

*A Test Case Execution Report on*

## **Person Finder | Google**

*Under the guidance of*

**P. Santhi Thilagam (Professor)**

*Submitted by*

**Hardik Rana - 16C0138**

**Harshal Shinde - 16C0223**

**VII Sem B.Tech (CSE)**

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE & ENGINEERING**



**Department of Computer Science & Engineering  
Technology**

**National Institute of Technology Karnataka, Surathkal.**

***7 November 2019***

# 1. Unit Tests [Functional Testing]

1.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU01
<b>Test case description</b>	To verify if new person's entry is created in the proper format	<b>Test priority</b>	High
<b>Prerequisite</b>	The entry should be already existing	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a new entry using model class 2. Using the Assert function to verify if the entry created is stored in the right format	
<b>Input</b>		Data should be according to the data type of the column	
<b>Expected Result</b>		If the format we are getting will match the one which we are expecting, then it will show test case passed, otherwise it will show test case failed.	
<b>Actual Result</b>		Since our function is returning exactly the same output as what is expected it prints <b>Test case 1 passed in 2.93 seconds</b>	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		28 October	

<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 2.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU02
<b>Test case description</b>	To verify if entry for new photo is created	<b>Test priority</b>	High
<b>Prerequisite</b>	For test case to pass the photo entry must already exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a new entry by providing image data 2.Using the url builder creating the url for the new image entry 3. Compare the expected url with the generated url	
<b>Input</b>		Image name	
<b>Expected Result</b>		If the url which we are getting by using url builder for new image matches with the expected one,then it will show test case passed,otherwise it will show test case failed.	
<b>Actual Result</b>		Since our function is returning exactly the same url as what is expected it prints <b>Test case 1 passed in 1.72 seconds</b>	

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

### 3.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU03
<b>Test case description</b>	To verify the entry for person search (full text search) is created or not	<b>Test priority</b>	High
<b>Prerequisite</b>	For test case to pass the person search entry must already exist [or we can create one]	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Create a new search person entry using model person class 2.Using the Assert function to verify if we can search for the entry created based on different functions in the full_text_search file		

<b>Input</b>	Person's given name,family_name,entry_date etc..
<b>Expected Result</b>	If the entry we are searching already exists in the database then it will show test case passed,otherwise it will show test case failed.
<b>Actual Result</b>	Since we are checking only those data entries which exists in the database it will show, <b>Test case 3,4 passed in 12.29 seconds</b> (since both test cases 3,4 are in the same file only,we will get results for both together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

#### 4.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU04
<b>Test case description</b>	To verify the entry for person search which is created in test case 3 can be deleted or not	<b>Test priority</b>	High

<b>Prerequisite</b>	For test case to pass the person search entry must already exist [or we can create one]	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create a new search person entry using model person class [if not already exists] 2. Delete the created entry using functions of full_text_search file. 3.Using the Assert function to verify if entry is deleted or not [by searching for that entry in database]	
<b>Input</b>		Person's given name, family_name,entry_date etc..	
<b>Expected Result</b>		If the delete function work as expected and deletes entry from the database then it will get test case passed as result	
<b>Actual Result</b>		Since delete function is working properly and it deletes entry from database it shows, <b>Test case 3,4 passed in 12.29 seconds</b> (since both test case 3,4 are in same file only,we will get results for both together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		28 October	
<b>Executed By</b>		Hardik Rana	

<b>Date of Execution</b>	28 October
--------------------------	------------

## 5.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU05
<b>Test case description</b>	To verify that various constants which are set in const.py file are matching with the pfif (person finder interchange format) or not	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Get various constants which are set in const.py file. 2. Using the Assert function to verify if this constants are matching with the pfif constants or not.	
<b>Input</b>		Constants which are set in const.py file	
<b>Expected Result</b>		Since the constants which are set in const.py file are matching with the original file, it should show the result as test case passed.	
<b>Actual Result</b>		<b>Test case 5 passed in 0.05 seconds</b>	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	

<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 6.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU06
<b>Test case description</b>	To verify the format in which date is stored	<b>Test priority</b>	High
<b>Prerequisite</b>	The entry should be already existing	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a new date entry using datetime library of python 2. Using Assert function verify if the format of date stored is correct 3. We can also check if it raises different value error or exception when the date is not in given format.	
<b>Input</b>		A valid date	
<b>Expected Result</b>		If the date which we are getting from the create_validate function matches with the expected one, then it will show test case passed, otherwise it will show test case failed.	
<b>Actual Result</b>		Since our function is returning exactly the date as what is expected it prints	



	<b>Test case 6 passed in 0.89 seconds</b>
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 7.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU07
<b>Test case description</b>	To verify if normalise function(it converts all string in set to lowercase) is working	<b>Test priority</b>	High
<b>Prerequisite</b>	Set of predefined bad words should be created [With the SpamDetector class]	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Make an entry to the set using uppercase strings. 2. Use Assert function to check if the uppercase of the input string has been converted to lowercase		
<b>Input</b>	Set of words		

