Assessment 2 - Advance Programming

Student's Name: Aliyya Yla Cortez Bohol
Id No: 2023-636
Date of Submission: January 6, 2025
GitHub Repository Name: aliyyayla

GitHub Repository Link:
Advance-Programming-Assessment-2-Aliyyaylabohol

API: the cocktailDB

Abstract:

The "Cocktail for one, please !" is a little project I worked on which allows the user to search the cocktail name they like, lookup the ingredient of the drink using an ID, can choose any random cocktails and filter a glass they want using tkinter.
Thanks to the GUI (graphical user interface) entry label, buttons, and images, the project turned out just the way I wanted it to look, simple and easy to use.

Project Plan:

Before creating the app, I already started working on the harry potterDB but after 30-40% of coding and testing, I found out that some of the urls aren't working so I hastily changed my API into the CocktailDB. Planning took me ½ hour, looking for a better and easier API to use.
Designing took 4 hours, combining color schemes, and planning the GUI layout.
Development took a week and 2 days, after christmas and before new years. Adding features, fetching data and building GUI in tkinter during the development of my project.
Testing 3-4 hours, making adjustments, checking any errors in my code.

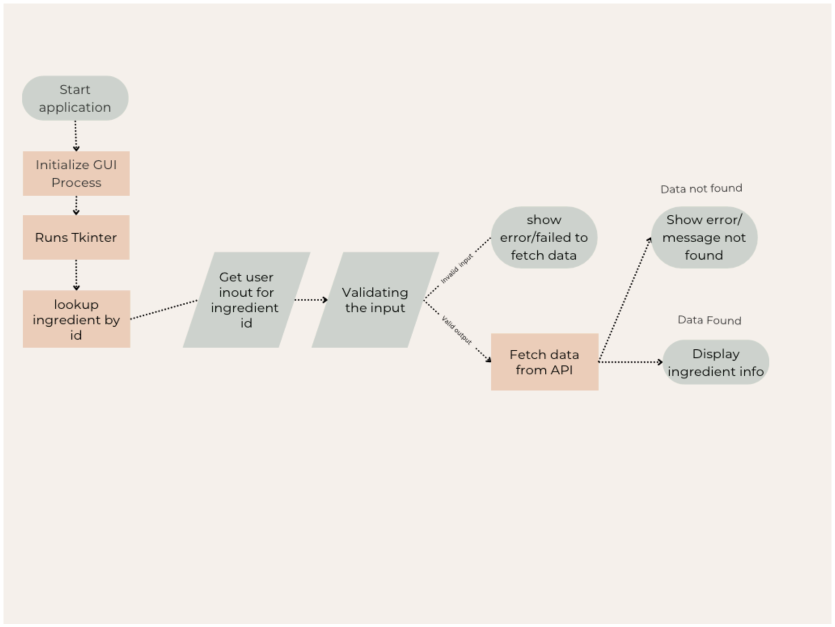
VISUAL TIMELINE:

Planning- ½ hour
Designing- 4 hours
(2 hours for planning the GUI layout, and 2 hours for the color scheme.)
Development- 1 week and 2 days
(adding features, doing small testing of the buttons and entry labels and building the layout of the GUI.)
Testing- 4 hours (also part of my development. 3 hours making adjustments, 1 hour for checking errors.)

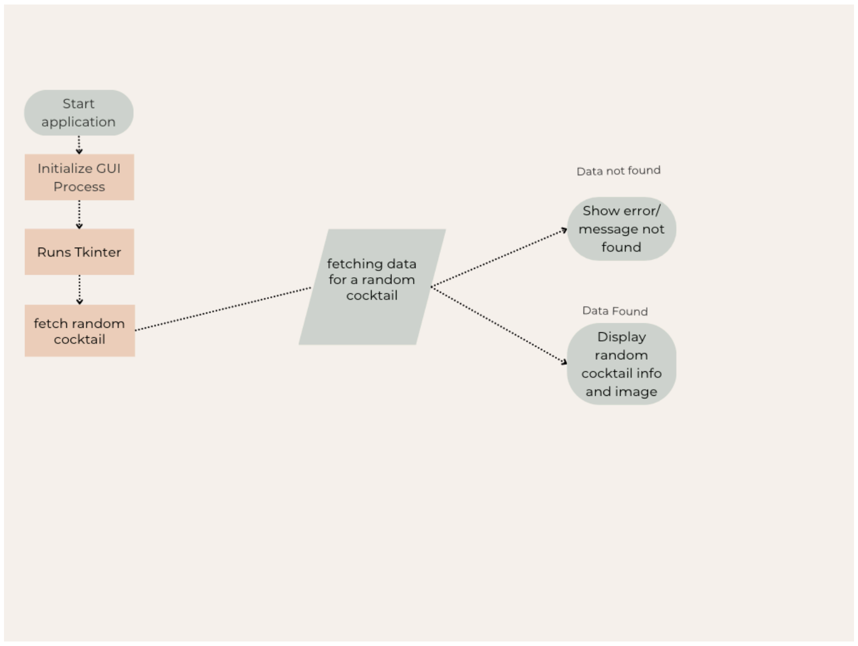
Evidence of Design:

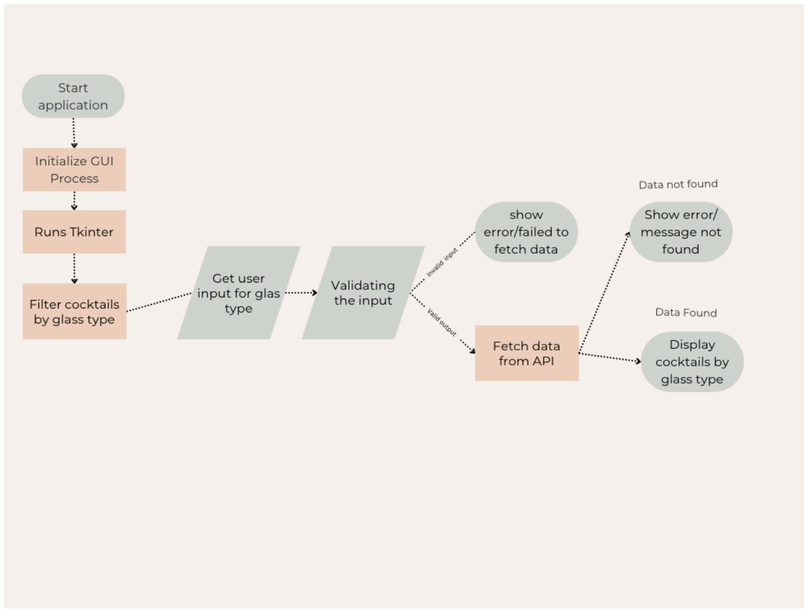
After building my little project, we are tasked to create our own flowchart, pseudo-code, wireframes and UML data structure. Here is the evidence of my design that I have created via canva.

FLOWCHART:

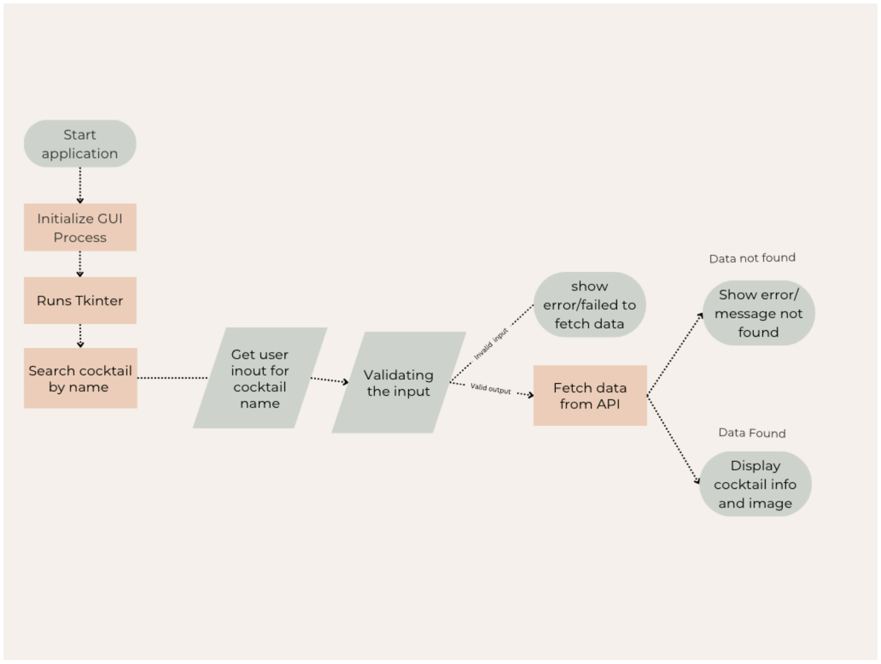


Flowchart for ingredient by ID


Start application
Initialize GUI Process
Runs Tkinter
fetch random cocktail
fetching data for a random cocktail
Data not found
Show error/ message not found
Data Found
Display random cocktail info and image

