Project 4– Mars Rover Photo App

by Junyi Zhu (AndrewID: junyizh2)

Description:

The Mars Rover Photo App allows users to fetch and display photos from NASA's Mars Rover API based on a user-specified Martian sol (solar day). The app includes both a native Android application and a web service backend, which follows the MVC architecture. The Android app features a dynamic user interface and connects to a hosted web service that handles API requests, logging, and analytics.

---

1. Implement a native Android application

The name of my native Android application project in Android Studio is: MarsRoverPhotoApp.

a. **Has at least three different kinds of views in your Layout (TextView, EditText, Button, RecyclerView)**

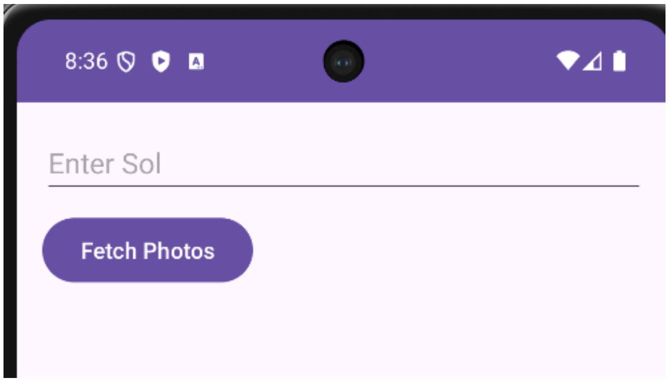
The app's layout utilizes multiple views:

- **EditText:** For entering the Martian sol number.
- **Button:** To trigger photo retrieval from the web service.
- **RecyclerView:** Displays fetched photo details, with each item including:
  - **Camera Name (TextView)**
  - **Sol Number (TextView)**
  - **Image URL (TextView),** displayed as a clickable hyperlink.

Implementation Details:

- The layout is defined in activity_main.xml, with individual RecyclerView items styled in item_url.xml.
- Photo objects store photo details, and the UrlAdapter manages the RecyclerView display.

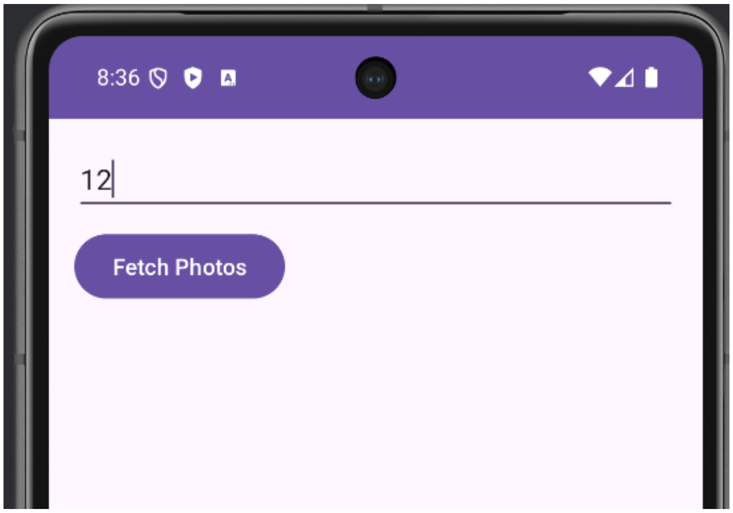
Here is a screenshot of the layout before the picture has been fetched.



b. Requires input from the user

The user inputs a Sol number in the **EditText** field to fetch Mars Rover data from NASA's API. If the input is invalid (e.g., empty), the app displays an error message.

Example:
When the user inputs "12" and clicks "Fetch Photos," the app fetches and displays data for Sol 12. If the input is blank, it displays "No Sol provided."



