Project4

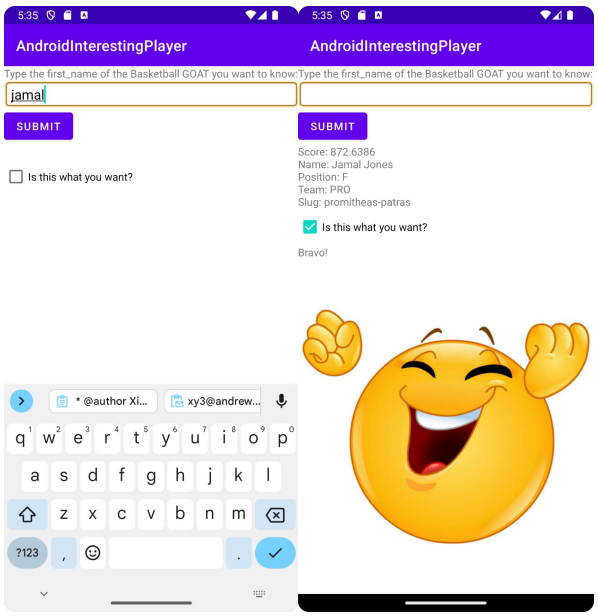
Xinyuan Yang(xy3)

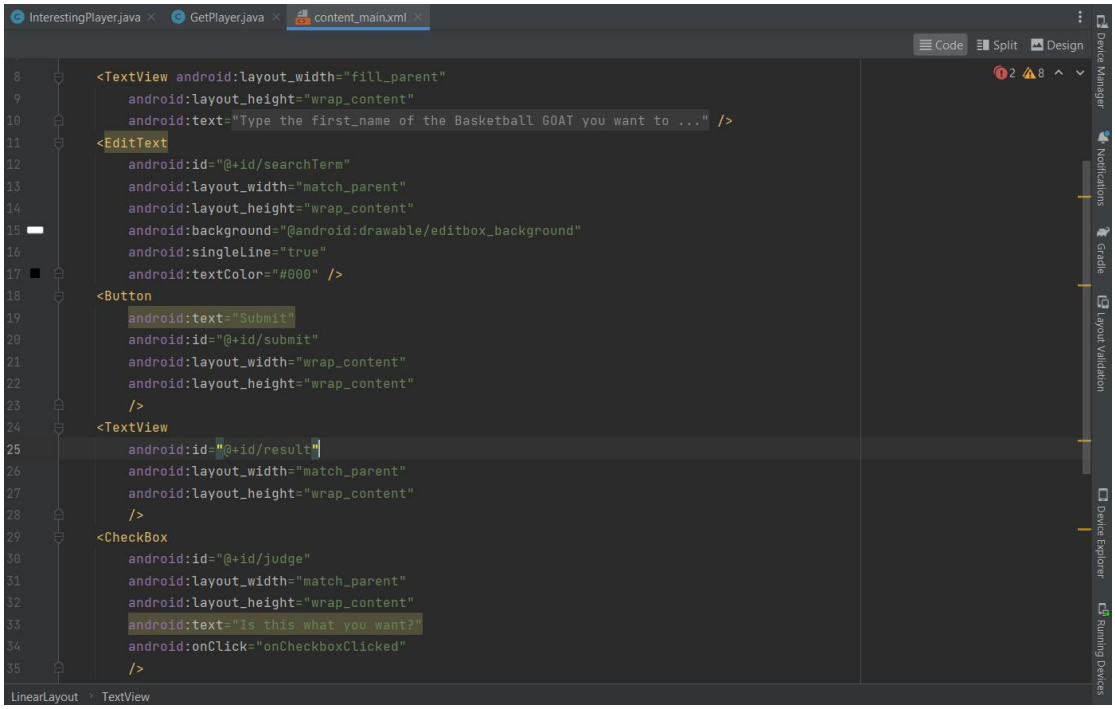
Distributed Application Requirements

1. Implement a native Android application

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.

To answer this question, I screenshot my app userface as well as the content_main.xml. We can see that there are EditText, Button, TextView, Checkbox and Image(which will show up after the checkbox is checked), satisfying the quantity requirements of different kinds of Views.

