Project 4 Task 2 - NBA Stats APP

Yiming Fu (Andrew ID: yimingfu)

Description

My application is designed for users to fetch NBA stats, including player and team level, which are fetched from https://app.balldontlie.io/.

Below I will show how my application meets all requirements.

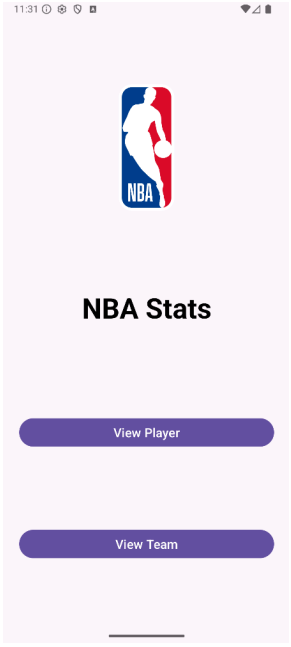
1. Implement a native Android application

The name of my native Android application project in Android Studio is: Project4App

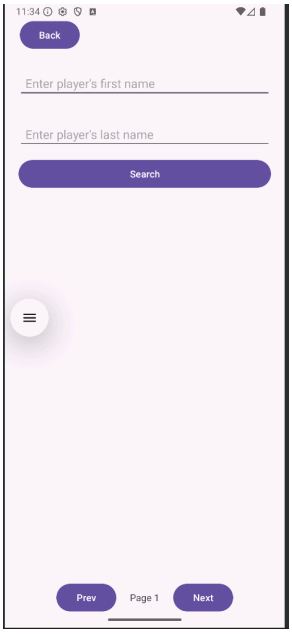
a. Has at least three different kinds of views in your Layout

My application uses ImageView, TextView, EditText, Button, and ProgressBar. For layout style, I use LinearLayout and ConstraintLayout, while the former one can organize different views, and the latter one can better fit the page.

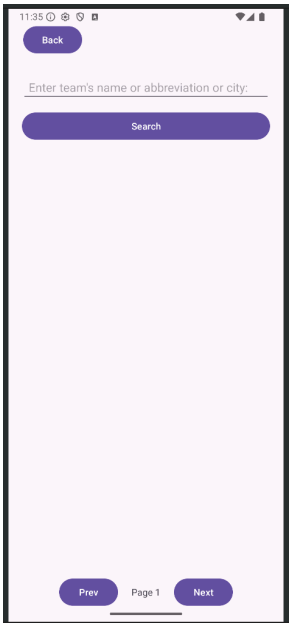
Here is a screenshot of the home page, where user can select "Player" or "Team" page.



If users select "Player" page, here is a screenshot of the player page.

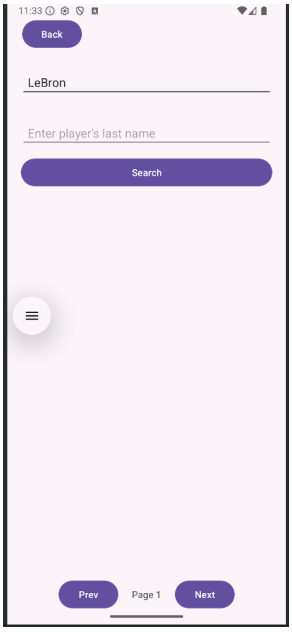


If users select "Team" page, here is a screenshot of the team page.

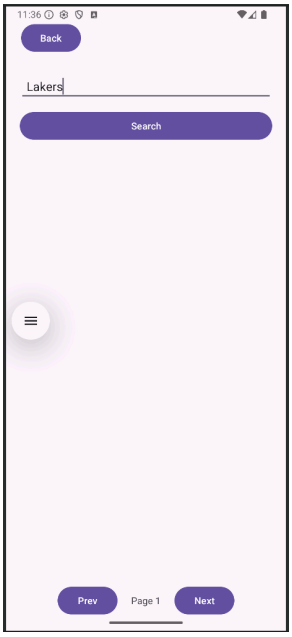


b. Requires input from users

If users select "Player" page, here is a screentshot of users searching for player with first name LeBron.



If users select "Team" page, here is a screenshot of users searching for team Lakers. (Users can also search by abbreviation or city.)



c. Makes an HTTP request to your web service

My application does an HTTP GET request in *PlayerActivity.java* and *TeamActivity.java*.

In *PlayerActivity.java*, the HTTP request is: (as the previous example)

```
https://crispy-chainsaw-7qj54556j9wfxrv9-8080.app.github.dev/api/player?first_name=LeBron
```

In *TeamActivity.java*, the HTTP request is: (as the previous example)

```
https://crispy-chainsaw-7qj54556j9wfxrv9-8080.app.github.dev/api/team?team=Lakers
```

Note: My web service is deployed in Github Codespace, so the HTTP request points to the corresponding URL.

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

An example of the JSON reply for player from server is:


```
{
  "data": [
    {
      "id": 237,
      "firstName": "LeBron",
      "lastName": "James",
      "position": "F",
      "jerseyNumber": "23",
      "height": "6'9\" (205.7 cm)",
      "weight": "250 lb (113.4 kg)",
      "college": "St. Vincent-St. Mary HS (OH)",
      "country": "USA",
      "draftYear": 2003,
      "draftRound": 1,
      "draftNumber": 1,
      "team": {
        "id": 14,
        "conference": "West",
        "division": "Pacific",
        "city": "Los Angeles",
        "name": "Lakers",
        "fullName": "Los Angeles Lakers",
        "abbreviation": "LAL"
      },
      "imageUrl": "https://cdn.nba.com/headshots/nba/latest/260x190/2544.png"
    }
  ]
}
```

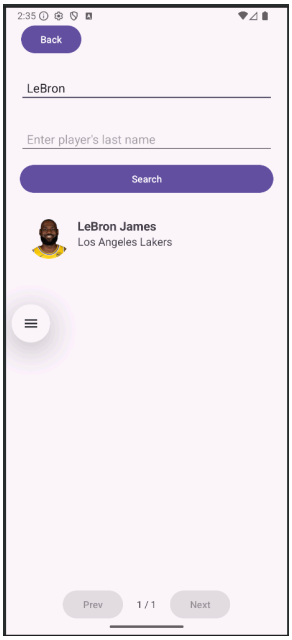

An example of the JSON reply for team from server is:


```
{
  "data": [
    {
      "id": 237,
      "firstName": "LeBron",
      "lastName": "James",
      "position": "F",
      "jerseyNumber": "23",
      "height": "6'9\" (205.7 cm)",
      "weight": "250 lb (113.4 kg)",
      "college": "St. Vincent-St. Mary HS (OH)",
```

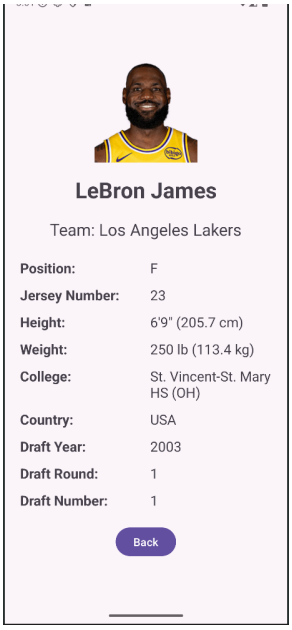
```
      "country": "USA",
      "draftYear": 2003,
      "draftRound": 1,
      "draftNumber": 1,
      "team": {
        "id": 14,
        "conference": "West",
        "division": "Pacific",
        "city": "Los Angeles",
        "name": "Lakers",
        "fullName": "Los Angeles Lakers",
        "abbreviation": "LAL"
      },
      "imageUrl": "https://cdn.nba.com/headshots/nba/latest/260x190/2544.png"
    }
  ]
}
```


e. Displays new information to the user

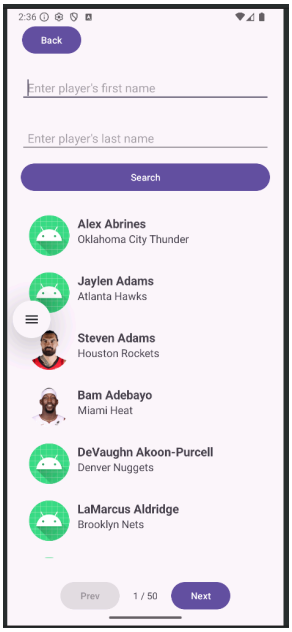
If users select "Player" page, here is the screenshot for all the returned players.



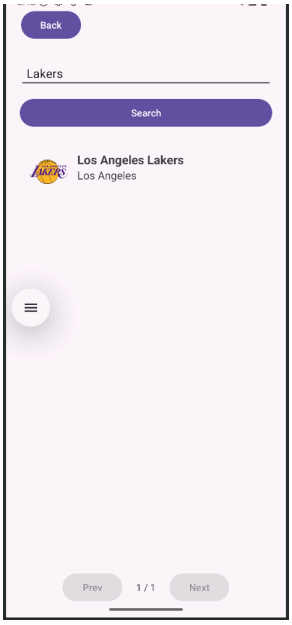
Users can then click the player section to view detailed information.



Note: If users don't input any keywords, the server will return all player information by default. Since I could only find pictures for current NBA players, others' pictures are displayed as the default Android icon.

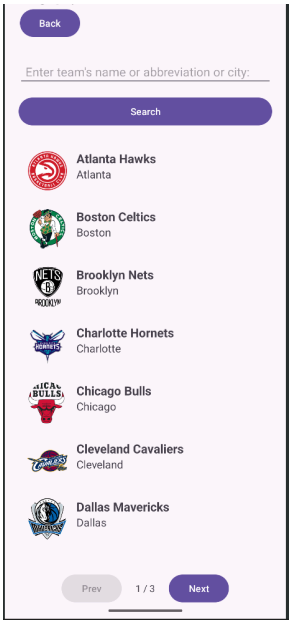


If users select "Team" page, here is the screenshot for all the returned teams.



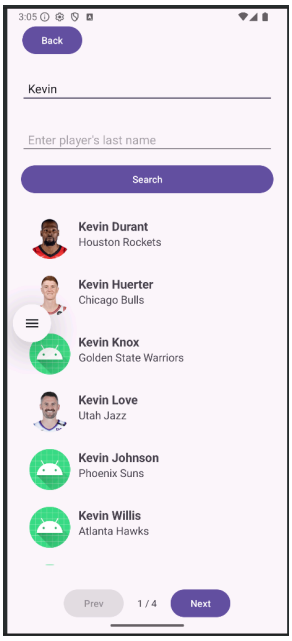
Users can then click the team section to view detailed information.

Note: If users don't input any keywords, the server will return all team information by default.

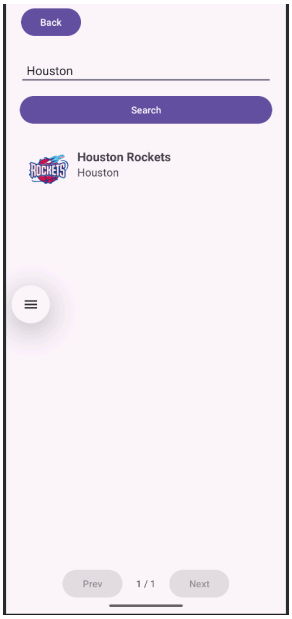


f. Is repeatable (I.e. users can repeatedly reuse the application without restarting it

For the "Player" page, users can directly type in names to search for another player.



For the "Team" page, users can directly type in information to search for another team.



2. Implement a web application, deployed to Github Codespace

The URL of my web service deployed to Github Codespace is:

```
https://crispy-chainsaw-7qj54556j9wfxrv9-8080.app.github.dev/
```

The project directory name is "CMU-95702-Project4". Be sure to set the port to **public** rather than private.

a. Using an HTTPServlet to implement a simple API

In my web app project, I have implemented two APIs for player (/api/player) and team (/api/team) respectively.

Player:

- Model: *PlayerServiceModel.java*
- View: **Not Needed** (because this API responses with JSON data)
- Controller: *PlayerServlet.java*

Team:

- Model: *TeamServiceModel.java*
- View: **Not Needed** (because this API responses with JSON data)
- Controller: *TeamServlet.java*

Besides, I have implemented *welcome.jsp*, which guides users with basic server APIs. (MVC for dashboard and logs will be covered in the following sections.)

b. Receives an HTTP request from the native Android application

For the "Player" page, *PlayerServlet.java* receives the HTTP GET request with the argument "first_name" and

For the "Player" page, *PlayerServlet.java* receives the HTTP GET request with the argument "first_name" and "last_name". It passes the input strings to the model *PlayerServiceModel.java*.

For the "Team" page, *TeamServlet.java* receives the HTTP GET request with the argument "team", representing the name, abbreviation or city of the team. It passes the input string to the model *TeamServiceModel.java*.

c. Executes business logic appropriate to your application

For the "Player" page, *PlayerServiceModel.java* makes an HTTP request to:

```
https://api.balldontlie.io/v1/players?per_page=100&first_name=XXX&last_name=XXX
```

The thrid party API will respond with corresponding players data. If there are more players to fetch, it will return with a field called "next_cursor", while the model will continuously request the API until all data is fetched. After that, it parses the JSON response and extracts the parts it needs to respond to the Android application.

Note: Since I am using the free plan for third party API, the limit for requests per minute is 5. Therefore, I can only have access to 500 entries at most per minute. To avoid waiting for too long in Android app, I just return all the entries fetched within one minue, so the data might not be totally complete. If I have access to premium plan, this wouldn't be an issue.

For the "Team" page, *TeamServiceModel.java* makes an HTTP request to:

```
https://api.balldontlie.io/v1/teams
```

Since the party API doesn't support find team by name, I just fetch all teams' information initially (only 30 teams). Then the model will parse the JSON response and extracts needed parts. When it receives a request from Android application, the model will select matching teams and return corresponding JSON data.

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

For the "Player" page, *PlayerServlet.java* responds to the Android application with simple JSON formatted strings, which has been shown in **1.d**. Since I only use JSON formatted response strings, I don't need a JSP file for View.

For the "Team" page, *TeamServlet.java* responds to the Android application with simple JSON formatted strings, which has been shown in **1.d**. Since I only use JSON formatted response strings, I don't need a JSP file for View.

3. Handle error conditions

This part does not need to be documented

Actually, both the web servlets and Android application have been coded to deal with different errors, including error codes from third party API and correspondance with server side.

4. Log useful information

For each request/reply with the mobile phone, I have logged 8 pieces of information:

For each request/reply with the mobile phone, I have logged 8 pieces of information:

- Timestamp: the time when the request/reply is logged (*long*)
- ClientIP: the IP address of mobile phone client (*String*)
- DeviceModel: device model of mobile phone (*String*)
- RequestPath: the server API that is called by client (*String*)
- RequestParams: parameters included in client request (*String*)
- ThirdPartyLatency: latency (ms) between requesting to and receiving from third party API (*long*)
- StatusCode: the status code returned to mobile client (*int*)
- ResponseSize: the length of JSON data replied to mobile client (*int*)

An example of the request/reply log might be:

```
{
  "timestamp": 1763499518405,
  "clientIP": "0:0:0:0:0:0:0:1",
  "deviceModel": "Android",
  "requestPath": "/api/player",
  "requestParmas": "first_name=lebron",
  "thirdPartyLatency": 915,
  "statusCode": 200,
  "responseSize": 39
}
```

5. Store the log information in a database

In Mongo DB, the three shards are displayed as:

```
ac-tzhmnnq-shard-00-00.b5tx8sw.mongodb.net:27017
ac-tzhmnnq-shard-00-01.b5tx8sw.mongodb.net:27017
ac-tzhmnnq-shard-00-02.b5tx8sw.mongodb.net:27017
```

So the corresponding connection string should be:

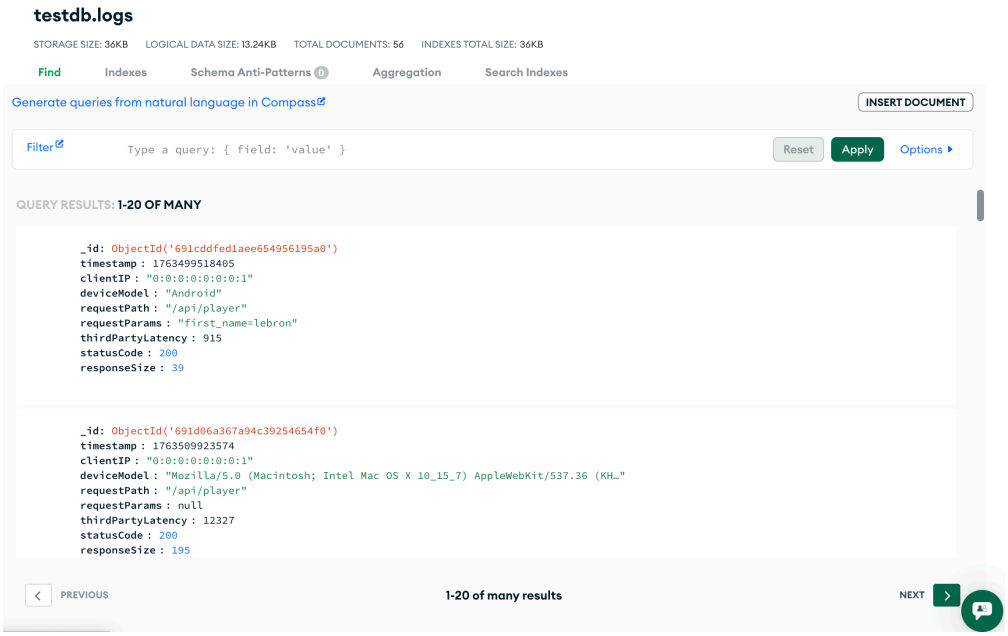
```
mongodb://yimingfu:mJt6njGpLAniRJa7@ac-tzhmnnq-shard-00-00.b5tx8sw.mongodb.net:27017,ac-tzhmnnq-shard-00-01.b5tx8sw.mongodb.net:27017,ac-tzhmnnq-shard-00-02.b5tx8sw.mongodb.net:27017/testdb?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1
```

Actually, I am able to use "+srv" URL to establish connection with Mongo DB, where the URL is:

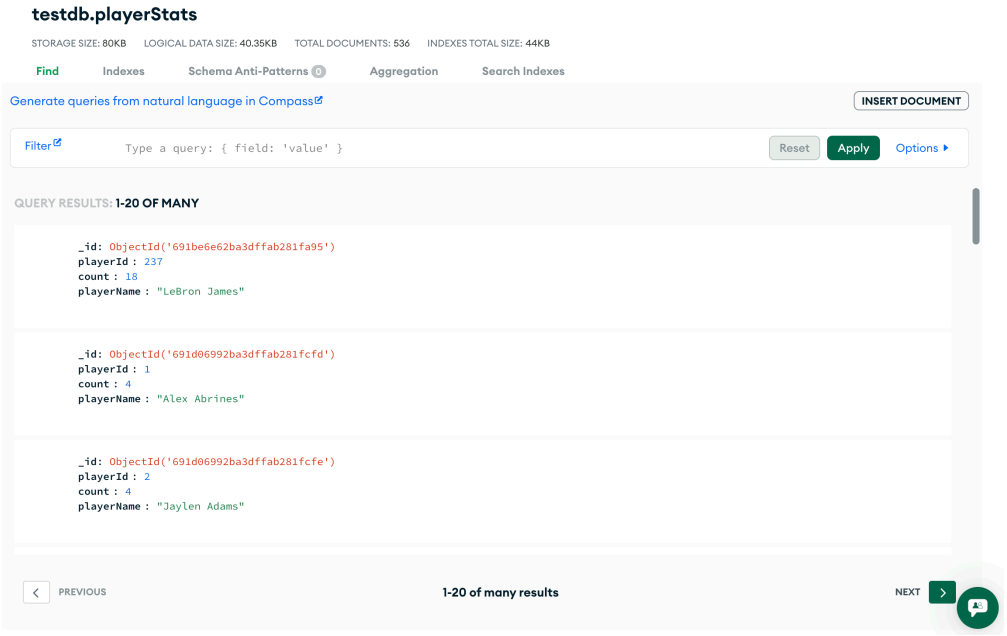
```
mongodb+srv://yimingfu:mJt6njGpLAniRJa7@cluster0.b5tx8sw.mongodb.net/?appName=Cluster0
```

After that, I just need to select the database "testdb" and collection "logs" / "playerStats" / "teamStats"

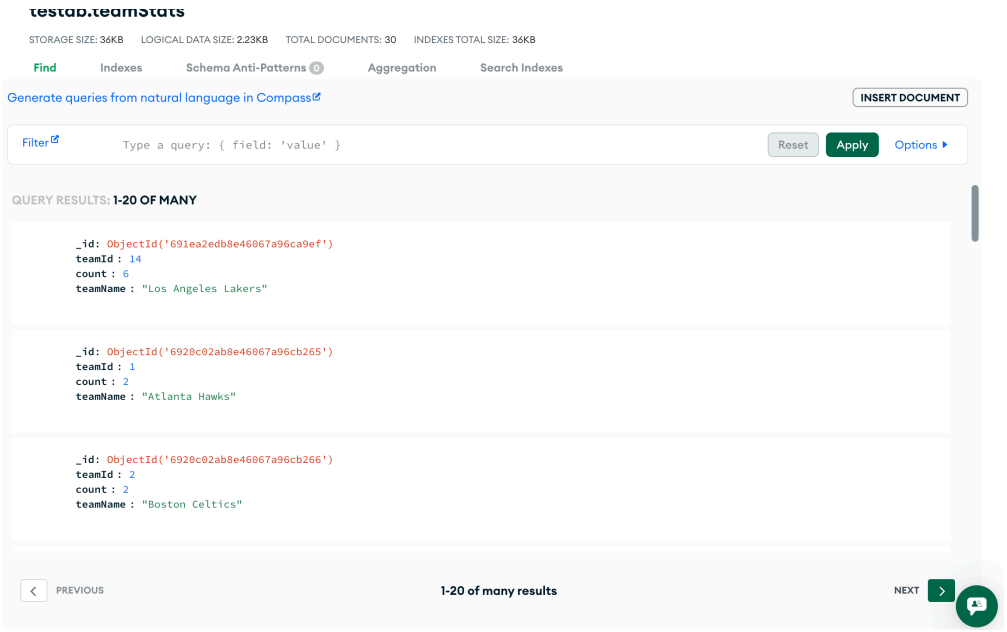
Below is a screenshot of samples from the "logs" collection, storing query logs information.



Below is a screenshot of samples from the "playerStats" collection, storing query times for players.



Below is a screenshot of samples from the "teamStats" collection, stroing query times for teams.