<b>Expected Result</b>	If the SpamDetector function works as expected then it will show test case passed as result.
<b>Actual Result</b>	Since SpamDetector function is working properly it prints, <b>Test case 7,8 passed in 0.03 seconds</b> (since both test cases 7,8 are in same file only,we will get results for both together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 8.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU08
<b>Test case description</b>	To verify if bad words and spam is detected [Input text is None or it contains comma in-between which is not needed]	<b>Test priority</b>	High
<b>Prerequisite</b>	Set of predefined	<b>Post-requirement</b>	NA

	bad words should be created [With the Spam-Detector class]		
<b>Test Procedure</b>		1. Use Assert function with the predefined bad words and check if their spam probability calculated by the function matches with that of the expected probability	
<b>Input</b>		Valid string	
<b>Expected Result</b>		If the spam probability we are getting by passing the word into the function matches with what is expected then it will show test case passed as result.	
<b>Actual Result</b>		Since our function is returning exactly the same spam probability as what is expected it prints, <b>Test case 7,8 passed in 0.03 seconds</b> (since both test cases 7,8 are in same file only,we will get results for both together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		28 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		28 October	

## 9.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU09
<b>Test case description</b>	To verify whether the strip function inside importer.py file is working correctly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Have some string to which will be passed to strip function of importer file. 2. Use Assert function for checking whether the string we are getting by applying the strip function of importer file on the string will remove leading and trailing spaces or not.	
<b>Input</b>		Set of Strings	
<b>Expected Result</b>		If the stripped words we are getting by passing the word into the function matches with what is expected then it will show test case passed as result.	
<b>Actual Result</b>		Since our function is returning exactly the same word as what is expected it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only, we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	

<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 10.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU10
<b>Test case description</b>	To verify whether the validate_datetime function inside importer.py file is working correctly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Using Assert function verify if the format of datetime stored is matching with the datetime we are getting by using datetime library of python 2. We can also check if it raises different value error when the datetime is not matching with the given format.	
<b>Input</b>		Set of Strings consisting of date-time	
<b>Expected Result</b>		If the result we are getting by passing the datetime into the function matches with what is expected then it will show test case passed as result.	

<b>Actual Result</b>	Since our function is returning exactly the same datetime as what is expected it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 11.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU11
<b>Test case description</b>	To verify whether the validate_boolean function inside importer.py file is working correctly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	Set of boolean flags (True,False)	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Using Assert function pass the boolean flag inside validate function of importer file		

	and check it
<b>Input</b>	Set of strings
<b>Expected Result</b>	If the bool word we are getting by passing the word into the function matches with what is expected then it will show test case passed as result.
<b>Actual Result</b>	Since our function is returning exactly the same word as what is expected it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 12.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU12
<b>Test case description</b>	To verify the format in which the note is stored	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA

<b>Test Procedure</b>	1. Create a new note entry 2. Use Assert function to check that the various fields of the note are in the correct format
<b>Input</b>	Name of the repo,Strings in the note
<b>Expected Result</b>	If the person record with notes is created properly then we should get result as a test case passed.
<b>Actual Result</b>	Since our function is creating person record with notes it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

### 13.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU13
<b>Test case description</b>	To verify if the create_person	<b>Test priority</b>	High



	function inside importer.py file is working properly or not		
<b>Prerequisite</b>	All fields[attributes of person class] which are there in the website should be known	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Take some dummy user fields data and create a model by using on create_person function in importer.py file 2.Use hasattr function of python with Assert to check whether particular attribute belongs to person class or not,and for other attributes check whether that particular attribute is set to what we gave in step1 or not	
<b>Input</b>		Attributes related to person	
<b>Expected Result</b>		If the person record is created properly then we should get result as a test case passed.	
<b>Actual Result</b>		Since our function is creating person record properly it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	

<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 14.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU14
<b>Test case description</b>	To verify if the import_records function inside importer.py file is working properly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return type of function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create dummy record list of different-different persons,some of which contains bad domain,some contains invalid date etc. 2.Call the import_records function by passing this created record list and other necessary parameters and store the results returned by this function in some other list/variables. 3.Use Assert function to check if total	

	written count is matching with what we are expecting and check whether the entries which are expected to be skipped are actually skipped or not and check total count of entries.
<b>Input</b>	List of some data related to different-different persons
<b>Expected Result</b>	If the person record is created properly then we will get result as a test case passed if all the parameters like written,skipped will match with the expected results.
<b>Actual Result</b>	Since our function is creating person record properly and all the parameters we get are as expected it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU15
<b>Test case description</b>	To verify if the import_records function inside importer.py file is while importing person records with notes also.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Steps will be the same as what we will do in test-case 14, just one more field note_id will be added. 2.Other steps like calling a function of importer.py file and storing results and comparison will be same as TU14.	
<b>Input</b>		List of some data related to different-different persons	
<b>Expected Result</b>		If the person record with notes is created properly then we will get result as a test case passed if all the parameters like written,skipped will match with the expected results.	
<b>Actual Result</b>		Since our function is creating person record with notes properly and all the parameters we get are as expected it prints, <b>Test case 9,10,11,12,13,14,15 passed in 3.13 seconds</b> (since all test cases 9-15 are	

	in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	28 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	28 October

## 16.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU16
<b>Test case description</b>	To verify whether the function rank_and_order inside indexing.py file is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Create some dummy person records by passing given_name,family_name fields. 2.Call the rank_and_order function by		

