ASSESSMENT 2: UTILITY APP

|  |  |
| --- | --- |
| **Students Name** | Myesha kashif |
| **Github Repository Name** | myesha\_data\_driven\_app |
| **Github Repository Link** | <https://github.com/myeshaa18/myesha_data_driven_app> |
| **Repository Screen Shot** |  |

**DOCUMENTATION**

***Abstract:***

This Juice World App is a cocktail exploration application built using python's Tkinter library for the GUI and the CocktailDB API for fetching cocktail data. It provides users with various features such as discovering random cocktails, alcoholic or non-alcoholic drinks, and ingrident details. The app is designed with a user-friendly interface, allowing user-friendly interface, allowing users to view cocktail names images and descriptions with just a few clicks.

The app communicates with The CocktailDB API, a public API that provides a wide variety of cocktail recipes and ingredient information. Users can search for cocktail recipes and ingrident information. Users can search for cocktails based on alochol content, explore a random cocktail, or search for specific ingredients. The app also provides real-time images of the cocktails fetched via the API.

The application consists of several frames, each providing a different feature. The home scfeen offers access to tje apps funtions, while other frames allow users to interact with specific cocktail categories. Each category pulls data from the API and displays it dynamically

To access the full source code, you can visit the GitHub repository: [myeshaa18/myesha\_data\_driven\_app](https://github.com/myeshaa18/myesha_data_driven_app)

***PROJECT PLAN:***

**DAY 1 : PLANNING ( Total 6 HOURS )**

**Project Concept and Requirements (2hr)**

Finalize the app's core features: random cocktail generator, alcoholic/non-alcoholic filtering, ingridient search, and cocktail details display.

Determine the API requirements, specifically which endpoints, from TheCocktailDB API to intergrate.

Outline the app's user flow, deciding on the essential frames and their interactions (e.g home, random cocktail, ingridient search).

**Timeline and resource planning (2hr)**

Establish the development timeline with daily goals to ensure steady progress.

Identify and allocate resources (e.g, Python libraries, images assets).

Assign roles and tasks if working in a team (e.g, one person focuses on API integration, another on UI design).

**Risk Assessment and Mitigation (2hr)**

Identify potential risks such as API response failures, image loading issues, or UI bugs.

Plan for backup solutions and testing strategies to mitigate those risks.

**Day 2 - Design (TOTAL 6 hrs)**

**UI / UX Design ( 3hr )**

Design wireframes for each app screen ( e.g., home screen, random cocktail screen, ingredient search).

Decide on color schemes, fonts, and layout to ensure the app is visually appealing and easy to use.

Sketch how each feature ( like cocktail name display image display) will appear on the screen.

**App Architecture Design ( 3hr )**

Plan the structure of the app, including how different frames ( screens ) interact with each other.

Define the key components in the app, such as buttons, labels, and images and where they will placed within the apps interface.

Organize the flow of data from the API into the apps GUI to ensure smooth transitions between different views.

**Day 3-4 DEVELOPMENT ( TOTAL 12 hrs)**

**Frame Setup and basic GUI ( 6hr ):**

Start developing the basic structure of the app using Tkinter, creating frames for each screen (e.g, welcome screen, button options, cocktail display).

Design and integrate static elements like labels, buttons, and backgrounds.

Setup the main window with necessary attributes like title, size, and layout.

**API Intergrate (6hr):**

Implement functions to interact with TheCocktailDB API to fetch cocktail data ( random cocktails, alcoholic/non-alcoholic cocktails).

Develop methods to extract relevant data from the API's response (cocktail names, images, descriptions, etc.).

Integrate image loading functionality ( fetching images via URLs and resizing them for the GUI).

test API calls to ensure the data is correctly retrieved and displayed in the app.

**DAY 5 DEVELOPMENT CONTINUED (TOTAL 6hr)**

**Ingrident Search and display (3hr):**

Implement the search functionality for ingredients using the appropriate API endpoint.

Create a text box for the user to input an ingrident name, description, alcohol content).

Ensure that the data is dynamically fetched and displayed in the app.

**Feture Testing and debugging (3hr)**

Start debugging code for any issues with the API response or UI layout.

Conduct initial testing to check the flow between frames, ensuring that buttons navigate users to the correct screens.

Test if images and data appear correctly after API requests.

**DAY 6 TESTING (TOTAL 6hr)**

**User interface testing (3hr)**

Test the apps user interface on different screen sizes to ensure it's responsive.

Check the clarity of text, button placements, and visual appeal of the images. Make adjustments to the UI to fix any issues with alignment, font readability, or button size.

**Functional Testing ( 3hr)**

Test the apps core functionalities:

Check if random cocktails are displayed with the correct images and names.

Test alcoholic and non-alcoholic filters to ensure they return relevant results.

Validate the ingredient search functionality and its dynamic data display.

Conduct stress testing on the API by making multiple requests to ensure it responds well under different conditions.

**DAY 7 FINAL ADJUSTMENTS AND DEPLOYMENY (TOTAL 6 hr)**

**Buf Fixes and final adjustments (3hr):**

Address any last-minute minute bugs or issues that appeared during testing.

Ensure that error handling ( e.g, for failed API requests) is in place to provide users with a smooth experience.

Make any necessary tweaks to the design based on user feedback or testing.

**FINAL Deployment and documentation (3hr):**

Prepare the app for final deployment by ensuring all required files are included and properly linked.

Create detailed documentation for the project, including setup instructions, key features, and any known

**Final Deployment & Documentation (3 hours):**

Prepare the app for final deployment by ensuring all required files are included and properly linked.

Create detailed documentation for the project, including setup instructions, key features, and any known issues.

Push the code to Github with a well organised repository structure.

**TOTAL ESTIMATED TIME 42 HOURS**

This project plan ensures steady progress over a week while allowing time for testing and refinement. It balances the core tasks of design, development, and testing while ensuring that the app is both functionaland user-friendly. Each milestone focuses on delivering a complete component of the app.

***EVIDENCE OF DESIGN***

***PSEUDO CODE***

1. START APPLICATION

- Initialize the main window with the title and dimensions.

2. DEFINE FUNCTIONS

- \*\*Get Random Cocktail:\*\*

-Fetch a random alcoholic colcltail from the API.

-Display the name and ID of cocktail.

-Download and sow the cocktail image.

- \*\*Get Alcoholic Cocktail:\*\*

-Fetch a list of cocltail from the API.

-Select a random cocktail.

-Display the name, ID, and image of selected cocktail.

- \*\*Get Non-Alcoholic Cocktail:\*\*

-Fetch a list of non aloholic cocltail from the API.

-Select a random cocktail.

-Display the name, ID, and image of selected cocktail.

- \*\*Search Ingridient:\*\*

- Get the user input for an ingredient name.

- Fetch Ingridient details from the API.

- Display ingridient information or an error message if not found.

- \*\*Show Instructions:\*\*

- Show a message with app instructions.

\*\*Switch Frames:\*\*

-Make the specified frame visible.

3.CREATE FRAMES

\*\*Welcome Screen (Frame1):\*\*

-Display app name and background image.

-Add buttons for instructions and navigating to the app menu.

\*\*Button Menu (Frame2):\*\*

-Display button for each feature.

-Random Cocktail.

-Alcoholic Cocktail.

-Non-Alcoholic Cocktail.

-Search Ingredient.

\*\*Random Cocktail Frame (Frame3):\*\*

-Show a button to fetch a random cocktail.

-Display cocktail details and image.

\*\*Alcoholic Cocktail Frame (Frame4):\*\*

-Show a button to fetch a random alcoholic cocktail.

-Display cocktail details and image.

\*\*Non-Alcoholic Cocktail Frame (Frame5):\*\*

-Show a button to fetch a random non-alcoholic cocktail.

-Display cocktail details and image.

\*\*Ingridient Search Frame(Frame6):\*\*

-Porvide a text ox for user input.

4. SET BACKGROUND LAYOUTS

.\*\*Set Backgrounds and Layouts:\*\*

-Add background images to each frame.

-Arrange buttons, labels, and other components.

5. SHOW INITIAL FRAME

\*\*Show Initial Frame:\*\*

-Display the welcome screen as the first frame.

6. RUN APPLICATION

.\*\*Run Application:\*\*

-Start the app's main loop.

This pseudo-code breaks down my code into logical components, making it easy to understand and follow.

**FLOWCHART**



**WIREFRAMES**



***TECHNICAL DESCRIPTION:***

The "Juice World App" is an interactive program built using Tkinter to create a user-friendly graphical interface. It uses the CocktailDB API to fetch and display information about random cocktails, alcoholic drinks, non-alcoholic drinks, and ingredients. The app is divided into multiple frames for different features, allowing easy navigation and a clean interface.

**1. GUI Structure:**

The app uses Tkinter frames to create different screens for functionalities like random cocktails, alcoholic drinks, non-alcoholic, and ingredient search.

Labels, buttons, and text fields display information and let users interact with the app.

Images are dynamically resized using the Pillow (PIL) library to fit the interface.

**2. Fetching cocktail data:**

The app interacts with the CocktailDB API to fetch data.Functions like get\_random\_cocktail().get\_alcoholic\_cocktail(). and get\_non\_alcoholic\_cocktail() send requests to the API and display the drink name, ID and image.

For ingredient search, search\_ingredient() allows users to input an ingredient name and fetch its details, including type, alcohol content, and description.

**3. Navigation:**

Users can switch between frames using buttons like "Go Back" or Go to App." The show\_frame() function ensures a smooth transition between views.

**4. Error Handling:**

If no input is provided for ingredient search, a message box displays an error, improving usability.

The app integrates dynamic updates with a structured design, offering a seamless experience for exploring cocktail recipes and ingredients in real-time.

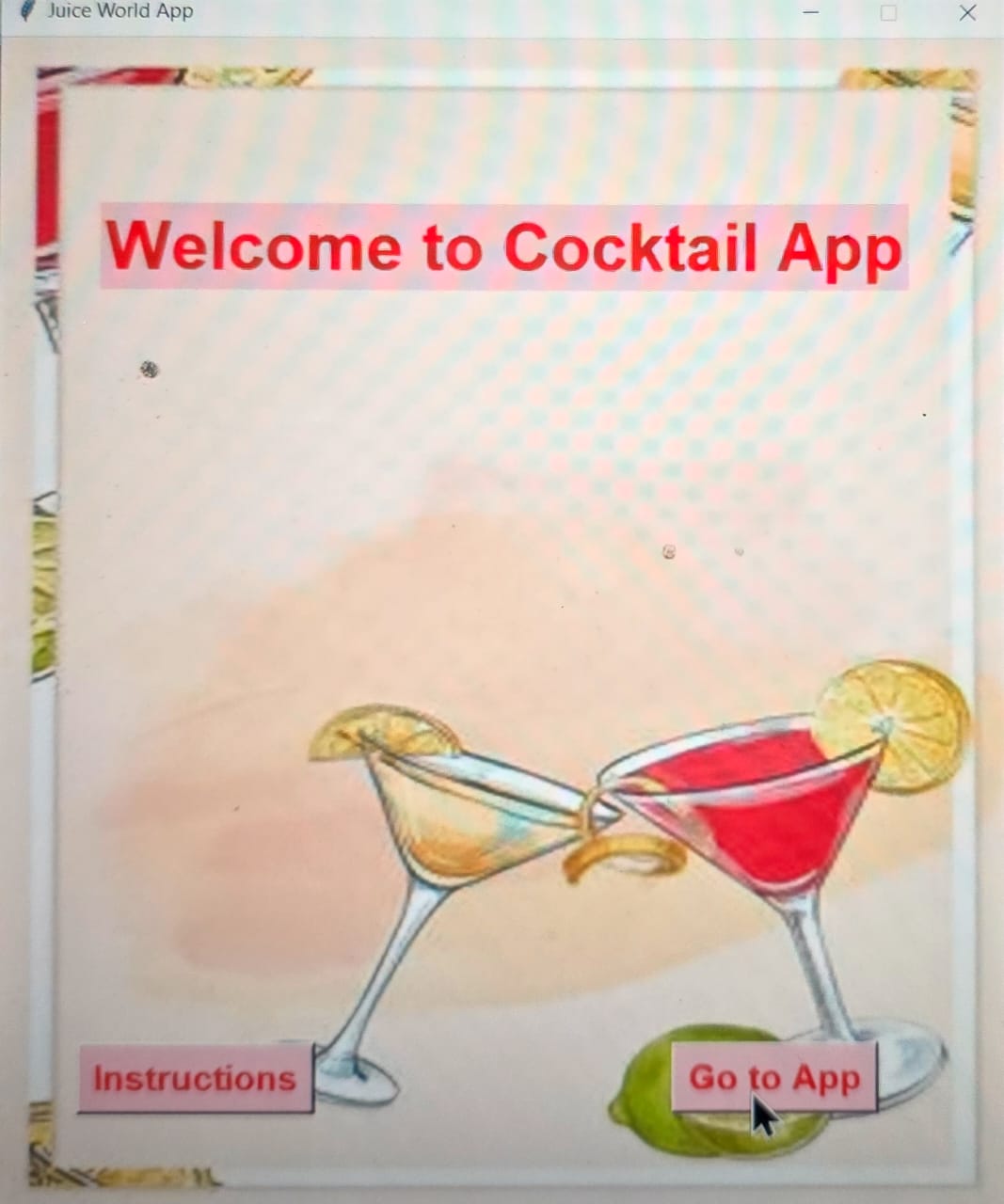
***WALKTROUGH VIDEO:***

<https://youtube.com/shorts/47PbqkHGDdw?feature=share>

***TESTING***

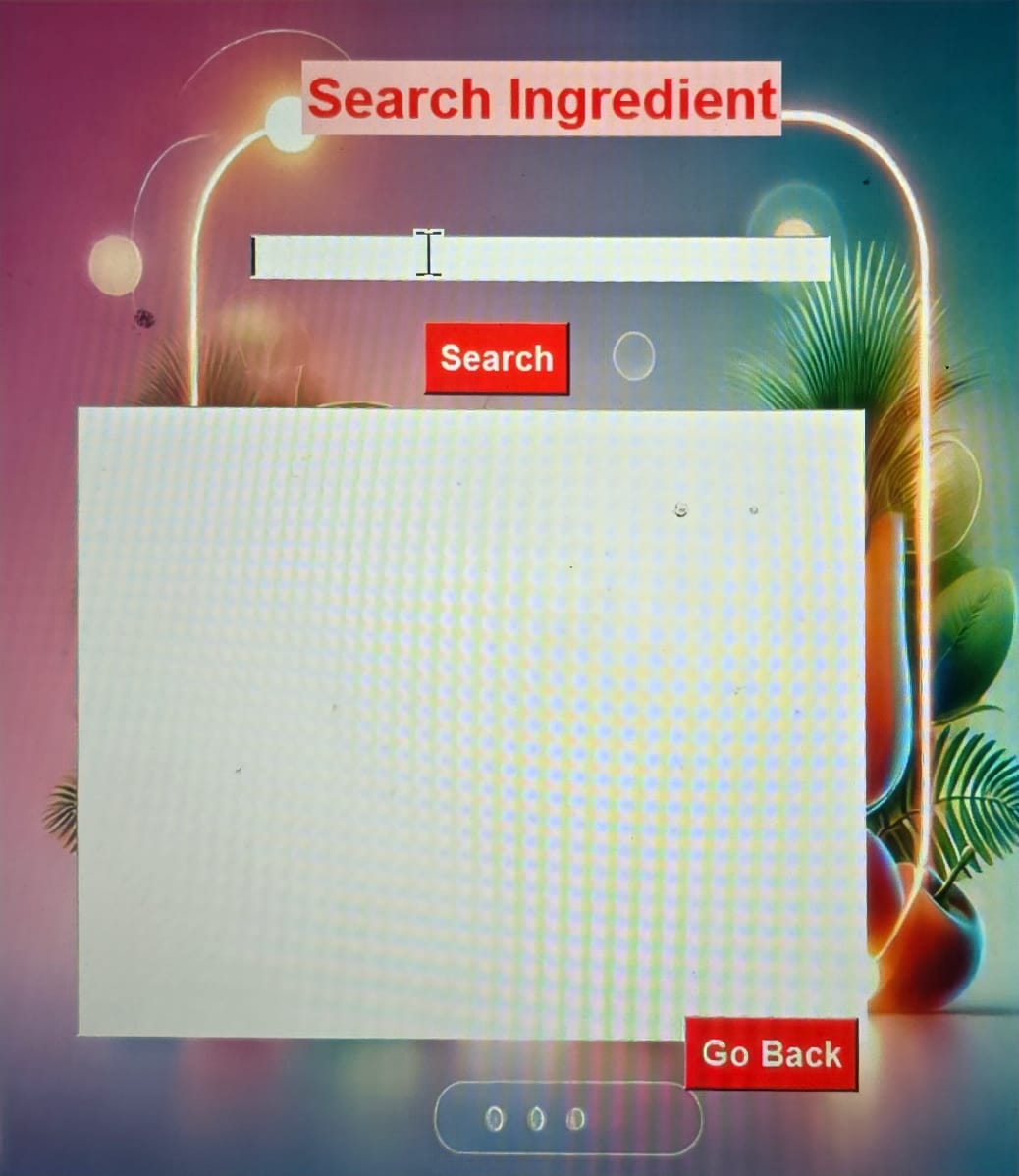
|  |  |  |  |
| --- | --- | --- | --- |
| Functionality | input | Ecpected output | pass/fail |
| Welcome Screen navigation | Click “Go to App” | App navigates to the button Menu Frame | pass |
| Instructions Popup | Click “Instructions” | A popup with instuctions is displayed | pass |
| Random cocktail | Click “random Cocktail” | Fetch and display random cocktail name, ID, and image | Pass |
| alcoholic Cocktail | Click “alcoholic Cocktail” | Fetch and display an alcoholic cocktails name, ID, and image. | pass |
| Non-alcoholic Cocktail | Click “non-alcoholic Cocktail” | Fetch and display an non-alcoholic cocktails name, ID, and image. | pass |
| Ingridient Search | Enter ingridient name (e.g, Vodka) and click “Search” | Display ingridient details (name, type alcohol content, desription) | pass |
| Ingridient Search  (Error) | Enter invalid ingridient name (eg, XYZ) and click “Search” | Display “No ingridient found” message | pass |
| Navigation to previous Frame | Click “Go back” on any frame | Return to the Button Menu frame | pass |
| API Error Handling | Disconnect from internet and try fetching a cocktail | Show error or handle gracefully (e.g, display “Unable to fetch data”) | pass |
| UI Responsiveness | Test UI Layout on different window sizes | UI components stay aligned and usable | pass |

**SCREENSHOTS**

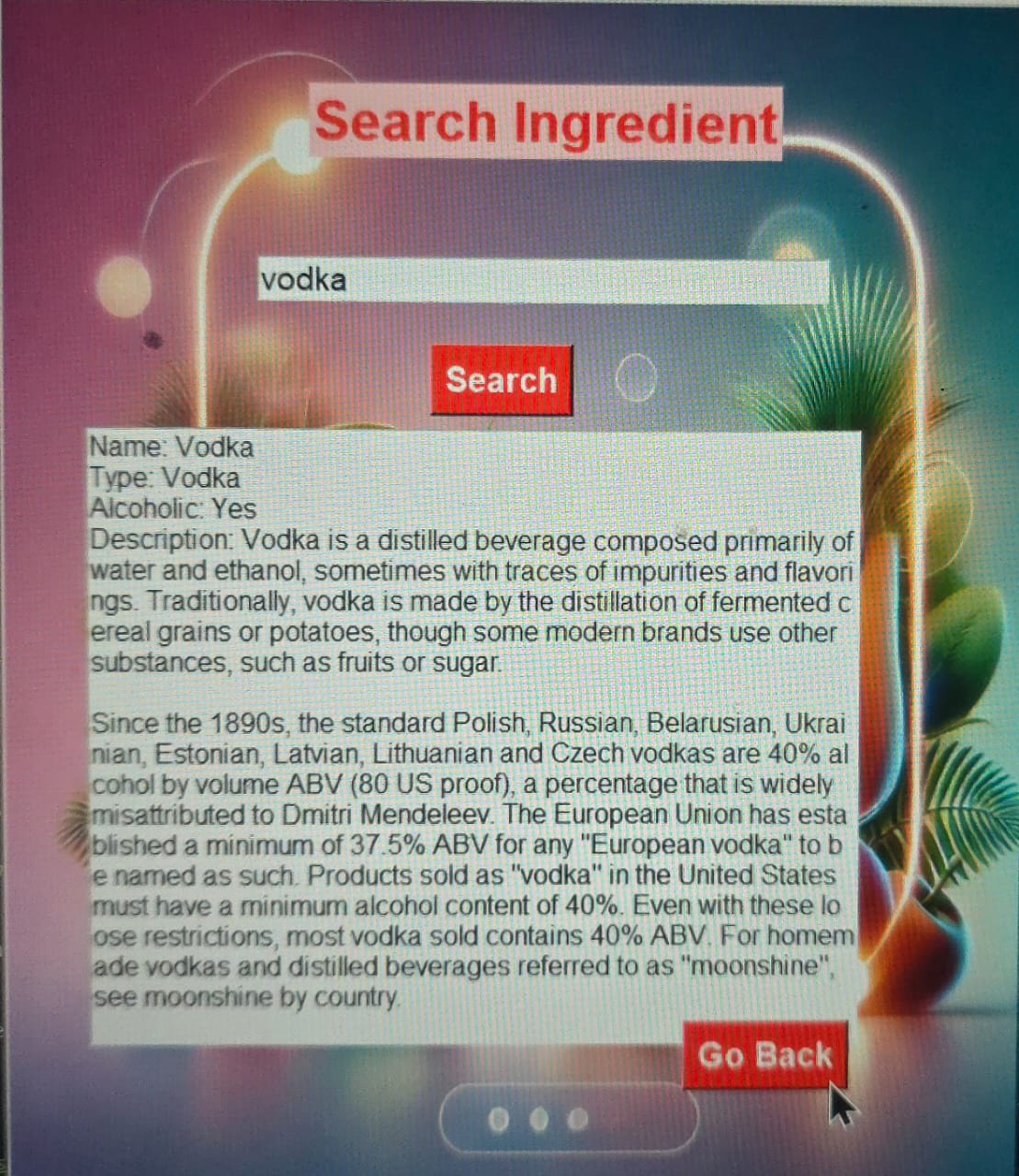


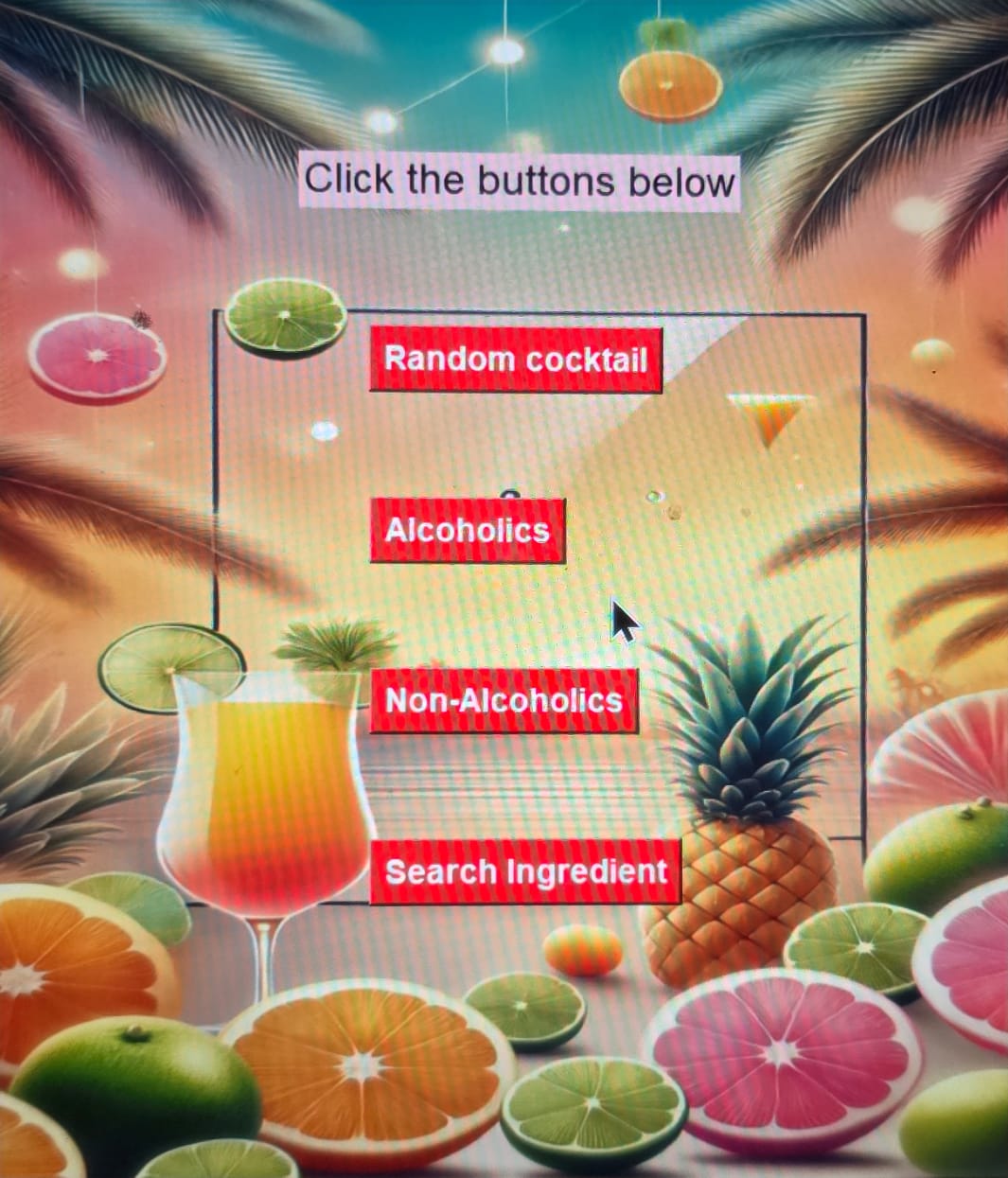












***CRITICAL REFLECION:***

**STRENGHT**

1. User-friendly: In Intuitive mavigation and clear functionality enhance the user expierence.

2. API Integration: Real time cocktail data makes the app engaging.

3. Visual Appeal: Dynamic images add an aesthethic touch.

4. Modular Code: Separate frames and functions makes maintenance easier.

**IMPROVEMENTS NEEDED**

1. Error Handling: Add better validation for API failures and network issues.

2. UI Design: Modernize the interface with consistent themes and layouts.

3. Code Optimization: Refactor repetitive patterns for efficiency.

4. Scalibility: Introduce features like favourites or mocktail categories.

5. Testing: formal testing for edge cases and robustness.

**LEARNING GOALS**

-Improve Tkinter design skills.

-Explore advanced error handling and logging.

-Learn modern UI frameworks like PyQt or Kivy for better visuals.

This app has great potential but could benefit from refined UI, better robustness, and additional features.