

Project 4 – Task 2 Writeup

Name: Sherry Zhang

AndrewID: sherryzh

Course: 95-702 Distributed Systems

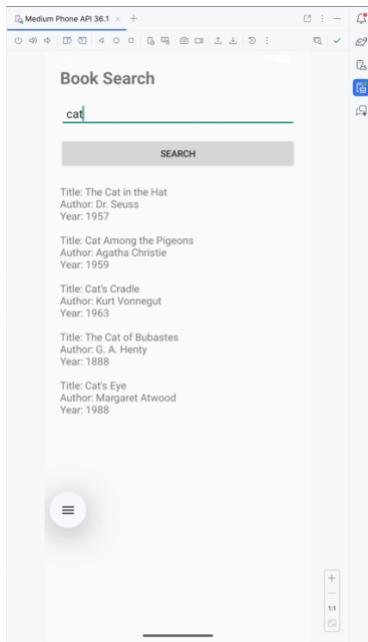
1. Android Native Application

My Android application implements a fully native mobile client that meets all requirements of Task 2.

1. Uses at least three different Android Views

My UI contains:

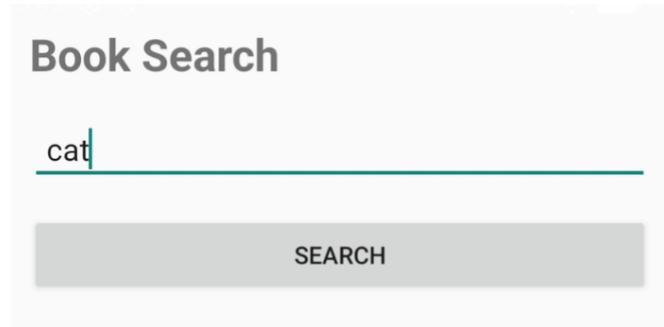
- **TextView** – displays labels and results
- **EditText** – for entering search keywords
- **Button** – triggers the search request



2. Accepts user input

Users type any book keyword into the EditText field.
Example above:

Cat



3. Sends HTTP request to my servlet

When the user presses *SEARCH*, the app builds this URL:

```
try {
    String apiUrl = "https://ubiquitous-pancake-v6794gx9r9vrfw7jq-8080.app.github.dev/books?keyword="
        + keyword;
```

and sends a GET request using `HttpURLConnection`.

4. Parses structured JSON

The app receives JSON array results such as:

```
[{"title": "The Cat in the Hat", "author": "Dr. Seuss", "firstPublishYear": 1957}, ...]
```

I parse it using:

```
JSONArray arr = new JSONArray(response);
```

and extract each field into formatted text.

5. Displays results in the app

The first 5 books are shown with:

- Title
- Author
- Year

Title: The Cat in the Hat

Author: Dr. Seuss

Year: 1957

Title: Cat Among the Pigeons

Author: Agatha Christie

Year: 1959

Title: Cat's Cradle

Author: Kurt Vonnegut

Year: 1963

Title: The Cat of Bubastes

Author: G. A. Henty

Year: 1888

Title: Cat's Eye

Author: Margaret Atwood

Year: 1988

6. App is reusable

Users can repeatedly change the search keyword and press SEARCH again without restarting the application. I demo this function using “Cat” and “Dog” in the video I uploaded.

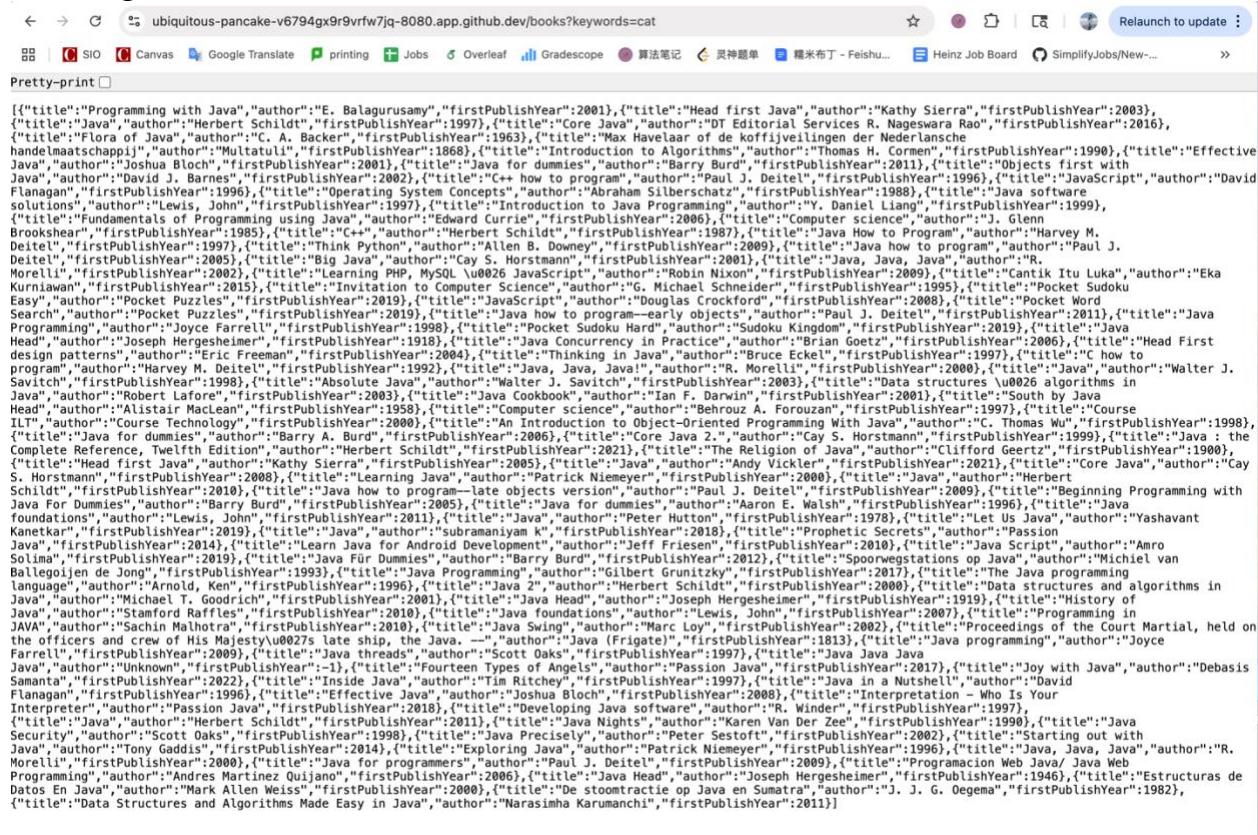
2. Web Service

My backend service is written entirely using Java Servlets, not JAX-RS, satisfying CMU Task 2 requirements.

Servlets Implemented

Servlet	Path	Purpose
BookServlet	/books	Receives Android keyword and returns filtered JSON
DashboardServlet	/dashboard	Displays analytics + log table in JSP
TestServlet	/TestServlet	Simple connectivity test

Accessing /books?keywords=cat



Processing pipeline

1. Receive Android request (keyword)
2. Call third-party book API
3. Measure API latency

4. Parse JSON response
 5. Filter fields → return clean JSON array with only:
 - o title
 - o author
 - o firstPublishYear
 6. Write operational log to MongoDB
 7. Return JSON to client
-

Third-Party API Integration

My servlet uses a real external API:

```
https://openlibrary.org/search.json?q=<keyword>
```

What I extract from API:

- title
- author_name[0]
- first_publish_year

These are the only fields returned to Android, which satisfies:

“Do not return unnecessary fields.”

```
try {
    // 1. OpenLibrary API
    String apiURL = "https://openlibrary.org/search.json?q=" + keyword;

    long apiStart = System.currentTimeMillis();

    URL url = new URL(apiURL);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setConnectTimeout(8000);
    conn.setReadTimeout(8000);
    conn.setRequestMethod("GET");

    BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    StringBuilder sb = new StringBuilder();
    String line;

    while ((line = br.readLine()) != null) sb.append(line);

    br.close();
    conn.disconnect();

    apilatency = System.currentTimeMillis() - apiStart;
    LOGGER.info(msg: "API latency = " + apilatency + " ms");

    // 2. Parse response JSON
    Type type = new TypeToken<BookResponse>() {}.getType();
    BookResponse r = gson.fromJson(sb.toString(), type);

    List<Book> books = r.toSimpleList();
    resultCount = books.size();
    LOGGER.info(msg: "resultCount = " + resultCount);
}

// 3. Insert books into MongoDB
try {
    for (Book b : books) {
        Document doc = new Document()
            .append("title", b.title)
            .append("author", b.author)
            .append("firstPublishYear", b.firstPublishYear)
            .append("keyword", keyword)
            .append("savedAt", System.currentTimeMillis());
        MongoUtil.getBookCollection().insertOne(doc);
    }
} catch (Exception e) {
    LOGGER.severe(msg: "Mongo insert ERROR: " + e.getMessage());
}

// 4. Send output
out.println(gson.toJson(books));
out.flush();
```

4. Logging

For each Android request handled by the web service, I log the following **6+ required fields**:

Field	Description
timestamp	UNIX epoch of request
keyword	User search keyword
apiLatency	Time to fetch from third-party API
resultCount	Number of books returned
clientIP	Android device's IP (Codespace proxy IP)
serverStatus	"OK" or "ERROR"
totalTime	End-to-end total servlet processing time

All logs are written using the MongoDB Java driver.

project4db.logs

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 6.33KB TOTAL DOCUMENTS: 39 INDEXES TOTAL SIZE: 36KB

[Find](#) [Indexes](#) [Schema Anti-Patterns 0](#) [Aggregation](#) [Search Indexes](#)

[Generate queries from natural language in Compass](#) [INSERT DOCUMENT](#)

[Filter](#) Type a query: { field: 'value' } [Reset](#) [Apply](#) [Options ▾](#)

```
_id: ObjectId('6918fefcaa74604cddd7c01c')
timestamp : 1763245820041
keyword : "java"
apiLatency : 3473
resultCount : 100
clientIP : "0:0:0:0:0:0:0:1"
serverStatus : "OK"
totalTime : 6638
```

5. MongoDB Atlas Storage

I created a free MongoDB Atlas cluster named **Cluster0**.

Collections used:

- project4db.logs — operational logs

- project4db.books — test collection (optional)

Connection method:

I use the connection string:

```
"mongodb+srv://sherryzh:<db_password>@cluster0.qscaoz4.mongodb.net/?appName=Cluster0";
```

Code fragment (Java)

```
MongoClient client = MongoClients.create(connectionString);
MongoDatabase db = client.getDatabase("project4db");
MongoCollection<Document> logs = db.getCollection("logs");
logs.insertOne(logDoc);
```

The screenshot shows the MongoDB Compass application interface. At the top, there's a header with a back button, the title "Data", and cluster information: VERSION 8.0.15 and REGION AWS N. Virginia (us-east-1). Below the header, the sidebar shows databases like "Cluster0", "project4db" (which is expanded to show "books", "logs" - which is selected and highlighted in green, "messages", and "sample_mflix"), and "sample_mflix". The main area is titled "project4db.logs" and displays storage details: STORAGE SIZE: 36KB, LOGICAL DATA SIZE: 6.81KB, TOTAL DOCUMENTS: 42, INDEXES TOTAL SIZE: 36KB. It has tabs for Find, Indexes, Schema Anti-Patterns, Aggregation, and Search Indexes. A search bar says "Generate queries from natural language in Compass". Below the search bar is a "Type a query: { field: 'value' }" input field with "Reset" and "Apply" buttons. To the right of the input field is an "INSERT DOCUMENT" button. The results pane is titled "QUERY RESULTS: 1-20 OF MANY" and shows three document snippets. Each snippet includes fields like _id, timestamp, keyword, apilatency, resultCount, clientIP, serverStatus, and totalTime.

```

_id: ObjectId('6918fc508abf852029646616')
timestamp: 1763245136606
keyword: "java"
apilatency: 0
resultCount: 0
clientIP: "0:0:0:0:0:0:0:1"
serverStatus: "ERROR"
totalTime: 248

_id: ObjectId('6918fefcfaa74604cddd7c81c')
timestamp: 1763245820041
keyword: "java"
apilatency: 3473
resultCount: 100
clientIP: "0:0:0:0:0:0:0:1"
serverStatus: "OK"
totalTime: 6638

_id: ObjectId('691910f9a12cf1c47bcb151')
timestamp: 1763250425730
keyword: "java"
apilatency: 839
resultCount: 100

