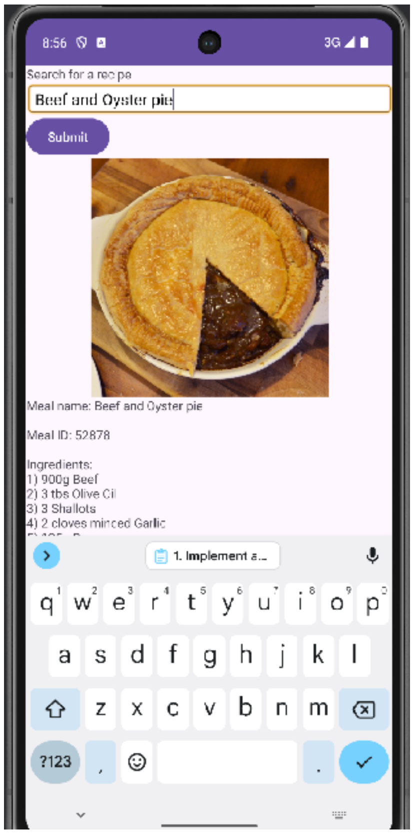
Lawrence Hua
Project 4 Write up

1. Implement a native Android application

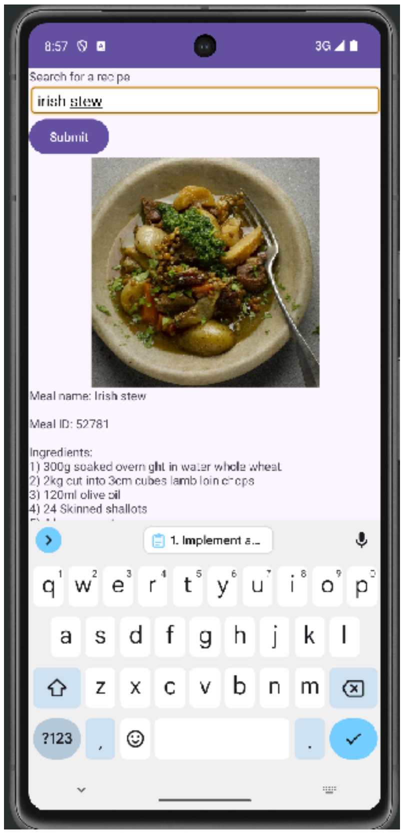


Beef and Oyster pie

Meal name: Beef and Oyster pie

Meal ID: 52878

Ingredients:
1) 900g Beef
2) 3 tbs Olive Cil
3) 3 Shallots
4) 2 cloves minced Garlic



irish stew

Meal name: Irish stew

Meal ID: 52781

Ingredients:
1) 300g soaked overn ght in water whole wheat
2) 2kg cut into 3cm cubes lamb loin chops
3) 120ml olive oil
4) 24 Skinned shallots

In my android app, there is an image view, scrollable textview and an editview for users to add in their input. Its repeatable as well. The images show two searches back to back.