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

For the dashboard, I have realized the MVC structure:

- Model: *LogHelper.java*, which handles with creating operations analytics
- View: *dashboard.jsp*, which displays analytics and logs using HTML scripts and tables
- Controller: *DashboardServlet.java*, which fetches data from Model and forward to View (simply formatted and easily readable)
    - Note: there is another Controller called *LogsDataServlet* with the API path "/api/logs", which will return all the analytics and logs in JSON format. **This is just for internal tests!**

a. A unique URL addresses a web interface dashboard for the web service

The unique URL address for the web dashboard is:

```
https://crispy-chainsaw-7qj54556j9wfxrv9-8080.app.github.dev/dashboard
```

b. The dashboard displays at least 3 interesting operations analytics

Here I have provided 6 interesting operations analytics:

- API Request Frequency: visit counts for each different API ("/api/team" and "/api/player" in this case)
- Average Latency (ms): average milliseconds of request/reply latency from server side
- Error Rate: numbers of successful / errorneous requests from mobile app
- Top Devices: most popular device model from mobile apps
- Top Searched Players: top-5 searched players (appearing in response JSON)
- Top Searched Teams: top-3 searched teams (appearing in response JSON)

- Top searched teams: top 3 searched teams (appearing in response JSON)

Below is a screenshot of all the interesting operations analytics:

Server Dashboard

API Request Frequency

API	Count
/api/team	27
/api/player	30

Average Latency (ms)

API	Average Latency
/api/team	997.1111111111111
/api/player	2218.0666666666666

Error Rate

Success: 57

Bad Request: 0

Top Devices

Device	Count
Android	42
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	15

Top Searched Players

Player Name	Search Count
LeBron James	19
Kevin Durant	15
Kevin Love	12
Kevin Knox	12
Kevin Huerter	12

Top Searched Teams

Team Name	Search Count
Los Angeles Lakers	6
Houston Rockets	4
Atlanta Hawks	2

c. The dashboard displays formatted logs

All the logs are displayed in **tabled** views, with all fields and corresponding values. Below is a screenshot:

All Logs

Time	IP Address	Device Model	API Path	Request Params	Latency (ms)	Status	Response Size
1763499518405	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	915	200	39
1763509923574	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/player	null	12327	200	195
1763514536579	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/player	first_name=lebron	2431	200	39
1763528277150	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	1423	200	45
1763528394193	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	team=lakers	1393	200	1
1763532260438	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	2592	200	39
1763532269679	0:0:0:0:0:0:0:1	Android	/api/team	team=laker	472	200	1
1763532461061	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	478	200	45
1763532466813	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	team=lakers	469	200	1
1763532598420	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	2589	200	39
1763532604058	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	430	200	1
1763532653650	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	430	200	1
1763564619235	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	1220	200	45
1763565747115	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	1319	200	45
1763566019266	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	561	200	39
1763566025767	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	1300	200	1
1763566095618	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	728	200	39
1763566573481	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	1305	200	1
1763566743742	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	1300	200	1
1763566810889	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	1299	200	1
1763566821088	0:0:0:0:0:0:0:1	Android	/api/team	null	1302	200	45
1763566844540	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	1301	200	45
1763567563933	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	1112	200	1
1763567569969	0:0:0:0:0:0:0:1	Android	/api/team	null	1098	200	30
1763567589037	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	null	1098	200	30
1763568154595	0:0:0:0:0:0:0:1	Android	/api/team	team=clippers	1294	200	1
1763568169169	0:0:0:0:0:0:0:1	Android	/api/player	last_name=leonard	626	200	39
1763615460448	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	2315	200	39
1763615469433	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	466	200	1
1763700443293	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/player	null	12437	200	194
1763705049017	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/player	first_name=lebron	694	200	39
1763705129610	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/team	team=Lakers	696	200	1
1763705132361	0:0:0:0:0:0:0:1	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36	/api/player	first_name=lebron	461	200	39
1763753703662	0:0:0:0:0:0:0:1	Android	/api/player	first_name=lebron	2458	200	39
1763753798351	0:0:0:0:0:0:0:1	Android	/api/player	null	10367	200	195
1763754025990	0:0:0:0:0:0:0:1	Android	/api/team	null	569	200	30
1763754070042	0:0:0:0:0:0:0:1	Android	/api/team	team=lakers	573	200	1
1763754385830	0:0:0:0:0:0:0:1	Android	/api/player	first_name=kevin	949	200	38
1763754439430	0:0:0:0:0:0:0:1	Android	/api/player	first_name=kevin	748	200	38