**Documentation For Automated Testing Using Selenium of Ushahidi App.**

1. **Objective:**

The objective of this project is to create, maintain and execute automated tests for Ushahidi Web application deployed on <http://pc-crowdmap-dev.systers.org>. It aims to create reusable and automated scripts for regression testing.

**2. Setting Up of Java based Test Framework using Selenium**

**2.1 Tools and Technologies:**

* **Language : Java:** Install latest JDK and JRE version of Java
* **IDE: Eclipse:** Install Eclipse Lunar/Kepler**.** [Follow this](https://www.eclipse.org/downloads/)
* **Build Automation Tool: Maven:** [Follow this](http://www.mkyong.com/maven/how-to-install-maven-in-windows/)
* **Framework: Selenium (Junit Test Framework)**
* **Web Browser: Chrome, Firefox, IE**
* **Version Control: Git**

**3. Building Ushahidi App from Source Steps:**

- Install WAMP.

Blocking: msvcr100.dll is missing wamp error while installing

Overcome By: Installing vcredist\_x64.exe from https://www.microsoft.com/en-us/download/confirmation.aspx?id=30679

- Follow https://github.com/systers/crowdmap

- go to localhost/<Ushahidi-project> under c:/wamp/www/<Ushahidi-project> It will prompt for installation - either basic or advanced

- I chose basic as it was simple and quick. Keep the Database name, username, password etc in hand, and complete the 4 step installation process.

**4. Getting setup for automated development**

* **Install Maven Plugin In Eclipse Steps:**

Eclipse -> Help -> Install New Software -> Add

A new dialog box will appear.

Name: m2e- maven

Location: <http://download.eclipse.org/technology/m2e/releases>

Click on ok.

Check the box for Maven Integration for Eclipse.

Click on Next -> Finish.

* **Make a new Maven project**
* **In pom.xml, add the following dependencies.**

<dependencies>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>2.46.0</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-firefox-driver</artifactId>

<version>2.46.0</version>

</dependency>

</dependencies>

**5. A check if Selenium is up and working:**

import org.openqa.selenium.By;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.htmlunit.HtmlUnitDriver;  
public class Example {  
 public static void main(String[] args) {  
 // Create a new instance of the html unit driver  
 // Notice that the remainder of the code relies on the interface,   
 // not the implementation.  
 WebDriver driver = new HtmlUnitDriver();  
  
 // And now use this to visit Google  
 driver.get("http://www.google.com");  
  
 // Find the text input element by its name  
 WebElement element = driver.findElement(By.name("q"));  
  
 // Enter something to search for  
 element.sendKeys("Cheese!");  
  
 // Now submit the form. WebDriver will find the form for us from the element  
 element.submit();  
  
 // Check the title of the page  
 System.out.println("Page title is: " + driver.getTitle());  
  
 driver.quit();  
 }

}

**If this runs successfully, you will see output as -**

**"Page title is: Cheese! - Google Search"**

**6. Screens of Ushahidi to be tested:**

There are 5 screens which needs exclusive dedicated testing as follows:

- Login Screen

-. Main page of Website

- Upload Report data

- Configuring Map settings

- SMS server testing

**7. Testing Different Pages:** This section contains the testing of the most important core features of Ushahidi i.e. Login Page, Opportunities Page, Contact Us Page, FAQ Page, Reports page.

**8 Login Page:**



**8.1 Testcases:**

1. The left hand box which has Login With is expanded by default. Clicking on it contracts the email and password pane. [UI]

2. The right hand box which has Create Page is contracted by default. Clicking on it expands the Create an Account pane. [UI]

3. Valid Username and Valid Password should be able to pass

4. Valid username and invalid password should fail with message "Please check that you entered the correct email and password."

5. Invalid username and valid password should fail with message "Please check that you entered the correct email and password."

6. Click Login with only password and without entering email should fail with message "The public profile URL field is required."

7. Click Login without password and with entering only email should fail with message " The password field is required."

8. Click Login without email and without password should display both the above messages.

9. Check if the checkbox "Stay logged in on this computer" works if it is checked.

10. Check if the checkbox "Stay logged in on this computer" doesn''t keep the user logged in if it is not checked.

11. On clicking Forgot Password, "Registered Email" field should pop-up [UI]

12 Writing a wrong/blank email in "Registered email" in "Forgot Password" should fail with the message " Sorry, we don't have your email address".

13. With the correct email address, instructions are sent on registered email to reset password

14. (ToCheck) Is there any upper bound on the number of unsuccessful attempts to login ?

15. Verify if the password can be copy-pasted or not.

16. Check that the password types is encrypted.

17. Verify that the back button doesn't logout the user.

18. Verify that back button doesn't log u back in if the account was previously logged in.

19. User can navigate using TAB key through various fields and click "ENTER" to login.

20. (To check?) Verify simultaneous login to application on different browsers.

21. In Create an Account, any not-registered-previously email should be able to work with Name field not blank and identical passwords in two fields namely password and verify password.

22. Creating an account with Name field left blank should fail with the error message "The name field is required."

23. Creating an account with already registered email should fail with the message "Sorry, a user account already exists for this email address."

24. Creating an account with Password and verify password nonidentical should fail with the message "Please enter the same password in the two password fields."

25. Any combination of 22, 23, 24 while creating a new account should result in multiple respective errors.

This should be tested across various platforms and I will think of some strategy to run the same testcase on multiple platforms. Currently I am just using the ChromeDriver to test it only on Chrome.

**8.2 Documentation of Login Page code:**

**Github:** [**https://github.com/systers/automated-testing/tree/develop/UshahidiTests**](https://github.com/systers/automated-testing/tree/develop/UshahidiTests)

I am going to explain how I started off with creating testcases, my page object model, how my code is neatly isolated to incorporate any future changes and a demo of the functionality of login page testing.

1. Analyzing the login page in detail and listing down all possible test cases: [here](http://keshaassysters.blogspot.in/2015/06/login-page-testing-cases.html)

2. Building the framework, I divided the testing of the entire login page into 4 major sections:

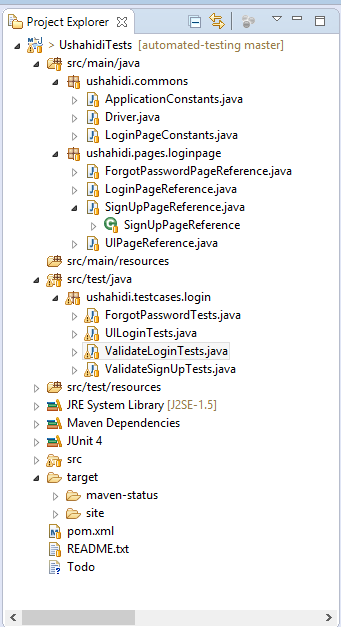
- UI testing

- Login feature testing

- SignUp feature testing

- Forgot password testing

3. The Package structure of my project currently looks like:



**Justifying the structure,**

At a higher level. there are two main folder structures:

A1. **src/main/java** - which contains all the java code related to main

A2. **src/test/java** - which contains all the java code related to testing

A1, Inside src/main/java, there are two packages:

1. **ushahidi.commons** - which contains all the common code related to the entire AUT

-- **ApplicationConstants:** This contains all constants common for the application. Eg: URL, links, extensions etc.

-- **Driver.java:** This contains a common webdriver instance that would be used throughout all tests. It currently has chrome test. Say, we want to test everything on IE now. Instead of going to all tests and changing the webdriver element, we can just change it in this class and it would be reflected in all test classes. No need to change driver element for different web browser testing in all 100 tests.

-- **LoginPageConstants.java:** This contains all the message displayed and the strings to checked for login page. In future, if the error message changes, no need to change that message in all the assert() for all the test cases that throw the error. This makes the code flexible.

2. **ushahidi,pages** - This folder will have all the code for referencing different pages

- **loginpage** - this package will have all references to the WebElements used in the test cases. So, for example, if the web app is changed so that the id of the email field is changed from "emailid" to "userid", there is no need to change it across all testcases where it is referenced. This makes the code flexible.

Instead of changing this line in all the testcases across the test-suit where it is used:

WebElement email\_element = driver.findElement(By.id("emailid"));

to

WebElement email\_element = driver.findElement(By.id("userid"));

In my code approach, the tests have webelements referenced by

SignUpPageReference pageref;

WebElement email\_element = driver.findElement(pageref.getEmail());

There will be no change in code of testsuite.

There will be only one time code change in SignUpPageReference:

private By email = By.id("emailid");

private By email = By.id("userid");

All the references are isolated into 4 classes namely

-- ForgotPasswordPageReference.java

-- LoginPageReference.java

-- SignUpPageReference.java

-- UIPageReference.java

A2. Inside **src/test/java** folder, there are packages for testcases. The first package done for this milestone is ushahidi.testcases.login which contains all code for testing. For login page there are 4 JUnit Test classes.

-- **ForgotPasswordTests.java** : contains 2 testcases

-- **UILoginTests.java** : contains 6 testcases

-- **ValidateLoginTests.java** : contains 6 testcases

-- **ValidateSignUpTests.java** : contains 18 testcases

So, in all, the login page testing contains 32 test cases.

**Link to screencasts demo:**

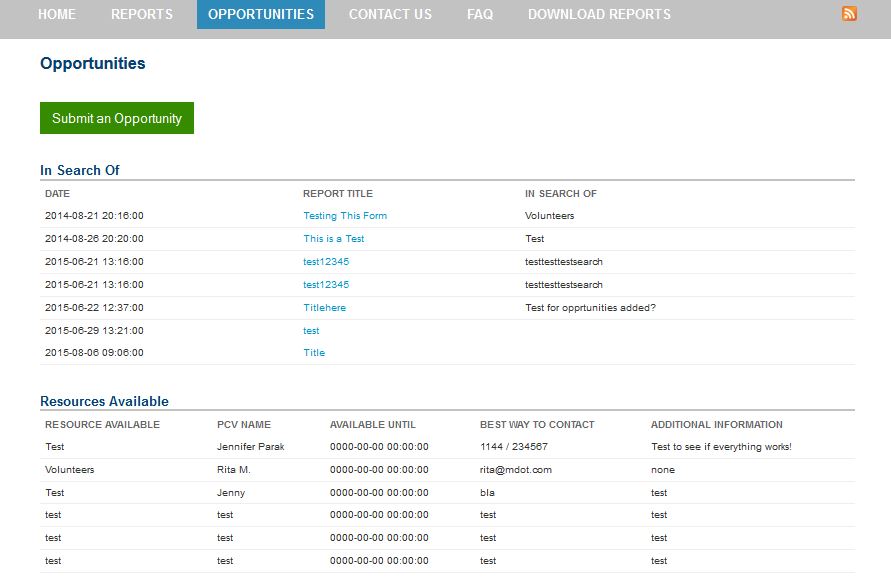
LoginTesting: <http://screencast.com/t/jPyf2Ms36L79>

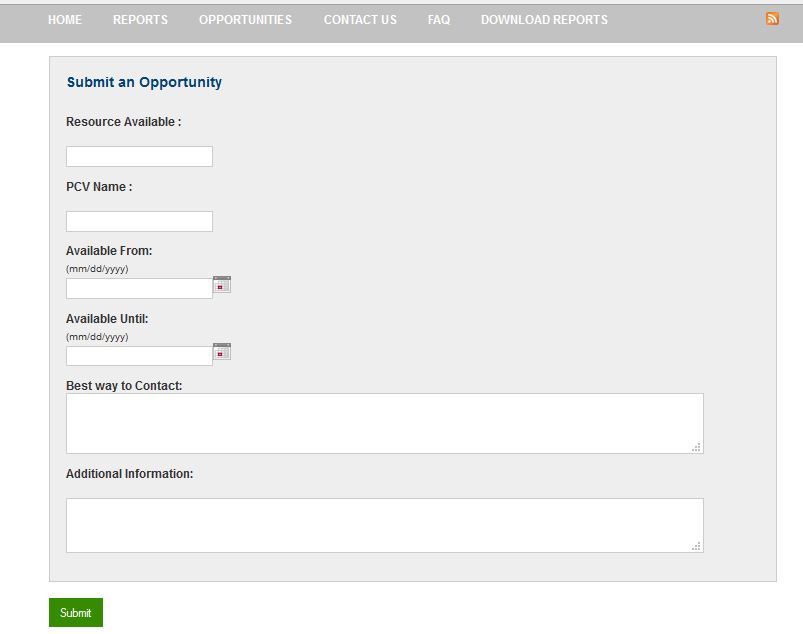
SignupTesting: <http://screencast.com/t/de7k5rjAL>

UI Testing: <http://screencast.com/t/lc0lBRzD6dEd>

Forgot Password Testing: <http://screencast.com/t/53MK1uAcs>

**9. Opportunities Page Testing**





**9.1 Testcases:**

**Pre-condition: @Before**

1. Need to login the web-app.

2. Click the opportunities menu option,

3. Click the submit opportunities button.

**Test-cases: @Test**

1. Submitting an opportunity leaving 'Resource Available' field blank should throw an error - opportunities.resource\_available.required

2. Submitting an opportunity leaving 'PCV Name ' field blank should throw an error -opportunities.pcv\_name.required

3. Submitting an opportunity leaving 'Available From' field blank should throw an error xopportunities.available\_from.required

4. Submitting an opportunity leaving 'Available Until' field blank should throw an error -opportunities.available\_until.required

5. Submitting an opportunity leaving 'Best way to Contact' field blank should throw an error opportunities.contact.required

6. Submitting an opportunity leaving 'Additional Information' field blank should throw an error -opportunities.add\_info.required

7. Submitting an opportunity leaving 'Available Until' field not in the format of mm/dd/yyyy should throw an error - opportunities.available\_until.date\_mmddyyyy

8. Submitting an opportunity leaving 'Available From' field not in the format of mm/dd/yyyy should throw an error - opportunities.available\_from.date\_mmddyyyy

9. Submitting an opportunity with all fields correct should give a success message.

10. On clicking the "Return to Opportunities" should go to opportunities page

11. On clicking the "Submit An Opportunity" should go to the submit opportunity page.

12. Newly added successful opportunity should be visible on the opportunities page. [DB check that the insert of new opportunity happens successfully and is reflected in the list of Resources Available]

**Post-condition: @After**

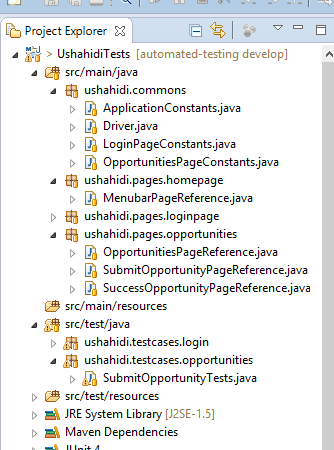
1. teardown - close the driver.

**9.2 Documentation:**

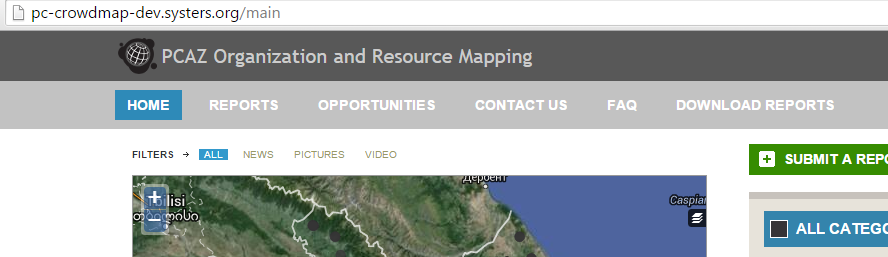
**Github:** [**https://github.com/systers/automated-testing/commit/b913880c29119abf98df164d639d760ab4c181b7**](https://github.com/systers/automated-testing/commit/b913880c29119abf98df164d639d760ab4c181b7)

Here, I will explain how I completed the testing of Opportunities tab of Ushahidi.

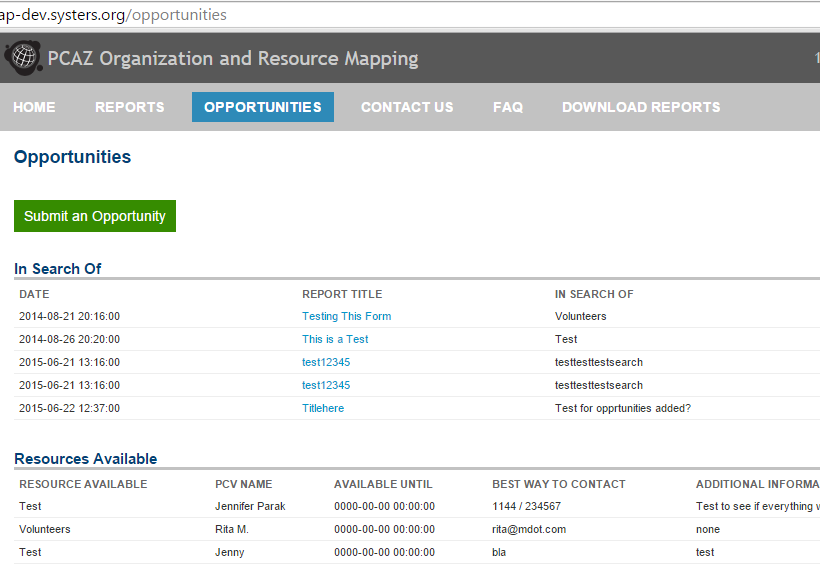
1. The testcases for testing Opportunites are listed at: [keshaassysters.blogspot.in/2015/06/test-cases-for-opportunities-page.html](http://keshaassysters.blogspot.in/2015/06/test-cases-for-opportunities-page.html)
2. Some new packages were added, some class were added in the existing package and some existing classes were changed to incorporate testcases for the opportunities functionality. The new project structure looks as shown below:



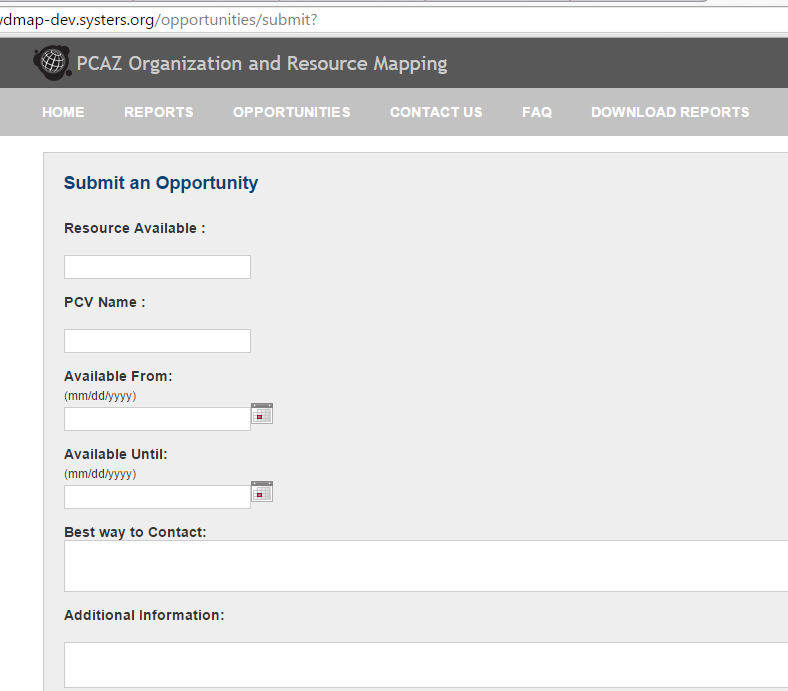
1. **src/main/java:**
   1. **ushahidi.commons:**
      1. **Application Constants.java:** In this, added the extensions in URL for opportunities page and submit opportunities page.
      2. **OpportunitiesPageConstants.java:** Added the constant string messages which were used to verify titles, error logs and success messages in testcases file i.e. SubmitOpportunityTests.java
   2. **ushahidi.pages.homepage:**
      1. **MenubarPageRefrence.java:** This class contains all page refrences that lie on the homepage and have menubar options which lead to different main screens. Signing in to Ushahidi, you will land up in this page.



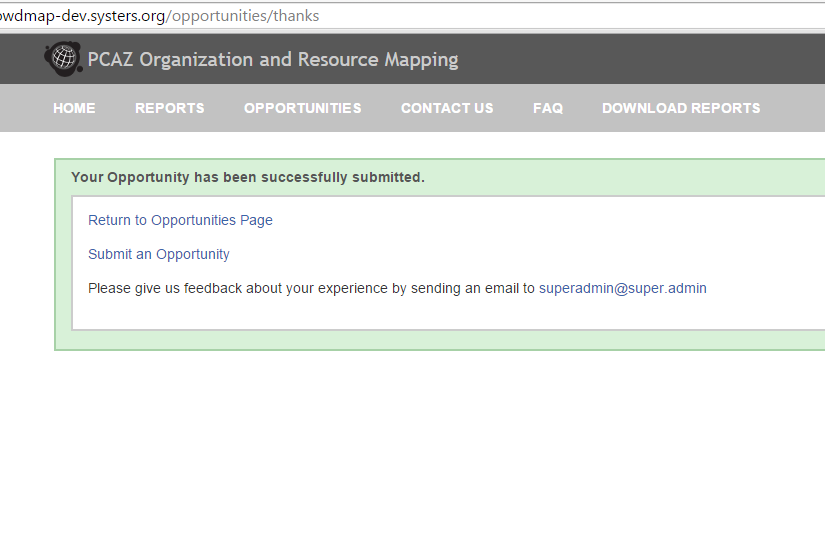
* 1. **ushahidi.pages.opportunities:** Each of them contains page reference for the web pages shown as screenshots:
     1. **OpportunitiesPageReference.java:** Clicking on the opportunities tab of menubar, you will end up here.