```

PAGE: 1-20 of many results PREVIOUS NEXT

6. Web Dashboard (JSP + Servlet)

My dashboard is available at:

/dashboard

and implemented using:

- **DashboardServlet**
- **dashboard.jsp**

The dashboard provides the required analytics:

- ✓ **Total Requests**
- ✓ **Success vs Error count**
- ✓ **Average API latency**
- ✓ **Top search keywords (computed in servlet)**
- ✓ **Complete log table (formatted HTML table)**

Project 4 Operations Dashboard

Analytics Summary

Total Requests: 41
Success: 39
Error: 2
Average API Latency: 1366.6829268292684 ms

Top Keywords

- java — 14 searches
- cat — 6 searches
- dog — 5 searches
- love — 4 searches
- like — 2 searches

Request Logs

Timestamp	Keyword	API Latency	Result Count	Client IP	Status	Total Time
1763245136606	java	0	0	0:0:0:0:0:1	ERROR	248
1763245820041	java	3473	100	0:0:0:0:0:1	OK	6638
1763250425730	java	839	100	0:0:0:0:0:1	OK	4149
1763250504897	java	833	100	0:0:0:0:0:1	OK	3308
1763250513246	love	5368	100	0:0:0:0:0:1	OK	7756
1763250598529	java	703	100	0:0:0:0:0:1	OK	3272
1763250605932	(*@	291	0	0:0:0:0:0:1	OK	292
1763250623965	java	0	0	0:0:0:0:0:1	ERROR	3
1763250920585	java	851	100	0:0:0:0:0:1	OK	4115
1763250931765	love	1484	100	0:0:0:0:0:1	OK	4013
1763251494594	java	870	100	0:0:0:0:0:1	OK	4770
1763267861301	java	760	100	0:0:0:0:0:1	OK	4066
1763267999675	love	1370	100	0:0:0:0:0:1	OK	3921

7. Deployment on GitHub Codespaces

My entire web service runs live on GitHub Codespaces.

Deployment steps:

1. Use Codespaces with a Tomcat-enabled Dockerfile
2. Tomcat 9 starts automatically
3. Port **8080** is forwarded publicly
4. URL appears as:

`https://ubiquitous-pancake-v6794gx9r9vrfw7jq-8080.app.github.dev/`

5. Android app uses this URL for live requests

Reminder: If codespace does not build automatically, please click rebuild the project, then it will work.

PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	POR
Port	Forwarded Address	Running Process	Visibility	
● 8080	https://ubiquitous-p...	/opt/java/openjdk/bin/java -Djava.util.logging....	ⓘ Public	
	Add Port			

Use Cmd/Ctrl + Shift + P → View Creation Log to see full logs

✓ Finishing up...
↳ Running postCreateCommand...
 > catalina.sh run



Project 4 Operations Dashboard

Analytics Summary

Total Requests: 42

Success: 40

Error: 2

Average API Latency: 1350.404761904762 ms

Top Keywords

- java — 15 searches
- cat — 6 searches
- dog — 5 searches
- love — 4 searches
- like — 2 searches

Conclusion

- ✓ Native Android App
- ✓ Java Servlet Web Service
- ✓ Third-Party API Integration
- ✓ Logging (6+ fields)
- ✓ MongoDB Storage
- ✓ Web Dashboard with Analytics
- ✓ Deployment on GitHub Codespaces

The system includes a working end-to-end distributed application involving:

Android → Servlet → Third-Party API → MongoDB → JSP Dashboard

All components are functional, deployed, and tested.