

Name: Sanjana Rinke
Andrew ID: srinke

Project 4-Task 2

DESCRIPTION-

My application takes a word input from the user. It also asks user to enter functionality like: Meanings, Origin, Synonym, Antonym, Examples. Depending on user input the response is fetched from <https://dictionaryapi.dev/> API and response is shown. Task 2 builds on task 1 and includes logging and database and analytics functionality

1. Implement native android application

The name of my android appl project is **Project4Task1**

NOTE: Since the name of my android application and web service is the same, for the purpose of submission, I have placed my entire android code in a folder named Project4Android.

Folder structure Submitted:

Project4Task1Writeup.pdf

Project4Task1Writeup.pdf

Project4Task1-> Task 1 web server code

Project4Task2-> Task 2 web server code with logging functionality

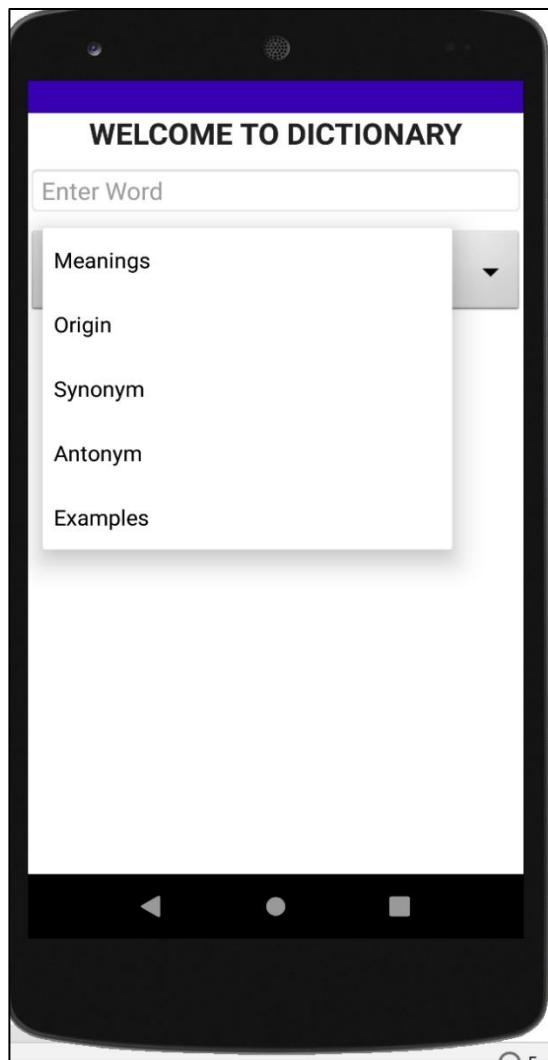
Project4Android-> Project4Task1 (Android studio code to be graded)

Android project has URL for Task 2 Web service configured, and Task-1 web service URL commented out

1.1. Has 5 different kinds of views- TextView, EditTextView, DropDown View (Spinner), ButtonView, ListView. See content_main.xml for details of how they are incorporated in linear layout.

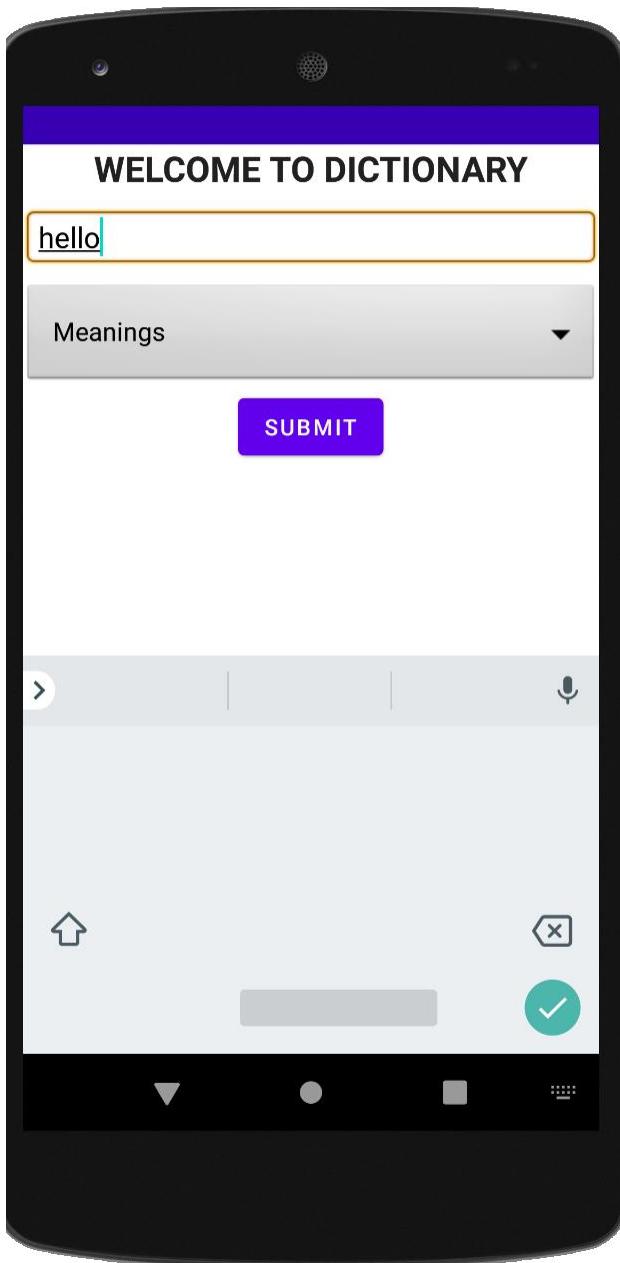
Here is a screenshot of layout before data being fetched.

Name: Sanjana Rinke
Andrew ID: srinke



Name: Sanjana Rinke
Andrew ID: srinke

1.2. Requires 2 input from user- word and the functionality. The result will fetch the meanings of word hello.



Name: Sanjana Rinke
Andrew ID: srinke

1.3. Makes HTTP request to web service using HTTP GET

My application does HTTP GET request in SearchWord.java

The base URL for my task 1 web server is: "<https://fast-mesa-20310.herokuapp.com/>"

The HTTP Requests for various functionalities are-

"<https://fast-mesa-20310.herokuapp.com/>" + "getExamples?word=" + searchTerm

"<https://fast-mesa-20310.herokuapp.com/>" + "getMeanings?word=" + searchTerm

"<https://fast-mesa-20310.herokuapp.com/>" + "getOrigin?word=" + searchTerm

"<https://fast-mesa-20310.herokuapp.com/>" + "getSynonym?word=" + searchTerm

"<https://fast-mesa-20310.herokuapp.com/>" + "getAntonym?word=" + searchTerm

1.4. Receives and parses JSON response from web service

In my application all the requested functionality is served on different URL catering to that specific functionality, my response format is a JSON Array with all necessary values.

Sample JSON Response

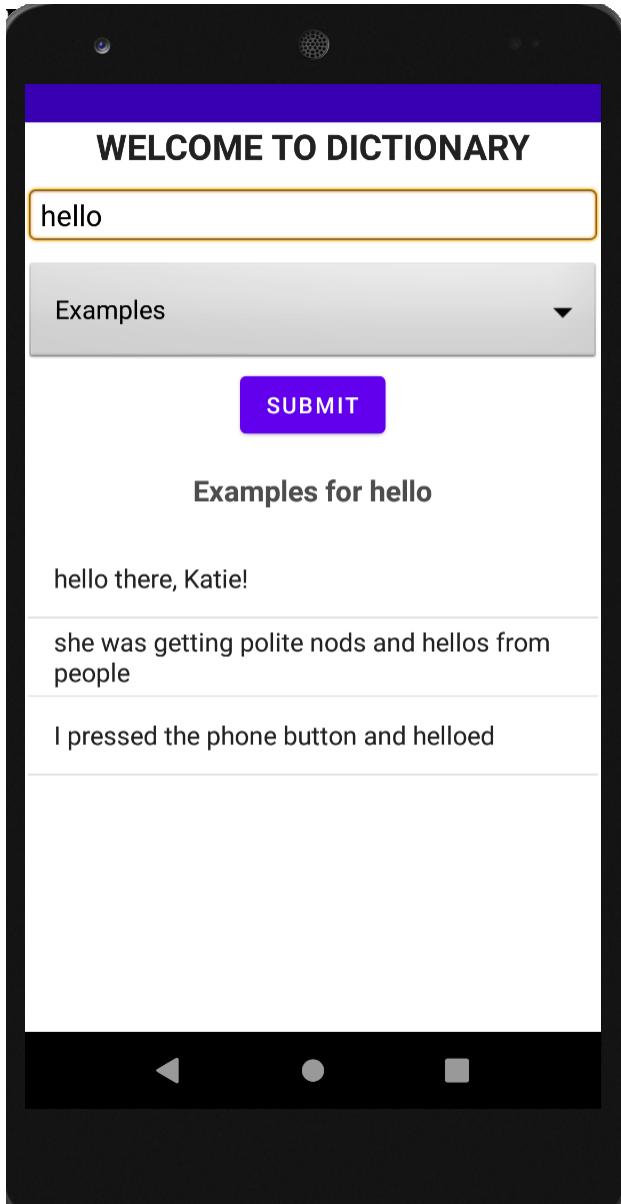
Following are 2 sample responses for different URL URL:

<https://fast-mesa-20310.herokuapp.com/getOrigin?word=hello>

Sending Origin for Hello as JSON Array: ["early 19th century: variant of earlier hollo; related to holla."]

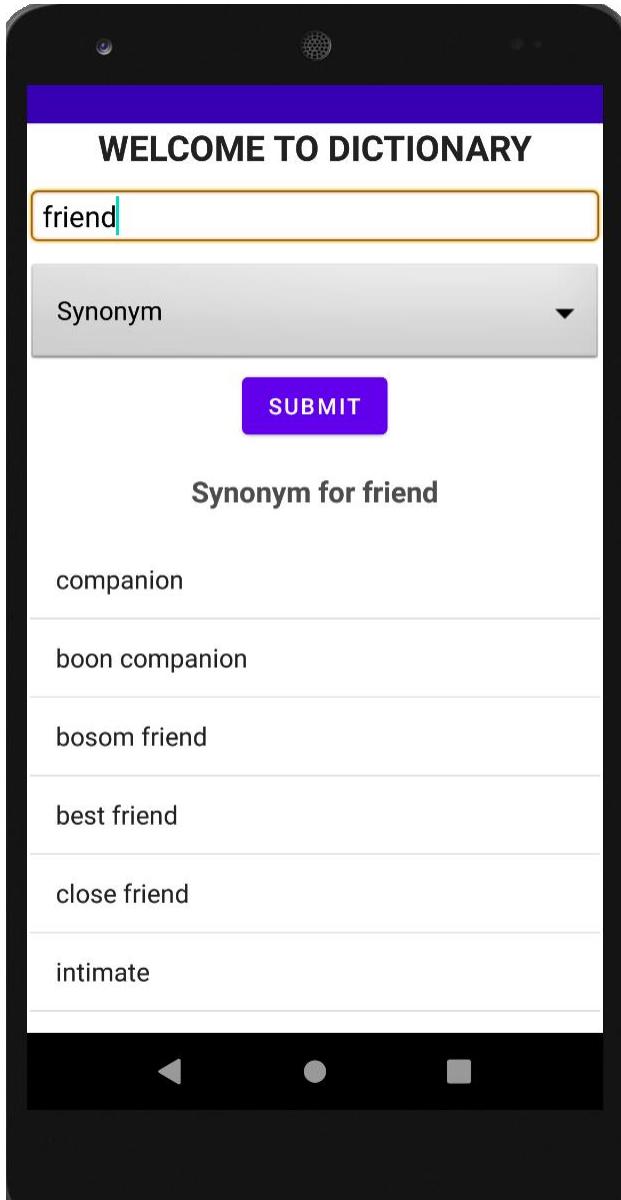
Name: Sanjana Rinke
Andrew ID: srinke

1.5. Displays information to the user



Name: Sanjana Rinke
Andrew ID: srinke

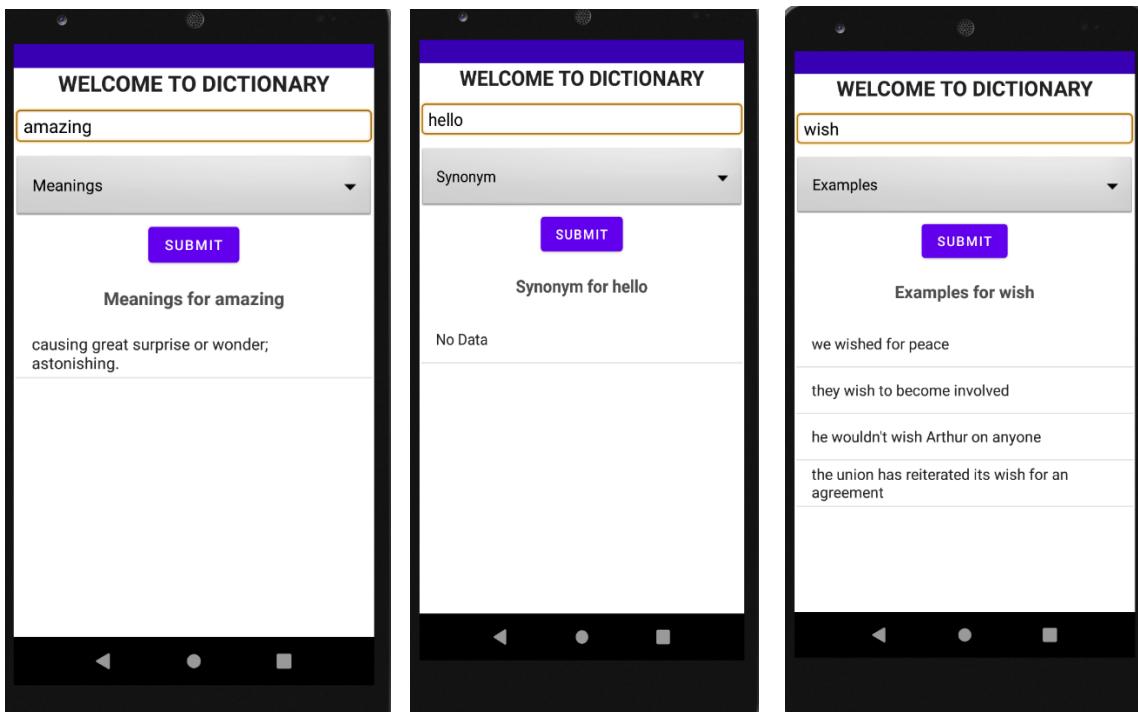
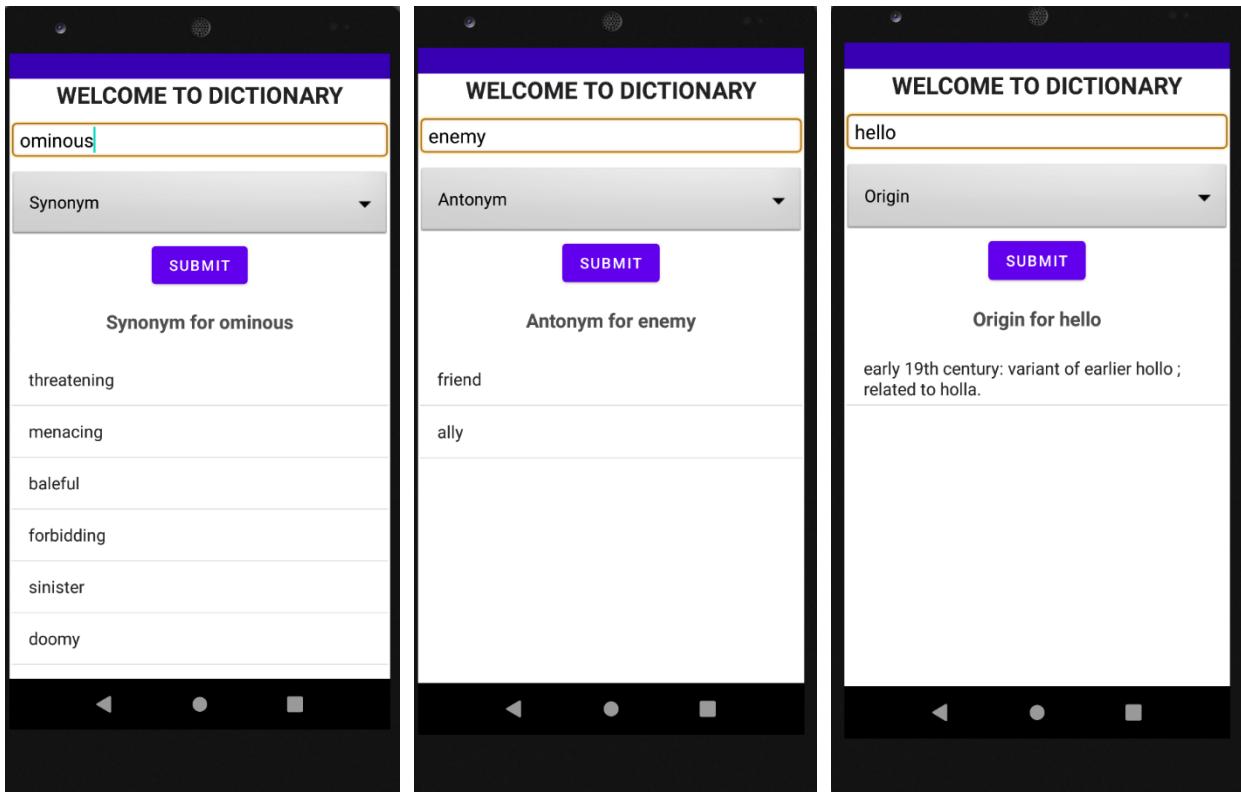
1.6. Is repeatable (User can use it without repeatedly restarting it)



Name: Sanjana Rinke

Andrew ID: srinke

Android UI Screen shots:



Name: Sanjana Rinke
Andrew ID: srinke

2. Implement web service deployed on Heroku with logging, dashboard and analytics functionality

The URL of my web service is: <https://fast-mesa-20310.herokuapp.com/>

Project directory name is Project4Task2

2.1. Using an HttpServlet to implement a simple (5 Paths) API

In my web app project:

Model: Dictionary.java

Controller: DictionaryServlet.java

View: dashboard.jsp

2.2. Receives an HTTP request from the native Android application.

DictionaryServlet.java receives the HTTP GET request with the argument "word". It passes this search word on to the model.

2.3. Executes business logic appropriate to your application

Dictionary.java makes an HTTP request to an external API to fetch information:

- 1) 5 different URL cater to various functionalities requested. Each URL will give a call internally to a 3rd party API: "<https://api.dictionaryapi.dev/api/v2/entries/en/>" + word.
 - a. The 3rd party API only takes word as a parameter and gives all information about the word as a single JSON.
 - b. Sample response:

```
[{"word":"hello","phonetic":"hə'ləʊ","phonetics":[{"text":"hə'ləʊ","audio":"//ssl.gstatic.com/dictionary/static/sounds/20200429/hello--_gb_1.mp3"}, {"text":"hə'ləʊ"}],"origin":"early 19th century: variant of earlier hollo; related to holla.","meanings":[{"partOfSpeech":"exclamation","definitions":[{"definition":"used as a greeting or to begin a phone conversation."}]}]}]
```

- 2) Web service specific URL parses the json response and extracts the parts it needs to respond to the Android application.

Name: Sanjana Rinke
Andrew ID: srinke

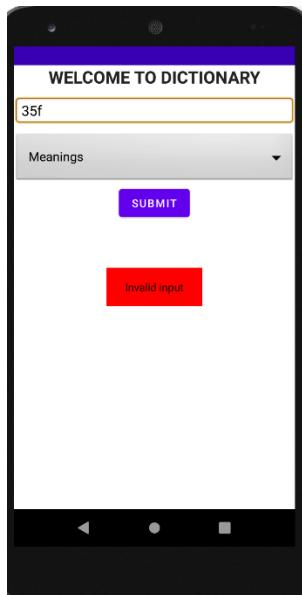
2.4. Replies to the Android application with an XML or JSON formatted response

Response for: <https://fast-mesa-20310.herokuapp.com/getOrigin?word=hello>

["early 19th century: variant of earlier hollo; related to holla."]

3) All the given error conditions are handled-

- Non-Alphabet input



Name: Sanjana Rinke
Andrew ID: srinke

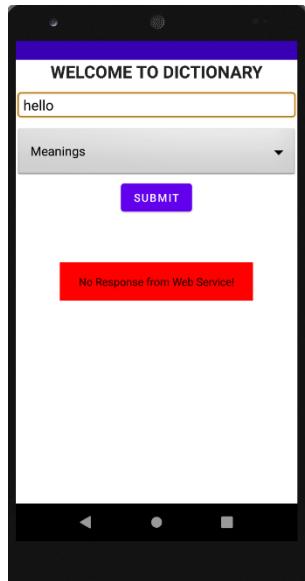
- Blank Input



Name: Sanjana Rinke
Andrew ID: srinke

- Invalid server-side input (regardless of mobile app input validation)
- Mobile app network failure, unable to reach server
- Third-party API unavailable
- Third-party API invalid data

All the above conditions are handled separately in the code, but gives out a generic message as below on UI



2.5. Log useful information

Here 7 pieces of information is being logged-

- 1) Functionality requested- taken from the request packer URL
- 2) Word requested- taken from the request parameter
- 3) Web service latency- maintained the request, response time to calculate latency
- 4) Web service response code- Taken from request packet
- 5) 3rd party API response code
- 6) 3rd party API latency- Maintained request, response time to calculate latency
- 7) Device Type- Taken from request packet's header

Name: Sanjana Rinke
Andrew ID: srinke

2.6. Store log information in database

My web service deployed on Heroku can connect to the mongoDB on atlas. Following are database details:

CONNECT- For connection following details are used to generate a connection URL. The connection URL of my database is-

`mongodb+srv://dbuser:1234@cluster0.7p5vm.mongodb.net/project4?retryWrites=true&w=majority`

DB Name: project4

Collection Name: metrics

STORE- 7 pieces of information listed above are stored in the DB as a document with a schema shown below-

Sample schema:

```
{"_id": {"$oid": "61916d305b3e1b22ec22ec57"}, "functionality": "Synonyms", "deviceType": "Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)", "word": "ominous", "WebServerlatency": {"$numberLong": "180"}, "extAPILatency": {"$numberLong": "157"}, "responseStatus": {"$numberInt": "200"}, "extApiResCode": {"$numberInt": "200"}}
```

RETRIEVE- The information is retrieved from the DB and displayed on dashboard.jsp after formatting it in appropriate format.

The screenshot shows a web application titled "Dictionary App Usage Statistics". The page contains several text-based statistics and a table of logs.

Statistics:

- Most searched word:: hello
- Most Requested Functionality:: Find Synonyms
- Device sending most requests:: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
- 3rd Party API Avg response time:: 175 ms
- Web server Avg response time:: 183 ms
- Number of Error Requests on Web server:: 0
- Number of Correct Requests on 3rd Party API:: 5

All logs

Functionality	Word	Web Service Latency(ms)	Web Service Response Code	3rd Party API Response code	3rd Party API Latency(ms)	Device
Synonyms	ominous	180	200	200	157	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Antonyms	enemy	298	200	200	297	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Origin	hello	44	200	200	36	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Meanings	amazing	317	200	200	312	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Synonyms	hello	77	200	200	75	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)

Name: Sanjana Rinke
Andrew ID: srinke

Dashboard.jsp



```
ArrayList<JSONObject> list = (ArrayList<JSONObject>) request.getAttribute("allLogs");
// print the information about every log of the list
for (JSONObject logs : list) {%
<tr>
<td><%=logs.get("functionality")%>
</td>
<td><%
    if (logs.get("word").toString().equalsIgnoreCase("null")) {%
        NA
    } else%><%=logs.get("word")%>
</td>
<td><%
    if (Long.parseLong(logs.get("WebServerlatency").toString()) == Long.MIN_VALUE) {%
        NA
    } else%><%=logs.get("WebServerlatency")%>
</td>
<td><%=logs.get("responseStatus")%>
```

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

- A unique URL has a web interface dashboard for the web service.

The Heroku URL is configured to open dashboard directly after launching. Alternatively, there is a dedicated URL /getAnalytics which will give same result.

Following URL is used to access dashboard-

<https://fast-mesa-20310.herokuapp.com/>

OR

<https://fast-mesa-20310.herokuapp.com/getAnalytics>

- The dashboard displays at least 3 interesting operations analytics.

My web app displays 7 operational analytics-

The information stored in database is used to analyze following parameters

A HashMap is used to find most searched word, most requested functionality, and most popular device.

Name: Sanjana Rinke
Andrew ID: srinke

Most searched word: hello
Most Requested Functionality: Find Synonyms
Device sending most requests: Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
3rd Party API Avg response time: 175 ms
Web server Avg response time: 183 ms
Number of Error Requests on Web server: 0
Number of Correct Requests on 3rd Party API: 5

c. The dashboard displays **formatted** full logs.

The information stored in database is retrieved to display the logs in tabular format

All logs						
Functionality	Word	Web Service Latency(ms)	Web Service Response Code	3rd Party API Response code	3rd Party API Latency(ms)	Device
Synonyms	ominous	180	200	200	157	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Antonyms	enemy	298	200	200	297	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Origin	hello	44	200	200	36	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Meanings	amazing	317	200	200	312	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)
Synonyms	hello	77	200	200	75	Dalvik/2.1.0 (Linux; U; Android 9; AOSP on IA Emulator Build/PSR1.180720.117)

4. Deploy the web service to Heroku

The web service is deployed to Heroku. The URL for this is- <https://fast-mesa-20310.herokuapp.com/>

The following web service has logging, database and analytics functionality in addition to task 1