**ADVANCED**

**PROGRAMMING**

Assessment 2 - Data-Driven Application

Ms. Lavanya Mohan

Khaetlie Jhill Skkhitzz Pinera

Creative Computing Level 5

**Abstract**

The Cocktail Finder is a simple Tkinter application that uses data retrieved from a cocktail API, where sources of all drinks and cocktails from around the world can be found. It allows easy navigation of drinks and provides all the information the users look for.

Github Repository Link: <https://github.com/khaetlie/Assessment-2-Data-Driven-Application/tree/main>

**Project Plan**

The Project Planning was expected to be done within a month. The planning, design, development, and testing each have a week to work on.

***Week 1: Planning***

The first goal was to choose the database. My first thought was to make it easier for users with different goals to use this application. If the user wants to learn a new drink to make their own, this application will be easy because it displays all the ingredients and instructions in just a few clicks. If the user wants to find some non-alcoholic options there are different options for them to choose from. It all depends on how accessible and easy to navigate the application will be for those who will use it.

***Week 2: Design***

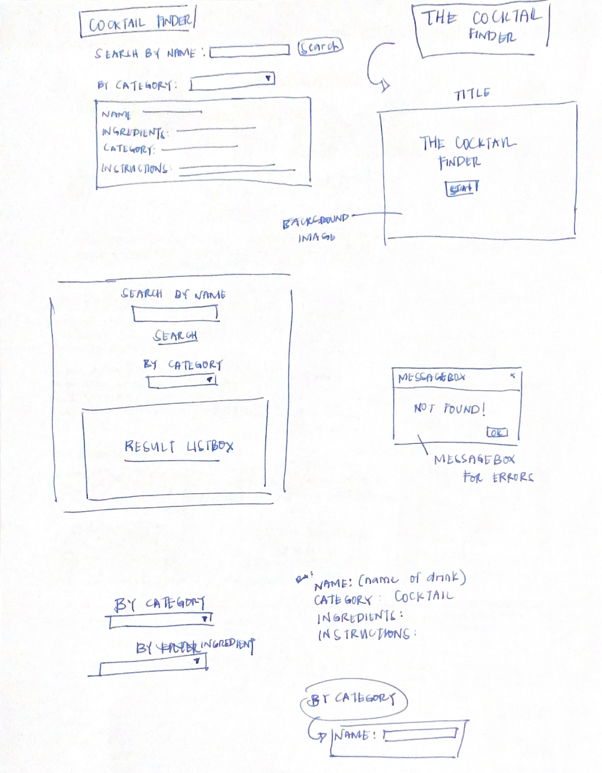
Initially, I thought of the UI elements to be simple but the background has to be appealing to have some balance, for it not to be too much. There has to be a title page with the name on it, and a button leading the user to the main page. Starting from the top, the search bar is the most important. This allows users to easily find what they want to see. Just below the search bar, there is an option to filter their search by category. And then the results will be shown just below the filter. To make this easier, I had to design a wireframe and a flowchart to easily figure out what I wanted the entire thing to look like. Nothing major has changed in the end, and it went exactly the way it was supposed to be.

***Week 3 & 4: Development & Testing***

The development took longer than what was initially planned. It required me two or more weeks to develop this as it was the main focus of the entire project. The planning and design didn’t take much time so there was enough time to focus on the development phase. First, I had to check if the API was accessible and was successful. Starting with the knowledge I already had, I started with the basic elements by coding them. The labels, buttons, and list box. Adding classes and functions consumed more time as it required testing. I made classes for better organization and made sure the functionality worked well. I made use of pygame to add the background music by the time the development phase is finished. For testing, I made a simple table to show what is expected and if the results will be the same as the initial planning. There was little to no error during the testing, providing good results in the end.

**Evidence of Design**

In the design phase, I made a simple wireframe to organize and plan what the GUI would look like. There must be a title page first. Then the main page should be simple yet engaging. the objective of the design is to present information clearly with user-friendly elements. I also added background music to make it more fun for the users.



**Technical Description & Walkthrough**

1. ***User Interface***

* The Cocktail API Tkinter application is built using Tkinter, with the following features: Search bar, Dropdown Menu, Results Listbox, and Details & Information.
* These features allow users to search and filter the details they want to view using the API data.

1. ***API Integration***

* Once the user looks for information, it asks a request to the API endpoints.

1. ***Error Handling***

* The application handles errors like empty search inputs, invalid names, or any problems with the API connection. It shows error messages for users to understand.