In the code below, I make a request to the webserver in the first method and parse the response in the second method:

```java
private Meal fetchMealInfo(String name) throws IOException, JSONException {
   String deviceModel = Build.MODEL;
   String manufacturer = Build.MANUFACTURER;
   String osVersion = Build.VERSION.RELEASE;

   String url =
"https://friendly-space-chainsaw-vr9qw7r6gq3prgw-8080.app.github.dev/recipe
-servlet?search=" +
           URLEncoder.encode(name, "UTF-8") +
            "&deviceModel=" + URLEncoder.encode(deviceModel, "UTF-8") +
            "&manufacturer=" + URLEncoder.encode(manufacturer, "UTF-8") +
            "&osVersion=" + URLEncoder.encode(osVersion, "UTF-8");

    Request request = new Request.Builder().url(url).build();

    try (Response response = client.newCall(request).execute()) {
        if (!response.isSuccessful()) throw new IOException("Unexpected code
" + response);
        String jsonResponse = response.body().string();
        return parseMealInfo(jsonResponse);
    }
}

private Meal parseMealInfo(String jsonResponse) throws JSONException {
    JSONObject jsonObject = new JSONObject(jsonResponse);
    Meal meal = new Meal();
    JSONArray mealsArray = null;
    // Extracting the meals array from the JSON object
    if (jsonObject.has("meals")) {
        // Get the JSONArray for the "meals" key
         mealsArray = jsonObject.getJSONArray("meals");

    // Checking if the meals array is not empty
    if (mealsArray.length() > 0) {
        // Getting the first meal object from the array
        JSONObject firstMeal = mealsArray.getJSONObject(0);

        // Extracting the idMeal from the first meal object
        String idMeal = firstMeal.optString("idMeal");

        // Extracting other meal details
        String strMeal = firstMeal.optString("strMeal");
        String strCategory = firstMeal.optString("strCategory");
        String strArea = firstMeal.optString("strArea");
        String strInstructions = firstMeal.optString("strInstructions");
        String strMealThumb = firstMeal.optString("strMealThumb");
        String strYoutube = firstMeal.optString("strYoutube");

        System.out.println(strMealThumb);
        // Extracting ingredients
        List<String> ingredients = new ArrayList<>();
        for (int i = 1; i <= 20; i++) {
            String ingredient = firstMeal.optString("strIngredient" + i);
            if (!ingredient.isEmpty()&&!ingredient.equals("null")) {
                ingredients.add(ingredient);
            }
        }

        // Extracting measures
        List<String> measures = new ArrayList<>();
        for (int i = 1; i <= 20; i++) {
            String measure = firstMeal.optString("strMeasure" + i);
            if (!measure.isEmpty() && !measure.equals("null")) {
                measures.add(measure);
            }
        }
        // Setting values to the Meal object
        meal.setIdMeal(idMeal);
        meal.setStrMeal(strMeal);
        meal.setStrCategory(strCategory);
        meal.setStrArea(strArea);
        meal.setStrInstructions(strInstructions);
        meal.setStrMealThumb(strMealThumb);
        meal.setStrYoutube(strYoutube);
        meal.setIngredients(ingredients);
        meal.setMeasures(measures);
    }
        return meal;
    } else {
        // Handle the case where there's no value for the "meals" key
        System.out.println("No meals found in the JSON object");
        return null;
    }

}
```

2. Implement a webservice:
In this doGet method, we get the request from android. Around the first if statement, we parse the response and get the search term.

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException {
   response.setContentType("text/html");
```

```java
    String searchTerm = request.getParameter("search");
    String deviceModel = request.getParameter("deviceModel");
    String manufacturer = request.getParameter("manufacturer");
    String osVersion = request.getParameter("osVersion");

    if(MealDBAPI.isAlpha(searchTerm)) {
        Instant start = Instant.now(); // Start time

        String mealJSON = MealDBAPI.searchMealByName(searchTerm);

        Instant end = Instant.now(); // End time
        Meal meal = MealDBAPI.parseMealFromJson(mealJSON);
        Duration duration = Duration.between(start, end);
        meal.setDuration(duration);

        List<String> ingredients = meal.getIngredients();
        String searchIngredients = null;
        if (ingredients != null) {
            // Filter out null and "null" ingredients
            List<String> nonNullIngredients = ingredients.stream()
                    .filter(ingredient -> ingredient != null &&
!ingredient.equals("null"))
                    .collect(Collectors.toList());

            // Join the non-null ingredients into a comma-separated string
            searchIngredients = String.join(", ", nonNullIngredients);
        }
        String searchRecipe = meal.getStrInstructions();

        // Display the meal details in the servlet response
        // Retrieve the search term from the request
        // Get current date and time
        LocalDateTime now = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        String dateTime = now.format(formatter);

        // Get system information
        String systemInfo = System.getProperty("os.name") + " " +
                System.getProperty("os.version") + ", " +
                System.getProperty("os.arch");

        // Create a MongoDB connection and insert search term, date, and system
information
        try {
            ConnectToMongo.insertData(searchTerm, searchIngredients, searchRecipe,
 dateTime, systemInfo, deviceModel, manufacturer, osVersion, duration);
            // Display success message
            PrintWriter out = response.getWriter();
            out.println(mealJSON);
        } catch (Exception e) {
            e.printStackTrace();
            // Display error message
            PrintWriter out = response.getWriter();
            out.println("<html><body>");
            out.println("<h1>Error occurred: " + e.getMessage() + "</h1>");
            out.println("</body></html>");
            throw new IOException(e);
        }
    } else {
        System.out.println("No meals found");
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("message", "No meal found");
        PrintWriter out = response.getWriter();
        out.println(jsonObject);
    }

