Vardhan Khara (vkhara)

95702 – Distributed Systems for ISM

Project 4 Task 2

1. **a.Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View). In order to figure out if something is a View, find its API. If it extends android.view.View then it is a View.**

   Name of my application is FootballData App.

   The different types of views are:
   Text view
   EditText
   Button
   Scroll View

b. Requires input from the user

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

```
no usages
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String teamId = req.getParameter( s: "teamId");
    logger.info("Fetching matches for team ID: {}", teamId);

    String apiUrl = "https://api.football-data.org/v2/competitions/PL/matches";
    Request request = new Request.Builder()
            .url(apiUrl)
            .addHeader( name: "X-Auth-Token",  value: "9d838ba2b7674b738cb599c8acb13987")
            .build();

    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            logger.error("Failed to fetch data from the Football-Data API, response code: {}", response.code());
            resp.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,  s: "Internal server error occurred while fetchi
            return;
        }

        String responseBody = response.body().string();
        JsonElement jsonElement = gson.fromJson(responseBody, JsonElement.class);
        logToDatabase(responseBody, teamId); // Include this method call

        resp.setContentType("application/json");
        resp.setCharacterEncoding("UTF-8");
        resp.getWriter().print(gson.toJson(jsonElement));
    } catch (IOException e) {
        logger.error("Error occurred while making HTTP request to Football-Data API", e);
        resp.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,  s: "Internal server error occurred while making HTT
    }
}
```

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

```
   private void fetchData(String apiUrl, HttpServletResponse resp, String
action) throws IOException {
       long startTime = System.currentTimeMillis();
       String userAgent = resp.getHeader("User-Agent");// Start time of the
request

       Request request = new Request.Builder()
               .url(apiUrl)
               .addHeader("X-Auth-Token", "9d838ba2b7674b738cb599c8acb13987")
// Use your actual API token here
               .build();

       try (Response response = httpClient.newCall(request).execute()) {

           long endTime = System.currentTimeMillis(); // End time of the
request
           long requestPayloadSize = (request.body() != null) ?
request.body().contentLength() : 0;
```

```
            long responsePayloadSize = (response.body() != null) ?
response.body().contentLength() : 0;
            boolean success = response.isSuccessful();// Determine if the
request was successful

            if (!success) {
                // Log the unsuccessful request and respond with an error
                logToMongoDB(action, apiUrl, success, startTime, endTime,
requestPayloadSize, responsePayloadSize, userAgent);

                // Handle exception
                logger.error("Failed to fetch data from the Football-Data API,
response code: {}", response.code());
                resp.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
"Error fetching data");
                return;
            }

            // If the request was successful, process the response
            String responseBody = response.body().string();
            JsonArray structuredResponse =
convertJsonToStructuredText(responseBody, action);

            // Log the successful request

            // Send back the structured response
            resp.setContentType("text/plain");
            resp.setCharacterEncoding("UTF-8");
            resp.getWriter().print(structuredResponse);
        } catch (IOException e) {
            long endTime = System.currentTimeMillis(); // Capture end time even
in case of an exception
            logToMongoDB(action, apiUrl, false, startTime, endTime, 0, 0,
userAgent);

            logger.error("Error occurred while making HTTP request", e);
            resp.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR, "HTTP
request failed");
        }
    }



    private JsonArray convertJsonToStructuredText(String json, String action) {
        StringBuilder structuredText = new StringBuilder();
        JsonObject jsonObj = gson.fromJson(json, JsonObject.class);
        JsonArray array = new JsonArray();

        switch (action) {
```

```
            case "teams":
                JsonArray teams = jsonObj.getAsJsonArray("teams");
                for (JsonElement teamElement : teams) {
                    JsonObject team = teamElement.getAsJsonObject();
                    JsonObject appendingObject = new JsonObject();

appendingObject.addProperty("Team_name",team.get("name").getAsString());
                    structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");
                    array.add(appendingObject);

                }
                break;
//              case "matches":
//                  JsonArray matches = jsonObj.getAsJsonArray("matches");
//                  for (JsonElement matchElement : matches) {
//                      JsonObject match = matchElement.getAsJsonObject();
//                      structuredText.append("Match Date:
").append(match.get("utcDate").getAsString())
//                              .append(", Home Team:
").append(match.getAsJsonObject("homeTeam").get("name").getAsString())
//                              .append(", Away Team:
").append(match.getAsJsonObject("awayTeam").get("name").getAsString()).append("
\n");
//                  }
//                  break;

            case "scorers":
                JsonElement scorersElement = jsonObj.get("scorers");
                if (scorersElement != null && scorersElement.isJsonArray()) {
                    JsonArray scorers = scorersElement.getAsJsonArray();
                    logger.info("Scorers array found, size: {}",
scorers.size());

                    for (JsonElement scorerElement : scorers) {
                        JsonObject scorer = scorerElement.getAsJsonObject();
                        JsonObject player = scorer.getAsJsonObject("player");
                        String playerName = player != null ?
player.get("name").getAsString() : "Unknown Player";



//                          structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");
//                          array.add(appendingObject);
                        String goals = scorer.has("numberOfGoals") ?
scorer.get("numberOfGoals").getAsString() : "0";
                        JsonObject appendingObject = new JsonObject();
                        appendingObject.addProperty("Player",playerName);
```

```
                    appendingObject.addProperty("Goals",goals);
                    structuredText.append("Player: ").append(playerName)
                            .append(", Goals: ").append(goals).append("\n");
                    array.add(appendingObject);
                }
            } else {
                logger.error("No 'scorers' array found or not a JSON
array.");
                structuredText.append("No top scorers information
available.\n");
            }
            break;
```

e. Display new information to the user

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

2. Implement a web service

a. Implement a simple (can be a single path) API.

https://urban-space-robot-65wwgpwg5w42xxq6-8080.app.github.dev/dashboard.jsp

b. Receives an HTTP request from the native Android application

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {

   String pathInfo = req.getPathInfo();

   if (pathInfo == null || pathInfo.equals("/")) {

       resp.sendError(HttpServletResponse.SC_BAD_REQUEST, "Missing action");

       return;

   }

   String[] splits = pathInfo.split("/");

   if (splits.length != 2) {

       resp.sendError(HttpServletResponse.SC_BAD_REQUEST, "Invalid action");

       return;

   }

   String action = splits[1];

   String apiUrl =
String.format("https://api.football-data.org/v2/competitions/PL/%s?season=2021"
, action);



   fetchData(apiUrl, resp, action);

}
```

c. Executes business logic appropriate to your application. This includes fetching XML or JSON information from some 3rd party API and processing the response.

```
   private JsonArray convertJsonToStructuredText(String json, String action) {

       StringBuilder structuredText = new StringBuilder();

       JsonObject jsonObj = gson.fromJson(json, JsonObject.class);
```

```
       JsonArray array = new JsonArray();



       switch (action) {

           case "teams":

               JsonArray teams = jsonObj.getAsJsonArray("teams");

               for (JsonElement teamElement : teams) {

                   JsonObject team = teamElement.getAsJsonObject();

                   JsonObject appendingObject = new JsonObject();


appendingObject.addProperty("Team_name",team.get("name").getAsString());

                   structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");

                   array.add(appendingObject);



               }

               break;

//             case "matches":

//                 JsonArray matches = jsonObj.getAsJsonArray("matches");

//                 for (JsonElement matchElement : matches) {

//                     JsonObject match = matchElement.getAsJsonObject();

//                     structuredText.append("Match Date:
").append(match.get("utcDate").getAsString())

//                             .append(", Home Team:
").append(match.getAsJsonObject("homeTeam").get("name").getAsString())

//                             .append(", Away Team:
").append(match.getAsJsonObject("awayTeam").get("name").getAsString()).append("
\n");

//                 }

//                 break;
```

```
            case "scorers":

                JsonElement scorersElement = jsonObj.get("scorers");

                if (scorersElement != null && scorersElement.isJsonArray()) {

                    JsonArray scorers = scorersElement.getAsJsonArray();

                    logger.info("Scorers array found, size: {}",
scorers.size());



                    for (JsonElement scorerElement : scorers) {

                        JsonObject scorer = scorerElement.getAsJsonObject();

                        JsonObject player = scorer.getAsJsonObject("player");

                        String playerName = player != null ?
player.get("name").getAsString() : "Unknown Player";







//                          structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");

//                          array.add(appendingObject);

                        String goals = scorer.has("numberOfGoals") ?
scorer.get("numberOfGoals").getAsString() : "0";

                        JsonObject appendingObject = new JsonObject();

                        appendingObject.addProperty("Player",playerName);

                        appendingObject.addProperty("Goals",goals);

                        structuredText.append("Player: ").append(playerName)

                                .append(", Goals: ").append(goals).append("\n");

                        array.add(appendingObject);

                    }
```

```
                } else {
                    logger.error("No 'scorers' array found or not a JSON
array.");
                    structuredText.append("No top scorers information
available.\n");
                }
                break;

            default:
                structuredText.append("Action not recognized.");
                break;
        }

        return array;
    }
```

d. Replies to the Android application with an XML or JSON formatted response. The schema of the response can be of your own design.

```
private JsonArray convertJsonToStructuredText(String json, String action) {
        StringBuilder structuredText = new StringBuilder();
        JsonObject jsonObj = gson.fromJson(json, JsonObject.class);
        JsonArray array = new JsonArray();

        switch (action) {
            case "teams":
                JsonArray teams = jsonObj.getAsJsonArray("teams");
```

```
                for (JsonElement teamElement : teams) {

                    JsonObject team = teamElement.getAsJsonObject();

                    JsonObject appendingObject = new JsonObject();


appendingObject.addProperty("Team_name",team.get("name").getAsString());

                    structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");

                    array.add(appendingObject);



                }

                break;

//            case "matches":

//                JsonArray matches = jsonObj.getAsJsonArray("matches");

//                for (JsonElement matchElement : matches) {

//                    JsonObject match = matchElement.getAsJsonObject();

//                    structuredText.append("Match Date:
").append(match.get("utcDate").getAsString())

//                            .append(", Home Team:
").append(match.getAsJsonObject("homeTeam").get("name").getAsString())

//                            .append(", Away Team:
").append(match.getAsJsonObject("awayTeam").get("name").getAsString()).append("
\n");

//                }

//                break;



            case "scorers":

                JsonElement scorersElement = jsonObj.get("scorers");

                if (scorersElement != null && scorersElement.isJsonArray()) {

                    JsonArray scorers = scorersElement.getAsJsonArray();
```

```
                    logger.info("Scorers array found, size: {}",
scorers.size());



                    for (JsonElement scorerElement : scorers) {

                        JsonObject scorer = scorerElement.getAsJsonObject();

                        JsonObject player = scorer.getAsJsonObject("player");

                        String playerName = player != null ?
player.get("name").getAsString() : "Unknown Player";







//                          structuredText.append("Team Name:
").append(team.get("name").getAsString()).append("\n");

//                          array.add(appendingObject);

                        String goals = scorer.has("numberOfGoals") ?
scorer.get("numberOfGoals").getAsString() : "0";

                        JsonObject appendingObject = new JsonObject();

                        appendingObject.addProperty("Player",playerName);

                        appendingObject.addProperty("Goals",goals);

                        structuredText.append("Player: ").append(playerName)

                                .append(", Goals: ").append(goals).append("\n");

                        array.add(appendingObject);

                    }

                } else {

                    logger.error("No 'scorers' array found or not a JSON
array.");

                    structuredText.append("No top scorers information
available.\n");

