Project 4 – Brewery Finder App

Shreyas Kailasanathan (AndrewID: skailas2)

Description:

My application allows users to search for breweries in a specific city. The app is divided into two parts: a **web service** and a **native Android app**. The web service retrieves brewery data from an API, stores logs in MongoDB, and serves them to the Android app. The Android app lets users search for breweries and displays the results in a list. Additionally, it provides a dashboard showing analytics and logs of user requests.

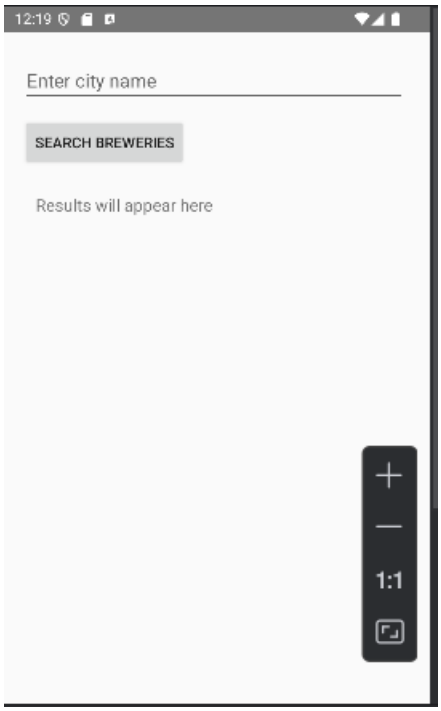
How my application meets the task requirements:

1. Implement a native Android application

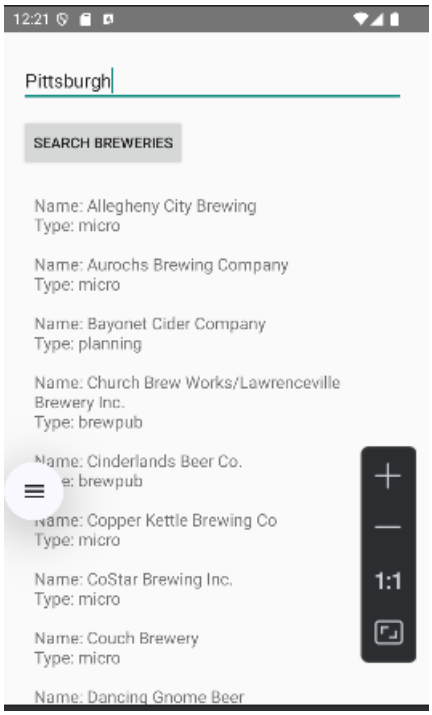
The name of my native Android application project in Android Studio is: **BreweryFinderApp**.
a. **Has at least three different kinds of views in your layout**:

- My application uses TextView, EditText, Button, and ListView and ImageView to display the search results. The main layout consists of a TextView for the app title, an EditText for entering the city, a Button to trigger the search, and a ListView to display the brewery results.
- **XML Layouts**: The layout is designed using LinearLayout and RelativeLayout, ensuring a simple and user-friendly interface.

Here is a screenshot of the layout when we start the application.



b. Requires input from the user – User keys in the name of the city to get the list of breweries present in the city



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

My application makes an **HTTP GET request** to retrieve brewery data from the backend web service. The HTTP request is constructed as follows:

"https://obscure-spork-4jvv7pg7975qfqj47-8080.app.github.dev/brewery?city="

Where city is the user's input for the city name.

The BrewerySearchTask class in the Android application is responsible for making this request. It sends the city name as a query parameter to the backend API, which in turn queries the **Open Brewery DB** and returns a list of breweries in the specified city. The BrewerySearchTask parses the returned **JSON data**, which includes the brewery name, type, and website URL.

Once the data is retrieved, the app updates the UI by displaying the list of breweries in a ListView (or similar UI element). If no breweries are found for the given city, the app shows an appropriate image or message indicating that no results were returned. The app also gracefully handles any errors, such as network failures, by displaying error messages or images.

This process ensures that the application can dynamically fetch and display brewery information based on user input, making it responsive to varying city search queries.

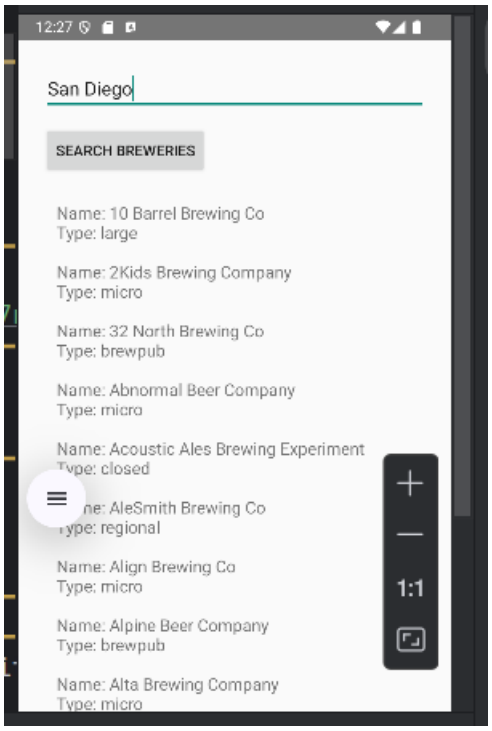
d. Receives and parses JSON formatted reply from the web service

didactic-waddle-4jvv7pg79qj9f5w7r-8080.app.github.dev/brewery?city=pittsburgh

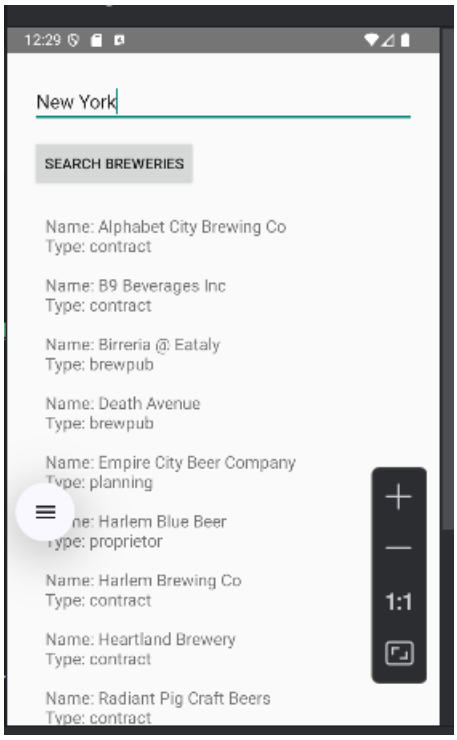
Pretty-print


```
[
  {
    "website_url": "http://www.alleghenycitybrewing.com",
    "brewery_type": "micro",
    "name": "Allegheny City Brewing"
  },
  {
    "website_url": "",
    "brewery_type": "micro",
    "name": "Aurochs Brewing Company"
  },
  {
    "website_url": "",
    "brewery_type": "planning",
    "name": "Bayonet Cider Company"
  },
  {
    "website_url": "http://www.churchbrew.com",
    "brewery_type": "brewpub",
    "name": "Church Brew Works/Lawrenceville Brewery Inc."
  },
  {
    "website_url": "http://www.cinderlands.com",
    "brewery_type": "brewpub",
    "name": "Cinderlands Beer Co."
  },
  {
    "website_url": "",
    "brewery_type": "micro",
    "name": "Copper Kettle Brewing Co"
```


e. Displays new information to the user



f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.) The user can type in another search term and hit Submit.



