## Project 4 Task 2 – Meal Search App
## <Khushi Bhuwania> (AndrewID: kbhuwani)

**Description:** MealDB is an mobile application which is used to search meals based on ingredients available.We are creating a WEB Service using JAVA Servlet which will handle All the API calls to TheMealDB endpoint documentation for which is available at “https://www.themealdb.com/api.php”. The android application will have a search box with a search button where user will enter an ingredients name i.e. “Rice”, “Chicken” will send this query to the web service in json format. Then our web service will call the action to fetch data from API and it will send it back to the android application. And in our android application we will show the results and clear the search area to be used again.

Also all the search queries will be logged in our mongoDB database by our web services, and it will present a page in which top 10 searched items will be displayed with the number or request done on that query.

The application will be developed using Android Native Code and The Web Services will be made in JAVA Servlet and will be hosted on GitHub Codespace

**1. Implement a native Android application**

The name of my native Android application project in Android Studio is: MealDBDemo

**Project Name:** MealDBDemo in Android Studio

**Views Used:** TextView, EditText, ImageView, Button, RecyclerView.

**User Input:** “Ingredient of meal” i.e. “Rice”

**HTTPRequests:**

POST requests to MealDB API endpoints via web service hosted on GitHub Codespaces.

**JSONParsing:**

Extracts and displays meal image, name and country of origin from database and creates and JSON array to be shown as in the form of a List

**Display Information:**

Once API hits it loads the available meals details in recyclerView with a dynamic adapter which maps all josn object to an view layout

**Reusability:**

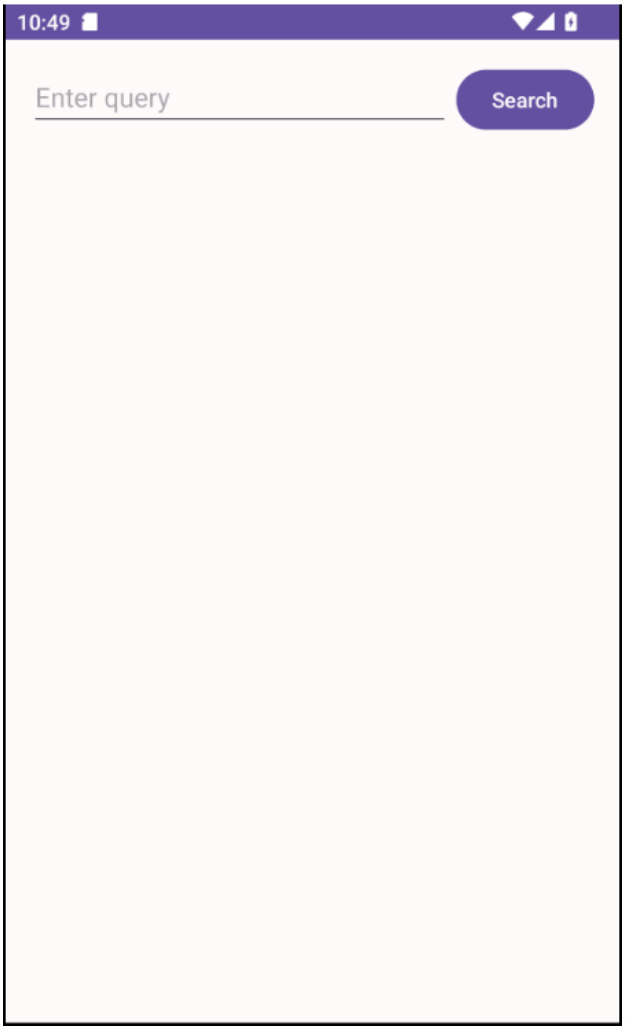
Once searched once ingredients it clears the edittext to be used for continuous interaction, providing real-time updates without needing to restart the application.

a. The main screen consists of the following views:

   TextView, EditText, ImageView, BUtton, RecyclerView

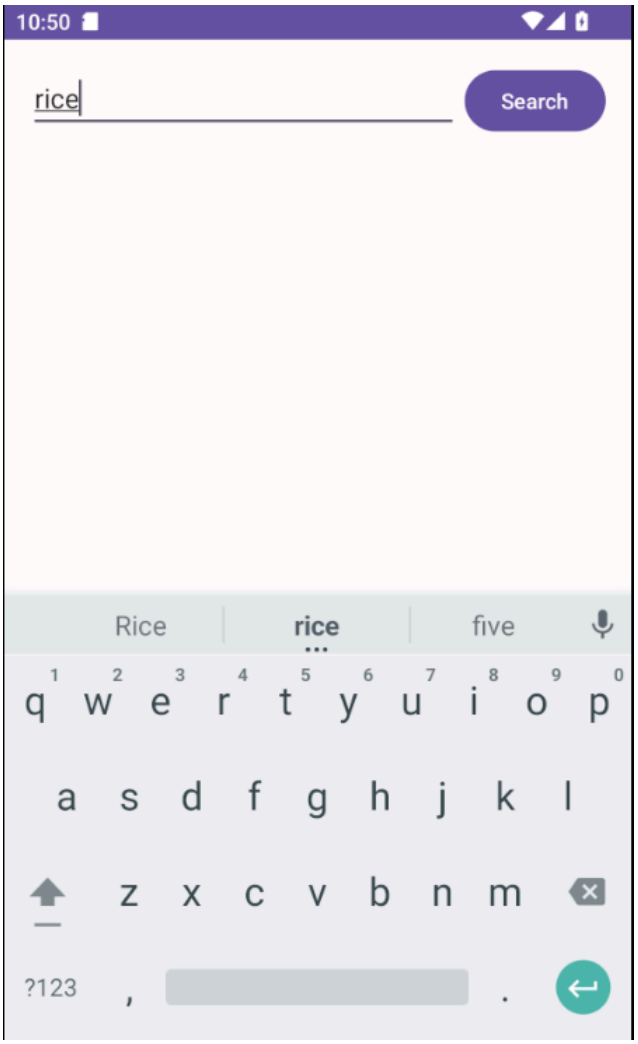
   My application uses TextView, EditText, Button, ImageView and RecyclerView See activity_main.xml for details of how they are incorporated into the Constraint Layout.

   Here is a screenshot of the layout before the picture has been fetched.

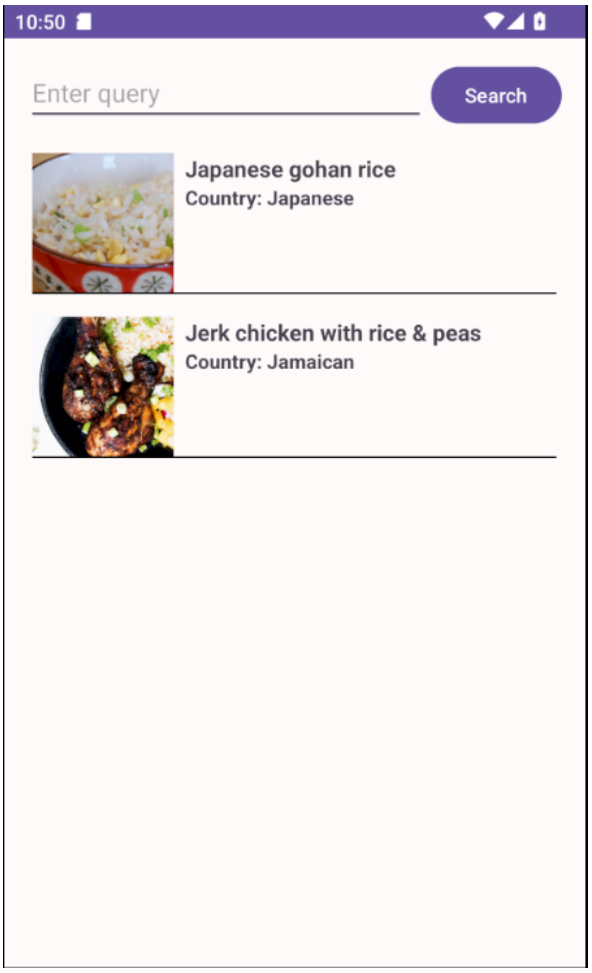


b. It requires user to input

Here is a screenshot of the user searching for a ingredient rice



c. Here is the screenshot after the results has been returned




Where query is the user's search term. The search method makes this request of my web application, parses the returned JSON to find the picture URL, Name, Origin Country. This parsed data is then shown into imageView and TextView in the list format using recyclerView in Android.

Also I have implemented an onClick listener so that when i click on the item in recyclerview it will open a browser link mentioned strSource which redirect user to origin of meal with recipe details on web browser.

**2. Implement a web application**

The URL of my web service deployed to CodeSpace is:

The MealDbApp service is accessible via GitHub Codespaces, leveraging the cloud environment for development and hosting. The dashboard can be found at https://obscure-dollop-r6g744gg9wxhpq9w-8080.app.github.dev/ Under IntelliJ the project app is saved as MealDbApp

**API Implementation:** The DataFetachgServlet is designed to handle HTTP POST requests for retrieving meals from meals endpoint.