	<p>passing the above created list of dummy person lists, Textquery (prefix) you want to check and maximum entries you want in your results and store it in some list.</p> <p>3.Using Assert check whether the result list you got in step2 is matching with what you are expecting or not.</p>
<b>Input</b>	List of some data related to different-different persons
<b>Expected Result</b>	If the result we are getting by passing list of persons to the rank_and_order function will be the same as what we expected than it will show test case passed as result.
<b>Actual Result</b>	<p>Since we are getting the same result as what we expected,it prints</p> <p><b>Test case 16,17,18,19 passed in 1.18 seconds</b> (since all test cases 16-19 are in same file only,we will get results for all test cases together only)</p>
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU17
<b>Test case description</b>	To verify whether the function rank_and_order inside indexing.py file is working properly or not if given_name or family_name is given as a concatenation of more than one words.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Same as step1 of test-case 16, but this time add entries which contains family_name or given_name as a concatenation of more than one words. 2. Calling the function and comparing results steps will be the same as what we will do in test-case 20.		
<b>Input</b>	List of some data related to different-different persons [In which for entries given_name or family_name should contain more than one words]		
<b>Expected Result</b>	If the result we are getting by passing list of persons (with given_name or family_name as a combination of more than one		

	words)to the rank_and_order function will be the same as what we expected than it will show test case passed as result.
<b>Actual Result</b>	Since we are getting the same result as what we expected,it prints <b>Test case 16,17,18,19 passed in 1.18 seconds</b> (since all test cases 16-19 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 18.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU18
<b>Test case description</b>	To verify whether the function sort_query_words inside indexing.py file is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the	<b>Post-requirement</b>	NA



	parameters passed to the function and return values of function.		
<b>Test Procedure</b>		1.Take list of different-different words. 2.Use Assert function and check whether the results we are getting by passing this lists to sort_query_words function is matching with our expectation results or not	
<b>Input</b>		List of different-different words. [Which can be used to check sort by length,sort by popularity functionalities]	
<b>Expected Result</b>		If the result we are getting by passing a list of words to the sort_query_words function will be the same as what we expected than it will show test case passed as result.	
<b>Actual Result</b>		Since we are getting the same result as what we expected,it prints <b>Test case 16,17,18,19 passed in 1.18 seconds</b> (since all test cases 16-19 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		29 October	
<b>Executed By</b>		Hardik Rana	

<b>Date of Execution</b>	29 October
--------------------------	------------

## 19.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU19
<b>Test case description</b>	To verify if empty TextQuery is passed to search function inside indexing.py, it will give expected results or not [regression test]	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of search function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Use Assert function and check if the results we are getting by passing empty TextQuery to search function is an empty list or not		
<b>Input</b>	None		
<b>Expected Result</b>	Since empty text query is passed it should return empty array as result.		
<b>Actual Result</b>	Since we are getting the same result as what we expected, it prints <b>Test case 16,17,18,19 passed in 1.18 seconds</b> (since all test cases 16-19 are in same file only, we will get results for all test		

	cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 20.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU20
<b>Test case description</b>	To verify if the get_repo_and_action function inside main.py file is working properly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of get_repo_and_action function.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Use Assert function and check if the results we are getting by passing some		

	request to get_repo_and_action function is matching with what we are expecting or not
<b>Input</b>	String consisting URL
<b>Expected Result</b>	If the result we are getting by passing a setup_request to the get_repo_and_action function will be the same as what we expected than it will show test case passed as result.
<b>Actual Result</b>	Since we are getting the same result as what we expected,it prints <b>Test case 20,21,22,23 passed in 1.99 seconds</b> (since all test cases 20-23 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 21.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU21
<b>Test case description</b>	To verify whether there are bad	<b>Test priority</b>	High

	characters in the lang parameter or not [By using env function of main.py to construct 'env' object]		
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Get the request path and construct a webapp.Request object. 2. Apply setup_env function of main.py file on the above created webapp.Request object and get the env object. 3. Using Assert to check whether the env object contains any bad characters	
<b>Input</b>		String consisting of request path	
<b>Expected Result</b>		If the result we are getting by passing a setup_request (with bad characters in lang parameter) to the get_repo_and_action function will be the same as what we expected than it will show test case passed as result.	
<b>Actual Result</b>		Since we are getting the same result as what we expected, it prints <b>Test case 20,21,22,23 passed in 1.99 seconds</b> (since all test cases 20-23 are in same file only, we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	

<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 22.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU22
<b>Test case description</b>	To verify whether the language_menu_options[0] can be used as the default	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Get the request path and construct a webapp.Request object. 2. Set different language_menu_options using set_for_repo function of config.py file. 3. Use Assert to check whether first language can be used as the default or not		
<b>Input</b>	list of language_menu_options which will be passed to set_for_repo function.		
<b>Expected Result</b>	If the result we are getting by passing a setup_request and webapp.response() to the Main handler will be the same as what we expected then it will show test case passed as result.		

<b>Actual Result</b>	Since we are getting the same result as what we expected, it prints <b>Test case 20,21,22,23 passed in 1.99 seconds</b> (since all test cases 20-23 are in same file only, we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 23.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU23
<b>Test case description</b>	To verify whether Content Security Policy(CSP) is set when the React UI is enabled.	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Create response from webapp.response() library and handler. 2.Use Assert function to verify whether CSP is there in the response's header or not.		
<b>Input</b>	None		

<b>Expected Result</b>	Test case will pass.
<b>Actual Result</b>	<b>Test case 20,21,22,23 passed in 1.99 seconds</b> (since all test cases 16-19 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 24.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU24
<b>Test case description</b>	To verify whether the function convert_description_to_other [which converts description in PFIF 1.4 format to other older version] in pfif.py file is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be	<b>Post-requirement</b>	NA



	aware of the parameters passed to the function and return values of function to be tested		
<b>Test Procedure</b>		1.Use Assert function to check whether the result we are getting by passing some string to convert_description_to_other function is matching with what we are expecting or not.	
<b>Input</b>		Strings which will be passed as a parameter to function	
<b>Expected Result</b>		If the result we are getting by passing a string to the convert_description_to_other will be the same as what we expected than it will show test case passed as result.	
<b>Actual Result</b>		Since we are getting the same result as what we expected,it prints <b>Test case 24,25,26 passed in 1.34 seconds</b> (since all test cases 24-26 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		29 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		29 October	