2. Implement a web application, deployed to GitHub Codespaces and served by Tomcat

The URL of my web service is:
"https://obscure-spork-4jvv7pg7975qfqj47-8080.app.github.dev/brewery?city="
The project directory name is distributed-systems-project-04-skailas2.

a. **Using an HttpServlet to implement a simple API**
In my web app project:

- **Model**: BreweryModel.java
- **View**: index.jsp
- **Controller**: BreweryServlet.java

b. **Receives an HTTP request from the native Android application**
The BreweryServlet.java class receives the HTTP GET request with the argument city. The city name is passed as a query parameter in the URL, such as:

https://obscure-spork-4jvv7pg7975qfqj47-8080.app.github.dev/brewery?city=

c. **Executes business logic appropriate to your application**
The BreweryModel.java class handles the logic of sending a request to the **Open Brewery DB** API with the city name as a query parameter. The API returns a list of breweries in the specified city. This data is then parsed, including the brewery name, type, and website URL.

d. **Replies to the Android application with a JSON formatted response**
Once the brewery data is fetched and parsed, the BreweryServlet.java returns the data as a **JSON** response. This allows the Android application to easily process the results and display them in the app's UI. If no breweries are found for the given city, the app displays an appropriate image or message indicating that no results were returned.

DASHBOARD

This covers the following :

4. Log useful information

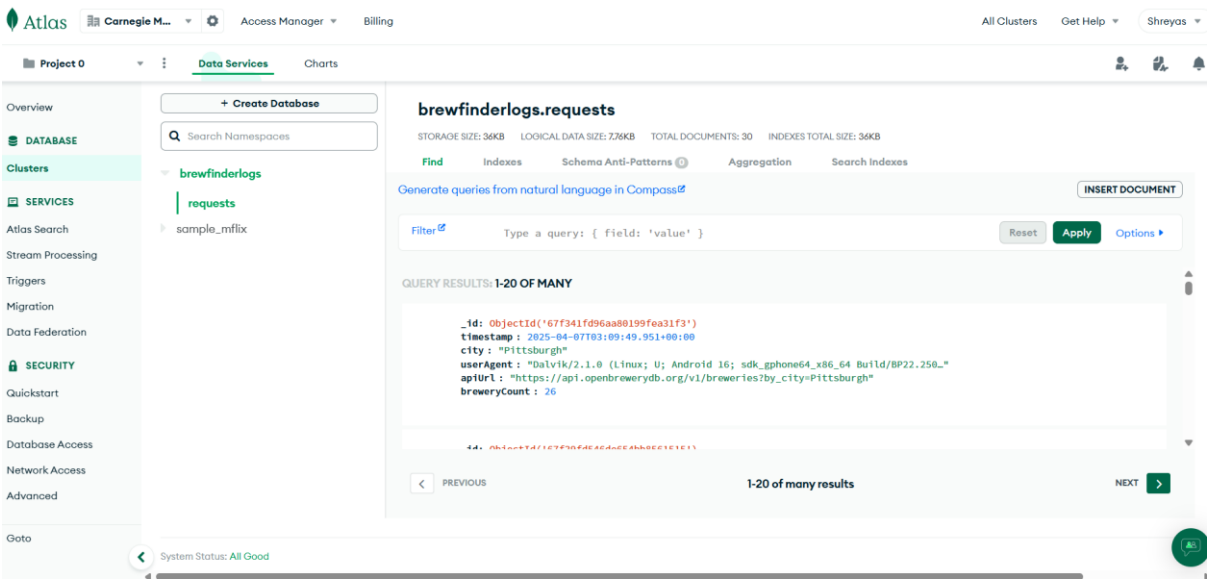
The dashboard gives us data based on the hits.

We get a list of the top 5 cities visited, we get the top devices used and the number of times they have hit the webservice URL

We get the number of times each of the 5 cities was hit along with number of breweries in all of them.

In addition to this, we also get the timestamp, device used and brewery count for each log.

5. Store the log information in a database



We are using MongoDB;

