Web-Based Evaluations

Testing Document

Group Number: 3

Class CPSC 488 Section 1

4/25/2022

Revised:

12/1/2022

Tanuj Rane txr1029@sru.edu

Dalton Stenzel drs1030@sru.edu

Logan Racer lsr1006@sru.edu

Anthony Cinicola [ajc1033@sru.edu](mailto:ajc1033@sru.edu)

Revisors:

J Abbigail Rowe jhr1002@sru.edu

Duncan Lawrence [dal1017@sru.edu](mailto:dal1017@sru.edu) duncanlawrence2000@gmail.com

Contents

[Section 1) Overview of Testing 3](#_Toc120655913)

[Section 2) Unit Testing 4](#_Toc120655914)

[2.1) Company Testing 4](#_Toc120655915)

[*City Test* 4](#_Toc120655916)

[*Company Test* 4](#_Toc120655917)

[*Continent Test* 4](#_Toc120655918)

[*Country Test* 4](#_Toc120655919)

[*Department Test* 4](#_Toc120655920)

[*Location Group Test* 4](#_Toc120655921)

[*Location Test* 4](#_Toc120655922)

[*Province Test* 4](#_Toc120655923)

[*World Test* 5](#_Toc120655924)

[2.2) Controller Testing 5](#_Toc120655925)

[*Add User Controller Test* 5](#_Toc120655926)

[*Archive Controller Test* 5](#_Toc120655927)

[*Data Visualization COntroller Test* 5](#_Toc120655928)

[*Eval Form Controller Test* 5](#_Toc120655929)

[*Evaluator Controller Test* 5](#_Toc120655930)

[*Group Controller Test* 5](#_Toc120655931)

[*Home Page Test* 5](#_Toc120655932)

[*Reset Password Controller Test* 5](#_Toc120655933)

[*Reviewee Controller Test* 6](#_Toc120655934)

[*Self Evaluation Controller Test* 6](#_Toc120655935)

[*User Controller Test* 6](#_Toc120655936)

[2.3) Domain Testing 6](#_Toc120655937)

[*Archive Test* 6](#_Toc120655938)

[*EvalRole Test* 6](#_Toc120655939)

[*EvalTemplates Test* 6](#_Toc120655940)

[*EvaluationLog Test* 6](#_Toc120655941)

[*EvaluatorId Test* 6](#_Toc120655942)

[*Evaluator Test* 6](#_Toc120655943)

[*Group Test* 7](#_Toc120655944)

[*MyUserDetails Test* 7](#_Toc120655945)

[*PasswordResetToken Test* 7](#_Toc120655946)

[*ResetPassword* *Test* 7](#_Toc120655947)

[*Reviewee Test* 7](#_Toc120655948)

[*SelfEvaluation Test* 7](#_Toc120655949)

[*User Test* 7](#_Toc120655950)

[2.4) EvalForm Testing 7](#_Toc120655951)

[*Compute Range Test* 7](#_Toc120655952)

[*Evaluation Test* 7](#_Toc120655953)

[*Question Test* 7](#_Toc120655954)

[2.5) Excel Testing 7](#_Toc120655955)

[*Excel Test* 7](#_Toc120655956)

[2.6) Integration Testing 8](#_Toc120655957)

[*Controller Integration Test* 8](#_Toc120655958)

[*Repository Integration Test* 8](#_Toc120655959)

[*Security Config Integration Test* 8](#_Toc120655960)

[*Service Integration Test* 8](#_Toc120655961)

[2.7) Service Testing 8](#_Toc120655962)

[*Admin Methods Service Test* 8](#_Toc120655963)

[*My User Details Service Test* 8](#_Toc120655964)

[Section 3) Integration Testing 8](#_Toc120655965)

[3.1) Integration Testing Troubles 8](#_Toc120655966)

[Section 4) Black Box Testing Examples 9](#_Toc120655967)

[4.1) Logging In 9](#_Toc120655968)

[4.2) Additions/Deletions/Edits 9](#_Toc120655969)

[4.3) Searching/Sorting 10](#_Toc120655970)

[4.4) Burp Packet Spoofing 10](#_Toc120655971)