**Business Logic:**

Geat Search query from HTTP POST Request and parse it to call mealdb API then send back a formatted JSON

**HTTP POST**

Makes an HTTP POST Request to web service, in MealdbDemo application does an HTTP POST request in MainActivity.java.

The HTTP request is:

**Endpoint**

https://obscure-dollop-r6g744gg9wxhpq9w-8080.app.github.dev

**Request Body**

```
{
    "query":"rice"
}
```

**Response Body**

```
{
  "meals": [
    {
      "idMeal": "53033",
      "strMeal": "Japanese gohan rice",
      "strCategory": "Side",
      "strArea": "Japanese",
```

```
"strMealThumb":
"https://www.themealdb.com/images/media/meals/kw92t41604181871.jpg"
,
"strSource":
"https://www.bbcgoodfood.com/recipes/japanese-rice-gohan"
    },
    {
      "idMeal": "52937",
      "strMeal": "Jerk chicken with rice & peas",
      "strCategory": "Chicken",
      "strArea": "Jamaican",
"strMealThumb":
"https://www.themealdb.com/images/media/meals/tytyxu1515363282.jpg"
,
"strSource":
"https://www.bbcgoodfood.com/recipes/2369635/jerk-chicken-with-rice
-and-peas"
    }
  ]
}
```


**JSON Response:**

Responses are delivered in a JSON format, consistent with the Custom schema, providing details such as meal image, name, origin country and meal recipe url

**Logging to Database:**

Apart from retrieving results from the MealDb API or web service and logging every result into mongoDB we are storing each query made by the user with a counter variable to see how many times users have used the same ingredients to find a meal. Every time a user hits Search query it logs the result into mongodb.

a. If query found in document Increment the counter parameter in document
b. If query not found in document Insert new document with count set as ‘1’

**Operations Analytics and Dashboard:**

A web-based dashboard, built using HTML and HTTP GET, to display analytics that shows top 10 searched ingredients with the number of times the item has been searched. It also presents full logs with the request and response details for each ingredient query.

**Deployment:**

The web service is deployed within a GitHub Codespaces environment, and the dashboard and service are accessed through specific endpoints set during the deployment phase. Configuration files like .devcontainer.json and Dockerfile define the environment setup.

**3. Handle error conditions**

In both the place in our web application and android native application we are using error handling method using try catch exceptions and printing it on console for debug and analysis purpose.

**4. Log useful information**

**MealDBDemo Web Service Logging Information:**

Search query Information:

Logs the search query run by each user for detailed usage analysis.

Logs the counter of number of times the items is been searched in previous users history

**MongoDB Logging:**

All relevant information is persistently stored in a MongoDB Atlas cloud database, which includes the following fields:

- _id: Unique identifier for each log entry.
- queryTerms: Search terms executed by user for ingredients
- counter: The number of time user executed the search terms specified

