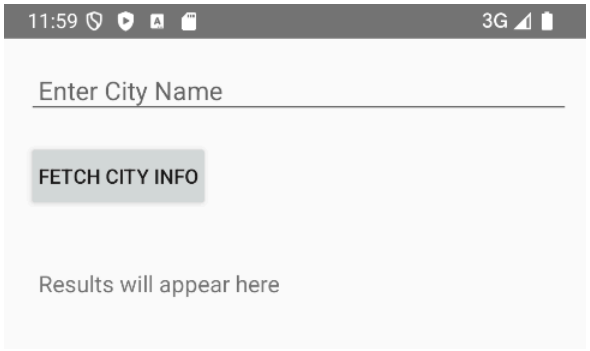
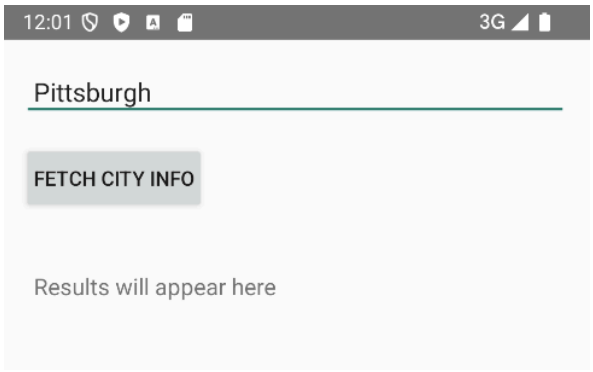
Name: Allison Miao
Email: yuhanmia@andrew.cmu.edu

Project 4

I used a different API for this project.
1. Implement a native Android application
The name of my native Android application project in Android Studio is: Project4Android
a. My application uses TextView, EditText, Button. See avicity_main.xml for details of how they are incorporated into the LinearLayout.
Here is a screenshot of the layout before the information is fetched.



b. Requires input from the user
Here is a screenshot of the user searching for city info.

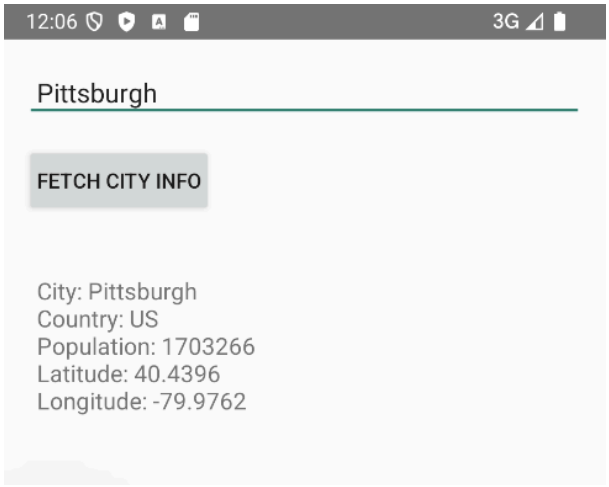


c. Makes an HTTP request (using an appropriate HTTP method) to your web service
My application does an HTTP GET request. The request is:
“http://100.110.161.66:8080/Project4Task2_war_exploded/cityinfo?city=” + city
The search method makes this request of my web application, parses the returned XML to find the URL.
d. Receives and parses an XML or JSON formatted reply from your web service

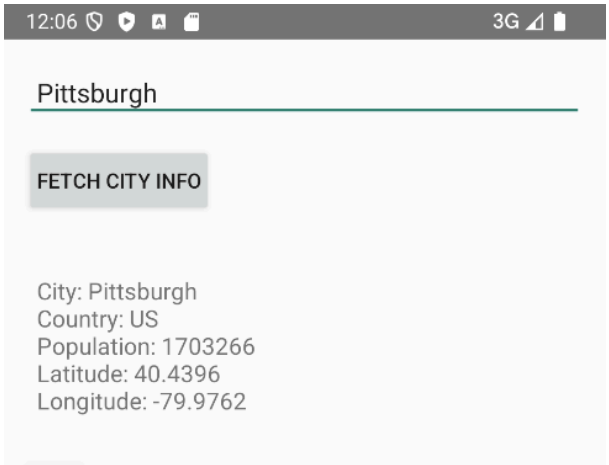
```
<cityinfo>
    <city>
        <name>Hangzhou</name>
        <country>China</country>
```

```
        <latitude>30.2741</latitude>
        <longitude>120.1551</longitude>
        <population>10000000</population>
        <timezone>Asia/Shanghai</timezone>
    </city>
</cityinfo>
```

e. Displays new information to the user



f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)



2. Implement a web service

a. Implement a simple (can be a single path) API.

```
<servlet>
    <servlet-name>MainServlet</servlet-name>
    <servlet-class>ds.project4task2.MainServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MainServlet</servlet-name>
    <url-pattern>/cityinfo</url-pattern>
</servlet-mapping>
```

b. Receives an HTTP request from the native Android application

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException {
    String city = req.getParameter("city");

    // Handle invalid input
    if (city == null || city.trim().isEmpty()) {
        resp.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        resp.setContentType("application/json");
        try {
            resp.getWriter().write("{\"error\": \"Invalid city name.\"}");
        } catch (Exception e) {
            e.printStackTrace();
        }
        return;
    }

    // Process valid requests (business logic and response generation below)
}
```

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

```
// Build the API URL
String apiUrl = String.format(API_URL_TEMPLATE, city);

// Fetch data from the third-party API
long startTime = System.currentTimeMillis();
URL url = new URL(apiUrl);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setRequestMethod("GET");
connection.setRequestProperty("X-Api-Key", API_KEY);

int responseCode = connection.getResponseCode();
long endTime = System.currentTimeMillis();

// Log API response time
Document log = new Document();
log.append("city", city)
   .append("api_request_timestamp", System.currentTimeMillis())
   .append("api_response_time", endTime - startTime);

// Handle the API response
if (responseCode == HttpURLConnection.HTTP_OK) {
    BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
    StringBuilder response = new StringBuilder();
    String line;
    while ((line = in.readLine()) != null) {
```

```
        response.append(line);
    }
    in.close();

    log.append("mobile_response_data", response.toString());
    logCollection.insertOne(log);
} else {
    log.append("mobile_response_error", "Failed to fetch data from the API");
    logCollection.insertOne(log);
}
```

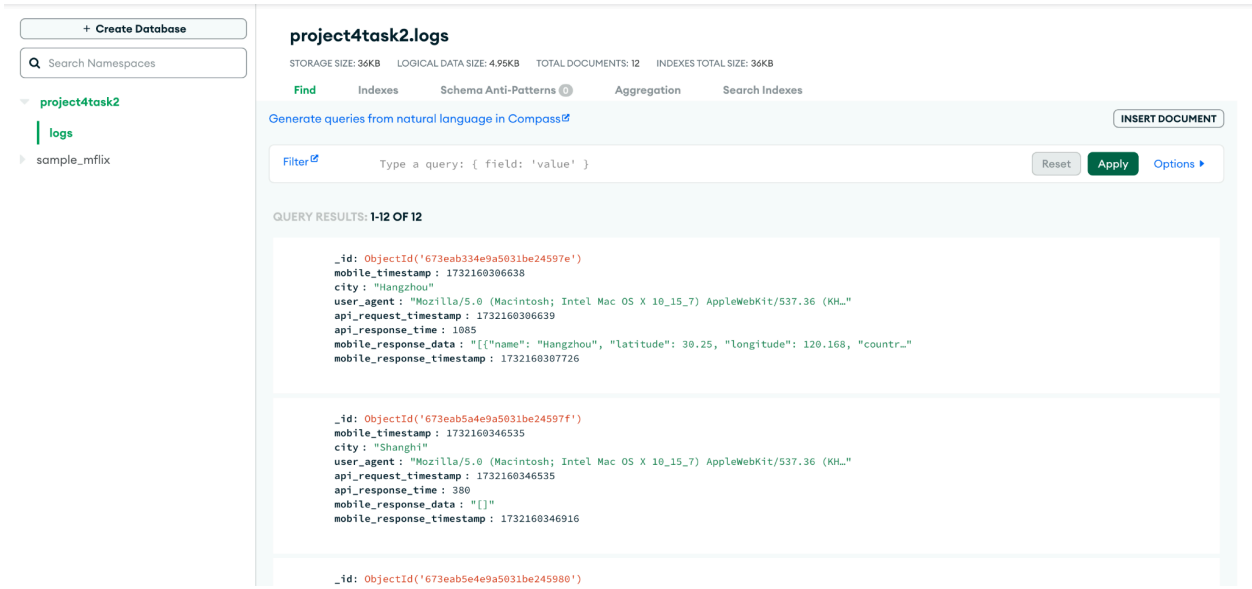
d. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.

```
String xmlResponse =
    "<cityinfo>" +
        "<city>" +
            "<name>" + cityName + "</name>" +
            "<country>" + country + "</country>" +
            "<latitude>" + latitude + "</latitude>" +
            "<longitude>" + longitude + "</longitude>" +
            "<population>" + population + "</population>" +
            "<timezone>" + timezone + "</timezone>" +
        "</city>" +
    "</cityinfo>";

resp.setContentType("application/xml");
resp.getWriter().write(xmlResponse);
```

4. Log useful information

At least 6 pieces of information is logged for each request/reply with the mobile phone. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. (You should NOT log data from interactions from the operations dashboard.)



This is a screenshot of the MongoDB database collection. More than 6 pieces of information are in each log. I do NOT log data from interactions from the operations dashboard.

5. Store the log information in a database
The web service can connect, store, and retrieve information from a MongoDB database in the cloud.
Refer to the screenshot above in part 4. Also, refer to the screenshot below in part 6. The web service can retrieve information from a MongoDB database, so I have this dashboard below.

6. Display operations analytics and full logs on a web-based dashboard
a. A unique URL addresses a web interface dashboard for the web service.
http://localhost:8080/Project4Task2_war_exploded/dashboard
b. The dashboard displays at least 3 interesting operations analytics.
c. The dashboard displays formatted full logs.

Web Service Dashboard

Operations Analytics

Top Cities Requested: {Beijing=1, Shanghi=1, Shanghai=2, Madrid=1, Hangzhou=4, Prague=1, Pittsburgh=2}

Total Requests: 12

Average API Response Time: 451.3333333333333 ms

Logs

Mobile Timestamp	City	User Agent	API Response Time	Response Data
1732160306638	Hangzhou	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	1085 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732160346535	Shanghi	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	380 ms	[]
1732160350418	Shanghai	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	303 ms	[{"name": "Shanghai", "latitude": 31.1667, "longitude": 121.467, "country": "CN", "population": 22120000, "region": "Shanghai", "is_capital": false}]
1732160358281	Madrid	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	342 ms	[{"name": "Madrid", "latitude": 40.4189, "longitude": -3.6919, "country": "ES", "population": 3266126, "region": "Community of Madrid", "is_capital": true}]
1732160363832	Prague	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	332 ms	[{"name": "Prague", "latitude": 50.0833, "longitude": 14.4167, "country": "CZ", "population": 1324277, "region": "Prague", "is_capital": true}]
1732161000097	Pittsburgh	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	734 ms	[{"name": "Pittsburgh", "latitude": 40.4396, "longitude": -79.9762, "country": "US", "population": 1703266, "region": "Pennsylvania", "is_capital": false}]
1732162545003	Hangzhou	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	498 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732163396145	Beijing	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	251 ms	[{"name": "Beijing", "latitude": 39.905, "longitude": 116.391, "country": "CN", "population": 19433000, "region": "Beijing", "is_capital": true}]
1732163584398	Hangzhou	okhttp/4.11.0	467 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732163586372	Hangzhou	okhttp/4.11.0	291 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732165578798	Pittsburgh	okhttp/4.11.0	413 ms	[{"name": "Pittsburgh", "latitude": 40.4396, "longitude": -79.9762, "country": "US", "population": 1703266, "region": "Pennsylvania", "is_capital": false}]
1732165602461	Shanghai	okhttp/4.11.0	320 ms	[{"name": "Shanghai", "latitude": 31.1667, "longitude": 121.467, "country": "CN", "population": 22120000, "region": "Shanghai", "is_capital": false}]

7. Deploy the web service to GitHub Codespaces

In order to deploy your web application to the cloud using GitHub Cloudspaces, first make sure you have the basics working.

a. Accept the GitHub Classroom Assignment that you have been given the URL for. You will find a repository with:

A .devcontainer.json and a Dockerfile which define how to create a Docker container, build a suitable software stack, and deploy the ROOT.war web application.

A ROOT.war file which, like in Lab 3, contains a web application that will deployed in the container. This is a simple "Hello World!" application.

An identical copy of this README.md

b. Click the green <> Code dropdown button, select the Codespaces tab, then click on "Create codespace on master".

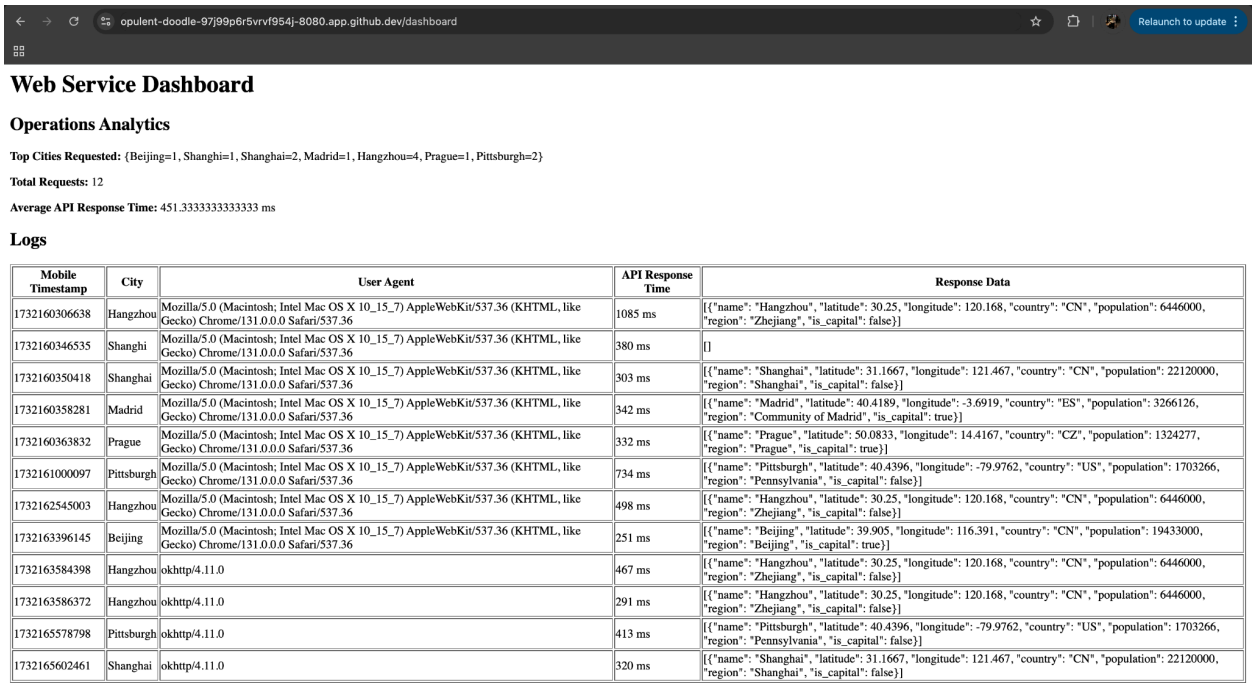
c. Once the Codespace is running, the Terminal tab will show that Catalina (the Servlet container) is running. You should also see a "1" next to the Ports tab. Click on the Ports tab and you should see that port 8080 has been made available.

d. Mouse over the Local address item of the port 8080 line and you will find three icons. The leftmost is to copy the URL of your deployed application, the middle one (a globe) is to launch that URL in a browser. Clicking on the globe is a quick way to test your web service in a browswer. The copy is useful to use the URL in your Android App.

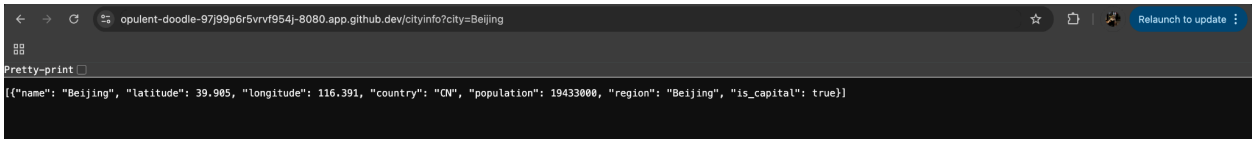
e. Click on the globe to confirm that the Hello World servlet is working.

f. By default, the URL in (d) requires you to be authenticated with Github. To test in a browser, that is fine, but when accessing your web service from your Android app, the Android app will not be authenticated. Therefore you must make the port visibility "Public". To do this, right or

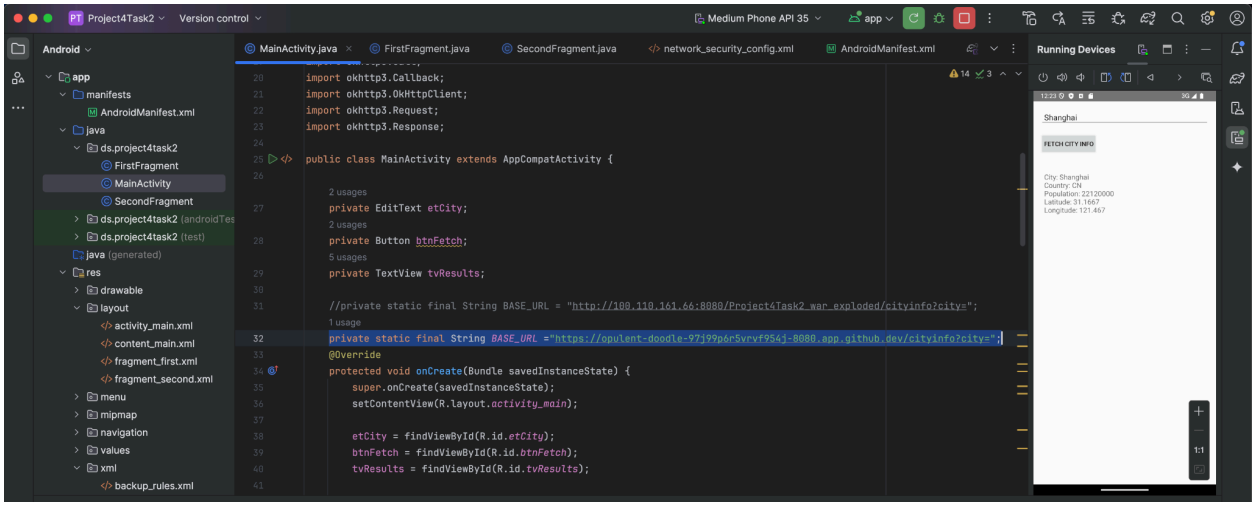