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

The app makes an HTTP GET request to the web service hosted on GitHub Codespaces:
GET https://verbose-space-waffle-7xgxq545qqg2x6r-8080.app.github.dev/api/photos?sol=[user_input]

Code Implementation:

String url = BASE_URL + "?sol=" + sol;

Request request = new Request.Builder().url(url).build();

client.newCall(request).enqueue(new Callback() { ... });

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

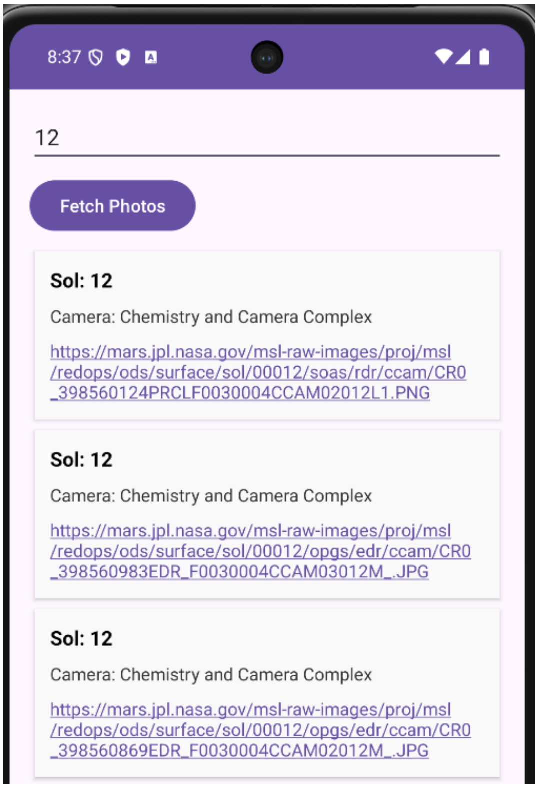
The app uses the org.json library to parse JSON responses from the web service. Parsed data is stored in a List<Photo>object and displayed in the RecyclerView.

Sample JSON Reply from the Web Service

```
{
  "photos": [
    {
      "sol": "1000",
      "camera_name": "Panoramic Camera",
      "img_src": "https://mars.nasa.gov/img1.jpg"
    },
    {
      "sol": "1000",
      "camera_name": "Navigation Camera",
      "img_src": "https://mars.nasa.gov/img2.jpg"
    }
  ]
}
```

e. Displays new information to the user

- The fetched photo details are displayed in a list (RecyclerView) with:
  - Sol: The Martian day for the photo.
  - Camera Name: The camera that captured the photo.
  - Image URL: A clickable hyperlink that opens the image in a browser.
- If no data is available, the app displays an appropriate error message.



12

Sol: 12

Camera: Chemistry and Camera Complex

https://mars.jpl.nasa.gov/msl-raw-images/proj/msl/redops/ods/surface/sol/00012/soas/rdr/ccam/CR0_398560124PRCLF0030004CCAM02012L1.PNG

Sol: 12

Camera: Chemistry and Camera Complex

https://mars.jpl.nasa.gov/msl-raw-images/proj/msl/redops/ods/surface/sol/00012/opgs/edr/ccam/CR0_398560983EDR_F0030004CCAM03012M_.JPG

Sol: 12

Camera: Chemistry and Camera Complex

https://mars.jpl.nasa.gov/msl-raw-images/proj/msl/redops/ods/surface/sol/00012/opgs/edr/ccam/CR0_398560869EDR_F0030004CCAM02012M_.JPG

f. Is repeatable (I.e., the user can repeatedly reuse the application without restarting it.)

The app allows users to input new sol values and fetch updated data without restarting the application.

---

2. Implement a web application, deployed to GitHub Codespaces

The web service is deployed on GitHub Codespaces and hosted at:
https://verbose-space-waffle-7xgxq545qqg2x6r-8080.app.github.dev/api/photos

The project directory name is: MarsRoverService

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

The web service follows the MVC pattern:

- **Model**: Handles NASA API interactions and logging using NASADataModel and LogDataModel.
- **Controller**: Implements request handling and business logic through PhotoController and DashboardController.
- **View**: Consists of JSP files (dashboard.jsp, index.jsp) that render analytics and logs for a browser-based dashboard.

b. Receives an HTTP request from the native Android application

- The CombinedServlet receives the sol value as a query parameter from the Android application.

