Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Description:

My application takes a search string from the user, and uses it to fetch and display a news from NewsData.io API key. The information includes:

status: to check if Server found a news or not

title: news title

link: news link

description: description of the news

language: what language did the news use

creator: creator of the news

country: where does news comes from?

1. **Implement a native Android application**

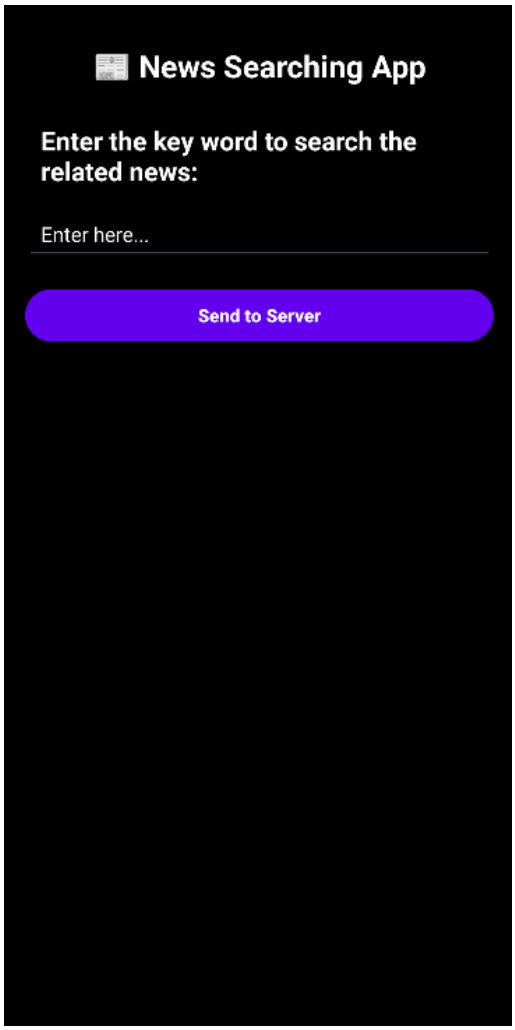
The name of my native Android application project in Android Studio is: Project4Task2 (I named my zip file as Project4Task2AndroidApp.zip)

a. **Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View)**

My application uses TextView, EditText, and Button in my layout. If you want see the details of my views, please check activity_main.xml to see the details.

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Screenshot of the layout before the news has been fetched:

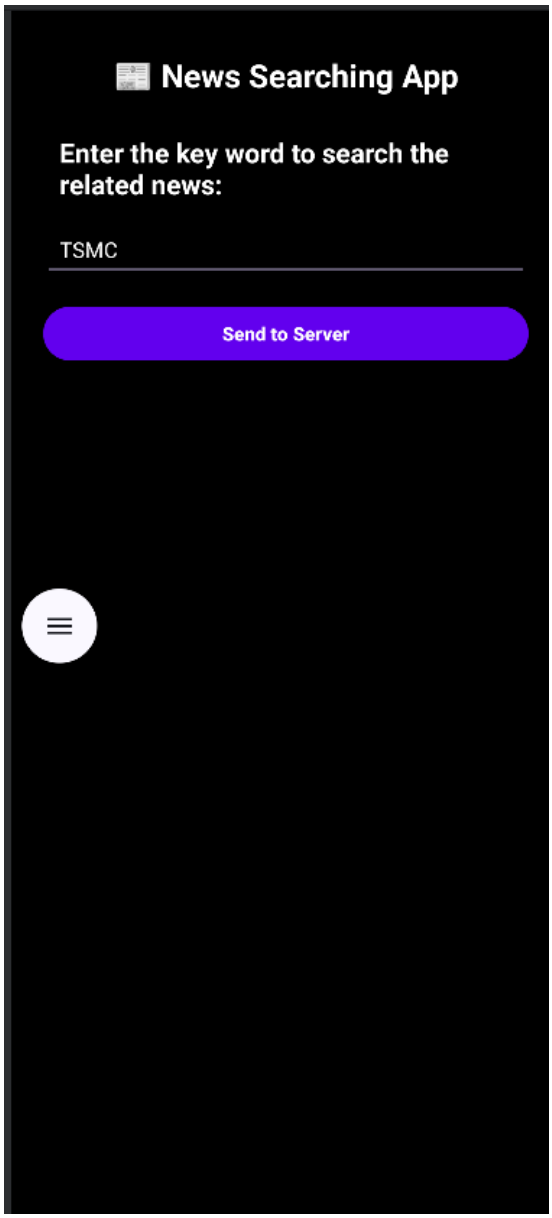


b. Requires input from the user

Required input from user: a string text that used for searching the news

Screenshot of the user searching for a news of TSMC:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



c. Makes an HTTP request (using an appropriate HTTP method)

My app does an HTTP POST request in MainActivity.java. The HTTP request is sent to https://supreme-yodel-p4wpv5rv7g43rjqp-8080.app.github.dev/messages. The request includes a JSON payload that contains the search keyword entered by the user and the device model information.

The `sendMessageToServlet` method builds this request and sends it to my web service. The server parses the request, retrieves related news articles using the NewsData.io API, and responds with a JSON object containing the title, link,

description, publication date, language, and other metadata. The application then parses the JSON response and displays the news information to the user.

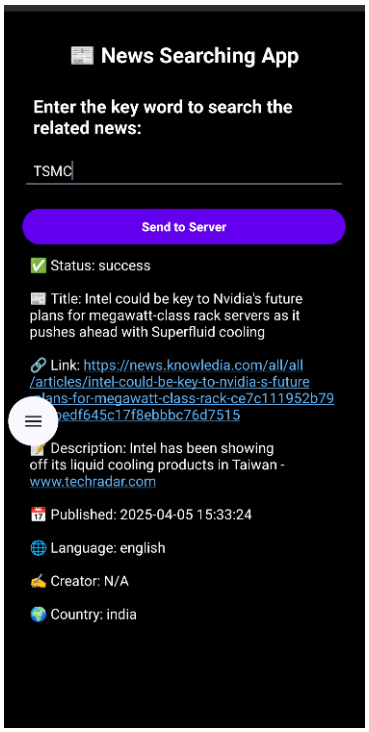
d. Receives and parses an XML or JSON formatted reply from your web service

Example of json reply:


```
{
    "status": "success",
     "tile": "Breaking: Major Event in Global Tech",
    "link": "https://www.example.com/news/global-tech-
   event",
    "description": "A significant event has taken place in
   the global technology industry...",
   "pubDate": "2025-04-05",
   "language": "en",
   "creator": ["Tech Insider", "Jane Doe"],
   "country": ["US", "UK"]
   }
```


e. Displays new information to the user

Screenshot after the news details have been returned:

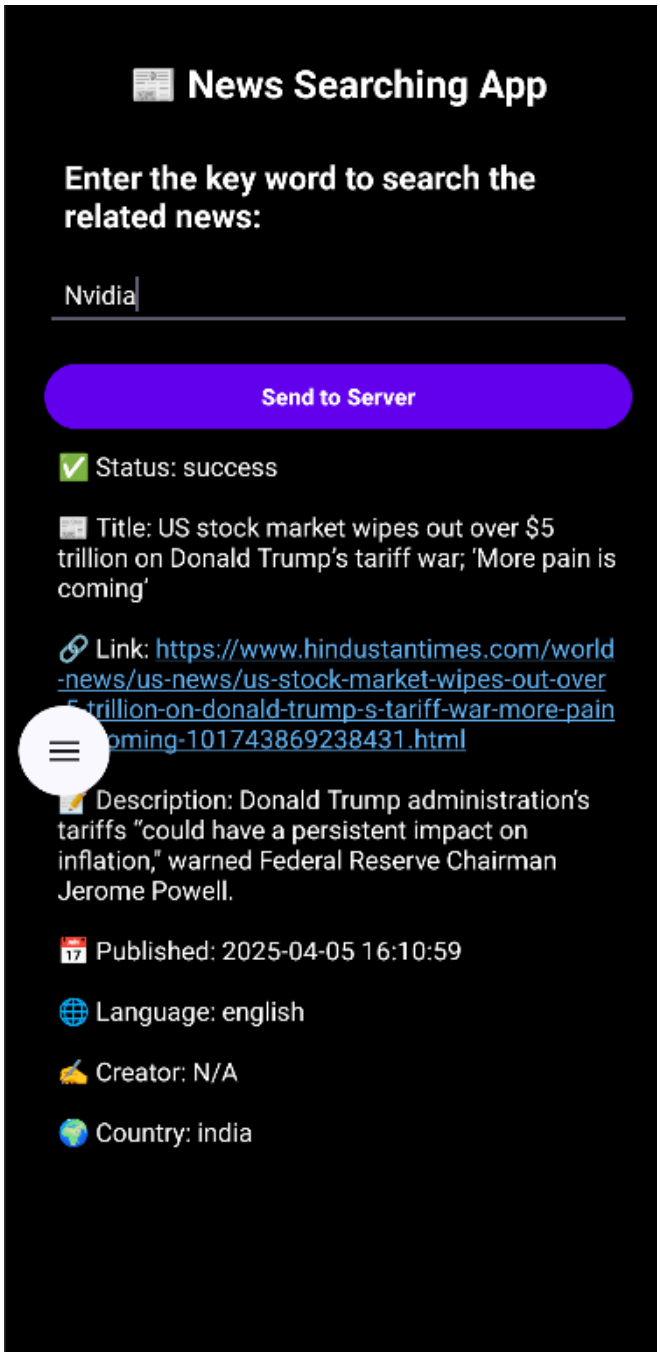


Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

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

The user can type in another search term and hit submit button.

Search the news after finishing searching TSMC:



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

2. implement a web service

The URL of my web service deployed to Dashboard is:

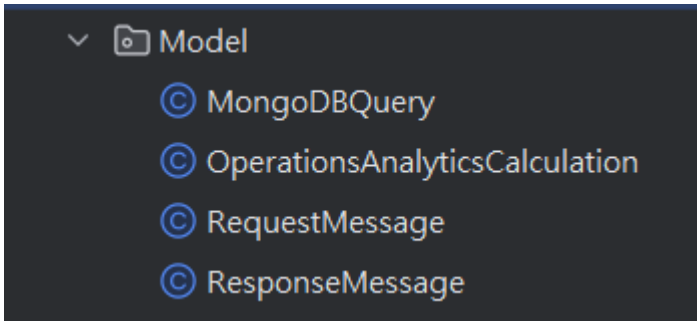
https://supreme-yodel-p4wpv5rv7g43rjqp-8080.app.github.dev/

The project directory name is Project4Task2 (I name the zip file as Project4Task2WebServlet.zip)

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

In my web app project:

Model:



View: **index.jsp**

Controller: **NewsWebAppServlet.java**

b. Receives an HTTP request from the native Android application

My web service, implemented as a Java servlet (NewsAppWebServlet), successfully receives HTTP POST requests from the native Android application. The servlet is mapped to the /messages endpoint using the @WebServlet("/messages") annotation.

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

When the Android app sends a JSON-formatted request containing the user's keyword and device model, the servlet reads this data using a BufferedReader, converts it into a RequestMessage Java object using the Gson library, and processes it to fetch news from the NewsData.io API. This interaction demonstrates the servlet's ability to receive and handle HTTP requests directly from the Android client.

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

My web service executes application-specific business logic by integrating with the NewsData.io third-party API. Upon receiving a POST request from the Android app, the servlet performs the following:

First: Extracts the keyword from the request payload.

Second: Builds a query URL to fetch news data from the NewsData.io API in JSON format.

```
//build up the url for fetching
String apiURL = "https://newsdata.io/api/1/news?apikey=" + apiKey + "&q=" + keyword;
```

Third: Fetches the JSON response using a custom method (You can check the details in my ResponseMessage.java), and generate to JSON format:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```
//Make the API data to JSON format
JsonObject jsonObject = JsonParser.parseString(responseMessage.fetchDataFromAPIKey(apiURL)).getAsJsonObject();
//Get the result object
JsonArray news = jsonObject.getAsJsonArray( memberName: "results");

//Set up the Response message by JsonArray
responseMessage.inputAPIDataToResponse(news);
```

Fourth: send back to client:

```
    // Return API response directly to app
    response.setContentType("application/json");
    //Write the json format response to client
    response.getWriter().write(gson.toJson(responseMessage));
}
```

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

Response will be in JSON format to client. Example will be in 1-d

3. Handle error conditions

e (you do **not** need to document *3. Handle error conditions.*)

4. Log useful information

The information I used for log data is:

clientIP: client's IP

deviceModel: what mobile does user use?

serverReplyStatus: does server search the result of the news?

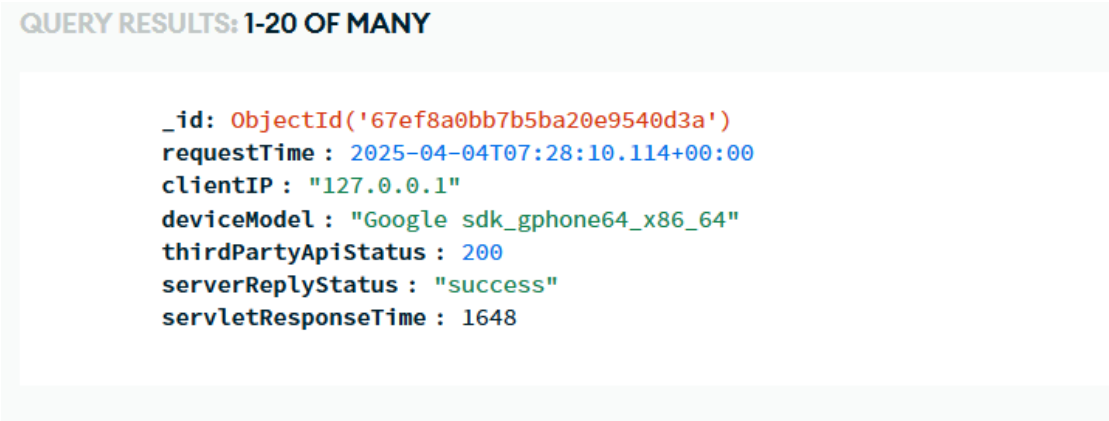
thirdPartyApiResponse: what is the response from third party API?

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

serrvletResponseTime: how long does server reply to client?

requestTime: when does server received client's request?

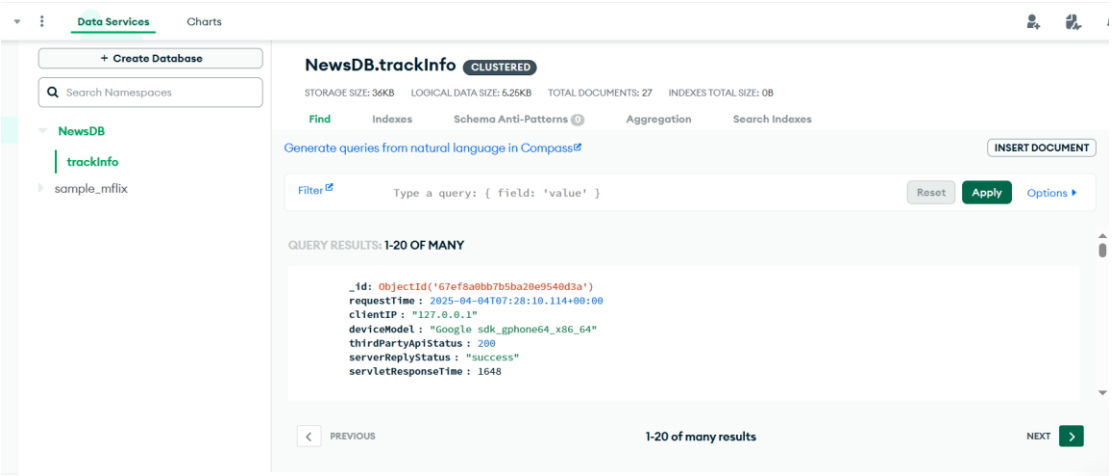
Example data storage:



Please check the MongoDBQuery.java to see the details of the log data

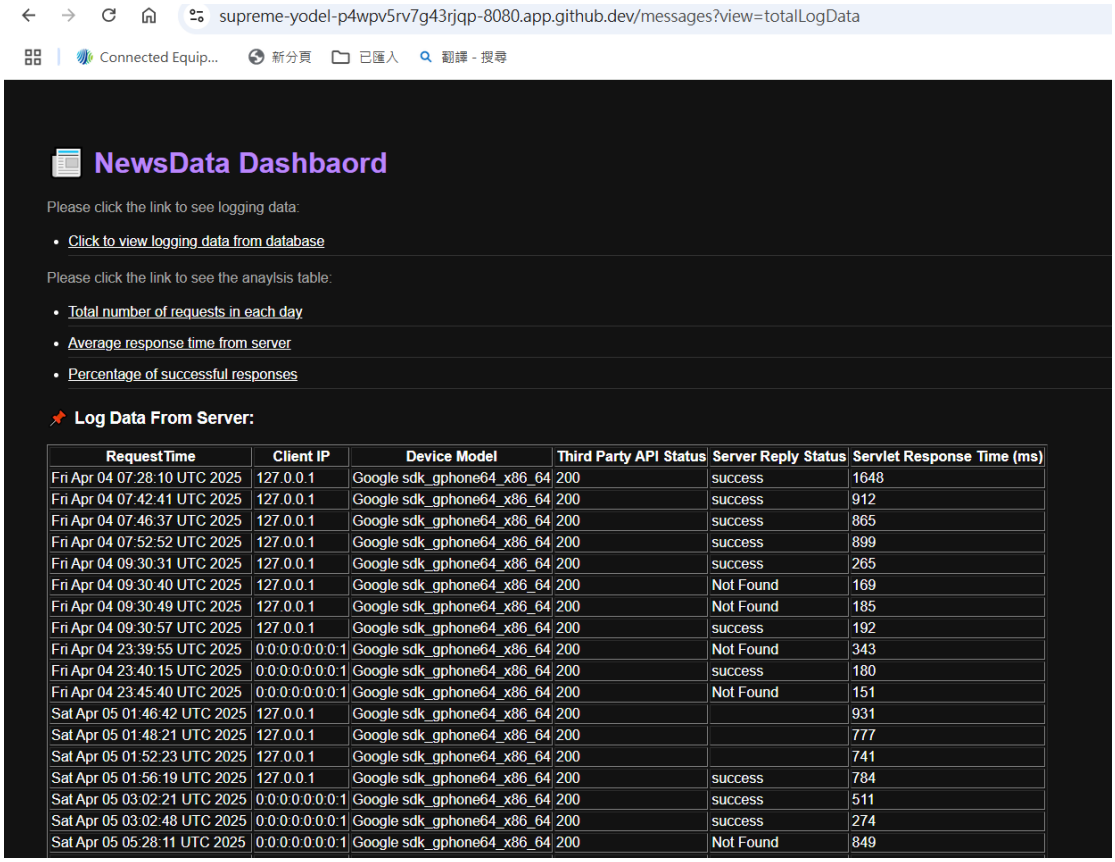
5. Store the log information in a database

My MongoDB:



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

In my web app service dash board, once click the link: *"Click to view logging data from database"*, you can see the log data that extract from the database:



RequestTime	Client IP	Device Model	Third Party API Status	Server Reply Status	Servlet Response Time (ms)
Fri Apr 04 07:28:10 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	1648
Fri Apr 04 07:42:41 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	912
Fri Apr 04 07:46:37 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	865
Fri Apr 04 07:52:52 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	899
Fri Apr 04 09:30:31 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	265
Fri Apr 04 09:30:40 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	Not Found	169
Fri Apr 04 09:30:49 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	Not Found	185
Fri Apr 04 09:30:57 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	192
Fri Apr 04 23:39:55 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	Not Found	343
Fri Apr 04 23:40:15 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	180
Fri Apr 04 23:45:40 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	Not Found	151
Sat Apr 05 01:46:42 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		931
Sat Apr 05 01:48:21 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		777
Sat Apr 05 01:52:23 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		741
Sat Apr 05 01:56:19 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	784
Sat Apr 05 03:02:21 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	511
Sat Apr 05 03:02:48 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	274
Sat Apr 05 05:28:11 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	Not Found	849

Code for connect & store MongoDB:

```
//Name for my mongoDB
String mongoDBURI = "mongodb+srv://jerryh:Fallterm2024@cluster0.jrgpb.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
String dbName = "NewsDB";
String collectionName = "trackInfo";

//Build up a DB query to store
MongoDBQuery storeLogData = new MongoDBQuery(request.getRemoteAddr(), requestMessage.getDeviceModel(), responseMessage.getStatus(), thirdPartyApiStatus, servletRe
//Store the query to target mongoDB
storeLogData.insertToMongoDB(storeLogData.getDBCollection(mongoDBURI, dbName, collectionName));
```

Code for retrieve information from MongoDB:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```java
@Override  no usages
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    //Get the "view" values from HTTP request
    String view = request.getParameter( s: "view");

    //Name for my mongoDB
    String NewsDBURI = "mongodb+srv://jerryh:Fallterm2024@cluster0.jrgpb.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0";
    String dbName = "NewsDB";
    String collectionName = "trackInfo";

    //Get the data from mongoDB
    MongoCollection<Document> collection = MongoDBQuery.getDBCollection(NewsDBURI, dbName, collectionName);
    //Build up to list
    List<Document> logs = new ArrayList<>();
    for (Document doc : collection.find()) {
        logs.add(doc);
    }
```

Please check the MongoDBQuery.java to see the details of the database modifying function.