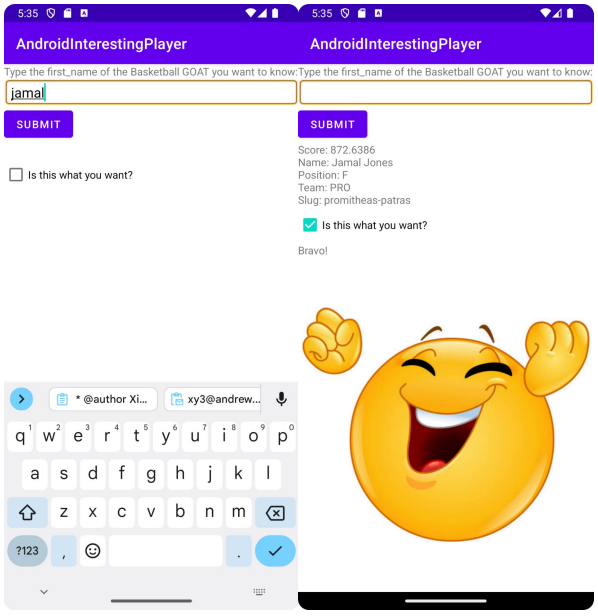




```
    <CheckBox
        android:id="@+id/judge"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Is this what you want?"
        android:onClick="onCheckboxClicked"
        />
    <TextView
        android:id="@+id/satisfy"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
        <ImageView
            android:id="@+id/interestingPicture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/smile"
            android:visibility="invisible"
            />
</LinearLayout>
```

b. Requires input from the user

See the attached screenshot, asking user to input the first_name of the basketball player they are interested in. Simiarly as the lab, EditText View.



```xml
<TextView android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Type the first_name of the Basketball GOAT you want to ..." />
<EditText
    android:id="@+id/searchTerm"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"
    android:singleLine="true"
    android:textColor="#000" />
<Button
    android:text="Submit"
    android:id="@+id/submit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
<TextView
    android:id="@+id/result"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    />
<CheckBox
    android:id="@+id/judge"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Is this what you want?"
    android:onClick="onCheckboxClicked"
    />
```

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

See attached screenshot on Android Side and Intellij Side. It is using a GET method.

My application does an HTTP GET request in GetPlayerModel.java:[See attached, the first_name is what user input]

```java
 *
 * @param first_name The first name of the basketball player.
 * @return The JSON object containing information about the selected player.
 * @throws ParseException If there's an error parsing the JSON response.
 */
1 usage
public static JSONObject generateInfo(String first_name) throws ParseException {
    String address = "https://basketapi1.p.rapidapi.com/api/basketball/search/" + first_name;
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MMM dd, yyyy");
    String output = api_fetch(address);
    JSONObject output_object = (JSONObject) JSONValue.parse(output);
    JSONArray results = (JSONArray) output_object.get("results");
    Random random = new Random();
    int randomNumber = random.nextInt( bound: (results.size() - 1 - 0) + 1) + 0;
    JSONObject currentPlayer = null;
    if (results != null) {
```

```java
static private String api_fetch(String urlString) {
    String response = "";
    try {
        URL url = new URL(urlString);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        String X_RapidAPI_Key = "efb11adcfemsh287c614b1afa931p1bb807jsndcdb97d50856";
        String X_RapidAPI_Host = "basketapi1.p.rapidapi.com";
        connection.setRequestProperty("X-RapidAPI-Key", X_RapidAPI_Key);
        connection.setRequestProperty("X-RapidAPI-Host", X_RapidAPI_Host);
        connection.setRequestMethod("GET");
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), charsetName: "UTF-8"));
        String str;
        while ((str = in.readLine()) != null) {
            response += str;
        }
        in.close();
    } catch (IOException e) {
        System.out.println("exception");
    }
    return response;
}
```

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

Here is the sample reply, we can see it is in JSON format:

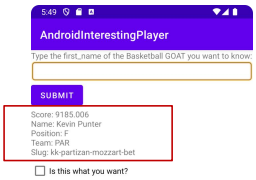

```
potential-space-cod-7x4w4j4grqq2x9g9-8080.app.github.dev/GetPlayer?first_name=kevin

{"score":9185.006,"type":"player","entity":{"country":{"alpha2":"US","name":"USA"},"userCount":1641,"name":"Kevin Punter","id":998697,"position":"F","team":{"country":
{"alpha2":"RS","name":"Serbia"},"gender":"M","type":0,"teamColors":{"secondary":"#ffffff","text":"#ffffff","primary":"#000000"},"userCount":48399,"nameCode":"PAR","name":"KK Partizan Mozzart
Bet","national":false,"disabled":false,"id":6637,"shortName":"Partizan Mozzart","sport":{"name":"Basketball","id":2,"slug":"basketball"},"slug":"kk-partizan-mozzart-bet"},"shortName":"K. Punter","slug":"punter-kevin"}}
```

