Project 4 Task 2 – Stock Data App, by Verissa Owusu
(AndrewID: vowusu)

Demo: https://youtu.be/v6w6_tzZSh8

Description:

My application takes a search string, which is a stock ticker, from the user, and uses it to fetch and display data about the stocks from two 3rd party APIs.

Here is how my application meets the task requirements

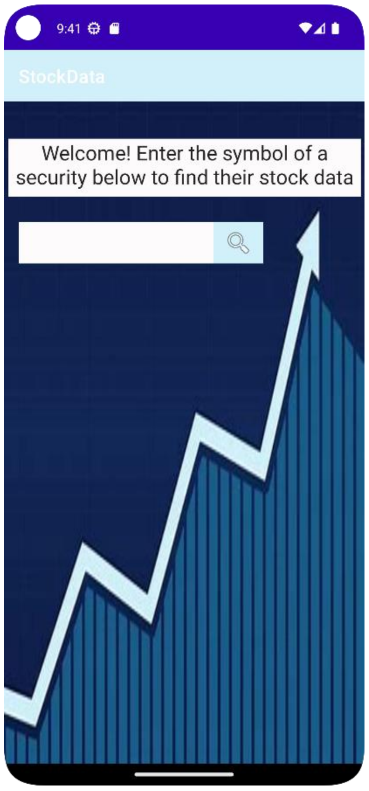
1. Implement a native Android application

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

a. Has at least three different kinds of views in your Layout (TextView, EditText, ImageView,etc.)

My application uses TextView, EditText, Button, and ImageButton, GraphView. See activity_main.xml and activity_results.xml for details of how they are incorporated into the ConstraintLayout.

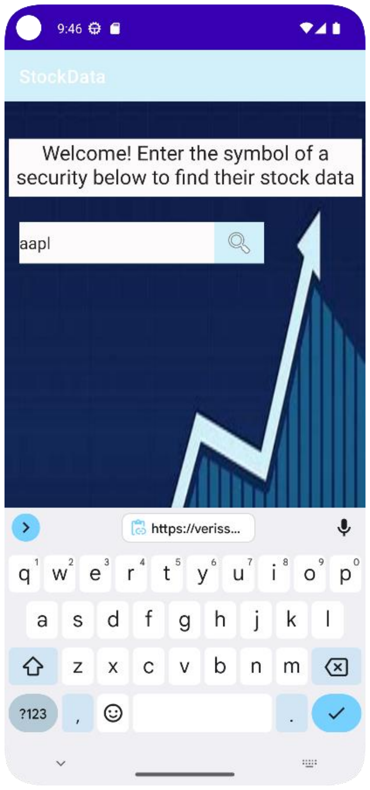
Here is a screenshot of the layout before the picture has been fetched.



Welcome! Enter the symbol of a security below to find their stock data

b. Requires input from the user

Here is a screenshot of the user searching for stock data of a symbol “aapl”


9:46
StockData
Welcome! Enter the symbol of a security below to find their stock data
aapl
https://veriss...

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

My application does an HTTP GET request in SearchStock.java. The HTTP request is:

private static final String SEARCH_ENDPOINT = "https://verissa-turbo-giggle-7rxgvgr9grphrp64-8080.preview.app.github.dev/search";

String urlWithParams = SEARCH_ENDPOINT + "?searchTerm=" + URLEncoder.encode(searchTerm, "UTF-8");

The search method makes this request to my web server, and my web server parses the returned data from the two URLs into a JSON. My search method in the app then parses the JSON data to create a Stock object and then sends the stock object to the main activity which then sends it to the results activity to update the UIs

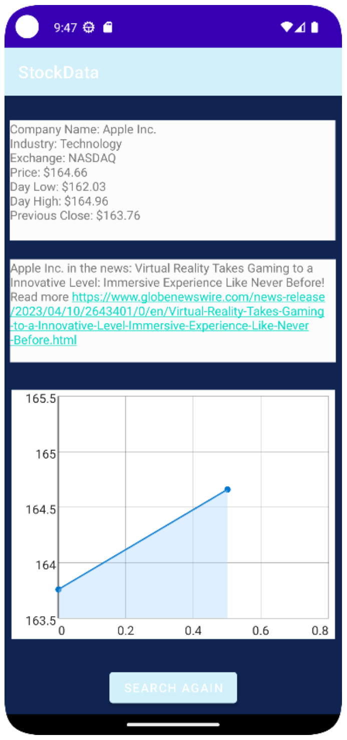
d. Receives and parses an JSON formatted reply from the web service

An example of the JSON reply is:

```
{
   "name": "Apple Inc.",
   "dayHigh": 164.96,
   "dayLow": 162.03,
   "previousClose": 163.76,
   "price": 164.66,
   "title": "Virtual Reality Takes Gaming to a Innovative Level: Immersive Experience Like Never
Before!",
   "url": "https://www.globenewswire.com/news-release/2023/04/10/2643401/0/en/Virtual-
Reality-Takes-Gaming-to-a-Innovative-Level-Immersive-Experience-Like-Never-Before.html",
   "exchange": "NASDAQ",
   "industry": "Technology"
}
```

e. Displays new information to the user

Here is the screen shot after the data has been returned.

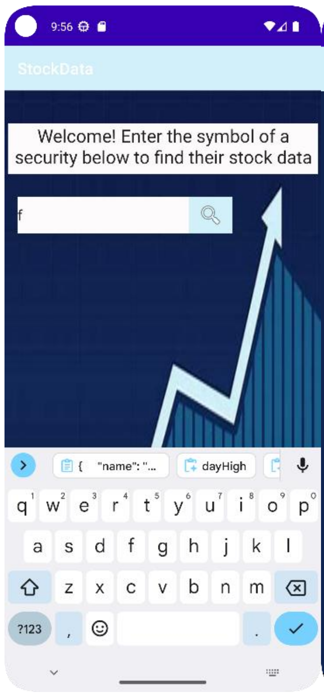


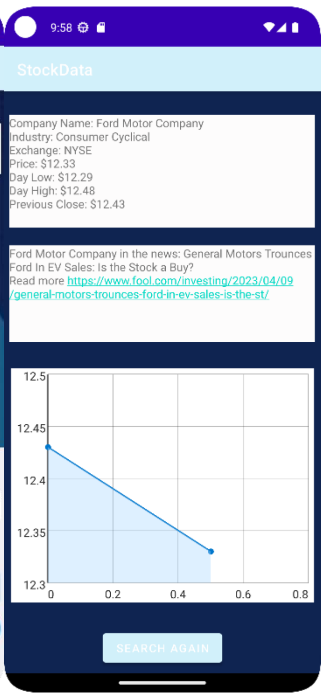
Company Name: Apple Inc.
Industry: Technology
Exchange: NASDAQ
Price: $164.66
Day Low: $162.03
Day High: $164.96
Previous Close: $163.76

Apple Inc. in the news: Virtual Reality Takes Gaming to a Innovative Level: Immersive Experience Like Never Before! Read more https://www.globenewswire.com/news-release/2023/04/10/2643401/0/en/Virtual-Reality-Takes-Gaming-to-a-Innovative-Level-Immersive-Experience-Like-Never-Before.html

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

The user hits the search again button and type in another search term and hit the image button. Here is an example of having typed in "f".


9:56
StockData
Welcome! Enter the symbol of a security below to find their stock data
f
{ "name": "...
dayHigh
?123



Company Name: Ford Motor Company
Industry: Consumer Cyclical
Exchange: NYSE
Price: $12.33
Day Low: $12.29
Day High: $12.48
Previous Close: $12.43

Ford Motor Company in the news: General Motors Trounces Ford In EV Sales: Is the Stock a Buy?
Read more https://www.fool.com/investing/2023/04/09/general-motors-trounces-ford-in-ev-sales-is-the-st/

2. Implement a web application, deployed to Codespaces

The URL of my web service deployed to Heroku is: serene-plateau-27771

The project directory name is Project4Task1.

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

```
@WebServlet(value = {"/search", "/dashboard"})
public class HelloServlet extends HttpServlet {
```

b. Receives an HTTP request from the native Android application

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    String path = request.getServletPath();
    // Retrieve the search term from the HTTP request
    if(path.contains("/search")) {
        String searchTerm = request.getParameter("searchTerm");
```

```
} else if (path.contains("/dashboard")) {
```

c. Executes business logic appropriate to your application

```
URL aletheiaApiUrl = new URL(ALETHEIA_API_URL + searchTerm);
HttpURLConnection aletheiaConnection = (HttpURLConnection)
aletheiaApiUrl.openConnection();
aletheiaConnection.setRequestProperty("Accept-Version", "2");
aletheiaConnection.setRequestMethod("GET");
```

```
// Make an HTTP request to the MarketAux API with the search term as a query
parameter
URL marketauxApiUrl = new URL(MARKETAUX_API_URL + searchTerm);
HttpURLConnection marketauxConnection = (HttpURLConnection)
marketauxApiUrl.openConnection();
marketauxConnection.setRequestMethod("GET");
```

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

```
e.    // Extract data from the Aletheia API JSON response
String shortName = aletheiaJsonObject.get("ShortName").getAsString();
float dayHigh = aletheiaJsonObject.get("DayHigh").getAsFloat();
float dayLow = aletheiaJsonObject.get("DayLow").getAsFloat();
float previousClose = aletheiaJsonObject.get("PreviousClose").getAsFloat();
float price = aletheiaJsonObject.get("Price").getAsFloat();


// Extract data from the Marketaux API JSON response
JsonArray data = (JsonArray) marketauxJsonObject.get("data");
JsonObject firstItem = data.get(0).getAsJsonObject();
String title = firstItem.get("title").getAsString();
String url = firstItem.get("url").getAsString();
JsonArray data2 = firstItem.get("entities").getAsJsonArray();
JsonObject data21 = data2.get(0).getAsJsonObject();
String exchange = data21.get("exchange").getAsString();
String industry = data21.get("industry").getAsString();

// Create a new JsonObject to store all the extracted data
JsonObject jsonResponse = new JsonObject();

// Add all the extracted data as properties to the JsonObject
jsonResponse.addProperty("name", shortName);
jsonResponse.addProperty("dayHigh", dayHigh);
jsonResponse.addProperty("dayLow", dayLow);
jsonResponse.addProperty("previousClose", previousClose);
jsonResponse.addProperty("price", price);
jsonResponse.addProperty("title", title);
jsonResponse.addProperty("url", url);
jsonResponse.addProperty("exchange", exchange);
jsonResponse.addProperty("industry", industry);

// Use PrintWriter object to write the JsonObject as a response to the client
PrintWriter out = response.getWriter();
out.print(jsonResponse.toString());
```

To document the rest of the requirements:

3. Handle error conditions - Does not need to be documented.
4. Log useful information - Itemize what information you log and why you chose it.
5. Store the log information in a database - Give your Atlas connection string with the three shards
6. Display operations analytics and full logs on a web-based dashboard - Provide a screenshot.

7.
```
    //Name: Verissa Owusu ID:vowusu
//Last modified: 9th April 2023
//This is a web servlet that handles requests through the/search and
/dashboard endpoints. The search searches 2
//3rd party APIs and returns data to the client and then makes inserts into a
database with records about the request
//Aletheia API get name,day high, day low, previousclose and price for graph
//MarketAux API get industry, exchange,  the latest news article which is at
index 0 of data and get the title and the url
//The /dashboard endpoint queries the database and presents some analytics to
the user
package com.example.webservice;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.*;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.annotation.WebServlet;
import com.mongodb.client.*;
import com.mongodb.client.result.InsertOneResult;
import org.bson.Document;
import org.bson.types.ObjectId;

@WebServlet(value = {"/search", "/dashboard"})
public class HelloServlet extends HttpServlet {
    private static final String ALETHEIA_API_URL =
"https://api.aletheiaapi.com/StockData?key=861059982CAA432CA72C26FFB3534D77&sy
mbol=";
    private static final String MARKETAUX_API_URL =
"https://api.marketaux.com/v1/news/all?api_token=8X0YkfefOOEQ9n5sA2ySwXcblvWU8
JECO2O61aXN&filter_entities=true&language=en&symbols=";
    private static final String MONGODB_URI =
"mongodb://Verrisa:eqR6KNP2LjNqtchV@ac-pxgawxe-shard-00-
00.vjicotq.mongodb.net:27017,ac-pxgawxe-shard-00-
01.vjicotq.mongodb.net:27017,ac-pxgawxe-shard-00-
02.vjicotq.mongodb.net:27017/myFirstDatabase?w=majority&retryWrites=true&tls=t
rue&authMechanism=SCRAM-SHA-1";
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        String path = request.getServletPath();
        // Retrieve the search term from the HTTP request
        if(path.contains("/search")) {
            String searchTerm = request.getParameter("searchTerm");
            response.setContentType("application/json");
            String userAgent = request.getHeader("User-Agent").split("\\(",
2)[0];
            //System.out.println("Device" + userAgent);
            Date currentDate = new Date();

            // Format the date and time as a string
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd
HH:mm:ss");
            String currentDateTime = dateFormat.format(currentDate);
            //System.out.println(currentDateTime);
            // Make an HTTP request to the Aletheia API with the search term
as a query parameter
            URL aletheiaApiUrl = new URL(ALETHEIA_API_URL + searchTerm);
            HttpURLConnection aletheiaConnection = (HttpURLConnection)
aletheiaApiUrl.openConnection();
            aletheiaConnection.setRequestProperty("Accept-Version", "2");
            aletheiaConnection.setRequestMethod("GET");

            // Getting response code
            int aletheiaResponseCode = aletheiaConnection.getResponseCode();

            if (aletheiaResponseCode == HttpURLConnection.HTTP_OK) {
                BufferedReader aletheiaIn = new BufferedReader(new
InputStreamReader(aletheiaConnection.getInputStream()));
                StringBuffer aletheiaJsonResponseData = new StringBuffer();
                String aletheiaReadLine;

                while ((aletheiaReadLine = aletheiaIn.readLine()) != null) {
                    aletheiaJsonResponseData.append(aletheiaReadLine);
                }

                aletheiaIn.close();

                // Create JSON object from the Aletheia API response data
                JsonObject aletheiaJsonObject = new
Gson().fromJson(aletheiaJsonResponseData.toString(), JsonObject.class);

                // Make an HTTP request to the MarketAux API with the search
term as a query parameter
                URL marketauxApiUrl = new URL(MARKETAUX_API_URL + searchTerm);
                HttpURLConnection marketauxConnection = (HttpURLConnection)
marketauxApiUrl.openConnection();
                marketauxConnection.setRequestMethod("GET");

                // Getting response code
                int marketauxResponseCode =
marketauxConnection.getResponseCode();

                if (marketauxResponseCode == HttpURLConnection.HTTP_OK) {
                    BufferedReader marketauxIn = new BufferedReader(new
InputStreamReader(marketauxConnection.getInputStream()));
                    StringBuffer marketauxJsonResponseData = new
StringBuffer();
                    String marketauxReadLine;
                    while ((marketauxReadLine = marketauxIn.readLine()) !=
null) {
                        marketauxJsonResponseData.append(marketauxReadLine);
                    }

                    marketauxIn.close();

                    // Create JSON object from the MarketAux API response data
                    JsonObject marketauxJsonObject = new
Gson().fromJson(marketauxJsonResponseData.toString(), JsonObject.class);

                    // Extract data from the Aletheia API JSON response
                    String shortName =
aletheiaJsonObject.get("ShortName").getAsString();
                    float dayHigh =
aletheiaJsonObject.get("DayHigh").getAsFloat();
                    float dayLow =
aletheiaJsonObject.get("DayLow").getAsFloat();
                    JsonElement f = aletheiaJsonObject.get("PreviousClose");
                    float previousClose = 0;
                    if(f != null){ previousClose =
aletheiaJsonObject.get("PreviousClose").getAsFloat();}
                    if(f == null){ previousClose =
aletheiaJsonObject.get("Open").getAsFloat();}
                    float price =
aletheiaJsonObject.get("Price").getAsFloat();


                    String title = null,url = null,exchange = null, industry =
null;

                    // Extract data from the Marketaux API JSON response
                    JsonArray data = (JsonArray)
marketauxJsonObject.get("data");
                    if(data.size()>0) {
                        JsonObject firstItem = data.get(0).getAsJsonObject();
                         title = firstItem.get("title").getAsString();
                         url = firstItem.get("url").getAsString();
                        JsonArray data2 =
firstItem.get("entities").getAsJsonArray();
                        JsonObject data21 = data2.get(0).getAsJsonObject();
                         exchange = data21.get("exchange").getAsString();
                         industry = data21.get("industry").getAsString();
                    }
                    else if (data.size() ==0) {
                        //JsonObject firstItem =
data.get(0).getAsJsonObject();
                         title = null;
                         url = null;
                        //JsonArray data2 =
firstItem.get("entities").getAsJsonArray();
                        //JsonObject data21 = data2.get(0).getAsJsonObject();
                        JsonElement e = aletheiaJsonObject.get("Exchange");
                         exchange = null;
                        if(e != null){ exchange =
aletheiaJsonObject.get("Exchange").getAsString();}
```

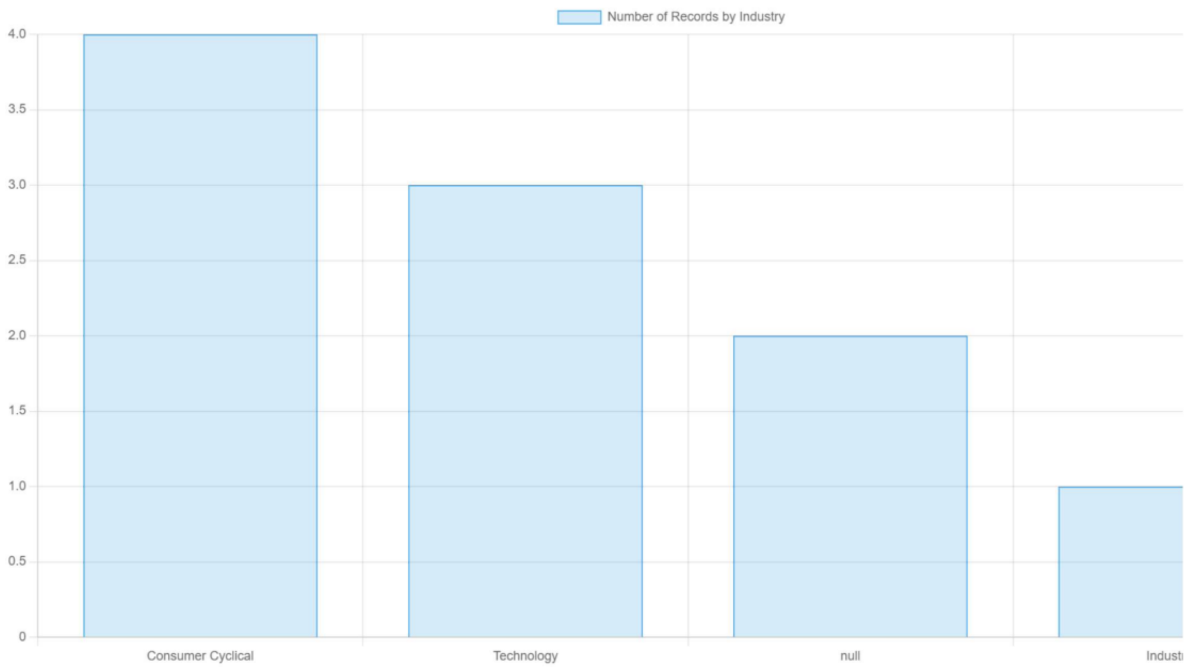
```
                        if(e == null){ exchange = null;}
                         industry = null;

                    }

                    // Create a new JsonObject to store all the extracted data
                    JsonObject jsonResponse = new JsonObject();

                    // Add all the extracted data as properties to the
JsonObject
                    jsonResponse.addProperty("name", shortName);
                    jsonResponse.addProperty("dayHigh", dayHigh);
                    jsonResponse.addProperty("dayLow", dayLow);
                    jsonResponse.addProperty("previousClose", previousClose);
                    jsonResponse.addProperty("price", price);
                    jsonResponse.addProperty("title", title);
                    jsonResponse.addProperty("url", url);
                    jsonResponse.addProperty("exchange", exchange);
                    jsonResponse.addProperty("industry", industry);

                    // Use PrintWriter object to write the JsonObject as a
response to the client
                    PrintWriter out = response.getWriter();
                    out.print(jsonResponse.toString());
                    out.flush();

                    try (MongoClient mongoClient =
MongoClients.create(MONGODB_URI)) {
                        MongoDatabase database =
mongoClient.getDatabase("Aletheia");
                        MongoCollection<Document> collection =
database.getCollection("StockData");
                        MongoCollection<Document> collection2 =
database.getCollection("Requests");
                        if(jsonResponse == null) {
                            InsertOneResult result1 =
collection2.insertOne(new Document()
                                    .append("_id", new ObjectId())
                                    .append("device",
userAgent).append("date", currentDateTime).append("symbolTicker", searchTerm)
                                    .append("status",
"failed").append("companyInfo",shortName + " " +
industry).append("httpResponseCode", "503"));
                            System.out.println("error");
                        }
                        else if(jsonResponse != null) {
                            InsertOneResult result1 =
collection2.insertOne(new Document()
                                    .append("_id", new ObjectId())
                                    .append("device",
userAgent).append("date", currentDateTime).append("symbolTicker", searchTerm)
                                    .append("status",
"success").append("companyInfo",shortName + " " +
industry).append("httpResponseCode", "200"));
                        }
                        InsertOneResult result = collection.insertOne(new
Document()
                                .append("_id", new ObjectId())
                                .append("name", shortName).append("industry",
industry).append("exchange", exchange)
                                .append("price", price).append("dayHigh",
dayHigh).append("dayLow", dayLow)
                                .append("previousClose",
previousClose).append("title", title).append("url", url));


                    }
                } else if (marketauxResponseCode != HttpURLConnection.HTTP_OK)
{
                    try (MongoClient mongoClient =
MongoClients.create(MONGODB_URI)) {
                        MongoDatabase database =
mongoClient.getDatabase("Aletheia");
                        MongoCollection<Document> collection =
database.getCollection("StockData");
                        MongoCollection<Document> collection2 =
database.getCollection("Requests");
                            InsertOneResult result1 =
collection2.insertOne(new Document()
                                    .append("_id", new ObjectId())
                                    .append("device",
userAgent).append("date", currentDateTime).append("symbolTicker", searchTerm)
                                    .append("status",
"failed").append("companyInfo"," " ).append("httpResponseCode", "500"));
                        }
                    PrintWriter out = response.getWriter();
                    out.print("MarketAux API is down");
                    out.flush();
                }
            } else if (aletheiaResponseCode != HttpURLConnection.HTTP_OK) {
                try (MongoClient mongoClient =
MongoClients.create(MONGODB_URI)) {
                    MongoDatabase database =
mongoClient.getDatabase("Aletheia");
                    MongoCollection<Document> collection =
database.getCollection("StockData");
                    MongoCollection<Document> collection2 =
database.getCollection("Requests");
                    InsertOneResult result1 = collection2.insertOne(new
Document()
                            .append("_id", new ObjectId())
                            .append("device", userAgent).append("date",
currentDateTime).append("symbolTicker", searchTerm)
                            .append("status", "failed").append("companyInfo","
" ).append("httpResponseCode", "500"));
                }
                PrintWriter out = response.getWriter();
                out.print("Aletheia API is down");

                out.flush();
            }
        } else if (path.contains("/dashboard")) {
            try (MongoClient mongoClient = MongoClients.create(MONGODB_URI)) {
                MongoDatabase database = mongoClient.getDatabase("Aletheia");
                MongoCollection<Document> collection =
database.getCollection("StockData");
                MongoCollection<Document> collection2 =
database.getCollection("Requests");
                FindIterable<Document> iterDoc = collection2.find();
                Iterator it = iterDoc.iterator();

                // Generate an HTML table to display the documents
                StringBuilder sb = new StringBuilder();
                sb.append("<html><head><title>Stock Data App
Dashboard</title></head><body><table>");
                sb.append("<table style=\"border-collapse: collapse;\">");
                sb.append("<thead style=\"background-color: #D1F0FA;\">");
                sb.append("<h2>Web Service Request Logs");
                sb.append("</h2>");
                sb.append("<tr><th style=\"border: 1px solid
black;\">Device</th>" +
                        "<th style=\"border: 1px solid black;\">Date</th>" +
                        "<th style=\"border: 1px solid black;\">Symbol
Ticker</th>"+
                        "<th style=\"border: 1px solid black;\">Status</th>" +
                        "<th style=\"border: 1px solid black;\">Response
Code</th>"+
                        "<th style=\"border: 1px solid black;\">Company
Info</th></tr>");
                sb.append("</thead>");
                sb.append("<tbody>");

                while (it.hasNext()) {
                    Document doc = (Document) it.next();
                    String device = doc.getString("device");
                    String date = doc.getString("date");
                    String symbolTicker = doc.getString("symbolTicker");
                    String status = doc.getString("status");
                    String responseCode = doc.getString("httpResponseCode");
                    String companyInfo = doc.getString("symbolTicker");
                    sb.append("<tr><td style=\"border: 1px solid
black;\">").append(device)
                            .append("</td><td style=\"border: 1px solid
black;\">").append(date)
                            .append("</td><td style=\"border: 1px solid
black;\">").append(symbolTicker)
                            .append("</td><td style=\"border: 1px solid
black;\">").append(status)
                            .append("</td><td style=\"border: 1px solid
black;\">").append(responseCode)
                            .append("</td><td style=\"border: 1px solid
black;\">").append(companyInfo)
                            .append("</td></tr>");
                }

                sb.append("</tbody>");
                sb.append("</table></body></html>");
                // Write the HTML table to the response output stream
                //response.setContentType("text/html");
                PrintWriter out = response.getWriter();
                out.println(sb.toString());

                // MongoDB Aggregation pipeline to group by industry and count
the number of documents in each group
                //https://studio3t.com/knowledge-base/articles/mongodb-
aggregation-framework/
                List<Document> pipeline = Arrays.asList(
                        new Document("$group", new Document("_id",
"$industry").append("count", new Document("$sum", 1))),
                        new Document("$sort", new Document("count", -1))
                );

                List<Document> results =
collection.aggregate(pipeline).into(new ArrayList<>());

                // Prepare the data for the bar chart
                List<String> labels = new ArrayList<>();
                List<Integer> counts = new ArrayList<>();
                for (Document doc : results) {
                    labels.add("'"+ doc.getString("_id")+"'");
                    counts.add(doc.getInteger("count"));
                }

                // Generate the HTML and JavaScript for the bar chart using
Chart.js library
                StringBuilder sb2 = new StringBuilder();
                sb2.append("<html><head><title>Stock Data App
Dashboard</title>");
                sb2.append("<h2>Stock Data Requests per Industry");
                sb2.append("</h2>");
                sb2.append("<script
src=\"https://cdn.jsdelivr.net/npm/chart.js\"></script>");
                sb2.append("</head><body>");
                sb2.append("<canvas id=\"myChart\"></canvas>");
                sb2.append("<script>");
                sb2.append("var ctx =
document.getElementById('myChart').getContext('2d');");
                sb2.append("var myChart = new Chart(ctx, {");
                sb2.append("type: 'bar',");
                sb2.append("data: {");
                sb2.append("labels: ").append(labels).append(",");
                sb2.append("datasets: [{");
                sb2.append("label: 'Number of Records by Industry',");
                sb2.append("data: ").append(counts).append(",");
                sb2.append("backgroundColor: 'rgba(54, 162, 235, 0.2)',");
                sb2.append("borderColor: 'rgba(54, 162, 235, 1)',");
                sb2.append("borderWidth: 1");
                sb2.append("}]");
                sb2.append("},");
                sb2.append("options: {");
                sb2.append("scales: {");
                sb2.append("y: {");
                sb2.append("beginAtZero: true");
                sb2.append("}");
                sb2.append("}");
                sb2.append("}");
                sb2.append("});");
                sb2.append("</script>");
                sb2.append("</body></html>");

                // Write the HTML and JavaScript to the response output stream
                response.setContentType("text/html");
                //PrintWriter out = response.getWriter();
                out.println(sb2.toString());

                List<Document> pipeline2 = new ArrayList<>();
                pipeline2.add(new Document("$group", new Document("_id",
"$name")
                        .append("name", new Document("$first", "$name"))
                        .append("industry", new Document("$first",
"$industry"))
                        .append("exchange", new Document("$first",
"$exchange"))
                        .append("price", new Document("$max", "$price"))));
                pipeline2.add(new Document("$sort", new Document("price", -
1)));
                pipeline2.add(new Document("$limit", 3));

                AggregateIterable<Document> iterDoc2 =
collection.aggregate(pipeline2);

                Iterator it2 = iterDoc2.iterator();

                StringBuilder sb3 = new StringBuilder();
                sb3.append("<html><head><title>Stock Data App
Dashboard</title></head><body><table>");
                sb3.append("<table style=\"border-collapse: collapse;\">");
                sb3.append("<thead style=\"background-color: #D1F0FA;\">");
                sb3.append("<h2>Top 3 Highest Stocks");
                sb3.append("</h2>");
                sb3.append("<tr>" +
                        "<th style=\"border: 1px solid black;\">Name</th>" +
                        "<th style=\"border: 1px solid black;\">Industry</th>"
+
                        "<th style=\"border: 1px solid black;\">Exchange</th>"
+
                        "<th style=\"border: 1px solid
black;\">Price</th></tr>");
                sb3.append("</thead>");
                sb3.append("<tbody>");

                while (it2.hasNext()) {
                    Document doc = (Document) it2.next();
                    String name = doc.getString("name");
                    String industry = doc.getString("industry");
                    String exchange = doc.getString("exchange");
                    Double price = doc.getDouble("price");
                    sb3.append("<tr><td style=\"border: 1px solid
black;\">").append(name).
```

```
                             append("</td><td style=\"border: 1px solid
black;\">").
                             append(industry).append("</td><td style=\"border:
1px solid black;\">")
                             .append(exchange).append("</td><td style=\"border:
1px solid black;\">")
                             .append(price).append("</td></tr>");
                }

                sb3.append("</tbody>");
                sb3.append("</table></body></html>");

                // Write the HTML table to the response output stream
                response.setContentType("text/html");
                out.println(sb3.toString());

                List<Document> pipeline3 = new ArrayList<>();
                pipeline3.add(new Document("$group", new Document("_id",
"$name")
                        .append("name", new Document("$first", "$name"))
                        .append("industry", new Document("$first",
"$industry"))
                        .append("exchange", new Document("$first",
"$exchange"))
                        .append("price", new Document("$min", "$price"))));
                pipeline3.add(new Document("$sort", new Document("price",
1)));
                pipeline3.add(new Document("$limit", 3));

                AggregateIterable<Document> iterDoc3 =
collection.aggregate(pipeline3);

                //FindIterable<Document> iterDoc2 = collection.find().sort(new
BasicDBObject("price", -1)).limit(3);
                Iterator it3 = iterDoc3.iterator();

                StringBuilder sb4 = new StringBuilder();
                sb4.append("<html><head><title>Stock Data App
Dashboard</title></head><body><table>");
                sb4.append("<table style=\"border-collapse: collapse;\">");
                sb4.append("<thead style=\"background-color: #D1F0FA;\">");
                sb4.append("<h2>Top 3 Lowest Priced Stocks");
                sb4.append("</h2>");
                sb4.append("<tr>" +
                        "<th style=\"border: 1px solid black;\">Name</th>" +
                        "<th style=\"border: 1px solid black;\">Industry</th>"
+
                        "<th style=\"border: 1px solid black;\">Exchange</th>"
+
                        "<th style=\"border: 1px solid
black;\">Price</th></tr>");
                sb4.append("</thead>");
                sb4.append("<tbody>");

                while (it3.hasNext()) {
                    Document doc = (Document) it3.next();
                    String name = doc.getString("name");
                    String industry = doc.getString("industry");
                    String exchange = doc.getString("exchange");
                    Double price = doc.getDouble("price");
                    sb4.append("<tr><td style=\"border: 1px solid
black;\">").append(name).
                            append("</td><td style=\"border: 1px solid
black;\">").
                            append(industry).append("</td><td style=\"border:
1px solid black;\">")
                            .append(exchange).append("</td><td style=\"border:
1px solid black;\">")
                            .append(price).append("</td></tr>");
                }

                sb4.append("</tbody>");
                sb4.append("</table></body></html>");

                // Write the HTML table to the response output stream
                response.setContentType("text/html");
                out.println(sb4.toString());
                out.flush();
            }

        }
    }


    public void destroy() {
    }
}
```

Web Service Request Logs

Device	Date	Symbol Ticker	Status	Response Code	Company Info
Dalvik/2.1.0	2023/04/10 03:44:49	aapl	success	200	aapl
Dalvik/2.1.0	2023/04/10 03:45:11	f	success	200	f
Dalvik/2.1.0	2023/04/10 03:45:44	tsla	success	200	tsla
Dalvik/2.1.0	2023/04/10 03:45:59	amzn	success	200	amzn
Dalvik/2.1.0	2023/04/10 03:46:22	amzn	success	200	amzn
Dalvik/2.1.0	2023/04/10 03:46:34	msft	success	200	msft
Dalvik/2.1.0	2023/04/10 03:47:01	adbe	success	200	adbe
Dalvik/2.1.0	2023/04/10 03:48:12	alk	success	200	alk
Mozilla/5.0	2023/04/10 00:34:26	ax1	success	200	ax1
Mozilla/5.0	2023/04/10 00:36:45	ab1	success	200	ab1
Mozilla/5.0	2023/04/10 00:39:29	ac1	failed	503	ac1
Dalvik/2.1.0	2023/04/10 04:52:37	az1	failed	500	az1
Dalvik/2.1.0	2023/04/10 04:55:01	az1	failed	500	az1
Dalvik/2.1.0	2023/04/10 04:55:46	az1	failed	500	az1
Dalvik/2.1.0	2023/04/10 04:57:51	az1	failed	500	az1

Stock Data Requests per Industry



Top 3 Highest Stocks

Name	Industry	Exchange	Price
Adobe Inc.	Technology	NASDAQ	380.6000061035156
Microsoft Corporation	Technology	NASDAQ	291.6000061035156
Tesla, Inc.	Consumer Cyclical	NASDAQ	185.05999755859375

Top 3 Lowest Priced Stocks

Name	Industry	Exchange	Price
Air Berlin PLC	null	Germany: Xetra	0.009999999776482582
Accent Group Ltd.	null	Australia: Sydney	2.430000066757202
Ford Motor Company	Consumer Cyclical	NYSE	12.329999923706055