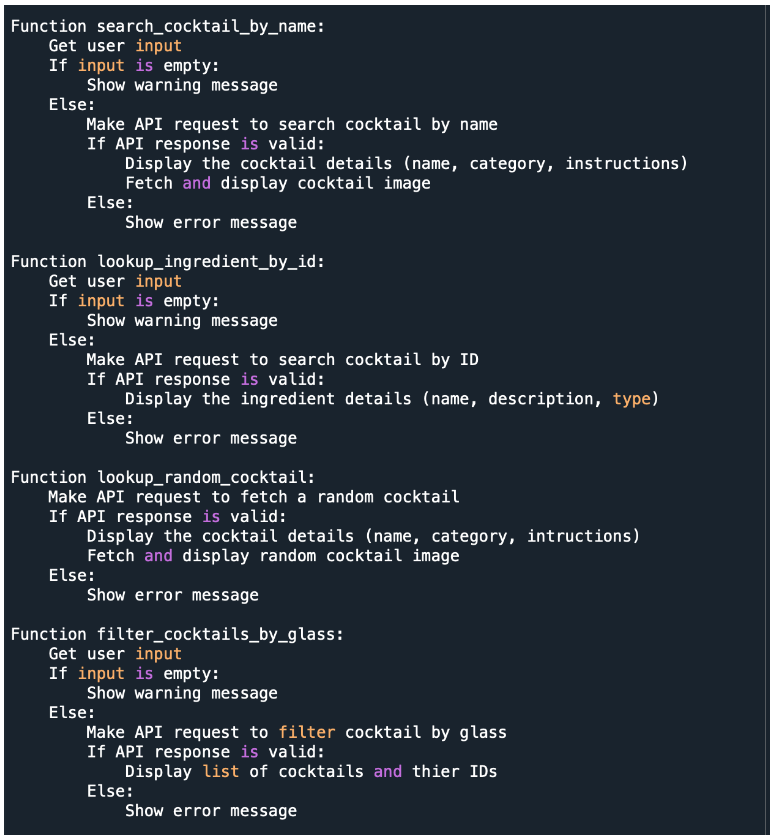

Start application
Initialize GUI Process
Runs Tkinter
Filter cocktails by glass type
Get user input for glas type
Validating the input
Invalid input
show error/failed to fetch data
Valid output
Fetch data from API
Data not found
Show error/ message not found
Data Found
Display cocktails by glass type