**5. Store the log information in a database**

**MealDbDemo App Database Logging:**

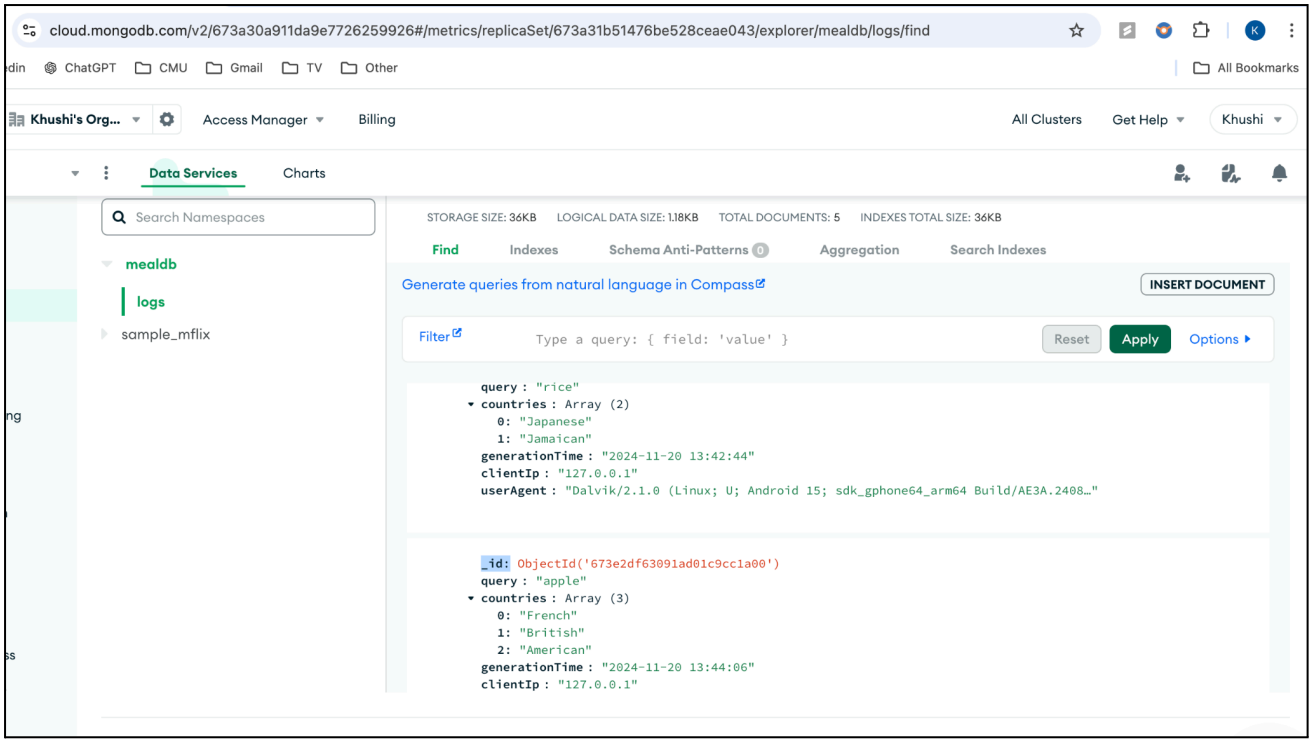
All relevant logs from the StockApp are stored in a MongoDB Atlas cloud database, providing persistent storage and efficient data management. The following is an example of the log entry structure when a request is made for stock data

```
{
  "_id": {
    "$oid": "6739b11d6d7dfa48d5997e1d"
  },
  "query": "rice",
  "count": 9
}
{
  "_id": {
    "$oid": "6739ca496d7dfa48d5998424"
  },
  "query": "chiken",
  "count": 1
}
```


The fields stored in the database log include:

- _id: Unique identifier for each log entry.
- queryTerms: Search terms executed by user for ingredients
- counter: The number of time user executed the search terms specified

This log format provides a comprehensive record of usage of each ingredient searched, allowing for detailed analysis and monitoring of the popularity of ingredients among users.



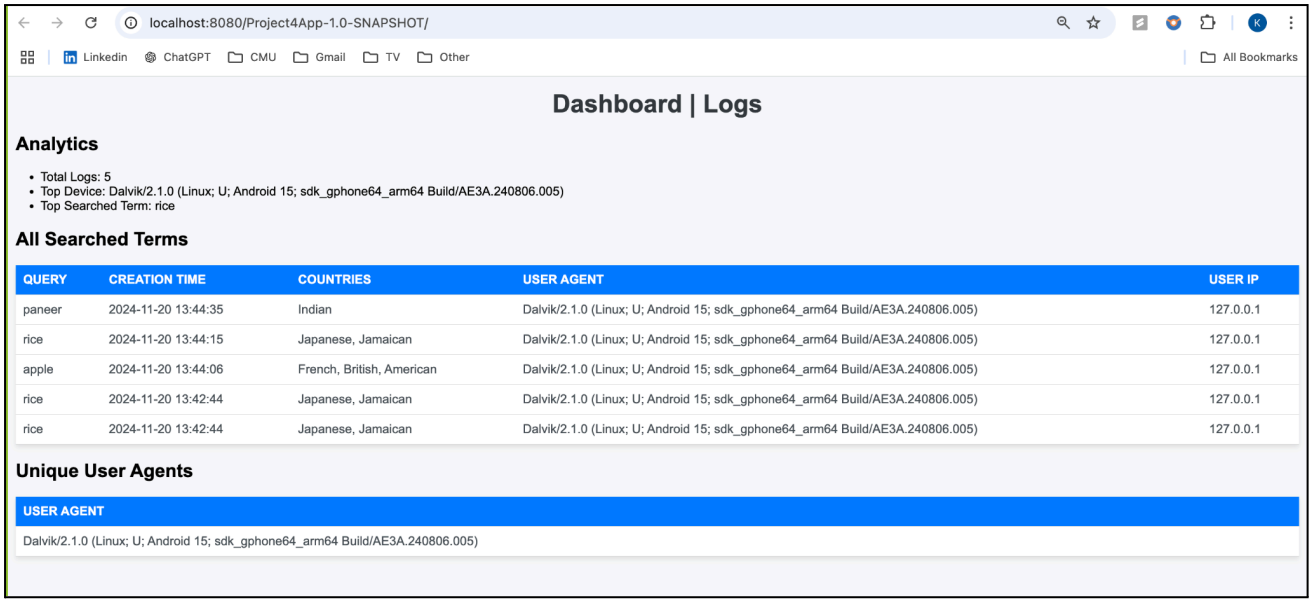

**6. Display operations analytics and full logs on a web-based dashboard**

My MealDBApp dashboard offers a streamlined view of application analytics and user activity.

