

**Assignment Cover Sheet**

# advance programming

To be completed **electronically** by the student and submitted with each piece of work. Please upload this completed cover sheet via Turnitin.

## Assignment Title:

### Tutor: **Muhammad Ali**

Student Number: 0325025

Date of Submission:

Details of your submission:

For an online submission, please enter the URL of where your project files can be accessed (e.g. Google Drive). If a physical artefact, please provide further details on how this project has been submitted to tutors.

|  |
| --- |
| Github link: |

In submitting this assignment, I am confirming that I have read and understood the regulations for assessment, and I am aware of the seriousness with which the institute regards unfair practice.

Please see the academic [Unfair Practice Policy](https://thehub.bathspa.ac.uk/reference/student-policies/unfair-practice-policy) for details.

Signed: Date:

**Abstract:**

This is a Tkinter-based desktop application that allows users to search for any country by name and retrieve real-time data from the REST Countries API. The application displays key information including the country’s capital, region, population, total area, and national flag in a clear and user-friendly interface. Users interact with the application through text input and buttons, enabling them to search for a country or clear the displayed results. The program incorporates structured functions and object-oriented programming techniques to ensure clean code organisation, error handling, and smooth interaction between the graphical interface and the external API.

**Project plan:**

Choosing rest countries API and defining features took me 60 minutes. GUI wireframe, flowchart, pseudo-code took me 60 minutes. Writing python Tkinter code took me 240 minutes. Test cases took me 30-40 minutes. Documentation took me 180 minutes.

**Design:**

|  |
| --- |
|  |

User enters country name

|  |
| --- |
|  |

Wait for next user action

Wait for next input

Country not found

Is country found

Send API request to Rest Countries

Please enter name

Is input empty

**Technical description:**

****

**Critical reflection:**

Experience gained from the development of the Rest Countries Explorer application is relevant to the learned topics of GUI application development, API incorporation, and object-oriented programming using Python. One of the objectives of carrying out this project was to develop a user-friendly Tkinter application that is capable of fetching data from the Rest Countries API.

One of the strengths of the application is that concerns are properly separated using classes. Concerns are separated since one class, named CountryAPI, deals with API matters, while another class, named CountryApp, is responsible for the graphical user interface. Using classes made the application more readable, maintainable, and extendible. It is also important to note that the application used functions to exchange information between components, which is important in ensuring that good programming principles are followed. Another strength of the application is that it utilized Tkinter, which helps one develop a GUI quickly. Additionally, the application used the PIL library to display flags of countries.

A personal challenge involved dealing with invalid input from the user as well as dealing with uncertain API results. This was solved by adding error handling using message boxes to ensure the application handled these issues gracefully. Image references using Tkinter also enhanced the understanding of memory management in GUI applications.

Despite these qualities, there are some shortcomings in the application. It is fairly static with synchronous API calls that could make it temporarily unresponsive over a slow network connection. It would be beneficial for it to feature asynchronous requests in the future, more information about the listed countries such as their currencies and languages, or more customizing possibilities for the user. This project has enhanced my knowledge of event-driven programming.

**Appendix:**

import tkinter as tk

from tkinter import messagebox

from PIL import Image, ImageTk

import requests

from io import BytesIO

class CountryAPI:

    """Handles all communication with the REST Countries API"""

    def get\_country(self, name):

        # Build API URL using the country name

        url = f"https://restcountries.com/v3.1/name/{name}"

        # Send GET request to the API

        response = requests.get(url)

        # If request fails, return None

        if response.status\_code != 200:

            return None

        # Return the first matching country result

        return response.json()[0]

class CountryApp:

    """Main GUI application for displaying country information"""

    def \_\_init\_\_(self, root):

        # Configure main window

        self.root = root

        self.root.title("Rest Countries Explorer")

        self.root.geometry("550x500")

        self.root.configure(bg="#212121")

        # Initialize API handler and flag image reference

        self.api = CountryAPI()

        self.flag\_image = None

        # Create GUI components

        self.create\_widgets()

    def create\_widgets(self):

        # Application title

        tk.Label(

            self.root,

            text="🌍 Rest Countries Explorer",

            font=("Century", 24, "bold"),

            bg="#212121",

            fg="white"

        ).pack(pady=15)

        # Frame to hold input widgets

        input\_frame = tk.Frame(self.root, bg="#212121")

        input\_frame.pack(pady=10)

        # Label for country input

        tk.Label(

            input\_frame,

            text="Country Name:",

            font=("Century", 14, "bold"),

            bg="#212121",

            fg="white"

        ).grid(row=0, column=0, padx=5)

        # Entry field for user input

        self.country\_entry = tk.Entry(

            input\_frame,

            font=("Century", 14),

            width=25,

            bg="#161616",

            fg="white",

            insertbackground="white"

        )

        self.country\_entry.grid(row=0, column=1, padx=5)

        # Button to search for country data

        tk.Button(

            input\_frame,

            text="Search",

            font=("Century", 12, "bold"),

            command=self.search\_country

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

        # Button to clear input and results

        tk.Button(

            input\_frame,

            text="Clear",

            font=("Century", 12, "bold"),

            command=self.clear\_all

        ).grid(row=0, column=3, padx=5)

        # Frame to display country information

        self.result\_frame = tk.Frame(self.root, bg="#161616")

        self.result\_frame.pack(padx=20, pady=20, fill="both", expand=True)

        # Label to show country details

        self.result\_label = tk.Label(

            self.result\_frame,

            font=("Century", 13),

            bg="#161616",

            fg="white",

            justify="left",

            anchor="nw"

        )

        self.result\_label.pack(padx=10, pady=10, fill="both", expand=True)

        # Label to display the country flag

        self.flag\_label = tk.Label(self.result\_frame, bg="#161616")

        self.flag\_label.pack(pady=5)

    def search\_country(self):

        # Get user input and remove extra spaces

        country\_name = self.country\_entry.get().strip()

        # Validate input

        if not country\_name:

            messagebox.showwarning("Input Error", "Please enter a country name.")

            return

        # Fetch country data from API

        data = self.api.get\_country(country\_name)

        # Handle invalid country names

        if data is None:

            messagebox.showerror("Error", "Country not found.")

            self.clear\_all()

            return

        # Display fetched country data

        self.display\_country(data)

    def display\_country(self, data):

        # Extract relevant information from API response

        name = data.get("name", {}).get("common", "N/A")

        capital = data.get("capital", ["N/A"])[0]

        region = data.get("region", "N/A")

        population = data.get("population", 0)

        area = data.get("area", "N/A")

        flag\_url = data.get("flags", {}).get("png", "")

        # Display formatted country information

        self.result\_label.config(

            text=(

                f"Country: {name}\n\n"

                f"Capital: {capital}\n"

                f"Region: {region}\n"

                f"Population: {population:,}\n"

                f"Area: {area} km²"

            )

        )

        # Download and display the country flag

        if flag\_url:

            image\_data = requests.get(flag\_url).content

            image = Image.open(BytesIO(image\_data)).resize((120, 80))

            self.flag\_image = ImageTk.PhotoImage(image)

            self.flag\_label.config(image=self.flag\_image)

    def clear\_all(self):

        # Clear input field and displayed results

        self.country\_entry.delete(0, tk.END)

        self.result\_label.config(text="")

        self.flag\_label.config(image="")

# Program entry point

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = CountryApp(root)

    root.mainloop()

**output**:

