Web Service CA Documentation: CocktailAPI

By Harsh Khatri

Date: 31.08.23

Table of Contents

[Research 2](#_Toc144409451)

[Design 2](#_Toc144409452)

[Visuals: 2](#_Toc144409453)

[User Page: 4](#_Toc144409454)

[Database Considerations 4](#_Toc144409455)

[Client Database 4](#_Toc144409456)

[Server Database 5](#_Toc144409457)

[API Documentation 5](#_Toc144409458)

[Premium 1: Search By name 6](#_Toc144409459)

[Premium 2: Search by Category and ingredient 7](#_Toc144409460)

[Problems Encountered 8](#_Toc144409461)

[Database Default names: 9](#_Toc144409462)

# Research

For the Web Service CA, I opted to integrate the [cocktailsdb](https://www.thecocktaildb.com/api.php) as a third-party API. This API offers various functions. I implemented a free-to-use random cocktail feature (with a limit of 10 uses per day). Premium features include searching for cocktails by name and the second feature being ingredient and category —features reserved exclusively for premium members. I selected this API because of my love for cocktails being in a student accommodation, and using this API simplifies the process of finding a cocktail so much easier.

# Design

Visuals: I used Bootstrap for the entire project making sure everything was symmetric and mobile friendly following a light color scheme to make the project visibly pleasing.

Example of Home UI


MVC SS
  
MVC: I followed a simple MVC framework to avoid any errors and settled on making as many functions as possible and using them whenever needed. Code was split into view as much as needed and only used whenever required.   
  
Validation: There was validation done in nav bar and indexes and action elements and service to make sure users weren’t accessing the pages they weren’t supposed to. Isset($variable) was used for session elements to make sure there weren’t any errors in handling them. Nav Links were disabled if user wasn’t premium. User couldn’t go into login page if he wasn’t logged in.   
  
Comments were added to code wherever needed to make it more understandable and easier to read.

Validation in nav bar code:

A screen shot of a computer code

Description automatically generated

## User Page:

Also I added a user profile page where user can change their status from free to premium on both client and server side. The page can be accessed by Clicking Hi, Username link next to logout.

A screenshot of a computer

Description automatically generated

# Database Considerations

Client Database: Only one table was taken called user where users username, password, member type and id were stored, this was done for simplicity and to make sure a lot of data wasn’t stored on the client side but on the server side

A screenshot of a computer

Description automatically generated

Server Database: Server database had a similar table as client for users but with an APIKey stored in it which was generated in the server as well, I decided not to store it into client side but only in server. The APIKEY is username were unique fields for obvious reasons and id was auto incremented. Ingrediant database wasn’t Implemented at the end as cocktailDB provided a wider range.  
UsageUser table had todays usage and premium date so when a premium membership expires a user is notified and to see if free usage exceeded 10 or not.

A screenshot of a computer

Description automatically generated

# API Documentation

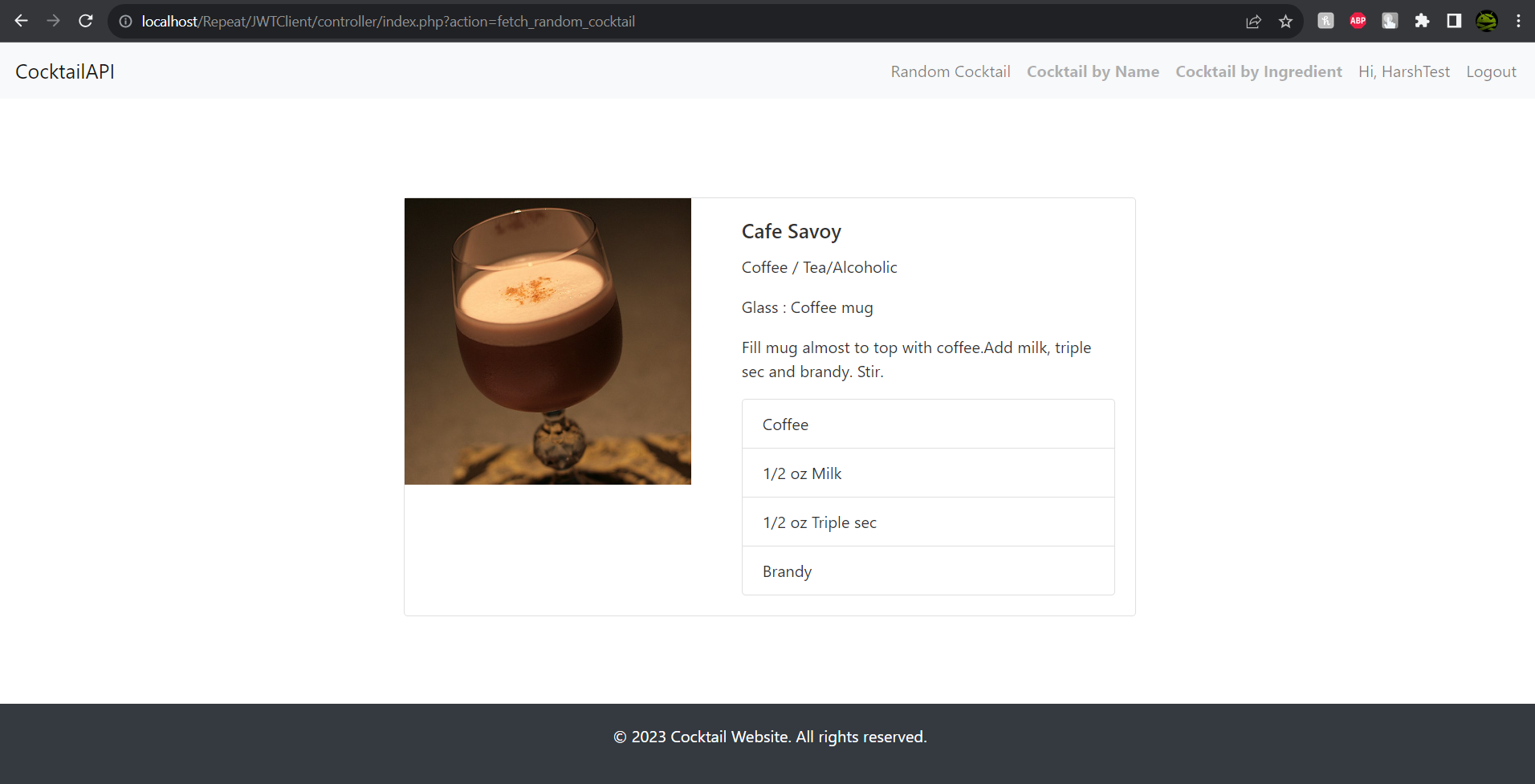
API documentation was simple for the three services offered.  
  
