Project 4 Write-Up – AniSearch
Patrick Serrano (AndrewID: pserrano)

Description
My application takes a search string from the user and uses it to fetch and display detailed anime data from the Jikan API. Additionally, it will display a catalogue of top anime and top seasonal anime.

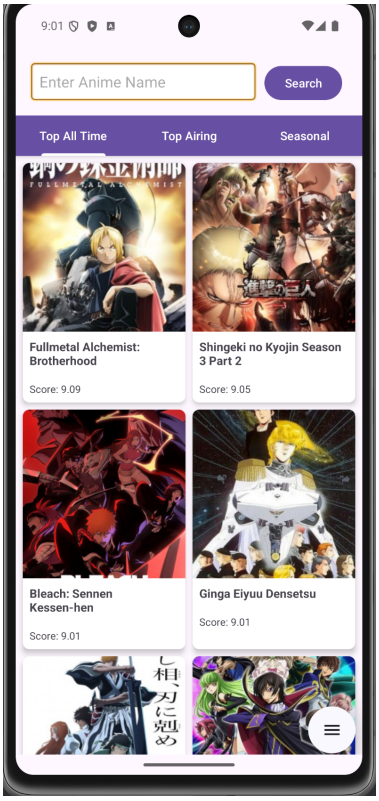
Here is how my application meets the task requirements:

1. Implement a native Android application

The name of my native Android application project in Android Studio is: Task2Application.

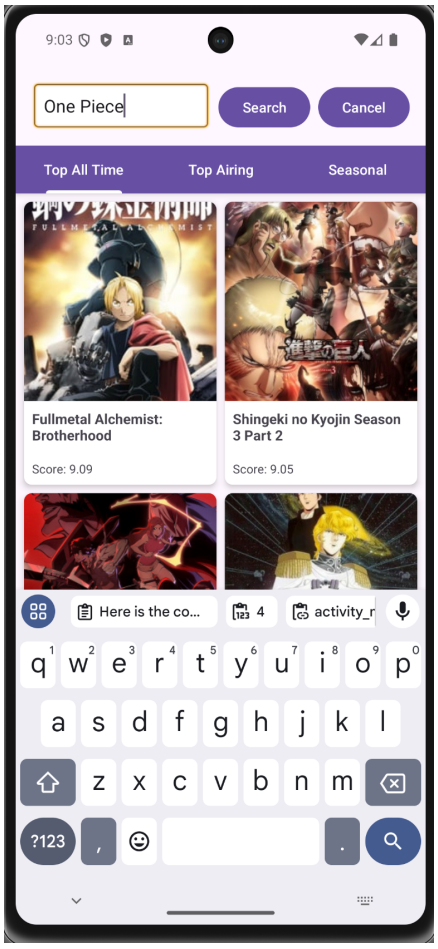
a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, etc.)

My application uses TextView, EditText, Button, ImageView, and RecyclerView. See activity_main.xml and item_anime.xml for details of how they are incorporated into the layout.
Screenshot:



b. Requires input from the user
The application requires the user to enter a search term in the EditText field to fetch anime information.

Screenshot:



c. Makes an HTTP request (using an appropriate HTTP method) to your web service
The application performs an HTTP GET request in MainActivity.java. The HTTP request is sent to my web service's /anime/search endpoint.

Example HTTP request:
https://api.jikan.moe/v4/anime?q=OnePiece&rating=g&rating=pg&rating=pg13&rating=r17

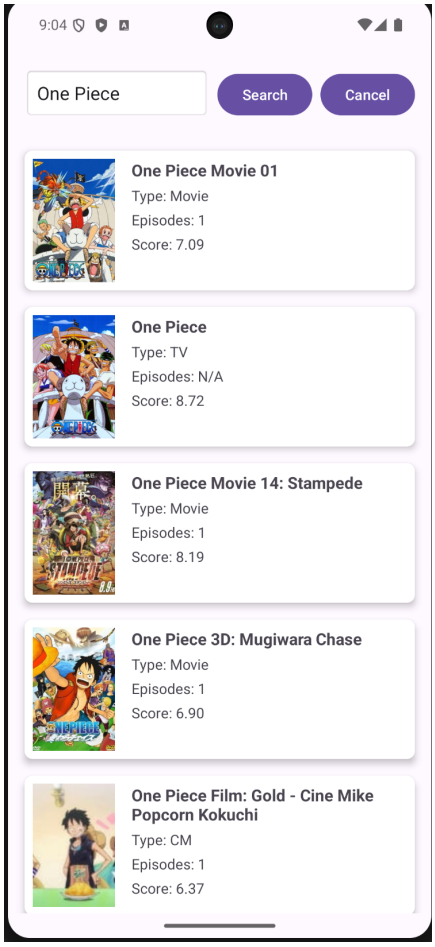
d. Receives and parses an XML or JSON formatted reply from the web service
The application processes JSON responses using the org.json library to parse data.

Example JSON reply:


```json
{
 "data": [
  {
   "mal_id": 1,
   "title": "Cowboy Bebop",
   "type": "TV",
   "episodes": 26,
   "score": 8.75,
   "status": "Finished Airing",
   "images": {
    "jpg": {
     "image_url": "https://cdn.myanimelist.net/images/anime/4/19644.jpg"
    }
   },
   "synopsis": "..."
  }
 ],
 "pagination": {
  "last_visible_page": 1,
  "has_next_page": false,
  "current_page": 1,
  "items": {
   "count": 1,
   "total": 1,
   "per_page": 25
  }
 }
}
```

e. Displays new information to the user
The application displays a list of anime search results with titles, types, episode counts, scores, and cover images in a RecyclerView. Clicking on a result opens the AnimeDetailActivity showing more information.

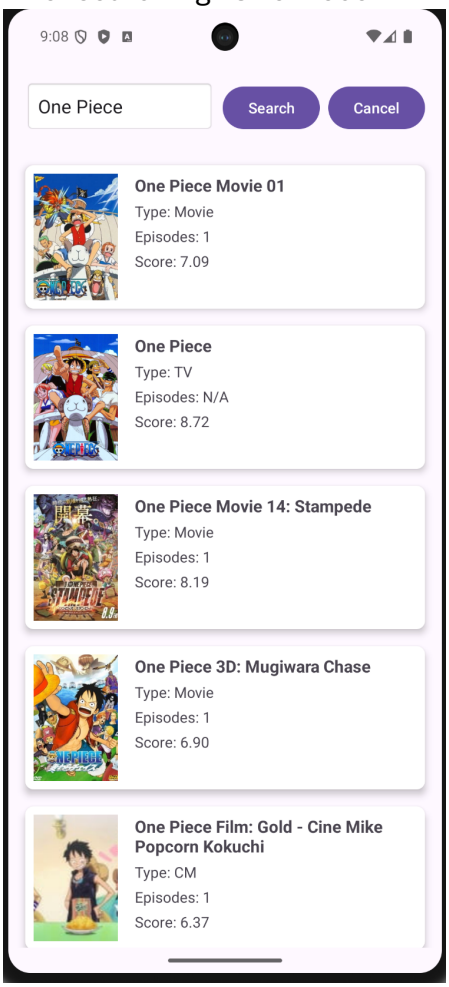
Screenshot:



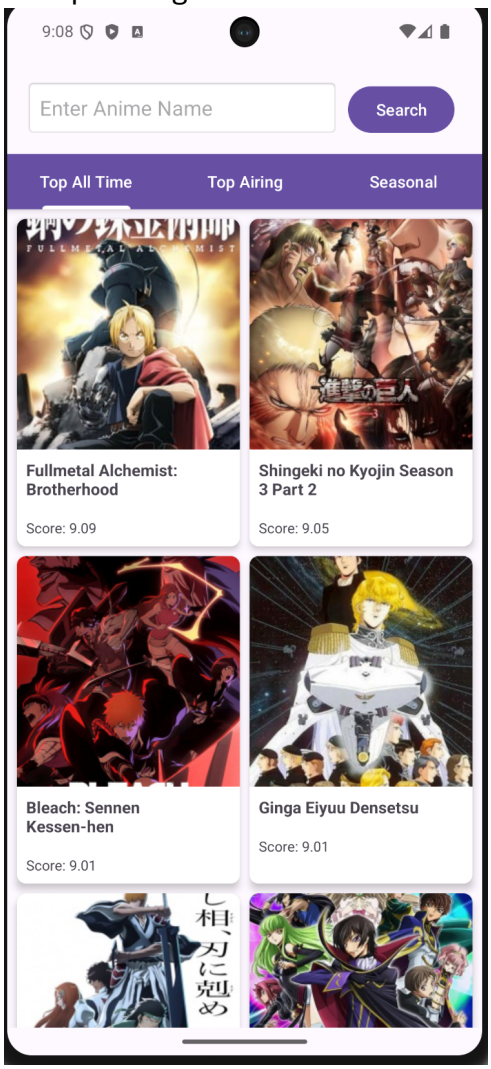
f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)
The application allows users to enter new search terms and fetch updated results without needing to restart. They can also navigate back from the details screen to the search.

Screenshot:

After searching “One Piece”:



After pressing Cancel:



Back Button to exit DetailedView:

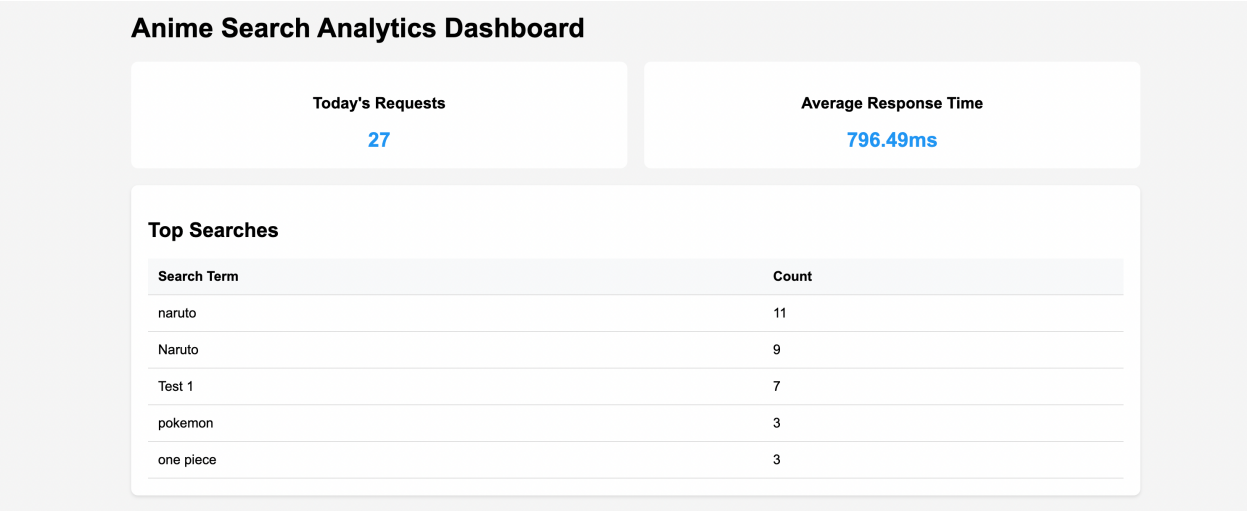
2. Implement a web service
The web service project directory name is task2server.
It is implemented as a Java web application using Jakarta Servlets.
It is deployed on Github Codespaces.

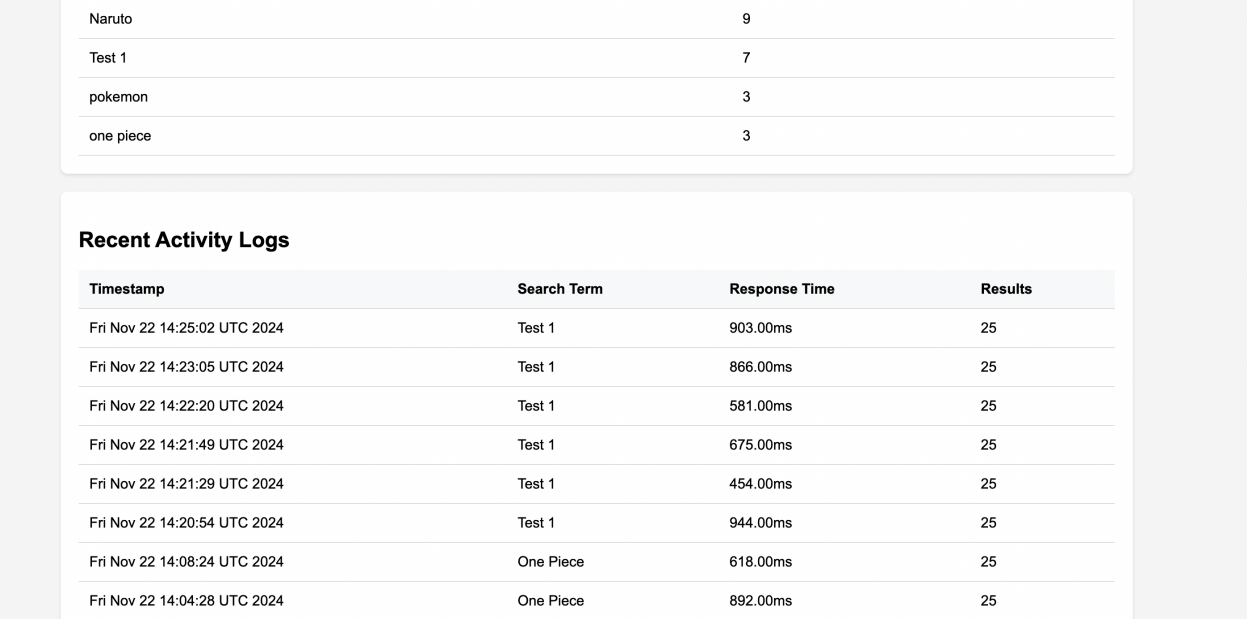
The Url is:
https://didactic-engine-vjjv6764xxfx6v7-8080.app.github.dev/anime/dashboard

