Name: **Snehal Khandve**
Andrew ID: **skhandve**

Project 4 Task 2 – Sunset Sunrise App

1. Description: My application takes latitude and longitude as compulsory inputs and date (YYYY-MM-DD and other formats mentioned in the documentation) to display the sunrise and sunset timings at that place along with a nice picture on the app.

2. My Dashboard URL - https://ideal-space-capybara-9jp5qjpgj4pcp67x-8080.app.github.dev/dashboard

3. API used - https://sunrise-sunset.org/api#documetation

1. Implement a native Android application

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

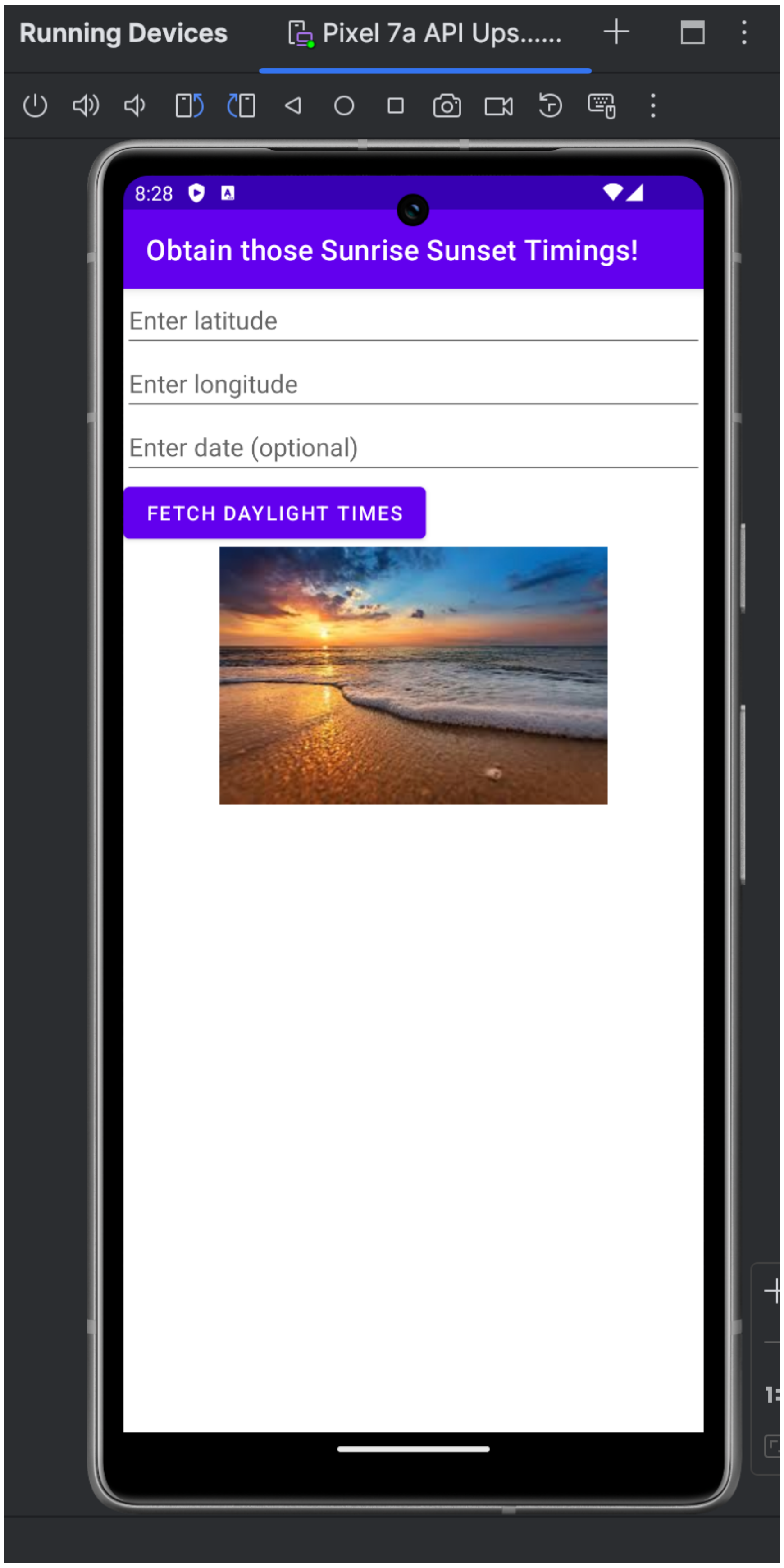
a. Has at least three different kinds of Views in your Layout

My application uses TextView, EditText, Button, and ImageView. See content_main.xml for details of how they are incorporated into the LinearLayout.

Code Snippet:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <EditText
        android:id="@+id/editTextLatitude"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter latitude"/>

    <EditText
        android:id="@+id/editTextLongitude"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter longitude"/>

    <EditText
        android:id="@+id/editTextDate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter date (optional)"/>

    <Button
        android:id="@+id/buttonFetch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fetch Daylight Times"/>

    <TextView
        android:id="@+id/textViewResults"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:textSize="18sp"
        android:textColor="@color/black"
        android:background="@color/teal_200"
        android:textStyle="bold"
        android:gravity="center_horizontal"
        android:layout_marginTop="20dp"
        android:visibility="gone"/>


    <ImageView
        android:id="@+id/interestingPicture"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/my_image"/>

</LinearLayout>
```

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

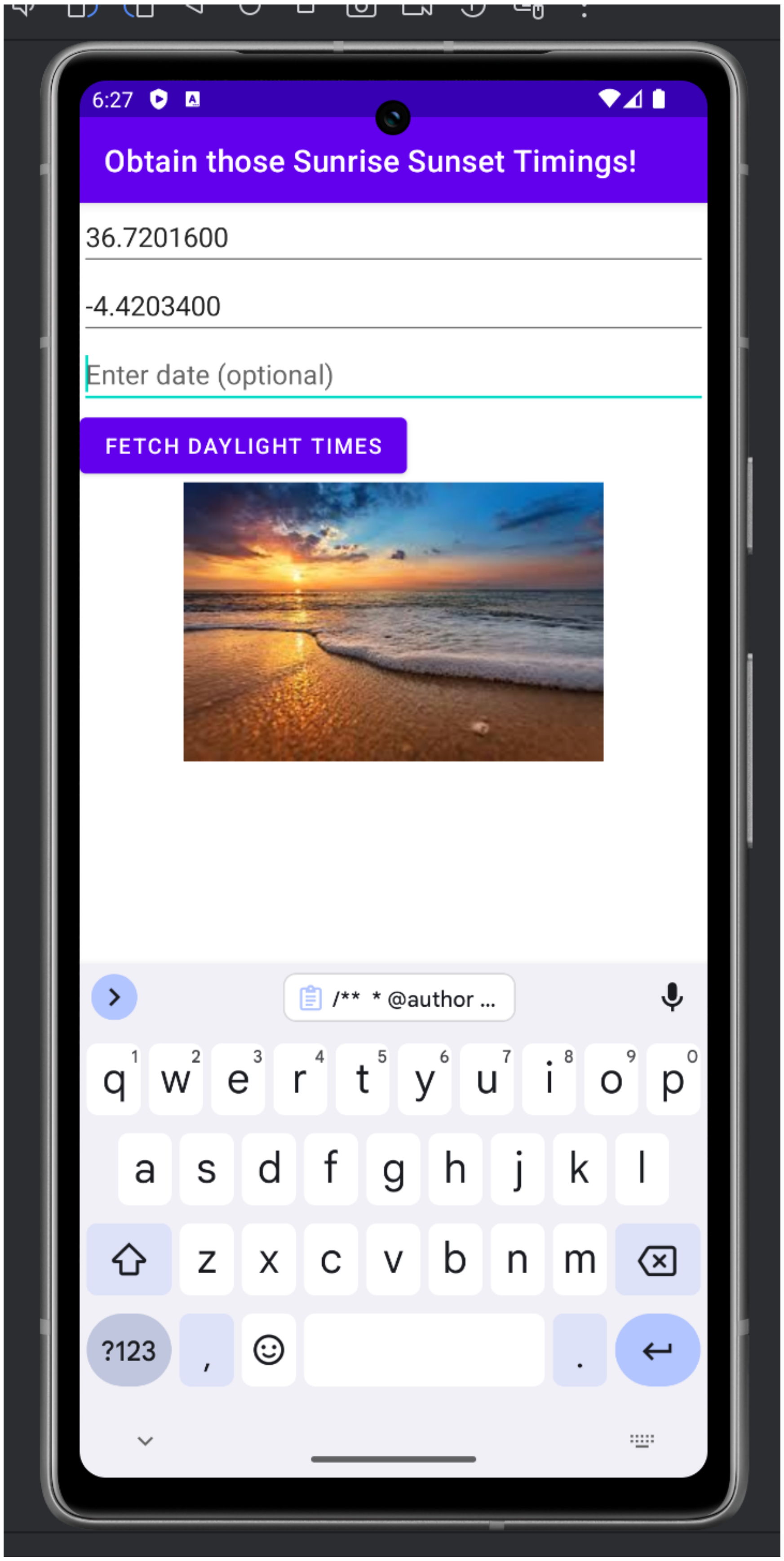

Running Devices
Pixel 7a API Ups......
8:28
Obtain those Sunrise Sunset Timings!
Enter latitude
Enter longitude
Enter date (optional)
FETCH DAYLIGHT TIMES

b. Requires input from the user

Code Snippet:

```
submitButton.setOnClickListener(new View.OnClickListener(){
    public void onClick(View viewParam) {
        //obtaining the attributes from the text field of the app

        String searchLatitude =
((EditText)findViewById(R.id.editTextLatitude))
                .getText().toString();

        String searchLongitude =
((EditText)findViewById(R.id.editTextLongitude))
                .getText().toString();

        String searchDate =
((EditText)findViewById(R.id.editTextDate))
                .getText().toString();

        // check is the input is as expected
        validateInput(searchLatitude, searchLongitude);

        //obtain required data from the server
        FetchData gp = new FetchData();
        gp.interactWithServer(searchLatitude, searchLongitude,
searchDate, me, ma);
    }
});
```

Here is a screenshot of the user entering the required fields:



6:27

Obtain those Sunrise Sunset Timings!

36.7201600

-4.4203400

Enter date (optional)

FETCH DAYLIGHT TIMES

c. Makes an HTTP request (using an appropriate HTTP method) to your web service

My application does an HTTP GET request in FetchData.java. The HTTP request is: "https://ideal-space-capybara-9jp5qjpgj4pcp67x-8080.app.github.dev/daylight?lat=" + latitude + "&lng=" + longitude + (date != null ? "&date=" + date : "")

The fetch method makes this request of my web application, parses the returned JSON to find the sunrise and sunset timings.

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

An example of the JSON reply is:

```
{
    "sunrise": "2:06:54 AM",
    "sunset":"3:50:53 PM"
}
```

Code Snippet:

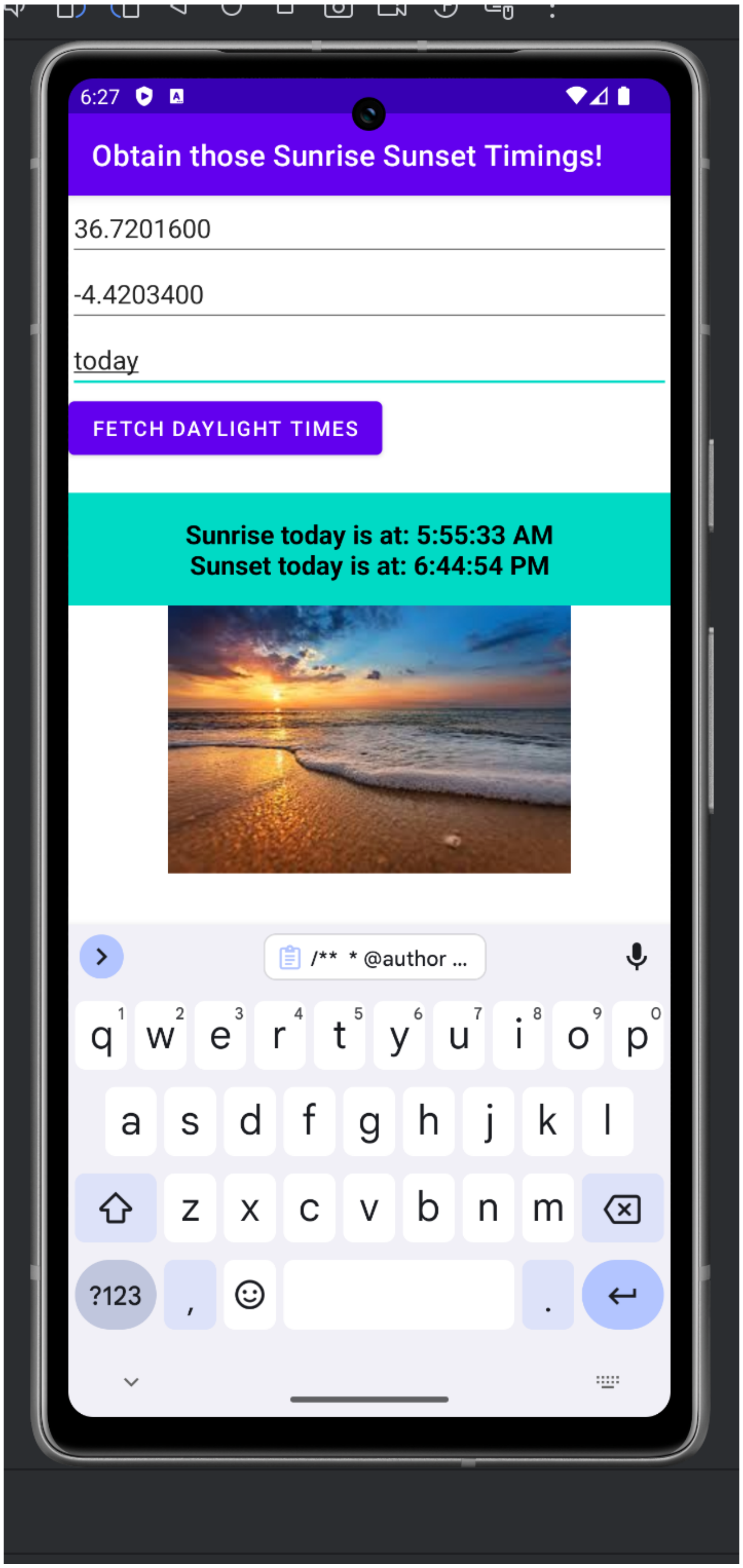
```
private boolean fetch(String latitude, String longitude,
String date) {
    try {
        String webServiceURL = "https://ideal-space-
capybara-9jp5qjpgj4pcp67x-
8080.app.github.dev/daylight?lat=" +
                latitude + "&lng=" + longitude + (date !=
null ? "&date=" + date : "");
        URL url = new URL(webServiceURL);
        HttpURLConnection urlConnection =
(HttpURLConnection) url.openConnection();

        //response obtained from the web service
        int responseCode = urlConnection.getResponseCode();

        InputStream inputStream;
        if (responseCode >= 200 && responseCode < 400) {
            // Success response
            inputStream = urlConnection.getInputStream();
        } else {
            return false;
        }

        try {

            //Read the response from the web server.
            BufferedReader bufferedReader = new
BufferedReader(new InputStreamReader(inputStream));
            StringBuilder stringBuilder = new
StringBuilder();
            String line;

            while ((line = bufferedReader.readLine()) !=
null) {
                stringBuilder.append(line).append("\n");
            }

            JSONObject jsonObject = new
JSONObject(stringBuilder.toString());
            sunriseTime = jsonObject.getString("sunrise");
            sunsetTime = jsonObject.getString("sunset");

            bufferedReader.close();

        } finally {
            urlConnection.disconnect();
        }

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
    return true;
}
```

e. Displays new information to the user

Code Snippet:

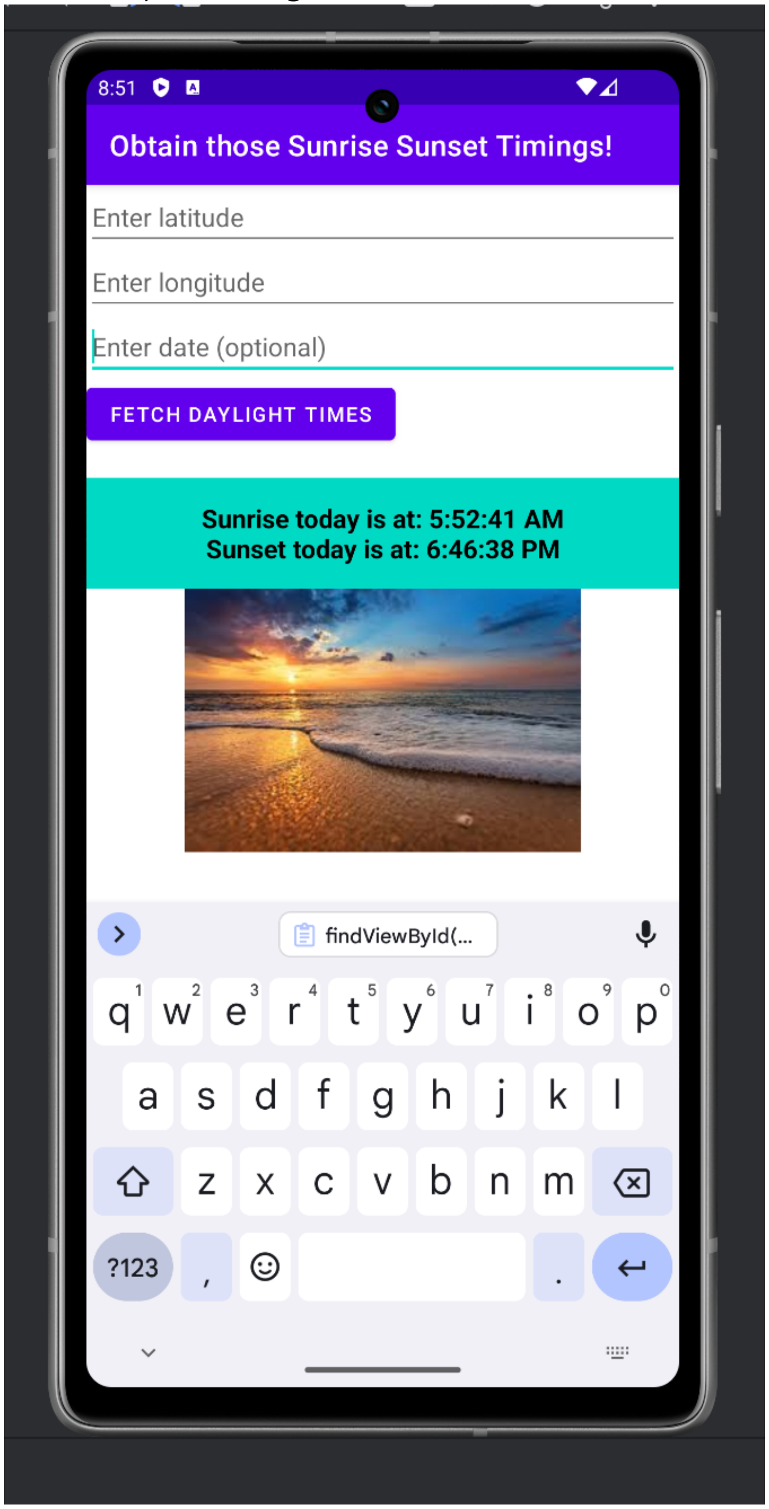
```
/**
 * Display the output on the right text view on the app.
 * */
public void updateTimings(String sunrise, String sunset) {
    TextView textViewResults = findViewById(R.id.textViewResults);

    // Check if the strings are not empty
    if (!sunrise.isEmpty() && !sunset.isEmpty()) {
        String timings = String.format(Locale.getDefault(), "Sunrise
today is at: %s\nSunset today is at: %s", sunrise, sunset);
        textViewResults.setText(timings);

        // Make the TextView visible
        textViewResults.setVisibility(View.VISIBLE);
    }
}
```

Here is the screen shot after the timings have been returned.


6:27
Obtain those Sunrise Sunset Timings!
36.7201600
-4.4203400
today
FETCH DAYLIGHT TIMES
Sunrise today is at: 5:55:33 AM
Sunset today is at: 6:44:54 PM
/** * @author ...

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

Below screenshot shows the timings for the previous inputs (36.7201600, -4.4203400) but asking for the next one too. Thus, the user can enter positions.



Sunrise today is at: 5:52:41 AM
Sunset today is at: 6:46:38 PM

2. Implement a web service

a. Implement a simple (can be a single path) API.
The URL of my web service deployed to CodeSpaces is: https://ideal-space-capybara-9jp5qjpgj4pcp67x-8080.app.github.dev/dashboard

The project directory name is: Project4WebApp
In my web app project:
1. Model: The request is received from the Mobile Application, hence the app is the model, meaning the web service does not have a model as such independently.
2. View: index.jsp and dashboard.jsp(for the dashboard)
3. Controller: DaylightServiceServlet.java

b. Receives an HTTP request from the native Android application:

DaylightServiceServlet.java receives the HTTP GET request with the argument "lat", “lng” and “date” that stand for latitude, longitude and date correspondingly. It passes these inputs as string to the 3rd party API.

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

DaylightServiceServlet.java makes an HTTP request to: https://api.sunrise-sunset.org/json?lat=" + latitude + "&lng=" + longitude + "&date=" + date
It then parses the JSON response and processes it to extract the required parts it needs to send to the Android application.

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

An example of the JSON reply sent back to the Android application:
{
"sunrise": "2:06:54 AM",
"sunset":"3:50:53 PM"
}

Code Snippet:

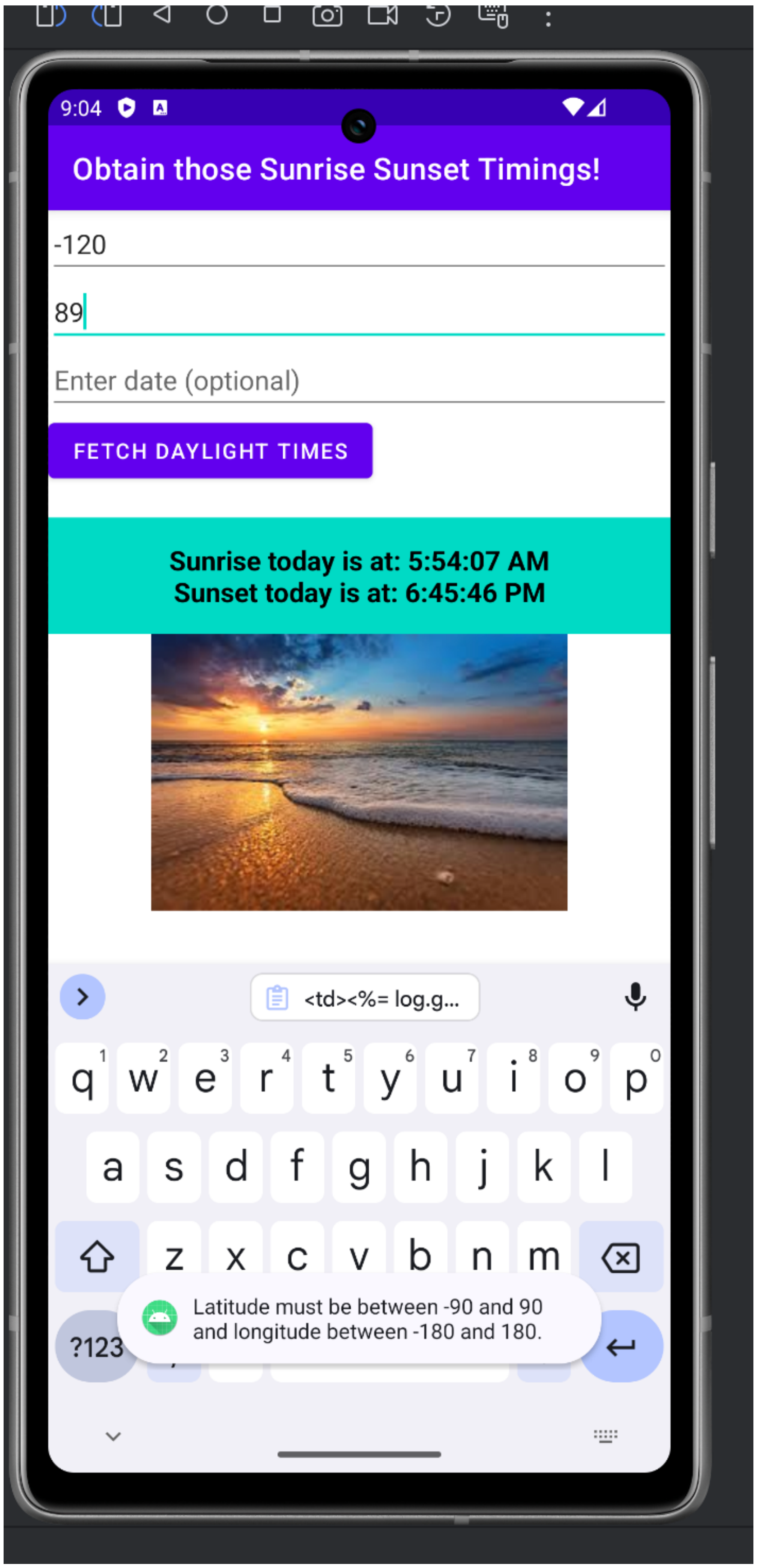
```
/**
 * @author Snehal Khandve
 * Andrew ID: skhandve
 * */
package ds.project4webapp;
```

