	0.0
shubham	0	1	0	1	0.0
sonali	0	2	2	4	0.0
	0	0	0	0	0.0

Leaderboard In dark theme.

6.2 Reports:

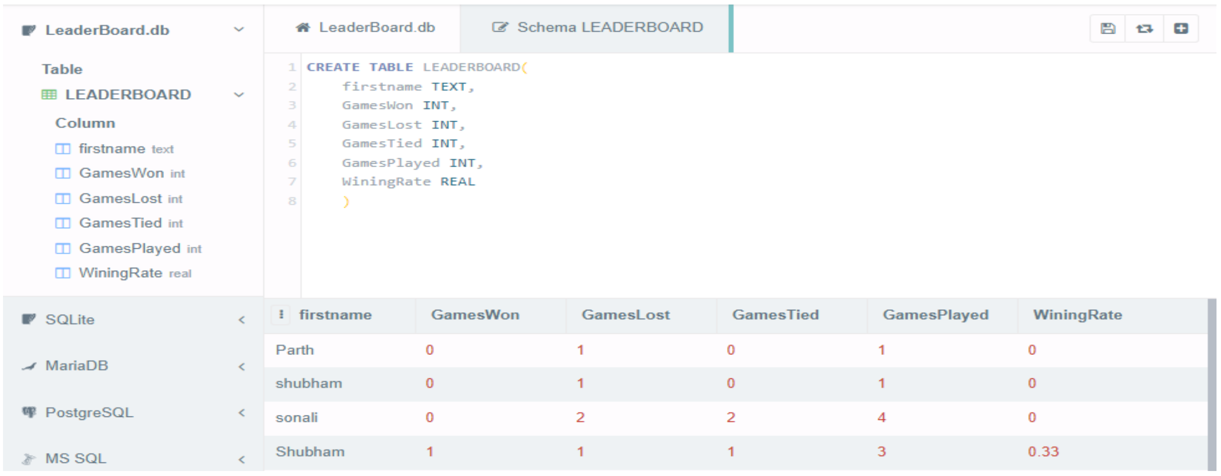


```
CREATE TABLE USERS(
    firstname TEXT,
    lastname TEXT,
    username TEXT,
    password TEXT,
    gender TEXT
    )
```

firstname	lastname	username	password	gender
Parth	Teli	parth	parth1234	Male
shubham	vartak	shubhu	shubhu33	Male
sonali	vartak	sona	sona@12	Female
Shubham	vartak	shubhu	shubhu@01	Male

6.2.1- Users Table

- Users table is created where all the user players details has been stored.
- This table contains Firstname, Lastname, Username, Password, Gender.
- This data is retrieved from database through Sqlite Online.



```
CREATE TABLE LEADERBOARD(
    firstname TEXT,
    GamesWon INT,
    GamesLost INT,
    GamesTied INT,
    GamesPlayed INT,
    WiningRate REAL
    )
```

firstname	GamesWon	GamesLost	GamesTied	GamesPlayed	WiningRate
Parth	0	1	0	1	0
shubham	0	1	0	1	0
sonali	0	2	2	4	0
Shubham	1	1	1	3	0.33

6.2.2 – Leaderboard table.

- Leaderboard table is created where all the user players scorces has been stored.
- This table contains Firstname, Gameswon, GameLost, GameTied, GamePlayed, Winning rate
- This data is retrieved from database through Sqlite Online.

7. Future Enhancement

- This project can be enhanced further by adding multiplayer options where you can play with your friends.
- More exciting themes can be added to be more interactive.
- Also we can add prizes for winning players.

8. Conclusion

- Game is played between Computer(System) and user, so it can also be played when you are along (no need to wait for anyone).
- Also there will be leaderboard which will keeps you challenging by showing number of matches won, number of matches losed.
- Also it can be played anywhere to relax and refresh mind by just opening in computer system.

9. References

- https://www.tutorialspoint.com/python/python_gui_programming.htm#:~:text=Tkinter%20Programming,to%20the%20Tk%20GUI%20toolkit.
- https://www.geeksforgeeks.org/python-gui-tkinter/
- https://youtu.be/EPwszp6Ecgs

10. Annexure

10.1 Figure List:

10.2 Table List: