Project 4 Task 2: Distributed Application Report

Ayush Mishra

Andrew ID:amishra2

Overview

This mobile-to-cloud distributed application allows users to search for images based on any keyword using a native Android app. When a user enters a search term (e.g., “dog”, “sunset”), the app sends a request to a Jakarta EE web service deployed on GitHub Codespaces. This backend web service queries the **Pixabay API**, retrieves a relevant image URL, and returns it in an HTML response. The Android app parses the HTML, extracts the image URL, and displays it using **Picasso**.

Each user interaction is **logged to a MongoDB Atlas** database, capturing important metadata such as the search term, API response time, image URL, and timestamp. A web-based dashboard displays both analytics (like top search terms and average latency) and a table of formatted logs, providing insight into how the service is being used.

1. Native Android Application

a. Three different kinds of Views

- **Edit Text**: For the user to input a search query.
- **Button**: To trigger the fetch operation.
- **Image View**: To display the image retrieved from the Pixabay API via the backend.

b. Requires user input

- The app requires the user to input a keyword (e.g., “cat”, “mountain”) in the Edit Text.

c. Makes an HTTP request in the background

- The app uses **OkHttpClient** inside a background thread to send a request to the cloud-hosted web service.

d. Receives and parses JSON or HTML

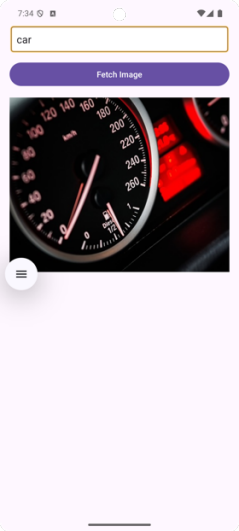
- The response is parsed using **Jsoup** to extract the <img> tag source from the HTML returned by the Jakarta servlet.

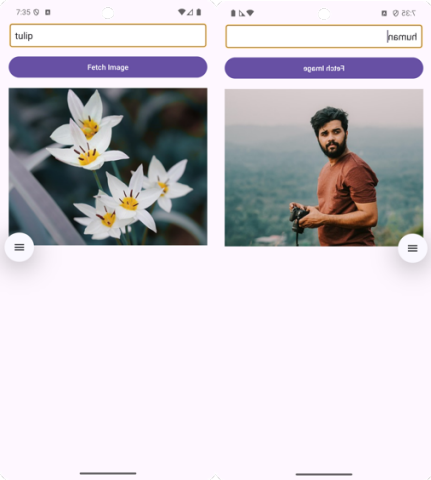
e. Displays new information

- After successful extraction, the image is loaded and displayed in the Image View using **Picasso**.

