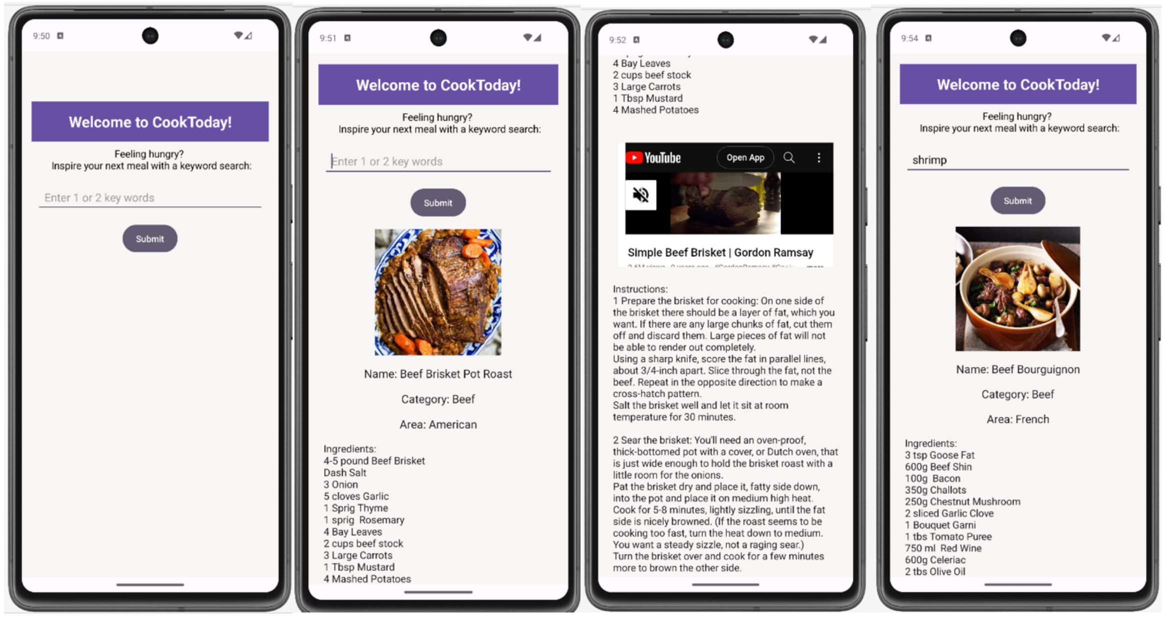
Name: Zhenqi Wang
Andrew ID: zhenqiw

Distributed Application Requirements

1. Implement a native Android application



Feeling hungry?
Inspire your next meal with a keyword search:

Enter 1 or 2 key words

Name: Beef Brisket Pot Roast

Category: Beef

Area: American

Ingredients:
4-5 pound Beef Brisket
Dash Salt
3 Onion
5 cloves Garlic
1 Sprig Thyme
1 sprig Rosemary
4 Bay Leaves
2 cups beef stock
3 Large Carrots
1 Tbsp Mustard
4 Mashed Potatoes

Instructions:
1 Prepare the brisket for cooking: On one side of the brisket there should be a layer of fat, which you want. If there are any large chunks of fat, cut them off and discard them. Large pieces of fat will not be able to render out completely.
Using a sharp knife, score the fat in parallel lines, about 3/4-inch apart. Slice through the fat, not the beef. Repeat in the opposite direction to make a cross-hatch pattern.
Salt the brisket well and let it sit at room temperature for 30 minutes.

2 Sear the brisket: You'll need an oven-proof, thick-bottomed pot with a cover, or Dutch oven, that is just wide enough to hold the brisket roast with a little room for the onions.
Pat the brisket dry and place it, fatty side down, into the pot and place it on medium high heat.
Cook for 5-8 minutes, lightly sizzling, until the fat side is nicely browned. (If the roast seems to be cooking too fast, turn the heat down to medium. You want a steady sizzle, not a raging sear.)
Turn the brisket over and cook for a few minutes more to brown the other side.

