University of New York Tirana

Assignment 2

Statstical Processing: Covid Cases

Professor:
Elton Ballhysa
Advanced Java
Department of Computer Science

Group:
Regi Nishani,
Daniele Llazo

Deadline: 11 February 2022

Contents

1 Introduction

1.1 Target Application

For the second milestone of this project, we are asked to work on the same covid data, but this time using a database, that will also be used to authenticate the users. We will have two different types of users, where the regular ones will have more functionalities:

- An anonymous user will only be able to see last day data and cumulative data for countries.
- An anonymous user will also be able to filter and sort the data available.
- A regular user, by the other hand, will be able to do the same, but also make "CRUD" operations, as well as update his profile.

We will have to use an API for the registered users, for some of the functionalities above.

1.2 Note for the professor

- Please note that lines in the screenshots below are subjects to change, and that reflect the code we wrote prior to formatting.
- Please note that we have used Tomcat as application server.

2 Database

2.1 MySQL Database

MySQL server is a open-source relational database management system which is a major support for web based applications. Databases and related tables are the main component of many websites and applications as the data is stored and exchanged over the web. Even all social networking websites mainly Facebook, Twitter, and Google depends on MySQL data which are designed and optimized for such purpose. For all these reasons, MySQL server becomes the default choice for web applications.

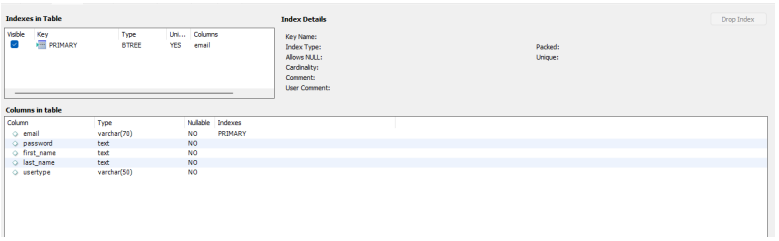
MySQL server is used for data operations like querying, sorting, filtering, grouping, modifying and joining the tables. Before learning the commonly used queries, let us look into some of the advantages of MySQL.

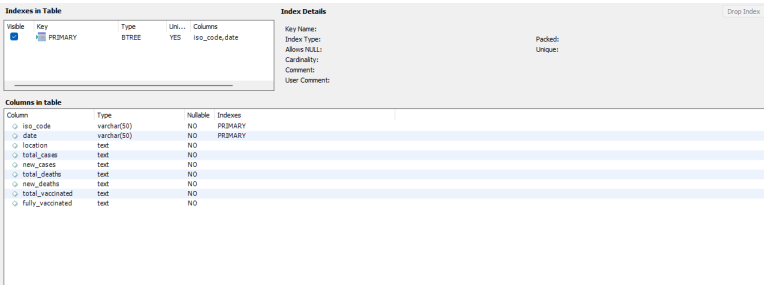
Advantages of MySQL :

- Fast and high Performance database.
- Easy to use, maintain and administer.
- Easily available and maintain integrity of database.
- Provides scalability, usability and reliability.
- Low cost hardware.
- MySQL can read simple and complex queries and write operations.
- InnoDB is default and widely used storage engine.
- Provides strong indexing support.
- Provides SSL support for secured connections.
- Provides powerful data encryption and accuracy.
- Provides Cross-platform compatibility.
- Provides minimized code repetition.

2.2 Tables

We have created two tables, one for the user and one for the data to be displayed.





3 Web Tier

3.1 Java Server Faces

JSF technology includes a set of APIs, which represent different UI components and helps in managing their states. These APIs further help in handling events on the UI components and validate user inputs through the UI components. JSF framework provides the flexibility of creating simple as well as complex applications as this technology uses the most popular Java server technologies (Servlet and Java Server Page) and does not limit a developer to a specific markup language or client device. The UI component classes bundled with JSF APIs contain logic implementation for various component functionalities and do not have any client-specific presentation logic therefore, the JSF UI components can be rendered for different client devices. Currently, SF provides a custom renderer and Java Server Page(JSP) custom tag for rendering UI components for an HTML client.