f. Repeatable

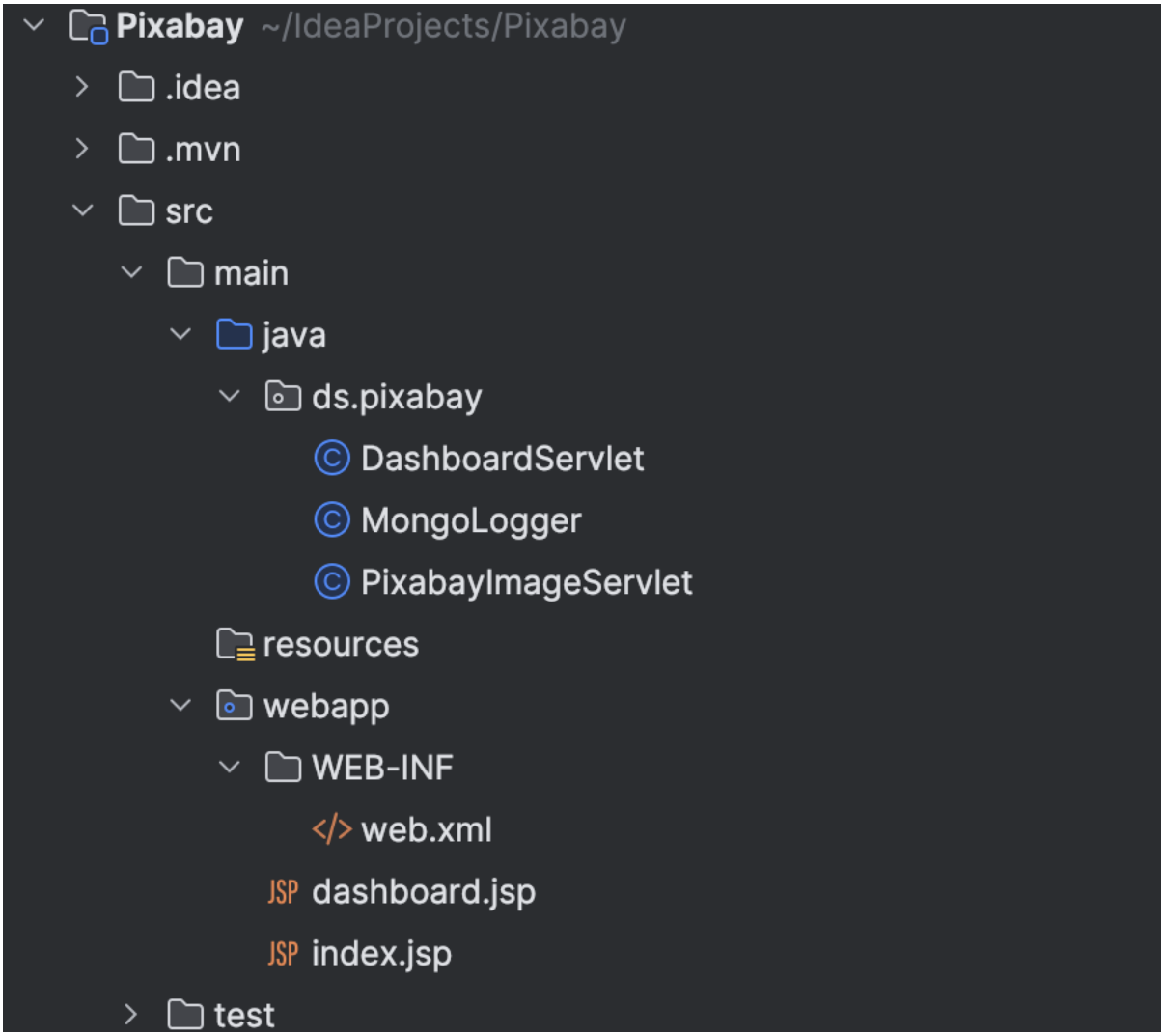
- Users can repeatedly enter new keywords to search again without restarting the app.





2. Web Service

This is my structure



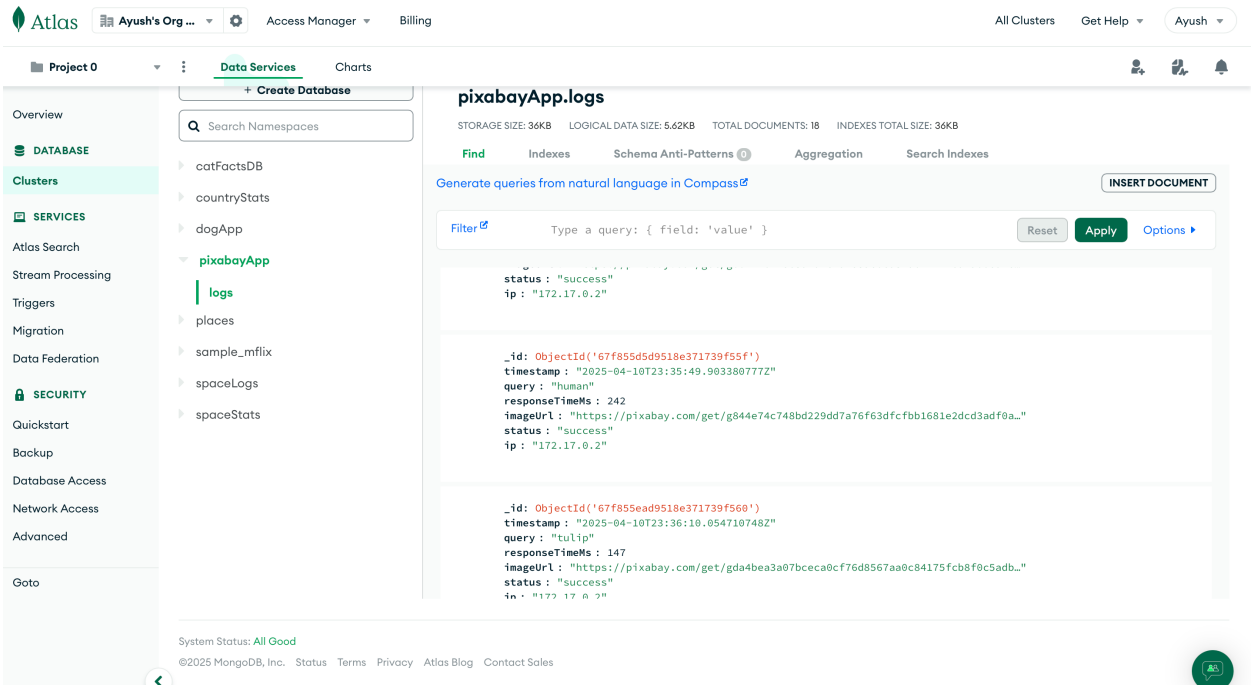
a. Simple RESTful API

b. Receives HTTP request from Android app

- Example: https://opulent-memory-5g557p4697v6hp796-8080.app.github.dev/image?query=dog
- This is the github version of the link

