Planning and Requirements Document
Lingxiao Yan (ly178@nyu.edu)

Cocktail Search Service

Background

As alcoholic drinks with spirits, fruit juice, flavored syrup, or cream, cocktails have a history over hundreds of years. So far, based on the number and kind of ingredients added, various types of cocktails have been invented by people all over the world. For example, the types of cocktails with one ingredient, gin, have exceed 50, and they all have their own special names.

On the one hand, the variety of cocktails represents how popular these drinks are. It is no exaggeration to say that cocktails are loved and well received across the world. The reason behind this may lie in their refreshing tastes and variety of ingredients which allow people to customize in their favor. Even you're not a cocktail guy, because of the frequent presence of cocktails in social settings, you may need to know some knowledge about cocktails.

On the other hand, so many kinds of cocktails make it hard to correctly order and taste them. This is because, as I said before, hundreds of cocktails are named differently and some of them have names which don't indicate ingredients at all.

Therefore, considering that cocktails are important for our lives and it's not an simple work to have a comprehensive understanding of cocktails, professional help seems to be in urgent need.

Problem Statement

User Needs

For people who may drink cocktails, whether for pleasing themselves or for social reasons, they would like to get relevant information about one or more cocktails easily and directly. In some special cases, if they want to look for fun, random recommendation may be helpful for them to try new things.

Current State Processes

Currently, people may gather information about cocktails through web search or introduction of others. For example, using Wikipedia or learning from cocktail experts like bartenders are common options. But these two approaches both have shortcomings. For the former, sometimes users may need to navigate around several web pages to get complete or desired information. While for the latter, it's definitely possible that there are no cocktail experts whom we can touch with right now.

Proposed Solution

System Objectives

My proposed solution is called the “Cocktail Search Service”. It mainly provides “one-stop” service which can enable users to search cocktails by name or ingredient through simple written commands in Command-Line Interface (CLI) or through gestures such as clicking in Graphical User Interface (GUI). Instead of opening multiple web pages or consulting others, this “Cocktail Search Service” is going to simplify the process to get cocktail information. Besides, it supports fuzzy search and refined search at the same time, which means when there are multiple cocktails whose names or ingredients have the keyword that a user enters, this system will return all of these cocktails. And if users are interested in one of them, they can enter the name of that particular cocktail to gather relevant information. For example, when users input “margarita” as name to search cocktail, what they get is a list that includes Margarita, Blue Margarita, Tommy's Margarita, Whitecap Margarita, Strawberry Margarita. In the next step, they can refine their search by entering the name of one of the cocktails in that list to know more about it.
Besides, if users want to look for more fun, this “Cocktail Search Service” can select one cocktail randomly and return its information.

Future State Processes

The system should simplify the current state process so that users can get complete information about one or more cocktails more easily.
After the system is developed and deployed, my future process will involve the following steps:

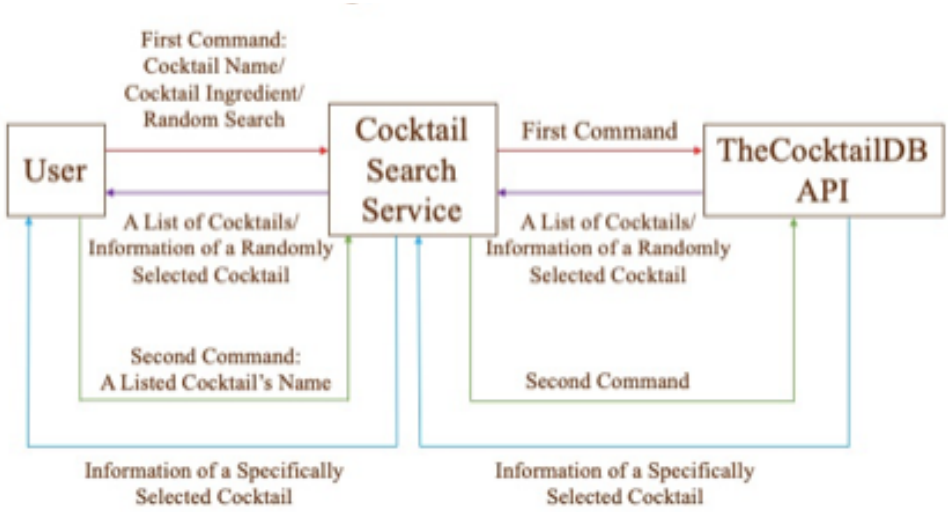
1. Allow users to search cocktails by name or by ingredient. It can be fuzzy search, which means the system will return all of cocktails when their names or ingredients have the keyword that users enter. It also can be refined search, which means users can obtain the information of a specific cocktail. Information includes its name, ingredients, type of glass, drinking instruction, and image.
2. Allow users to have some fun by randomly selecting one cocktail for them. Information like its name, ingredients, type of glass, drinking instruction, and image will be outputted.

Information Requirements

Information Inputs and Outputs

In the first step, users can enter “name” or “ingredient” in CLI or click on “search cocktail by name” or “search cocktail by name” in GUI to start searching. After that, they need to specify what name or ingredient they would like to search. Then, the system will return a list of cocktails with user-entered keyword in their names or ingredients. Next, if they’re interested in one cocktail in that list, they can enter its name to continue searching. The system will output the name, ingredients, type of glass, instruction, and image of that cocktail. If they want to stop searching, entering “DONE” in CLI or clicking on the “Cancel” button will make the system stop working.
Of course, in the first step, users can also enter “random” in CLI or click on “random recommendation” to make the system randomly select one type of cocktail for them. The information outputs include the name, ingredients, type of glass, instruction, and image.

Data Flow Diagram



Functionality Requirements

Reading API Key

The system should keep API Key as secret but have access to it in the main script. But for the API this system uses, TheCocktailDB API, all users can use the developer key “1” as the API key to do some basic search.

Compiling Request URLs

Because users may conduct multi-level searches (e.g. input “margarita” to search cocktail by name and then input “Blue Margarita” to obtain information about that

specific cocktail), the system should construct the expected URLs correctly every time.

Issuing API Requests

The system should return the expected response data in a usable format. One important thing is that the resources of this API are updated, so we can't define all the valid inputs in advance, which makes it possible for users to give an invalid input to the system, such as a cocktail name that doesn't exist. When this occurs, the particular API still uses 200 codes even when there is an error. Thus, it's necessary to guarantee that this system is able to prevent an HTTP request if users' input is not valid.

Processing API Response

For the final outputs, the system should process the raw data to make it more readable. In other words, the instruction and structure of outputs should be clear so that users can easily understand what information they get from their previous requests.

Passing Automated Testing

In the automated testing, I gather the information ahead of time about a specific cocktail "Margarita" (e.g. ingredients; type of glass; instruction) from the API's raw data in advance. Then, my goal is to test whether the "def" functions in the file containing the application's logic can return these desired information.

Interface Requirements

There are two ways for users to do cocktail search. One is operating directly via written textual commands in Terminal for Mac users or Command-Prompt for Windows users. The other one is via gestures such as clicking in GUI. Different from CLI, GUI needs to check the operation users have made as well as information input. Specifically, if users don't give information input or give an invalid information input, but click on the "OK" button, the system should show a window containing the error message. If users just click on the "Cancel" button, the search window should be closed directly.

Technology Requirements

Python language is used to implement the logic of this system. It will use at least the following Python modules and packages:

- The “json” module to convert a JSON-formatted string into a Python dictionary.
- The “os” module to access environment variables that have been loaded.
- The operation “BytesIO” of the “io” module to keep data as bytes in an in-memory buffer.
- The “Image” module from the library “PIL” to load images.
- The “requests” package to issue HTTP requests to exchange data with an API.
- The “dotenv” package to reference environment variables from a project-specific ".env" file.
- The “pytest” package to test whether a program is producing desired functionality.
- The “PySimpleGUI” package to provide an application with a graphical user interface capable of being run on a personal computer.

Development Plan

Collecting Users’ Command and Information Input

No matter it’s within CLI or GUI, the first step is to collect users’ command. That is, does a user want to search cocktail by name, ingredient, or just make the system has a random selection. If users choose to search cocktail by name or by ingredient, they need to input what name or ingredient they want to search.

Constructing URLs, Issuing API Requests, and Parsing API Response

Based on users’ command and information input, the system should construct corresponding URLs, and use these URLs to request data from TheCocktailDB API and parse it to a json format. For the two functions, “search cocktail by name” and “search cocktail by ingredient”, the system will check whether the name or ingredient users just give is valid. If it’s not, the system will pop up an error message for users.

Returning Meaningful Information about Cocktails

After checking the information input is valid, the system will return relevant information in a user friendly way. Basically, if users choose to conduct search by themselves at the beginning, what they should be able to know is a list of cocktails with the user-entered keyword either in their names or in their ingredients. If they keep searching, a listed specific cocktail’s name, ingredients, type of glass, drinking instructions and image should be shown. However, if users choose to get random recommendation, a randomly selected cocktail’s information should be outputted instead.