Flowchart for random cocktail and glass type



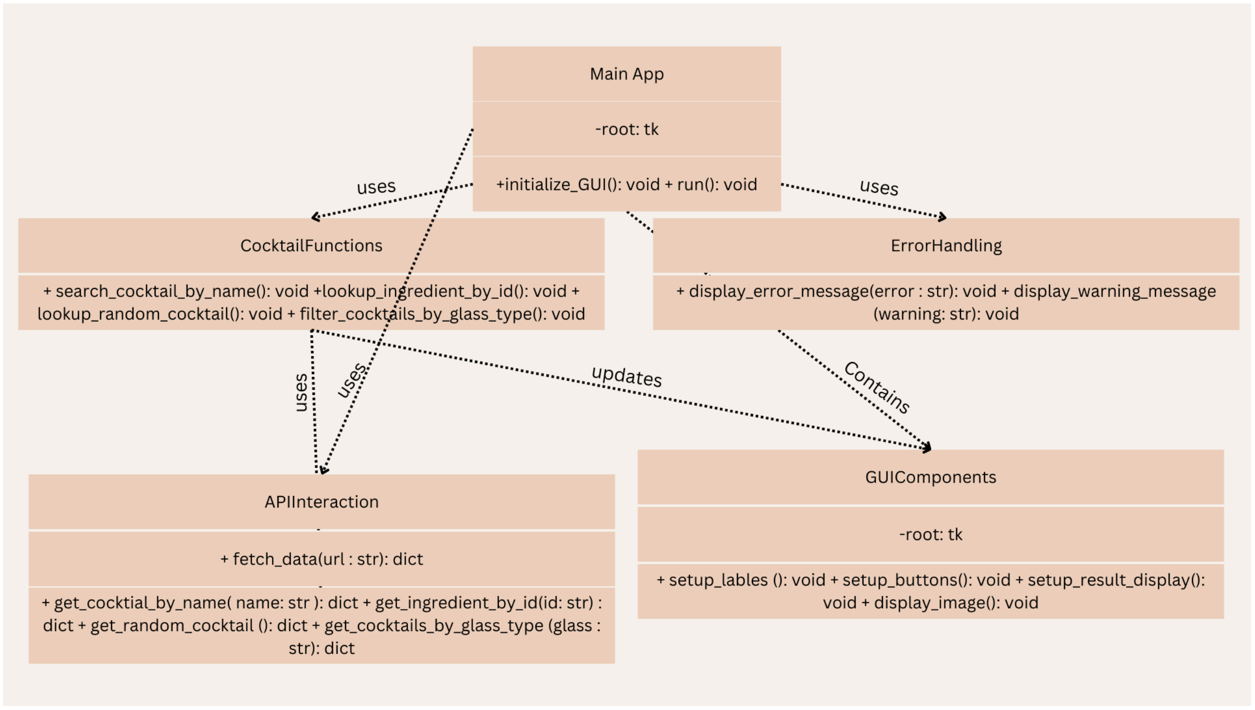
Flowchart for cocktail by name

PSEUDO-CODE:

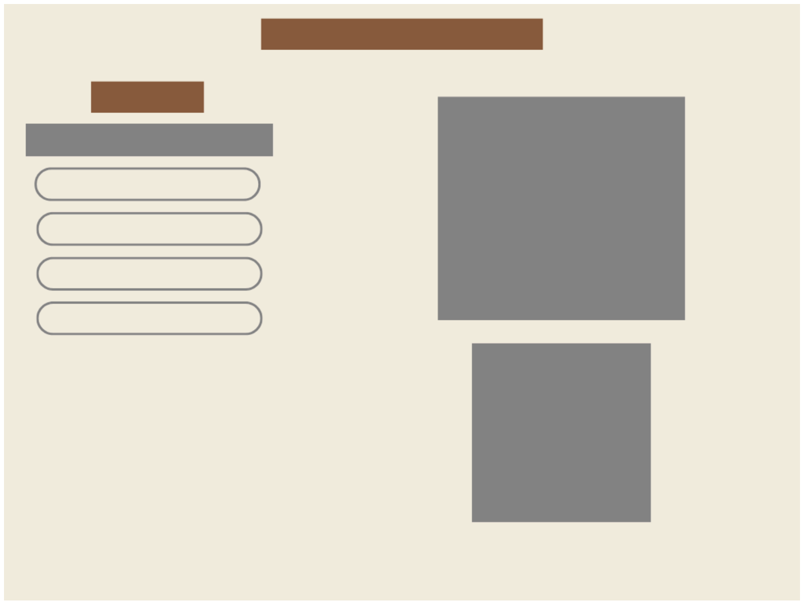


```
Function search_cocktail_by_name:
    Get user input
    If input is empty:
        Show warning message
    Else:
        Make API request to search cocktail by name
        If API response is valid:
            Display the cocktail details (name, category, instructions)
            Fetch and display cocktail image
        Else:
            Show error message

Function lookup_ingredient_by_id:
    Get user input
    If input is empty:
        Show warning message
    Else:
        Make API request to search cocktail by ID
        If API response is valid:
            Display the ingredient details (name, description, type)
        Else:
            Show error message

Function lookup_random_cocktail:
    Make API request to fetch a random cocktail
    If API response is valid:
        Display the cocktail details (name, category, intructions)
        Fetch and display random cocktail image
    Else:
        Show error message

Function filter_cocktails_by_glass:
    Get user input
    If input is empty:
        Show warning message
    Else:
        Make API request to filter cocktail by glass
        If API response is valid:
            Display list of cocktails and thier IDs
        Else:
            Show error message
```