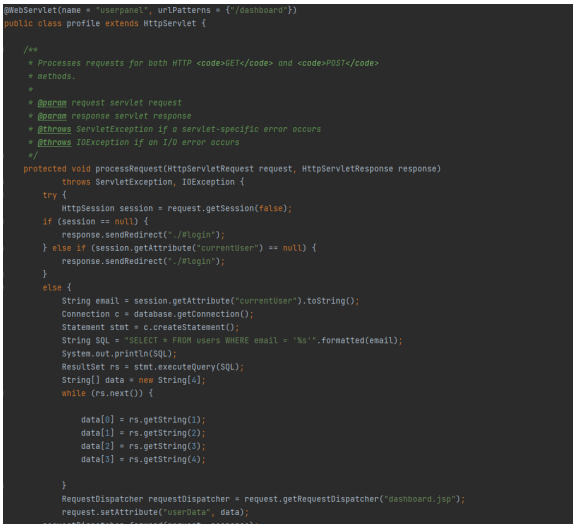
JSF is robust Web application framework that implements an event programming model to handle different events and actions performed by the client on different UI components. To handle each event, a listener should be registered on server side. While developing a Web application, a developer has to write navigation rules inside the source code to navigate from one Web page to another. JSF provides a simple way to define navigation rules in a configuration file and display different error messages showing the real cause of errors to clients. These messages are generated while validating user inputs against some validation rule and can be displayed on the same page that contains the UI components.

There are different Web application frameworks that implement one or more of MVC design pattern. JSF is based on MVC2 pattern and this pattern is based on component type development. In this pattern, the developers have to concentrate only on their respective component introduces separate layers, such as model view and controller and helps the developer to concentrate on a single type of component by making a Web application easy to maintain. The different categories of components, such as model view, and controller are created for different functionalities, such as use of Text Field, Dialog Box Simple Label and Color Chooser and can be used separately