[Section 5) White Box Testing Example 10](#_Toc120655972)

[5.1) PageCalc() Method 10](#_Toc120655973)

[5.2) File Upload Handling 10](#_Toc120655974)

# Section 1) Overview of Testing

The document within this file is tasked with covering the procedures used for testing the Web-Based Evaluations System. The testing types included are:

* Unit Testing
* Integration Testing
* Black Box Testing
* White Box Testing

# Section 2) Unit Testing

Any tests that are testing getters and setters are simply tests that set a value and get it to check if the set works. This applies to all tests that start with either get or set. All tests have a setup method annotated with @BeforeAll.

## 2.1) Company Testing

### City Test

* This test only tests getters and setters

### Company Test

* This test only tests getters and setters

### Continent Test

* This test only tests getters and setters

### Country Test

* This test only tests getters and setters

### Department Test

* This test only tests getters and setters

### Location Group Test

* This test only tests getters and setters

### Location Test

* This test only tests getters and setters

### Province Test

* This test only tests getters and setters

### World Test

* This test only tests getters and setters

## 2.2) Controller Testing

### Add User Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Archive Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Data Visualization COntroller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Eval Form Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Evaluator Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Group Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Home Page Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Reset Password Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Reviewee Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### Self Evaluation Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value

### User Controller Test

* This test makes sure that the controller created is not null and then makes a request on the site and expects the site to return a not null value
* tested the creation of group with reviewee and evaluators
* tested group,reviewee and evaluators repository(retrieving data based on certain perimeter like evaluator group number ).

## 2.3) Domain Testing

### Archive Test

* This test tests getters and setters
* nullValuesTest() – This test creates a bunch of null values and then makes sure the null values are null.

### EvalRole Test

* This test only tests getters and setters

### EvalTemplates Test

* This test only tests getters and setters

### EvaluationLog Test

* This test only tests getters and setters

### EvaluatorId Test

* This test only tests getters and setters

### Evaluator Test

* isSyncTest() – ensures that when isSync is set to either true or false that getIsSync returns the appropriate value
* isPreviewTest() - ensures that when isPreview is set to either true or false that getIsPreview returns the appropriate value

### Group Test

* This test only tests getters and setters

### MyUserDetails Test

* isAccountNonExpiredTest(), isAccountNonLockedTest(), isCredentialsNonExpiredTest(), and isEnabledTest() all check to see if the associated class returns true.

### PasswordResetToken Test

* This test only tests getters and setters

### ResetPassword Test

* This test only tests getters and setters

### Reviewee Test

* This test only tests getters and setters

### SelfEvaluation Test

* This test only tests getters and setters

### User Test

* isResetPTest()

## 2.4) EvalForm Testing

### Compute Range Test

* This test only tests getters and setters

### Evaluation Test

* This test only tests getters and setters

### Question Test

* This test only tests getters and setters

## 2.5) Excel Testing

### Excel Test

* loadFileTest() – does not currently work
* checkStringTest() – checks if a cell in the excel sheet is a string
* checkIntTest() - checks if a cell in the excel sheet is an int
* checkLongTest() - checks if a cell in the excel sheet is a long

## 2.6) Integration Testing

### Controller Integration Test

* Not implemented yet

### Repository Integration Test

* Not implemented yet

### Security Config Integration Test

* Not implemented yet

### Service Integration Test

* Not implemented yet

## 2.7) Service Testing

### Admin Methods Service Test

* adminMethoCapTest() – checks if the string parameter is in Title Case
* adminMethoSpaceTest() – checks if the string parameter has a space
* adminMethoCheckTest() – performs the check and update method on a user and returns true for users who pass and false for users who fail (have errors like spaces in email)

### My User Details Service Test

* Not in use

# Section 3) Integration Testing

## 3.1) Integration Testing Troubles

While unit testing was achievable in a rather simple manner, the act of automating a checking system for the relationships between classes proved to be difficult when trying to apply such a technique part way through development. With the way our program is structured and connected provides seemingly endless errors upon trying to call a class that calls multiple others due to the plethora of instances, objects, and services included in the project. An example being when trying to test the AddUserController with a proper link to our user repository/database and the requirements for an instance provided errors with other instances being called from inside the AddUserController. More time could have provided us the necessary knowledge and abilities to overcome this challenge, but for now, we’ve been bested.