UML Data Structure:


Main App
-root: tk
+initialize_GUI(): void + run(): void
uses
uses
CocktailFunctions
+ search_cocktail_by_name(): void +lookup_ingredient_by_id(): void + lookup_random_cocktail(): void + filter_cocktails_by_glass_type(): void
ErrorHandling
+ display_error_message(error : str): void + display_warning_message (warning: str): void
uses
uses
updates
Contains
APIInteraction
+ fetch_data(url : str): dict
+ get_cocktial_by_name(name: str): dict + get_ingredient_by_id(id: str) : dict + get_random_cocktail (): dict + get_cocktails_by_glass_type (glass : str): dict
GUIComponents
-root: tk
+ setup_lables (): void + setup_buttons(): void + setup_result_display(): void + display_image(): void

Wireframe:



Technical Description & Walkthrough:

Testing:

The testing worked 100%. Each button worked perfectly to show a displayed image and instructions at the result label. For more information about the testing, the link to the video testing and the testing table can be found below.

TESTING TABLE;

Description	Step by Step	Expected Result	Actual Result	Status
Search cocktail by name	1. Enter "Mojito" or any drink of your preference. 2. Click the "Search Cocktail by Name" button.	The Mojito's details along with the instruction and image will be displayed.	Will display the image of Mojito with details of the drink.	Passed (O)
Lookup ingredient by ID	1. Enter the ID "522" in the search bar. 2. Click the "Lookup ingredient by ID" button.	The details of the ID 522 will display.	White Port drink will be displayed along with the description of the drink.	Passed (O)
Lookup random cocktail	1. Click the "Lookup Random Cocktail" button.	A random cocktail drink will be displayed along with an image and the details.	Shows a random drink with the details of the drink and image.	Passed (O)

Filter cocktail by glass	1. Search “wine glass”. 2. Click the “Filter Cocktail by Glass” button.	A list of drinks that use wine glass will be displayed.	Shows a list of cocktails that use wine glass.	Passed (O)

Link of the video test: https://youtu.be/pVMpaKlJn7k

Technical Description & Walkthrough:

“Cocktail For One, Please !” is thoroughly explained through a 27 minute video. technical description and the walkthrough of the code are included.
Link for the Video description and walkthrough:
https://youtu.be/aiANKX3RUek?si=TbgJBPz_ypLmRpoo

Critical Reflection:

Throughout making this project, the first part is the GUI. It could use more design to make it more lively, the compelling part about this assessment is understanding how your code is going to create a simple with exact results I like based on my preferences. You need to understand each block, line of your code to get the results you want from your GUI layout until the result display. I did backtrack on the previous lessons and also learn from chatgpt and other website resources like W3schools for help. The code could improve but the GUI layout could do so much better with background designs and fun pop of colors.

My takeaways from this assessment is first and foremost, don’t cram. Secondly, to understand your own code and how it works, my performance could improve more if I only participated a lot at school.

Appendix:

Here is the code for my assessment 2.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 30 13:16:51 2024

@author: aliyya
"""

import tkinter as tk # is used to create graphical user interfaces that provides all the layout
widgets.
from tkinter import messagebox # will show important messages (see line 18, 43, 89, 95, 108)
from PIL import Image, ImageTk # it will allow you to fetch images
import requests # will help fetch data from the cocktail db app

# Function to search cocktail by name
def search_cocktail_by_name(): # fetches the cocktail's data from the API entered in the
entry_search field.
    cocktail_name = entry_search.get() # retrives the text the user input into the entry_search
field.
    if not cocktail_name: # this meets the condition and a message box of show warning will
appear to re-enter the given data when the entry_field is incorrect or empty.
        messagebox.showwarning("Input Error", "Enter a cocktail name.")
        return

    try: # attempting to let you try a code to see if there might cause an error.
        response =
requests.get(f"https://www.thecocktaildb.com/api/json/v1/1/search.php?s={cocktail_name}") #
sends an HTTP GET requests to fetch a specific data from the cocktail db.
        data = response.json() #  will take the API response (in JSON format).

        if data["drinks"]: # API response checks if there is a drink data inside.
            drink = data["drinks"][0] # starting from the first cocktail drink.
            result_text.set(f"Name: {drink['strDrink']}\nCategory:
{drink['strCategory']}\nInstructions: {drink['strInstructions']}") # then converts it to a python
dictionary.

