**A**

**Report on**

**CRICKET SCORECARD**

**CS205ES-PYTHON PROGRAMMING PROJECT**

Submitted in partial fulfillment of the requirements for

The Award of the degree of

**BACHELOR OF TECHNOLOGY**

In

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

By

**G. JAI RAKESH -22K81A0585**

**Under the esteemed guidance of**

**Mrs. M. NAGA TRIVENI**

**ASSISTANT PROFESSOR**

**DEPARTMENT OF CSE**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**St. MARTIN’S ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

**NBA & NAAC A+ Accredited**

**Dhulapally, Secunderabad – 500100**

**St. MARTIN’S ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

**NBA & NAAC A+ Accredited**

**Dhulapally, Secunderabad – 500100**



**College Code: K8**

**CERTIFICATE**

This is to certify that the **PYTHON PROGRAMMING PROJECT** entitled “**CRICKET SORECARD**” is a bonafide record of independent work done by **G. JAI RAKESH(22K81A0585)** under my supervision and guidance, submitted to **St. MARTIN’S ENGINEERING COLLEGE, Hyderabad**, in partial fulfillment for the award of the Degree of Bachelor of Technology in **Computer Science and Engineering**.

**Project Internal Guide Head of Department**

Mrs. M. NAGA TRIVENI Dr. R.Ranadheer Reddy

(Dept. of CSE) (Dept. of FME)

…………………………….. …………………………………

**Principal**

(Dr. **P.SANTOSH KUMAR PATRA**)

……………………………………………..

**ABSTRACT**

The provided code is an abstract implementation of a cricket scorekeeper using the Tkinter library in Python. It presents a graphical user interface (GUI) where the user can input the number of overs for a cricket match. Once the number of overs is entered and the "Enter Overs" button is clicked, the code simulates the match by generating random scores for each ball and prompting the user to enter the actual score. The simulation keeps track of the total runs, wickets, and the progress of the match. It displays the predicted and actual scores for each ball, and at the end of each over, it shows the score and wicket count. The match summary is displayed at the end, showing the total score and the number of wickets. The GUI features a background image and uses various Tkinter widgets such as labels, entries, and buttons to create the interface.

**i**

**CONTENTS**

**ABSTRACT i**

**CHAPTER 1- INTRODUCTION 1**

**CHAPTER 2-LITERATURE SURVEY 3**

**CHAPTER 3-SYSTEM ANALYSIS 5**

**3.1 EXISTING SYSTEM 5**

**3.2 PROPOSED SYSTEM 5**

**CHAPTER 4-SYSTEM REQUIREMENT 6**

**CHAPTER 5-SYSTEM ARCHITECTURE 7**

**CHAPTER 6-** **SYSTEM IMPLEMENTATION 10**

**CHAPTER 7-SYSTEM TESTING 13**

**CHAPTER 8-OUTPUT SCREENS 15**

**CHAPTER 9-CONCLUSION 22**

**CHAPTER 10-FUTURE ENHANCEMENT 23**

**CHAPTER 11-REFERENCES 24**

**1. INTRODUCTION**

The provided code is a simple implementation of a cricket, it allows users to simulate a cricket match by inputting the number of overs and then tracking the scores and wickets throughout the match. It displays an image background and includes a label and entry field where users can enter the number of overs for the match. A button is also provided to initiate the simulation.

Upon clicking the "Enter Overs" button, the program starts the match simulation. It generates random scores for each ball and prompts the user to input the actual score. The simulation keeps track of the total runs and wickets as the match progresses. At the end of each over, a message box displays the score and wicket count. The simulation continues until all the balls for the specified number of overs are completed. Finally, a match summary is displayed, showing the total score and the number of wickets.

**1.1 INTRODUCTION TO TKINTER:**

Tkinter is a Python library widely used for creating graphical user interfaces (GUIs). The name "Tkinter" is derived from "Tk interface," as it provides a binding to the Tk GUI toolkit, originally developed for the Tcl programming language. Tkinter is included with most Python installations, making it easily accessible and convenient for developers. With Tkinter, developers can build interactive and visually appealing desktop applications by combining various widgets like buttons, labels, text fields, and more.

**1.2 WIDGETS USED:**

The code uses the following Tkinter widgets:

* Tk: It represents the main window of the application.
* Label: It displays a static text or an image.
* Entry: It provides a text entry field for the user to enter data.
* Button: It creates a button that can trigger a specific action when clicked.

The code also uses the following non-widget components:

* PhotoImage: It represents an image that can be displayed in a Tkinter application.

