Student Name: Yingjie Feng;

Andrew ID: yfeng3;

The name of the API: OPENWEATHER

This mobile application, named 'WeatherMate,' provides real-time weather updates for major cities around the world.

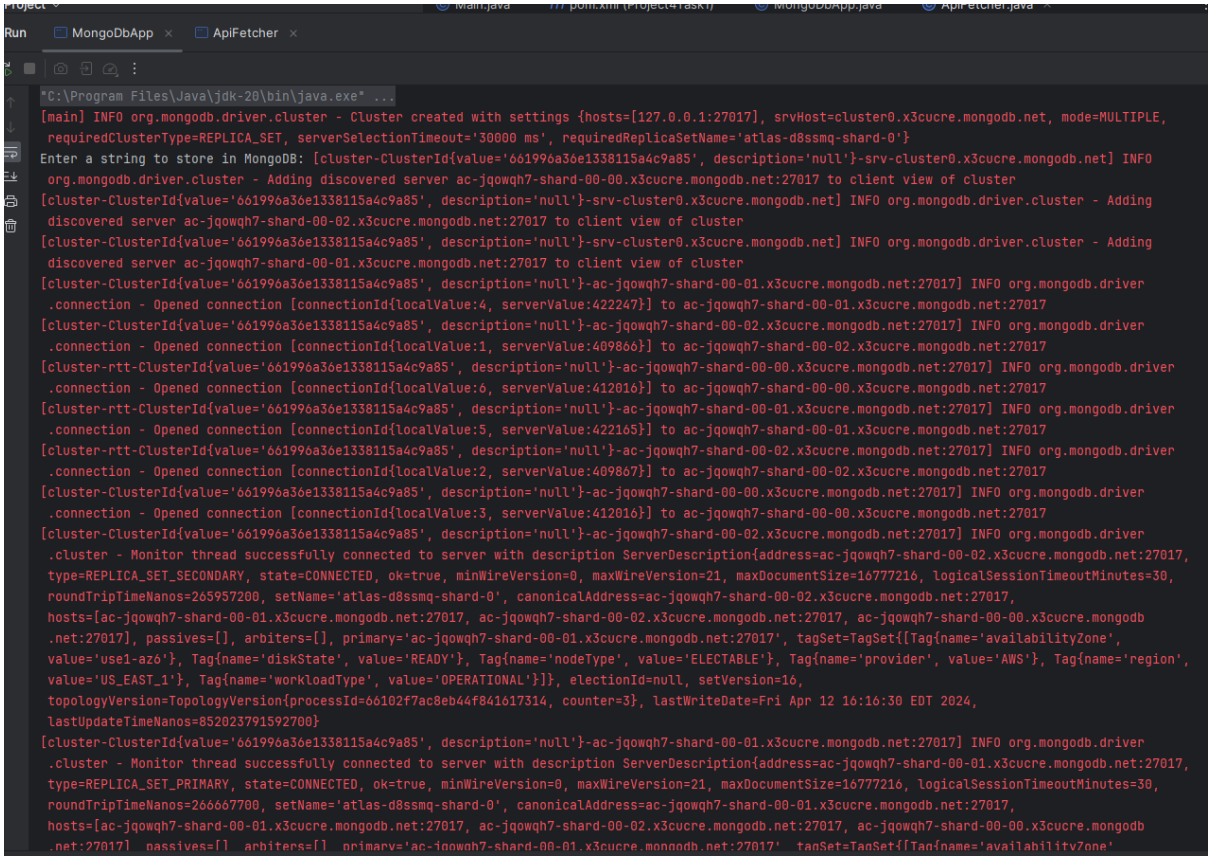
Users can enter the name of a city, and the app will display current weather conditions including temperature, humidity, cloudiness, and wind speed.

'WeatherMate' app aims to help users plan their day better by providing accurate and timely weather forecasts.

1b:


```
{
    "visibility": 10000,
    "timezone": 3600,
    "main": {
        "temp": 288.26,
        "temp_min": 286.4,
        "humidity": 72,
        "pressure": 1026,
        "feels_like": 287.71,
        "temp_max": 289.31
    },
    "clouds": {"all": 100},
    "sys": {
        "country": "GB",
        "sunrise": 1712898620,
        "sunset": 1712947897,
        "id": 2075535,
        "type": 2
    },
    "dt": 1712951262,
    "coord": {
        "lon": -0.1257,
        "lat": 51.5085
    },
    "weather": [{
        "icon": "04n",
        "description": "overcast clouds",
        "main": "Clouds",
        "id": 804
    }],
    "name": "London",
    "cod": 200,
    "id": 2643743,
    "base": "stations",
    "wind": {
        "deg": 230,
        "speed": 5.14
```


2c:



Application Description and Documentation

Author: Yingjie Feng

Project Name: Weather Info App

Description:

This application allows users to enter a **city** name and fetches weather information for that city using the **OpenWeatherMap API**. The weather data is displayed on a native Android app, and interactions are logged in a MongoDB database. The application consists of a backend servlet that handles API requests and interacts with MongoDB, and an Android front end for user interactions.

1. Implement a native Android application

Project Name in Android Studio: WeatherInfoAndroid

Views Used: The layout includes TextView for displaying results, EditText for user input, and Button to submit the search.

User Input: The user is required to enter the name of a city to fetch weather information.

HTTP Requests: The application makes HTTP GET requests to the backend servlet. This is done in the **fetchWeather method** in the MainActivity class.

JSON Parsing: The app receives a JSON formatted response from the backend and parses it to display the temperature and weather description.

Display of Information: Displays weather data fetched from the backend. Example screenshots can be included here.

Repeatability: The app allows users to repeatedly search for different cities without needing to restart the application.

2. Implement a web service

- **Web Service URL**: (URL of your deployed web service, e.g., Heroku or another cloud service)
- **Servlet Implementation**: The backend uses **ApiServlet** which extends **HttpServlet**.
- **HTTP Request Handling**: The servlet receives GET requests with a city name as a parameter.
- **Business Logic**:
    - The **ApiServlet** makes a request to the OpenWeatherMap API using the received city

name.

    - It parses the JSON response to extract and format the temperature and weather conditions.

- **Response**: The servlet sends back a JSON formatted response containing the weather data to the Android app.
- **Servlet Deployment**: Details about deployment, such as using Heroku or another cloud service, can be added here.

The backend of the application is handled by a Java servlet deployed on a web server, which receives requests from the Android app. Upon receiving a **city name via the GET request**, the servlet makes an API call to OpenWeatherMap to retrieve the current weather conditions for that city. After processing the API's JSON response, the servlet formats this information into a JSON structure and sends it back to the Android app.It parses the JSON response to extract and format the temperature and weather conditions.

3. Log useful information

Logged Information:

Request details from the mobile app including the requested city and timestamp.

API request details including the full request URL to OpenWeatherMap and the timestamp.

API response details including temperature and weather description.

Response details sent back to the mobile application.

Reason for Logging: Logging is used to track user activity, monitor application usage, and for debugging purposes in case of issues.

Every interaction with the app triggers logging mechanisms which capture detailed information such as the city requested, timestamp of the request, details of the API request made to OpenWeatherMap, and the corresponding response received. This data is essential for debugging, monitoring app performance, and analyzing user behavior trends over time.

4. Store the log information in a database

MongoDB Atlas Setup:

Connection String: Details of the MongoDB Atlas connection string (general format, do not include actual credentials).

Database and Collection: Logs are stored in the **weather_db** database and the **logs** collection.

All logs generated by the application are stored in MongoDB Atlas, which provides a robust and scalable cloud database solution.

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

Dashboard Implementation:

The **DashboardServlet** retrieves logs from MongoDB and displays them on a simple web page.

The dashboard provides insights such as request frequencies, most searched cities, etc.