            # Display image
            img_url = drink["strDrinkThumb"] # will get the image url of the cocktail from the API
response.
            img_data = requests.get(img_url).content # will send a request to download the image
from the given url.
            with open("cocktail_image.jpg", "wb") as handler: # this will the downloaded image as a
file name cocktail_image.jpg.
                handler.write(img_data) # will write a binary data of the img_url to
cocktail_image.jpg.
            img = Image.open("cocktail_image.jpg") # the file name of the cocktail drink image.
            img = img.resize((300, 300)) # for resizing the image.
            img = ImageTk.PhotoImage(img) # to prepare the image that has been converted to be
compatible to tkinter widgets.
            image_label.config(image=img) # for the image to show.
            image_label.image = img # keeps the reference of the image.
        else:
            result_text.set("Cocktail not found.") # this code will update the text result in "cocktail
not found" if an image isn't return.
            image_label.config(image="")
    except Exception as e: # the execption block will avoid your program from crashing when
something goes wrong.
        messagebox.showerror("An error has occured.", f"Failed to fetch data: {e}")

# Function to lookup ingredient by ID
def lookup_ingredient_by_id(): # fetches the cocktail's ingredient from the API when user
provided the ingredient ID.
    ingredient_id = entry_search.get() # retrives the text the user input into the entry_search
field. Will read the integer as string.
    if not ingredient_id: # to make sure the ingredient ID is input into the entry_search, if not
then the condition is true.
        messagebox.showwarning("Input Error", "Enter an ingredient ID.")
        return

    try: # attempting to let you try a code to see if there might cause an error.
```

```
        response =
requests.get(f"https://www.thecocktaildb.com/api/json/v1/1/lookup.php?iid={ingredient_id}")
# sends an HTTP GET requests to fetch a specific data from the cocktail db.
        data = response.json() # will take the API response (in JSON format).

        if data["ingredients"]: # API response checks if the ingredient has data inside.
            ingredient = data["ingredients"][0] # starting from the first ingredient.
            result_text.set(f"Name: {ingredient['strIngredient']}\nDescription:
{ingredient.get('strDescription', 'No description')}\nType: {ingredient.get('strType', 'N/A')}") #
then converts it to a python dictionary.
        else: # when the data is not found, this line of code will be executed.
            result_text.set("Ingredient ID not found.")
    except Exception as e: # when an error occures when something is wrong.
        messagebox.showerror("An error has occured.", f"Failed to fetch data: {e}") # will show this
error message.

# Function to lookup random cocktail
def lookup_random_cocktail(): # fetches a random cocktail from the API when user presses
the button.
    try: # attempting to let you try a code to see if there might cause an error.
        response = requests.get("https://www.thecocktaildb.com/api/json/v1/1/random.php") #
sends an HTTP GET requests to fetch a specific data from the cocktail db.
        data = response.json() #  will take the API response (in JSON format).

        if data["drinks"]: # API response checks if there is a drink data inside.
            drink = data["drinks"][0] # starting from a random cocktail drink.
            result_text.set(f"Name: {drink['strDrink']}\nCategory:
{drink['strCategory']}\nInstructions: {drink['strInstructions']}") # then converts it to a python
dictionary.

            # Display image
            img_url = drink["strDrinkThumb"] # will get the image url of the cocktail from the API
response.
            img_data = requests.get(img_url).content # will send a request to download the image
from the given url.
            with open("random_cocktail.jpg", "wb") as handler: # to open a file for the random
cocktail image.
```

```
                handler.write(img_data) # will write a binary data of the img_url to
random_cocktail.jpg.
                img = Image.open("random_cocktail.jpg") # the file name for the random cocktail
drink image.
                img = img.resize((200, 200)) # for resizing the image.
                img = ImageTk.PhotoImage(img) # to prepare the image that has been converted to be
compatible to tkinter widgets.
                image_label.config(image=img) # for the image to show.
                image_label.image = img # keeps the reference of the image.
        else:
            result_text.set("No random cocktail found.") # this code will update the text result in "no
random cocktail not found" if an image isn't return.
            image_label.config(image="") # to remove the displayed image.

    except Exception as e: # the exception block will let you avoid your program from crashing
when something goes wrong.
        messagebox.showerror("An error has occured.", f"Failed to fetch data: {e}")