* + 1. **SubmitOpportunitiesPageReference.java:** Clicking ‘Submit An Opportunity’ button from the above page, you will land up here.



* + 1. **SuccessOpportunityPageReference.java:**Filling appropriate details in the above page and clicking on Submit button, you will land up in this page.

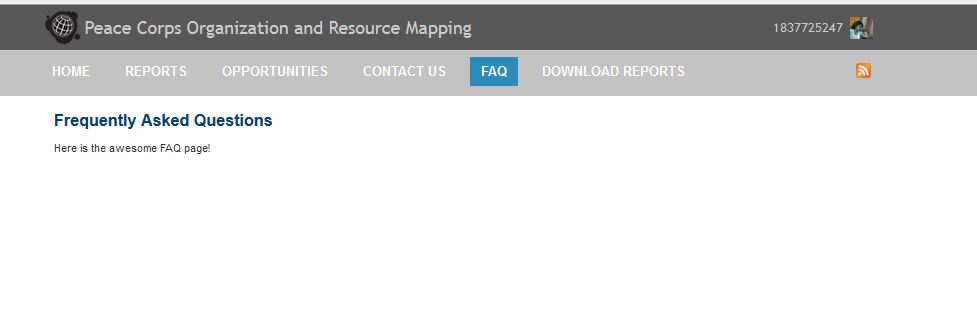


1. **src/test/java:**
   1. **ushahidi.testcases.opportunites:**
      1. **SubmitOpportunityTests.java:** This is the class which contains 12 testcases related to testing opportunities feature. It contains implementation of testcases mentioned above.

**Screencast Demo:** [**http://screencast.com/t/2TFlpxJ6**](http://screencast.com/t/2TFlpxJ6)

**10. Testing of FAQ Page:**

FAQ page is quite simple, and so there is no major testing involved in it in the current page. Raised an issue on github to request the developers to update the FAQ with appropriate and valuable information.(Issue 28: <https://github.com/systers/crowdmap/issues/28> )



**Pre-condition: @before**

1. Need to login the web-app.

2. Click the faq menu option.

**Test-cases: @tests**

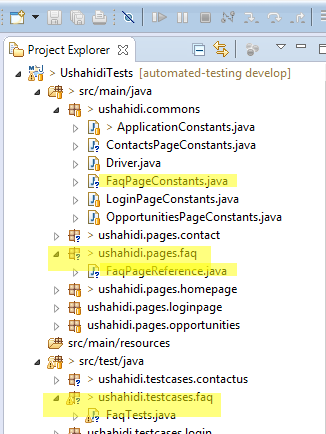
1. Check if clicking on FAQ on menubar takes us to Frequently Asked Question Page by checking title of the page.

2. Check if there are some contents on the FAQ page.

**Post-condition: @after**

1. teardown - close the driver.

**Documentation:**

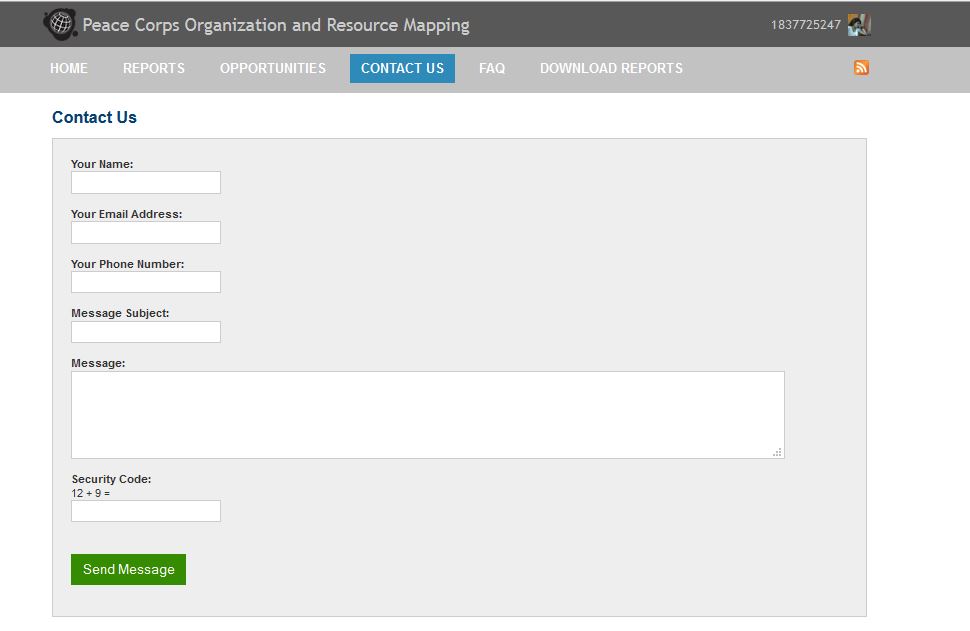


Some new packages and classes were added as highlighted in the screenshot.

**Github:** [**https://github.com/systers/automated-testing/commit/8000b4a87a3bd44d6d7f62876f7bd1a40a1e4f58**](https://github.com/systers/automated-testing/commit/8000b4a87a3bd44d6d7f62876f7bd1a40a1e4f58)

**Screencast:**[**http://screencast.com/t/ouaNXU8ePr**](http://www.google.com/url?q=http%3A%2F%2Fscreencast.com%2Ft%2FouaNXU8ePr&sa=D&sntz=1&usg=AFQjCNESmsgNu2oKNsEc2UgvOBAlq-6ifA)

**11. Testing of Contact Us Page:**



**11.1 Testcases:**

**@PreCondition:**

1. Login to Ushahidi application
2. Click on Contact-Us Tab on the menu bar.

**@Test:**

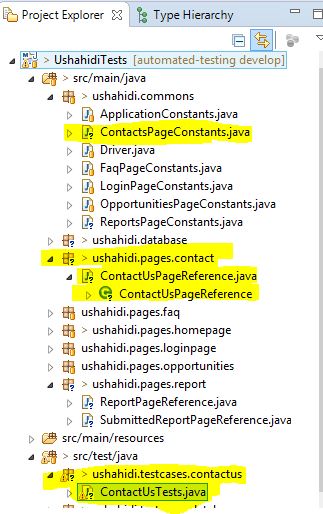
##### Submit a form with all details. This should work successfully, with no errors, but while testing on the version deployed at <http://pc-crowdmap-dev.systers.org> it shows an error. Raised [issue on github](https://github.com/systers/crowdmap/issues/24).

1. Submit a form with no name. This should fail with the message - "The name field is required."
2. Submit a form with no email. This should fail with the message - "The Email field is required."
3. Submit a form with no subject. This should fail with the message - "The subject field is required."
4. Submit a form with no message. This should fail with the message - "The message field is required."
5. Submit a form with no security answer. This should fail with the message - "Please enter the security code."
6. Submitting a form with no phone. This should work fine with the message - “Your Message Has Been Sent!”
7. Submitting a form with all details should work successfully with the message - “Your Message Has Been Sent!” (Updated after solving the issue on github)

**Documentation:**

As in the previous model, we have the following related files:

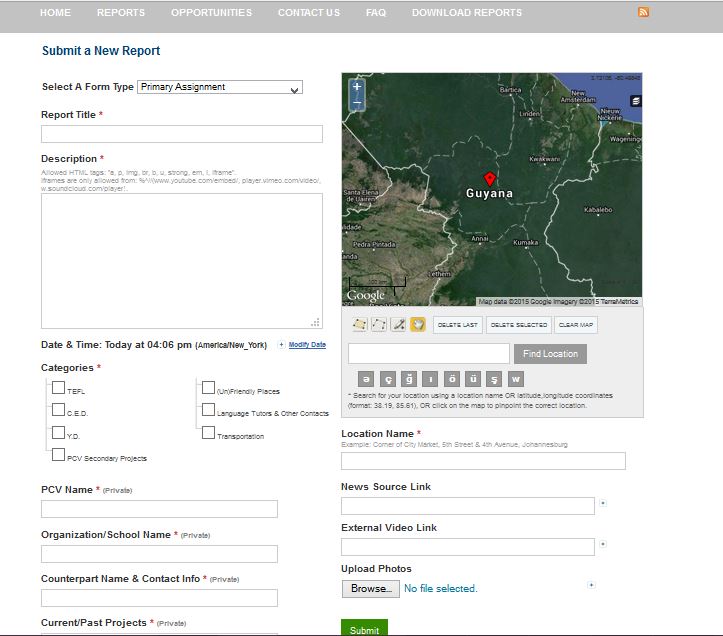
1. ContactsPageConstants.java in ushahidi.commons package in src/main/java
2. ContactUsPageReference.java in ushahidi.pages.contact package in src/main/java
3. ContactUsTests.java in ushahidi.testcases.contactus in src/test/java

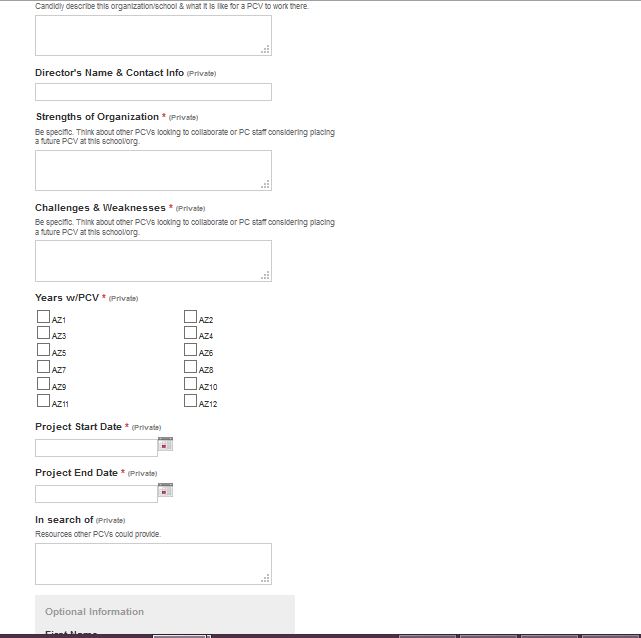


**Screencast:** <http://screencast.com/t/fkhs9pMYJ>

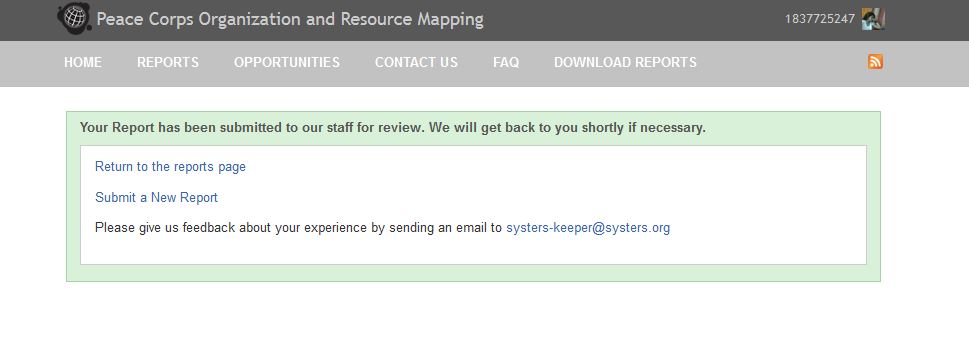
**Updated screencast (for Test8 after solving** [**issue 26 on github**](https://github.com/systers/crowdmap/issues/26)**):** [**http://screencast.com/t/KsTPG53vV69**](http://screencast.com/t/KsTPG53vV69)

**12. Testing of Reports:**





After filling the report with appropriate information and clicking on the Submit button, you will be transferred to the below page:



**@PreCondition:**

1. Login to the Ushahidi web application.
2. Click on Submit a Report on the right hand side/footbar
3. It redirects to the page for submitting a new report.

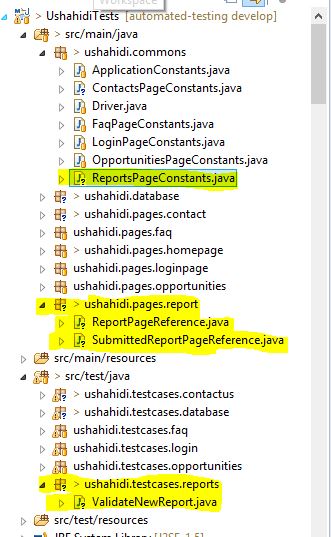
**@Tests:**

1. Submitting the report without Report Title should fail with the message - “The "Report Title" field is required.”
2. Submitting the report without Description should fail with the message - "The "Description" field is required.”
3. Submitting the report without Categories should fail with the message - "The "Categories" field is required."
4. Submitting the report without PCV Name should fail with the message - "The "PCV Name" field is required."
5. Submitting the report without Location name should fail with the message - "The "Location Name" field is required."
6. Submitting the report without Organization/School Name should fail with the message - "The "Organization/School Name" field is required."
7. Submitting the report without CounterPart Name and Contact information should fail with the message - "The "Counterpart Name & Contact Info" field is required."
8. Submitting the report without Current/Past Projects should fail with the message - "The "Current/Past Projects" field is required."
9. Submitting the report without Work environment should fail with the message - "The "Work Environment" field is required."
10. Submitting the report without Strengths of organization should fail with the message - "The "Strengths of Organization" field is required."
11. Submitting the report without Weakness of organization should fail with the message - "The "Challenges & Weaknesses" field is required."
12. Submitting the report without Years with PCV should fail with the message - "The "Years w/PCV" field is required."
13. Submitting the report without Project Start date with PCV should fail with the message - "The "Project Start Date" field is required."
14. Submitting the report without Project End Date with PCV should fail with the message - "The "Project End Date" field is required."
15. Submitting the report without director name with PCV should work perfectly fine.
16. Submitting the report without first name with PCV should work perfectly fine.
17. Submitting the report without last name with PCV should work perfectly fine.
18. Submitting the report without email with PCV should work perfectly fine.
19. Submitting the report without news source link with PCV should work perfectly fine.
20. Submitting the report without external video link with PCV should work perfectly fine.
21. On the page that appears after submitting a successful report, it says - “Your Report has been submitted to our staff for review. We will get back to you shortly if necessary.”
22. On the page that appears after submitting a successful report, clicking on the ‘Return to the reports page’ hyperlink takes you to <http://pc-crowdmap-dev.systers.org/reports> URL.
23. On the page that appears after submitting a successful report, clicking on the ‘Submit New Report’ hyperlink takes you to <http://pc-crowdmap-dev.systers.org/reports/submit> URL.

**Documentation:**

As in the model in previous testcases, we have the following related files for the testing of Create a New Report page:

1. ReportsPageConstants.java in ushahidi.commons package in src/main/java
2. ReportPageReference.java in ushahidi.pages.report package in src/main/java
3. SubmittedReportPageReference.java in ushahidi.pages.report package in src/main/java
4. ValidateNewReports.java in ushahidi.testcases.reports in src/test/java



**Screencast:**

**13. Database Testing:**

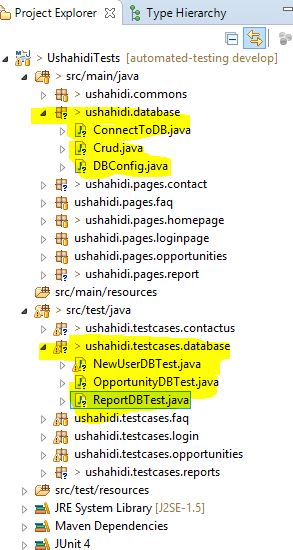
Usually, as seen in the previous tests, Selenium is used to do browser-testing. To use Selenium to do database testing, it is done with the help of JDBC (Java Database Connectivity). It provides an API to do database level of queries.