Screenshot attached

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

Operations Dashboard for Brewfinder Service

Analytics

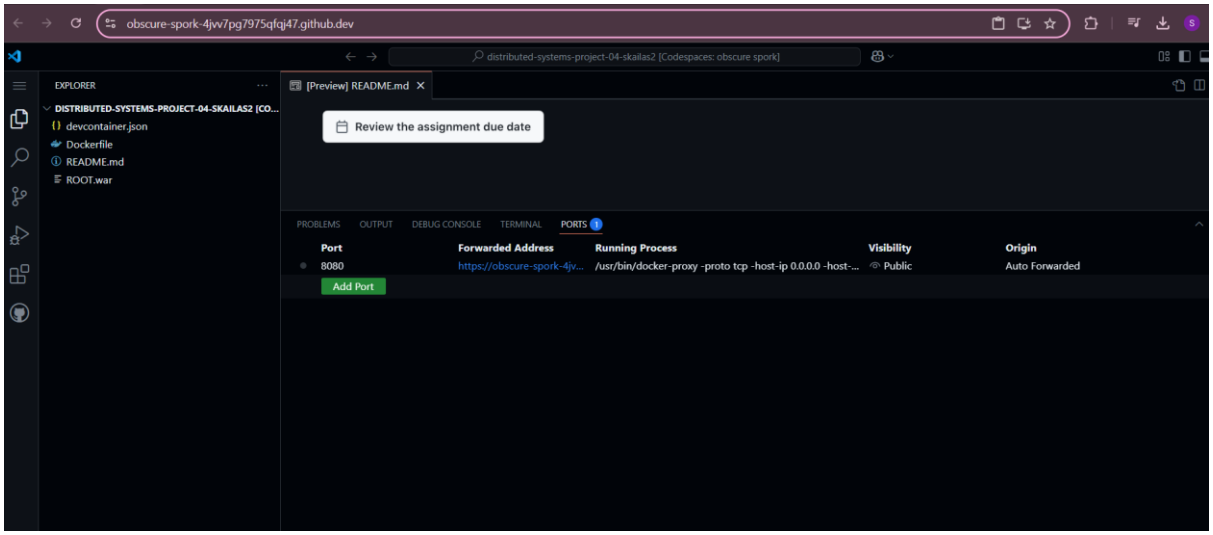
- **Total Requests:** 30
- **Top 5 Cities:**
  - pittsburgh — 16 requests
  - New York — 3 requests
  - Pittsburgh — 3 requests
  - Chicago — 2 requests
  - San Diego — 2 requests
- **Top Devices:**
  - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36 — 16 uses
  - Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010) — 14 uses

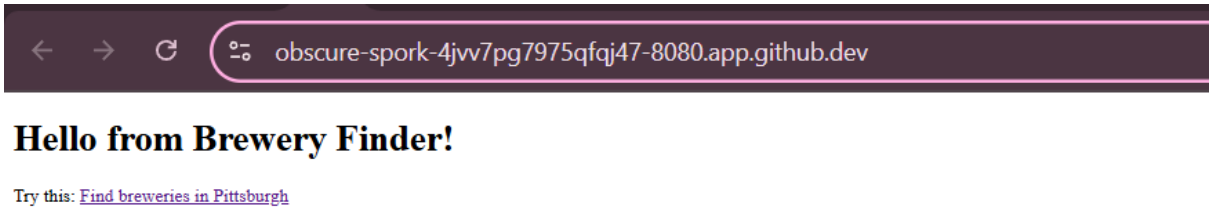
Full Logs

Timestamp	City	User Agent	Result Count
Tue Apr 08 16:04:30 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 15:39:45 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 15:38:34 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 15:20:00 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 15:19:32 UTC 2025	New York	Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010)	13
Tue Apr 08 15:18:03 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 15:09:42 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Tue Apr 08 10:45:12 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Mon Apr 07 16:29:22 UTC 2025	New York	Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010)	13
Mon Apr 07 16:27:29 UTC 2025	San Diego	Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010)	50
Mon Apr 07 16:24:28 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Mon Apr 07 16:21:27 UTC 2025	Pittsburgh	Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010)	26
Mon Apr 07 15:51:54 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Mon Apr 07 15:49:59 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26
Mon Apr 07 15:41:56 UTC 2025	San Diego	Dalvik/2.1.0 (Linux; U; Android 16; sdk_gphone64_x86_64 Build/BP22.250221.010)	50
Mon Apr 07 15:41:21 UTC 2025	pittsburgh	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/134.0.0.0 Safari/537.36	26

The dashboard is accessible at - https://obscure-spork-4jvv7pg7975qfqj47-8080.app.github.dev/dashboard

Part 7 : We can run the War using docker





Hello from Brewery Finder!

Try this: Find breweries in Pittsburgh