# Function to filter cocktails by glass
def filter_cocktails_by_glass(): # fetches a cocktail depending what kind of glass is input from
the API when user presses the button.
    glass_type = entry_search.get() # retrives the text the user input into the entry_search field.
    if not glass_type: # to make sure the glass is input into the entry_search, if not then the
condition is true.
        messagebox.showwarning("Input Error", "Please enter a glass type.")
        return

    try: # attempting to let you try a code to see if there might cause an error.
        response =
requests.get(f"https://www.thecocktaildb.com/api/json/v1/1/filter.php?g=Cocktail_glass") #
sends an HTTP GET requests to fetch a secific data from the cocktail db.
        data = response.json() # will take the API response (in JSON format).

        if data["drinks"]: # API response checks if the drinks has data inside.
            drinks_list = "\n".join([drink["strDrink"] for drink in data["drinks"]]) # will loop though
every drink there is in the data["drinks"] to get the value of the drink's name or the "StrDrink".
            result_text.set(f"Cocktails served in {glass_type} glass:\n{drinks_list}") # it will show the
updated displayed text in the result section of GUI.
```

```
        else: # if drinks doesn't exist or is empty, else block is then true.
            result_text.set("No cocktails found for this glass type.")
    except Exception as e: # when an error occures when something is wrong.
        messagebox.showerror("An error has occured.", f"Failed to fetch data: {e}")

# GUI setup
root = tk.Tk() # it initializes a new tkinter app.
root.title("The Cocktail DB App") # setting the title of tkinter.
root.geometry("790x500") # default tkinter size.
root.configure(bg="#F3ECE0") # for a default background color.

# The upper/top label
label_search = tk.Label(root, text="Cocktail for one, please !", font=("Baskerville", 40),
bg="#F3ECE0", fg="#8B5E3C")
label_search.pack(pady=6)

# The main frame container
main_frame = tk.Frame(root, bg="#F3ECE0")
main_frame.pack(fill="both", expand=True) #

# The left frame for buttons
left_frame = tk.Frame(main_frame, bg="#F3ECE0")
left_frame.pack(side="left", padx=10, pady=10, fill="y")

# the search bar and entry label
label_search = tk.Label(left_frame, text="Search:", font=("Baskerville", 20), bg="#F3ECE0",
fg="#8B5E3C")
label_search.pack(pady=0)

# The entry search bar
entry_search = tk.Entry(left_frame, width=30)
entry_search.pack(pady=5)

# The search button for cocktail by name
search_button_cocktail_name = tk.Button(left_frame, text="Search Cocktail by Name",
command=search_cocktail_by_name, font=("Baskerville", 18), bg="#EAC7A3", fg="#5E4A3B") #
used a command button for the
search_button_cocktail_name.pack(pady=5)
```

```
# The button for ingredient by ID
lookup_button_ingredient_id = tk.Button(left_frame, text="Lookup Ingredient by ID",
command=lookup_ingredient_by_id, font=("Baskerville", 18), bg="#EAC7A3", fg="#5E4A3B")
lookup_button_ingredient_id.pack(pady=5)

# The button for random cocktails
random_button_cocktail = tk.Button(left_frame, text="Lookup Random Cocktail",
command=lookup_random_cocktail, font=("Baskerville", 18), bg="#EAC7A3", fg="#5E4A3B")
random_button_cocktail.pack(pady=10)

# The button for filtering cocktails by glass
filter_by_glass_button = tk.Button(left_frame, text="Filter Cocktails by Glass",
command=filter_cocktails_by_glass, font=("Baskerville", 18), bg="#EAC7A3", fg="#5E4A3B")
filter_by_glass_button.pack(pady=5)

# The right frame for image and result
right_frame = tk.Frame(main_frame, bg="#F3ECE0")
right_frame.pack(side="right", padx=10, pady=10, fill="both", expand=True)

# The display image
image_label = tk.Label(right_frame, bg="#F8ECE0")
image_label.pack(pady=10)

# The display result
result_text = tk.StringVar()
result_label = tk.Label(right_frame, textvariable=result_text, justify="left", wraplength=400) #
added a justify left for the
result_label.pack(pady=10)

# for running the application
root.mainloop()
```

Repository Screenshot: