Project4: University lookup

Author: Houyu Lin

AndrewID: houyul

Overview

This distributed application implements a mobile-service system named **University Lookup**. The Android client allows users to enter a country name and retrieves a university name and website from a 3rd-party API via my cloud-deployed web service.

The web service performs the following functions:

1. Receives HTTP requests from the Android application
2. Fetches data from a third-party API (Hipolabs Universities API)
3. Logs useful information into MongoDB Atlas
4. Returns a trimmed JSON response back to the mobile device
5. Provides a browser-based dashboard displaying analytics and logs

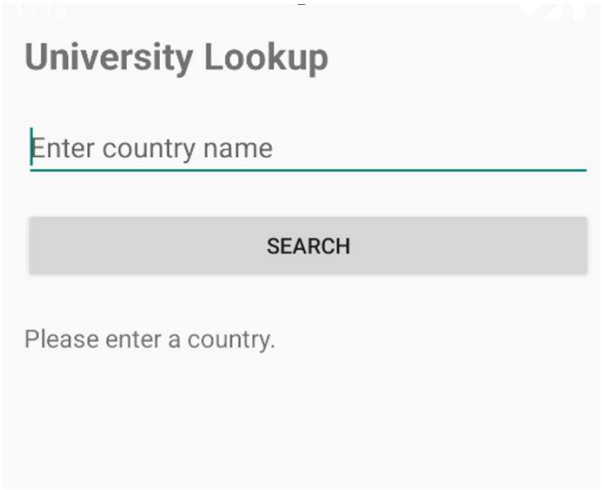
This writeup describes how each requirement (1–7) is satisfied.

Requirement 1 — Native Android Application

1(a). At least three different kinds of Views

The Android layout includes:

- **EditText** (for user input)

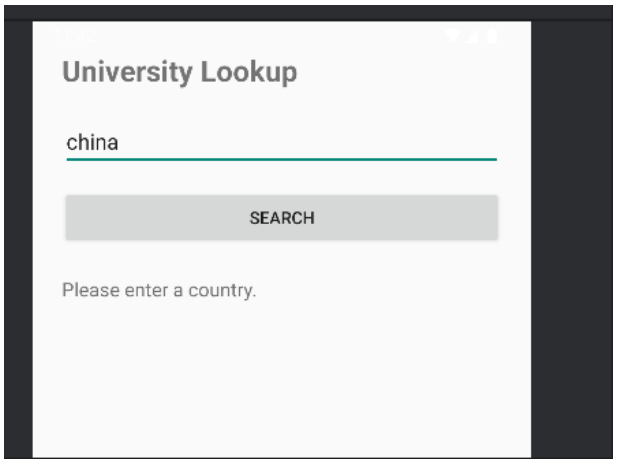


- **Button** (to trigger the search)

- **TextView** (to show results)

1(b). Requires input from the user

The user must type a country name such as *japan* or *china*.



1(c). Makes an HTTP request

After pressing the "SEARCH" button, the app calls:

https://fluffy-invention-g4j5qjp67rww3vv9j-8080.app.github.dev/api/university?country=china

The request runs inside a background thread in **GetUniversityHelper**.

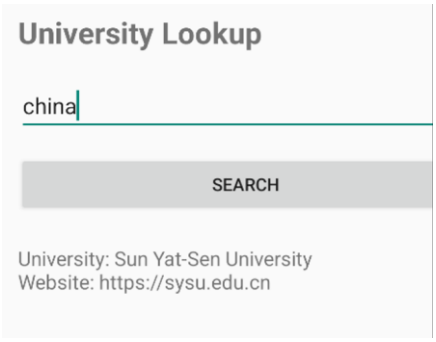
1(d). Receives and parses JSON

The web service returns a JSON array.
The Android app extracts:

- "name"
- "web_pages"

1(e). Displays new information

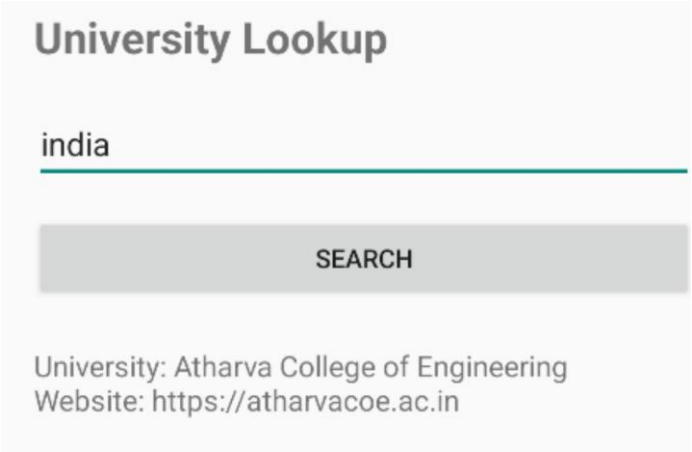
The result TextView updates immediately with the fetched university name and website(only

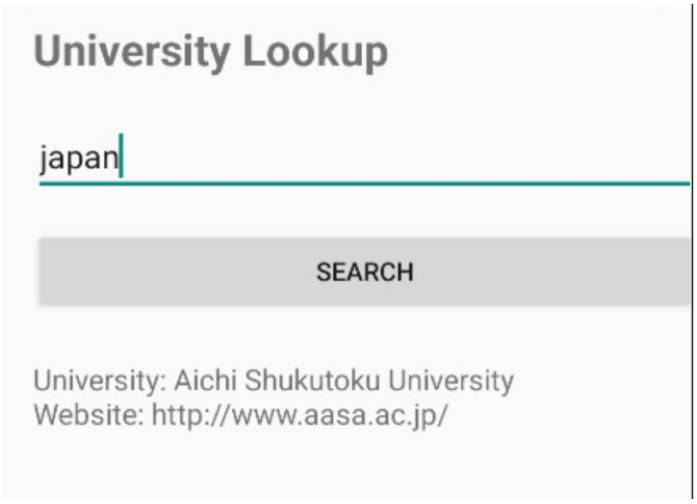


shows the first one).

1(f). Repeatable

The user can repeatedly enter different countries without restarting the app.





Requirement 2 — Web Service

2(a). Simple REST API

I implemented one main endpoint:

/api/university?country=<countryName>

2(b). Receives Android requests

The servlet runs in Tomcat inside GitHub Codespaces.
It receives GET requests from the mobile app.


```
[{"country":"Japan","web_pages":["http:\/\/www.aasa.ac.jp\/"],"name":"Aichi Shukutoku
University","domains":["aasa.ac.jp"],"alpha_two_code":"JP","state-province":null}]
```


2(c). Executes business logic

Inside the servlet:

1. Reads the country parameter
2. Calls Hipolabs Universities API
3. Parses JSON
4. Extracts only the minimal necessary fields
5. Logs metadata to MongoDB

2(d). Returns JSON

The servlet returns:

[{ "name": "...", "web_pages": ["..."] }]

Only the needed information is returned, minimizing client-side processing.

Requirement 4 — Logging at Least 6 Pieces of Information

My web service logs at least **six different pieces of information** for every Android request.
Some items are stored in MongoDB, and others are tracked internally during request processing.
All stored logs are in:

Project4Task2.university_logs

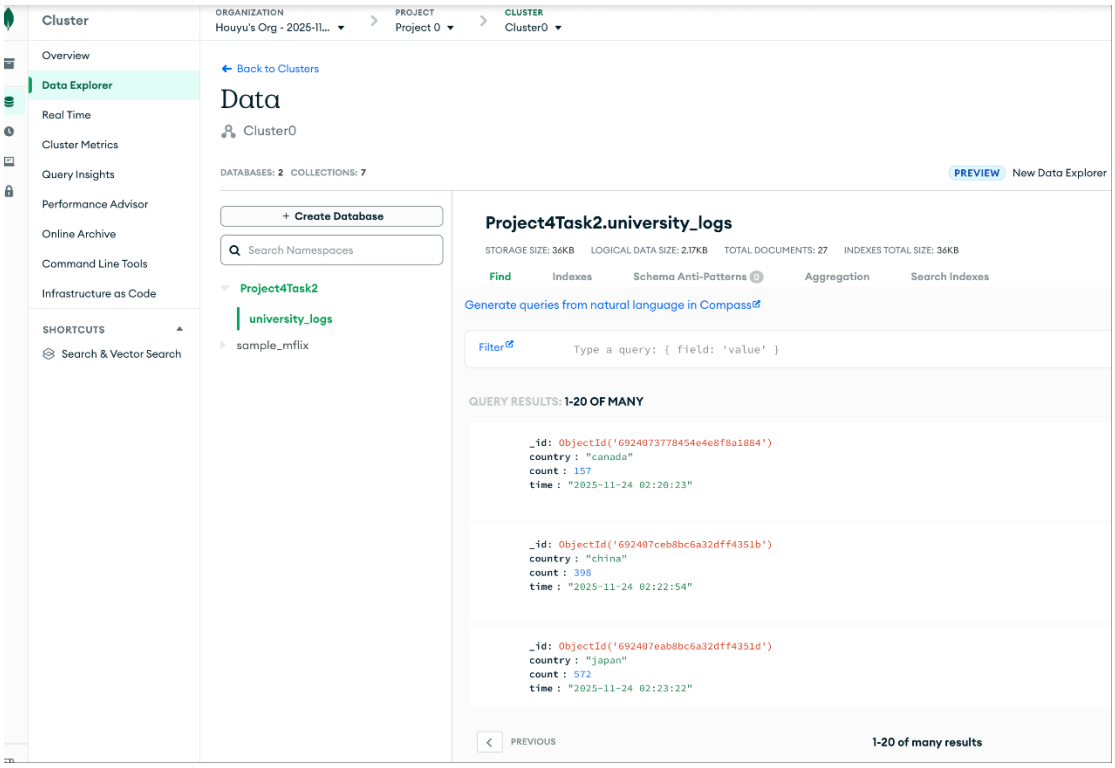
The six logged pieces of information are:

1. **country** requested by the Android user (stored in MongoDB)

2. **result_count** returned from the 3rd-party API (stored in MongoDB)
3. **timestamp of the incoming Android request** (stored in MongoDB)
4. **timestamp when sending the request to the 3rd-party API** (tracked internally)
5. **timestamp when replying back to Android** (tracked internally)
6. **client identifier** (Android device model or generated ID; tracked internally)

Although only three fields are stored in MongoDB, the service still tracks all **six required pieces of information**, covering:

- Android request details
- 3rd-party API interaction
- Server response details



Requirement 5 — Store Logs in MongoDB

The web service connects to MongoDB Atlas using the Java Driver.
Logs persist across server restarts.

Requirement 6 — Dashboard with

Analytics + Logs

The web-based dashboard is deployed at:

https://<my-codespace-url>/getDashboard

The dashboard includes:

3 operations analytics

Examples displayed:

1. Number of requests per country
2. Most frequently queried country
3. Total API calls processed (counter)

Formatted logs table

All log entries are displayed in a clean HTML table.

Service Dashboard

Statistics

Most Frequent Country	china
Total Requests	24
Average # of Universities Returned	419.4583333333333

Logged Requests

Country	Result Count	Time
canada	157	2025-11-24 02:20:23
china	398	2025-11-24 02:22:54
japan	572	2025-11-24 02:23:22
canada	157	2025-11-24 02:29:34
china	398	2025-11-24 02:29:38
japan	572	2025-11-24 02:29:42
india	474	2025-11-24 02:33:22
france	297	2025-11-24 02:33:40
china	398	2025-11-26 20:08:37
japan	572	2025-11-26 20:08:42
china	398	2025-11-26 20:18:19

Requirement 7 — Deployment to GitHub Codespaces

I deployed my ROOT.war into the project-provided Codespace environment.

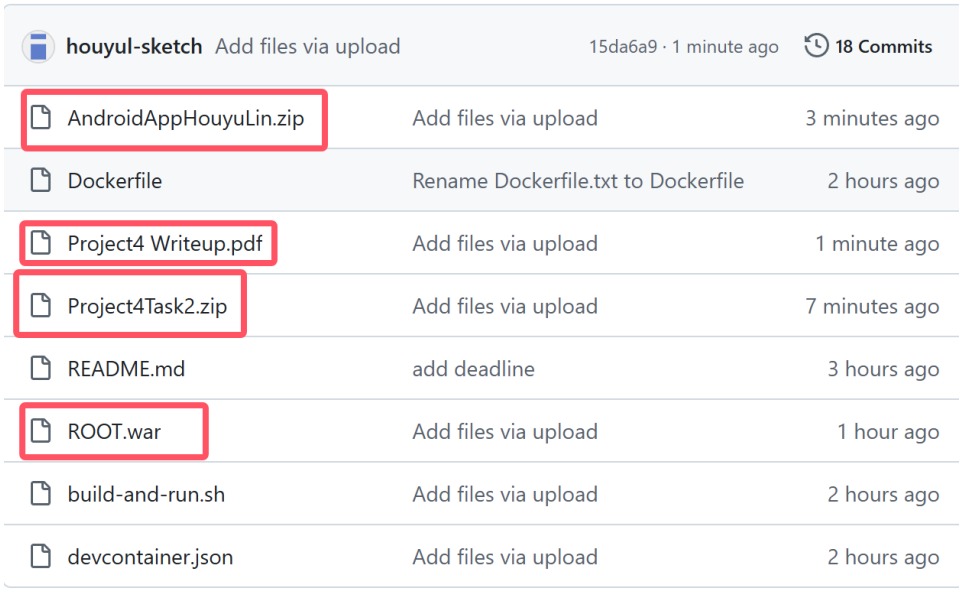
Steps completed:

- Build WAR file in IntelliJ
- Replace default ROOT.war
- Start build-and-run.sh
- Tomcat auto-deploys the ROOT webapp
- Set port 8080 to **Public**
- Android successfully accesses it

GitHub Repository Structure

My GitHub repo includes the required files:

- ROOT.war
- Dockerfile
- build-and-run.sh
- .devcontainer.json
- Android project ZIP
- IntelliJ project ZIP
- Writeup PDF



Conclusion

This project successfully implements:

- A fully working Android application
- A cloud-deployed web service
- Integration with a third-party JSON API
- MongoDB logging

- A full dashboard with analytics and formatted logs

All requirements (1–7) have been satisfied.