## 25.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU25
<b>Test case description</b>	To verify whether the function maybe_convert_other_to_description [Converts 'other' in PFIF 1.3 and earlier to 'description' in PFIF 1.4 if 'other' has only 'description' field. Otherwise it returns 'other' without 'other' has only 'description' field. Otherwise it returns 'other' without] in pfif.py file is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function to be tested	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Use Assert function to check whether the result we are getting by passing some string to maybe_convert_other_to_description function is matching with what we are	

	expecting or not.
<b>Input</b>	None
<b>Expected Result</b>	If the result we are getting by passing a string to the maybe_convert_description_to_other will be the same as what we expected than it will show test case passed as result.
<b>Actual Result</b>	Since we are getting the same result as what we expected, it prints <b>Test case 24,25,26 passed in 1.34 seconds</b> (since all test cases 24-26 are in same file only, we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 26.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU26
<b>Test case description</b>	To verify parsing of an XML file for each test case.	<b>Test priority</b>	High

<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create person_records, note_records by passing test_case.xml file to parse_file function of pfif.py file 2.Use Assert to verify the person_records and note_records we got are matching with the expected results or not	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 24,25,26 passed in 1.34 seconds</b> (since all test cases 24-26 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		29 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		29 October	

## 27.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU27
<b>Test case description</b>	To verify whether test data is cached	<b>Test priority</b>	High

	in ram or not		
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create an instance of cache by using RamCache() class. 2.Put some data (key,value and ttl)in cache by using put function of RamCache() class. 3.Using Assert function check whether by passing key to get function of RamCache() class you are getting its associated value or not.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		29 October	
<b>Executed By</b>		Hardik Rana	

<b>Date of Execution</b>	29 October
--------------------------	------------

## 28.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU28
<b>Test case description</b>	To verify that test data for which ttl (time to leave) set to 0 will not be cached in ram	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create an instance of cache by using RamCache() class. 2.Put some data (key,value and ttl=0)in cache by using put function of RamCache() class. 3.Using Assert function check whether by passing key to get function of RamCache() class you are getting None or not	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for	

	all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

## 29.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU29
<b>Test case description</b>	To verify whether this cached data expires after ttl or not	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Create an instance of cache by using RamCache() class. 2.Put some data (key,value and ttl)in cache by using put function of RamCache() class.		

	3.Using Assert function before and after expiration of ttl to check whether you are getting value or not.
<b>Input</b>	None
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	29 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	29 October

### 30.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU30
<b>Test case description</b>	To check whether the clear function to clear the cache is working properly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be	<b>Post-requirement</b>	NA



	aware of the parameters passed to the function and return values of function which will be used		
<b>Test Procedure</b>		1.Create an instance of cache by using RamCache() class. 2.Put some data (key,value and ttl)in cache by using put function of RamCache() class. 3.Using Assert function before and after cache and compare both results	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		29 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		29 October	

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU31
<b>Test case description</b>	To verify whether the get function inside Resources class of resources.py function is working properly or not. [To verify whether Resources.get fetches a Resource from the datastore]	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.First create a function that will put a resource in datastore for testing. 2. After that with the use of Assert function we can check whether that resources is there or not. [We can also write delete function and assert to check after that]	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)	

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

## 32.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU32
<b>Test case description</b>	To verify whether the set_active_bundle_name function inside resources.py function is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.First create a function that will put a resource in datastore for testing. 2.Check the results of Assert function used		

	with get_localized and get_rendered before and after the set_active_bundle_name function is used
<b>Input</b>	None
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

### 33.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU33
<b>Test case description</b>	To verify whether the get_localized function inside resources.py function is working properly or not.	<b>Test priority</b>	High

<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Use Assert function to verify by passing the name,lang to get_localized and self.fetched() function we are getting what we expected or not.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		30 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		30 October	

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU34
<b>Test case description</b>	To verify whether the get_rendered function inside resources.py function is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Use Assert function to verify by passing the name,lang and other parameters to get_rendered and self.fetched() function we are getting what we expected or not.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 27,28,29,30,31,32,33,34 passed in 0.81 seconds</b> (since all test cases 27-34 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		30 October	

<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

### 35.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU35
<b>Test case description</b>	To verify whether the romaize_japanese_word function inside script_variant.py.py function is working properly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Using assert function to check whether the roman word we are getting by passing the japanese word to the function is same as actual roman word or not		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 35,36,37 passed in 0.91 seconds</b> (since all test cases 35-37 are in same file		

	only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

### 36.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU36
<b>Test case description</b>	To verify whether the romaize_word_by_unidecode function inside script_variant.py function is working properly or not.	<b>Test priority</b>	High



<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Using assert function to compare the result we are getting by passing word as an argument to function is same as what we expected or not	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 35,36,37 passed in 0.91 seconds</b> (since all test cases 35-37 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		30 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		30 October	

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU37
<b>Test case description</b>	To verify whether the romaize_search_query function inside script_variant.py.py function is working properly or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Using assert function to compare the result we are getting by passing word as an argument to function is same as what we expected or not.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 35,36,37 passed in 0.91 seconds</b> (since all test cases 35-37 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	

<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

### 38.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU38
<b>Test case description</b>	To verify whether the send_email functions error handling is working properly or not	<b>Test priority</b>	High
<b>Prerequisite</b>	None	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Take subject, receiver and sender email_address etc.. details. 2.Send mail using send_mail function and check whether there is any error or not.		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 38 passed in 1.32 seconds</b>		
<b>Status</b>	Pass		
<b>Remarks</b>	Test case verified		
<b>Created By</b>	Hardik Rana		

<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

### 39.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU39
<b>Test case description</b>	To verify if the email is empty then false is returned or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	None	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Use the set function of config.py file and set notification_email as empty message and unreviewed_notes_threshold 2. Create Handler and using Assert, check whether handler should_notify or not		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 39,40 passed in 1.32 seconds</b>		
<b>Status</b>	Pass		
<b>Remarks</b>	Test case verified		

<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

## 40.

<b>Test scenario ID</b>	Unit Test	<b>Test case ID</b>	TU40
<b>Test case description</b>	To verify whether results of handler to call should_notify changed based on threshold or not	<b>Test priority</b>	High
<b>Prerequisite</b>	None	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Steps will be same as what will be used in previous test case. 2. While using Assert different-different threshold values will be used.		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 39,40 passed in 2.07 seconds</b>		
<b>Status</b>	Pass		
<b>Remarks</b>	Test case verified		

<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	30 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	30 October

## 2. Server Tests [Functional Testing]

41.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU41
<b>Test case description</b>	Test that verifies that a config will return None, and	<b>Test priority</b>	High

	not throw an exception, when asked for the value of an unknown key		
<b>Prerequisite</b>	User should be aware of the parameters passed to the function and return values of function which will be used while writing tests.	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.First Set configuration settings for a particular repository. 2.Use Assert to verify whether it throws an error or not when the value for unknown key will be asked	
<b>Input</b>		Repo name and other parameters which will be passed to set_for_repo function	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 41,42,43 passed in 5.19 seconds</b> (since all test cases 41-43 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	

<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 42.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU42
<b>Test case description</b>	Test that verifies that in a config file a get will work with default value or not	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.First Set configuration settings for a particular repository. 2.Use Assert to verify whether the results we get by passing default_value and unknown_key to get function is equal to default_value or not		
<b>Input</b>	Repo name and other parameters which will be passed to set_for_repo function		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 41,42,43 passed in 5.19 seconds</b> (since all test cases 41-43 are in same file only,we will get results for all test cases together only)		
<b>Status</b>	Pass		
<b>Remarks</b>	Test case verified		



<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

### 43.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU43
<b>Test case description</b>	Test to verify that configuration instance is not callable	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Call the Configuration class constructor by providing repo name and store it in a variable. 2.For variable which we got in step1,using Assert verify that it is callable or not		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 41,42,43 passed in 5.19 seconds</b> (since all test cases 41-43 are in same file only,we will get results for all test cases together only)		
<b>Status</b>	Pass		

<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

#### 44.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU44
<b>Test case description</b>	Test to verify download of xml file	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Get url (location) from where you want to download file. 2.Use download_feed library and download file at that stored at that location 3.Using Assert check some of fields that they are matching with what you are expecting or not.		
<b>Input</b>	String consisting of Url from where you want to download xml file		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 44,45,46 passed in 4.73 seconds</b> (since all test cases 44-46 are in same file only,we will get results for all test cases together only)		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 45.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU45
<b>Test case description</b>	Test to verify download of csv file	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Get url (location) from where you want to download file. 2.Use download_feed library and download file at that stored at that location 3.Using Assert check some of fields that they are matching with what you are expecting or not.		
<b>Input</b>	String consisting of Url from where you want to download csv file		

<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	<b>Test case 44,45,46 passed in 4.73 seconds</b> (since all test cases 44-46 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 46.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU46
<b>Test case description</b>	Test to verify download of notes	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Get url (location) from where you want to download file. 2.Use download_feed library and download file at that stored at that location		

	3.Using Assert check some of fields that they are matching with what you are expecting or not
<b>Input</b>	String consisting of Url from where you want to download notes
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	<b>Test case 44,45,46 passed in 4.73 seconds</b> (since all test cases 44-46 are in same file only,we will get results for all test cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 47.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU47
<b>Test case description</b>	Test which will Verify that an error	<b>Test priority</b>	High

	message is shown when no CSV file is uploaded.		
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Get the id where file will be uploaded. 2.Use assert function to show error message when no csv file is uploaded	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 47,48 passed in 6.35 seconds</b> (since both test cases 47,48 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		31 October	

48.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU48
<b>Test case description</b>	Test which will Verify that an error message is shown when broken CSV file is uploaded.	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Use the same steps performed in previous test and check whether the csv file is broken or not	
<b>Input</b>		Broken csv file (which contains nul)	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 47,48 passed in 6.35 seconds</b> (since both test cases 47,48 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		31 October	

## 49.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU49
<b>Test case description</b>	Test to Verify that a photo is uploaded and properly served on the server.	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create a new person record with a profile photo 2.Verify the image is uploaded and displayed on the view page. 3.Verify the image is served properly by checking the image metadata using Assert 4.Follow the link on the image and verify the same image is served using Assert	
<b>Input</b>		Png/jpg image	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 49,50,51,52 passed in 5.92 seconds</b> (since all test cases 49-52 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	