 }
```

Within the first if statement, we use the searchMealByName method, which essentially connects to the API and returns the response from it:

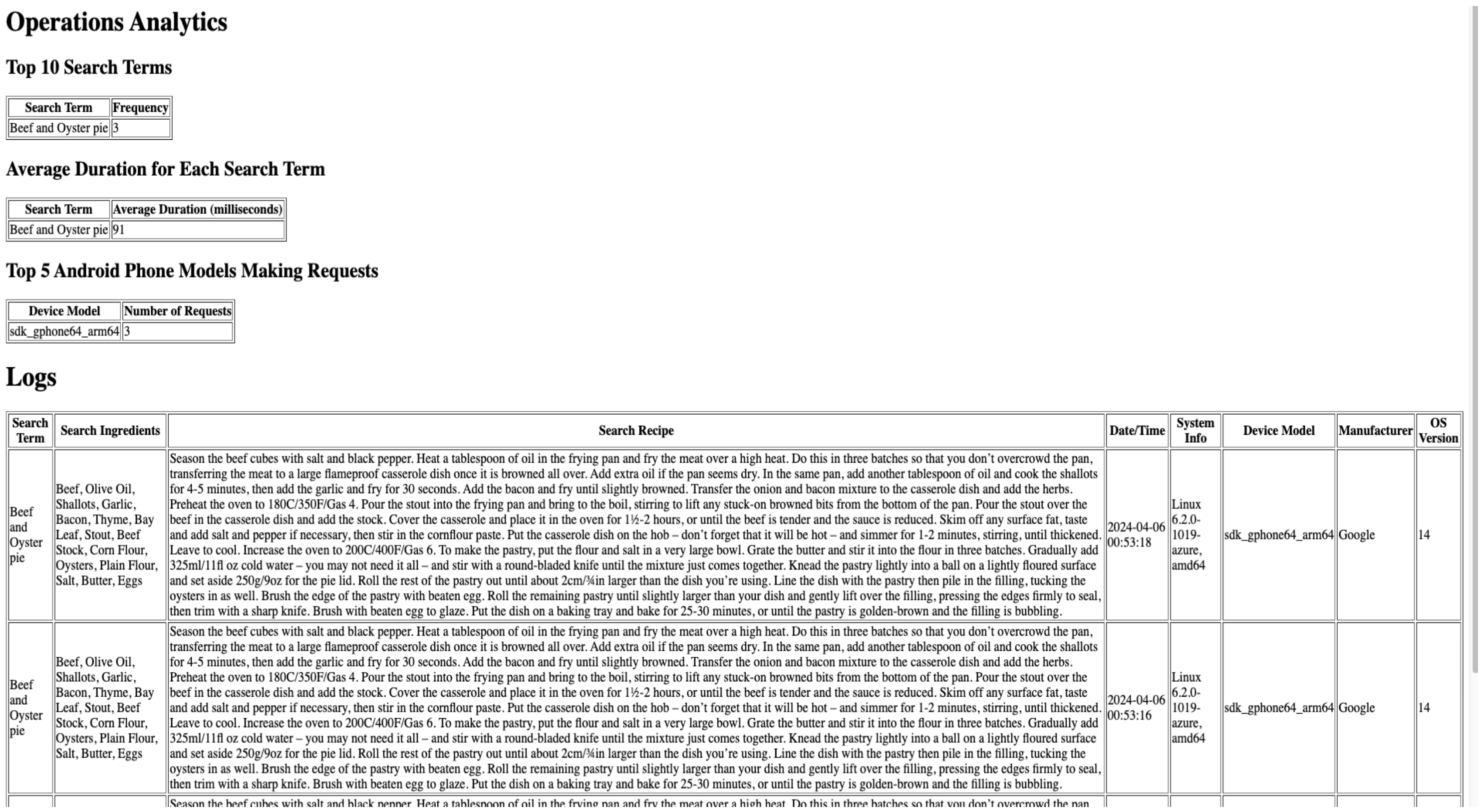
```
public static String searchMealByName(String mealName) throws IOException {
   Instant start = Instant.now(); // Start time

   try {
       URL url = new
URL("https://www.themealdb.com/api/json/v1/1/search.php?s=" + mealName);
       HttpURLConnection conn = (HttpURLConnection) url.openConnection();

       BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
       String inputLine;
       StringBuilder response = new StringBuilder();

       while ((inputLine = in.readLine()) != null) {
           response.append(inputLine);
       }
```

```
        // Close the buffered reader
        in.close();
        // Check if the response contains no meals
        if (response.toString().contains("\"meals\":null")) {
            return "No meals found";
        }
        // Return the JSON response
        return response.toString();
    } catch (IOException e) {
        e.printStackTrace();
        return "No meals found";
    }
}
```

The doGet method mentioned previously then adds the meal into the db and responds with the response json within the try/catch at the end.

3. Directions say it is not needed. I just wanted add a small note for the graders:
I did handle all of them. Within android, “no meals found” will show up if there is no meals. Within the webserver, if there is invalid input for either the api or to mongo, itll handle it gracefully with try/catch methods. Itll also output a response like “no meals found” or send out IOExceptions.

4 and 6. Log useful information + Display dashboard



Operations Analytics

Top 10 Search Terms

Search Term	Frequency
Beef and Oyster pie	3

Average Duration for Each Search Term

Search Term	Average Duration (milliseconds)
Beef and Oyster pie	91

Top 5 Android Phone Models Making Requests

Device Model	Number of Requests
sdk_gphone64_arm64	3

Logs

Search Term	Search Ingredients	Search Recipe	Date/Time	System Info	Device Model	Manufacturer	OS Version
Beef and Oyster pie	Beef, Olive Oil, Shallots, Garlic, Bacon, Thyme, Bay Leaf, Stout, Beef Stock, Corn Flour, Oysters, Plain Flour, Salt, Butter, Eggs	Season the beef cubes with salt and black pepper. Heat a tablespoon of oil in the frying pan and fry the meat over a high heat. Do this in three batches so that you don't overcrowd the pan, transferring the meat to a large flameproof casserole dish once it is browned all over. Add extra oil if the pan seems dry. In the same pan, add another tablespoon of oil and cook the shallots for 4-5 minutes, then add the garlic and fry for 30 seconds. Add the bacon and fry until slightly browned. Transfer the onion and bacon mixture to the casserole dish and add the herbs. Preheat the oven to 180C/350F/Gas 4. Pour the stout into the frying pan and bring to the boil, stirring to lift any stuck-on browned bits from the bottom of the pan. Pour the stout over the beef in the casserole dish and add the stock. Cover the casserole and place it in the oven for 1½-2 hours, or until the beef is tender and the sauce is reduced. Skim off any surface fat, taste and add salt and pepper if necessary, then stir in the cornflour paste. Put the casserole dish on the hob – don't forget that it will be hot – and simmer for 1-2 minutes, stirring, until thickened. Leave to cool. Increase the oven to 200C/400F/Gas 6. To make the pastry, put the flour and salt in a very large bowl. Grate the butter and stir it into the flour in three batches. Gradually add 325ml/11fl oz cold water – you may not need it all – and stir with a round-bladed knife until the mixture just comes together. Knead the pastry lightly into a ball on a lightly floured surface and set aside 250g/9oz for the pie lid. Roll the rest of the pastry out until about 2cm/¾in larger than the dish you're using. Line the dish with the pastry then pile in the filling, tucking the oysters in as well. Brush the edge of the pastry with beaten egg. Roll the remaining pastry until slightly larger than your dish and gently lift over the filling, pressing the edges firmly to seal, then trim with a sharp knife. Brush with beaten egg to glaze. Put the dish on a baking tray and bake for 25-30 minutes, or until the pastry is golden-brown and the filling is bubbling.	2024-04-06 00:53:18	Linux 6.2.0-1019-azure, amd64	sdk_gphone64_arm64	Google	14