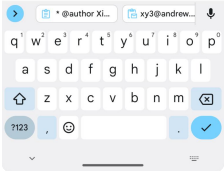

We can also double confirm from the GetPlayerModel.java from the Intellij server side code. We are returning currentPlayer for the generateInfo function, which is a JSONObject type.

```
Project4Task2 > Version control                                   TomEE 10.0.27
GetPlayerModel.java | GetPlayerServlet.java | dashboard.jsp | result.jsp | index.jsp
62          */
            1 usage
63          public static JSONObject generateInfo(String first_name) throws ParseException {
64              String address = "https://basketapi1.p.rapidapi.com/api/basketball/search/" + first_name;
65              SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MMM dd, yyyy");
66              String output = api_fetch(address);
67              JSONObject output_object = (JSONObject) JSONValue.parse(output);
68              JSONArray results = (JSONArray) output_object.get("results");
69              Random random = new Random();
70              int randomNumber = random.nextInt( bound: (results.size() - 1 - 0) + 1) + 0;
71              JSONObject currentPlayer = null;
72              if (results != null) {
73                  currentPlayer = (JSONObject) results.get(randomNumber);
74                  JSONObject entity = (JSONObject) currentPlayer.get("entity");
75                  String full_name = (String) entity.get("name");
76                  String position = (String) entity.get("position");
77                  String teamSlug = "no-team";
78                  String nameCode = "N/A";
79                  Double score = 0.0;
80                  if (full_name.toLowerCase().contains(first_name.toLowerCase())) {
81                      if (entity.get("team") != null) {
82                          JSONObject team = (JSONObject) entity.get("team");
83                          teamSlug = (String) team.get("slug");
84                          nameCode = (String) team.get("nameCode");
85                          score = (Double) currentPlayer.get("score");
86                      }
87                  }
88                  String time = sdf.format(new Date(System.currentTimeMillis()));
89                  connect();
90                  pushToMongoDB(first_name, full_name, position, teamSlug, nameCode, score, time);
91              }
92              return currentPlayer;
ject4Task2 > src > main > java > ds > project4task2 > GetPlayerModel > generateInfo
```

e. Displays new information to the user

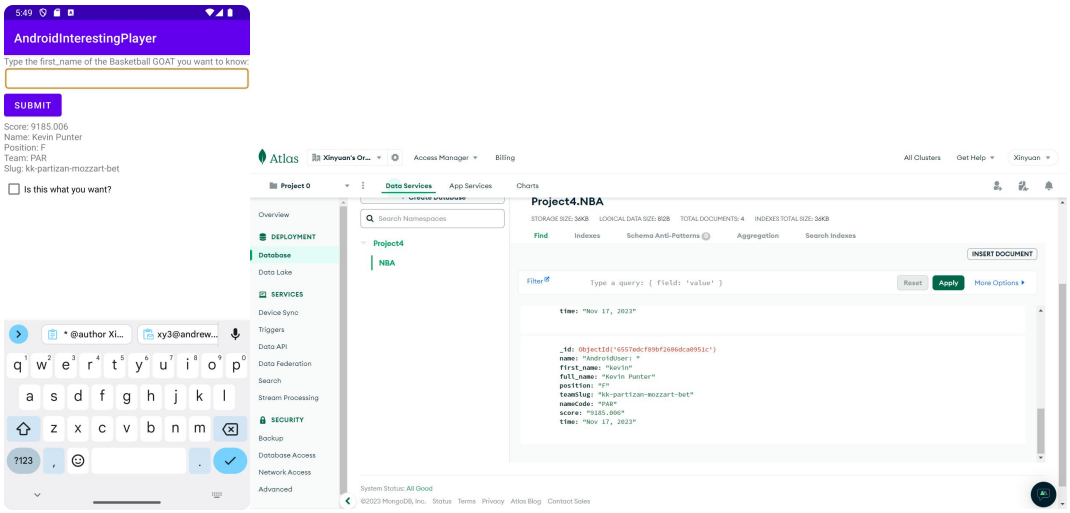
See the screenshot below. After users input the first_name of the basketball players they want to search, the information such as full_name, team, position, slug and score will be seized from the JSONObject response, and displayed on the userface.



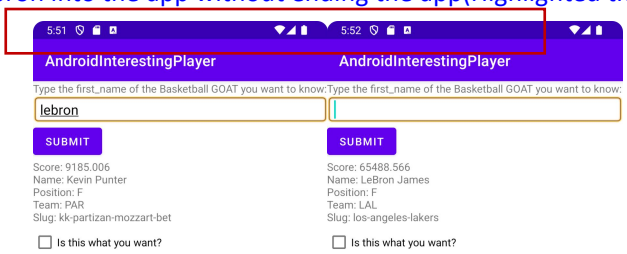


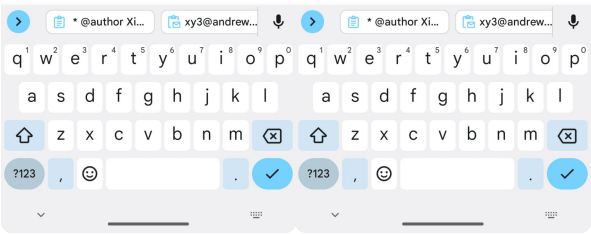
f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

We can see that I just search Kevin, and get Kevin Punter as response:



Now, I am typing lebron into the app without ending the app(Highlighted the screenshot time):





2. Implement a web service

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

In my web app project:

Model: GetPlayerModel.java

View: result.jsp

Controller: GetPlayerServlet.java

b. Receives an HTTP request from the native Android application

My GetPlayerServlet.java receives the HTTP GET request with the argument "first_name". It passes this search string, namely first_name on to the model.