Name: Beef Bourguignon

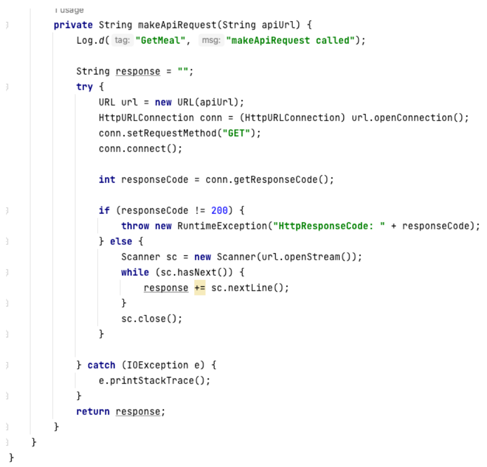
Category: Beef

Area: French

Ingredients:
3 tsp Goose Fat
600g Beef Shin
100g Bacon
350g Challots
250g Chestnut Mushroom
2 sliced Garlic Clove
1 Bouquet Garni
1 tbs Tomato Puree
750 ml Red Wine
600g Celeriac
2 tbs Olive Oil

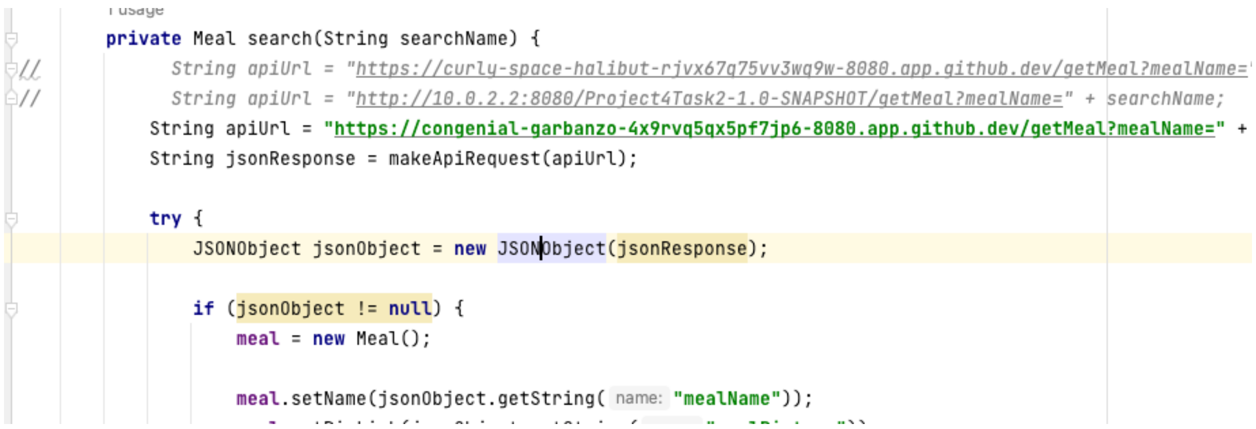
This is my android application. It allows the user to input keywords of a meal [b], and the program would return ImageView, TextView, WebView to the user [e]. There is a EditText in this program as well [a]. The user can reinput new key words [f].

The HTTP request is used. Here is a sample code [c]:



```
    1 usage
    private String makeApiRequest(String apiUrl) {
        Log.d( tag: "GetMeal",  msg: "makeApiRequest called");

        String response = "";
        try {
            URL url = new URL(apiUrl);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.connect();

            int responseCode = conn.getResponseCode();

            if (responseCode != 200) {
                throw new RuntimeException("HttpResponseCode: " + responseCode);
            } else {
                Scanner sc = new Scanner(url.openStream());
                while (sc.hasNext()) {
                    response += sc.nextLine();
                }
                sc.close();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
        return response;
    }
  }
}
```

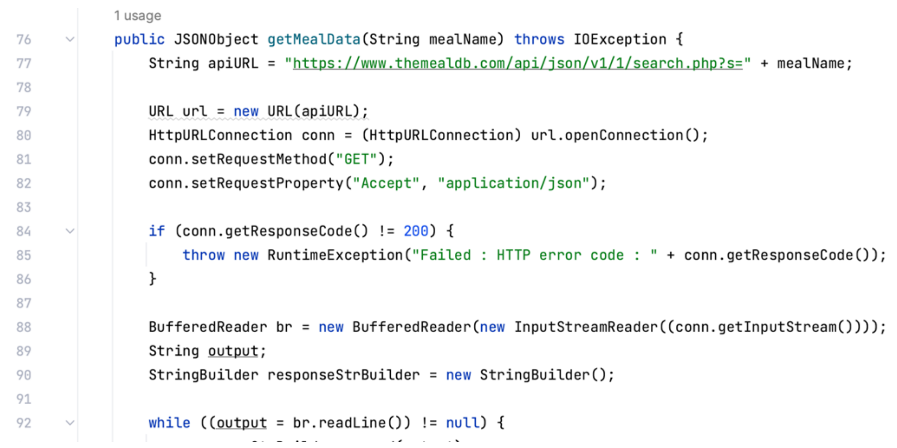
This is how it receives and parses JSON reply [d]:



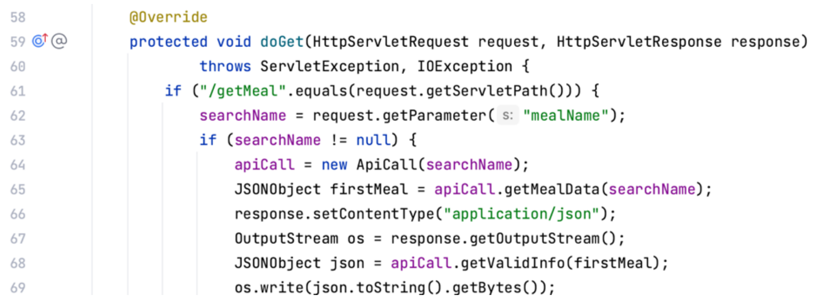
```java
1 usage
private Meal search(String searchName) {
//      String apiUrl = "https://curly-space-halibut-rjvx67q75vv3wq9w-8080.app.github.dev/getMeal?mealName="
//      String apiUrl = "http://10.0.2.2:8080/Project4Task2-1.0-SNAPSHOT/getMeal?mealName=" + searchName;
    String apiUrl = "https://congenial-garbanzo-4x9rvq5qx5pf7jp6-8080.app.github.dev/getMeal?mealName=" +
    String jsonResponse = makeApiRequest(apiUrl);

    try {
        JSONObject jsonObject = new JSONObject(jsonResponse);

        if (jsonObject != null) {
            meal = new Meal();

            meal.setName(jsonObject.getString( name: "mealName"));
```

2. Implement a web service

As the two screenshots of code shown above, it is using an API I implemented with servlet, and deployed on codespace. These are two screenshots of how I use "themealdb" API, and use servlet to have one API myself to receive http request from application. The detailed code is in package Project4Task2->ApiCall class and MealServlet class [a,b,c].



```java
1 usage
public JSONObject getMealData(String mealName) throws IOException {
    String apiURL = "https://www.themealdb.com/api/json/v1/1/search.php?s=" + mealName;

    URL url = new URL(apiURL);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Accept", "application/json");

    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " + conn.getResponseCode());
    }

    BufferedReader br = new BufferedReader(new InputStreamReader((conn.getInputStream())));
    String output;
    StringBuilder responseStrBuilder = new StringBuilder();

    while ((output = br.readLine()) != null) {
```



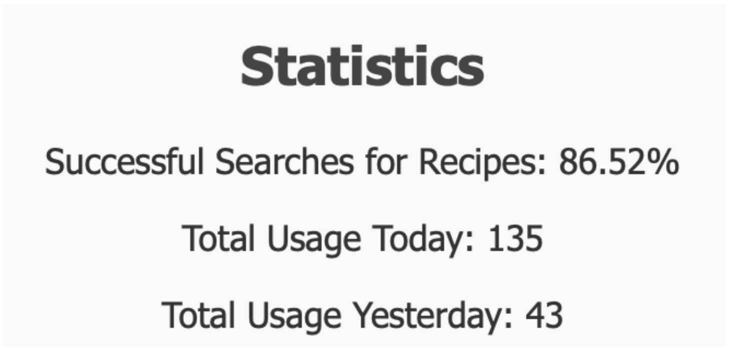
```java
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    if ("/getMeal".equals(request.getServletPath())) {
        searchName = request.getParameter( s: "mealName");
        if (searchName != null) {
            apiCall = new ApiCall(searchName);
            JSONObject firstMeal = apiCall.getMealData(searchName);
            response.setContentType("application/json");
            OutputStream os = response.getOutputStream();
            JSONObject json = apiCall.getValidInfo(firstMeal);
            os.write(json.toString().getBytes());
```

3. Handle error conditions

Can be evaluated while running the code.

4. Log useful information

Timestamp: To track when users are most active and to find usage patterns over time.
User Input: To understand what users are searching for.
Possible Results: To gauge the effectiveness of the third-party API in providing relevant results for user.
Meal Area: To determine geographical or regional preferences in meal choices.
Meal Category: To identify popular meal categories among users.
Number of Ingredients: To assess the complexity of recipes that users are interested in.
Model: To understand the range of devices the app is being used on.