                }
```

```
                break;


            default:
                structuredText.append("Action not recognized.");
                break;
        }


        return array;
    }
```

4. Log useful information

1. Total Requests: This metric shows the total number of API requests that have been made. It's useful for understanding the load on your system and can help in capacity planning and scaling.

2. Average Response Time: This represents the average time taken for the server to process and respond to a request. Monitoring this helps in ensuring that your service is performing efficiently and is critical for maintaining a good user experience.

3. Most Recent Endpoint Accessed: It displays the last API endpoint that was called. This can give insights into the current active areas of your application and can be useful for debugging or monitoring which features are being used.

4. Average Duration of Requests: Similar to the average response time, this metric specifically measures the duration from when a request starts to when it ends, giving you an idea of the backend processing time.

5. Total Response Size: The average or total size of the response payloads sent back to the clients. This metric can help in identifying the endpoints that return large amounts of data, which might be optimized for performance and cost.

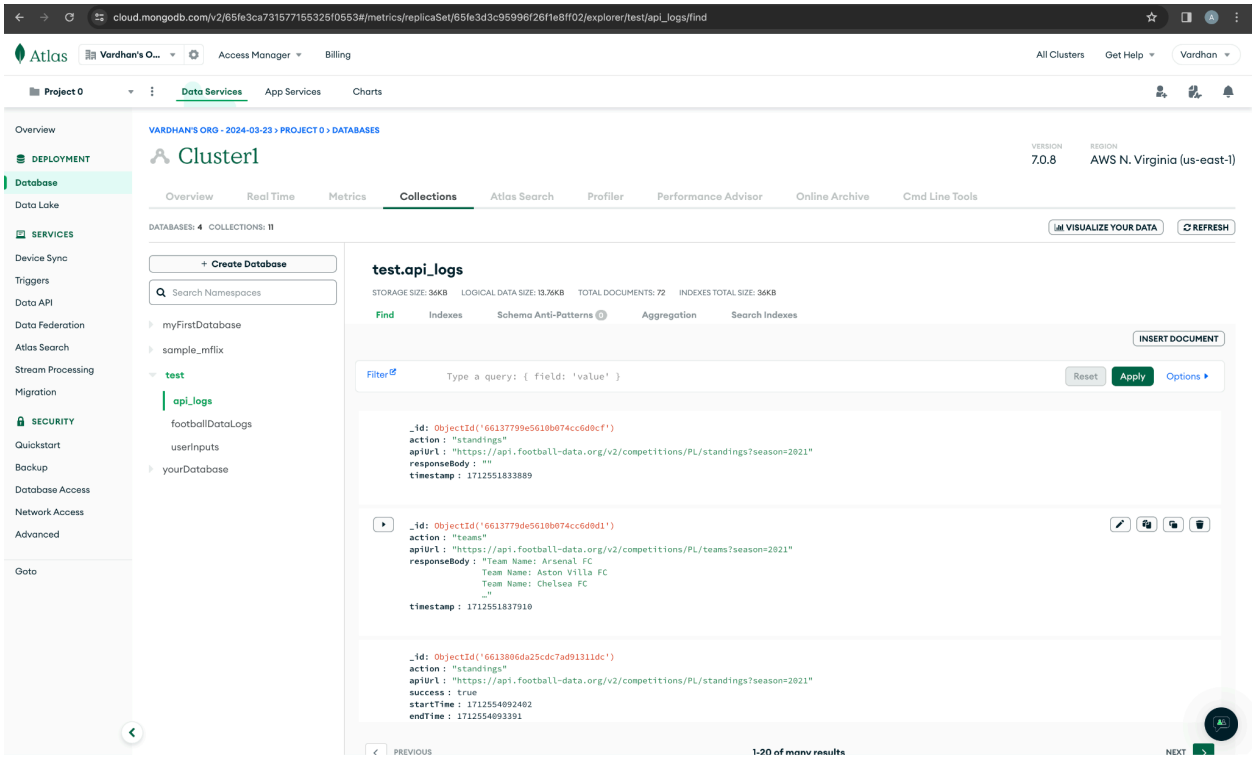
6. Error Rate: Although not explicitly stated in the provided document, if you're logging request success or failure, the error rate would be the percentage of requests that resulted in an error. This is crucial for maintaining the reliability of your service.

7. Timestamp: Recorded for each log entry, this indicates the exact time an event occurred, such as a request to your API. This can help you identify peak usage times or spot trends over time.

5. Store the log information in a database

Connection string:

```
mongodb+srv://vkhara:vardhan2001@cluster1.pcagjk1.mongodb.net/?retryWrites=tru
e&w=majority&appName=Cluster1
```



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

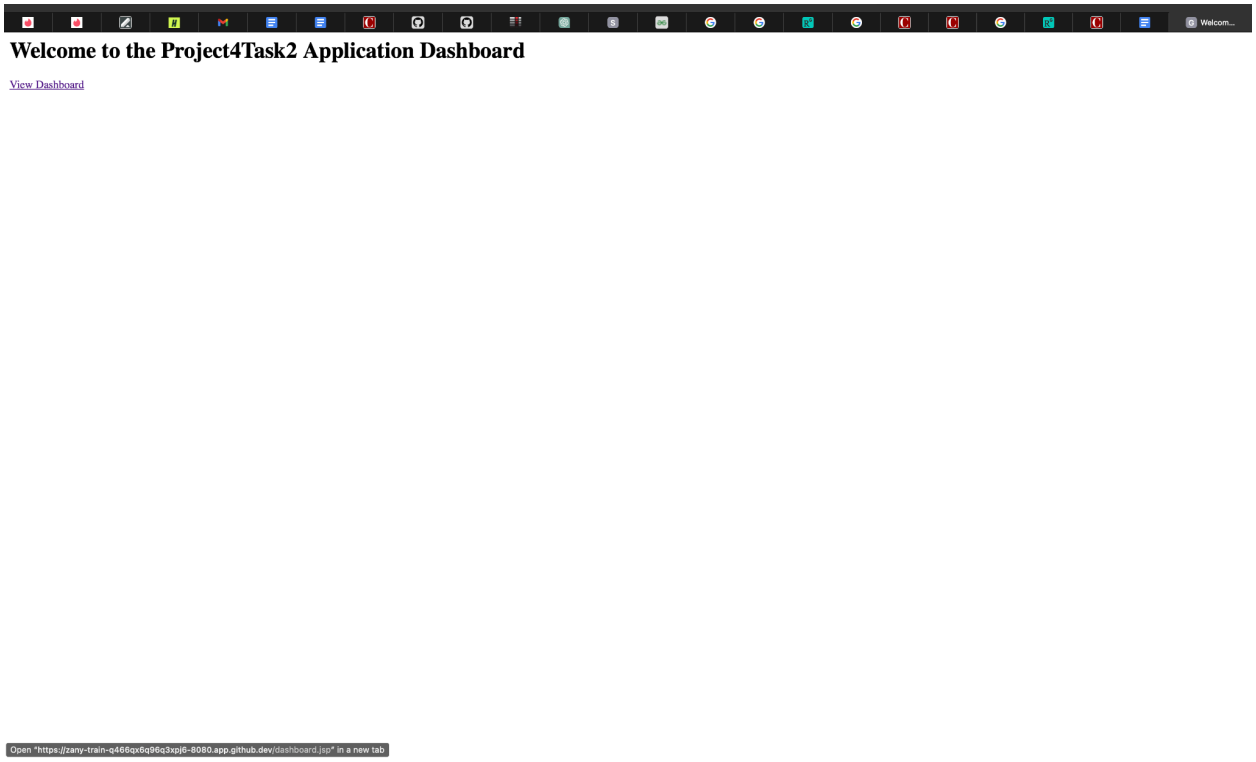
a. A unique URL addresses a web interface dashboard for the web service.

https://urban-space-robot-65wwgpwg5w42xxq6-8080.app.github.dev/dashboard.jsp

b. The dashboard displays at least 3 interesting operations analytics.

And

c. The dashboard displays formatted full logs



Vardhan Khara (vkhara)