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

-10 if you use a banned API

-10 if screen scrape instead of fetching XML or JSON via a published API

GetPlayerModel.java makes an HTTP request to the third-party API "basketapi1.p.rapidapi.com" (See attached), in the format of: basketapi1.p.rapidapi.com/api/basketball/search/kevin

*kevin can be replaced by another user_Input.

GetPlayerModel.java × GetPlayerServlet.java

```java
     * Generate player information by making an API call, extracting relevant data,
     * and storing it in a MongoDB database.
     *
     * @param first_name The first name of the basketball player.
     * @return The JSON object containing information about the selected player.
     * @throws ParseException If there's an error parsing the JSON response.
     */
    1 usage
    public static JSONObject generateInfo(String first_name) throws ParseException {
        String address = "https://basketapi1.p.rapidapi.com/api/basketball/search/" + first_name;
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MMM dd, yyyy");
        String output = api_fetch(address);
        JSONObject output_object = (JSONObject) JSONValue.parse(output);
        JSONArray results = (JSONArray) output_object.get("results");
        Random random = new Random();
        int randomNumber = random.nextInt( bound: (results.size() - 1 - 0) + 1) + 0;
        JSONObject currentPlayer = null;
        if (results != null) {
            currentPlayer = (JSONObject) results.get(randomNumber);
            JSONObject entity = (JSONObject) currentPlayer.get("entity");
            String full_name = (String) entity.get("name");
            String position = (String) entity.get("position");
            String teamSlug = "no-team";
            String nameCode = "N/A";
            Double score = 0.0;
```

```java
static private String api_fetch(String urlString) {
    String response = "";
    try {
        URL url = new URL(urlString);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        String X_RapidAPI_Key = "efb11adcfemsh287c614b1afa931p1bb807jsndcdb97d50856";
        String X_RapidAPI_Host = "basketapi1.p.rapidapi.com";
        connection.setRequestProperty("X-RapidAPI-Key", X_RapidAPI_Key);
        connection.setRequestProperty("X-RapidAPI-Host", X_RapidAPI_Host);
        connection.setRequestMethod("GET");
        BufferedReader in = new BufferedReader(new InputStreamReader(connection.getInputStream(), charsetName: "UTF-8"));
        String str;
        while ((str = in.readLine()) != null) {
            response += str;
        }
        in.close();
    } catch (IOException e) {
        System.out.println("exception");
    }
    return response;
}
```

It parses the json response and extracts only one player at a time(the player is randomly selected, but associated with the first_name users input), then respond to the Android application(Attached is the randomly selected player parsing code):

```java
public static JSONObject generateInfo(String first_name) throws ParseException {
    String address = "https://basketapi1.p.rapidapi.com/api/basketball/search/" + first_name;
    SimpleDateFormat sdf = new SimpleDateFormat( pattern: "MMM dd, yyyy");
    String output = api_fetch(address);
    JSONObject output_object = (JSONObject) JSONValue.parse(output);
    JSONArray results = (JSONArray) output_object.get("results");
    Random random = new Random();
    int randomNumber = random.nextInt( bound: (results.size() - 1 - 0) + 1) + 0;
    JSONObject currentPlayer = null;
    if (results != null) {
        currentPlayer = (JSONObject) results.get(randomNumber);
        JSONObject entity = (JSONObject) currentPlayer.get("entity");
        String full_name = (String) entity.get("name");
        String position = (String) entity.get("position");
        String teamSlug = "no-team";
        String nameCode = "N/A";
        Double score = 0.0;
        if (full_name.toLowerCase().contains(first_name.toLowerCase())) {
            if (entity.get("team") != null) {
                JSONObject team = (JSONObject) entity.get("team");
                teamSlug = (String) team.get("slug");
                nameCode = (String) team.get("nameCode");
                score = (Double) currentPlayer.get("score");
            }
        }
        String time = sdf.format(new Date(System.currentTimeMillis()));
        connect();
        pushToMongoDB(first_name, full_name, position, teamSlug, nameCode, score, time);
    }
```

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

-5 if more information is returned to the Android app that is needed, forcing the mobile app to do more computing than is necessary. The web service should select and pass on only the information that the mobile app needs.

