* **How do you test for functional requirements?**
* Gathering the requirements from the user stories.
* Understanding the requirements.
* Identify the test data based on requirements.
* Divide the testcases into smoke and Regression testcases to identify the bugs
* after deployments.
* Execute the testcases.
* Compare the actual and expected output.

**In this assignment we have covered below testcases.**

TC1: Insert a Single Record of working-class hero consist of Natural Id

(natid), Name, Gender, Birthday, Salary and Tax paid into database

an API. Then verifying the actual and expected results.

TC2: Insert More than one working class hero consist of Natural Id

(natid), Name, Gender, Birthday, Salary and Tax paid into database via an

API. Then verifying the actual and expected results.

TC3: Upload csv file into portal via portal.

TC4: Upload csv file into portal via Api the verify the success message.

TC5: When you perform the Get request it should return the natid, tax relief

amount and name then verify the natid field must be masked from the

5th character onwards with the dollar sign ‘$’.

TC6: Verify the tax relief amount is calculating properly or not.

TC7: Verify the Dispense Now button color is red or not.

TC8: Verify the Dispense Now button text is displaying as expected.

TC9: Verify the when user clicks on Dispense Now button it should navigate

to another page and verify the text “Cash dispensed”.

* **How do you test for non-functional requirements?**

After performing the regression test run testers can able to start performance testing, load testing and compatibility testing.

Increase and decrease the load [gradually](https://www.webperformance.com/load-testing-tools/blog/2009/05/estimating-load-test-ramp-up-times/) as this way we will be able to correlate increasing response time.

* **What are the test cases you’ve covered?**

We have covered all the API and Web Testcases.

**In this assignment we have covered below testcases.**

TC1: Insert a Single Record of working-class hero consist of Natural Id

(natid), Name, Gender, Birthday, Salary and Tax paid into database

an API. Then verifying the actual and expected results.

TC2: Insert More than one working class hero consist of Natural Id

(natid), Name, Gender, Birthday, Salary and Tax paid into database via an

API. Then verifying the actual and expected results.

TC3: Upload csv file into portal via portal.

TC4: Upload csv file into portal via API the verify the success message.

TC5: When you perform the Get request it should return the natid, tax relief

amount and name then verify the natid field must be masked from the

5th character onwards with the dollar sign ‘$’.

TC6: Verify the tax relief amount is calculating properly or not.

TC7: Verify the Dispense Now button color is red or not.

TC8: Verify the Dispense Now button text is displaying as expected.

TC9: Verify the when user clicks on Dispense Now button it should navigate

to another page and verify the text “Cash dispensed”.

* **A short Readme on how to run these tools will definitely put a smile on our face =)?**

* Before executing the testcases change the configuration properties.

Below is the Path of the configuration properties

src/test/java/resources/Config/config.properties

**Description of configuration properties:**

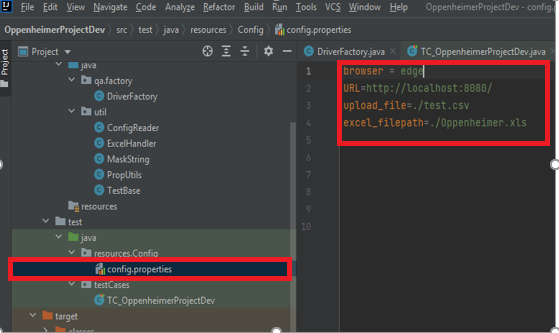
browser – On which browser you need to execute.

URL – URL of the application.

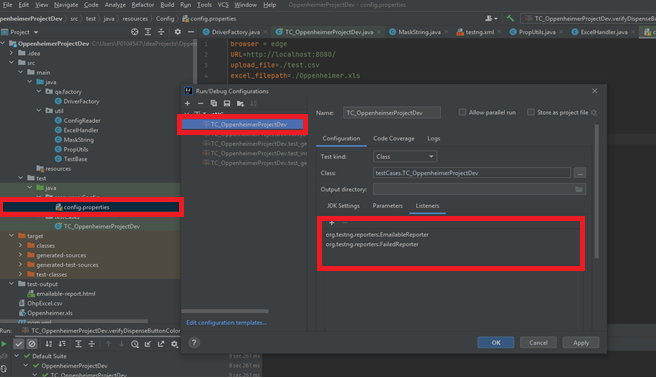
Upload\_file – The path of the excel To upload data into

portal with the extension of csv.

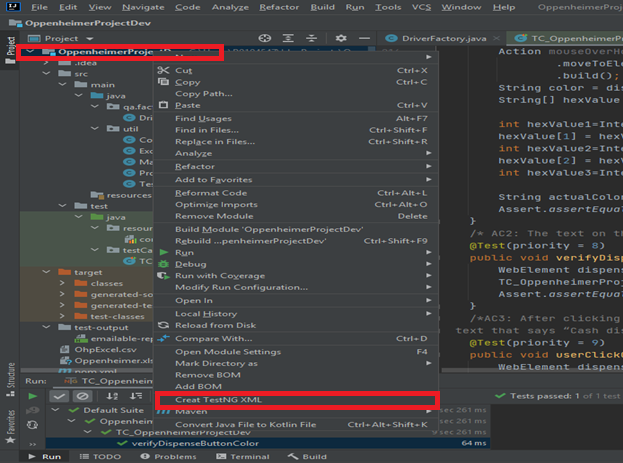
Excel\_filepath – The path of the excel to insert data.



* Add listeners in the Configuration settings to generate Emailable and failed testcase report.



* Right click on Project -> Create TestNG XML -Click On Ok and then refresh the page. It will generate the testng.xml file.



* Right click on testng.xml file and run test cases. After executing testcases it will generate one test-output folder. Open test-output folder and check for emailable report.

**Below is the sample Report**

