CO3102/CO7102
Coursework 2

Web-based Application - MyCoVTest Hub

Important Dates:

Handed out: 04-Nov-2020
Deadline: 1-Dec-2020 at 16:59 GMT

Please ensure that you submit your work on time.

- This coursework counts as 15% of your final mark.
- Please read guidelines on plagiarism on https://www2.le.ac.uk/offices/sas2/assessments/plagiarism/penalties
- Learning outcome:
  - Use appropriate server-side and client-side scripting languages to create a web application
  - Solve security and session handling issues and use supporting techniques

Coursework Description

The valley of Shangri-La is in lockdown - a deadly virus called SARS-CoV-3 spreading across the world is threatening the future of the valley. Fortunately, a new Rapid Swab Home Test Kit (RSHT) is developed, which can give the test results straight away in a few minutes without laboratory involvement. Shangri-La government plans to conduct mass testing and track down every case of the virus by distributing the home test kits to every household.

Your task is to develop a web application called **MyCoVTest Hub** to help collect the test results and to provide an accurate picture of the local cases, including more detailed demographics, as well as a regional breakdown by postcode.

Requirements

In response to the pandemic, all residents in Shangri-La will receive a Home Test Kit by post. Each pack of the Home Test Kit delivered has a unique Test Tracking Number (TTN). Swab tests that can be easily done at home and anyone tested will get the test result within a few minutes as there is no need to return the test kit to a laboratory for analysis. Anyone completed the test at home will need to report the test result via **MyCoVTest Hub** platform.

To submit a test result via MyCoVTest Hub, you will need to provide some detail as follows:

- Email
- Full name
- Age
- Gender (M/F/Other)
- Address
- Postcode
- TTN Code
- Test result (Positive/Negative/Inconclusive)

In addition to SARS-CoV-3 home test result collection, **MyCoVTest Hub** platform also provides a password-protected Admin dashboard for viewing some real-time demographics statistics, including:

- Total number of positive/negative cases
- Positive cases distribution by postcode/age group*[1] [number of positive cases per postcode/age group]
- Infection rate by postcode
- Infection rate by age group

(Admin login credentials are pre-defined in the system, see Appendix 1.2 for more information)

MyCoVTest Hub should show corresponding messages when:

- TTN code does not match the record in the database.[2]
- Another person has already used the provided TTN code.
- The provided email is already associated with another (used) Home Test Kit
- Invalid username or password (for Admin).

Your tasks are to implement the following functions:

(1) Home Test Self-reporting page (including validation, TTN verification)	[35 marks]
(2) Log-in/sign-out feature for Admin	[20 marks]
(3) Demographics statistics for Admin Dashboard	[35 marks]
(5) Error page(s) (or Ajax error messages)	[10 marks]

Note that you should

- Use appropriate techniques to prevent SQL Injection vulnerabilities.
- Take all necessary measures to prevent unauthorised access to the Admin Dashboard
- Use Secure Hash Algorithm SHA256 to secure passwords (see Appendix 1.3)
- Use cookies to remember the last admin User ID.

The SQL file **Shangri-La.sql** provided on Blackboard contains three tables (Admin, HomeTestKit, TestResult). You are free to edit the provided schema, add extra columns or create new tables if necessary.

Submission

- Zip all files in a single zip file for submission:
  - Your Web Project folder
  - Your SQL schema and data (Your_email_ID.sql)

- The archive should be named **CO3012_CW2_email_id.zip** or **CO7012_CW2_email_id.zip** (e.g. WT_CW2_yh37.zip, please replace "email_id" with your University email id).

