**ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to my Computer Science teacher,**Ms Vidhya N P**, for her invaluable guidance and support throughout the completion of this project.

I am also thankful to our Principal, **Fr Johny Kanjirathinkal**, for providing me with the facilities and environment to work on this project.

Lastly, I would like to extend my heartfelt thanks to my family and

friends for their constant encouragement and motivation.

**Index**

* Certificate
* Acknowledgement
* Introduction
* Objectives of the Project
* Tools & Technologies Used
* Source Code
* Output Screenshots
* Conclusion
* Bibliography

**Introduction**

This project titled **“Scoreboard System and Hangman Game in Python”** has been developed to apply and demonstrate key programming concepts of the Class XII Computer Science syllabus, with a special focus on **file handling and data management**.

The main component of the project is the **Scoreboard System**. It allows players’ names and scores to be recorded, stored, and retrieved using text files. Each time a game is played, the scoreboard is updated with the latest results. The system is capable of checking if a player already exists in the file and then updating their score accordingly, or else creating a new record for a new player. An additional feature of the scoreboard is its **sorting functionality**, which arranges the players’ scores in descending order. This makes it easy to identify the top performers, much like a real leaderboard in competitive games. Through this functionality, the project highlights the practical application of **file operations, searching, updating records, and sorting algorithms**.

The second component is the **Hangman Game**, a word-guessing program that serves as the source of scores for the scoreboard. The game uses Python’s **loops, conditionals, and string operations** to provide an interactive experience, and after every match, the results are transferred to the scoreboard system.

Together, the two modules form a complete project that combines **entertainment with persistent record-keeping**. More importantly, the project emphasizes how Python’s file handling can be used to design small-scale applications that manage data efficiently while also being user-friendly.

**Objectives of the Project**

* To design and implement a **file-based scoreboard system** that records and maintains players’ scores.
* To provide functionality to **update existing records** and create new records for new players.
* To implement a **sorting feature** that arranges scores in descending order for easy identification of top players.
* To develop an interactive **Hangman Game** that generates scores for the scoreboard.
* To integrate both modules into a single project that is simple, practical, and entertaining.
* To strengthen problem-solving and programming skills through the use of **functions, loops, conditionals, strings, and file handling** in Python.

**Tools & Technologies Used**

* Programming Language: Python 3.13
* Concepts Used: File Handling, Functions, Loops, Lists, Strings, Sorting
* IDE/Editor: VS Code
* Platform: Windows 11
* PC specs:
  + Processor:12th Gen Intel(R) Core (TM) i7-1255U
  + Memory: 16GB RAM

SOURCE CODE

**hangman.py**

"""

Hangman Game with Scoreboard System

Note: This program uses ANSI escape characters to move the cursor around

the terminal for clearing lines, overwriting text, and displaying hangman stages.

Does not work with python IDLE python! Use command prompt to execute the file

/033[A - moves cursor up

/033[B - moves cursor down

/033[C - moves cursor right

/033[D - moves cursor left

/033[E - moves cursor to beginning of next line

/033[F - moves cursor to beginning of previous line

/033[2J - clear screen

"""

import os

from random import randint

import scoreboard as sc

def clear\_screen():

    """Clears the terminal screen."""

    os.system('cls')

# Load hangman ASCII art images

hangman\_images = []

with open("hangman.txt") as f:

    stage = ""

    for line in f:

        if "#" in line:

            hangman\_images.append(stage)

            stage = ""

        else:

            stage += line

def display\_hangman(stage\_num):

    """Displays hangman stage, clearing previous lines."""

    print("\033[F" \* 9, end='')  # move cursor up 9 lines

    print(hangman\_images[stage\_num], end='')

# Difficulty settings

difficulty\_levels = {1: 2, 2: 3, 3: 4}

difficulty\_points = {1: 100, 2: 300, 3: 500}

difficulty = 1  # default

# Display game title

clear\_screen()

with open("title.txt") as f:

    for line in f:

        print(line, end='')

# Ask for username

existing\_users = sc.check\_existing\_user()

while True:

    name = input("Enter Username (less than 10 letters): ").strip()

    if len(name) > 10 or not name:

        print("Username must be 1-10 characters long.")

        continue

    if name in existing\_users:

        use\_old = input("This user already exists. Do you want to use this name? (y/n): ").lower()

        if use\_old == "y":

            break

        else:

            print("Choose a different username.")

    else:

        break

# Main game loop

while True:

    state = 0

    clear\_screen()

    # Main menu

    while True:

        clear\_screen()

        print("\n--- Main Menu ---")

        print("1. Play")

        print("2. Change Difficulty")

        print("3. Scoreboard")

        print("4. Clear Scoreboard")

        print("5. Rules")

        try:

            choice = int(input("Enter your choice: "))

            if choice not in [1,2,3,4,5]:

                print("Invalid choice. Choose 1-5.")

                input("Press Enter to continue...")

                continue

        except ValueError:

            print("Invalid input. Enter a number.")

            input("Press Enter to continue...")

            continue

        if choice == 1:

            break

        elif choice == 2:

            while True:

                try:

                    difficulty = int(input(

                        "Choose Difficulty (Points)\n"

                        "1. Easy          100\n"

                        "2. Intermediate  300\n"

                        "3. Hard          500\n"

                        "Enter choice: "

                    ))

                    if difficulty not in [1,2,3]:

                        print("Invalid difficulty. Choose 1-3.")

                        continue

                    break

                except ValueError:

                    print("Enter a number.")

            input("Press Enter to continue...")

        elif choice == 3:

            sc.disp\_scores()

            input("Press Enter to return to menu...")

        elif choice == 4:

            sc.clear\_scoreboard()

            input("Press Enter to return to menu...")

        elif choice == 5:

            print("Rules:\n"

                "1. First, pick a difficulty – your points depend on it.\n"

                "2. You’ve got 6 chances to mess up before the game’s over.\n"

                "3. Guess the word correctly and you’ll score points based on the difficulty.\n"

                "4. Lose the round and you’ll drop 50 points.\n"

                "5. Play as many rounds as you like until you quit.\n\n"

                "Good luck, and have fun!\n"

            )

            input("Press Enter to return to menu...")

    # Start game

    clear\_screen()

    print(hangman\_images[state], end='')

    # Pick a random word

    with open("wordbank.txt") as wb:

        words = wb.readlines()

        word = words[randint(0, len(words)-1)].strip().lower()

    # Decide which letters to hide

    hidden\_indices = [i for i in range(len(word))]

    for i in range(len(word) // difficulty\_levels[difficulty]):

        hidden\_indices.pop(randint(0, len(hidden\_indices)-1))

    # Guessing loop

    while True:

        # Display word and hangman

        clear\_screen()

        display\_hangman(state)

        for i in range(len(word)):

            if i in hidden\_indices:

                print("\_", end="")

            else:

                print(word[i], end="")

        print("\n")

        # Get user guess

        guess = input("Guess a letter: ").strip()

        if not guess or len(guess) != 1:

            print("Enter a single letter.")

            input("Press Enter to continue...")

            continue

        wrong\_guess = True

        for i in range(len(word)):

            if word[i] == guess and i in hidden\_indices:

                hidden\_indices.remove(i)

                wrong\_guess = False

        # Win condition

        if not hidden\_indices:

            score = difficulty\_points[difficulty]

            print("You Won! Score = %d" % score)

            sc.add\_score(name, score, won=True)

            input("Press Enter to continue...")

            break

        # Wrong guess handling

        if wrong\_guess:

            state += 1

        # Loss condition

        if state == len(hangman\_images) - 1:

            clear\_screen()

            print(hangman\_images[state])

            print("You Lost! The word was: %s" % word)

            sc.add\_score(name, -50, won=False)  # negative score for loss

            input("Press Enter to continue...")

            break

    # Continue?

    cont = input("Do you want to continue? (y/n): ").lower()

    if cont == "n":

        print("Thanks for Playing!")

        break

**scoreboard.py**

**’’’**

scoreboard module made for cli games

**’’’**

import csv

import os

file = "scoreboard.csv"

# Create the scoreboard file if it doesn't exist

if not os.path.exists(file):

    with open(file, "w", newline="") as f:

        writer = csv.writer(f)

        writer.writerow(["Player", "Score", "Games Played", "Wins", "Losses"])

def add\_score(player, score, won=True):

    """Add or update a player's score and statistics."""