```
import java.io.*;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.TimeZone;

import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import org.json.JSONObject;

/**
 * The servlet for the app to call.
 * */
@WebServlet(name = "DaylightServiceServlet", urlPatterns = {"/daylight"})
public class DaylightServiceServlet extends HttpServlet {

    //instance of the dashboard servlet
    DashboardServlet dashboardServlet =  new DashboardServlet();

    /**
     * This function will be called by the app when it calls this servlet.
     * */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws IOException {

        //To log browser and device details of the app making the request
        String userAgent = request.getHeader("User-Agent");

        //startTime for the ASI request
        long startTime = System.currentTimeMillis();

        //get the required parameters from the request
        String latitude = request.getParameter("lat");
        String longitude = request.getParameter("lng");
        String date = request.getParameter("date");

        //validate the parameters
        if (!isValidInput(latitude, longitude)) {
            respondWithError(response, "Invalid input parameters.");
            return;
        }

        try {
            //if the date is not set, set it as an empty string
            date = (date != null ? date : "");


            String apiUrl = "https://api.sunrise-sunset.org/json?lat=" +
latitude + "&lng=" + longitude + "&date=" + date;
            URL url = new URL(apiUrl);
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setRequestMethod("GET");
            //reponse from the API
            int responseCode = conn.getResponseCode();

            //end time of API request processing
            long apiResponseTime = System.currentTimeMillis() - startTime;

            //update the logs to the DB.
            ServiceLog logEntry = new ServiceLog(latitude, longitude,
date, String.valueOf(apiResponseTime),
                    String.valueOf(responseCode), userAgent);
            dashboardServlet.updateDb(logEntry);

            // Handle third-party API unavailability or errors
            if (responseCode != HttpURLConnection.HTTP_OK) {
                respondWithError(response, "Third-party API unavailable or
returned an error.");
                return;
            }

            BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            String inputLine;
            StringBuffer content = new StringBuffer();
            while ((inputLine = reader.readLine()) != null) {
                content.append(inputLine);
            }
            reader.close();
            conn.disconnect();

            // Parse the original JSON response
            JSONObject jsonResponse = new JSONObject(content.toString());

            // Handle invalid data from the third-party API
            if (!jsonResponse.getString("status").equals("OK")) {
                respondWithError(response, "Third-party API returned
invalid data.");
                return;
            }

            JSONObject results = jsonResponse.getJSONObject("results");

            // Extract required fields, processing the response
            String sunrise = results.getString("sunrise");
            String sunset = results.getString("sunset");

            // Construct a new JSON object with only required fields
            JSONObject newJsonResponse = new JSONObject();
            newJsonResponse.put("sunrise", sunrise);
            newJsonResponse.put("sunset", sunset);

            // Assuming we simply pass the new JSON response back to the
client
            PrintWriter out = response.getWriter();
            response.setContentType("application/json");
            response.setCharacterEncoding("UTF-8");
            out.print(newJsonResponse);
            out.flush();
```

```
        } catch (IOException e) {
            // Handle network failures
            respondWithError(response, "Network failure: Unable to reach
the third-party API.");
        } catch (Exception e) {
            // Handle any other errors
            respondWithError(response, "An error occurred.");
        }
    }

    //validation is present at the app level, but double validating the
attributes
    private boolean isValidInput(String latitude, String longitude) {
        if (latitude == null || latitude.isBlank() || longitude == null ||
longitude.isBlank()) {
            return false;
        }

        double lat = Double.parseDouble(latitude);
        double lon = Double.parseDouble(longitude);

        if (lat < -90 || lat > 90 || lon < -180 || lon > 180) {
            return false;
        }
        return true;
    }

    /**
     * To respond with errors in case of bad request.
     * */
    private void respondWithError(HttpServletResponse response, String
errorMessage) throws IOException {
        JSONObject errorResponse = new JSONObject();
        errorResponse.put("error", errorMessage);

        PrintWriter out = response.getWriter();
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        response.setStatus(HttpServletResponse.SC_BAD_REQUEST);
        out.print(errorResponse);
        out.flush();
    }
}
```

3. Handle error conditions

Below are some of the screenshots where I am giving a Toast message to the Android app user in case of invalid inputs.


9:04

Obtain those Sunrise Sunset Timings!

-120

89

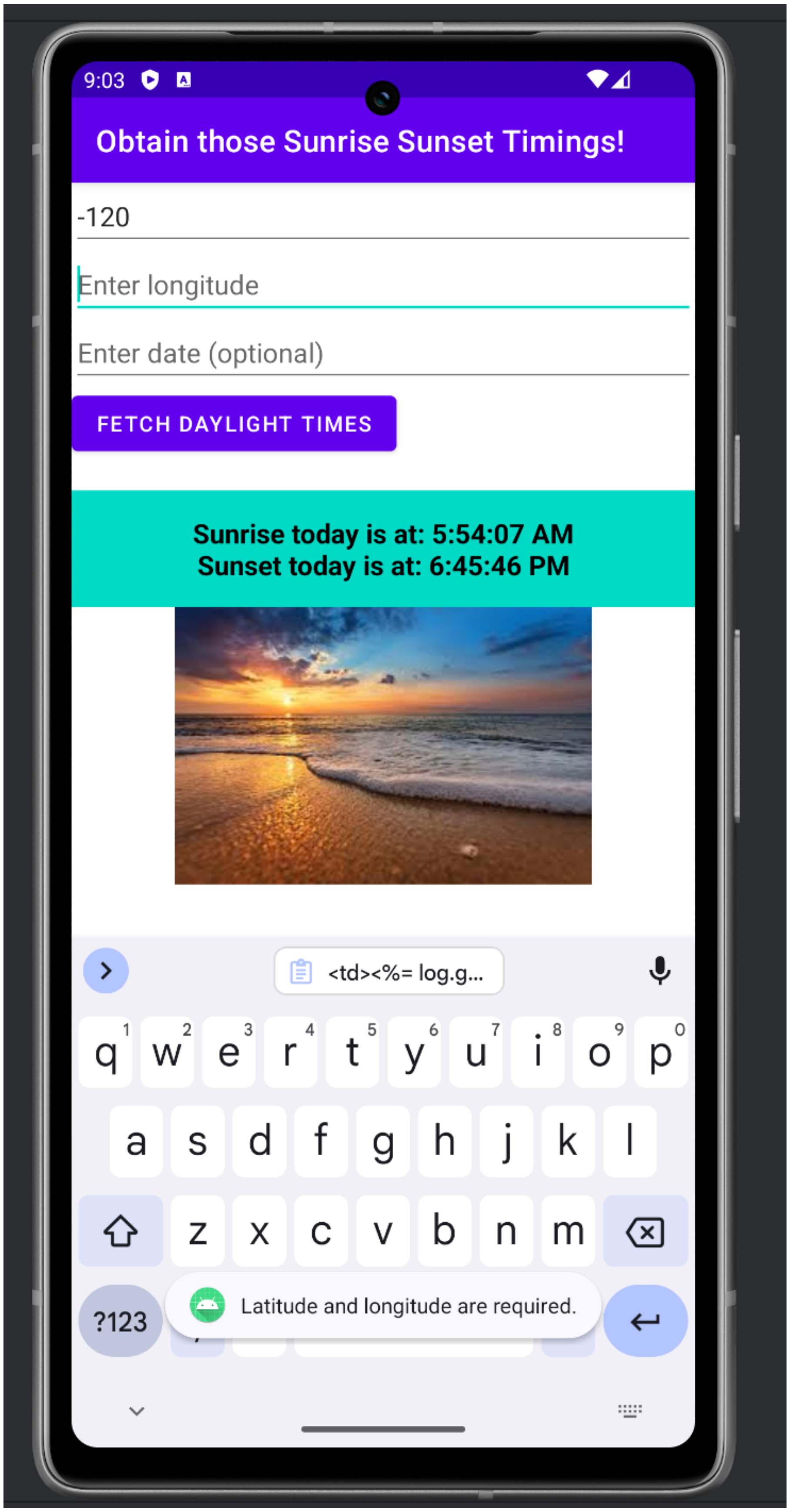
Enter date (optional)

FETCH DAYLIGHT TIMES

Sunrise today is at: 5:54:07 AM
Sunset today is at: 6:45:46 PM

<td><%= log.g...

Latitude must be between -90 and 90 and longitude between -180 and 180.



9:03

Obtain those Sunrise Sunset Timings!

-120

Enter longitude

Enter date (optional)

FETCH DAYLIGHT TIMES

Sunrise today is at: 5:54:07 AM
Sunset today is at: 6:45:46 PM

Latitude and longitude are required.

Server side validation is also done in the DaylightServiceServlet.java in the web service.

4. Log useful information:

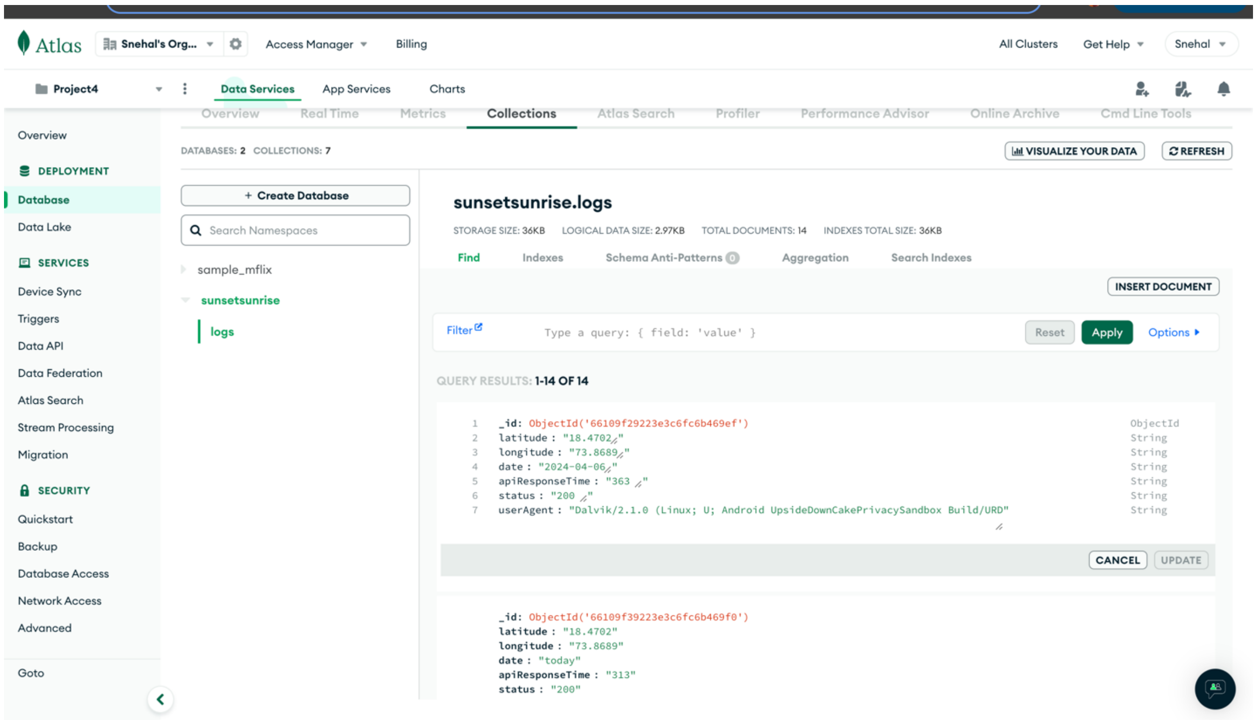
   Logging attributes:
   1) Latitude – The record of latitude value entered by the user in a request will help to keep track of the incoming data in the request.
   2) Longitude - The record of longitude value entered by the user in a request will help to keep track of the incoming data in the request.
   3) Date - The record of date value entered by the user in a request will help to keep track of the incoming data in the request.
   4) API Response Time – The time taken by the third-party API to respond to the request. Will help to know how slow/fast the API is working.
   5) User Agent Details – This entails details like which device, OS, and browser is being used to access the Android application.
   6) Status - This is the status code returned by the third-party API for the request.

   Operations analytics:
   1) Geographic distribution – How many requests are coming from a particular location, maintaining only the latitude as the location specification. This will help analyze which location has the relatively high requests.
   2) Total API Requests – The total requests hit from the Android application and received by the web service. This will help us to keep track of the total incoming requests.
   3) Response Time Analysis – The average time taken by the third-party API to respond to a request. The more the average time, less performing the API is.
   4) Status Code Distribution – The response status code analysis. This will help check how successfully the users are able to hit the API.
   5) Error Logs Summary – The error messages sent to the Android application. The error messages sent to the application from the web server.

5. Store the log information in a database - Give your Atlas connection string with the three shards

   mongodb://skhandve:YOuKkslbgwJnXxO9@ac-xlbvwwf-shard-00-00.cyi86m3.mongodb.net:27017,ac-xlbvwwf-shard-00-01.cyi86m3.mongodb.net:27017,ac-xlbvwwf-shard-00-02.cyi86m3.mongodb.net:27017/test?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1

   Following is the MongoDB console screenshot:



sunsetsunrise.logs

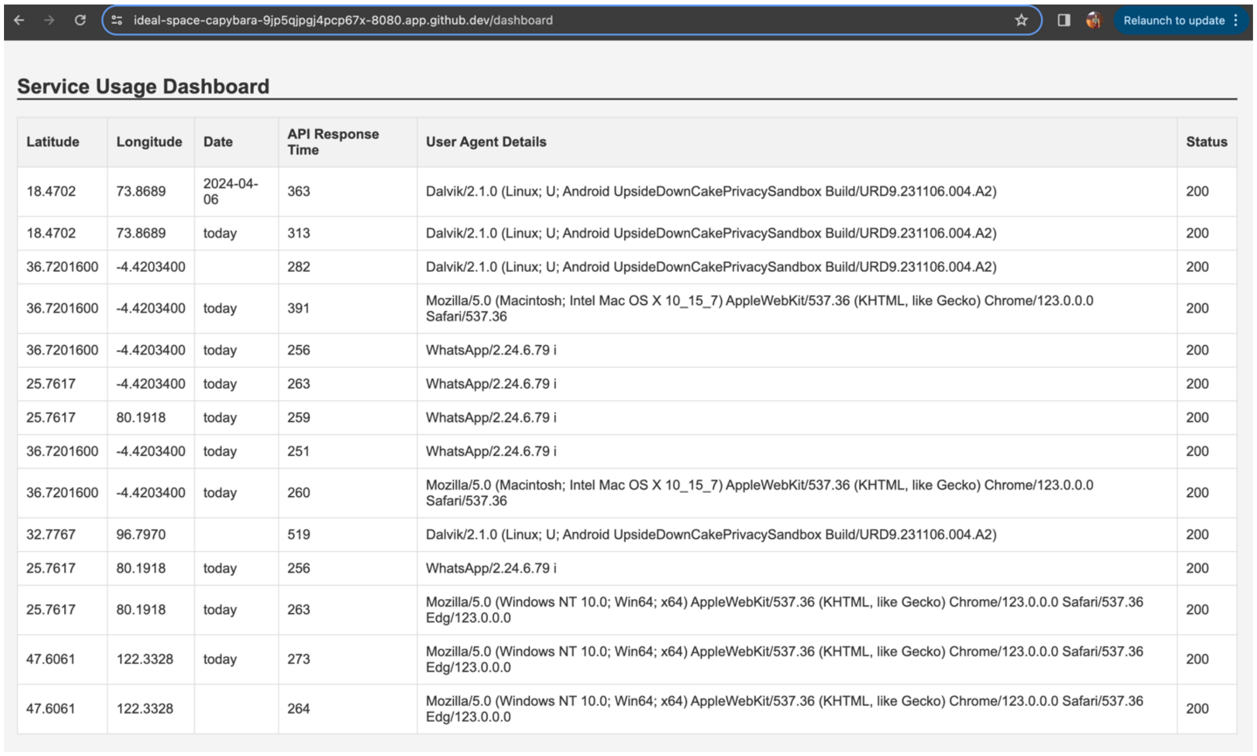
STORAGE SIZE: 36KB LOGICAL DATA SIZE: 2.97KB TOTAL DOCUMENTS: 14 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-14 OF 14

```
_id: ObjectId('66109f29223e3c6fc6b469ef')                                             ObjectId
latitude : "18.4702"                                                                  String
longitude : "73.8689"                                                                 String
date : "2024-04-06"                                                                   String
apiResponseTime : "363 "                                                              String
status : "200 "                                                                       String
userAgent : "Dalvik/2.1.0 (Linux; U; Android UpsideDownCakePrivacySandbox Build/URD"   String
```

```
_id: ObjectId('66109f39223e3c6fc6b469f0')
latitude : "18.4702"
longitude : "73.8689"
date : "today"
apiResponseTime : "313"
status : "200"
```

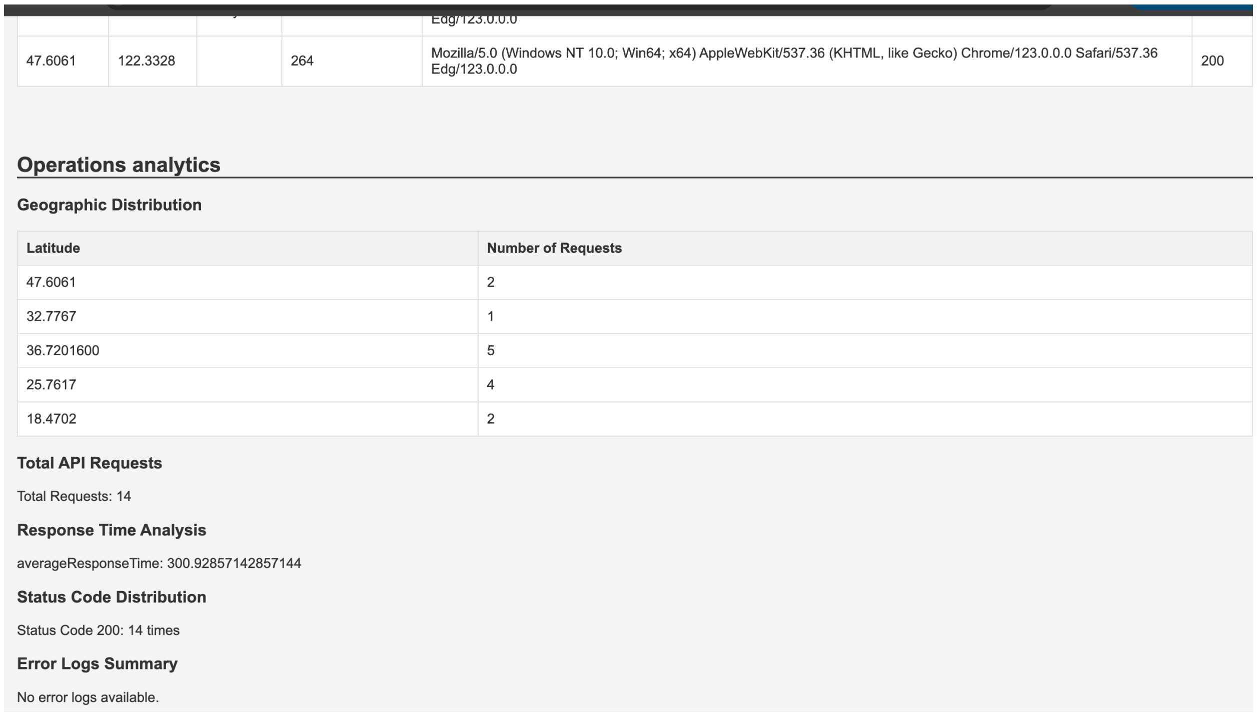
6. Display operations analytics and full logs on a web-based dashboard - Provide a screen shot.



ideal-space-capybara-9jp5qjpgj4pcp67x-8080.app.github.dev/dashboard
Relaunch to update

Service Usage Dashboard

Latitude	Longitude	Date	API Response Time	User Agent Details	Status
18.4702	73.8689	2024-04-06	363	Dalvik/2.1.0 (Linux; U; Android UpsideDownCakePrivacySandbox Build/URD9.231106.004.A2)	200
18.4702	73.8689	today	313	Dalvik/2.1.0 (Linux; U; Android UpsideDownCakePrivacySandbox Build/URD9.231106.004.A2)	200
36.7201600	-4.4203400		282	Dalvik/2.1.0 (Linux; U; Android UpsideDownCakePrivacySandbox Build/URD9.231106.004.A2)	200
36.7201600	-4.4203400	today	391	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36	200
36.7201600	-4.4203400	today	256	WhatsApp/2.24.6.79 i	200
25.7617	-4.4203400	today	263	WhatsApp/2.24.6.79 i	200
25.7617	80.1918	today	259	WhatsApp/2.24.6.79 i	200
36.7201600	-4.4203400	today	251	WhatsApp/2.24.6.79 i	200
36.7201600	-4.4203400	today	260	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36	200
32.7767	96.7970		519	Dalvik/2.1.0 (Linux; U; Android UpsideDownCakePrivacySandbox Build/URD9.231106.004.A2)	200
25.7617	80.1918	today	256	WhatsApp/2.24.6.79 i	200
25.7617	80.1918	today	263	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0	200
47.6061	122.3328	today	273	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0	200
47.6061	122.3328		264	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0	200



				Edg/123.0.0.0	
47.6061	122.3328		264	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36 Edg/123.0.0.0	200

Operations analytics

Geographic Distribution

Latitude	Number of Requests
47.6061	2
32.7767	1
36.7201600	5
25.7617	4
18.4702	2

Total API Requests

Total Requests: 14

Response Time Analysis

averageResponseTime: 300.92857142857144

Status Code Distribution

Status Code 200: 14 times

Error Logs Summary

No error logs available.