DISTRIBUTED SYSTEMS Kumar Amartya andrewID:aamartya

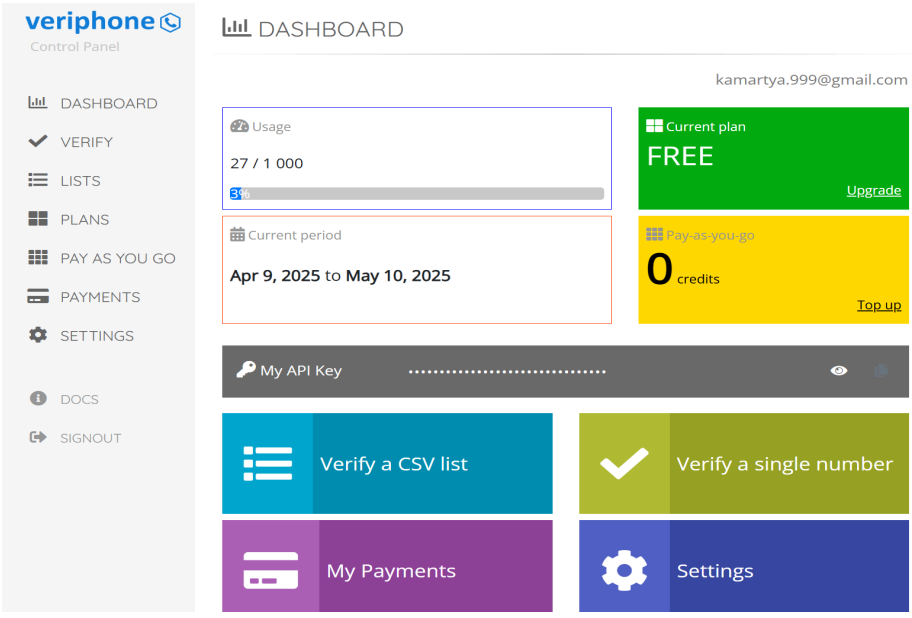
Project 4 Task 2 – Phone Number Verification App

Description:

My distributed application allows users to verify phone numbers via a native Android app. It fetches phone verification data (validity, country, carrier, type) from the Veriphone API and logs each request to MongoDB. A web-based dashboard displays verification analytics such as request counts, status, country, timestamp etc.

API Documentation v2

Veriphone API is a REST based JSON API. It provides a set of stateless endpoints that any program or web browser can call by sending a standard HTTP request. Veriphone will respond with a standard HTTP response carrying a JSON payload. This documentation describes these endpoints, their input/output parameters and authentication methods.



Veriphone Dashboard

How My Application Meets Each Requirement

1. Implement a native Android application

Project name in Android Studio: `PhoneNumberApp`

a. Has at least three different kinds of views

My Android layout includes:

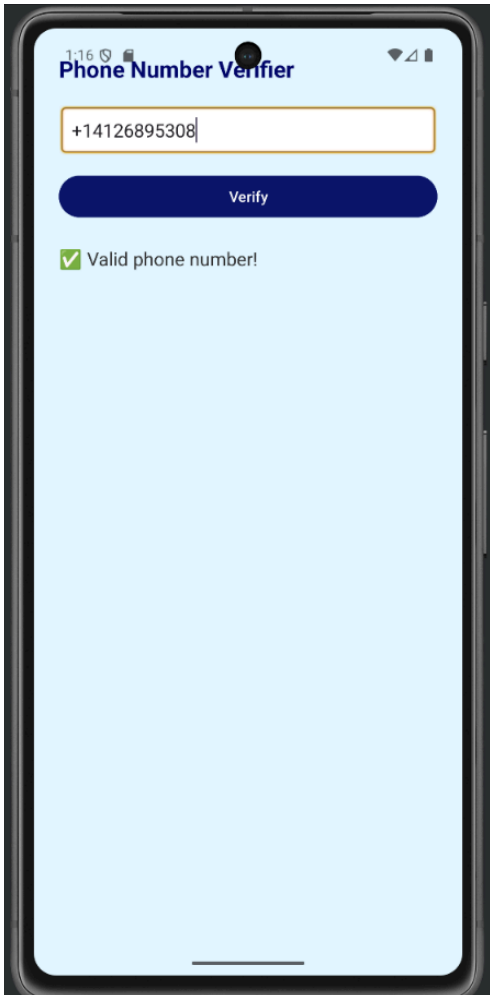
- `TextView` – displays the title and result
- `EditText` – allows phone number input
- `Button` – triggers verification

The layout is defined in activity_main.xml. See `activity_main.xml` for details.