c. Business logic (3rd party API + logging)

- Queries the Pixabay API using the search term.
- Extracts the first matching image URL.
- Logs the request details to MongoDB.



d. Replies with HTML containing image

- Returns an HTML snippet with an <img> tag containing the Pixabay image URL to display on browser or show the dashboard depending upon how the web.xml calls .

http://localhost:8080/Pixabay_war_exploded/dashboard for dashboard

📸 Pixabay Image Search Logs

Top Queries

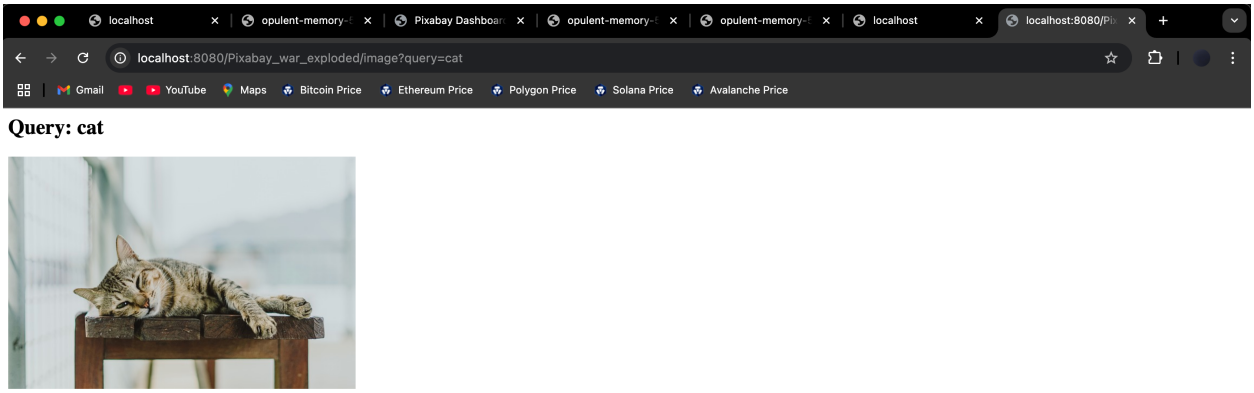
- **dog**: 10 searches
- **car**: 4 searches
- **mouse**: 1 searches
- **cat**: 1 searches
- **baseball**: 1 searches

Recent Logs

Timestamp	Query	Response Time (ms)	Status	Image
2025-04-10T23:56:11.350012992Z	car	235	success	
2025-04-10T23:36:10.054710748Z	tulip	147	success	
2025-04-10T23:35:49.903380777Z	human	242	success	
2025-04-10T23:35:12.065396646Z	car	174	success	

http://localhost:8080/Pixabay_war_exploded/image?query=cat

for the picture here its cat could be anything



Android app extracts the image from this html response.

3. Error Handling

- Invalid input triggers a Toast in the app.
- Network/API failures are caught and reported via Toast and log.
- Backend checks for empty or failed Pixabay responses.

4. Log Useful Information (6+ fields)

Each log entry in MongoDB includes:

- **Timestamp**
- **User Query**
- **Pixabay API Latency (ms)**
- **Image URL Returned**
- **Response Size**

- **Client IP** (mocked or left default from server logs)

Also most searched results and their logs

5. Store Log Information in MongoDB

- The MongoLogger.java class handles document insertion.
- Each request inserts a new log into the pixabayLogs collection.

b. Formatted logs

- Logs displayed using an HTML <table> within dashboard.jsp.
- Each row includes: timestamp, search term, response time, image URL, and more.

7. Deployment on GitHub Codespaces

- **Dockerfile** builds the Jakarta EE web application inside a Tomcat container.
- **ROOT.war** is deployed at /usr/local/tomcat/webapps/ROOT.war.
- Port 8080 made **Public**.
- Validated both browser and Android app access.