[1] (0-17; 18-44 ;45-64, over 65)

[2] See Appendix 1.1 for a list of valid TTN codes for Home Test Kits

Note:

You may use any programming languages or frameworks for the implementation, including but not limited to:

- Java (e.g. Servlet/JSP/Spring)
- JavaScript (e.g. React.js/Express.js//Node.js)
- C# (e.g. ASP.NET Core etc.)
- Python (e.g. Flask, Django etc.)
- Ruby On Rails
- PHP (Laravel)

You are allowed to change the DB schema but make sure that you have imported all tables to the departmental MySQL server (mysql.mcscw3.le.ac.uk) and tested the connection string before submission. You may use **Shangri-La.sql** provided on Blackboard. However, **you do not have to use this file** if you intend to use NoSQL (e.g. Mongodb, Oracle Spatial Graph) or other persistence frameworks (e.g. Spring JPA). You will need to upload the zip file via Blackboard, and you are allowed to re-submit as many times as you like **before** the deadline.

Anonymous marking

We operate an anonymous marking scheme. All submitted WAR files (or other project files) will be deployed to Tomcat using anonymous fingerprinting generated by SHA256.

Appendix

1.1. Below is a list of all valid Test Tracking Number (TTN). A tester will need to enter one of the following TTN codes to report the test result.

```
MM2874Z6
FEQQ6UUG
34GC829B
CB8FBCCM
8RL4ENTK
57UBS5J6
4F7YKH9G
R9KZ2NXL
YBQUVXHL
CCZTQ8KW
```

1.2 Admin credentials

```
Username:admin
Password:12345
 [SHA256-Hash= 5994471ABB01112AFCC18159F6CC74B4F511B99806DA59B3CAF5A9C173CACFC5]

Username:tester
Password:abcde
[SHA256-Hash= 36BBE50ED96841D10443BCB670D6554F0A34B761BE67EC9C4A8AD2C0C44CA42C]
```

* Feel free to edit the sample records in **Shangri-La.sql** or make any changes you deem necessary.

1.3 Generating SHA256 Password hash in Java

See HashGenerator.java

```
public static String getSHA256(String data) {
        String result = null;
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(data.getBytes("UTF-8"));
            return DatatypeConverter.printHexBinary(hash);
        }catch(Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
```

Marking Scheme

F

<30%

- User cannot report the test result; no form validation.
- Login/sign-out/redirection failed; HTTP session (or Spring Security) not in use; not roles/permissions control.
- Admin page not found.
- Error page not found; 404 or exception stack trace printed out to JSP page directly.
- The website cannot communicate with the database backend.

E

40-50%

- Home Test reporting partially implemented (e.g. email/full name etc) without TTN code validation.
- Admin login authentication works to a certain extent; page redirection works but Admin Dashboard is not protected by sessions (e.g. guests can access the result page)
- Admin page exists, but with hard-coded statistical data (or not available)
- Error messages are displayed, but the information provided is minimal (e.g. "An error occurs").
- SQL injection vulnerabilities or security not addressed.

D

50-60%

- Home Test reporting page works to a certain extent; server-side TTN code validation implemented.
- User authentication and redirection works, Admin dashboard pages are protected by sessions, although there are still issues with roles and permissions.
- Admin can view the statistics, despite some minor issues.
- Useful error messages (e.g. "login failed")
- Some measures are taken to reduce the risk of SQL injection (use of preparedStatement)

C

60-70%

- Home Test reporting page works reasonably well.
- Client-sided validation for most fields; server-sided TTN code verification.
- Role/permission control in place despite some minor issues.
- Admin can view the statistics, most of the data are correct.
- Detailed error messages (e.g. "Login failed: incorrect password")
- Some other measures are taken apart from preparedStatement (e.g. filtering special characters in the URLs and text fields etc.)

B

70-80%
Meeting the criteria above in C, and

- Robust control over roles and permissions.
- Use of Ajax for form validations and TTN code verification.
- Admin can view the statistics, and all data are displayed correctly.
- Very detailed error page and Ajax message box.
- Use Cookies to remember last login name in the Admin dashboard
- Use appropriate chart for presenting data. (e.g. Google Chart, C3 JS or Chart JS etc)

A

80%+

Meeting the criteria above in B,
Extra bonus points will be awarded for additional features, included but not limited to:

- Google Map integration
- Responsiveness (RWD)
- REST-API for Admin dashboard report