Beef and Oyster pie	Beef, Olive Oil, Shallots, Garlic, Bacon, Thyme, Bay Leaf, Stout, Beef Stock, Corn Flour, Oysters, Plain Flour, Salt, Butter, Eggs	Season the beef cubes with salt and black pepper. Heat a tablespoon of oil in the frying pan and fry the meat over a high heat. Do this in three batches so that you don't overcrowd the pan, transferring the meat to a large flameproof casserole dish once it is browned all over. Add extra oil if the pan seems dry. In the same pan, add another tablespoon of oil and cook the shallots for 4-5 minutes, then add the garlic and fry for 30 seconds. Add the bacon and fry until slightly browned. Transfer the onion and bacon mixture to the casserole dish and add the herbs. Preheat the oven to 180C/350F/Gas 4. Pour the stout into the frying pan and bring to the boil, stirring to lift any stuck-on browned bits from the bottom of the pan. Pour the stout over the beef in the casserole dish and add the stock. Cover the casserole and place it in the oven for 1½-2 hours, or until the beef is tender and the sauce is reduced. Skim off any surface fat, taste and add salt and pepper if necessary, then stir in the cornflour paste. Put the casserole dish on the hob – don't forget that it will be hot – and simmer for 1-2 minutes, stirring, until thickened. Leave to cool. Increase the oven to 200C/400F/Gas 6. To make the pastry, put the flour and salt in a very large bowl. Grate the butter and stir it into the flour in three batches. Gradually add 325ml/11fl oz cold water – you may not need it all – and stir with a round-bladed knife until the mixture just comes together. Knead the pastry lightly into a ball on a lightly floured surface and set aside 250g/9oz for the pie lid. Roll the rest of the pastry out until about 2cm/¾in larger than the dish you're using. Line the dish with the pastry then pile in the filling, tucking the oysters in as well. Brush the edge of the pastry with beaten egg. Roll the remaining pastry until slightly larger than your dish and gently lift over the filling, pressing the edges firmly to seal, then trim with a sharp knife. Brush with beaten egg to glaze. Put the dish on a baking tray and bake for 25-30 minutes, or until the pastry is golden-brown and the filling is bubbling.	2024-04-06 00:53:16	Linux 6.2.0-1019-azure, amd64	sdk_gphone64_arm64	Google	14
		Season the beef cubes with salt and black pepper. Heat a tablespoon of oil in the frying pan and fry the meat over a high heat. Do this in three batches so that you don't overcrowd the pan					

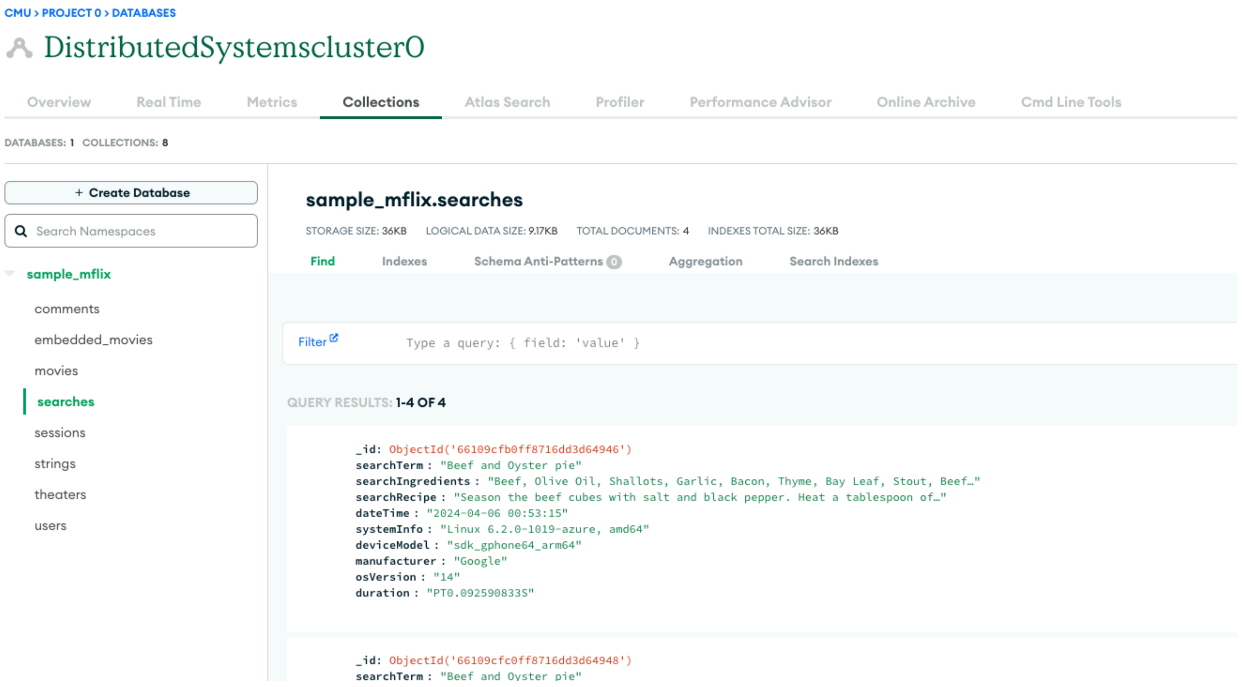
Within my dashboard, there is 3 different analytics/statistics. One for frequency, one for latency, and one for the top 5 phone models. There also is 8 different parameters for the log portion. The recipe may seem like just a bunch of text, but it is the instructions of how to make the search term.

I log all the information within the "RecipeServlet" class and the ConnectToMongo class. Then I display all the information with the DashboardServlet class.

By the way, my webserver automatically goes to the dashboard. You may also add in "dashboard" at the end of the initial url to get to the same page. After talking to a TA, they mentioned either way is fine? Thanks!

5. Store the information in a database.

Within my webserver, I created a class "ConnectToMongo", which has the method "insertData". This method basically connects to my DB and specific collection, then creates a document with the required information and stores it all. Below is a screenshot of my database in MongoDB.



```
_id: ObjectId('66109cfb0ff8716dd3d64946')
searchTerm : "Beef and Oyster pie"
searchIngredients : "Beef, Olive Oil, Shallots, Garlic, Bacon, Thyme, Bay Leaf, Stout, Beef…"
searchRecipe : "Season the beef cubes with salt and black pepper. Heat a tablespoon of…"
dateTime : "2024-04-06 00:53:15"
systemInfo : "Linux 6.2.0-1019-azure, amd64"
deviceModel : "sdk_gphone64_arm64"
manufacturer : "Google"
osVersion : "14"
duration : "PT0.092590833S"

_id: ObjectId('66109cfc0ff8716dd3d64948')
searchTerm : "Beef and Oyster pie"
```

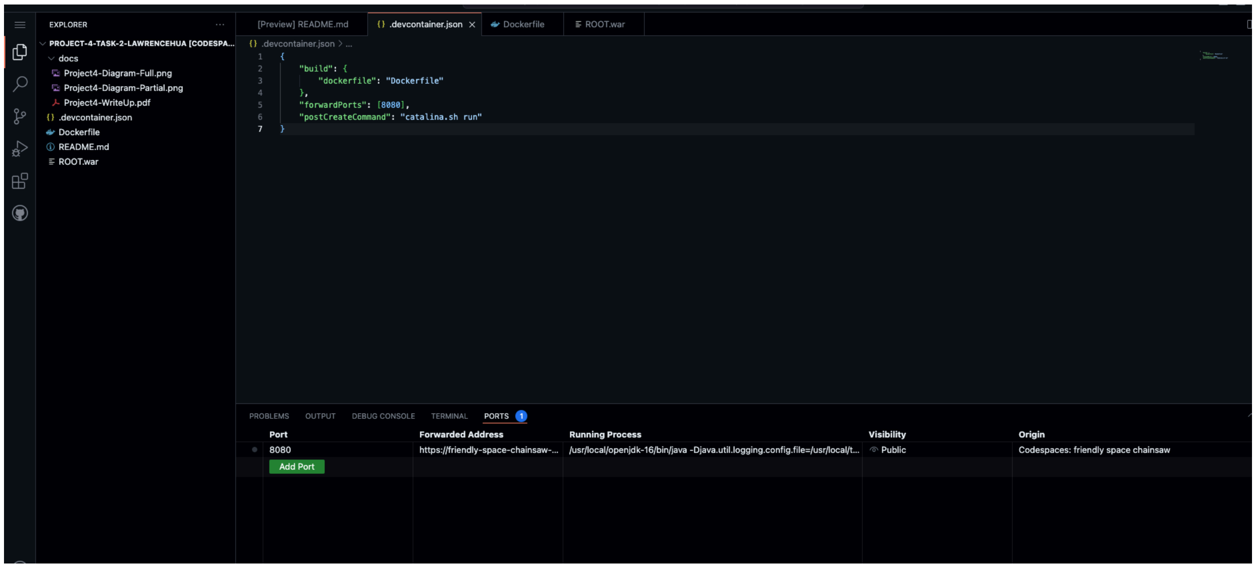
And below this is the insertData code:

```java
public static void insertData(String searchTerm, String searchIngredients, String
searchRecipe, String dateTime, String systemInfo, String deviceModel, String
manufacturer, String osVersion, Duration duration) {
   String connectionString =
"mongodb+srv://lhua:Lwh0410Lwh0410@distributedsystemsclust.yecuro5.mongodb.net/?ret
ryWrites=true&w=majority&appName=DistributedSystemscluster0";