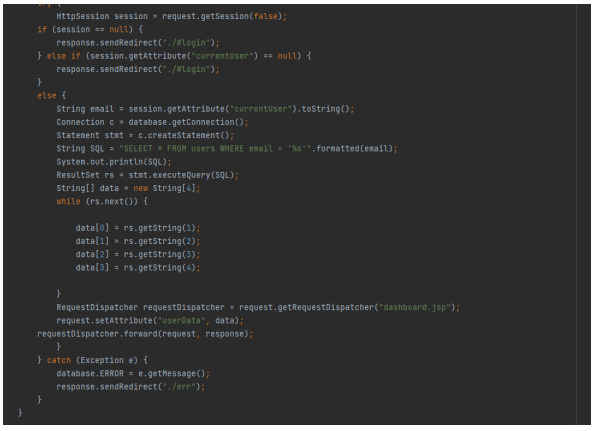
3.2 Our Implementation of Web Services

```
@WebServlet(name = "analysis", urlPatterns = {"/analysis"})
public class analyser extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        try {
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("analysis_tables.jsp");
            HttpSession session = request.getSession(false);
            String SQL = "";
            Connection c = database.getConnection();
            Statement stmt = c.createStatement();
            ArrayList<String[]> data = new ArrayList<>();
            if (session == null) {
                SQL = "select DISTINCT iso_code,location,total_Cases,new_Cases,total_deaths,new_deaths from data where date="
                        + "'" + database.getDate() + "'";

                ResultSet rs = stmt.executeQuery(SQL);

                while (rs.next()) {
                    String[] r = new String[6];
                    r[0] = rs.getString(1);
                    r[1] = rs.getString(2);
                    r[2] = rs.getString(3);
                    r[3] = rs.getString(4);
                    r[4] = rs.getString(5);
                    r[5] = rs.getString(6);
                    data.add(r);
                }
```

```
                ResultSet rs = stmt.executeQuery(SQL);

                while (rs.next()) {
                    String[] r = new String[6];
                    r[0] = rs.getString(1);
                    r[1] = rs.getString(2);
                    r[2] = rs.getString(3);
                    r[3] = rs.getString(4);
                    r[4] = rs.getString(5);
                    r[5] = rs.getString(6);
                    data.add(r);
                }
                request.setAttribute("email", "0");
                request.setAttribute("password", "0");
            } else {
                SQL = "select DISTINCT iso_code,location,total_Cases,new_Cases,total_deaths,new_deaths from data";
                ResultSet rs = stmt.executeQuery(SQL);
                while (rs.next()) {
                    String[] r = new String[6];
                    r[0] = rs.getString(1);
                    r[1] = rs.getString(2);
                    r[2] = rs.getString(3);
                    r[3] = rs.getString(4);
                    r[4] = rs.getString(5);
                    r[5] = rs.getString(6);
                    data.add(r);
                }
                request.setAttribute("email", session.getAttribute("currentUser"));
                request.setAttribute("password", session.getAttribute("password"));
            }

            request.setAttribute("0", data);
            c.close();
            requestDispatcher.forward(request, response);
        } catch (Exception e) {
            database.ERROR = e.getLocalizedMessage();
            response.sendRedirect("./err");
        }

    }
```



```java
@WebServlet(name = "userpanel", urlPatterns = {"/dashboard"})
public class profile extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            HttpSession session = request.getSession(false);
        if (session == null) {
            response.sendRedirect("./#login");
        } else if (session.getAttribute("currentUser") == null) {
            response.sendRedirect("./#login");
        }
        else {
            String email = session.getAttribute("currentUser").toString();
            Connection c = database.getConnection();
            Statement stmt = c.createStatement();
            String SQL = "SELECT * FROM users WHERE email = '%s'".formatted(email);
            System.out.println(SQL);
            ResultSet rs = stmt.executeQuery(SQL);
            String[] data = new String[4];
            while (rs.next()) {

                data[0] = rs.getString(1);
                data[1] = rs.getString(2);
                data[2] = rs.getString(3);
                data[3] = rs.getString(4);

            }
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("dashboard.jsp");
            request.setAttribute("userData", data);
```



```java
            HttpSession session = request.getSession(false);
        if (session == null) {
            response.sendRedirect("./#login");
        } else if (session.getAttribute("currentUser") == null) {
            response.sendRedirect("./#login");
        }
        else {
            String email = session.getAttribute("currentUser").toString();
            Connection c = database.getConnection();
            Statement stmt = c.createStatement();
            String SQL = "SELECT * FROM users WHERE email = '%s'".formatted(email);
            System.out.println(SQL);
            ResultSet rs = stmt.executeQuery(SQL);
            String[] data = new String[4];
            while (rs.next()) {

                data[0] = rs.getString(1);
                data[1] = rs.getString(2);
                data[2] = rs.getString(3);
                data[3] = rs.getString(4);

            }
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("dashboard.jsp");
            request.setAttribute("userData", data);
        requestDispatcher.forward(request, response);
            }
        } catch (Exception e) {
            database.ERROR = e.getMessage();
            response.sendRedirect("./err");
        }
    }
```

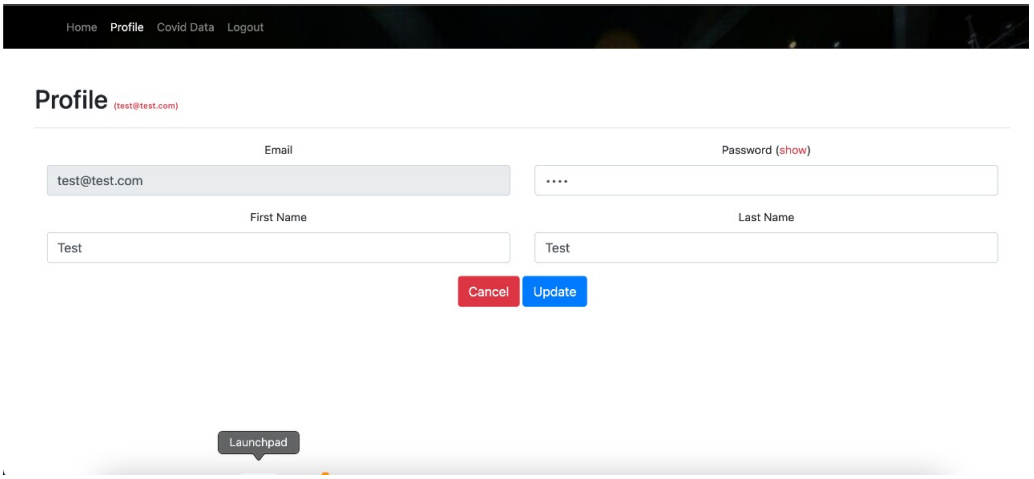
```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try {

        HttpSession session = request.getSession(false);
        No documentation found.
        } else if (session.getAttribute("currentUser") == null) {
            response.sendRedirect("./#login");
        } else {

            Connection c = database.getConnection();
            Statement stmt = c.createStatement();
            String SQL = "select distinct location, iso_code from data";
            System.out.println(SQL);
            ResultSet rs = stmt.executeQuery(SQL);
            ArrayList<String[]> d = new ArrayList<String[]>();
            ArrayList<String[]> d1 = new ArrayList<String[]>();
            while (rs.next()) {
                String[] data = new String[2];
                data[0] = rs.getString(2);
                data[1] = rs.getString(1);
                d.add(data);

            }
            String SQL1 = "select iso_code,date,location,new_cases,new_deaths from data";
            rs = stmt.executeQuery(SQL1);
            while (rs.next()) {
                String[] data = new String[5];
                data[0] = rs.getString(1);
                data[1] = rs.getString(2);
                data[2] = rs.getString(3);
                data[3] = rs.getString(4);
                data[4] = rs.getString(5);
                d1.add(data);

            }
            c.close();
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("Covid_Data.jsp");
            request.setAttribute("countries", d);
            request.setAttribute("covid_data", d1);
```

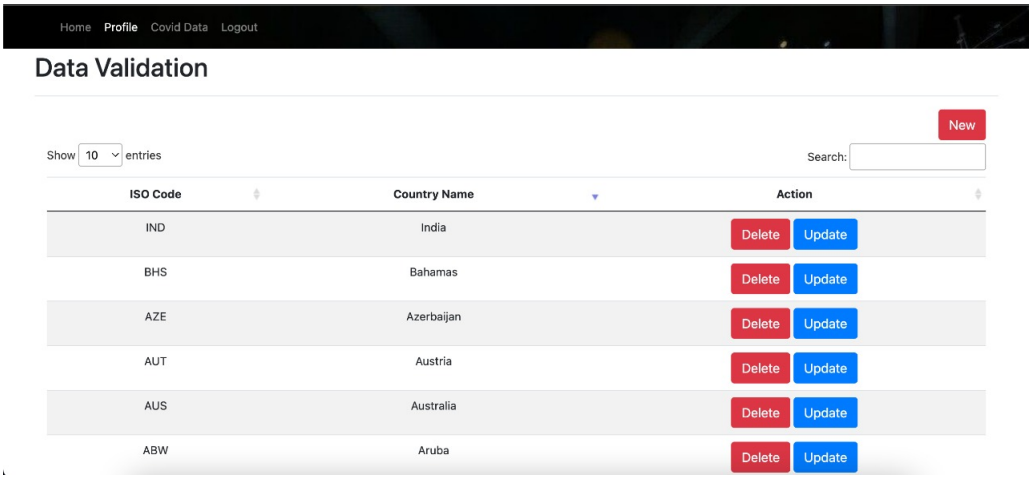
```
            }
            String SQL1 = "select iso_code,date,location,new_cases,new_deaths from data";
            rs = stmt.executeQuery(SQL1);
            while (rs.next()) {
                String[] data = new String[5];
                data[0] = rs.getString(1);
                data[1] = rs.getString(2);
                data[2] = rs.getString(3);
                data[3] = rs.getString(4);
                data[4] = rs.getString(5);
                d1.add(data);

            }
            c.close();
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("Covid_Data.jsp");
            request.setAttribute("countries", d);
            request.setAttribute("covid_data", d1);
            requestDispatcher.forward(request, response);
        }
    } catch (Exception e) {
        database.ERROR = e.getLocalizedMessage();
        response.sendRedirect("./err");
    }
}
```