**1.3 PROPOSED METHODOLOGY:**

1. Importing Libraries: The code begins by importing the required libraries for the GUI application. It imports the Tkinter module to create the graphical user interface components, random for generating random ball scores, and specific modules from tkinter to use simple dialogs and message boxes.
2. Define the simulate\_match() Function: This function is the core of the application. It simulates a cricket match based on the user's input for the number of overs. It initializes variables to keep track of the match statistics, such as total\_runs, wickets, overall\_runs, and overall\_wickets.
3. Input Validation and Simulation: Inside the simulate\_match() function, it runs a loop to simulate each ball of the match. For each ball, it generates a random prediction of the score using random.randint(0, 6). It then prompts the user to input the actual score of the ball using a Simpledialog pop-up. The user can input a score between 0 and 6, or use -1 for a wicket and -2 for a wide/no-ball. If the user provides an invalid input, it shows an error message and prompts for the score again.
4. Over and Match Summary: After each over (every 6 balls), it shows a message box with the score and wicket summary of that over. It updates the overall match statistics accordingly.
5. GUI Creation: The code creates a graphical user interface using tkinter. It sets up the main window with the title "Cricket Scorekeeper" and a fixed size (400x300) with disabled resizing. It loads an image as the background and places it on the window. It also creates a label and an entry widget for the user to input the number of overs they want to simulate. A "Enter Overs" button is provided to initiate the simulation.
6. Main Event Loop: The code starts the main event loop using window.mainloop(). This loop keeps the application running and responsive to user interactions.

**2. LITERATURE SURVEY**

Python is a high-level, interpreted programming language that emphasizes simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes code readability and follows a clean and elegant syntax. Python's extensive standard library and third-party packages provide a wide range of functionalities, making it suitable for various applications.

**2.1 APPLICATIONS:**

* Web Development: Python offers several frameworks like Django, Flask, and Pyramid that simplify web development tasks.
* Data Science and Machine Learning: Python has become the de facto language for data science and machine learning. Packages like NumPy, pandas, and SciPy provide powerful tools for data manipulation and analysis.
* Automation and Scripting: Python's simplicity and cross-platform compatibility make it an excellent choice for automation and scripting tasks.
* Desktop Applications: Python's cross-platform support and GUI frameworks like Tkinter, PyQt, and wxPython make it suitable for developing desktop applications.

**2.2VERSIONS OF PYTHON:**

There are currently two major versions of Python in use: Python 2 and Python 3.

* Python 2 was the first version of Python that was widely used, and it is still used today in some legacy systems. However, it is no longer being actively developed and will no longer receive updates or security patches after January 1, 2020.
* Python 3 is the current version of Python and is the version that is recommended for new projects. Python 3 is not backward-compatible with Python 2, which means that code written in Python 2 may need to be updated to work with Python 3.
* Python 3 has several advantages over Python 2, including improved Unicode support, better syntax and error handling, and improved performance. The latest version of Python 3 is Python 3.9.6, which was released on June 28, 2021.

Here are the major versions of Python and their significant developments:

* Python 1.0 (1994): The first stable release of Python. - Python 2.0 (2000): Introduced list comprehensions, a garbage collector, and support for Unicode.
* Python 2.2 (2001): Introduced the "with" statement for file handling and support for iterators. - Python 2.3 (2003): Introduced the "enumerate" function, the "set" data type, and support for generators.
* Python 2.5 (2006): Introduced the "with" statement for use with arbitrary context managers and support for conditional expressions
* Python 2.6 (2008): Introduced the "collections" module and support for "itertools" module.
* Python 2.7 (2010): Introduced the "print" function and support for dictionary comprehensions. - Python 3.0 (2008): A major release that introduced significant changes to the language, including improved Unicode support, a new I/O library, and changes to the print statement.
* Python 3.1 (2009): Introduced support for the "bytearray" data type.
* Python 3.2 (2011): Introduced the "os.scandir()" function and support for the "pickle" protocol3.
* Python 3.4 (2014): Introduced the "asyncio" module and support for the "enum" module. - Python 3.5 (2015): Introduced support for the "async" and "await" keywords and the "typing" module.
* Python 3.6 (2016): Introduced support for formatted string literals and the "f-strings" syntax.
* - Python 3.7 (2018): Introduced support for the "asyncio.run()" function .
* - Python 3.9 (2020): Introduced support for dictionary union operators and the "zoneinfo" module.