**Here's a brief overview:**

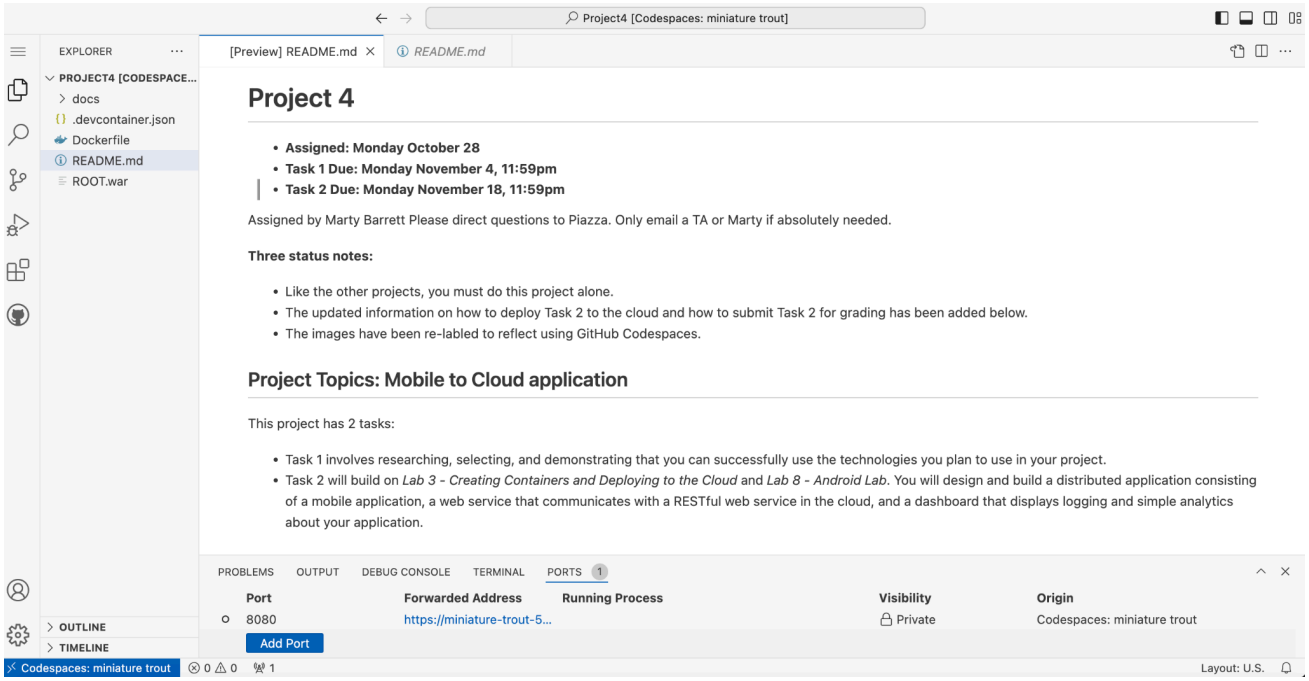
**Analytics Metrics:** It shows the average response time, which sits at 593 ms, highlighting the efficiency of my service.

**Top Ingredients Used:** This section reveals the most common ingredients searched by users with search occurrence number of each ingredient present




## 7. Deploy the web service to GitHub Codespaces

I deployed my meals App web service using GitHub Codespaces, which offers a robust cloud-based development platform. The deployment process was defined by the configuration in the .devcontainer.json and the Dockerfile. To ensure a smooth setup, I adhered to the deployment steps outlined in the CMU Heinz Project 4 repository. This approach has streamlined my development workflow and allowed me to efficiently manage the web service's deployment lifecycle.



**Code Highlights for StockApp:**

Android App: Developed MainActivity.Java for searching ingredients in from MealDb API

WebService: Built FetchData Servlet to manage API requests and craft responses effectively.

Dashboard: Utilized FetchData Servlet dynamically generates the content for the analytics dashboard.