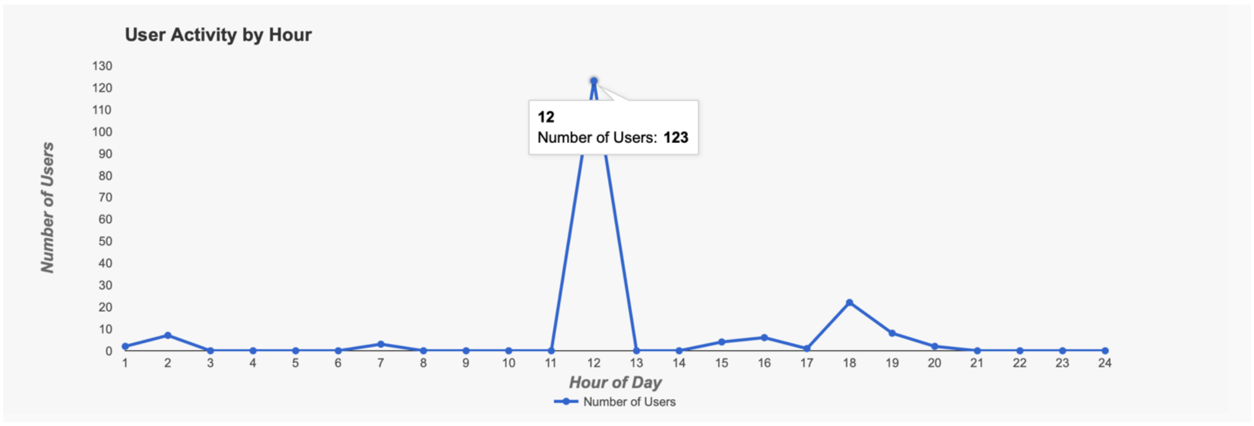
Statistics

Successful Searches for Recipes: 86.52%

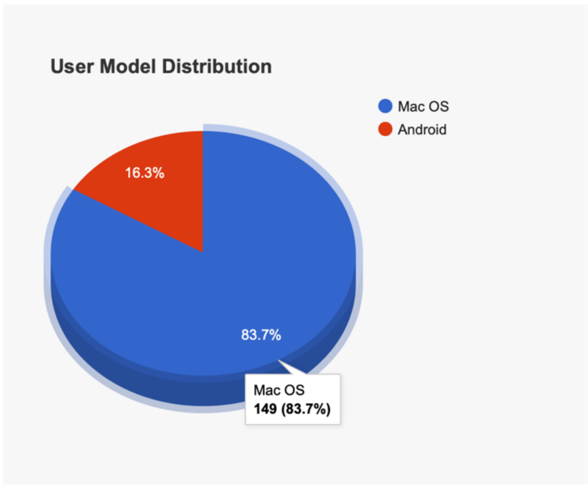
Total Usage Today: 135

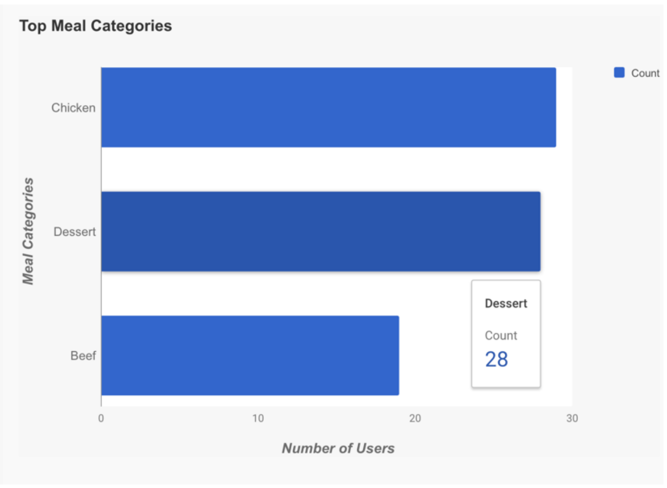
Total Usage Yesterday: 43

The successful searches are used to find out how much chances the users can get a recipe. The third party API is not that perfect, there are some uncommon meals the API cannot find. It can test my program's usefulness.



This plot is used to track the users' activity hour during the day. As an application is considering commercialize transformation, it needs to know the traffic of itself, and better assign advertisements at specific period during the day.





The left plot is a simplified measurement of user model distribution, same as above, the owner of this application can assign different advertisements to different model users. For example, if a user uses mac, we can post an advertisement of "game" released in app store especially for her.

The right plot is an analysis of meal categories users may retrieve. It is an interesting figure of which type of food people usually don't know their recipe. At the same time, we can also use this data to attract some companies to advertise with us. For example, if "Dessert" is a top category, we may reach to some bakery or café, ask them to post ads on our platform, since users are usually confused about how to make dessert, they may eat out.