**3. SYSTEM ANALYSIS**

**3.1 EXISTING AND DISADVANTAGES:**

The program utilizes Tkinter to create a GUI with a window, labels, entry field, and button. The GUI provides a visual interface for the user to interact with the program. The user can enter the number of overs for the cricket match using the entry field provided in the GUI. Once the user enters the number of overs and clicks the "Enter Overs" button, the program simulates the cricket match. It generates random scores for each ball and prompts the user to input the actual score. The program keeps track of the total runs and wickets as the match progresses. After every over, a message box pops up displaying the score and wicket count. At the end of the simulation, a final match summary is displayed in a message box.

**3.2 PROPOSED AND ADVANTAGES:**

Enhance the program by allowing the user to select teams before starting the match. This could involve creating a team selection screen where users can choose players for each team and customize team names. Implement a feature to display real-time score updates during the match.

Create a separate scoreboard display within the GUI to show the current score, wickets, and other match statistics. Improve the input validation by providing clear error messages and allowing the user to correct their input without restarting the simulation. Allow users to save the current match progress and load saved matches for future reference.

**4**. **SYSTEM REQUIREMENT**

**4.1 SOFTWARE REQUIREMENTS:**

Operating system: Windows

IDE: python 3.11

GUI: Tkinter

**4.2 HARDWARE REQUIREMENTS:**

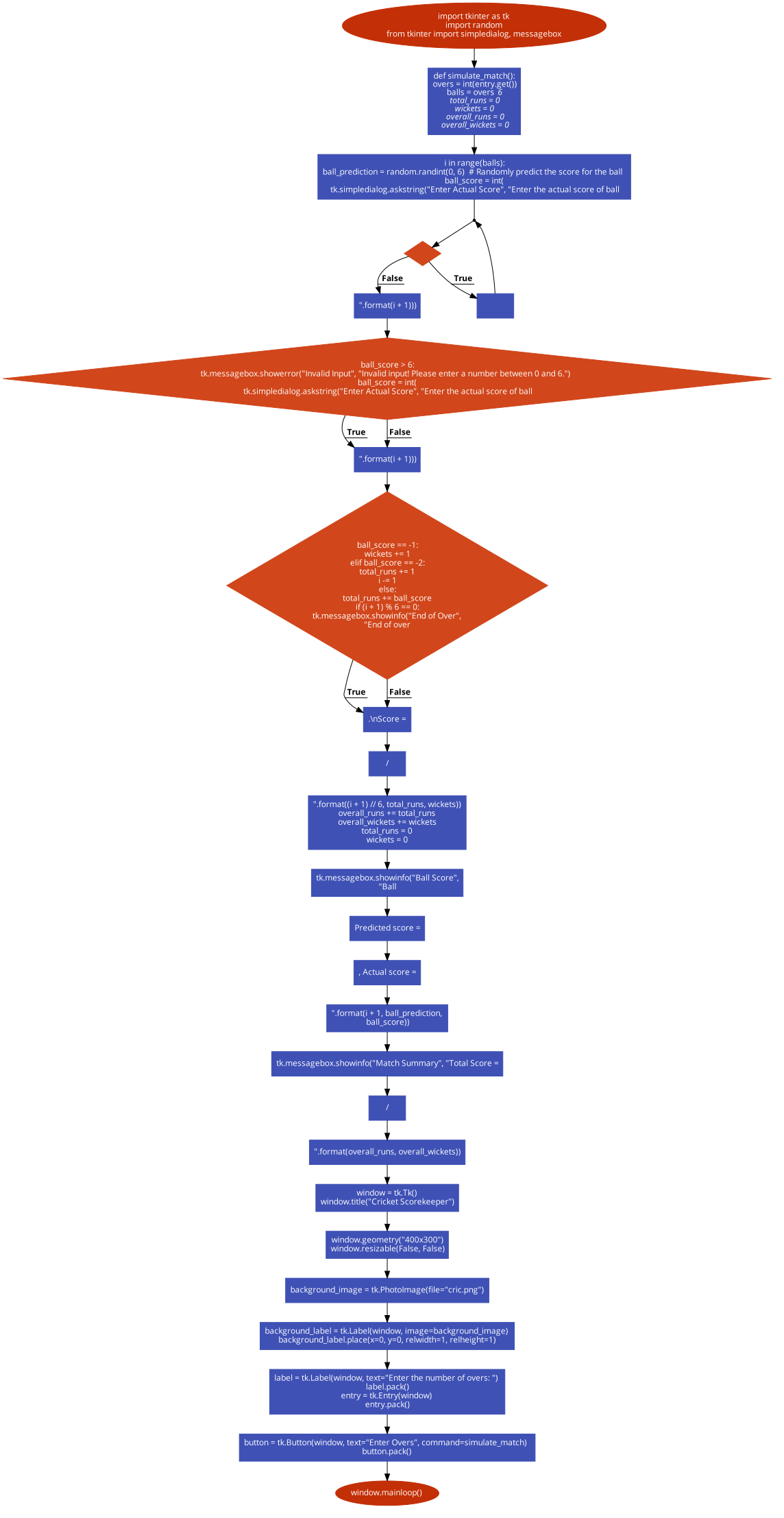
Processor: Intel i3

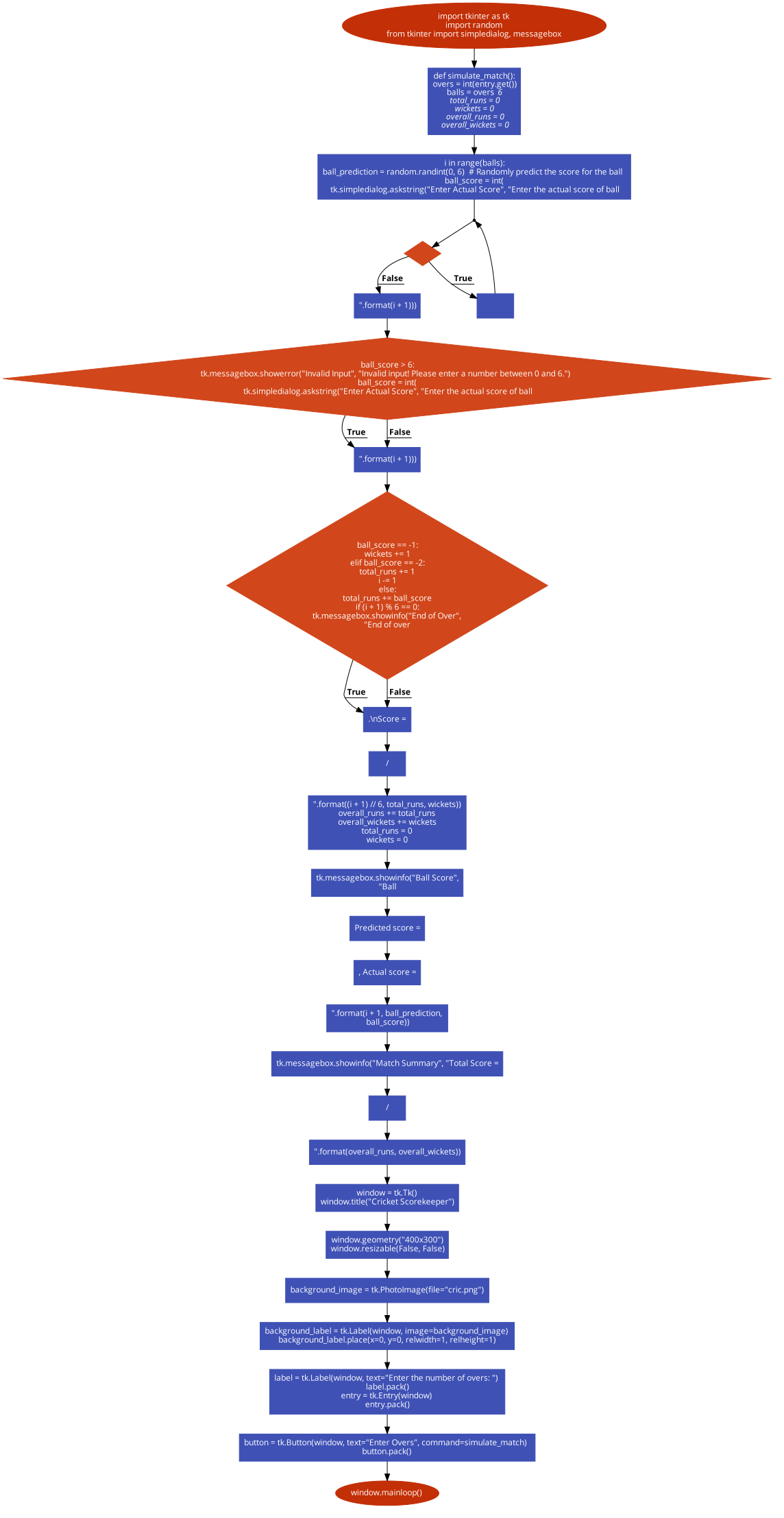
Hard disk:500

gb Ram:4gb

**5. SYSTEM ARCHITECTURE**

**5.1 FLOWCHART AND IT’S DESCRIPTION**





The flowchart represents a "Cricket Scorekeeper" program. It starts by prompting the user to input the number of overs for the match. After clicking "Enter Overs," the program initiates the match simulation. Inside the simulation function, each ball is simulated, generating a random prediction and prompting the user for the actual score. The program validates the input, updates match statistics, and displays summaries. Finally, it shows the match summary with the total score and wickets.