Here is screenshot of the layout before the number is verified

b. Requires input from the user

The app requires users to input a phone number in the `EditText` field.



Here is screenshot of the layout after the number has been verified

c. Makes an HTTP request (using an appropriate HTTP method)

The app performs an HTTP `GET` request to my web service:

"https://didactic-orbit-pw6776v79w4h74qg-8080.app.github.dev/verify-phone"

The code for this is in `MainActivity.java`.

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

The server responds with a JSON object:

```
{
  "phone": "+1234567890",
  "valid": true
}
```

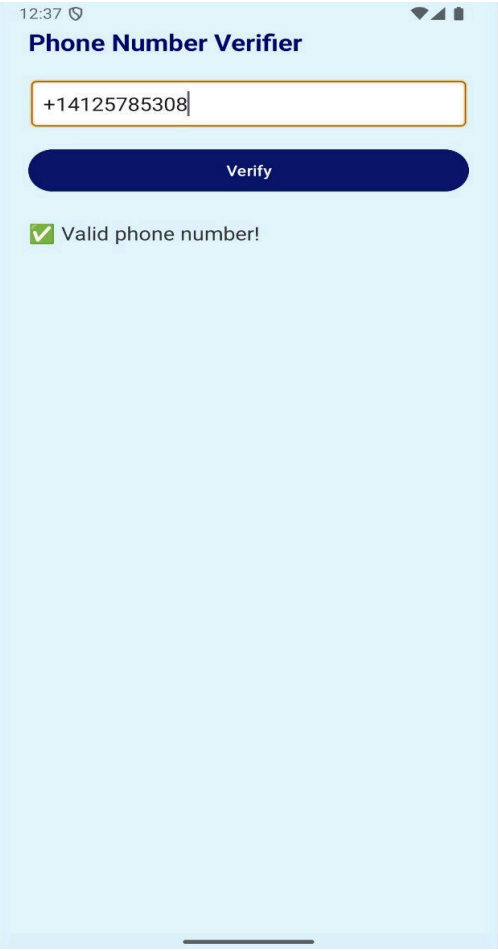
Parsed in `MainActivity.java` using `JSONObject`.

e. Displays new information to the user

The `TextView` (`@+id/resultText`) shows whether the phone number is valid or not based on the API response.

f. Is repeatable

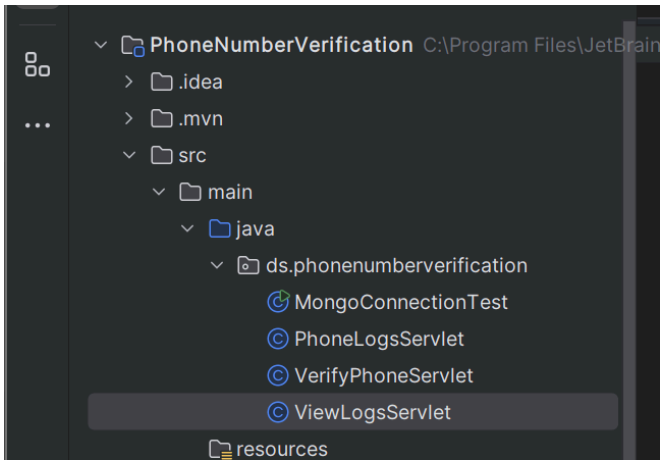
Users can verify as many phone numbers as they want without restarting the app.



2. Implement a web application, deployed to GitHub Codespaces

URL of the deployed backend:
`https://didactic-orbit-pw6776v79w4h74qg-8080.app.github.dev/verify-phone`

Project name in IntelliJ: `PhoneNumberVerification`



Here is screenshot of the IntelliJ project directory

a. Using an HttpServlet to implement a simple API

In my web app project:

- **Model:**
  Handled within the `VerifyPhoneServlet.java` by integrating the Veriphone API and parsing JSON. Endpoint: `/verify-phone`

- **View:**
  The Android app displays the response. For logs, `dashboard.jsp` serves the visual output.

- **Controller:**
  `VerifyPhoneServlet.java`
  Handles incoming requests from the Android client at `/verify-phone`, calls the Veriphone API, logs data to MongoDB, and returns a JSON response.

