Adeleye Oriola Mesogboriwon
Amesogbo

Short explanation on how my codes satisfy the requirements of this project.
Kindly let me know which one does not after you have performed an operation or is missing from my codes.

1. Implement a Native Android Application

a. Three kinds of Views
TextView (textViewTitle, textViewArtist)

EditText (editTextQuery)

ImageView (imageViewArt)

Button (buttonSearch)

All of these extend android.view.View.

b. Requires input from the user
The user inputs a search term via the EditText.

c. Makes an HTTP request on a background thread
Uses ExecutorService and Handler to make a background HTTP GET request to your web service.

d. Receives and parses XML or JSON
Parses a JSON response using JSONObject and JSONArray from your web service.

e. Displays new information to the user
Dynamically updates the song title, artist name, and artwork using the data from the response.

f. Is repeatable
The user can input new queries as often as they like without restarting the app.

2. Implement a Web Service

a. Simple API endpoint
/api/music is a single, clean RESTful path.

b. Receives HTTP request from the Android app
The Android app hits /api/music?query=... using GET.

c. Executes business logic
Fetches from the iTunes API and processes the JSON to forward only the relevant data.

You are using a published JSON API (iTunes), not screen scraping.

d. Replies in JSON format
Returns only the needed fields: trackName, artistName, and artworkUrl100.

The response is optimized and does not include extra data.

The service is implemented using Servlets, not JAX-RS.

3. Handle Error Conditions
Input validation is done on the Android app to ensure a query is provided.

The Android app handles network failures and displays an error message using Toast.

The server checks for invalid or empty inputs and handles them.

If the third-party iTunes API is unavailable or returns invalid data, the app shows default values or an error.

Exceptions in the servlet return a JSON error object with a message.

4. Log Useful Information
You log at least six useful pieces of data:

timestamp

clientIp

externalApiUrl

resultCount

processingTime

Extracted query term (used in analytics)

This information covers the client request, third-party API request, and server reply.

5. Store the Log Information in a Database
Log information is stored in a MongoDB collection (logs) within the task2db database.

The connection is made using MongoClients.

6. Display Operations Analytics and Logs on Dashboard
a. Unique URL for dashboard
The dashboard is available at /dashboard, routed through a servlet.

b. Displays 3 operations analytics
Total number of requests

Average processing time

Top search queries and their frequency

c. Displays formatted full logs
Full logs are displayed in an HTML table (not as JSON or XML), showing:

Timestamp

Client IP

External API URL

Result Count

Processing Time

7. Deployment via GitHub Codespaces
The project uses ROOT.war as the deployed artifact.

Port 8080 is used and made publicly visible for Android access.

Deployment is confirmed via the browser and Android emulator using the public Codespaces URL.

Servlet and MongoDB technologies are used as required.

You followed the deployment procedure using Docker and .devcontainer provided in the GitHub classroom repository.