5. Store the log information in a database

```
connectionString = "mongodb://Bettie:bettie2000@ac-sy56oav-shard-00-
00.gvamiky.mongodb.net:27017,ac-sy56oav-shard-00-
01.gvamiky.mongodb.net:27017,ac-sy56oav-shard-00-
02.gvamiky.mongodb.net:27017/myFirstDatabase?w=majority&retryWrites=true&tls=
true&authMechanism=SCRAM-SHA-1"
```

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

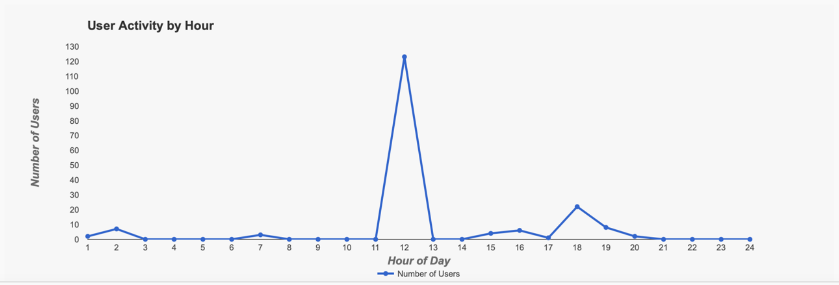
Welcome to Dashboard

Statistics

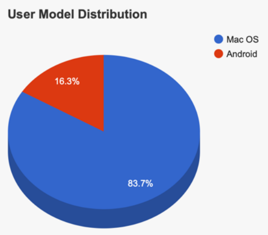
Successful Searches for Recipes: 86.52%

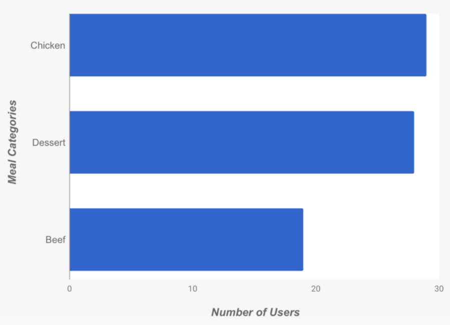
Total Usage Today: 135

Total Usage Yesterday: 43



Top Meal Categories





Database Records

#	Timestamp	User Input	Possible Results	Meal Area	Meal Category	Number of Ingredients	Model
1	2023-11-16 15:00:14	pizza	1	Italian	Miscellaneous	11	Intel Mac OS X 10_15_7
161	2023-11-17 12:50:08	orange	1	Tunisian	Dessert	7	Intel Mac OS X 10_15_7
162	2023-11-17 12:50:12	lamb	11	Greek	Lamb	7	Intel Mac OS X 10_15_7
163	2023-11-17 12:50:19	lemon	1	Greek	Lamb	7	Intel Mac OS X 10_15_7
164	2023-11-17 12:50:23	lemon	1	Greek	Lamb	7	Intel Mac OS X 10_15_7
165	2023-11-17 12:50:26	lemon	1	Greek	Lamb	7	Intel Mac OS X 10_15_7
166	2023-11-17 12:50:33	tuna	2	Tunisian	Seafood	11	Intel Mac OS X 10_15_7
167	2023-11-17 12:50:36	burgur	0	Empty	Empty	0	Intel Mac OS X 10_15_7
168	2023-11-17 12:50:41	beef	20	British	Beef	19	Intel Mac OS X 10_15_7
169	2023-11-17 12:50:44	beef	20	Filipino	Beef	14	Intel Mac OS X 10_15_7
170	2023-11-17 12:51:10	beef	20	British	Beef	10	Intel Mac OS X 10_15_7
171	2023-11-17 12:51:16	chicken	25	Indian	Chicken	9	Intel Mac OS X 10_15_7
172	2023-11-17 12:51:20	chicken	25	British	Chicken	17	Intel Mac OS X 10_15_7
173	2023-11-17 12:51:23	chicken	25	Japanese	Chicken	16	Intel Mac OS X 10_15_7
174	2023-11-17 12:51:26	chicken	25	Portuguese	Chicken	17	Intel Mac OS X 10_15_7
175	2023-11-17 12:51:30	chicken	25	Jamaican	Chicken	13	Intel Mac OS X 10_15_7
176	2023-11-17 12:51:39	chicken	25	Indian	Chicken	9	Intel Mac OS X 10_15_7
177	2023-11-17 12:51:45	chicken	25	Japanese	Chicken	16	Intel Mac OS X 10_15_7
178	2023-11-17 12:51:55	chicken	25	Greek	Chicken	11	Intel Mac OS X 10_15_7

-----------------The End-------------------