See the sample response, it extract the information of only one player related to the first_name users input at a time, in the format of json.

potential-space-cod-7x4w4j4grqq2x9g9-8080.app.github.dev/GetPlayer?first_name=kevin

All Bookmarks

{"score":9185.006,"type":"player","entity":{"country":{"alpha2":"US","name":"USA"},"userCount":1641,"name":"Kevin Punter","id":998697,"position":"F","team":{"country":{"alpha2":"RS","name":"Serbia"},"gender":"M","type":0,"teamColors":{"secondary":"#ffffff","text":"#ffffff","primary":"#000000"},"userCount":48399,"nameCode":"PAR","name":"KK Partizan Mozzart Bet","national":false,"disabled":false,"id":6637,"shortName":"Partizan Mozzart","sport":{"name":"Basketball","id":2,"slug":"basketball"},"slug":"kk-partizan-mozzart-bet"},"shortName":"K. Punter","slug":"punter-kevin"}}

4. Log useful information

At least 6 pieces of information is logged for each request/reply with the mobile phone. It should include information about the request from the mobile phone, information about the request and reply to the 3rd party API, and information about the reply to the mobile phone. (You should NOT log data from interactions from the operations dashboard.)

See attached, we record the first_name(which is exactly what users have input), full_name, position, teamSlug, nameCode, score, time.

```java
        Random random = new Random();
        int randomNumber = random.nextInt( bound: (results.size() - 1 - 0) + 1) + 0;
        JSONObject currentPlayer = null;
        if (results != null) {
            currentPlayer = (JSONObject) results.get(randomNumber);
            JSONObject entity = (JSONObject) currentPlayer.get("entity");
            String full_name = (String) entity.get("name");
            String position = (String) entity.get("position");
            String teamSlug = "no-team";
            String nameCode = "N/A";
            Double score = 0.0;
            if (full_name.toLowerCase().contains(first_name.toLowerCase())) {
                if (entity.get("team") != null) {
                    JSONObject team = (JSONObject) entity.get("team");
                    teamSlug = (String) team.get("slug");
                    nameCode = (String) team.get("nameCode");
                    score = (Double) currentPlayer.get("score");
                }
            }
            String time = sdf.format(new Date(System.currentTimeMillis()));
            connect();
            pushToMongoDB(first_name, full_name, position, teamSlug, nameCode, score, time);
        }
        return currentPlayer;
    }
```

5. Store the log information in a database

The web service can connect, store, and retrieve information from a MongoDB database in the cloud.

Connect:

```java
    /**
     * Establish a connection to the MongoDB database and retrieves data from
     * the "NBA" collection; then invoke the statistics method to perform
     * statistical analysis on the retrieved data.
     */
    //Work Cited:https://www.mongodb.com/docs/atlas/tutorial/insert-data-into-your-cluster/
    //Work Cited:https://www.mongodb.com/docs/drivers/java/sync/v4.3/usage-examples/findOne/
    2 usages
    static public void connect() {
        MongoClient mongoClient = MongoClients.create(settings);
        MongoDatabase database = mongoClient.getDatabase( s: "Project4");
        MongoCollection<Document> col = database.getCollection( s: "NBA");
        FindIterable<Document> iterDoc = col.find();
        mongodb_result = new ArrayList<>();
        MongoCursor<Document> cursor = iterDoc.iterator();
        statistics(cursor);
    }
```

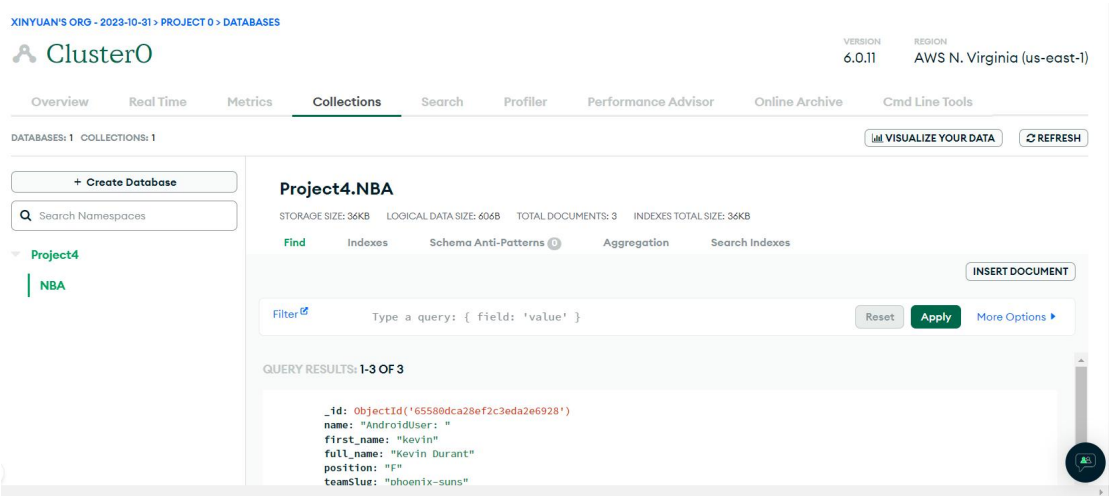
Store:

```java
GetPlayerModel.java  GetPlayerServlet.java  dashboard.jsp  result.jsp  index.jsp

    /**
     * Inserts player information into the MongoDB database in the "NBA" collection.
     *
     * @param first_name The first name of the basketball player.
     * @param full_name The full name of the basketball player.
     * @param position The position of the basketball player.
     * @param teamSlug The team slug of the basketball player.
     * @param nameCode The name code of the basketball player.
     * @param score The score of the basketball player.
     * @param time The timestamp of when the information was stored.
     */
    //Work Cited:https://www.mongodb.com/docs/atlas/tutorial/insert-data-into-your-cluster/
    1 usage
    static private void pushToMongoDB(String first_name, String full_name, String position, String teamSlug, String nameCode, Double score, String time) {
        MongoClient mongoClient = MongoClients.create(settings);
        MongoDatabase database = mongoClient.getDatabase( s: "Project4");
        MongoCollection col = database.getCollection( s: "NBA");
        Document doc = new Document("name", "AndroidUser: ");
        doc.append("first_name", first_name);
        doc.append("full_name", full_name);
        doc.append("position", position);
        doc.append("teamSlug", teamSlug);
        doc.append("nameCode", nameCode);
        doc.append("score", String.valueOf(score));
        doc.append("time", time);
        col.insertOne(doc);
    }
```