<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 50.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU50
<b>Test case description</b>	Test to Upload an empty image and verifying that there will be no img tag in the view page	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1.Create a new person record with a zero-byte profile photo. 2.Verify there is no img tag in the view page using Assert		
<b>Input</b>	Empty image		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	<b>Test case 49,50,51,52 passed in 5.92 seconds</b> (since all test cases 49-52 are in same file only,we will get results for all test		

	cases together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 51.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU51
<b>Test case description</b>	Test that Uploads both profile photo and note photo and verifies the images are properly transformed and served on the server[i.e.,jpg is converted to png and a large image is resized to match	<b>Test priority</b>	High

	MAX_IMAGE_DIMENSION.]		
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create a new person record with a profile photo and a note photo. 2.Verify the images are uploaded and displayed on the view page. 3.Verify the profile image is converted to png using Assert. 4.Verify the note image is resized to match MAX_IMAGE_DIMENSION using Assert	
<b>Input</b>		Profile photo and note photo	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 49,50,51,52 passed in 5.92 seconds</b> (since all test cases 49-52 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		31 October	

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU52
<b>Test case description</b>	Test which Uploads a broken image and verifies an error message is displayed.	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Create a new person record with a broken profile photo. 2.Verify an error message is displayed using Assert	
<b>Input</b>		Broken image	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 49,50,51,52 passed in 5.92 seconds</b> (since all test cases 49-52 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		31 October	

53.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU53
<b>Test case description</b>	Test which Check the home page with no config (generic welcome page)	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Go to home page url 2.Use Assert to verify the text on home page	
<b>Input</b>		Home page url	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 53,54,55 passed in 3.74 seconds</b> (since all test cases 53-55 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	
<b>Executed By</b>		Hardik Rana	
<b>Date of Execution</b>		31 October	

## 54.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU54
<b>Test case description</b>	Test which Checks the start page with no language specified	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Go to Url of start page 2.Use Assert to verify the text on home page	
<b>Input</b>		Url of start page	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 53,54,55 passed in 3.74 seconds</b> (since all test cases 53-55 are in same file only,we will get results for all test cases together only)	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Hardik Rana	
<b>Date of creation</b>		31 October	

<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

## 55.

<b>Test scenario ID</b>	Server Test	<b>Test case ID</b>	TU55
<b>Test case description</b>	Test which Checks the start page with English language specified	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1.Go to Url of start page with english language specified. 2.Use Assert to verify the text on home page	
<b>Input</b>		Url of start page with english language specified	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		<b>Test case 53,54,55 passed in 3.74 seconds</b> (since all test cases 53-55 are in same file only,we will get results for all test cases	

	together only)
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Hardik Rana
<b>Date of creation</b>	31 October
<b>Executed By</b>	Hardik Rana
<b>Date of Execution</b>	31 October

### 3. Usability Tests [Non-Functional Testing]

56.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU56
<b>Test case description</b>	Search for a missing person	<b>Test priority</b>	High
<b>Prerequisite</b>	1. The website should be opened in a compatible web browser 2. Record of the	<b>Post-requirement</b>	NA



	missing person should exist		
<b>Test Procedure</b>		1. On the home page of the website, click on 'I'm looking for someone 2. Enter the name of the person 3. Click on the appropriate record	
<b>Input</b>		1. Name of the person to be searched	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed, and user is able to search the person successfully	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	

## 57.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU57
<b>Test case description</b>	Choose the Japan disaster and create a new record in it	<b>Test priority</b>	High

<b>Prerequisite</b>	1. The Japan disaster should exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on the 3 horizontal bars on top left 2. Select Japan 3. Click on 'I have information about someone' 4. Enter name of the person 5. Enter details of the person	
<b>Input</b>		Details for the new record	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and new record is created in japan disaster	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	

58.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU58
<b>Test case description</b>	Provide latest information of a missing person after record already exists	<b>Test priority</b>	High
<b>Prerequisite</b>	1. Record of person must already exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter name of the person 3. Click on the correct record 4. Click on 'I have information about someone' 5. Enter the details	
<b>Input</b>		Name of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and latest information about missing person is successfully added	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	

## 59.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU59
<b>Test case description</b>	Insert information of yourself as a missing person	<b>Test priority</b>	High
<b>Prerequisite</b>	1. There should be a disaster entry	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on the 3 horizontal bars on top left 2. Select Japan 3. Click on 'I have information about someone' 4. Enter your 5. Enter details	
<b>Input</b>		Details for entering into the record	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and latest information about missing person (yourslef) is successfully added	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	

<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 60.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU60
<b>Test case description</b>	Upload photo of missing person	<b>Test priority</b>	High
<b>Prerequisite</b>	1. Record of the missing person must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'I have information about this person' 5. Click on 'More information about status of the person' 6. Click on URL or Upload and then select the image 7. Enter your details and then click Submit	

<b>Input</b>	Photo in valid format
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	Test case passed and photo of a missing person is successfully added
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 61.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU61
<b>Test case description</b>	Delete a specific record	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on Delete this record	

<b>Input</b>	Name of the person
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	Test case passed and record is successfully deleted.
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 62.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU62
<b>Test case description</b>	Extend the expiry date of specified record	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'Extend expiration date by 60 days'		

<b>Input</b>	Name of the person
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	Test case passed and expiry date of a person record is successfully extended.
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 63.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU63
<b>Test case description</b>	Subscribe to Updates about missing person	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'Subscribe to updates about this person'	



<b>Input</b>	Name of the person
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	Test case passed and successfully subscribed to updates about missing person.
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 64.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU64
<b>Test case description</b>	Provide latest known location of the missing person using map	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person	