1. ***Workflow***
2. Search bar

* Search the names of the drinks/cocktails in the search bar and click the search button.
* This sends a request to the database endpoint.
* The results are shown in the list box showing all the detailed information.

1. Categories & Listbox

* The user can select the category using the dropdown menu to filter what they are looking for
* This shows all the similar drinks that are displayed.

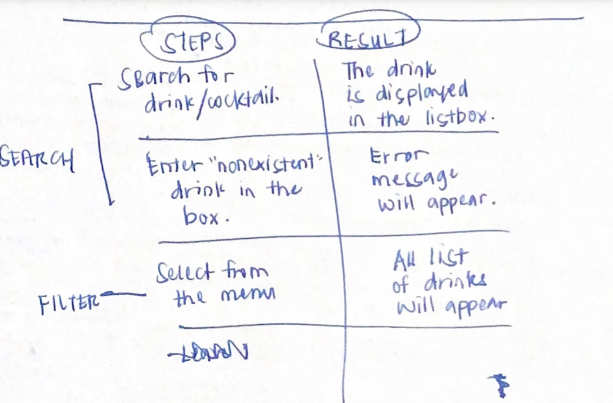
1. Error Handling

* If the API request fails, error messages will be shown in the message box.

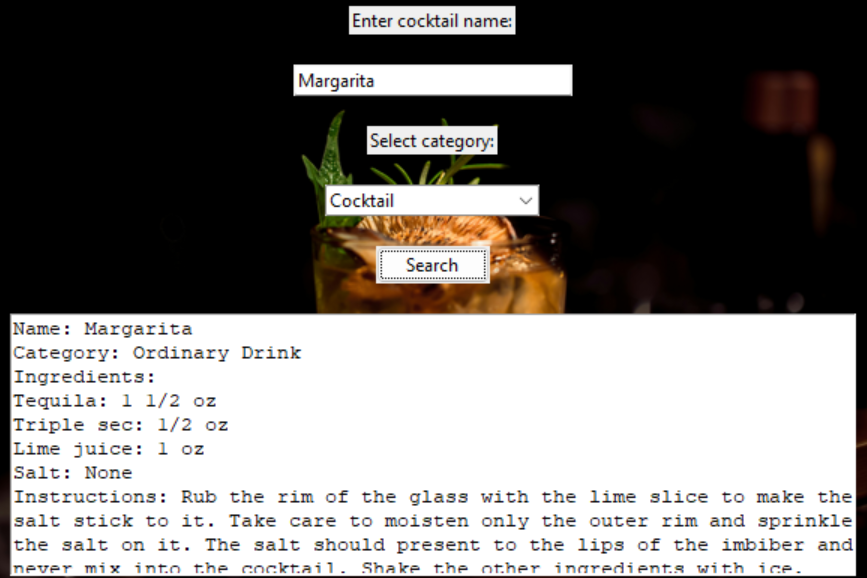
*Four key features shown above are*: The functionality of the search bar, options to filter by categories, details and information view, and the ability to handle errors that appear in the application.

Walkthrough Video Link: <https://drive.google.com/drive/folders/1AtvChmKJb6SsuFUvgkEIYK9Bs1eJxbco?usp=sharing>

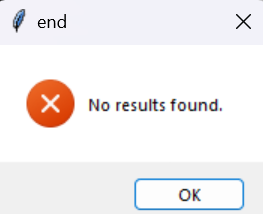
**Testing**



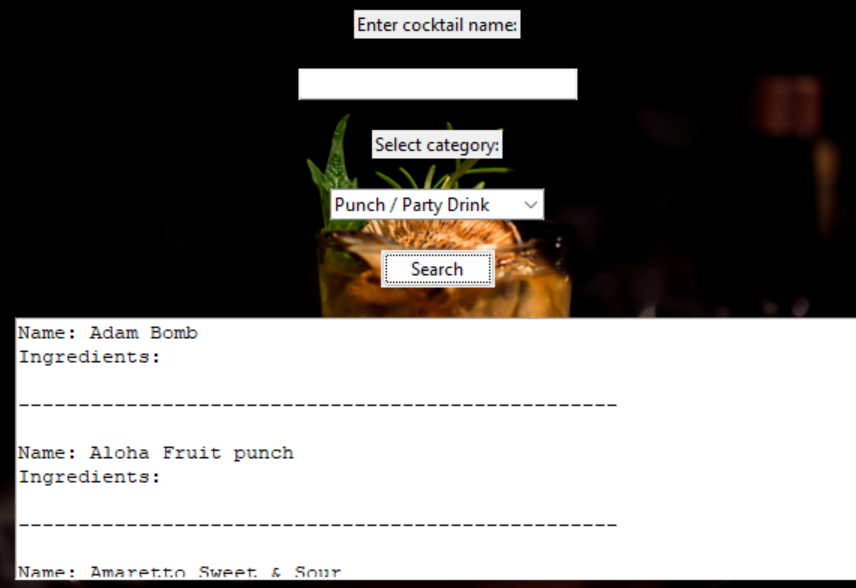
The beginning of testing was written on paper, and I thought of the basic things that would be added to the application.



First, I expected if the user searches, it will display all the information of the drink they want. This worked in the end, showing all the possible results related to the name searched.



When the user enters something that does not exist in the list, will it show the error message? In the end, the message box showed that this specific drink/cocktail did not exist in the list.



One part of the testing wasn’t included in the first part, which was for the category filter and this was added during the development phase. When the user chooses a category without searching, it only shows the name of the drink as it only identifies the drink alone. Then if the user searches for the name, it will show detailed information only for that specific drink.

The testing worked out well, as there were no major changes from the start and it all succeeded in the end but there was still something to learn.

**Critical Reflection**

***What is compelling about the work***

Creating something should always be about what the user needs. Compelling work appeals to the user and is interactive and fun. This application focuses on attaining information and making it easy for the user to find what they are looking for. The application is easy to navigate and understand. The UI elements are simple, but the background image makes the design look more engaging, balancing the simple UI elements. I added background music to enhance the user experience while using the application.

***What could be improved***

The planning should have more brainstorming for more project ideas. I could have added a separate filter for ingredients after the category. This can help the users have more options to filter their search. The design for the GUI could be better too. It could have had some extra interactions like a messagebox for the instructions alone. For testing, it was successful for most of it but I could have added more features that would have made the application better. Handling errors more effectively will also give the users a seamless experience while using the application.

Overall, it was successful and the plans were executed almost perfectly and were the same as expected. By learning, there are many things to be improved when I begin a new project involving Tkinter and APIs. It is important to keep track of the progress and never stop learning.

**Appendix**

**import tkinter as tk**

**from tkinter import ttk, messagebox**

**import requests**

**from PIL import Image, ImageTk**

**import pygame #use pygame for the background music**

**#a class made to organize all the elements for the Title Page**

**class TitlePage:**

**def \_\_init\_\_(self, root, switch\_to\_main):**

**self.root = root**

**self.switch\_to\_main = switch\_to\_main**

**self.frame = tk.Frame(self.root)**

**self.frame.pack(fill=tk.BOTH, expand=True)**

**#background image used**

**self.bg\_image = Image.open(r"C:\Users\aacun\Downloads\KHAETLIE\drink.jpg")**

**self.bg\_image = self.bg\_image.resize((600, 400))**

**self.bg\_photo = ImageTk.PhotoImage(self.bg\_image)**

**#make the label**

**self.bg\_label = tk.Label(self.frame, image=self.bg\_photo)**

**self.bg\_label.place(x=0, y=0, relwidth=1, relheight=1)**

**self.title\_label = tk.Label(**

**self.frame,**

**text="The Cocktail Finder",**

**font=("Bonheur Royale", 40, "bold"),**

**pady=20,**

**bg=None,**

**fg='black'**

**)**

**self.title\_label.pack(pady=10)**

**self.start\_button = ttk.Button(**

**self.frame,**

**text="Start",**

**command=self.switch\_to\_main**

**)**

**self.start\_button.pack(pady=20)**

**#make the title page disappear and show the main page**

**def destroy(self):**

**self.frame.destroy()**

**#a class made to organize the main page and its elements**

**class CocktailApp:**

**def \_\_init\_\_(self, root):**

**self.root = root**

**self.root.title("Cocktail Finder")**

**self.root.geometry("600x400") #window size**

**#background image used**

**self.bg\_image = Image.open(r"C:\Users\aacun\Downloads\KHAETLIE\drink.jpg")**

**self.bg\_image = self.bg\_image.resize((600, 400))**

**self.bg\_photo = ImageTk.PhotoImage(self.bg\_image)**

**self.bg\_label = tk.Label(self.root, image=self.bg\_photo)**

**self.bg\_label.place(x=0, y=0, relwidth=1, relheight=1)**

**#input label and entry**

**self.label = ttk.Label(self.root, text="Enter cocktail name:")**

**self.label.pack(pady=10)**

**self.entry = ttk.Entry(self.root, width=30)**

**self.entry.pack(pady=10)**

**#add a dropdown menu for categories**

**self.category\_label = ttk.Label(self.root, text="Select category:")**

**self.category\_label.pack(pady=10)**

**self.category\_var = tk.StringVar()**

**self.category\_dropdown = ttk.Combobox(self.root, textvariable=self.category\_var, state="readonly")**

**self.category\_dropdown.pack(pady=10)**

**self.load\_categories()**

**#search button**

**self.search\_button = ttk.Button(self.root, text="Search", command=self.search\_cocktail)**

**self.search\_button.pack(pady=10)**

**#result listbox**

**self.result\_area = tk.Text(self.root, wrap=tk.WORD, width=70, height=15)**

**self.result\_area.pack(pady=10)**

**#a function created for getting the categories from the API**

**def load\_categories(self):**

**api\_url = "https://www.thecocktaildb.com/api/json/v1/1/list.php?c=list"**

**try:**

**response = requests.get(api\_url)**

**response.raise\_for\_status()**

**data = response.json()**

**categories = [item['strCategory'] for item in data.get('drinks', [])]**

**self.category\_dropdown['values'] = categories**

**if categories:**

**self.category\_var.set(categories[0])**

**#show an error if the API request fails**

**except requests.RequestException as e:**

**messagebox.showerror("API Error", f"An error occurred while loading categories: {e}")**

**#a function which allows users to search for cocktails**

**def search\_cocktail(self):**

**cocktail\_name = self.entry.get()**

**category = self.category\_var.get()**

**if not cocktail\_name.strip() and not category.strip():**

**messagebox.showwarning("Input Error", "Please enter a cocktail name or select a category.")**

**return**

**if cocktail\_name.strip():**

**api\_url = f"https://www.thecocktaildb.com/api/json/v1/1/search.php?s={cocktail\_name}"**

**else:**

**api\_url = f"https://www.thecocktaildb.com/api/json/v1/1/filter.php?c={category}"**

**try:**

**response = requests.get(api\_url)**

**response.raise\_for\_status()**

**data = response.json()**

**#get the list of drinks from the API**

**drinks = data.get("drinks")**

**#drinks will be displayed when they are found in the API**

**if drinks:**

**self.display\_cocktail\_info(drinks)**

**#no drinks found means an error message is shown**

**else:**

**self.result\_area.delete("1.0", tk.END)**

**messagebox.showerror(tk.END, "No results found.")**

**except requests.RequestException as e:**

**messagebox.showerror("API Error", f"An error occurred: {e}")**

**# a function made to display all the information to the users**

**def display\_cocktail\_info(self, drinks):**

**#clear the result before displaying new information**

**self.result\_area.delete("1.0", tk.END)**

**for drink in drinks:**

**#show the name, cattegory, and instructions**

**name = drink.get("strDrink")**

**category = drink.get("strCategory")**

**instructions = drink.get("strInstructions")**

**#get all the ingerdients needed**

**ingredients = [**

**f"{drink.get(f'strIngredient{i}')}: {drink.get(f'strMeasure{i}', '')}".strip()**

**for i in range(1, 16)**

**if drink.get(f"strIngredient{i}")**

**]**

**#insert the name to the result listbox**

**self.result\_area.insert(tk.END, f"Name: {name}\n")**

**#display provided category**

**if category:**

**self.result\_area.insert(tk.END, f"Category: {category}\n")**

**self.result\_area.insert(tk.END, "Ingredients:\n")**

**self.result\_area.insert(tk.END, "\n".join(ingredients) + "\n")**

**#display any provided instructions**

**if instructions:**

**self.result\_area.insert(tk.END, f"Instructions: {instructions}\n\n")**

**self.result\_area.insert(tk.END, "-" \* 50 + "\n\n") #to separate each drink**

**def main():**

**#background music used for the application**

**pygame.mixer.init()**

**pygame.mixer.music.load(r"C:\Users\aacun\Downloads\KHAETLIE\cocktaildb.mp3") #downlaoded music**

**pygame.mixer.music.play(-1)**

**root = tk.Tk()**

**root.title("Cocktail Finder")**

**root.geometry("600x400")**

**def switch\_to\_main():**

**title\_page.destroy()**

**app = CocktailApp(root)**

**#run the application**

**title\_page = TitlePage(root, switch\_to\_main)**

**root.mainloop()**

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

**main()**