Retrieve:

```java
GetPlayerModel.java  GetPlayerServlet.java  dashboard.jsp  result.jsp  index.jsp
C:\Users\Administrator\Desktop\intelij-Project4Task2\Project4Task2\src\main\java\ds\project4task2\GetPlayerModel.java

    */
    1 usage
    static public void statistics(MongoCursor<Document> cursor) {
        int totalCount = 0;
        double totalScore = 0;
        while (cursor.hasNext()) {
            Document document = cursor.next();
            int get1 = keyword_frequency.getOrDefault((String) document.get("first_name"),  defaultValue: 0);
            keyword_frequency.put((String) document.get("first_name"), get1 + 1);

            if (document.containsKey("score")) {
                totalCount++;
                totalScore += Double.parseDouble(document.get("score").toString());
            }
            String s = document.getString( key: "name") + document.get("first_name") + ", " + document.getString( key: "full_name") + ", " + document.getString( key: "position"
                    document.getString( key: "teamSlug") + ", " + document.getString( key: "nameCode") + ", " + document.get("score") + ", " + document.get("time");
            mongodb_result.add(s);
        }
        for (Map.Entry<String, Integer> entry : keyword_frequency.entrySet()) {
            System.out.println(entry.getKey() + ": " + entry.getValue());
            currentSaverForTheToppestFirstName.add(new AbstractMap.SimpleEntry<>(entry.getKey(), entry.getValue()));
            if (currentSaverForTheToppestFirstName.size() > 1) {
                currentSaverForTheToppestFirstName.poll();
            }
        }
        if (totalCount > 0) {
            average_score = totalScore / totalCount;
        }
        searchAmount = totalCount;
    }
```

MongoDB side:



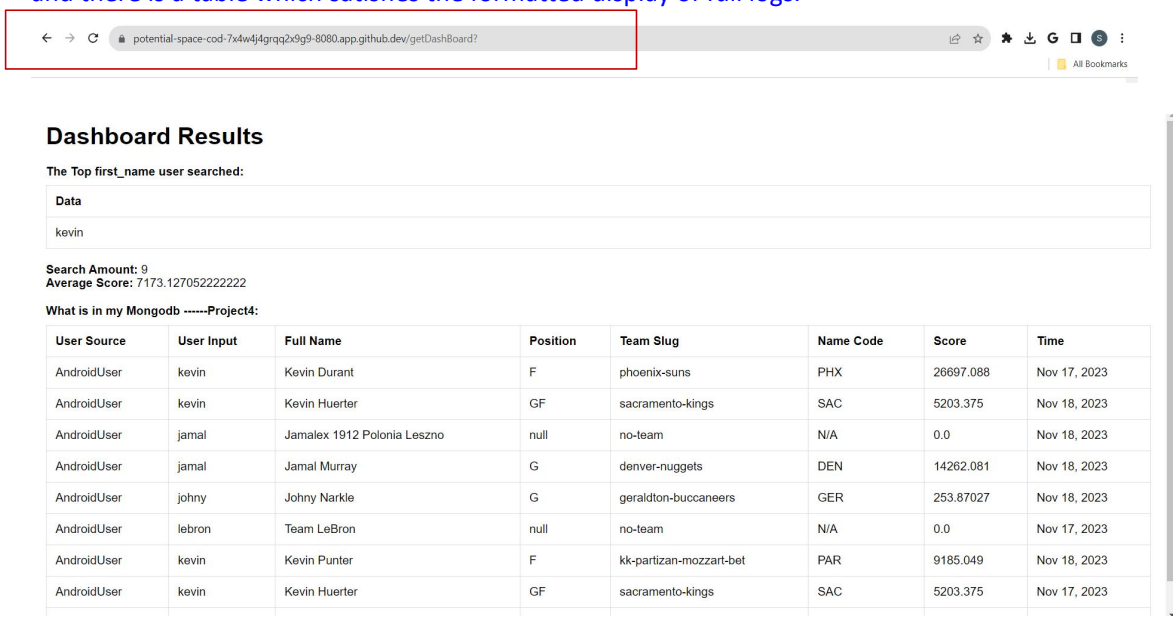
6. Display operations analytics and full logs on a web-based dashboard