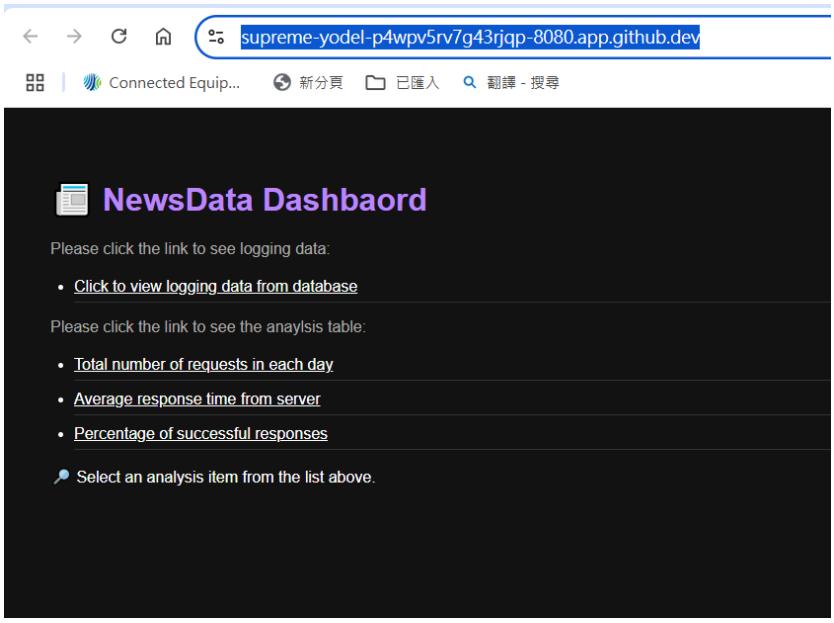
6. Display operations analytics and full logs on a web-based dashboard

a. A unique URL addresses a web interface dashboard for the web service.

My dashboard's URL:

https://supreme-yodel-p4wpv5rv7g43rjqp-8080.app.github.dev/

Screen shot for my dashboard:



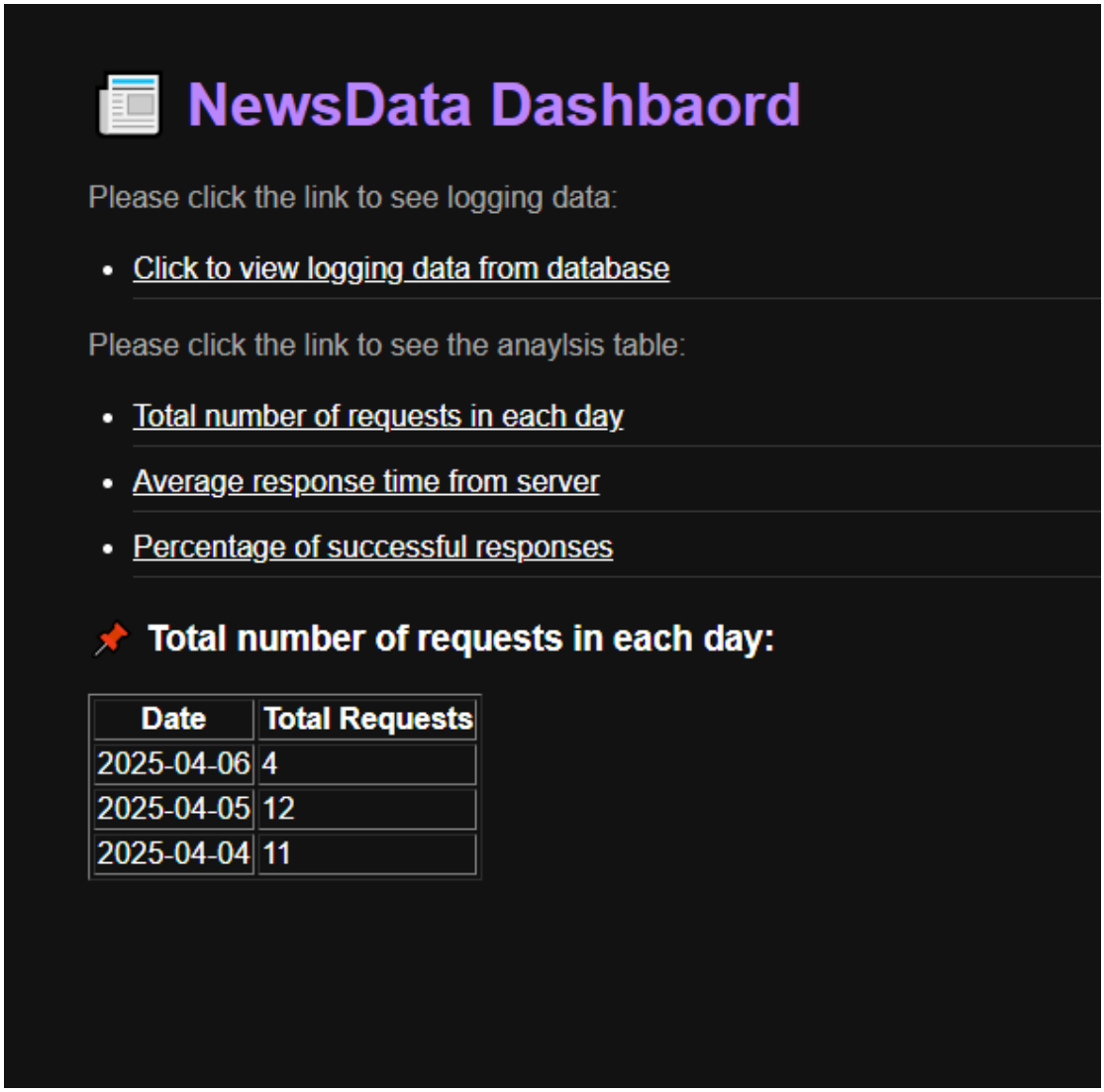
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

b. The dashboard displays at least 3 interesting operations analytics.

In the dashboard, I provide the 4 links to present out the log data, total number of requests in each day, average response time from server, and percentage of successful response.

- Total number of requests in each day

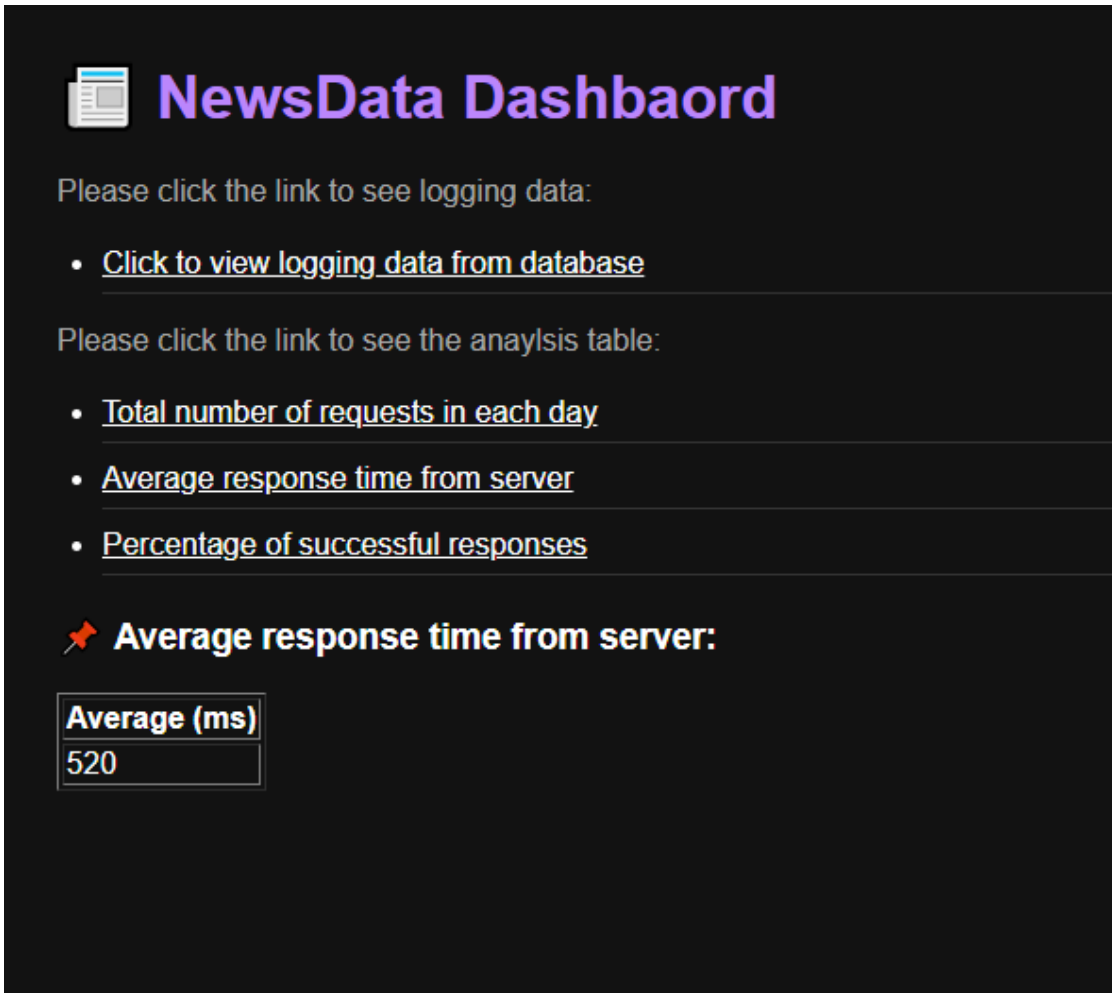
Once you click the link “Total number of requests in each day” you can see the total request in each day:



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

- Average response time from server

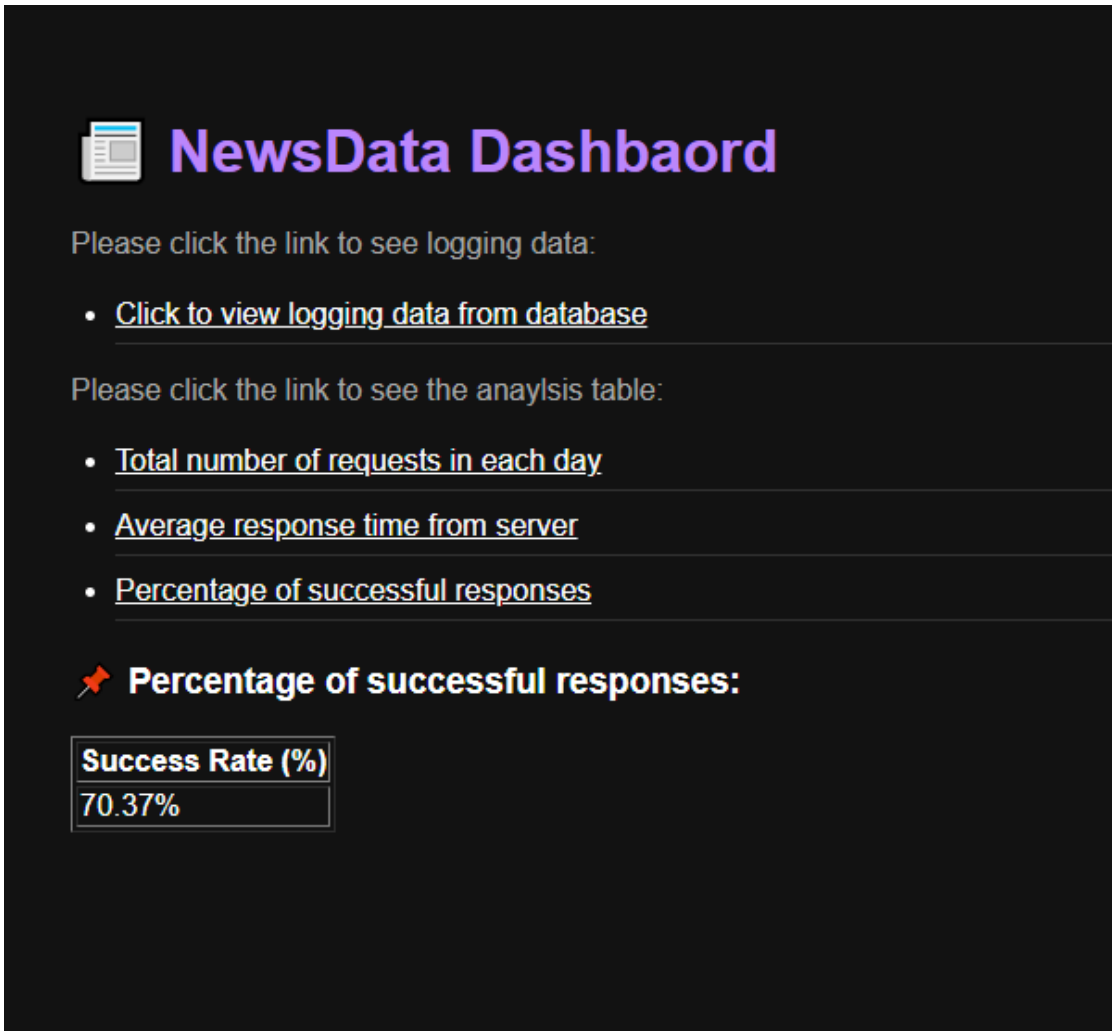
Once you click the link “Average response time from server”, you can see the average time that server response to client:



- Percentage of successful responses

Once you click the link “Percentage of successful responses”, you can check the rate for server successfully search a news for user:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh



OperationsAnalyticsCalculation.java shows the details for the following analytics:

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

```java
//A map that collect the total request in each day
public static Map<LocalDate, Integer> totalRequestInEachDay(List<Document> logs) {  1 usage
    Map<LocalDate, Integer> requestInEachDay = new HashMap<>();

    for (Document log : logs) {
        Date fullDate = log.getDate( key: "requestTime");
        //Convert to date-only data
        LocalDate dateOnly = fullDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDate();
        //Check if the date is existed in Map, if so, add 1 to the value.
        //If not, build up new date and set 1 for that key
        requestInEachDay.put(dateOnly, requestInEachDay.getOrDefault(dateOnly,  defaultValue: 0) + 1);
    }
    return requestInEachDay;
}
```

```java
//Calculate the average server response time
public static long averageServerResponseTime(List<Document> logs) {  1 usage
    //Get the numbers of data
    int totalData = logs.size();
    //Set up the total Server Response Time
    long totalServerResponseTime = 0;
    //Set up the average server Response Time
    long averageServerResponseTime = 0;

    //Get the total server response time
    for (Document log : logs) {
        totalServerResponseTime += log.getLong( key: "servletResponseTime");
    }
    // Calculate the average time for server response
    averageServerResponseTime = totalServerResponseTime/totalData;
    return averageServerResponseTime;
}
```

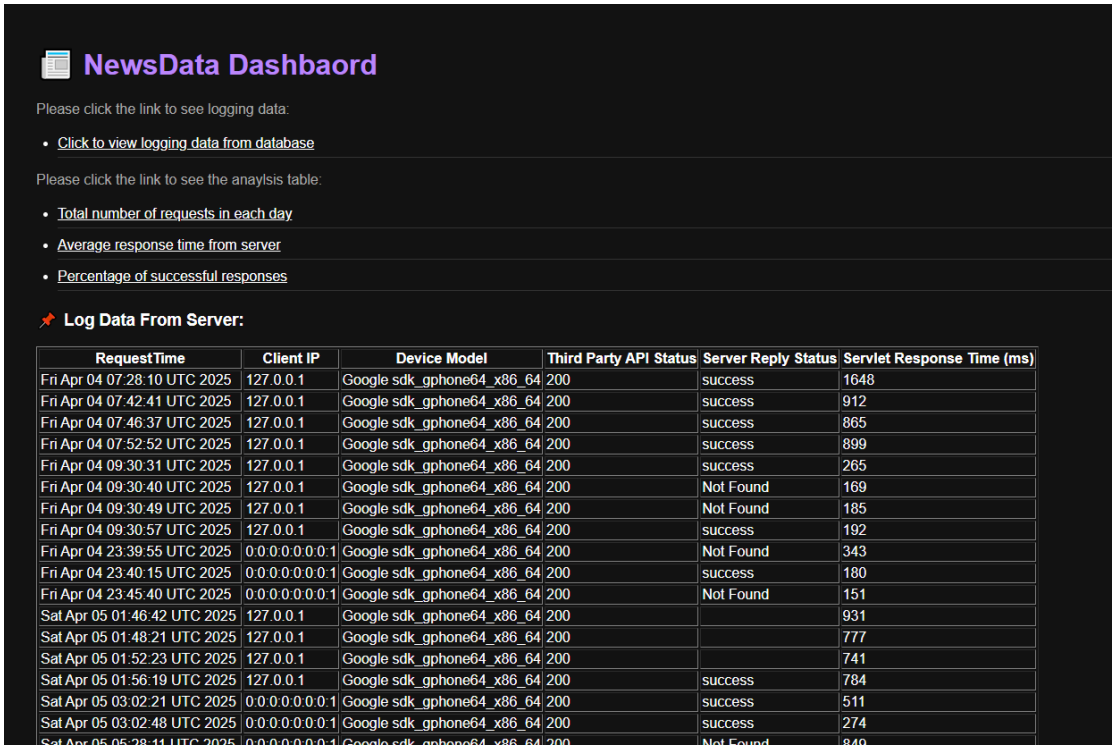
```java
//Calculate the success rate
public static String successRate(List<Document> logs) {  1 usage
    //Get the numbers of data
    double totalData = logs.size();
    //Set up the success data counter
    double successCount = 0;
    //String to show out the success rate
    String successRate = "";
    //Count the success data (which means it does search something from API key)
    for (Document log : logs) {
        if(log.getString( key: "serverReplyStatus").equals("success")) {
            successCount++;
        }
    }
    //Calculate the success rate
    double sucessPercentage = successCount/(double)totalData;
    //Build up the format
    sucessPercentage *= 100;
    successRate = String.format("%.2f", sucessPercentage);

    return successRate + "%";
}
```

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

c. The dashboard displays formatted full logs.

Screenshot for the format of the data logs:



NewsData Dashbaord

Please click the link to see logging data:

- Click to view logging data from database

Please click the link to see the anaylsis table:

- Total number of requests in each day
- Average response time from server
- Percentage of successful responses

Log Data From Server:

RequestTime	Client IP	Device Model	Third Party API Status	Server Reply Status	Servlet Response Time (ms)
Fri Apr 04 07:28:10 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	1648
Fri Apr 04 07:42:41 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	912
Fri Apr 04 07:46:37 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	865
Fri Apr 04 07:52:52 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	899
Fri Apr 04 09:30:31 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	265
Fri Apr 04 09:30:40 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	Not Found	169
Fri Apr 04 09:30:49 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	Not Found	185
Fri Apr 04 09:30:57 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	192
Fri Apr 04 23:39:55 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	Not Found	343
Fri Apr 04 23:40:15 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	180
Fri Apr 04 23:45:40 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	Not Found	151
Sat Apr 05 01:46:42 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		931
Sat Apr 05 01:48:21 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		777
Sat Apr 05 01:52:23 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200		741
Sat Apr 05 01:56:19 UTC 2025	127.0.0.1	Google sdk_gphone64_x86_64	200	success	784
Sat Apr 05 03:02:21 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	511
Sat Apr 05 03:02:48 UTC 2025	0:0:0:0:0:0:0:1	Google sdk_gphone64_x86_64	200	success	274

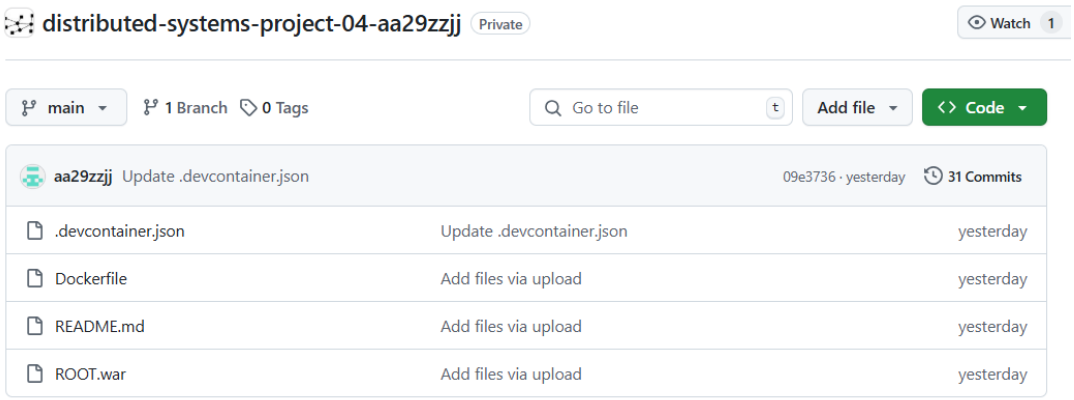
7. Deploy the web service to GitHub Codespaces

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

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Screenshot of my github repo:

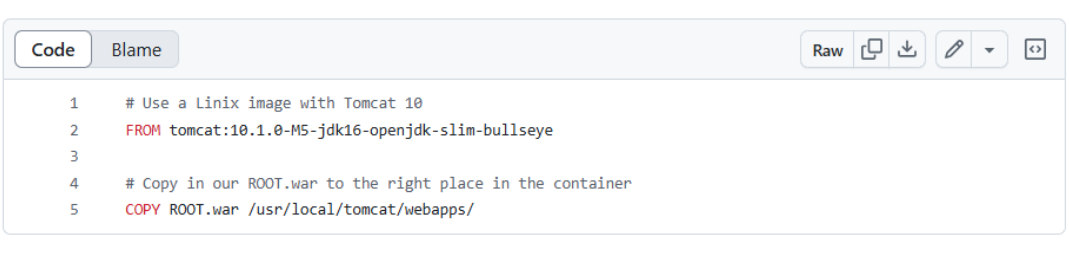


.devcontainer.json:

I add some attribute to my port to make the port visibility to be public

Dockerfile:

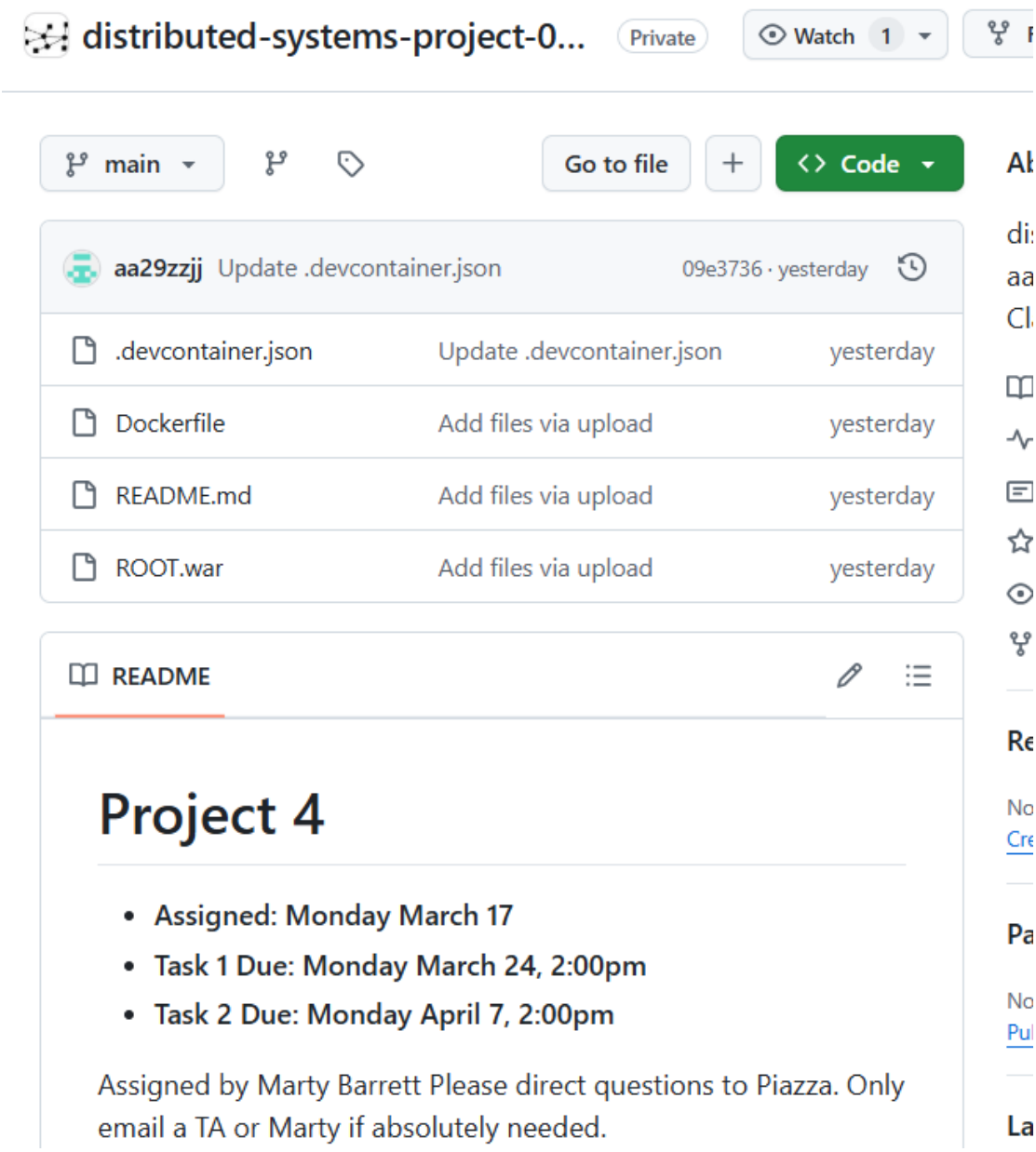
Remain the same



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

ROOT.war: my web app serlvet's war file

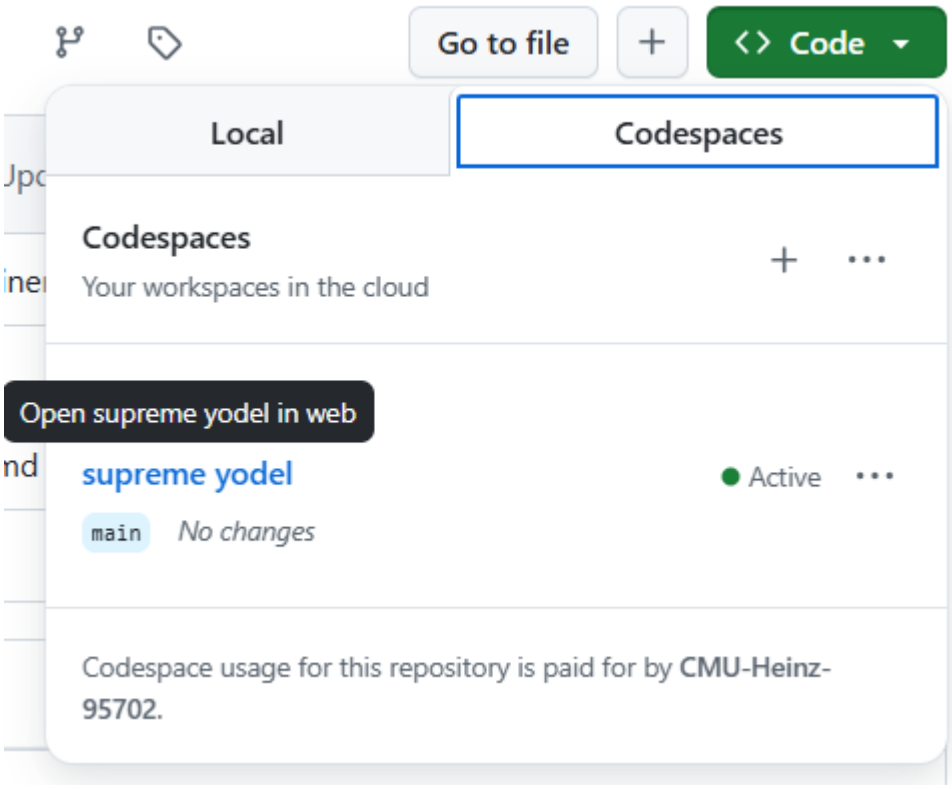
README.md: An identical copy of this README.md



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

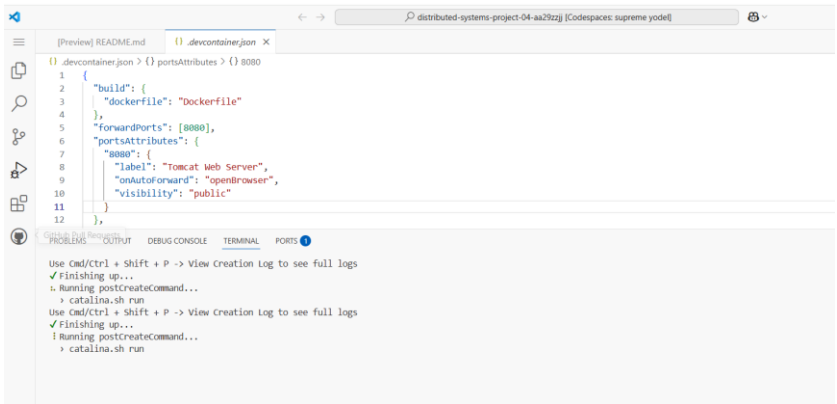
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Screen shot of my codespaces that launch my web server (supreme yodel):



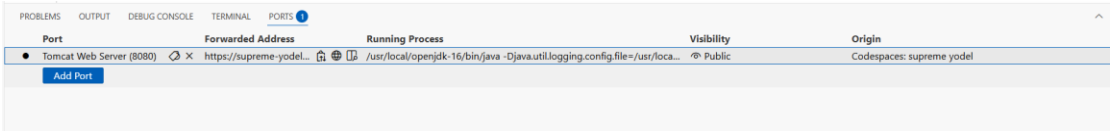
c. Once the Codespace is running, the Terminal tab will show that Catalina (the Servlet container) is running. You should also see a "1" next to the Ports tab. Click on the Ports tab and you should see that port 8080 has been made available.

Terminal:



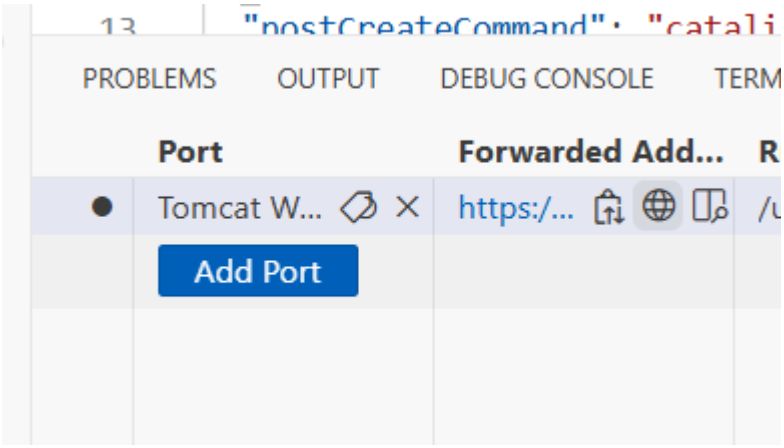
Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Port:



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.

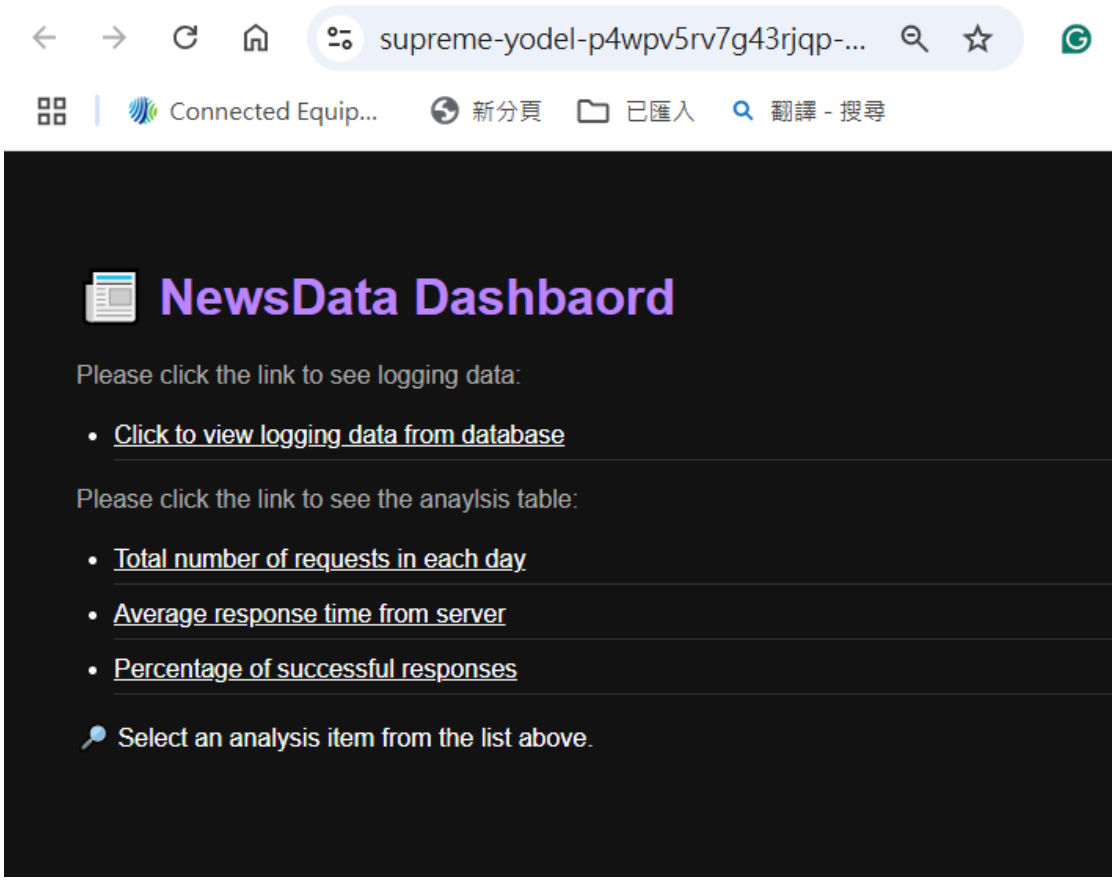
Click the middle one (a globe) button to get into browser:



Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

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

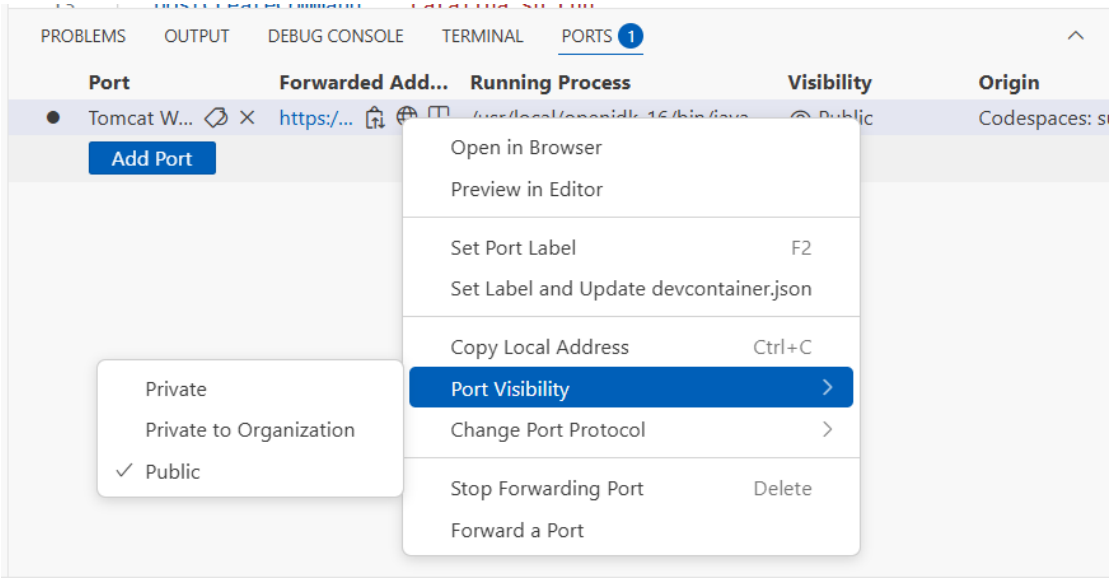
Browser:



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.

Course: Distribution System Management
Instructor: Prof. McCarthy, Prof. Barrett
Name: Jerry Huang (Tzu-Chieh Huang)
Andrew ID: jerryh

Set the port visibility to public:



g. Copy the URL and paste into an Incognito Chrome window to confirm that the Hello World web app can be reached without authentication.

As what I did previously, after set into public, my browser can be seen 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.

Already put on my github repo.