Test Plan | The Oppenheimer Project

Author : Balu Paulraj

Version-1.0

QA Owner : Balu Paulraj

Dev Owner: <Developer name>

Introduction

In the year 1969, the City of Carson is a growing city that houses close to millions of working class heroes. To support the growing city population, a bill was passed that:

‘each working class hero is to contribute a fraction of their yearly income towards city building’

This year, as part of the governor’s initiative to pony up surplus cash in the vault, each working class hero is gifted with taxation relief as recognition to their voluntary contribution to city building efforts. To facilitate this, the governor has drafted out the Oppenheimer Project.
This is a software system that has to support 3 features:

- Enable Clerks to populate a list of working class heroes to the system
- Enable Bookkeepers to retrieve the payable taxation relief for each working class hero
- Enable Governor to dispense the money to each working class hero at her discretion

Objective

- To test taxation relief software system (WebApp)

Acceptance Criteria

USER STORIES:

(1) As the Clerk, I should be able to insert a single record of working class hero into database via an API
Acceptance Criteria 1: Single record of a working class hero should consist of Natural Id(natid), Name, Gender, Birthday, Salary and Tax paid

(2) As the Clerk, I should be able to insert more than one working class hero into database via an API
Acceptance Criteria 1: Enhancement of (1), with the ability to insert a list

(3) As the Clerk, I should be able to upload a csv file to a portal so that I can populate the database from a UI
Acceptance Criteria 1: First row of the csv file must be natid, name, gender, salary, birthday, tax
Acceptance Criteria 2: Subsequent rows of csv are the relevant details of each working class hero
Acceptance Criteria 3: A simple button that allows me to upload a file on my pc to the portal

(4) As the Bookkeeper, I should be able to query the amount of tax relief for each person in the database so that I can report the figures to my Bookkeeping Manager
Acceptance Criteria 1: a GET endpoint which returns a list consist of natid, tax relief amount and name
Acceptance Criteria 2: natid field must be masked from the 5th character onwards with dollar sign ‘$’
Acceptance Criteria 3: computation of the tax relief is using the formula as described:
Acceptance Criteria 4: After calculating the tax relief amount, it should be subjected to normal rounding rule to remove any decimal places
Acceptance Criteria 5: If the calculated tax relief amount after subjecting to normal rounding rule is more than 0.00 but less than 50.00, the final tax relief amount should be 50.00

Acceptance Criteria 6: If the calculated tax relief amount before applying the normal rounding rule gives a value with more than 2 decimal places, it should be truncated at the second decimal place and then subjected to normal rounding rule

(5) As the Governor, I should be able to see a button on the screen so that I can dispense tax relief for my working class heroes

Acceptance Criteria 1: The button on the screen must be red-colored
Acceptance Criteria 2: The text on the button must be exactly “Dispense Now”
Acceptance Criteria 3 : After clicking on the button, it should direct me to a page with a text that says “Cash dispensed”

Test Bed Configurations

Platforms Covered:

Windows 2019 server

Windows 2016 Server both standard, r2 Edition

Windows Client OS 10

Linux - Ubuntu, RHEL

Browsers Covered:

Google Chrome

Firefox

IE

Edge

Safari

Regression Areas to be Covered:

WebApp Integrations
api Integrations SIEM, Syslog, AD Integrations and all other integrations possible

Use-Cases for SIEM Integrations:

- Perform Initial data push to SIEM from WebApp
- Push the Data to WebApp
- Commit failure reason should be displayed somewhere

Integration Areas: (Dashboard/api)

  - During WebApp installation, provide credentials to be able to connect to SIEM

Functional and Integration Testing

Common Areas

- **Enable all possible Integrations**
- **SIEM Integrations and functionality check**
- **Syslog Integrations and functionality check**
- **Checkpoint Integrations**
- **AD Integrations**
- **Proxy Integrations**
- **Active/Active configurations (with and without)**

End to End Testing

Common Areas

- **WebApp to Dashboard/DB verifications**
- **Delivery Report on Dashboard (Data Feed information page)**
- **Invalidities, incidents on Dashboard**
- **Anomalies, Custom Attributes verifications**
- **E2E Automation Execution**

Negative Testing:

- With Large Size CSV file upload
- with Invalid file upload
- Files Otherthan CSV file

Performance Testing:

- **WebApp Performance when complete Large amount data load is done**

Security Testing:

- **WebApp Performance when complete data load is done using OWASP, Kali linux tools**

Upgrade Testing

Common Areas

- **WebApp upgrade from older version to newer versions.**
- **WebApp Upgrade from an older version with Integrations**
- **Post upgrade enable Active/Active and verify above configurations are intact and working**

Disaster Recovery

- **WebApp Crash and failover**
- **WebApp Restart and Active-Active Mode with other applications integrations**
- **Ability to recover the files picked by crashed WebApp by other WebApp's**
- **File locking mechanism**
- **H/A verification**
- **Reliability and endurance tests**

Automation Execution

- **Execute Testcases with Robot Framework**

Hardware and Software Requirements

- **Windows and Linux Machines**
- **Shared File System with different techniques SMB/CIFS, NFS etc. for active active modes**
- **Single WebApp and multiple WebApp's with/without cloud**

QA Owners and Estimations

Balu Paulraj - on complete Integration Activities

12 days (approximately) to complete E2E verification. This is excluding bug (filed if any) verification ...

Sample Testcases :

1. Validate Clerk able to insert a single record of working class hero into database via an API
2. Validate Clerk able to insert more than one working class hero into database via an API
3. Validate Clerk able to upload a csv file to a portal to populate the database from a UI
   3 a) Validate that First row of the csv file must be natid, name, gender, salary, birthday, tax
   3 b) Validate that Subsequent rows of csv are the relevant details of each working class hero
   3 c) Validate that A simple button that allows to upload a file from pc to the portal

4. Validate Bookkeeper, able to query the amount of tax relief for each person in the database
4 a)GET endpoint which returns a list consist of natid, tax relief amount and name
4 b) natid field must be masked from the 5th character onwards with dollar sign ‘$’
4 c) computation of the tax relief is using the formula as described:
4 d) After calculating the tax relief amount, it should be subjected to normal rounding rule to remove any decimal places
4 e) If the calculated tax relief amount after subjecting to normal rounding rule is more than 0.00 but less than 50.00, the final tax relief amount should be 50.00
4 f) If the calculated tax relief amount before applying the normal rounding rule gives a value with more than 2 decimal places, it should be truncated at the second decimal place and then subjected to normal rounding rule

5. Validate Governor able to see a button on the screen to dispense tax relief
5 a) The button on the screen must be red-colored
5 b) The text on the button must be exactly “Dispense Now”
5 c) After clicking on the button, it should direct me to a page with a text that says “Cash dispensed”