a. Implement a simple (can be a single path) API using Servlets

The AnimeWebService class extends HttpServlet to provide API endpoints for searching anime, fetching top/seasonal anime, and displaying analytics.
Key endpoints:
- /anime/search
- /anime/top
- /anime/seasonal
- /anime/details
- /anime/dashboard

Model: AnimeModel.java
View: dashboard.jsp
C: AnimeControler.java





Naruto	9
Test 1	7
pokemon	3
one piece	3

Recent Activity Logs

Timestamp	Search Term	Response Time	Results
Fri Nov 22 14:25:02 UTC 2024	Test 1	903.00ms	25
Fri Nov 22 14:23:05 UTC 2024	Test 1	866.00ms	25
Fri Nov 22 14:22:20 UTC 2024	Test 1	581.00ms	25
Fri Nov 22 14:21:49 UTC 2024	Test 1	675.00ms	25
Fri Nov 22 14:21:29 UTC 2024	Test 1	454.00ms	25
Fri Nov 22 14:20:54 UTC 2024	Test 1	944.00ms	25
Fri Nov 22 14:08:24 UTC 2024	One Piece	618.00ms	25
Fri Nov 22 14:04:28 UTC 2024	One Piece	892.00ms	25

b. Receives an HTTP request from the native Android application

The doGet method in AnimeWebService handles the incoming HTTP GET requests from the Android app, such as the /anime/search endpoint with the q parameter.

c. Executes business logic appropriate to your application.

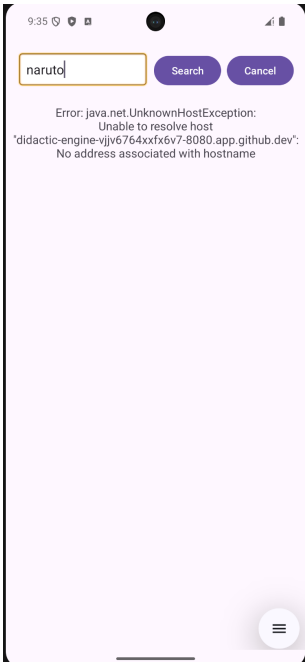
The servlet methods call the Jikan API with the appropriate URL to fetch the requested anime data. The responses are parsed from JSON. The requests are logged to a MongoDB database including data like search terms, response times, result counts, etc.

d. Replies to the Android application with an XML or JSON formatted response.

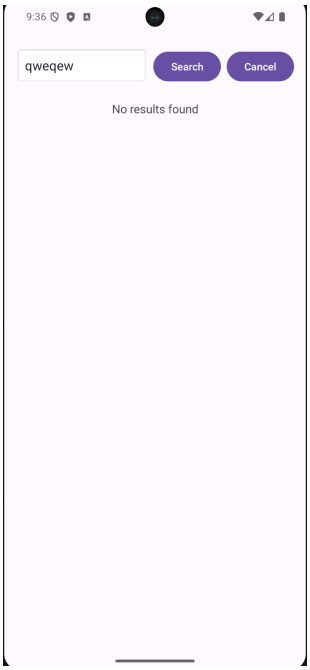
The fetched anime data is passed through mostly as-is in JSON format to the Android application. Some filtering is done to remove entries with age ratings above R17+.

3. Handle error conditions

Both the Android app and web service implement error handling:
- The Android app displays user-friendly error messages if the search fails due to network issues or if no results are found.
- The web service sends appropriate HTTP status codes (400 for invalid requests, 500 for server errors) and JSON error responses. Errors are logged.



Doesn’t Crash if internet is off.



Displays message when results not found.

4. Log useful information

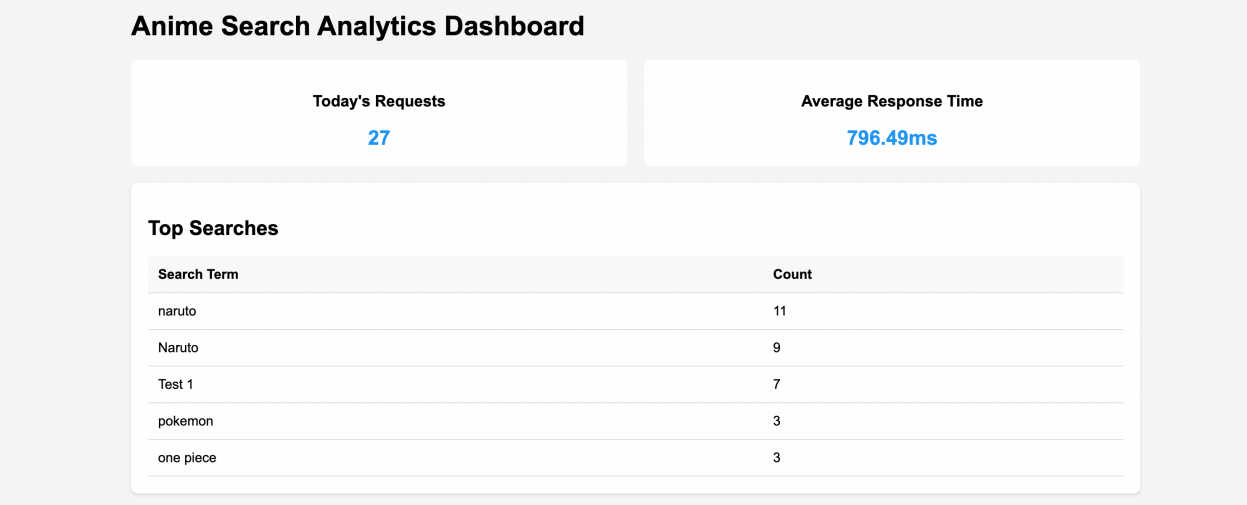
The following information is logged to MongoDB for each request:
- timestamp: When the request was received
- responseTime: Time taken in milliseconds
- searchTerm: Search keyword (if applicable)
- endpoint: The full URL requested
- clientIP: IP address of the Android client
- resultCount: Number of anime returned

5. Store the log information in a database
Logs are stored in a MongoDB database "project4db" in an "anime_logs" collection.

String connectionString =
"mongodb+srv://pserrano:Dismproj1234@dism.zanxk.mongodb.net/?retryWrites=true&w=majority&appName=DISM";

6. Display operations analytics and full logs on a web-based dashboard
The dashboard endpoint /anime/dashboard displays:
- Total number of requests today
- Average response time across all endpoints
- Most searched anime keywords
- Recent request logs
- Graphs for average response times and request type breakdown

Screenshot:



Naruto	9
Test 1	7
pokemon	3
one piece	3

Recent Activity Logs

Timestamp	Search Term	Response Time	Results
Fri Nov 22 14:25:02 UTC 2024	Test 1	903.00ms	25
Fri Nov 22 14:23:05 UTC 2024	Test 1	866.00ms	25
Fri Nov 22 14:22:20 UTC 2024	Test 1	581.00ms	25
Fri Nov 22 14:21:49 UTC 2024	Test 1	675.00ms	25
Fri Nov 22 14:21:29 UTC 2024	Test 1	454.00ms	25
Fri Nov 22 14:20:54 UTC 2024	Test 1	944.00ms	25
Fri Nov 22 14:08:24 UTC 2024	One Piece	618.00ms	25
Fri Nov 22 14:04:28 UTC 2024	One Piece	892.00ms	25

7. Deploy the web service to GitHub Codespaces
The web service .war file and a Dockerfile is included in a GitHub repo. The Dockerfile copies the .war into a Tomcat image for deployment. When a Codespace is created from the template, the postCreateCommand builds and deploys the app.

</document_content>
</document></documents>