Server Side Foundation Assessment

Date: Fri Jun 09 2023
Assessment Time: 0900 - 1700 (including meal breaks)

Overview

There are **7 tasks** in this assessment. Complete all tasks.

Passing mark is **65%** (**55 marks**). Total marks is **84**.

Read this entire document before attempting the assessment. There are 12 pages in this document.

Application Overview

A 'front controller' is one or more controllers that are used in web applications to manage the flow of HTTP requests into an application. A typical use case of the front controller is to act as a gatekeeper for requests accessing sensitive information. The front controller will allow or deny a request depending on whether the request is authorised to access the resource. Figure 1 shows a typical processing logic of a front controller.

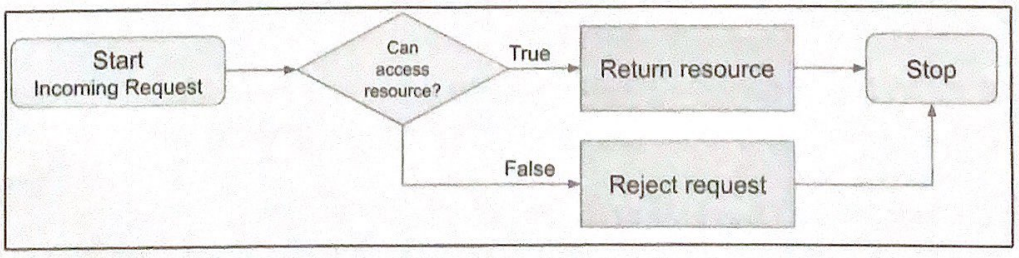

Figure 1 Front controller algorithm

In this assessment you will be implementing a front controller that protects a sensitive web page. Users who wish to access this web page will have to login and be authenticated before access can be granted.

The application consists of the following 3 views. The flow of these views are shown in Figure 2

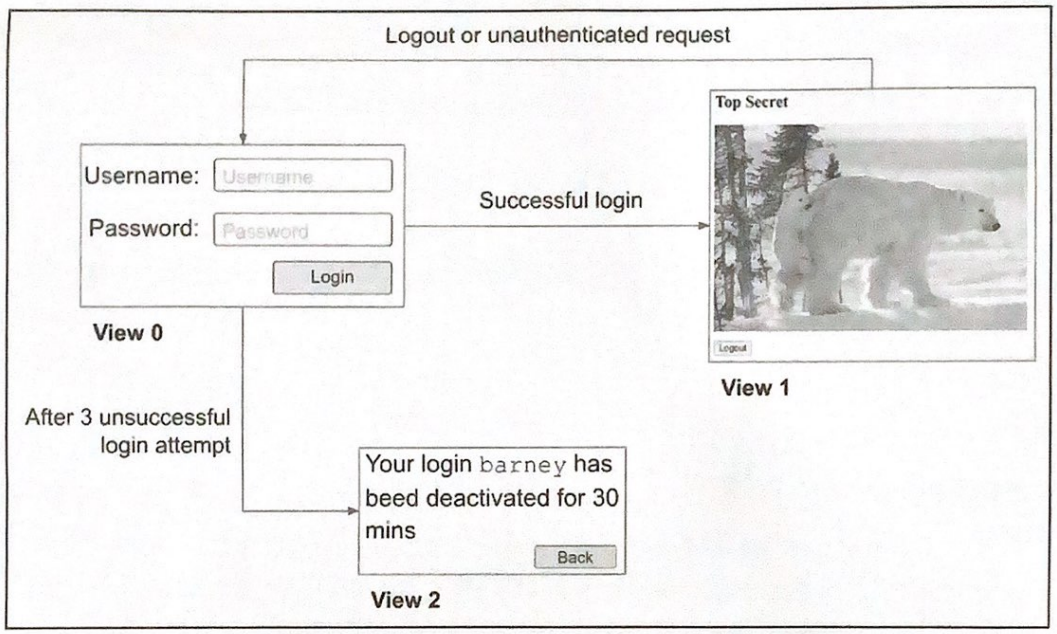

Figure 2 Application flow

Description of the views

- **View 0** - this is the login page where users are required to login (authenticate) before attempting to access the protected resource View 1. View 1 will also display a captcha if the user fails to consecutively login a second and a third time.
- **View 1** - the protected resources which will be displayed after a successful login. Any request to view it without authentication will be redirected to View 0
- **View 2** - the front controller keeps track of the number of unsuccessful logins made by a user. After 3 unsuccessful attempt at login, the login account will be suspended for 30 minutes

The entire login and authentication process of the front controller is illustrated in Figure 3 below

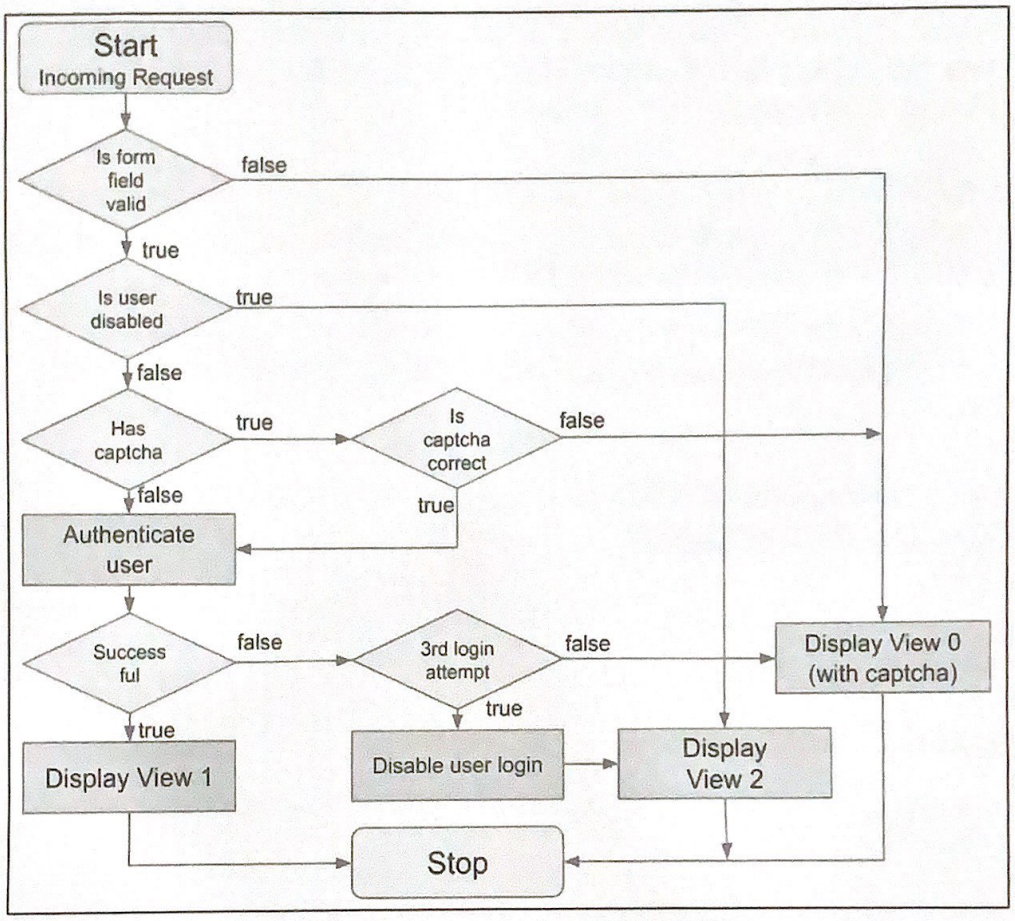

Figure 3 Front controller process flow

Details of the individual views and process steps will be described in subsequent tasks.

Setup

Unzip the provided assessment ZIP template. This is a Spring Boot application running on Java 19. All the dependencies required by this assessment have been added into the `pom.xml` file. Feel free to add any additional dependencies that you may need.

Unzip the archive and initialise the directory as a Git repository; perform a commit and push it to Github. Do not wait until the end of the assessment.

Your remote Github repository must be a **PRIVATE** repository. Make your repository **PUBLIC after 1700 (5.00PM) Fri Mar 24 2023** so that the instructors can access your work.

IMPORTANT: it is important that your assessment repository remain PRIVATE during the duration of the assessment so that your work is only accessible to you and nobody else. In the event that your work is plagiarised by others before the end of the assessment, you will be considered as a willing party in the aiding and abetting of the dishonest act.

Remember to commit and push your work regularly and do not wait until the end of the assessment.

Assessment

Task 1 (5 marks)

The file `view0.html` is the login page (View 0) shown in Figure 4

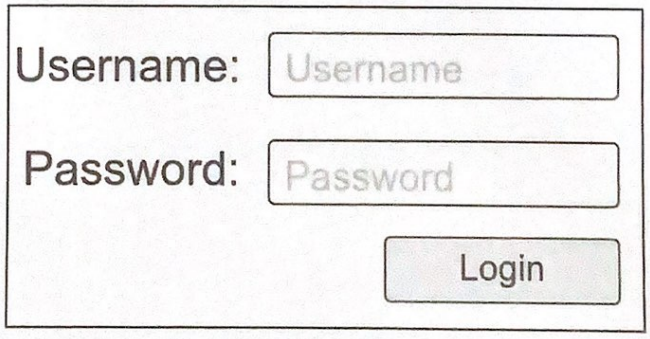

Figure 4 File `view0.html`

Make this the 'landing page' for the application; a landing page is the first view that a user sees when a user opens the web application on the browser.