There are 3 broad level steps in database testing:

1. Make connection to the database
2. Send queries to the database
3. Process the result

Looking at each of them,

1. Make connection to the database



1. Send queries to the database
2. Process the result

**14. Database Testing of Users:**

When a new user signs up, the entities associated with a particular user are name, email and password. Email is unique for each account. According to CRUD [Create, Read, Update and Delete] way of testing for database, we will first create a new user with Name N, Email E and Password P.

Here, the sequence in which test are run is important. Only after the new account is created, the read queries will execute successfully. Only after it is read, can the entry be deleted, otherwise deleting it and reading it afterwards would lead to failure of the test. Hence to maintain the sequence in which test are run, I added the annotation ”@FixMethodOrder(MethodSorters.NAME\_ASCENDING)” which makes sure, TestA runs before TestB, which in turn runs before TestC and so on and so forth.

The tests for testing database for new user are:

* Create new account for any user
* Check database if name of new user exists
* Check database if email of new user exists
* Check database if name and email of new user are associated correctly
* Delete the new user

**15. Database Testing of Opportunities:**

Using the same rationale behind the importance of running the tests in particular order, the annotation ”@FixMethodOrder(MethodSorters.NAME\_ASCENDING)” is used in database testing of opportunities.

The tests for testing database for opportunities are:

* Create a new opportunity
* Check database if the ‘Resource Available’ exists
* Check database if the ‘PCV Name’ exists
* Check database if the ‘Available From’ exists
* Check database if the ‘Available Until’ exists
* Check database if the ‘Contact’ exists
* Check database if the ‘Additional Information’ exists
* Check database if the entire opportunity with Resource Available, PCV Name, Available From, Available Until, Contact, Additional Information exists
* Delete the entry of new added Report

**16. Database Testing of Reports:**

Using the same rationale behind the importance of running the tests in particular order, the annotation ”@FixMethodOrder(MethodSorters.NAME\_ASCENDING)” is used in database testing of reports.

The tests for testing database for reports are:

* Submit a new report
* Check database if the title exists
* Check database if the description exists
* Check database if the categories exists
* Check database if the PCV Name exists
* Check database if the organization name exists
* Check database if the contact exists
* Check database if the past projects exists
* Check database if the work environment exists
* Check database if the director information exists
* Check database if the strength exists
* Check database if the weakness exists
* Check database if the years exists
* Check database if the start Date exists
* Check database if the end Date exists
* Check database if the in search of field exists
* Check database if the first name exists
* Check database if the last name exists
* Check database if the email exists
* Check database if the location name exists
* Check database if the news source exists
* Check database if the video source exists
* Delete the entry of new report

**17. Admin Testing**

<Week 13 work>

**18. Continuous Integration:**

Continuous Integration a must if someone is working in collaborative environment. Most of the projects on github have configured a CI for it, in general open-source projects use Jenkins, travis, Hudson etc which is an open-source CI tool.

Continuous Integration basically means that you had some code initially with all tests running and deploying successfully. Now, let's say you plan to add a new feature so you code up that and write tests for these modules. But how to check that when you integrate this new feature into the already existing code-base, you don't break up the system as a whole, which means that your new functionality should be compatible with the already existing codebase and also it should not affect break the functionalities.

Now, coming to how to check if the new code pushed or which you are about to merge does not break the existing code ? So, we write scripts, which basically say that these commands should run successfully without any error to have the project in an OK state.

So, we write the test suite and include it in the code-base and all of the tests should run successfully.

1. Download the jenkins.war file.
2. Place it in appropriate location

**18. Issues Raised on Github:**

1. [**https://github.com/systers/crowdmap/issues/23**](https://github.com/systers/crowdmap/issues/23)
2. [**https://github.com/systers/crowdmap/issues/24**](https://github.com/systers/crowdmap/issues/24)
3. [**https://github.com/systers/crowdmap/issues/25**](https://github.com/systers/crowdmap/issues/25)
4. [**https://github.com/systers/crowdmap/issues/26**](https://github.com/systers/crowdmap/issues/26)
5. [**https://github.com/systers/crowdmap/issues/28**](https://github.com/systers/crowdmap/issues/28)

**19. Future Work:**

1. DRY - Dont Repeat Yourself: Refactor the code to remove any duplicate code.
2. Use Logger Factory in-place of System.out.println (logback/slf4j)
3. One of the performance issue I see is it takes a lot of time to run all those tests individually. After one tests completes, then only the second tests starts. This is important where Database tests are run and a specific sequence of create, read, update and delete tests are performed. But, for other non-database tests, all tests are disjoint and independent of each other. So, the time to run all of them can be minimized by spawning a thread and assigning a test to each individual thread.
4. Perform performance tests, load balance tests, cross platform etc
5. Extend the config to Test with different platforms and different browser.
6. Support with reading the test data from a database/xml file/excel file/csv file
7. Making reports with sure-fire plugin. Tried that in past, but had some issues. Solve the report generation part.
8. Continuous integration - Did set up Jenkins and this project, but will have to check how to move it from localhost
9. How to bring passwords securely? I am not able to update Github and maintaining everything on local coz the passwords are written right up front. Can't really push that code. Research on how to bring passwords securely in Selenium. (maybe an XML/from Database)
10. Do admin testing. Only admin login have the rights to accept a report, so check that functionality.
11. Selenium has a plugin to take screenshot of failure test cases. Install that and use it. On every failure, the screenshots go to an error- package.
12. Research if the project can be migrated to Spring? It has feature of dependency injection and inversion of control. So DB config, constant storing and other config can be maintained well with context.xml. So research if the efforts are worth moving it to Spring!