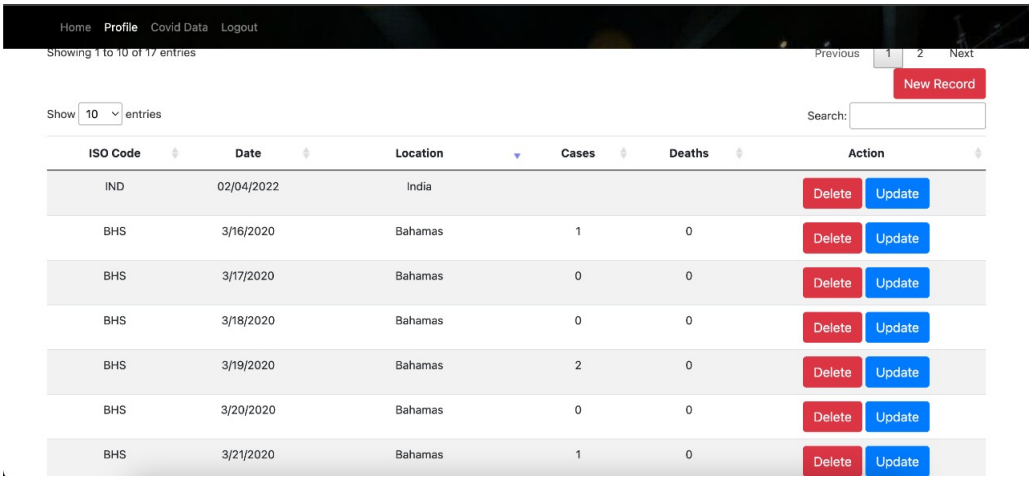
```
@WebServlet(name = "error", urlPatterns = {"/err"})
public class notFound extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific notFound occurs
     * @throws IOException if an I/O notFound occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        RequestDispatcher requestDispatcher = request.getRequestDispatcher("error.jsp");

        requestDispatcher.forward(request, response);
    }
```

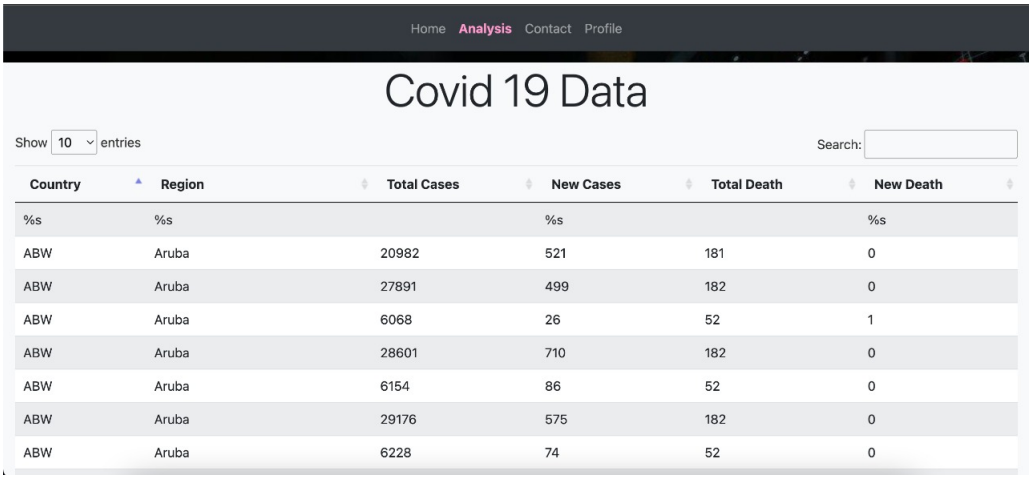
4 UI

We have done an user friendly UI for our application, using HTML, CSS and Javascript. The screenshots below demonstrate that:




Home Profile Covid Data Logout
Data Validation
New
Show 10 entries
Search:
ISO Code
Country Name
Action
IND
India
Delete Update
BHS
Bahamas
Delete Update
AZE
Azerbaijan
Delete Update
AUT
Austria
Delete Update
AUS
Australia
Delete Update
ABW
Aruba
Delete Update


Home Profile Covid Data Logout
Showing 1 to 10 of 17 entries
Previous 1 2 Next
New Record
Show 10 entries
Search:
ISO Code
Date
Location
Cases
Deaths
Action
IND 02/04/2022 India Delete Update
BHS 3/16/2020 Bahamas 1 0 Delete Update
BHS 3/17/2020 Bahamas 0 0 Delete Update
BHS 3/18/2020 Bahamas 0 0 Delete Update
BHS 3/19/2020 Bahamas 2 0 Delete Update
BHS 3/20/2020 Bahamas 0 0 Delete Update
BHS 3/21/2020 Bahamas 1 0 Delete Update


Home Analysis Contact Profile
Covid 19 Data
Show 10 entries
Search:
Country
Region
Total Cases
New Cases
Total Death
New Death
%s %s %s %s
ABW Aruba 20982 521 181 0
ABW Aruba 27891 499 182 0
ABW Aruba 6068 26 52 1
ABW Aruba 28601 710 182 0
ABW Aruba 6154 86 52 0
ABW Aruba 29176 575 182 0
ABW Aruba 6228 74 52 0

5 API

5.1 Rest API

A REST API (also known as RESTful API) is an application programming interface (API or web API) that conforms to the constraints of REST architectural style and allows for interaction with RESTful web services. REST stands for representational state transfer and was created by computer scientist Roy Fielding.

The resources are acted upon by using a set of simple, well-defined operations. Also, the resources have to be decoupled from their representation so that clients can access the content in various formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others.

The clients and servers exchange representations of resources by using a standardized interface and protocol. Typically HTTP is the most used protocol, but REST does not mandate it.

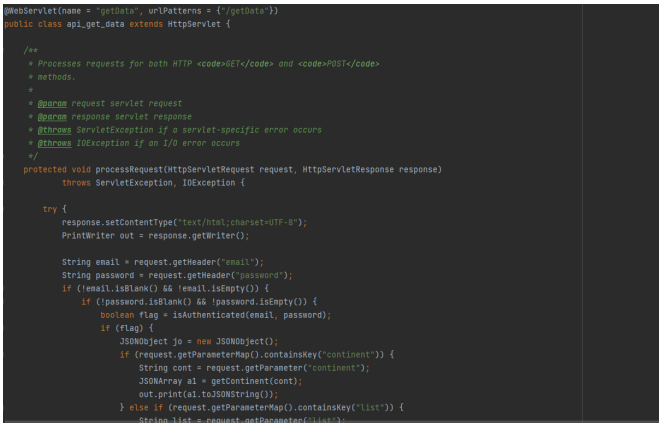
Metadata about the resource is made available and used to control caching, detect transmission errors, negotiate the appropriate representation format, and perform authentication or access control.

And most importantly, every interaction with the server must be stateless.

All these principles help RESTful applications to be simple, lightweight, and fast.

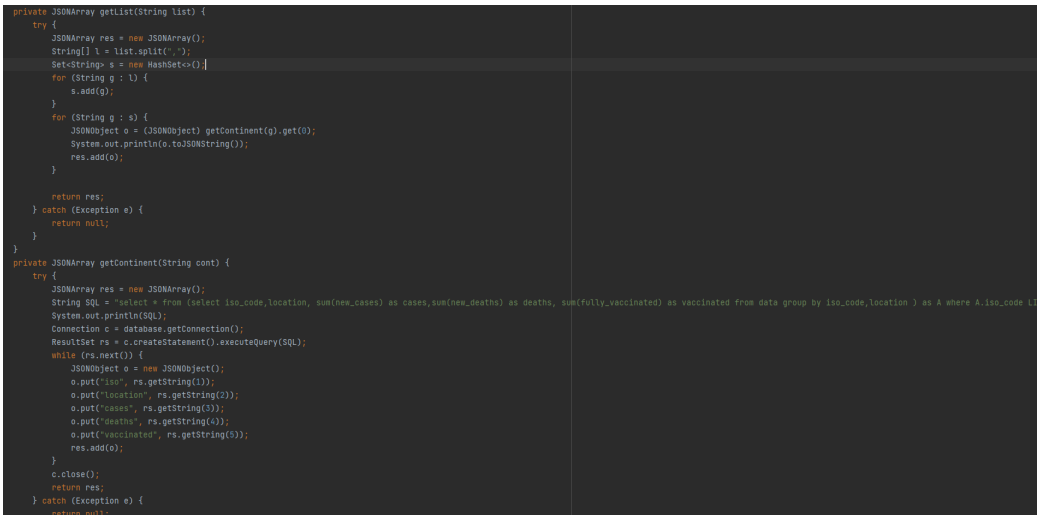
5.2 API implementation

As required, we have used the API to return cumulative covid data for the countries, but only to be used by the registered user.