**5.2 ALGORITHM**

STEP-1 Import the required modules and libraries.

STEP- 2 Define the **simulate\_match()** function:

* 1. Get the number of overs from the user's input.
  2. Calculate the total number of balls based on the number of overs (6 balls per over).
  3. Initialize variables for total runs, wickets, overall runs, and overall wickets.
  4. Start a loop to simulate each ball: Generate a random predicted score for the ball. ii. Prompt the user to enter the actual score for the ball. iii. Validate the input: - If the input is greater than 6, display an error message and prompt the user again for valid input. - If the input is -1, increment the wickets count. - If the input is -2, increment the total runs count and decrement the ball index to repeat the current ball. - Otherwise, add the input to the total runs count. iv. Check if an over has been completed: - If the current ball index is divisible by 6, display an end of over message box with the score and wicket count. - Update the overall runs and wickets counts. - Reset the total runs and wickets counts for the next over. v. Display a ball score message box with the predicted and actual scores for the ball.
  5. Display a match summary message box with the overall score and wicket count.

STEP-3 Create the main window using Tkinter.

STEP-4 Set the window title and dimensions.

STEP-5 Load the background image for the GUI.

STEP-6 Create a label and entry field for the user to input the number of overs.

STEP-7 Create a button to initiate the match simulation.

STEP-8 Start the main event loop with **window.mainloop()**.

**6. SYSTEM IMPLEMENTATION**

**6.1 SOURCE CODE:**

import tkinter as tk

import random

from tkinter import simpledialog, messagebox

def simulate\_match():

overs = int(entry.get())

balls = overs \* 6

total\_runs = 0

wickets = 0

overall\_runs = 0

overall\_wickets = 0

for i in range(balls):

ball\_prediction = random.randint(0, 6) # Randomly predict the score for the ball

ball\_score = int(

tk.simpledialog.askstring("Enter Actual Score", "Enter the actual score of ball {}: ".format(i + 1)))

if ball\_score > 6:

tk.messagebox.showerror("Invalid Input", "Invalid input! Please enter a number between 0 and 6.")

ball\_score = int(

tk.simpledialog.askstring("Enter Actual Score", "Enter the actual score of ball {}: ".format(i + 1)))

if ball\_score == -1:

wickets += 1

elif ball\_score == -2:

total\_runs += 1

i -= 1

else:

total\_runs += ball\_score

if (i + 1) % 6 == 0:

tk.messagebox.showinfo("End of Over",

"End of over {}.\nScore = {}/{}".format((i + 1) // 6, total\_runs, wickets))

overall\_runs += total\_runs

overall\_wickets += wickets

total\_runs = 0

wickets = 0

tk.messagebox.showinfo("Ball Score",

"Ball {}: Predicted score = {}, Actual score = {}".format(i + 1, ball\_prediction,

ball\_score))

tk.messagebox.showinfo("Match Summary", "Total Score = {}/{}".format(overall\_runs, overall\_wickets))

# Create the main window

window = tk.Tk()

window.title("Cricket Scorekeeper")

# Set the window size and disable resizing

window.geometry("400x300")

window.resizable(False, False)

background\_image = tk.PhotoImage(file="cric.png")

# Create a label with the image and place it at the back of the window

background\_label = tk.Label(window, image=background\_image)

background\_label.place(x=0, y=0, relwidth=1, relheight=1)

# Create and place the label and entry for overs input

label = tk.Label(window, text="Enter the number of overs: ")

label.pack()

entry = tk.Entry(window)

entry.pack()

# Create and place the button to start the simulation

button = tk.Button(window, text="Enter Overs", command=simulate\_match)

button.pack()

# Start the main event loop

window.mainloop()

**6.2 MODULUES USED:**

* tkinter
* random
* simpledialog (from tkinter)
* messagebox (from tkinter)

**7. SYSTEM TESTING**

**7.1 Test Case 1**

**Input:** Number of overs = 1

* For each ball, enter the actual score as follows:
* Ball 1: Predicted score = 5, Actual score = 1
* Ball 2: Predicted score = 5, Actual score = 2
* Ball 3: Predicted score = 6, Actual score = 3
* Ball 4: Predicted score = 3, Actual score = 5
* Ball 5: Predicted score = 4, Actual score = -1
* Ball 6: Predicted score = 6, Actual score = 6

**Expected Output:**

* After each over, a message box will display the score and wickets (e.g., End of over 1. Score = 17/1).
* At the end of the simulation, a final message box will display the total score and wickets (e.g., Total Score = 17/1).

