Application Testing

Back-End: JUnit Testing

A class named Tests was defined which has the utility functions for all the JUnit tests for particular resources in our product. It adds things directly to the database so that they can be tested. The JUnit tests are run after deploying the server since we have an authentication system which requires cookies to access all the functions. Obviously this is possible by going around the login system and making changes directly but then we are not testing the authorization part of the project. Due to these reasons our JUnit test requires the server to be running when running the JUnit tests.

1. The JUnit tests were on these resources:
    a. Event
        i. GET event
        ii. PUT event
        iii. DELETE event
    b. Events
        i. GET all events
        ii. POST an event
            1. With valid Format
            2. With invalid date format
        iii. DELETE all events
    c. Map
        i. GET map
        ii. PUT map
        iii. DELETE map
    d. Maps
        i. GET all maps
        ii. POST map
        iii. DELETE all maps
    e. EventMap
        i. POST eventmap
        ii. DELETE eventmap
        iii. DELETE all relations for the event (delete all eventmap for an event)
        iv. DELETE all eventmap
    f. Resources
        i. GET resource
        ii. DELETE resource
        iii. GET all resources
        iv. DELETE all resource

        v. GET all materials
        vi. PUT material
        vii. DELETE material
        viii. GET all drawings
        ix. PUT drawing
        x. DELETE drawing
    g. User
        i. GET user
        ii. PUT user
        iii. DELETE user
    h. Users
        i. GET all users
        ii. POST users
        iii. DELETE all users
    i. MapObjects
        i. GET all mapobjects for map
        ii. DELETE all objects on map
        iii. GET map report
        iv. Add new objects to map
        v. DELETE objects from map
        vi. PUT objects to map
        vii. DELETE all objects from map
    j. AuthenticationEndpoint (Login and Logout)
        i. POST login details to login.
        ii. DELETE to logout.
    k. Exception Testing
        i. Testing multiple Exceptions in JUnit tests like NOT_FOUND errors, since those were not explicitly tested.

2. For all the tests the following was done:
    a. Tested with all 3 roles, ADMIN, EDITOR, VISITOR
    b. Test was done for what their functions were according to the REST verb.
    c. Tested the proper HTML status code for each role, to test authorization
    d. Where changes were made in database, the results of those were verified
    e. The tests can run on an empty database, with the active resource list as given by the product owner added, as is defined in the schema.sql.

3. In the LoginLogoutTests, the tests were further done on the kind of response when the user tries to make requests without logging in, which should not be allowed, and should be a bad request, since a login cookie is missing.

4. For all these resources all the methods inside them were tested, the methods for deleteAll were tested but were either commented out or removed to not accidentally

delete the all the database content again and again in the duration of the testing. This is the reason why in the coverage report there is often not a 100% method tested but one less.

5. SessionDao was not tested explicitly since that is not a resource but something that is explicitly only part of a resource, so it is tested when we login and logout. The extra methods in SessionDao are just for completion and if we want to ever implement revoking sessions while sessions are active by some authority like admin. This is the reason in coverage report classes related to Session have the lowest coverage.

6. Shortcoming of the test: Not every exception was thrown during the test, but as many as were possible were. Exceptions like many SQL Exceptions were not thrown due to lack of multiple accesses during a test, which can be done only on a beta test of the application.

Front-End: Authentication

For authentication, the user is initially directed to the login.html page. There were three cases tested for this: user logging in with the correct credentials, with wrong credentials, and leaving the input field blank. As a logical consequence, the user is able to log in successfully with the correct credentials, and otherwise gets a message indicating that the user has given the wrong credentials. Also, any user (either visitor, editor or admin) is able to log out at whichever page they are located.

Front-End: Events

Testing the event page using Selenium IDE helped in finding bugs which were then fixed. The test included creating an event which went well since this part of the project was done in the beginning. Then, edit an event which didn't go well. The test showed that some of the changes were not being saved properly. After that was fixed, the next step was testing if the search bar for events was working properly. There are also plenty of ways to order the event list and that functionality was tested successfully. There were some problems with adding a predefined map to an event but after some minor fixes, it worked fine. The last test done was viewing the list of all maps of one event. There were no bugs, the list of maps was generated properly.

Front-End: Maps

Logging in as admin makes it possible to edit, add and delete maps. The test is separated into 7 tests. The tests conducted for the functionalities of maps are viewing, adding, deleting, editing info, adding items and exporting to PNG. On top of this, we wished to further test the map editing functionalities in Selenium. However, Selenium did not pair well with our Leaflet map and as such had to be left behind. Editing map names (which can be found on the 'List Of Events' page), creating, viewing, editing info, exporting and deleting the maps are processed correctly by pressing the right button. When it comes to editing a map in Selenium, Selenium is unable to target the object when recreating the test. Therefore, scaling, deleting and moving could not be performed properly because Selenium must be able to target the HTML element to interact with them.

In a test for maps, the following tests were successfully conducted using Selenium and this test passed when exported as a JUnit test and run in IntelliJ. In the test, the following is done: logging in, go to the list of maps page, search for the map with the name "Presentation" and open the map. This takes the user to the "mapView.html" page for the map with the name "Presentation". Then, it goes back to the list of maps page where there is a search for the map with the name "does not exist". This map does not exist so an empty table is given.

Front-End: Navigation as a Visitor, Editor and Admin

Admin must have the full potential of navigating to any page they desire. This specifically includes the Users page, Maps page, Resources page and the access to any event or map. Although the navigation test regarding the “back button” worked successfully, some issues with navigating as visitor, editor and admin have been encountered due to Selenium issues (see below).

Front-End: Resources

To test the functionality related to resources, a test was carried out using Selenium and exported as a JUnit Test. Adding, editing and deleting a resource could not be tested with Selenium. If a resource is deleted during the test, it cannot be found when the test is run later. Adding and editing a resource cannot be tested because we are required to upload an image which is not possible to test with Selenium. Therefore, what is done in the test is logging in, go to the resources page, open then close the image of two resources, search for the resource with the name "skittles" and open its image, clear the input box and search for a resource with the name "does not exist" which is not in the database so an empty table is given and clear the input box and search for a resource with the name "d". All resources that contain the letter "d" in the name can be found in the table.

Front-End: Users

In order to test the users page, we used the admin clearance level as both editor and visitor do not have the privilege to change any information on the page, as they are only allowed to get the list of the names of the users and their respective emails. Using the admin privileges, the tests could successfully create, edit, view, search for and sort the users in both ascending and descending orders of the name of the user. In testing the users page, the only thing of interest to be noted is that sorting and searching for specific users only works once the users have been loaded and won’t retroactively apply. Apart from this, no specialties or bugs were encountered. All the functionalities of the users page can be performed successfully. Unfortunately, using

Selenium, we were not able to test the deletion of the user as Selenium requires a target button. This button no longer exists once the user in question has been deleted, therefore the deletion test is not repeatable.

Issues with Selenium

1. The ID of the HTML element changes when a new Event, Map, Resource is added to the map. The way selenium works is by finding the ID of the object on the HTML page and clicking on it. So when a new resource is added, or deleted the ID of the object being modified is different every time, and so Selenium can not find it. This causes the Selenium test to pause and eventually fail. For these reasons all such tests were one one-off way and do not have Java script like many others.

2. Unfortunately, Selenium cannot detect alert popup boxes that are necessary for error detection. Because of this failure, the test eventually does not continue, hence the JUnit test fails consequently. For these reasons all such tests were one one-off way and do not have Java script like many others.

3. Also, Selenium does not work with the leaflet map, so we were not able to test the map functionality using Selenium.