Project 4 Task 2

Distributed Systems

Name : Abhishek Venkatadri

Andrew id : av3

This project illustrates an Android application and a Java Servlet-based web service that I developed to allow users to browse and search various dog breeds using The Dog API. I designed and implemented both the frontend (native Android) and backend (Java Servlet) components. The backend logs all user requests to MongoDB Atlas, and I developed a web-based dashboard to display operational statistics such as request counts, status codes, and API response times. The name of my native Android application project in Android Studio is: MobileApplication

1. **Has at least three different kinds of views in your Layout (TextView, EditText, ImageView, etc.)**  
   My application uses **TextView**, **EditText**, **Button**, **RadioButton**, **ImageView**, and a **RecyclerView** to display the search results. These views are incorporated using **ConstraintLayout**, and the layout is defined in activity\_main.xml.

Here is screenshot of the layout before the picture has been fetched

A black rectangular device with a white screen

AI-generated content may be incorrect.

**b) Requires input from the user**

A screen shot of a phone

AI-generated content may be incorrect.

**c. Makes an HTTP request (using an appropriate HTTP method) to your web service**

My application makes an HTTP GET request using the NetworkTask.java class. Depending on the user interaction, the app sends requests to:

https://fuzzy-space-pancake-jj4pxvq6xgg6hw4x-8080.app.github.dev/api/breeds – to get all dog breeds.

https://fuzzy-space-pancake-jj4pxvq6xgg6hw4x-8080.app.github.dev/api?limit=5 – to fetch random dog images.

d)These endpoints are handled by a Java servlet (ApiServlet.java) on the backend. It forwards the request to The Dog API, retrieves the data, logs the request, and sends the response back to the app.

**e) Displays new information to the user**

A screenshot of a phone

AI-generated content may be incorrect.

**f) Is repeatable when you give another data**

A screenshot of a phone

AI-generated content may be incorrect.

**2. Implement a web application, deployed to GitHub Codespaces**

The URL of my web service deployed to GitHub Codespaces is:

https://fuzzy-space-pancake-jj4pxvq6xgg6hw4x-8080.app.github.dev

The project directory name is: Project4Task2

**a. Using an HttpServlet to implement a simple (can be a single path) API**

In my web application project:

Model: Logic is handled within ApiServlet.java and JSON parsing is done using org.json libraries.

Controller:

ApiServlet.java — Handles /api and /api/breeds endpoints

DashboardServlet.java — Handles /dashboard endpoint for analytics

These servlets act as the controller and directly handle routing and response formatting. A separate DAO layer isn't needed because MongoDB operations are handled through a singleton helper class: MongoDBConnection.java.

**b. Receives an HTTP request from the native Android application**

The Android app sends a GET request to the web service using NetworkTask.java.

Example URLs:

/api/breeds for listing breeds

/api?limit=5 for random dogs

These requests are received by ApiServlet.java, which handles parsing, validation, logging, and forwarding the request to The Dog API.

**c. Executes business logic appropriate to your application**

ApiServlet.java performs the following:

Parses the user query or default parameters

Forwards the request to The Dog API (https://api.thedogapi.com/v1/breeds or /v1/images/search)

Parses the JSON response from The Dog API

Logs request data into MongoDB Atlas

Sends back a formatted JSON response to the Android app

This enables seamless integration between the mobile app and the external API, with analytics tracked through MongoDB.

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

The backend replies to the Android application in JSON format. The response is constructed inside the ApiServlet.java based on data retrieved from The Dog API. Depending on the endpoint (/api or /api/breeds), the servlet processes the external API response and returns a simplified and structured JSON array back to the client.

Here’s an example of the JSON response returned for a breed query:

API JSON

[

{

"name": "Pug",

"temperament": "Docile, Clever, Charming",

"life\_span": "12 - 14 years",

"weight": "14 - 18 lbs",

"height": "10 - 12 inches",

"breed\_group": "Toy",

"image\_url": "https://cdn2.thedogapi.com/images/BJa4kxc4X.jpg"

}

]

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Note = Not every dog might have an image but all dogs will have description**