**7.2 Test case 2**

**Input:** Number of overs = 3

* For each ball, enter the actual score as follows:
* Ball 1: Predicted score = 4, Actual score = 4
* Ball 2: Predicted score = 2, Actual score = 6
* Ball 3: Predicted score = 4, Actual score = -2
* Ball 4: Predicted score = 5, Actual score = 6
* Ball 5: Predicted score = 4, Actual score = -1
* Ball 6: Predicted score = 0, Actual score = 2
* Ball 7: Predicted score = 3, Actual score = 3
* Ball 8: Predicted score = 4, Actual score = 6
* Ball 9: Predicted score = 5, Actual score = -1
* Ball 10: Predicted score = 4, Actual score = 2
* Ball 11: Predicted score = 0, Actual score = 6
* Ball 12: Predicted score = 5, Actual score = 3
* Ball 13: Predicted score = 2, Actual score = -1
* Ball 14: Predicted score = 1, Actual score = 6
* Ball 15: Predicted score = 4, Actual score = 2
* Ball 16: Predicted score = 6, Actual score = 4
* Ball 17: Predicted score =4, Actual score = 6
* Ball 18: Predicted score = 6, Actual score = -1

**Expected output:**

* After each over, a message box will display the score and wickets (e.g., End of over 1. Score = 19/1).
* At the end of the simulation, a final message box will display the total score and wickets (e.g., Total Score = 57/4).

**7.3 Test case 3**

**Input:** Number of overs = 1

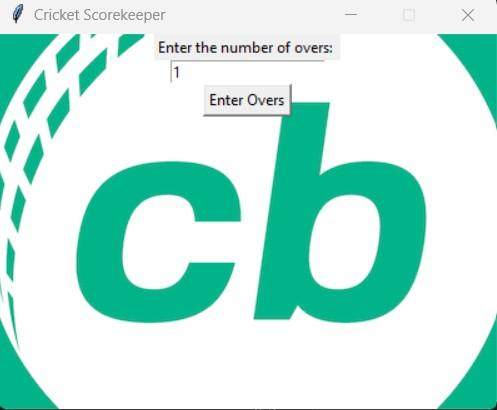
* For each ball, enter the actual score as follows:
* Ball 1: Predicted score = 6, Actual score = 6
* Ball 2: Predicted score = 6, Actual score = 6
* Ball 3: Predicted score = 5, Actual score = 6
* Ball 4: Predicted score = 6, Actual score = 6
* Ball 5: Predicted score = 4, Actual score = 6
* Ball 6: Predicted score = 3, Actual score = 6

**Expected Output:**

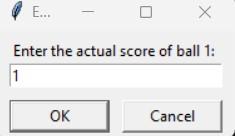
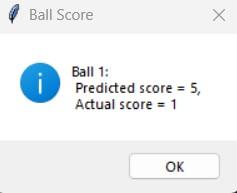
* After each over, a message box will display the score and wickets (e.g., End of over 1. Score = 36/0).
* At the end of the simulation, a final message box will display the total score and wickets (e.g., Total Score = 36/0)

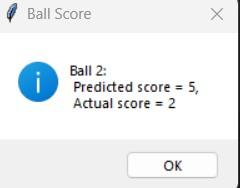
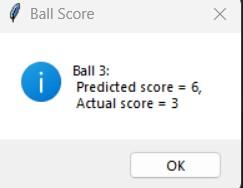
**8. OUTPUT SCREENS**

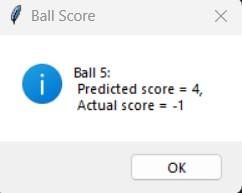
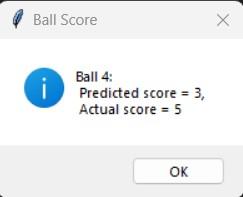
**8.1** **CRICKET SCORECARD SYSTEM**