When the 'Login' button is pressed, it should make the following HTTP request to the front controller to be authenticated

```
POST /login
Content-Type: application/x-www-form-urlencoded
Accept: text/html
\r\n
username=<username>&password=<password>
```

where `<username>` and `<password>` is the username and password that the user has entered.

Use `view0.html` as the landing page; update it so that `view0.html` behaves as described. The `view0.html` file is currently in the `static` directory. You may move the file to any other directory of your choice.

Task 2 (24 marks)

Write a request handler to process the HTTP request made by View 0 from Task 1.

Ensure that the username and password must be at least 2 characters in length. If they are not, redisplay View 0 with an appropriate error message. These errors should not be considered as a login attempt.

If the username and password are syntactically correct, use them to authentication with the authentication REST endpoint on https://authservice-production-e8b2.up.railway.app The endpoint will accept the following HTTP request

```
POST /api/authenticate
Content-Type: application/json
Accept: application/json
```

The authentication POST request should have the following payload

```
{
   "username": <username>,
   "password": <password>
}
```

where `<username>` and `<password>` is the username and password to be authenticated from the HTTP request.

If the authentication check is valid, the authentication endpoint will return the following HTTP request

```
201 Accepted
Content-Type: application/json
\r\n
{ "message", "Authenticated <username>" }
```

'Mark' the user as authenticated and redirect the request to `/protected/view1.html`. More details will be provided in a **Task 5**.

Use the following classes to write your controller and service for this task

- `FrontController.java`
  Write the HTTP request handler in this class. Use the `AuthenticationService.authenticate()` method. Redisplay View 0 if authentication fails
- `AuthenticationService.java`
  Write the HTTP request to the authentication endpoint in the `authenticate()` method; the method should throw an exception if the endpoint returns any errors

You can use the following username and password for testing

Username	Password
fred	fredfred
barney	barneybarney
wilma	wilmawilma
betty	bettybetty

Task 3 (20 marks)

The authentication endpoint will return a 400 range status code if either the POST payload is invalid or the username/password is invalid as show in the following table

Error	Response

Invalid payload, viz the `POST` request's payload is incorrect	`400 Bad Request`
	`Content-Type: application/json`
	`\r\n`
	`{ "message": "Invalid payload" }`

Invalid username and/or password	`401 Unauthorized`
	`Content-Type: application/json`
	`\r\n`
	`{ "message": "Incorrect username and/or password" }`

look at status code to determine if login has failed.

If either of these errors occurs, count the request as a failed login attempt.

Redisplay View 0 with an appropriate error message and a captcha. The captcha is a simple maths problem involving operations on randomly generated numbers. The captcha to be display includes

- 2 random numbers in the range of 1 to 50
- randomly select one of the following mathematical operations to be used on the 2 numbers: plus, minus, divide or multiply
- an input field for the user to enter the data

An example of this is shown in Figure 5

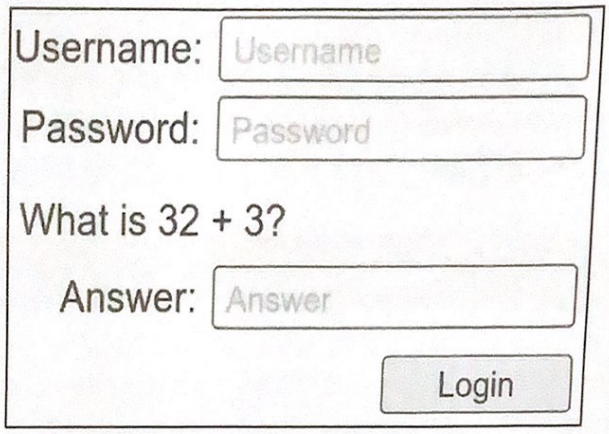

Figure 5 View 0 with captcha

In Figure 5, View 0 displays the captcha is '`32 + 3`'. The user is required to now enter the username, password and answer to the captcha.

When the login button is pressed, perform the usual validation as described in **Task 2**. Before performing the username/password authentication, check if the captcha result is correct. If it is not correct, regenerate a new captcha and redisplay View 0 with the new captcha and an appropriate error message.

Count the incorrect captcha answer as an incorrect login attempt.

If the captcha is correct, perform the REST endpoint authentication as described in **Task 2**.

Refer to Figure 3 for the process flow.

Task 4 (14 marks)

If the authentication fails for the 3rd time viz. call to the authentication REST endpoint, disable the user from further login for 30 minutes; this means that if user `barney` has been disabled, then a user cannot use `barney` to login for 30 minutes. Do this by creating an entry in the Redis database that has a 30 minutes duration viz. entry will be deleted after 30 minutes.