It fetches details from the Veriphone API and returns a minimal JSON object.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 with the `phone` parameter to `/verify-phone`.

c. Executes business logic appropriate to your application

The`VerifyPhoneServlet.java` performs:

- Extracts the phone number from the request parameters
- External API call to Veriphone.
- Parses JSON response.
- Extracts and stores: `valid`, `country`, `carrier`, `phone_type`.
- Logs the extracted data into the `phonelogs` collection in MongoDB Atlas

d. Replies to the Android application with a JSON response

The backend replies to the Android application in **JSON format**.
The response is constructed inside the `VerifyPhoneServlet.java` class using the parsed output from the Veriphone API.

The servlet extracts relevant fields (`phone`, `valid`) from the JSON and returns a simplified response to the Android client.

Here's an example of the JSON response returned for a phone verification:

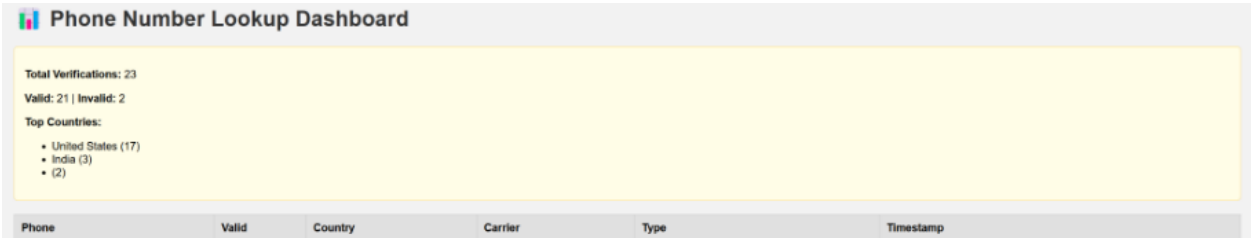
```
{
  "phone": "+1234567890",
  "valid": true
}
```

This response is parsed in the Android app and rendered in the UI as either a valid or invalid phone number.

4. Log useful information

Logged fields:

- `timestamp`
- `phone`
- `valid`
- `country`
- `carrier`
- `phone_type`
- `source`



Logged fields

See: `VerifyPhoneServlet.java`

Why is this logged?

This helps in debugging, monitoring country-wise usage, and tracking API performance.

5. Store the log information in a database

Database: **MongoDB Atlas**
Connection string (uses cluster with 3 shards):

mongodb+srv://aamartya:cmumism@androidproject.l70o0.mongodb.net/?retryWrites=true&w=majority&appName=AndroidProject

Use this connection string in your application

View full code sample

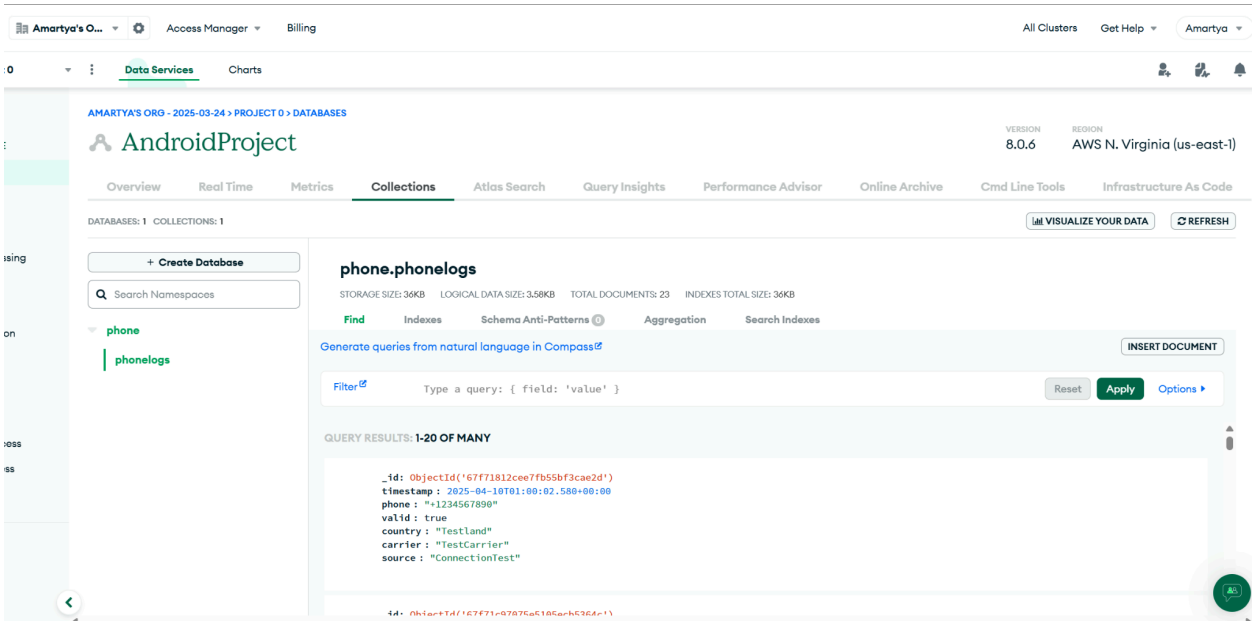
```
mongodb+srv://aamartya:<db_password>@androidproject.l70o0.mongodb.net/?
retryWrites=true&w=majority&appName=AndroidProject
```

Replace **<db_password>** with the password for the **aamartya** database user. Ensure any option params are URL encoded .

Database name: `phone`

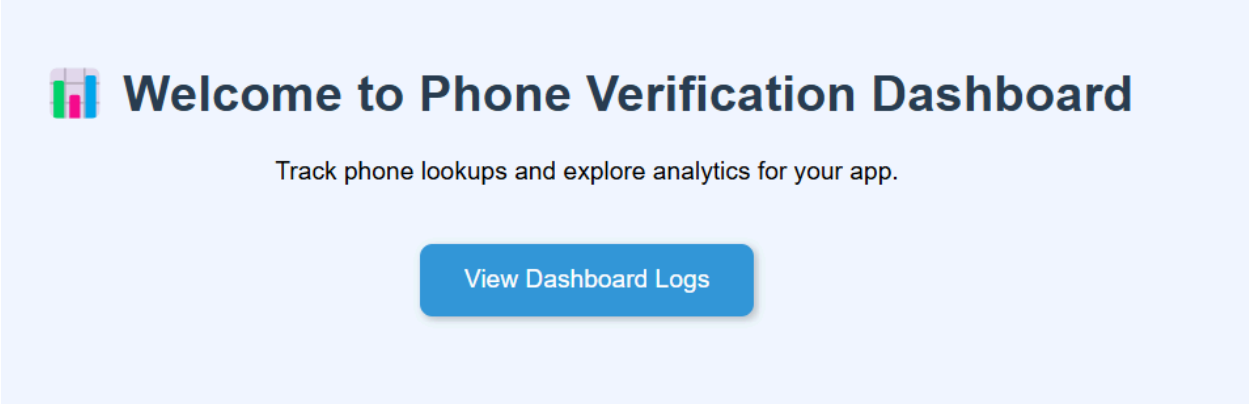
Collection: `phonelogs`

See: `VerifyPhoneServlet.java` and `PhoneLogsServlet.java`



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

Dashboard URL:
`https://didactic-orbit-pw6776v79w4h74qg-8080.app.github.dev/phone-logs`