Implementation in CombinedServlet:

```
String sol = request.getParameter("sol");

if (sol == null || sol.isEmpty()) {

    sol = "1";

}

String nasaData = getNASAData(sol);

response.getWriter().write(nasaData);
```

c. Executes business logic appropriate to your application

- The servlet constructs an HTTP GET request to NASA's Mars Rover API:

https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/photos?sol=[user_input]&api_key=YOUR_API_KEY

- The JSON response is filtered to include only the sol, camera_name, and img_src fields, which are forwarded to the Android application.

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

- The response to the Android app is formatted as a JSON object:

```
{

  "photos": [

    {

      "sol": 12,

      "camera_name": "Chemistry and Camera Complex",

      "img_src": "http://mars.jpl.nasa.gov/sample.jpg"

    }

  ]

}
```

---

3. Handle error conditions

1. Invalid mobile app input: Displays "No Sol provided" if input is empty.
2. Invalid server-side input: Defaults to Sol 1 if the Sol parameter is invalid.
3. Network failure: Displays an error message if the web service is unreachable.
4. Third-party API unavailable: Returns a JSON error message.
5. Invalid third-party data: Ensures robust JSON parsing with error handling.

---

4. Log useful information

- Logs include:

The web service logs the following data to MongoDB:

1. Timestamp of the request.
2. Sol number requested.
3. User agent of the requesting device.
4. URL of the third-party API request.
5. Status of the response from NASA's API.
6. Number of photos returned.

Example Log Document:

```
{
  "timestamp": "2024-11-20T10:00:00Z",
  "sol": "12",
  "userAgent": "Mozilla/5.0 (Android)",
  "apiUrl": "https://api.nasa.gov/mars-photos/api/v1/rovers/curiosity/photos?sol=12&api_key=YOUR_API_KEY",
  "status": 200,
  "photoCount": 10
}
```

---

5. Store the log information in a database

- Logs are stored in a MongoDB Atlas database hosted in the cloud.

- Connection String:

mongodb+srv://[username]:[password]@cluster0.mongodb.net/project4

---

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

The dashboard is hosted at:
https://verbose-space-waffle-7xgxq545qqg2x6r-8080.app.github.dev/dashboard

Features

The dashboard provides:

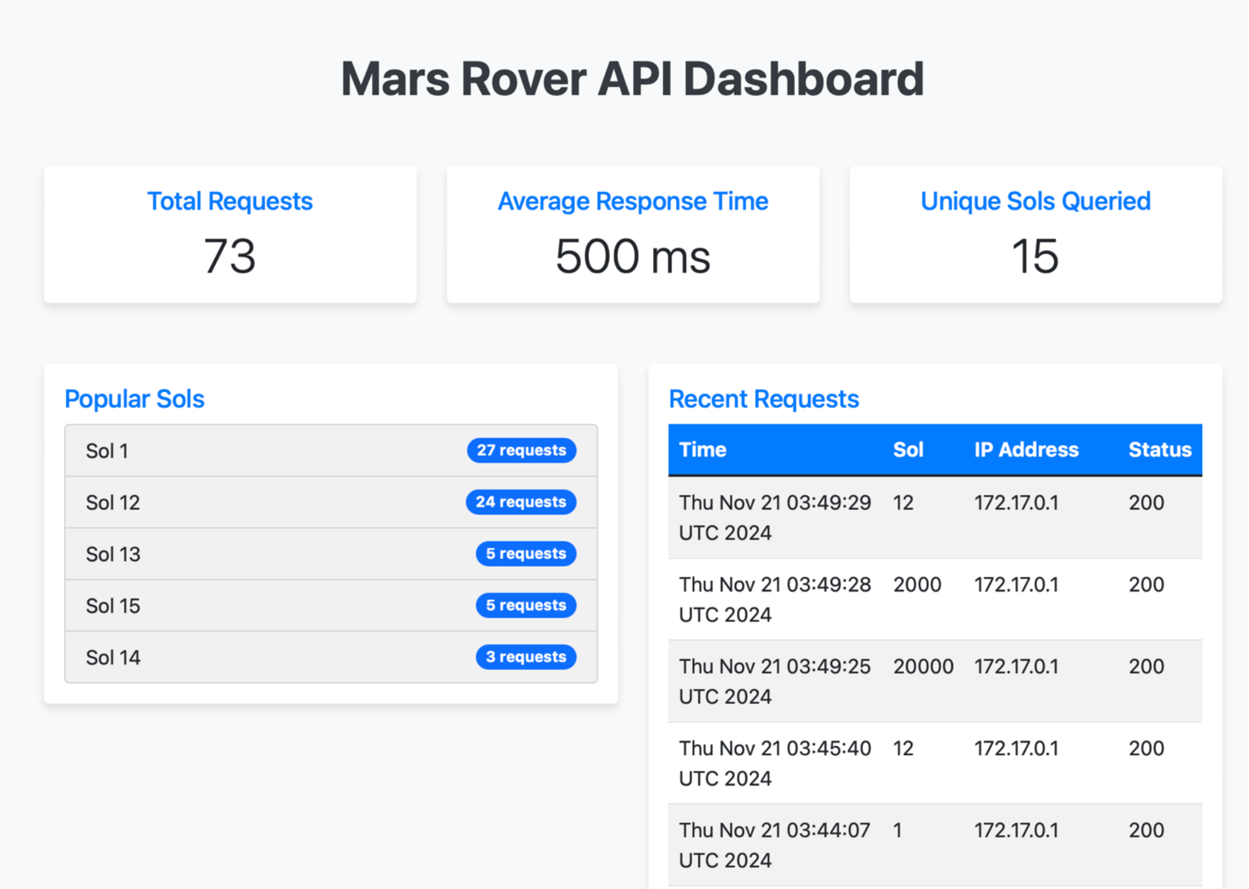
1. **Total Requests:** Displays the total number of requests made.
2. **Popular Sols:** Shows the top 5 most requested sols and their counts.
3. **Recent Requests Table:** Displays details for recent requests, including:
    - Timestamp
    - Sol number
    - IP address
    - User agent

Example UI:

- dashboard.jsp renders analytics in a tabular format.

Screenshots and Supporting Evidence

Dashboard Analytics



Mars Rover API Dashboard

Total Requests
73

Average Response Time
500 ms

Unique Sols Queried
15

Popular Sols

Sol 1	27 requests
Sol 12	24 requests
Sol 13	5 requests
Sol 15	5 requests
Sol 14	3 requests

Recent Requests

Time	Sol	IP Address	Status
Thu Nov 21 03:49:29 UTC 2024	12	172.17.0.1	200
Thu Nov 21 03:49:28 UTC 2024	2000	172.17.0.1	200
Thu Nov 21 03:49:25 UTC 2024	20000	172.17.0.1	200
Thu Nov 21 03:45:40 UTC 2024	12	172.17.0.1	200
Thu Nov 21 03:44:07 UTC 2024	1	172.17.0.1	200

6. Technical Notes

Android Application

- **Files**:
    - MainActivity.java: Manages user interaction and HTTP requests.
    - Photo.java: Represents photo data as objects.
    - UrlAdapter.java: Binds photo data to the RecyclerView.
    - Layouts: activity_main.xml, item_url.xml.

Web Service

- **Files**:
    - Controllers: PhotoController.java, DashboardController.java.
    - Models: NASADataModel.java, LogDataModel.java.
    - Views: dashboard.jsp, index.jsp.

This concludes the implementation of Mars Rover Photo App.