	3. Click on the correct record 4. Click on 'I have information about this person' 5. Click on 'More information about status of the person' 6. In the drop down section click on Show Map and then select the location on the map
<b>Input</b>	Name of the person, Location of the person
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	Test case passed and latest known location of a missing person is successfully added using map.
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

65.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU65
<b>Test case description</b>	Provide latest known location of the missing person	<b>Test priority</b>	High

	using address		
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'I have information about this person' 5. Click on 'More information about status of the person' 6. In the drop down section enter the address in the text box	
<b>Input</b>		Name of the person, Location of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and latest known location of a missing person is successfully added using address.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	

66.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU66
<b>Test case description</b>	Provide latest known location of the missing person using current location	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'I have information about this person' 5. Click on 'More information about status of the person' 6. In the drop down section click on Use Current Location	
<b>Input</b>		Name of the person, Location of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and latest known location of a missing person is successfully added using current location.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	

<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

## 67.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU67
<b>Test case description</b>	Mark a note as spam	<b>Test priority</b>	High
<b>Prerequisite</b>	Record and a note in the record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I have information about someone' 2. Enter the name of the person 3. Click on the correct record 4. Click on 'Report Spam' 5. Specify the reason and verify the captcha	
<b>Input</b>		Name of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and note is successfully marked as spam.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	

<b>Date of creation</b>	1 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	1 November

68.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU68
<b>Test case description</b>	Leave a feedback	<b>Test priority</b>	High
<b>Prerequisite</b>	NA	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Click on the three horizontal bars 2. Click on feedback 3. Fill the Google form		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	Test case passed and feedback is successfully submitted.		
<b>Status</b>	Pass		
<b>Remarks</b>	Test case verified		
<b>Created By</b>	Harshal Shinde		
<b>Date of creation</b>	1 November		
<b>Executed By</b>	Harshal Shinde		
<b>Date of Execution</b>	1 November		

69.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU69
<b>Test case description</b>	Find phone number of the person specified	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I am looking for someone' 2. Enter name of person 3. Select the person 4. Next to the phone number label click 'Click to reveal' 5. Verify the captcha	
<b>Input</b>		Name of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and phone number of the person specified is successfully found.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	

70.

<b>Test scenario ID</b>	Usability Test	<b>Test case ID</b>	TU70
<b>Test case description</b>	Find email id of the person specified	<b>Test priority</b>	High
<b>Prerequisite</b>	Record must exist	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Click on 'I am looking for someone' 2. Enter name of person 3. Select the person 4. Next to the email id label click 'Click to reveal' 5. Verify the captcha	
<b>Input</b>		Name of the person	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Test case passed and email id of the person specified is successfully found.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		1 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		1 November	



## 4. Load Tests [Non-Functional Testing]

71.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU71
<b>Test case description</b>	What happens if a sudden spike of users that the website cannot handle happens?	<b>Test priority</b>	High
<b>Prerequisite</b>	Website should be running	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1) Find the Max User Load capacity of your software application 2) Create a Testing Environment and configure it to record performance parameters. 3) Set up test scenarios based on expected	

	<p>maximum load to your Software Application using a Performance Tool</p> <p>4) Rapidly increase in load, to the system for a set period.</p> <p>5) Gradually reduce the load back to its original level.</p> <p>6) Analyze the performance graphs. Metric to be considered are Failures, Time Taken, Virtual Users, etc.</p>
<b>Input</b>	None
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	The website was able to deal with a spike of 400 users ramped up for 30 seconds. Test passed
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 72.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU72
<b>Test case description</b>	What if a high	<b>Test priority</b>	High

	number of users are working with the API/website for an extended period?		
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a dedicated test environment 2. Predict the following and set up test scenarios: -Number of users -Duration of connection 3. Run the test and measure the results	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		The website was put on a load of 100 users for 30minutes. The test passed without any server error	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		2 November	

## 73.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU73
<b>Test case description</b>	Check if the page load time is within the acceptable range	<b>Test priority</b>	High
<b>Prerequisite</b>	1. There must be an internet connection 2. Web servers should be running	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Open a compatible browser 2. Enter the URL in the address bar and press 'Enter' 3. Measure the amount of time taken by the website to load using a stopwatch	
<b>Input</b>		URL of the website	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		The website loaded within 10 seconds 9 out of 10 times. Test case passed	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	

<b>Date of Execution</b>	2 November
--------------------------	------------

## 74.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU74
<b>Test case description</b>	Check the response time for any action under a light load conditions.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a dedicated test environment 2. Predict the following and set up test scenarios: -Number of users -Connection speeds 3. Decide on a specific action(eg: login) to be performed 4. Measure the time taken for the login to happen	
<b>Input</b>		Depends on the action to be performed	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		The website was able to perform the other action properly	
<b>Status</b>		Pass	

<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 75.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU75
<b>Test case description</b>	Check the response time for any action under a moderate load conditions.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a dedicated test environment 2. Predict the following and set up test scenarios: -Number of users -Connection speeds 3. Decide on a specific action(eg: login) to be performed 4. Measure the time taken for the login to happen	

<b>Input</b>	Depends on the action to be performed
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	The website was able to perform the other action properly
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 76.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU76
<b>Test case description</b>	Check the response time for any action under a heavy load conditions.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a dedicated test environment 2. Predict the following and set up test	

	scenarios: -Number of users -Connection speeds 3. Decide on a specific action(eg: login) to be performed 4. Measure the time taken for the login to happen
<b>Input</b>	Depends on the action to be performed
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	The website was able to perform the other action properly
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 77.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU77
<b>Test case description</b>	Check CPU and memory usage under peak load conditions.	<b>Test priority</b>	High