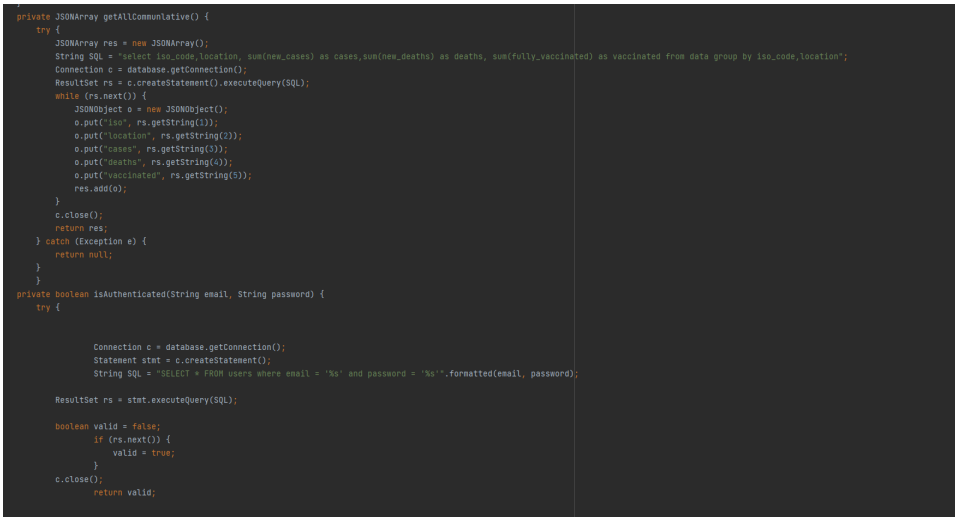


```
@WebServlet(name = "getData", urlPatterns = {"/getData"})
public class api_get_data extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        try {
            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();

            String email = request.getHeader("email");
            String password = request.getHeader("password");
            if (!email.isBlank() && !email.isEmpty()) {
                if (!password.isBlank() && !password.isEmpty()) {
                    boolean flag = isAuthenticated(email, password);
                    if (flag) {
                        JSONObject jo = new JSONObject();
                        if (request.getParameterMap().containsKey("continent")) {
                            String cont = request.getParameter("continent");
                            JSONArray a1 = getContinent(cont);
                            out.print(a1.toJSONString());
                        } else if (request.getParameterMap().containsKey("list")) {
                            String list = request.getParameter("list");
```

```
                    out.print(a1.toJSONString());
                } else if (request.getParameterMap().containsKey("list")) {
                    String list = request.getParameter("list");
                    JSONArray a1 = getList(list);
                    out.print(a1.toJSONString());
                } else {
                    JSONArray a = getAllCommunlative();
                    out.print(a.toJSONString());

                }


                //  out.print(jo.toJSONString());
            } else {
                out.print(
                        "unauthorized  :" + flag);
            }

        } else {

            out.print("Password Is Not Provided");
        }
    } else {

        out.print("Email Is Not Provided");
    }


} catch (Exception e) {
    response.getWriter().write(e.getMessage());
}
}
private JSONArray getList(String list) {
    try {
        JSONArray res = new JSONArray();
```



```
private JSONArray getList(String list) {
    try {
        JSONArray res = new JSONArray();
        String[] l = list.split(",");
        Set<String> s = new HashSet<>();
        for (String g : l) {
            s.add(g);
        }
        for (String g : s) {
            JSONObject o = (JSONObject) getContinent(g).get(0);
            System.out.println(o.toJSONString());
            res.add(o);
        }

        return res;
    } catch (Exception e) {
        return null;
    }
}
private JSONArray getContinent(String cont) {
    try {
        JSONArray res = new JSONArray();
        String SQL = "select * from (select iso_code,location, sum(new_cases) as cases,sum(new_deaths) as deaths, sum(fully_vaccinated) as vaccinated from data group by iso_code,location ) as A where A.iso_code LI
        System.out.println(SQL);
        Connection c = database.getConnection();
        ResultSet rs = c.createStatement().executeQuery(SQL);
        while (rs.next()) {
            JSONObject o = new JSONObject();
            o.put("iso", rs.getString(1));
            o.put("location", rs.getString(2));
            o.put("cases", rs.getString(3));
            o.put("deaths", rs.getString(4));
            o.put("vaccinated", rs.getString(5));
            res.add(o);
        }
        c.close();
        return res;
    } catch (Exception e) {
        return null;
```



```
private JSONArray getAllCommunlative() {
    try {
        JSONArray res = new JSONArray();
        String SQL = "select iso_code,location, sum(new_cases) as cases,sum(new_deaths) as deaths, sum(fully_vaccinated) as vaccinated from data group by iso_code,location";
        Connection c = database.getConnection();
        ResultSet rs = c.createStatement().executeQuery(SQL);
        while (rs.next()) {
            JSONObject o = new JSONObject();
            o.put("iso", rs.getString(1));
            o.put("location", rs.getString(2));
            o.put("cases", rs.getString(3));
            o.put("deaths", rs.getString(4));
            o.put("vaccinated", rs.getString(5));
            res.add(o);
        }
        c.close();
        return res;
    } catch (Exception e) {
        return null;
    }
    }
private boolean isAuthenticated(String email, String password) {
    try {


                Connection c = database.getConnection();
                Statement stmt = c.createStatement();
                String SQL = "SELECT * FROM users where email = '%s' and password = '%s'".formatted(email, password);

        ResultSet rs = stmt.executeQuery(SQL);

        boolean valid = false;
                if (rs.next()) {
                    valid = true;
                }
        c.close();
                return valid;
```

6 Conclusion

All in all, we have been able to fulfill all the requirements of the Milestone2 of the Advanced Java Assignment. We have been able to authenticate the user using a MySQL database. We have been able to implement the API to return the cumulative covid data, by also using the database. We have also implemented an user friendly UI. We have worked together by using Git and GitHub, and also provided you the GitHub repository.

7 References

1. REST API Tutorial (2019). What is REST – Learn to create timeless REST APIs. [online] Restfulapi.net. Available at: https://restfulapi.net/.
2. GeeksforGeeks. (2019). JSF — Java Server Faces. [online] Available at: https://www.geeksforgeeks.org/jsf-java-server-faces/
3. GeeksforGeeks. (2019). MySQL — Common MySQL Queries. [online] Available at: https://www.geeksforgeeks.org/mysql-common-mysql-queries/.