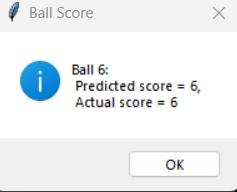


**Fig No 8.1:** Enter no of overs.

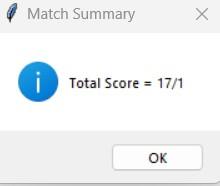






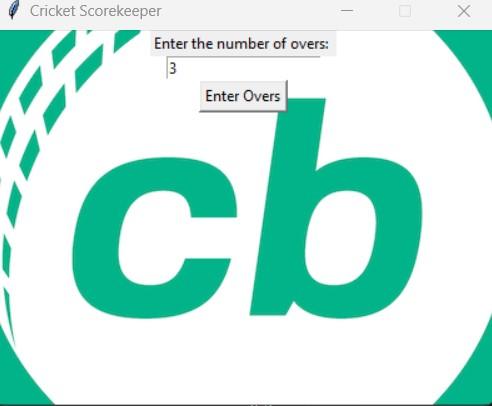


**Fig No 8.2:** Enter the score for each ball.

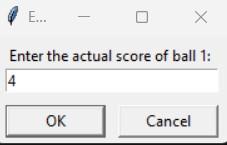
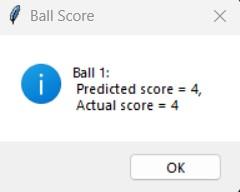


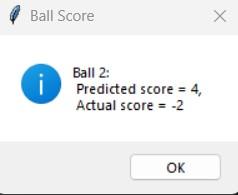
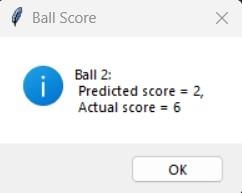
**Fig No 8.3:** Match summary will be displayed.

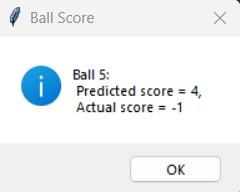
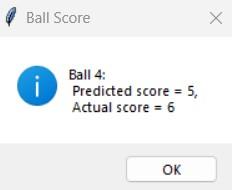
**8.2** **CRICKET SCORECARD SYSTEM**

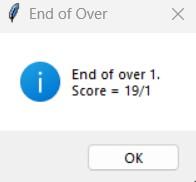
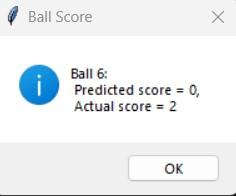


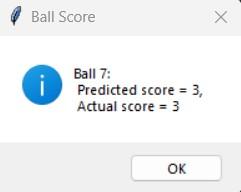
**Fig No 8.4:** Enter no of overs.

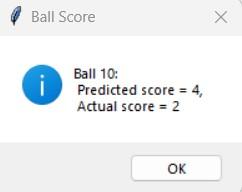


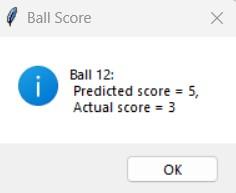


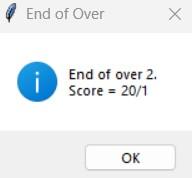
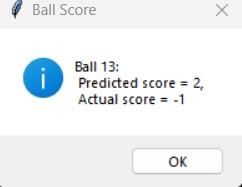




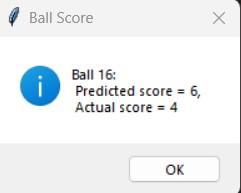


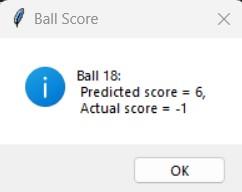
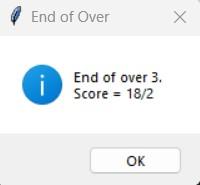


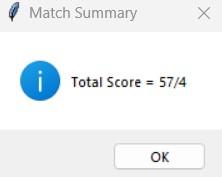




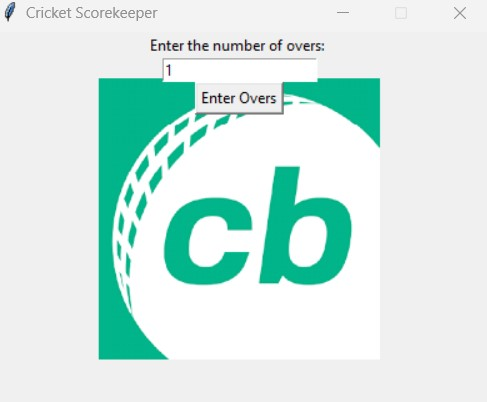


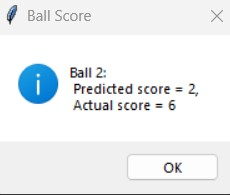
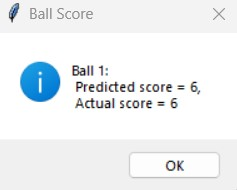


