FakeStore API Service and Dashboard

Pleng Witayaweerasak (pwitayaw)

Project 4 Task 1 – FakeStore Android App

Description: My application allows users to search for products using the FakeStore API and view detailed product information on their Android device. The web service processes these requests, communicates with the FakeStore API, and returns filtered product data to the mobile application.

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: FakeStoreApp

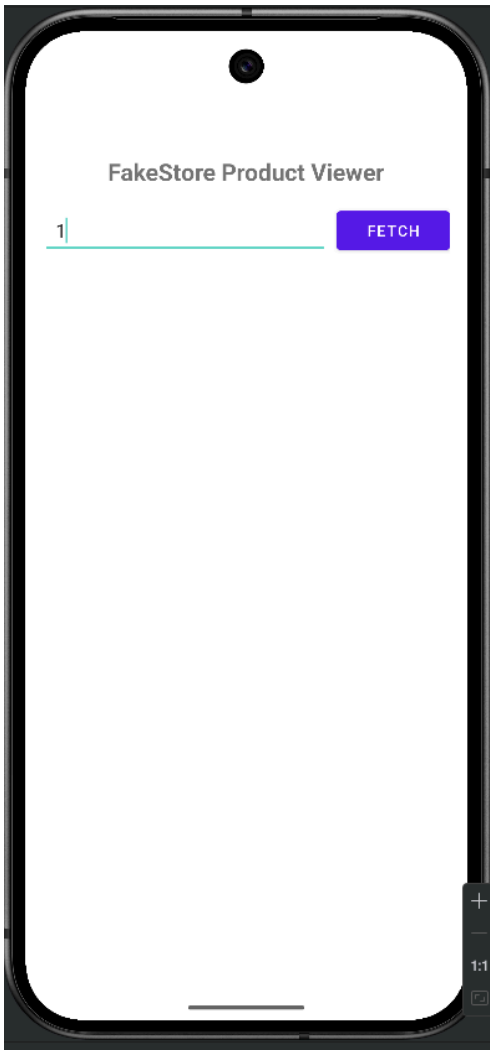
a. Has at least three different kinds of Views in the Layout

My application uses TextView, EditText, Button, and ImageView. See activity_main.xml for details of their implementation in the ConstraintLayout.

Here is a screenshot of the layout before a product has been fetched:

b. Requires input from the user

The user enters a product ID (1-20) in the EditText field and clicks the "Fetch" button to search for product information.



If user clicks Fetch without entering a value, a message is displayed to remind the user to enter a value

c. Makes an HTTP request to the web service

My application performs an HTTP GET request in NetworkUtils.java. The request is sent to:

“https://friendly-barnacle-gp5rvrgpx46cwpp4-8080.app.github.dev/api/product/”+productID

where productID is the product ID entered by the user.

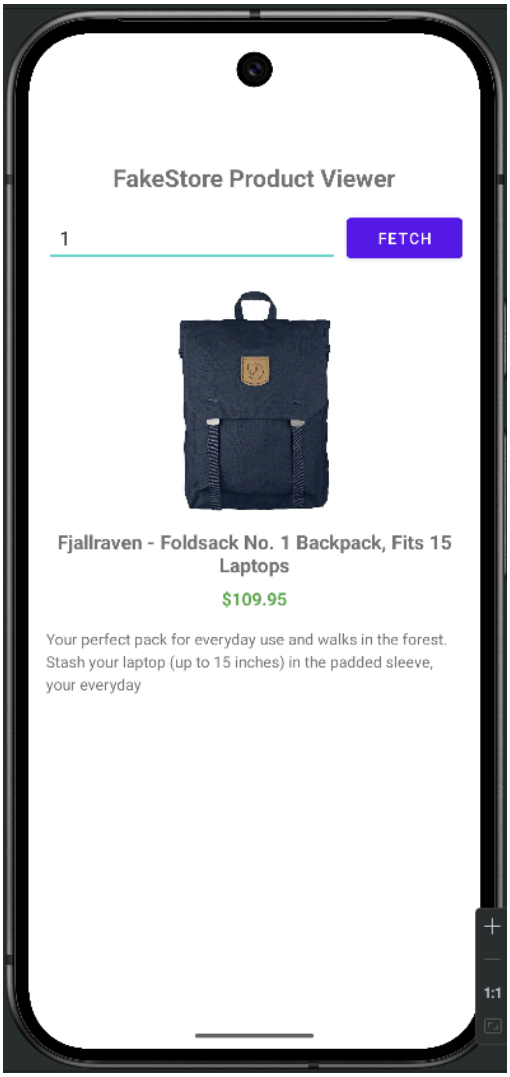
d. Receives and parses a JSON formatted reply from the web service

An example of the JSON reply is:


```
{
    "id": 1,
    "title": "Fjallraven - Foldsack No. 1 Backpack",
    "price": 109.95,
    "description": "Your perfect pack for everyday use and walks in the forest.",
    "image": "https://fakestoreapi.com/img/81fPKd-2AYL._AC_SL1500_.jpg"
}
```

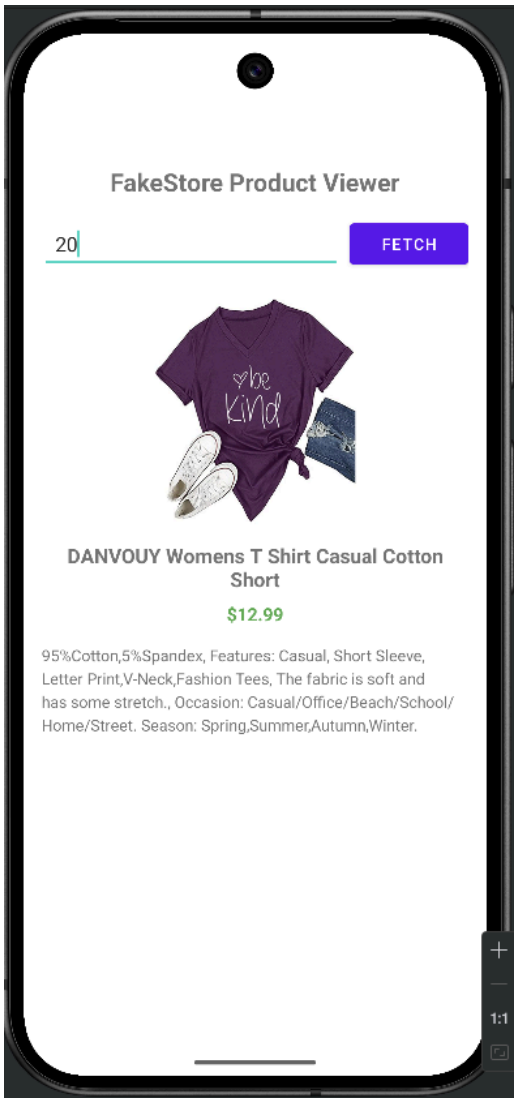
e. Displays new information to the user

The app displays the product title, price, description, and image after receiving the response.



f. Is repeatable

The user can enter a different product ID and tap "Fetch" again to search for a new product without restarting the app.



2. Implement a web service

The URL of my web service deployed to GitHub Codespaces is:
https://friendly-barnacle-gp5rvrgpx46cwpp4-8080.app.github.dev/

a. Using a Servlet to implement a simple API

In my web service project:

- Model: LogEntry.java
- Service: MongoDBService.java
- Controllers: ProductServlet.java, DashboardServlet.java

b. Receives an HTTP request from the native Android application

ProductServlet.java receives the HTTP GET request with the product ID in the path parameter. It extracts this ID and uses it for the business logic.

c. Executes business logic appropriate to the application

The ProductServlet fetches data from the FakeStore API (https://fakestoreapi.com/products/{id}), processes it to filter only the needed fields, and logs information about the request in MongoDB.

d. Replies to the Android application with a JSON formatted response

The response includes only the necessary product information:


```
{
    "id": 5,
    "title": "John Hardy Women's Legends Naga Gold & Silver Dragon Station Chain Bracelet",
    "price": 695.0,
    "description": "From our Legends Collection, the Naga was inspired by the mythical water dragon that protects the ocean's pearl. Wear facing inward to be bestowed with love and abundance, or outward for protection.",
    "image": "https://fakestoreapi.com/img/71pWzhdJNwL._AC_UL640_QL65_ML3_.jpg"
}
```


3. Handle error conditions - does not need to be documented

4. Log useful information

The web service logs the following information for each request:

1. Timestamp of the request
2. Device model making the request
3. Product ID requested
4. HTTP status code of the response
5. API response time (how long the FakeStore API took to respond)
6. Total response time (from request receipt to response sent)
7. User IP address
8. API endpoint called
9. Size of the result returned

This information was chosen to provide comprehensive monitoring capabilities, allowing for both performance analysis and user behavior tracking.

5. Store the log information in a database

All logged information is stored in MongoDB Atlas using the connection string:

```
mongodb+srv://pwitayaw:ZqEblYRGpxADZgR7@cluster0.x2vwv.mongodb.net/fakeStoreDB?retryWrites=true&w=majority&appName=Cluster0
```

6. Display operations analytics and full logs on a web-based dashboard

A web-based dashboard is available at:
https://friendly-barnacle-gp5rvrgpx46cwpp4-8080.app.github.dev/dashboard

The dashboard displays:

- Total number of requests
- Average API response time
- Average total response time
- Top 5 most requested products
- Top 5 device models making requests
- Detailed logs of all requests

FakeStore API Dashboard

Monitoring and analytics for your FakeStore API service

Total Requests

52

Avg API Response Time

0.00 ms

Avg Total Response Time

7.00 ms

Top Requested Products

Product ID	Request Count
1	28
null	11
2	7
3	2
20	2

Top Device Models

Device Model	Request Count
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Ap...	35
Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	16
null	1

Request Logs

Timestamp	Product ID	Device Model	Status	API Response Time	Total Response Time	Result Size
Tue Apr 08 03:15:45 UTC 2025	20	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	126 ms	127 ms	374 bytes
Tue Apr 08 03:15:19 UTC 2025	1	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	160 ms	161 ms	302 bytes
Tue Apr 08 02:57:56 UTC 2025	1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Ap...	200	150 ms	151 ms	302 bytes
Tue Apr 08 02:48:57 UTC 2025	3	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	128 ms	128 ms	441 bytes
Tue Apr 08 02:48:54 UTC 2025	2	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	139 ms	140 ms	507 bytes
Tue Apr 08 02:48:22 UTC 2025	2	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	113 ms	114 ms	507 bytes
Tue Apr 08 02:48:18 UTC 2025	1	Android 16; sdk_gphone64_arm64 Build/BP22.250221.010	200	125 ms	126 ms	302 bytes
Tue Apr 08 02:47:49 UTC 2025	1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) Ap...	200	191 ms	193 ms	302 bytes

7. Deploy the web service to GitHub Codespaces

The ROOT.war was first added to branch `plengchanok-patch-98` and deployed to Codespace then merged back to main branch. There were some issues with cached image in Codespace on main branch so I maintain the friendly barnacle codespace as the web server for the rest of my project.