CODE LAB II

**Assessment 2:**

|  |  |
| --- | --- |
| **Students Name** | Jace Christen M. Bocobo |
| **Student Id** | 2023-311 |
| **Tutor** | Ms. Lavanya Mohan |
| **Repository Link** | https://github.com/jacemapagmahal/Advance-Programming-Assessment-2 |
| **YouTube Link** | https://youtu.be/5N7SYa\_3Ero |

**Jace’s Pokedex**

**Abstract:**

Jace’s Pokedex is a simple interactive desktop application that was built using Python's tkinter library. It allows users to get a detailed information about any Pokémon directly from the **Pokémon API (PokeAPI) website**. Designed with a clean and Pokémon-themed user interface, this application is perfect for Pokémon fans and anyone curious about the iconic creatures from the Pokémon universe.

**Project Plan:**

| **Milestone** | **Tasks** | **Estimated Time** |
| --- | --- | --- |
| **Planning** | Selecting of the API that suits my interests, define the objectives of the app, designing of the flowchart and how to integrate it with the code. | 1 hours |
| **Design** | Creating of the GUI | 2 hours |
| **Development** | Implementation of the API into the GUI. | 2 hours |
| **Testing** | Testing the errors and the funcitionality of the app. | 1 hour |
| **Documentation** | Writing of the development document and record walkthrough video | 3 hours |

**Evidence of Design:**

**Specification List:**

* Use PokiAPI to fetch pokemon data.
* Provide a user interface that gives them the choice to get information about their favorite pokemon.
* Display the details of the chosen pokemon.
* Make a visually aesthetic and functional GUI using Tkinter.

**Wireframe:**

1. **Welcome Screen:** 
   * Title: "Welcome to Jace’s Pokedex”
   * Buttons: "Start" to proceed.
2. **Pokemon Name Screen:** 
   * Buttons: "Clear" "Get Data", “Close”
3. **Pokemon Details Screen:** 
   * Display Pokemon’s Name.
   * Display Pokemon’s Abilities.
   * Display Pokemon’s Type.
   * Display Pokemon’s Description.
   * Buttons: “Clear” “Get Data”, “Close"

**UML Data Structure Diagram:**

* Fetch Pokemon: Calls API and returns a list of pokemon.
* Fetch Pokemon details: Calls API and retrieves detailed information about a specific Pokemon.
* GUI classes: Handle the interface and user interactions.

**Technical Description:**

1. **API Integration**:
   * pokemon = pokemon\_entry.get()
   * response = requests.get(url)
   * response.raise\_for\_status()
   * name = data['name'].capitalize()

* abilities = ", ".join([ability['ability']['name'] for ability in data['abilities']])
* types = ", ".join([t['type']['name'].capitalize() for t in data['types']])
* species\_url = data['species']['url']

1. **GUI Setup**:
   * Tkinter is used to design a clean and responsive user interface.
   * Screens include the welcome screen, and pokemon screen
2. **Dynamic GUI Elements**:
   * Buttons dynamically created for the Pokemons.

**Testing:**

**Test Plan:**

| **Test Case** | **Input** | **Expected Output** | **Status** |
| --- | --- | --- | --- |
| Open welcome screen | Launch application | Welcome screen displays with title and button | Pass |
| Proceed to Pokemon selection | Click "start" | Pokemon selection screen | Pass |
| Request Pokemon list | Input "Name" | Name shows up | Pass |
| Request Pokemon list | Click “get data” | Pokemon’s Name, Abilities, Type, Description. | Pass |
| Clear button | Click “clear” | Pokemon’s name gets erased | Pass |
| Close Button | Click “close” | Application closes | Pass |

**Critical Reflection:**

The **Jace's Pokedex** application is a simple yet engaging Pokémon data retrieval tool built using Python's **Tkinter** for the graphical user interface (GUI) and the **PokéAPI** for fetching Pokémon data.

The goal of this application is to allow users to input a Pokémon name and fetch relevant information about that Pokémon, such as its name, abilities, type, and description. The application can be considered a basic, digital version of a Pokedex, providing an easy way to get information on Pokémon from the database provided by the PokéAPI. Although i found a somewhat succes with the application, I did face a few challenges along the way. Challenges like the time, my skills and abilities, and a few learning curves in making this app. Although i had a lot of challenges, i found that in the process of making this app, my skills as a programmer has tremendously improved. So i would like to thank Miss Lavanya for concucting a fun yet challenging exercise like this.