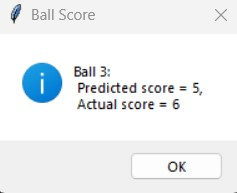


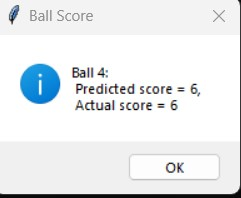
****

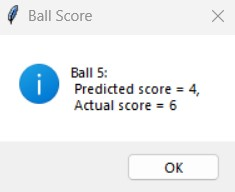
**Fig No 8.5:** Match summary will be displayed.

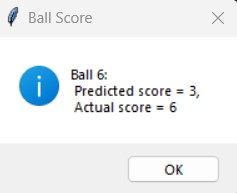
**8.3** **CRICKET SCORECARD SYSTEM**

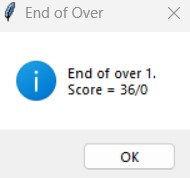












**Fig No 8.6:** Match summary will be displayed.

**9. CONCLUSION**

In conclusion, the provided code offers a basic implementation of a cricket scorekeeper using the Tkinter library in Python. It presents a graphical user interface (GUI) where users can input the number of overs for a cricket match and then simulate the match by tracking scores and wickets.

The code demonstrates the usage of Tkinter widgets such as labels, entry fields, buttons, and message boxes to create an interactive interface. It incorporates error handling to validate user input and ensure the entered scores are within the expected range. Additionally, the code utilizes random number generation to simulate the predicted scores for each ball. While the existing code provides a foundation for simulating and tracking scores in a cricket match, there is room for further enhancements and additional features. Proposed improvements can include team selection, real-time score updates, player statistics, enhanced graphics, and customizable match settings. By building upon this code and incorporating the suggested improvements, you can create a more comprehensive and engaging cricket scorekeeping application with a user-friendly interface and enhanced functionality.

**10. FUTURE ENHANCEMENTS**

Player Profiles and Statistics: Develop a system to create and maintain player profiles with detailed statistics such as batting average, bowling average, highest score, and best bowling figures. This will allow users to track individual player performance over multiple matches. Match History and Archives: Implement a feature to store and display match history and archives. Users can view past match results, scores, and player performances, providing a comprehensive record of the team's progress. Graphical Representation: Create visual graphs and charts to represent match progress, scoring rates, and player statistics. This will provide a more intuitive and visually appealing way to analyze and interpret match data. Multiplayer Mode: Enable multiplayer functionality to allow users to compete against each other in a virtual cricket match. This can be implemented through network connectivity, allowing players to take turns entering their scores and compete for the highest total. Commentary and Live Updates: Integrate a commentary system to provide live updates and descriptions of the match. This can be done by leveraging APIs to fetch real-time match data or by using pre-defined commentaries based on the match situation. Customizable GUI Themes: Allow users to customize the appearance of the GUI by selecting different themes, color schemes, or background images. This will enhance the personalization options and improve the user experience. Scorecard Generation: Implement a feature to generate a printable scorecard at the end of the match. The scorecard can include details such as batting and bowling statistics, partnerships, and fall of wickets. Integration with External Data Sources: Integrate the application with external cricket data sources or APIs to fetch real-time match scores, player information, and match schedules. This will provide up-to-date and accurate information for users. Match Simulation Options: Provide options to simulate different match scenarios, such as limited-overs matches (T20, ODI) or Test matches. Allow users to customize match parameters like target scores, run rates, and required run rates.

Notifications and Reminders: Implement a notification system to remind users of upcoming matches, important milestones, or changes in match schedules. This can be achieved through email notifications or push notifications in a mobile application

**11. REFERENCES**

1. https://docs.python.org/3/library/tkinter.html
2. https://docs.python.org/3/
3. OANDA Currency Converter: <https://www.oanda.com/currency/converter/>
4. Fixer API: <https://fixer.io/>
5. Official Python Documentation: <https://docs.python.org/3/library/tkinter.html>
6. Tkinter Course on Real Python: <https://realpython.com/tkinter-python-gui-tutorial/>