control click on the word "Private" in the Visibility column, and change Port Visibility to "Public". You will now be able to access the web service from your Android App or from an unauthenticated browser.
g. Copy the URL and paste into an Incognito Chrome window to confirm that the Hello World web app can be reached without authentication.
h. To deploy your own web service, create a ROOT.war like you did in Lab 3, upload or push the ROOT.war to your repository, and create a Codespace as has just been described.



Mobile Timestamp	City	User Agent	API Response Time	Response Data
1732160306638	Hangzhou	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	1085 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732160346535	Shanghi	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	380 ms	[]
1732160350418	Shanghai	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	303 ms	[{"name": "Shanghai", "latitude": 31.1667, "longitude": 121.467, "country": "CN", "population": 22120000, "region": "Shanghai", "is_capital": false}]
1732160358281	Madrid	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	342 ms	[{"name": "Madrid", "latitude": 40.4189, "longitude": -3.6919, "country": "ES", "population": 3266126, "region": "Community of Madrid", "is_capital": true}]
1732160363832	Prague	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	332 ms	[{"name": "Prague", "latitude": 50.0833, "longitude": 14.4167, "country": "CZ", "population": 1324277, "region": "Prague", "is_capital": true}]
1732161000097	Pittsburgh	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	734 ms	[{"name": "Pittsburgh", "latitude": 40.4396, "longitude": -79.9762, "country": "US", "population": 1703266, "region": "Pennsylvania", "is_capital": false}]
1732162545003	Hangzhou	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	498 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732163396145	Beijing	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.36	251 ms	[{"name": "Beijing", "latitude": 39.905, "longitude": 116.391, "country": "CN", "population": 19433000, "region": "Beijing", "is_capital": true}]
1732163584398	Hangzhou	okhttp/4.11.0	467 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732163586372	Hangzhou	okhttp/4.11.0	291 ms	[{"name": "Hangzhou", "latitude": 30.25, "longitude": 120.168, "country": "CN", "population": 6446000, "region": "Zhejiang", "is_capital": false}]
1732165578798	Pittsburgh	okhttp/4.11.0	413 ms	[{"name": "Pittsburgh", "latitude": 40.4396, "longitude": -79.9762, "country": "US", "population": 1703266, "region": "Pennsylvania", "is_capital": false}]
1732165602461	Shanghai	okhttp/4.11.0	320 ms	[{"name": "Shanghai", "latitude": 31.1667, "longitude": 121.467, "country": "CN", "population": 22120000, "region": "Shanghai", "is_capital": false}]



It also works well when I input the codespace URL onto the Android Studio



For Project 4, I referred to OpenAI's ChatGPT as a large language model for guidance and reference.