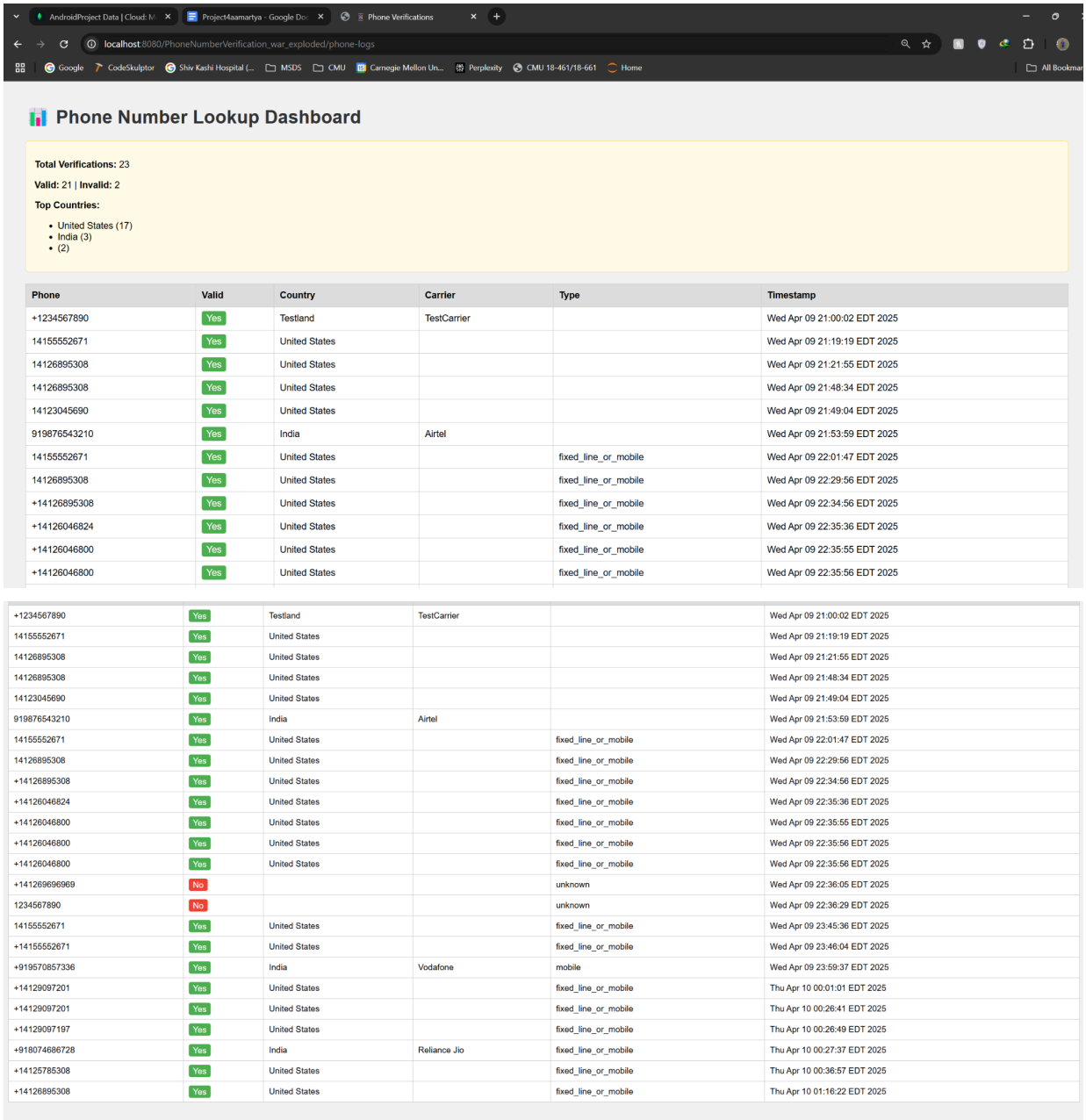


Welcome page

Dashboard features:

- Total verifications
- Count of valid and invalid numbers
- Top 3 countries
- Full log table (phone, status, country, carrier, type, timestamp)

See: `PhoneLogsServlet.java`
Dashboard UI is styled and served via dynamic HTML.



+1234567890	Yes	Testland	TestCarrier		Wed Apr 09 21:00:02 EDT 2025
14155552671	Yes	United States			Wed Apr 09 21:19:19 EDT 2025
14126895308	Yes	United States			Wed Apr 09 21:21:55 EDT 2025
14126895308	Yes	United States			Wed Apr 09 21:48:34 EDT 2025
14123045690	Yes	United States			Wed Apr 09 21:49:04 EDT 2025
919876543210	Yes	India	Airtel		Wed Apr 09 21:53:59 EDT 2025
14155552671	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:01:47 EDT 2025
14126895308	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:29:56 EDT 2025
+14126895308	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:34:56 EDT 2025
+14126046824	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:35:36 EDT 2025
+14126046800	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:35:55 EDT 2025
+14126046800	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:35:56 EDT 2025
+14126046800	Yes	United States		fixed_line_or_mobile	Wed Apr 09 22:35:56 EDT 2025
+141269696969	No			unknown	Wed Apr 09 22:36:05 EDT 2025
1234567890	No			unknown	Wed Apr 09 22:36:29 EDT 2025
14155552671	Yes	United States		fixed_line_or_mobile	Wed Apr 09 23:45:36 EDT 2025
+14155552671	Yes	United States		fixed_line_or_mobile	Wed Apr 09 23:46:04 EDT 2025
+919570857336	Yes	India	Vodafone	mobile	Wed Apr 09 23:59:37 EDT 2025
+14129097201	Yes	United States		fixed_line_or_mobile	Thu Apr 10 00:01:01 EDT 2025
+14129097201	Yes	United States		fixed_line_or_mobile	Thu Apr 10 00:26:41 EDT 2025
+14129097197	Yes	United States		fixed_line_or_mobile	Thu Apr 10 00:26:49 EDT 2025
+918074686728	Yes	India	Reliance Jio	fixed_line_or_mobile	Thu Apr 10 00:27:37 EDT 2025
+14125785308	Yes	United States		fixed_line_or_mobile	Thu Apr 10 00:36:57 EDT 2025
+14126895308	Yes	United States		fixed_line_or_mobile	Thu Apr 10 01:16:22 EDT 2025

Tested two invalid numbers hence there's no country for two results