**Project 4 – Emoji Application**
**By Lee Ge (Andrew ID: yge2)**

**Description:**

My application offers the user three different approaches to fetch an Emoji:

- **Approach 1**: prompt user to choose the category related to the Emoji. Eg: activities.
- **Approach 2**: prompt user to type a search word related to the Emoji. Eg: flags
- **Approach 3**: randomly generate an Emoji for the user. This approach does not take any user Input.

When the application launches, the user will be prompted with a main menu where they can choose one of approaches as described above.

Based on the user choices (will be passed to the web service as an Http request), eventually the output (an Emoji Hash code) is fetched and displayed from a third Party API called **EmojiHub**.

Here is 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: **Project4**

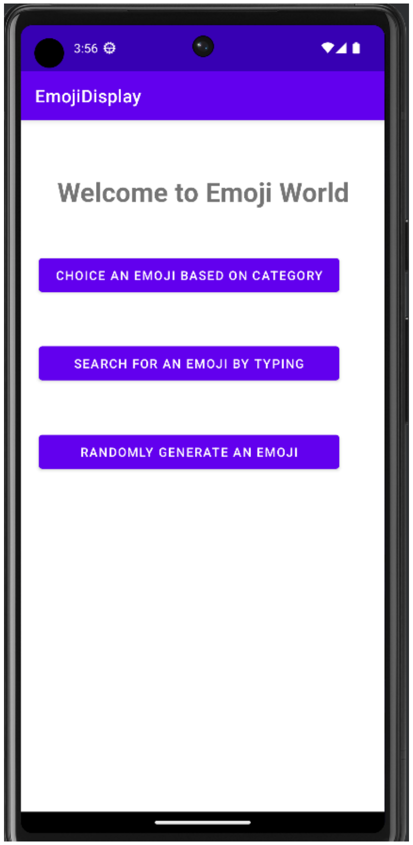
   a. **Has at least three different kind of views in your layout**

      My application uses TextView, EditText, Button, RadioButton, RadioGroup, and ProgressBar. Please refer to the following for details of how they are incorporated into the RelativeLayout.

      ○ activity_emoji_main.xml
      ○ activity_emoji_one.xml
      ○ activity_emoji_two.xml
      ○ activity_emoji_three.xml

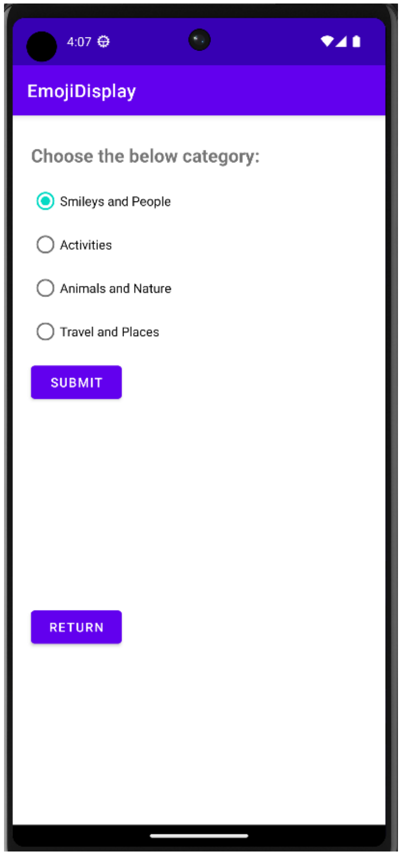
      Here a screenshot of the layout for **main menu** (activity_emoji_main.xml)

      *(Displayed in Pixel 6 API 33)*


3:56
EmojiDisplay
Welcome to Emoji World
CHOICE AN EMOJI BASED ON CATEGORY
SEARCH FOR AN EMOJI BY TYPING
RANDOMLY GENERATE AN EMOJI

**b. requires input from the user**

Here are screenshot of the user choosing approach 1, 2, 3 respectively:



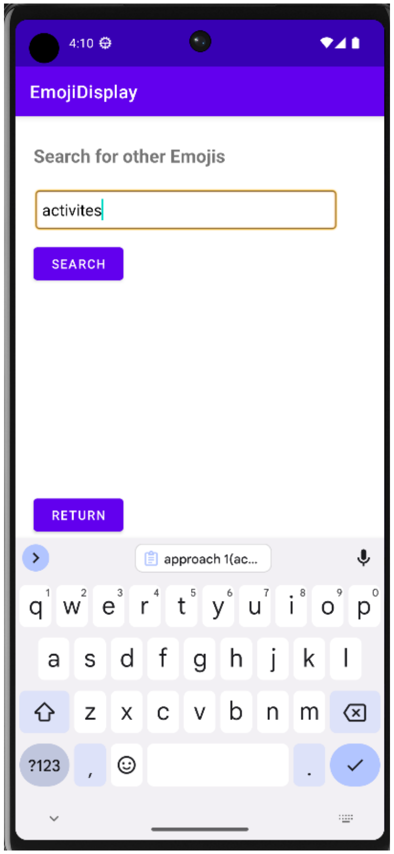
Choose the below category:

● Smileys and People

○ Activities

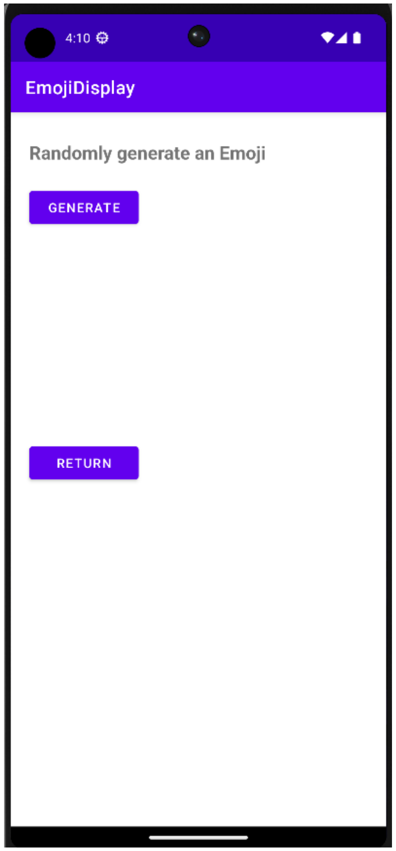
○ Animals and Nature

○ Travel and Places



Search for other Emojis

activites



Randomly generate an Emoji

- *activity_emoji_one.xml* *activity_emoji_two.xml* *activity_emoji_three.xml*

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

My application does an HTTP GET request in **EmojiModel.Java**. The HTTP request is:
"https://leege8-jubilant-carnival-qr5x6764g54f9xgx-8080.preview.app.github.dev/getEmoji/" + searchTag
Where searchTag is the user's input:

- approach 1: choice of the four categories
- approach 2: typed search term
- approach 3: searchTag is set to "random".

The search method makes this request of my web application, parses the returned Json message to get the hashcode of Emoji and return, display the Emoji in HTML format.

Simple workflow to aid understanding:

| User Input | message | Emoji Hashcode | HTML display | Emoji |
|---|---|---|---|---|
| Animals and Nature | { Json message….} | &#129409; | `<html>`<br>`<body>`<br>`<p>&#129409;</p>`<br>`</body>`<br>`</html>` | |

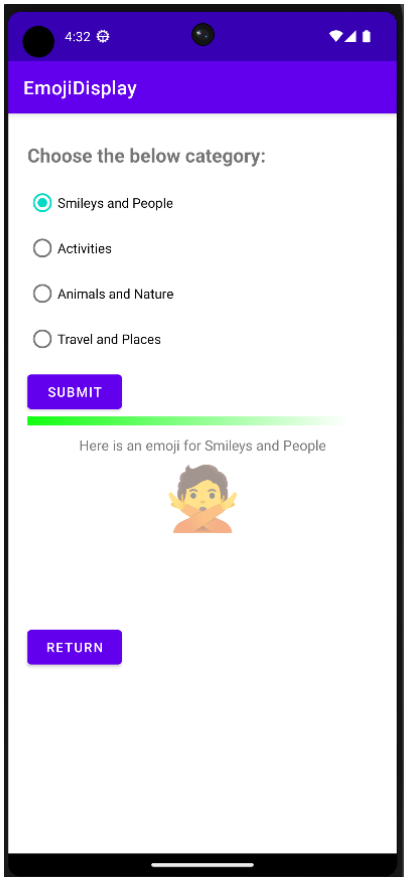
**d. Receives and Parses an XML or JSON formatted reply from the web service**

An example of the Json reply is:

{"name":"flag in hole","chosenHtmlCode":"\u0026#9971;","category":"activities","group":"activities","htmlCode":["\u0026#9971;"],"unicode":["U+26F3"]}

**e. Displays new information to the user**

Here are the screenshots after the Emoji has been returned from approach 1, 2, 3 in **b** respectively.



4:32

EmojiDisplay

Choose the below category:

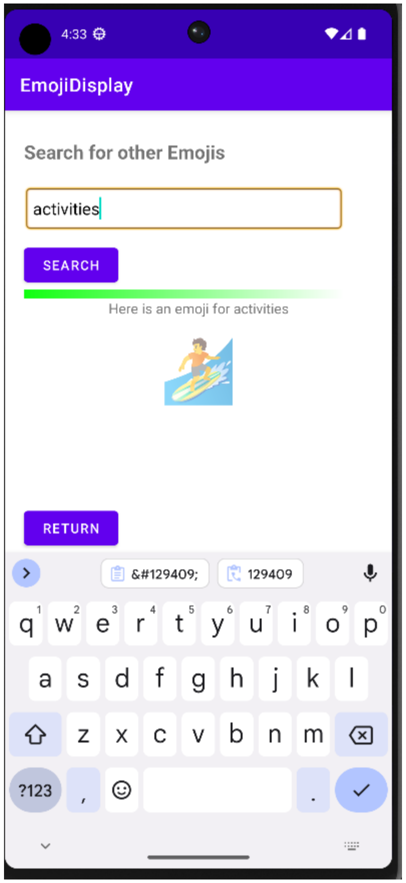
● Smileys and People

○ Activities

○ Animals and Nature

○ Travel and Places

Here is an emoji for Smileys and People



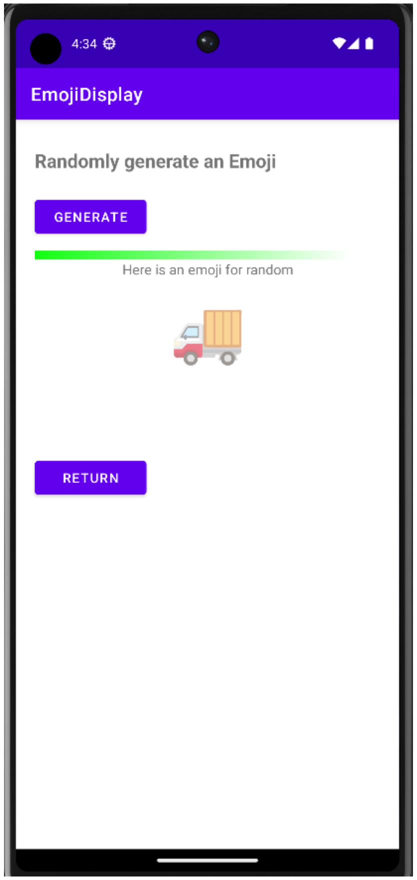
4:33

EmojiDisplay

Search for other Emojis

activities

Here is an emoji for activities



4:34

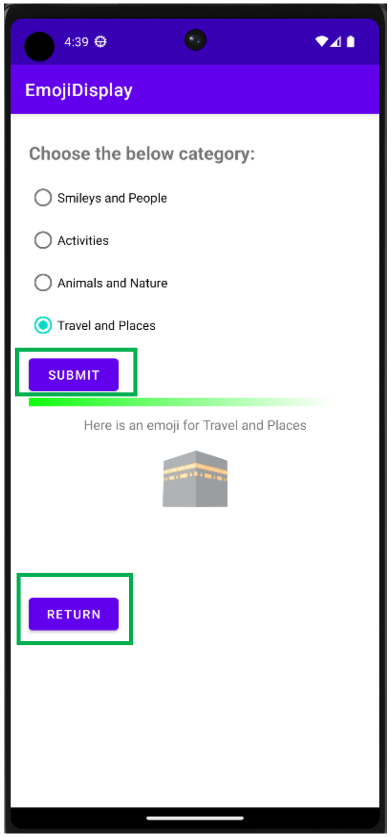
EmojiDisplay

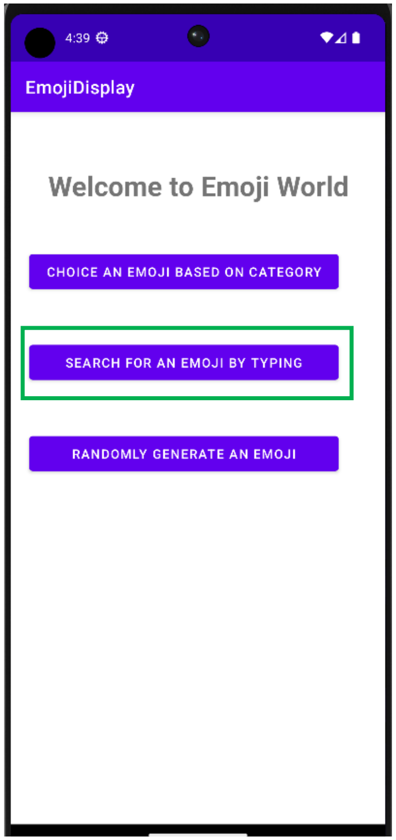
Randomly generate an Emoji

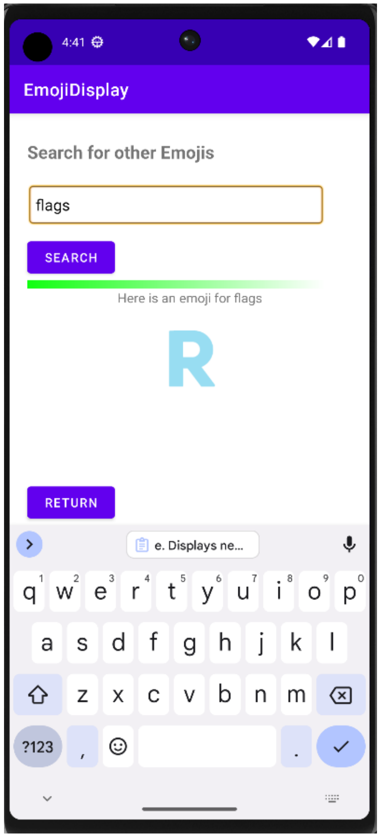
Here is an emoji for random

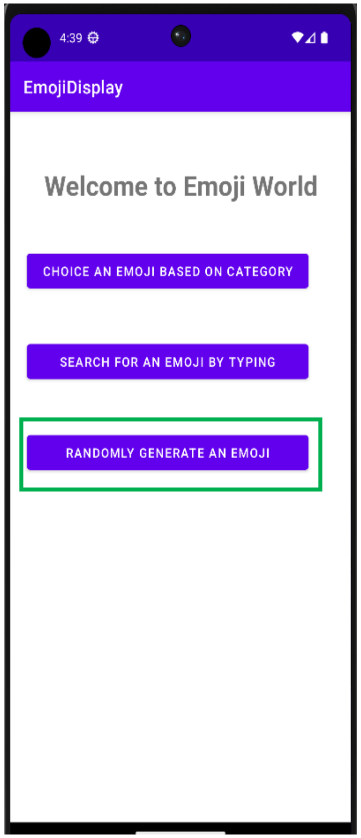
**f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)**

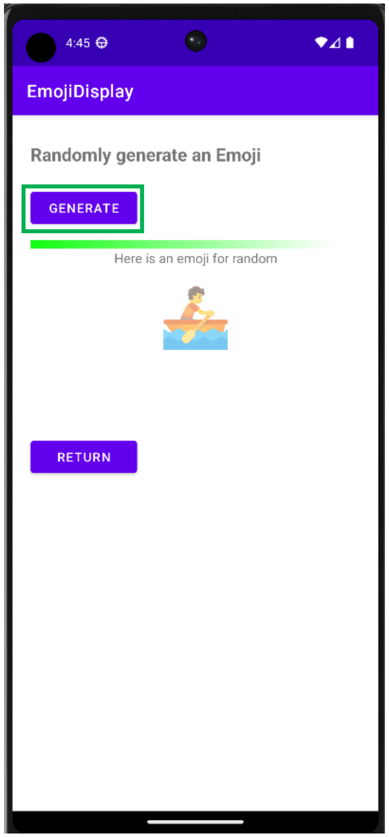
The user can select the same or other categories and hit submit; type in another search terms and hit search; hit generate to get random Emoji; hit return to go back to the main menu. Here are some examples:




4:39
EmojiDisplay
Welcome to Emoji World
CHOICE AN EMOJI BASED ON CATEGORY
SEARCH FOR AN EMOJI BY TYPING
RANDOMLY GENERATE AN EMOJI




4:39
EmojiDisplay
Welcome to Emoji World
CHOICE AN EMOJI BASED ON CATEGORY
SEARCH FOR AN EMOJI BY TYPING
RANDOMLY GENERATE AN EMOJI



2. **Implement a web application, deployed to GitHub**

   The URL of my web service deployed to GitHub is:

   https://leege8-jubilant-carnival-qr5x6764g54f9xgx-8080.preview.app.github.dev/

   The Github repository is: **project-4-task-2-leege8**

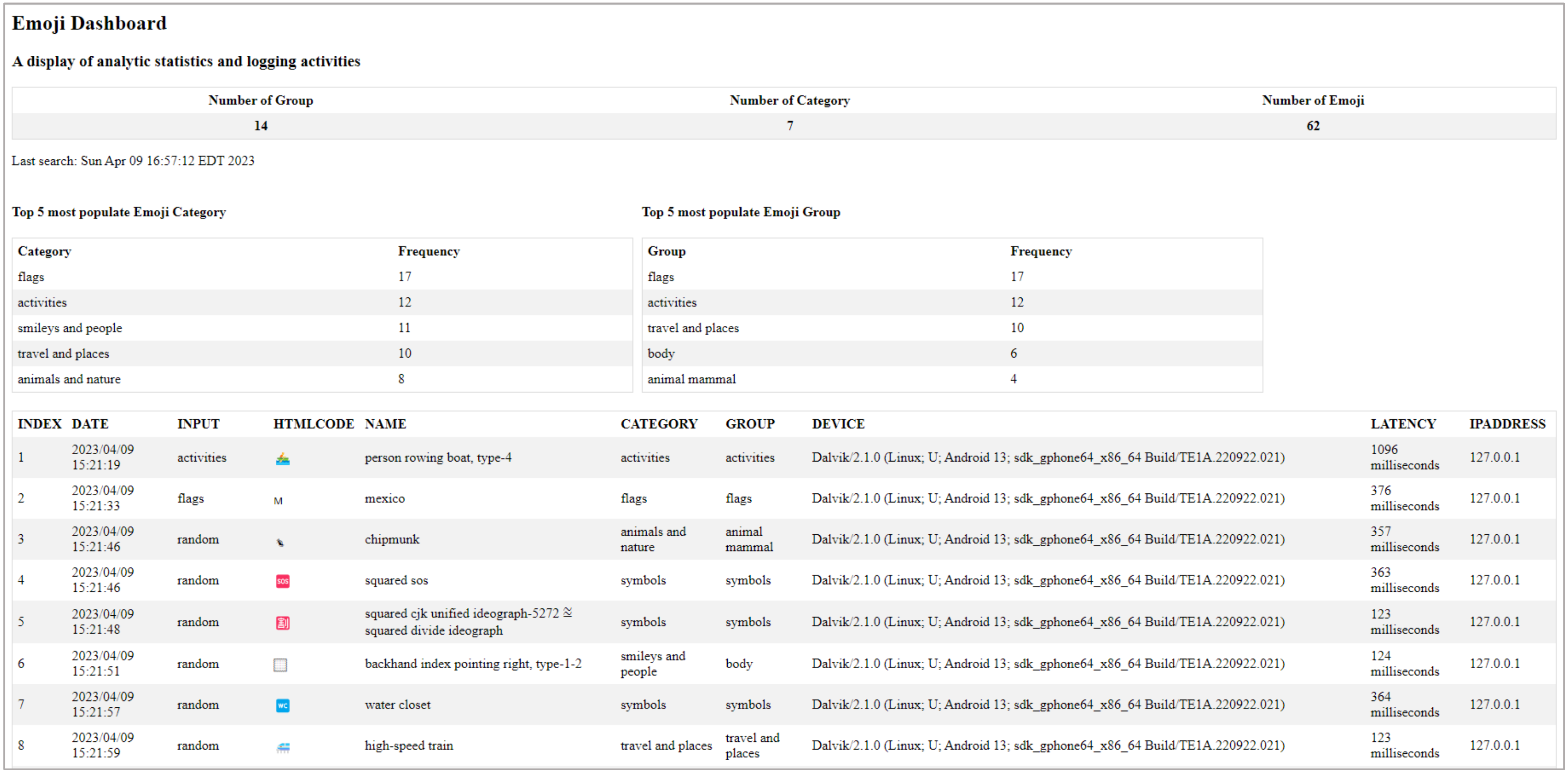
   https://github.com/CMU-Heinz-95702/project-4-task-2-leege8.git

   a. Using an HttpServlet to implement a simple (can be a single path) API In my web app project:
      Model: EmojiModel.Java
      Controller: EmojiServlet.Java
      View: directly writing response back to the mobile application in a JSON format of my own design:

```
{"name":"owl","chosenHtmlCode":"\u0026#129417;","category":"animals and nature","group":"animal bird","htmlCode":["\u0026#129417;"], "unicode":["U+1F989"]}
```

3. **Handle error conditions –** *Does not need to be documented.*
4. **Log useful information**
   - **User IP address**: to avoid potential security threats; to better identify potential target group of users.
   - **Search Date and Time**: to track when the searches are made most frequently.
   - **User Input**: to track what category the user is most interested in and searched a lot of times.
   - **Response**, including name, category, and group: to assess if the users will be satisfied with such response or not.
   - **Latency**: how long it took for a user to get the output, which is an emoji. We should optimize the mobile application if there is high latency.
   - **Device**: track the devices the user used for the mobile application.
   - **Index**: a series of constructive integers starting from 1. 1 indicate the first search made by the user. This helps to track the details of the latest search made.
5. **Store the log information in a database – Given your Atlas connection string with the three shards**
   - **connectionURL** = mongodb://yge2:13813659155q@ac-p3oagra-shard-00-01.j81zkga.mongodb.net:27017,ac-p3oagra-shard-00-02.j81zkga.mongodb.net:27017,ac-p3oagra-shard-00-00.j81zkga.mongodb.net:27017/test?w=majority&retryWrites=true&tls=true&authMechanism=SCRAM-SHA-1
     (*three shards included*)
   - **region clusters:**
     - ac-p3oagra-shard-00-01.j81zkga.mongodb.net:27017
     - ac-p3oagra-shard-00-02.j81zkga.mongodb.net:27017
     - ac-p3oagra-shard-00-00.j81zkga.mongodb.net:27017
   - **cluster name:** Project4Cluster
   - **passwords:** 13813659155qQ@
   - **DataBase Name:** EmojiDatabase
   - **Collection Name:** EmojiCollection

## 6. Display operations analytics and full logs on a web-based dashboard – provide a screenshot



# Emoji Dashboard

### A display of analytic statistics and logging activities

| Number of Group | Number of Category | Number of Emoji |
|---|---|---|
| **14** | **7** | **62** |

Last search: Sun Apr 09 16:57:12 EDT 2023

**Top 5 most populate Emoji Category**

| Category | Frequency |
|---|---|
| flags | 17 |
| activities | 12 |
| smileys and people | 11 |
| travel and places | 10 |
| animals and nature | 8 |

**Top 5 most populate Emoji Group**

| Group | Frequency |
|---|---|
| flags | 17 |
| activities | 12 |
| travel and places | 10 |
| body | 6 |
| animal mammal | 4 |

| INDEX | DATE | INPUT | HTMLCODE | NAME | CATEGORY | GROUP | DEVICE | LATENCY | IPADDRESS |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2023/04/09 15:21:19 | activities | 🚣 | person rowing boat, type-4 | activities | activities | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 1096 milliseconds | 127.0.0.1 |
| 2 | 2023/04/09 15:21:33 | flags | M | mexico | flags | flags | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 376 milliseconds | 127.0.0.1 |
| 3 | 2023/04/09 15:21:46 | random | 🐿 | chipmunk | animals and nature | animal mammal | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 357 milliseconds | 127.0.0.1 |
| 4 | 2023/04/09 15:21:46 | random | 🆘 | squared sos | symbols | symbols | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 363 milliseconds | 127.0.0.1 |
| 5 | 2023/04/09 15:21:48 | random | 🈹 | squared cjk unified ideograph-5272 ≊ squared divide ideograph | symbols | symbols | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 123 milliseconds | 127.0.0.1 |
| 6 | 2023/04/09 15:21:51 | random | □ | backhand index pointing right, type-1-2 | smileys and people | body | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 124 milliseconds | 127.0.0.1 |
| 7 | 2023/04/09 15:21:57 | random | 🚾 | water closet | symbols | symbols | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 364 milliseconds | 127.0.0.1 |
| 8 | 2023/04/09 15:21:59 | random | 🚄 | high-speed train | travel and places | travel and places | Dalvik/2.1.0 (Linux; U; Android 13; sdk_gphone64_x86_64 Build/TE1A.220922.021) | 123 milliseconds | 127.0.0.1 |