   ServerApi serverApi = ServerApi.builder()
           .version(ServerApiVersion.V1)
           .build();

   MongoClientSettings settings = MongoClientSettings.builder()
           .applyConnectionString(new ConnectionString(connectionString))
           .serverApi(serverApi)
           .build();

   // Create a new client and connect to the server
   try (MongoClient mongoClient = MongoClients.create(settings)) {
       // Send a ping to confirm a successful connection
       MongoDatabase database = mongoClient.getDatabase("sample_mflix"); // Change
"sample_mflix" to your database name
       database.runCommand(new Document("ping", 1));
       System.out.println("Pinged your deployment. You successfully connected to
MongoDB!");
```

```
        // Write the search term, date, and system information as part of a document
to the MongoDB database
        MongoCollection<Document> collection = database.getCollection("searches");
// Change "searches" to your collection name
        Document document = new Document("searchTerm", searchTerm)
                .append("searchIngredients", searchIngredients)
                .append("searchRecipe", searchRecipe)
                .append("dateTime", dateTime)
                .append("systemInfo", systemInfo)
                .append("deviceModel", deviceModel)
                .append("manufacturer", manufacturer)
                .append("osVersion", osVersion)
                .append("duration", duration.toString());
        System.out.println(duration.toString());
        collection.insertOne(document);
        System.out.println("Data inserted into MongoDB database.");
    } catch (MongoException e) {
        e.printStackTrace();
        System.out.println("fail to add into DB");
    }
}
```

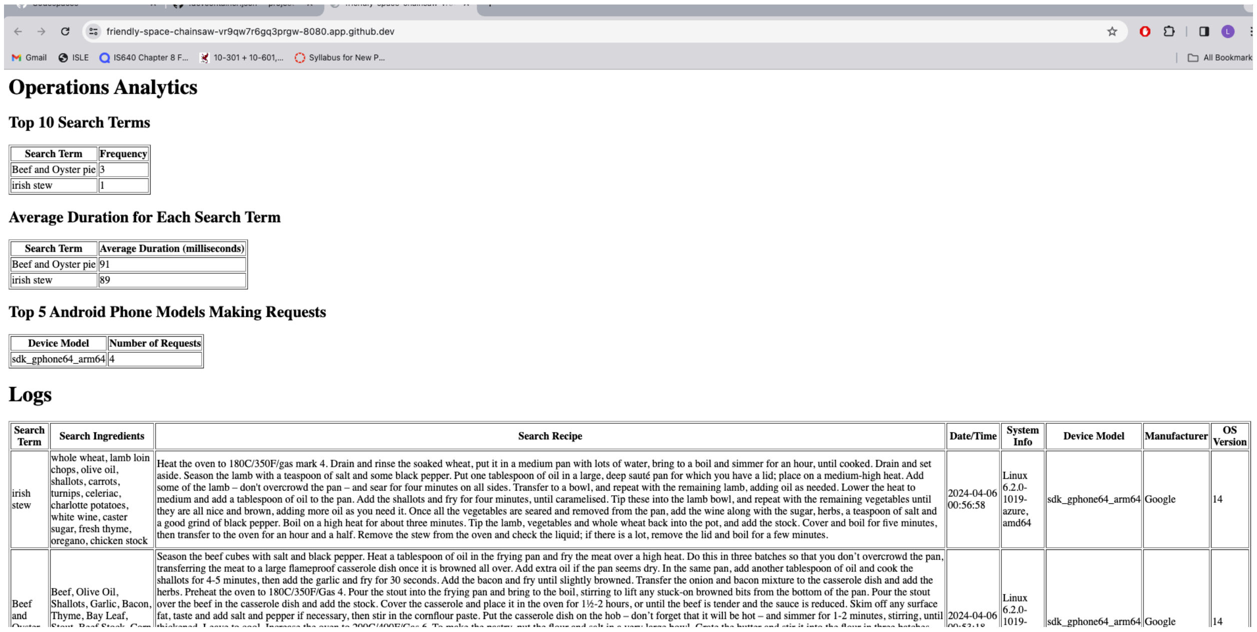
7. Deploy the web service to GitHub Codespaces

I was able to do this.
After adding in my ROOT.war file and making the port public, I was able to open the dashboard in a browser:



```
{
    "build": {
        "dockerfile": "Dockerfile"
    },
    "forwardPorts": [8080],
    "postCreateCommand": "catalina.sh run"
}
```

Port	Forwarded Address	Running Process	Visibility	Origin
8080	https://friendly-space-chainsaw-...	/usr/local/openjdk-16/bin/java -Djava.util.logging.config.file=/usr/local/t...	Public	Codespaces: friendly space chainsaw



Operations Analytics

Top 10 Search Terms

Search Term	Frequency
Beef and Oyster pie	3
irish stew	1

Average Duration for Each Search Term

Search Term	Average Duration (milliseconds)
Beef and Oyster pie	91
irish stew	89

Top 5 Android Phone Models Making Requests

Device Model	Number of Requests
sdk_gphone64_arm64	4

Logs

Search Term	Search Ingredients	Search Recipe	Date/Time	System Info	Device Model	Manufacturer	OS Version
irish stew	whole wheat, lamb loin chops, olive oil, shallots, carrots, turnips, celeriac, charlotte potatoes, white wine, caster sugar, fresh thyme, oregano, chicken stock	Heat the oven to 180C/350F/gas mark 4. Drain and rinse the soaked wheat, put it in a medium pan with lots of water, bring to a boil and simmer for an hour, until cooked. Drain and set aside. Season the lamb with a teaspoon of salt and some black pepper. Put one tablespoon of oil in a large, deep sauté pan for which you have a lid; place on a medium-high heat. Add some of the lamb – don't overcrowd the pan – and sear for four minutes on all sides. Transfer to a bowl, and repeat with the remaining lamb, adding oil as needed. Lower the heat to medium and add a tablespoon of oil to the pan. Add the shallots and fry for four minutes, until caramelised. Tip these into the lamb bowl, and repeat with the remaining vegetables until they are all nice and brown, adding more oil as you need it. Once all the vegetables are seared and removed from the pan, add the wine along with the sugar, herbs, a teaspoon of salt and a good grind of black pepper. Boil on a high heat for about three minutes. Tip the lamb, vegetables and whole wheat back into the pot, and add the stock. Cover and boil for five minutes, then transfer to the oven for an hour and a half. Remove the stew from the oven and check the liquid; if there is a lot, remove the lid and boil for a few minutes.	2024-04-06 00:56:58	Linux 6.2.0-1019-azure, amd64	sdk_gphone64_arm64	Google	14
Beef and	Beef, Olive Oil, Shallots, Garlic, Bacon, Thyme, Bay Leaf,	Season the beef cubes with salt and black pepper. Heat a tablespoon of oil in the frying pan and fry the meat over a high heat. Do this in three batches so that you don't overcrowd the pan, transferring the meat to a large flameproof casserole dish once it is browned all over. Add extra oil if the pan seems dry. In the same pan, add another tablespoon of oil and cook the shallots for 4-5 minutes, then add the garlic and fry for 30 seconds. Add the bacon and fry until slightly browned. Transfer the onion and bacon mixture to the casserole dish and add the herbs. Preheat the oven to 180C/350F/Gas 4. Pour the stout into the frying pan and bring to the boil, stirring to lift any stuck-on browned bits from the bottom of the pan. Pour the stout over the beef in the casserole dish and add the stock. Cover the casserole and place it in the oven for 1½-2 hours, or until the beef is tender and the sauce is reduced. Skim off any surface fat, taste and add salt and pepper if necessary, then stir in the cornflour paste. Put the casserole dish on the hob – don't forget that it will be hot – and simmer for 1-2 minutes, stirring, until	2024-04-06	Linux 6.2.0-1019-	sdk_gphone64_arm64	Google	14

In the first screenshot, I show the port and the url for my codespaces workspace. Then, in the second screenshot I display the webpage that the browser opens up to. If you add in "/dashboard" at the end, it should bring you to the same page too, but isn't necessary.
Also, in my android app that I turned in on github, the URL is made for this specific codespaces. If the TA's create a new one for grading, please replace the url within the Android app's Main activity's "fetchMealInfo" method.

Use of chatGPT and stackoverflow throughout all the code in the webserver and android app.

I also want to mention, this is limited to 100 items. More details here:
https://www.themealdb.com/api.php
^^ this is also the API doc.