    updated = False

    rows = []

    with open(file, "r", newline="") as f:

        reader = csv.reader(f)

        header = next(reader)

        for row in reader:

            if row and row[0] == player:

                row[1] = str(int(row[1]) + int(score))        # update score

                row[2] = str(int(row[2]) + 1)                 # games played

                row[3] = str(int(row[3]) + (1 if won else 0)) # wins

                row[4] = str(int(row[4]) + (0 if won else 1)) # losses

                updated = True

            rows.append(row)

    with open(file, "w", newline="") as f:

        writer = csv.writer(f)

        writer.writerow(header)

        writer.writerows(rows)

        if not updated:

            wins = 1 if won else 0

            losses = 0 if won else 1

            writer.writerow([player, score, 1, wins, losses])

def disp\_scores():

    """Display the scoreboard with ranks and statistics."""

    sort\_scores()

    with open(file, "r") as f:

        reader = csv.reader(f)

        next(reader) #skip header

        # Proper table with aligned columns

        print("+------+----------+-------+--------------+------+--------+")

        print("| Rank |  Player  | Score | Games Played | Wins | Losses |")

        print("+------+----------+-------+--------------+------+--------+")

        rank = 1

        for row in reader:

            if row:

                n, s, games, wins, losses = row

                print("| %4d | %8s | %5s | %12s | %4s | %6s |" %

                      (rank, n, s, games, wins, losses))

                rank += 1

        print("+------+----------+-------+--------------+------+--------+")

def sort(l, k):

    """Custom bubble sort by column k."""

    for i in range(len(l)):

        j = 0

        while j < len(l) - 1:

            if int(l[j][k]) < int(l[j+1][k]):  # descending by score

                l[j], l[j+1] = l[j+1], l[j]

            j += 1

    return l

def sort\_scores():

    """Sort all scores descending by score (column 1)."""

    with open(file, "r", newline="") as f:

        reader = csv.reader(f)

        header = next(reader)

        rows = list(reader)

    rows = sort(rows, 1)

    with open(file, "w", newline="") as f:

        writer = csv.writer(f)

        writer.writerow(header)

        writer.writerows(rows)

def clear\_scoreboard():

    """Clear all scores after user confirmation."""

    confirm = input("Are you sure you want to clear the scoreboard? (y/n): ").lower()

    if confirm == "y":

        with open(file, "w", newline="") as f:

            writer = csv.writer(f)

            writer.writerow(["Player", "Score", "Games Played", "Wins", "Losses"])

        print("Scoreboard cleared!")

def check\_existing\_user():

    """Return list of existing users."""

    with open(file, "r", newline="") as f:

        reader = csv.reader(f)

        next(reader)  # Skip header

        return [row[0] for row in reader if row

**ASCII art used:**

**title.txt**

**\_**

**| |**

**| |\_\_ \_\_ \_ \_ \_\_ \_\_ \_ \_ \_\_ \_\_\_ \_\_ \_ \_ \_\_**

**| '\_ \ / \_` | '\_ \ / \_` | '\_ ` \_ \ / \_` | '\_ \**

**| | | | (\_| | | | | (\_| | | | | | | (\_| | | | |**

**|\_| |\_|\\_\_,\_|\_| |\_|\\_\_, |\_| |\_| |\_|\\_\_,\_|\_| |\_|**

**\_\_/ |**

**|\_\_\_/**

**hangman.txt**

**+---+**

**| |**

**|**

**|**

**|**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**|**

**|**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**| |**

**|**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**/| |**

**|**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**/|\ |**

**|**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**/|\ |**

**/ |**

**|**

**=========**

**#**

**+---+**

**| |**

**O |**

**/|\ |**

**/ \ |**

**|**

**=========**

**#**

**Conclusion**

This project helped me apply Python programming concepts to create a working application that is both useful and interactive. The scoreboard system demonstrated the practical use of file handling, allowing records to be stored permanently, updated when needed, and displayed in a sorted order to highlight top players. This gave me insight into how data is managed in real-world applications.

The Hangman game served as the source of scores and provided an engaging way to test logic and problem-solving. Integrating it with the scoreboard showed me how multiple modules can be combined into a complete system. Overall, the project strengthened my skills in file operations, data management, and modular programming, while also improving my ability to design and debug programs.

**Bibliography**

* Computer Science With Python: Textbook For CBSE Class 12 (Preeti Arora)
* [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
* [www.w3schools.com](http://www.w3schools.com)
* [www.stackoverflow.com](http://www.stackoverflow.com)
* docs.python.org
* gist.github.com