Free: Random Cocktail

Free service was to generate a random cocktail retrieved from the [www.thecocktaildb.com/api/json/v1/1/random.php](http://www.thecocktaildb.com/api/json/v1/1/random.php) . Before doing it was checked if the service was used 10 times today or not and if it was then user wasn’t allowed to use the service again till cooldown of next day by ensuring todaysUsage and UsageDate and server side code. The link to access it in server was as follow.

A screen shot of a computer

Description automatically generated

Screenshot of service in use:



Premium 1: Search By name  
  
The service fetched all cocktail from the server matching the name provided. As filler in the code I used my name as there are no cocktails starting from my name ( I will create one, one day). The service could only be accessed by premium users and displays all the cocktails starting with the name, The cocktail db link used was:   
www.thecocktaildb.com/api/json/v1/1/search.php?s=margarita.  
An example screenshot of it in use when searched sour:

A screenshot of a website

Description automatically generated

Premium 2: Search by Category and ingredient  
  
This service fetches all the drinks which matches a certain category and ingredient type from the database it works on two links (listed below) and then finds the drink that matches both categories. The dropdown menu consists of all the inggrediants and category listed on the cocktaildb

Filter by Category: [www.thecocktaildb.com/api/json/v1/1/filter.php?c=Cocktail](http://www.thecocktaildb.com/api/json/v1/1/filter.php?c=Cocktail)

Search by ingredient: [www.thecocktaildb.com/api/json/v1/1/filter.php?i=Gin](http://www.thecocktaildb.com/api/json/v1/1/filter.php?i=Gin)  
  
The Ingredients and categories are stored in an array and then looped to retrieve them, the result produced by them takes long as the cocktaildb has a lot of drinks. The example of result is below:   
  
A screenshot of a recipe

Description automatically generated  
  
In the above ss default values stored in the code were used, in the example below a dry vermouth based cocktail was used   
  
A screenshot of a recipe

Description automatically generated

# Problems Encountered

The development process was not without its challenges.There were a lot of times were I had to print out every variable on both client and server side to see where the error was occurring and sometimes even that wasn’t enough.  
  
Session: For a while for some reason I assumed that I could save session stuff in server and access it in client but it wasn’t the case at all and I used curl and urlencode to return and access variables whenever I needed to.  
  
Urlencode: I had a glitch where apikey wasn’t transferred properly and after 3 painful hours of figuring why I realized it was because of a space saved in session api key which was fixed by adding a simple trim statement.

Database Default names: Apparently usage cant be used as a table name as it doesn’t let php accesss it at all which I didn’t realise for a while and thought the problem was in either server or client code which finally led to me having a breakdown and asking a question on stackoverflow ( I never did that caus of how mean people on that website are) and to my surprise they were really mean about a lot of my syntax and helped me find the error which was database name. The link to it is below for your amusement. [Mean Link](https://stackoverflow.com/questions/77002589/why-isnt-sql-adding-to-table-but-returning-true)