The previous group built their program in such a way that integration testing is not possible without reworking a majority of the project. The controller classes must be rewritten to have @PostMapping and @GetMapping methods, and the classes should be annotated @RequestMapping. Additionally, something is broken in the Beans for the project causing a need for a cyclic call in construction of beans when the SecurityConfiguration is called. This causes errors in loading the ApplicationContext and forces the program to terminate. Note that the implementation of the Beans and controllers will affect the code in other classes and will therefore require additional work on the existing code not mentioned to be reworked/refactored to meet the new standards. This will not affect every class but will affect any who utilize the functions the Beans and controllers provide.

The main packages/classes to be focused on are as follows:

* Entire controller package
* Configuration/SecurityConfiguration
* Repository
* Service
* Domain potentially
* Excel potentially

However, through reworking some of the classes (specifically some controllers) we were able to set up the classes to accept the mocked POST requests. Now all that needs to happen is to write more integration tests to cover the full functionality of the program’s methods. Examples that are close to working are available in the integration testing package.

# Section 4) Black Box Testing Examples

## 4.1) Logging In

A user’s credentials and security are a primary concern of the program. Users need to be able to access their own accounts without having troubles such as being redirected to a wrong page. Users of various roles were logged into their accounts both at different and same times in order to see what would happen. It is expected that users would not have troubles accessing their own respected accounts and webpages. Logging into different users at the same time protects from users having their logins merge or swap as long as the test occurs in different browsers or instances of the browser. Having two or more instances of different users logged in on the same browser will merge the cookies, and thus the users will get mixed up, but such behavior is to be expected. Users with different instances of browser cookies, which would be the normal situation as multiple people using one laptop to browse the internet isn’t common, will work as it should.

## 4.2) Additions/Deletions/Edits

Any sort of tinkering with the database entries were able to be tested with ease with the ability to see our data being present to us on the respective webpages. Any changes to users can be seen by viewing the edit buttons or by seeing the default admin users page’s table of users. Feedback is provided on all pages that have actions that provide changes to the database such as notifying if a user was deleted or if the changes attempted failed to be completed.

## 4.3) Searching/Sorting

Users with the “ADMIN” role can search and sort through users on the admin user page via a search bar and a couple of drop-down buttons as well as just normal buttons as well right about the user table. Changing the options of the search should result in the expected changes, and rightly do so.

## 4.4) Burp Packet Spoofing

One of the more security focused black box test cases was using Burp to spoof packets and send modified post requests to the web app. During these tests, we found that the security was not able to withstand the wordlists being run against them (especially on the login page). This helped us see that we needed to update the security we were using. Additionally, we found that spoofing a packet to the download log page caused us to arrive at a webpage that should not be accessed. We were also able to validate that the text boxes in the web pages could withstand data that is not meant to be inputted in those locations and properly catch the errors.

# Section 5) White Box Testing Example

## 5.1) PageCalc() Method

The pageCalc() method located in the AdminMethodsService class is one that could have easily left room for problems if left unsupervised. The method relies on calculations relating to modulus and division which can easily lead to “division by zero” errors. Through analysis and trial and error lead to a seemingly robust defense against any input and regardless of the size of the file or list used. One situation the method takes into consideration is when the user is loading the final page, but there aren’t enough users to fill the list. Another problem taken care of is when the current page is the last page of users and the users per page is set to a low value such as 10; when the user changes the sort/search to a significantly higher value of users per page, the user will be booted to the first page as the previous page would be out of the range of pages available.

## 5.2) File Upload Handling

The ability to upload files arrives on a few occasions, such as on the admin users page, or on the evaluator-admin group or evaluation form page for uploading user, group, or evaluation form information files to the database. Those files must be called by their respective upload methods from each controller. Those methods each are designed in a way to either check whether a file has been uploaded, whether the file contains the right tag/information, and whether some (or any) of the information is even valid. This is done with a series of checks and try/catch exceptions.