Hint: See
https://docs.spring.io/spring-data/redis/docs/current/api/org/springframework/data/redis/core/ValueOperations.html

Any subsequent attempt by the same user to login during the 30 minutes duration will be directed to View 2; an example is shown in Figure 6

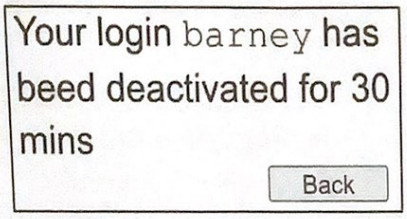

Figure 6 View 2

Write View 2; include the disabled account name eg `barney`. Add a link for the user to return to View 0.

Refer to Figure 3 for the process flow.

Use the class `AuthenticationService.disableUser()` to implement your solution for disable a user account.

Task 5 (10 marks)
Protected resources, viz. resources that required authentication prior to access, are rooted under `/protected`. Accessing any resource under `/protected` with a `GET` will require the request to be authenticated.

If an unauthenticated request attempts to access a protected resource eg `/protected/view1.html`, then the request should be redirected to the View 0.

Figure 7 describes how access to resources rooted under `/protected` are handled.

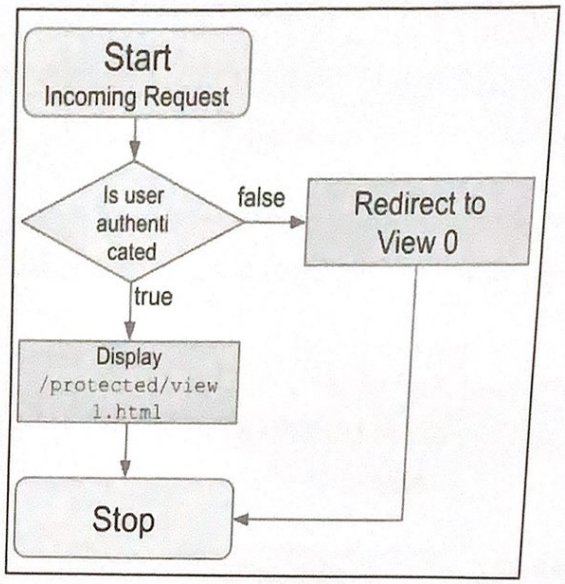

Figure 7 Accessing protected resource

Implement the described behaviour in the following classes

- `ProtectedController.java`
- `AuthenticationService.java`
- `AuthenticationRepository.java`

IMPORTANT: You should protect (viz require authentication) any resource rooted under `/protected` not just `view1.html`. `view1.html` is provided as an example.

Task 6 (5 marks)

The `view1.html` has a Logout button. When the button is pressed, the front controller should logout the user and redisplay View 0.

Once a user has been logged out, the user will not be able to access any protected resource until the user is re-authenticated (relogin).

Task 7 (6 marks)

Deploy your application along with the Redis database to Railway.

To deploy a JDK 19 application to Railway, create an environment variable called `NIXPACKS_JDK_VERSION` in your service and set the value to 19. See https://nixpacks.com/docs/providers/java

Submission

You must submit your assessment by pushing it to your repository to Github.

Only commits on or before **1700 Fri Jun 09 2023** will be accepted. Any commits after **1700 Fri Jun 09 2023** will not be accepted. No other form of submission will be accepted (eg. ZIP file).

Remember to **make your repository public after 1700 Fri Jun 09 2023** so the instructors can review your submission.

After committing your work, post the following information to Slack channel `#02-ssf-submission`

1. Your official name as it is shown in your NRIC
2. Your email
3. Git repository URL
4. Railway URL of the deployed application

It is your responsibility to ensure that all the above submission requirements are met. Your assessment submission will not be accepted if

1. Any of the 4 items mentioned above is missing from `#02-ssf-submission` channel, and/or
2. Your information did not comply with the submission requirements eg. not providing your full name as per your NRIC, and/or
3. Your repository is not publicly accessible after **1700 Fri Jun 09 2023**

You should post the submission information to the Slack channel `#02-ssf-submission` no later than **1710 Fri Jun 09 2023**.

Academic Integrity

This is an open book assessment. You may search the Internet for resources or use reference books during the assessment. The assessment must be your own work. You cannot ask a third party to write any part of this assessment or use AI tools such as ChatGPT to

generate output and submit it as part of your assessment. This will result in an automatic disqualification from the assessment.

The NUS ISS takes a strict view of cheating in any form, deceptive fabrication, plagiarism and violation of intellectual property and copyright laws. Any student who is found to have engaged in such misconduct will be subject to disciplinary action by NUS ISS.

You are to ensure the integrity and working condition of your PC/notebooks (eg. wireless/internet connection, battery, screen, accidents like water spillage) during the assessment. NUS ISS will not accept any of these as a reason for deferring or retaking your assessment.