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

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

c. The dashboard displays formatted full logs.

See attached. I have suing a unique URL address:
https://potential-space-cod-7x4w4j4grqq2x9g9-8080.app.github.dev/getDashBoard?
The dashboard counts the total amount of user inputs, the first_name that users have input the most, calculate the average score of all the selected player paired with the first_names user inputs, or we can say the average score of all the players be searched and stored in the mongodb, and there is a table which satisfies the formatted display of full logs.



Dashboard Results

The Top first_name user searched:

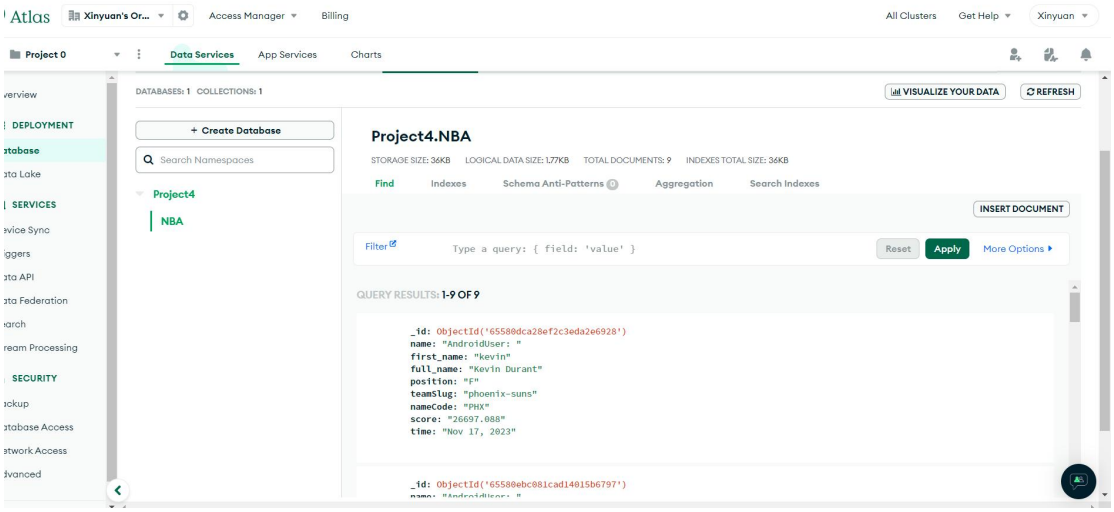
Data
kevin

Search Amount: 9
Average Score: 7173.127052222222

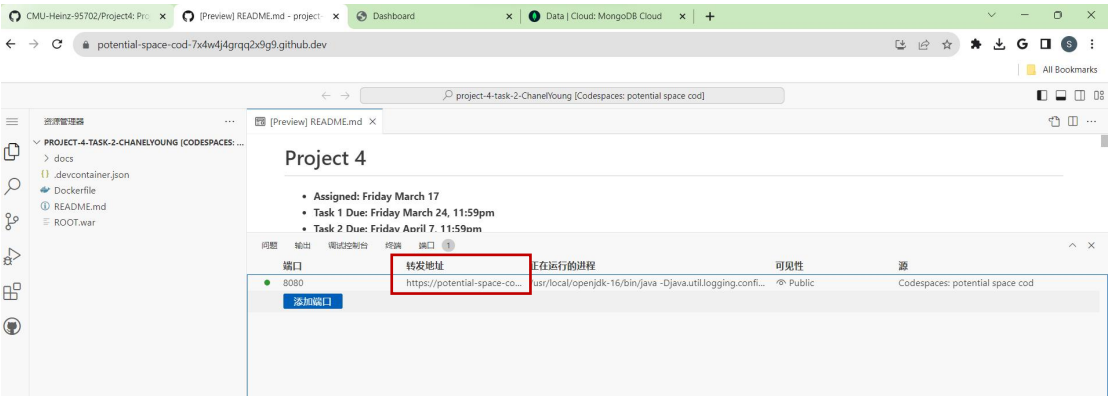
What is in my Mongodb ------Project4:

User Source	User Input	Full Name	Position	Team Slug	Name Code	Score	Time
AndroidUser	kevin	Kevin Durant	F	phoenix-suns	PHX	26697.088	Nov 17, 2023
AndroidUser	kevin	Kevin Huerter	GF	sacramento-kings	SAC	5203.375	Nov 18, 2023
AndroidUser	jamal	Jamalex 1912 Polonia Leszno	null	no-team	N/A	0.0	Nov 18, 2023
AndroidUser	jamal	Jamal Murray	G	denver-nuggets	DEN	14262.081	Nov 18, 2023
AndroidUser	johny	Johny Narkle	G	geraldton-buccaneers	GER	253.87027	Nov 18, 2023
AndroidUser	lebron	Team LeBron	null	no-team	N/A	0.0	Nov 17, 2023
AndroidUser	kevin	Kevin Punter	F	kk-partizan-mozzart-bet	PAR	9185.049	Nov 18, 2023
AndroidUser	kevin	Kevin Huerter	GF	sacramento-kings	SAC	5203.375	Nov 17, 2023

The amount aligns with the mongodb side.



7. Deploy the web service to GitHub Codespaces

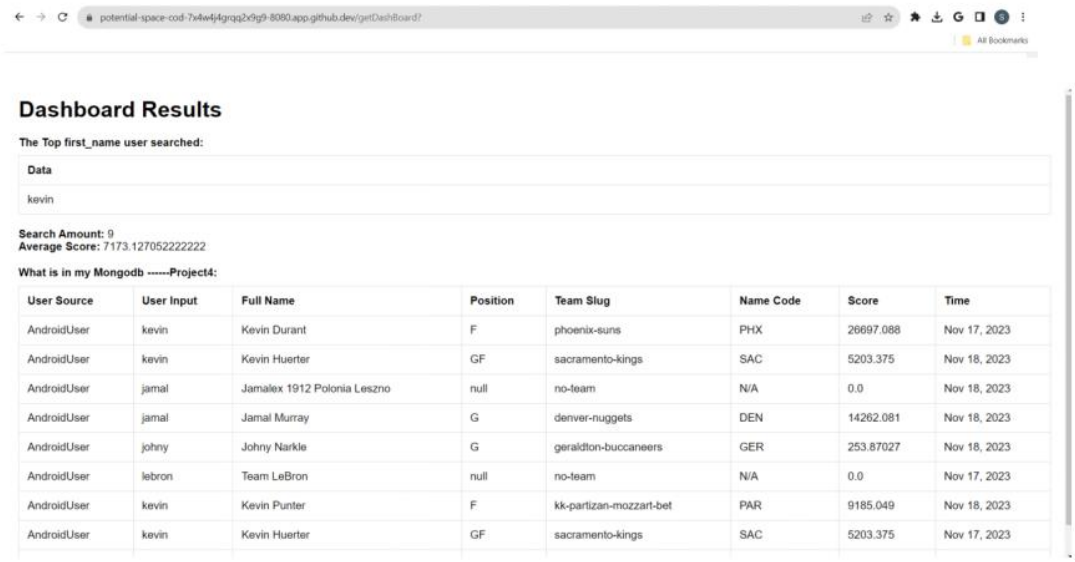


After clicking the address here, you will be directed to the following page, to view the dashboard, click the "Click Here" button after the sentence: Details for my current MongoDB: Project4, to simulate the Android App about player search, do the first blank and its click.

Welcome to explore the world of basketball!

WHO IS YOUR GOAT?

Give me the first name of the player you want to know: [ ]
Click Here
Details for my current MongoDB: Project4 Click Here



Dashboard Results

The Top first_name user searched:

Data
kevin

Search Amount: 9
Average Score: 7173.127052222222

What is in my Mongodb ------Project4:

User Source	User Input	Full Name	Position	Team Slug	Name Code	Score	Time
AndroidUser	kevin	Kevin Durant	F	phoenix-suns	PHX	26697.088	Nov 17, 2023
AndroidUser	kevin	Kevin Huerter	GF	sacramento-kings	SAC	5203.375	Nov 18, 2023
AndroidUser	jamal	Jamalex 1912 Polonia Leszno	null	no-team	N/A	0.0	Nov 18, 2023
AndroidUser	jamal	Jamal Murray	G	denver-nuggets	DEN	14262.081	Nov 18, 2023
AndroidUser	johny	Johny Narkle	G	geraldton-buccaneers	GER	253.87027	Nov 18, 2023
AndroidUser	lebron	Team LeBron	null	no-team	N/A	0.0	Nov 17, 2023
AndroidUser	kevin	Kevin Punter	F	kk-partizan-mozzart-bet	PAR	9185.049	Nov 18, 2023
AndroidUser	kevin	Kevin Huerter	GF	sacramento-kings	SAC	5203.375	Nov 17, 2023

Work Cited:
https://rapidapi.com/fluis.lacasse/api/basketapi1