**Oppenheimer Project - Test Automation**

**Table of Contents**

|  |  |
| --- | --- |
| **Topic** | **Page** |
| 1. Overall Summary | 2 |
| 1. Key Issues Identified | 2 |
| 1. Non-Functional Testing | 3 |
| 1. Additional Scenarios for Future | 3 |
| 1. Verification of Automation Script | 4 |
| 1. Detailed Description of Framework | 4 |

1. **OVERALL SUMMARY**

* Used the Behavioural Driven Development (BDD) framework for the test automation
* Executed 6 User Stories (with 21 test cases)
* Framed and executed 27 additional test cases

***1.1 Execution summary***

**Acceptance criteria execution summary (for the 6 user stories)**

|  |  |  |
| --- | --- | --- |
| **Total Test Cases** | **Pass** | **Fail** |
| 21 | 18 | 3 |

* **Additional test cases execution summary (new cases identified)**

|  |  |  |
| --- | --- | --- |
| **Total Test Cases** | **Pass** | **Fail** |
| 27 | 25 | 2 |

Summary report:

****

***1.2 Test data covered***

Titanic dataset (available on internet) is used as input for heroes’ names. The input file values for salary, Date of Birth, tax is randomized in Excel in order to cover a wide range of data including boundary values

**Table A**

|  |  |  |  |
| --- | --- | --- | --- |
| Total records | Correct Results | Tax calculated incorrectly | Rounding not applied correctly |
|  |  |  |  |
| 100 | 38 | 8 | 54 |

Data mismatch records are written to 2 separate files by the script (attached below).

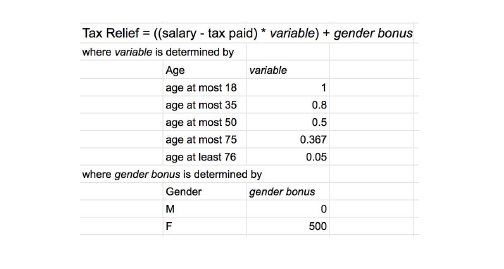
The records in these files show mismatch observed between the response from the application and the script calculated figures.



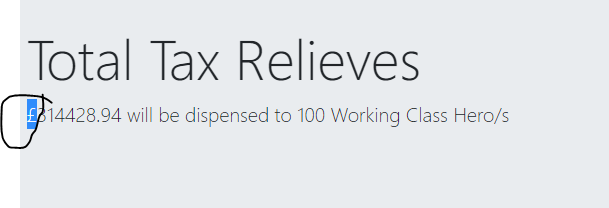
incorrect\_rounding\_logic\_file.txt contains 54 pairs of records (54 from the api response and 54 from the script calculated data)

incorrect\_tax\_calc\_file.txt contains 8 pairs of records (8 from the api response and 8 from the script calculated data)

1. **KEY ISSUES IDENTIFIED**
2. The 8 records have failed at the boundary values where the age of the hero is exactly at the Age slab provided. E.g.: If the age is 18 years and 6 months, the application picks up the variable as ‘1’ from the previous slab. Since the age has crossed 18, it has to pick up the variable as ‘0.8 as per below logic



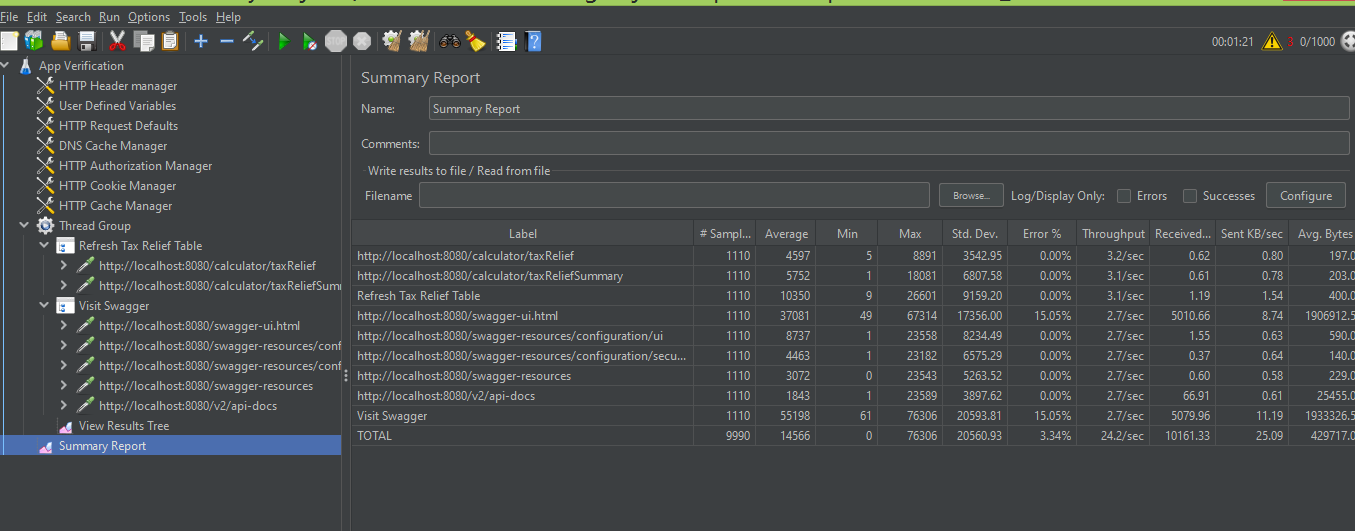
1. The 54 records failed under ‘Rounding not applied correctly’ category (refer to Table A) because the relief amount did not undergo rounding
2. The summary relief amount is displayed in ‘£’ instead of ‘$’ which is incorrect.



1. Currently the system allows duplicate natids which could be a potential gap in the application.
2. **NON-FUNCTIONAL TESTING**

Tested using JMeter. Attaching the response for taxRelief GET Request API with 100 & 1000 users respectively. The average response time is 4 seconds to 55 seconds for a set of 1000 users .





1. **ADDITIONAL SCENARIOS FOR FUTURE**

* Providing null values for mandatory fields like date and throw appropriate error message or skip the record and proceed
* Providing a value of 0 salary or tax. We should either handle it or assert and fail the test case

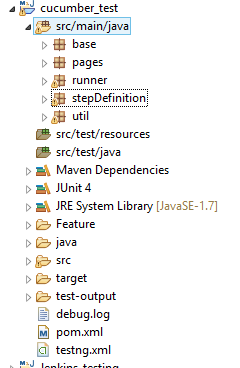
1. **VERIFICATION OF AUTOMATION SCRIPT**

* In order to check if my automation script calculated values are correct, I have performed the same calculation on a spreadsheet. The output from my automation script calculated tax relief (post rounding) value matches with that of the values calculated in spreadsheet using formulas. Based on this, I confirmed that my automation script has calculated values correctly as per the acceptance criteria logic provided in the requirement
* Attaching the spreadsheet calculation for reference (calculation.xls). Please refer to column ‘S’ for the comparison result.



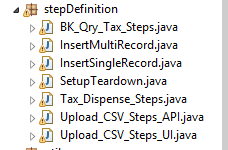
1. **DETAILED DESCRIPTON OF FRAMEWORK**
2. **Framework**

* Behavioural Development approach is followed to test this product in order to map the test scenarios provided in the acceptance criteria
* Scenarios are written in Cucumber feature files
* POM: Page Object Model is followed for the UI testing
* Test Runner: TestNG
* Reporting: Extent Report
* Project Structure



* Packages: There are 5 main user stories have to be tested. In this framework, each user story is mapped to a feature file. Feature files and their respective step definitions (under stepDefinition package) are explained below . Each feature has a corresponding runner class under ‘runner’ package

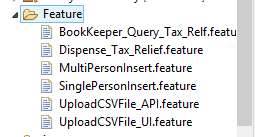
1. base:
2. base.java is a parent class. It has static data members to store values across classes and methods
3. ExtentReportUtil.java: Instantiates extent report objects and attach to html report
4. pages: To store the respective page’s web elements. It also has methods that operate on these web elements
5. Tax\_Dispense\_Page.java has methods that operate on the web elements on the Tax Dispense Page
6. UploadCSV\_API.java : To test uploading the data in the form of csv file using the UI page by clicking the link to upload under “[uploadLargeFileForInsertionToDatabase](http://localhost:8080/swagger-ui.html#/operations/calculator-controller/uploadFileUsingPOST)”
7. UploadCSV\_UI.java : To test uploading the data in the form of csv file using the UI page by clicking the upload file button
8. Runner:
9. BookKeeperTax\_TestRunner.java : Runner class to execute the feature file(BookKeeper\_Query\_Tax\_Relf.feature) and step definition(BK\_Qry\_Tax\_Steps.java) for BookKeeper action scenarios
10. Dispense\_Tax\_Relief\_TestRunner.java: Runner class to execute the feature file (Dispense\_Tax\_Relief.feature) and step definition (Tax\_Dispense\_Steps.java) to test scenarios related to tax dispense
11. MultiPersonInsert\_TestRunner.java: Runner class to execute the feature file MultiPersonInsert.feature and step definitions (InsertMultiRecord.java)
12. NGTestListener.java: Runner class for TestNG Listener
13. SingleInsert\_TestRunner.java: Runner class to execute the feature file SinglePersonInsert.feature and step definition InsertSingleRecord.java
14. UploadCSVFile\_API\_TestRunner.java: Runner class to execute the feature file UploadCSVFile\_API.feature and step definition upload\_CSV\_Steps\_API.java
15. UploadCSVFile\_UI\_TestRunner.java: Runner class to execute the feature file UploadCSVFile\_UI.feature and step definition Upload\_CSV\_Steps\_UI.java
16. StepDefinitions:



* + - 1. These step definitions as mentioned in step c. are executed for respective user stories

1. Util:
   * + 1. CSVRowValues.java : This is a common utility that reads a csv file based on header and converts it to an Array List. This utility is used for the scenario where we have to read data to validate the header and data and also to upload to API
       2. DataComparison.java: This utility compares data between system calculated tax relief(based on formula given in the acceptance criteria) and the response received from GET Tax Relief API. It returns the validations performed in terms of whether rounding has happened, truncation of excess decimal places to 2 decimal places, rounding off the tax relief, setting the relief value to 50.00 in case the calculated tax relief amount is between 0.00 and 50.00
       3. ReadFile.java : Reads a .xls file using Apache POI library and returns data in the form of an Array List
       4. TaxCalculator.java: Reads the input file and calculates tax as per the formula provided in the acceptance criteria. It calculates the age based on date of birth given in the input data, identifies the variable pay , calculates bonus based on gender and finally calculates the tax relief. While calculating, it also truncates the excessive decimal places to 2 , rounds off the tax relief value, sets the relief value to 50.00 in case the calculated tax relief amount is between 0.00 and 50.00

* Feature: This folder has the feature files for the respective user stories as shown below



* Config.properties: This file is a repository that stores the url and file path information
* Report: TestNG html report is stored in the below folder. A sample report is attached below.



* Input: This is the project path that stores input data: C:\Users\Sahana Rangarajan\eclipse-workspace\cucumber\_test\input
* Data Mismatch: Contains 2 files, one for the tax calculation mismatch records and another to store rounding logic mismatch records