**Appendix:**

**Code:**

import tkinter as tk

from tkinter import messagebox

import requests

def show\_pokedex():

welcome\_frame.pack\_forget()

pokedex\_frame.pack(expand=True)

def get\_data():

pokemon = pokemon\_entry.get()

url = f"https://pokeapi.co/api/v2/pokemon/{pokemon.lower()}/"

try:

response = requests.get(url)

response.raise\_for\_status()

data = response.json()

name = data['name'].capitalize()

abilities = ", ".join([ability['ability']['name'] for ability in data['abilities']])

types = ", ".join([t['type']['name'].capitalize() for t in data['types']])

species\_url = data['species']['url']

species\_response = requests.get(species\_url)

species\_response.raise\_for\_status()

species\_data = species\_response.json()

description = species\_data['flavor\_text\_entries'][0]['flavor\_text'].replace("\n", " ")

pokemon\_name\_label.config(text=f"Name: {name}")

pokemon\_abilities\_label.config(text=f"Abilities: {abilities}")

pokemon\_type\_label.config(text=f"Type: {types}")

pokemon\_description\_label.config(text=f"Description: {description}")

except requests.exceptions.HTTPError as errh:

messagebox.showerror("HTTP Error", f"HTTP Error: {errh}")

except requests.exceptions.ConnectionError as errc:

messagebox.showerror("Connection Error", f"Connection Error: {errc}")

except requests.exceptions.Timeout as errt:

messagebox.showerror("Timeout Error", f"Timeout Error: {errt}")

except requests.exceptions.RequestException as err:

messagebox.showerror("Request Error", f"Something went wrong: {err}")

except KeyError:

messagebox.showerror("Error", f"Could not find data for Pokémon '{pokemon}'.")

def close\_app():

root.destroy()

root = tk.Tk()

root.title("Jace's Pokedex")

root.geometry("360x640")

root.resizable(False, False)

root.config(bg="#f1f1f1")

welcome\_frame = tk.Frame(root, bg="#f1f1f1")

welcome\_frame.pack(expand=True)

welcome\_label = tk.Label(welcome\_frame, text="Welcome to Jace's Pokedex", font=("Arial", 18), bg="#f1f1f1")

welcome\_label.pack(pady=100)

start\_button = tk.Button(welcome\_frame, text="Start", font=("Arial", 14), bg="#4CAF50", fg="white", bd=0, relief="solid", command=show\_pokedex)

start\_button.pack(pady=20, ipadx=40, ipady=10)

pokedex\_frame = tk.Frame(root, bg="#f1f1f1")

pokemon\_label = tk.Label(pokedex\_frame, text="Pokémon Name", font=("Arial", 16), bg="#f1f1f1")

pokemon\_label.pack(pady=20)

pokemon\_entry = tk.Entry(pokedex\_frame, font=("Arial", 14), bd=0, relief="solid", width=20, justify="center")

pokemon\_entry.pack(pady=10)

clear\_button = tk.Button(pokedex\_frame, text="Clear", font=("Arial", 14), bg="#ff6347", fg="white", bd=0, relief="solid", command=lambda: pokemon\_entry.delete(0, tk.END))

clear\_button.pack(pady=10, ipadx=20, ipady=5)

get\_data\_button = tk.Button(pokedex\_frame, text="Get Data", font=("Arial", 14), bg="#4CAF50", fg="white", bd=0, relief="solid", command=get\_data)

get\_data\_button.pack(pady=10, ipadx=20, ipady=5)

pokemon\_name\_label = tk.Label(pokedex\_frame, text="Name: N/A", font=("Arial", 14), bg="#f1f1f1")

pokemon\_name\_label.pack(pady=10)

pokemon\_abilities\_label = tk.Label(pokedex\_frame, text="Abilities: N/A", font=("Arial", 14), bg="#f1f1f1")

pokemon\_abilities\_label.pack(pady=10)

pokemon\_type\_label = tk.Label(pokedex\_frame, text="Type: N/A", font=("Arial", 14), bg="#f1f1f1")

pokemon\_type\_label.pack(pady=10)

pokemon\_description\_label = tk.Label(pokedex\_frame, text="Description: N/A", font=("Arial", 14), bg="#f1f1f1", wraplength=300)

pokemon\_description\_label.pack(pady=10)

close\_button = tk.Button(pokedex\_frame, text="Close", font=("Arial", 14), bg="#d32f2f", fg="white", bd=0, relief="solid", command=close\_app)

close\_button.pack(pady=20, ipadx=20, ipady=5)

root.mainloop()