<b>Prerequisite</b>	Tools should be set up to monitor the CPU and memory usage	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Create a dedicated test environment 2. Predict the following and set up test scenarios: -Number of users -Connection speeds 3. Decide on a specific action(eg: login) to be performed 4. Measure the CPU and memory usage for the login to happen	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		The CPU and Memory usage was high but they did not fail. Test case passed	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case passed	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		2 November	

78.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU78
-------------------------	-----------	---------------------	------

<b>Test case description</b>	Verify if a proper error message is shown when the system reaches the breakpoint i.e. crosses the maximum no. of permitted users or requests. The RAM, memory and network are all good.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Web servers must be running	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.		
<b>Input</b>	None		
<b>Expected Result</b>	Test case will pass		
<b>Actual Result</b>	We were unable to test it to that level that the servers failed		

<b>Status</b>	Incomplete
<b>Remarks</b>	-
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 79.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU82
<b>Test case description</b>	Check if more than the permitted no. of users or requests are performing different operation (searching for a person, adding information about someone etc) and if the system becomes unresponsive, an appropriate error message is shown about the data (not saved? – depends on the implementation).	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an	<b>Post-requirement</b>	NA

	internet connection and the page should be loaded on a compatible browser. Automation scripting tool must be set up		
<b>Test Procedure</b>		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>			
<b>Status</b>			
<b>Remarks</b>			
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		2 November	

80.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU80
<b>Test case description</b>	Verify if the system works as expected when maximum no. of users or requests are being processed.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Automation scripting tool must be set up	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		The system worked properly when maximum number of users were using it. Test case passed	
<b>Status</b>		Pass	

<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 81.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU81
<b>Test case description</b>	Verify that while more than the permitted no. of users or requests are performing the same operation (like searching for a person, etc) and if the system becomes unresponsive, an appropriate error message is shown about the data	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Automation	<b>Post-requirement</b>	NA

	scripting tool must be set up		
<b>Test Procedure</b>		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		We were unable to take the website to the stage that it became unresponsive	
<b>Status</b>		Incomplete	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		2 November	

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU82
<b>Test case description</b>	Verify if a proper error message is shown when the system reaches the breakpoint i.e. crosses the maximum no. of permitted users or requests. The RAM, memory and network are all slow.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Web servers must be running	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	



<b>Actual Result</b>	We were unable to test it to that level that the servers failed
<b>Status</b>	Incomplete
<b>Remarks</b>	-
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

### 83.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU83
<b>Test case description</b>	Verify if the response time for breaking point users or requests is in an acceptance value.	<b>Test priority</b>	High
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Automation scripting tool must be set up	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Gather the system data, analyze the system, define the stress test goals		

	2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Analyze the Stress Test results.
<b>Input</b>	None
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	The response time for breakpoint users was in acceptance.
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 84.

<b>Test scenario ID</b>	Load Test	<b>Test case ID</b>	TU84
<b>Test case description</b>	Verify the performance of the app or website when the network is very slow, a proper error message	<b>Test priority</b>	High

	should be shown for 'timeout' condition.		
<b>Prerequisite</b>	There should be an internet connection and the page should be loaded on a compatible browser. Automation scripting tool must be set up	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Verify if timeout message is shown	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		A timeout message was printed when the network was slow.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		2 November	
<b>Executed By</b>		Harshal Shinde	

Date of Execution	2 November
-------------------	------------

## 85.

Test scenario ID	Load Test	Test case ID	TU85
Test case description	Verify server which has more than one application running on it to check if the other application gets affected in high load	Test priority	High
Prerequisite	There should be an internet connection and the page should be loaded on a compatible browser. Automation scripting tool must be set up The server must have 2 applications running on it	Post-requirement	NA
Test Procedure		1. Gather the system data, analyze the system, define the stress test goals 2. Create the Stress testing automation scripts, generate the test data for the stress scenarios. 3. Run the Stress testing automation scripts and store the stress results. 4. Results Analysis: See the effects on the	

	other applications execution.
<b>Input</b>	None
<b>Expected Result</b>	Test case will pass
<b>Actual Result</b>	The other application did not get affected in high load conditions
<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	2 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	2 November

## 5. GUI Tests [Non-Functional Testing]

86.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU86
<b>Test case description</b>	To verify the size, width, height of different elements in web page	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Open the website in a compatible browser 2. Verify if the size and position of the elements are correct	
<b>Input</b>		None	
<b>Expected Result</b>		Test case will pass	
<b>Actual Result</b>		Pass, The height and width of the elements is displayed.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test Case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		3 November	
<b>Executed By</b>		Harshal Shinde	
<b>Date of Execution</b>		3 November	

87.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU87
<b>Test case description</b>	To verify the error messages that are getting displayed	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Open the website in a compatible browser 2. Perform a transaction which would generate an error	
<b>Input</b>		1. Name of the person to be searched	
<b>Expected Result</b>		The user is able to find the person	
<b>Actual Result</b>		As the error message is displayed, the test passes.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test Case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		3 November	
<b>Executed By</b>		Harshal Shinde	

<b>Date of Execution</b>	3 November
--------------------------	------------

## 88.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU88
<b>Test case description</b>	To test the different sections which are there on web-screen.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Open the website in a compatible browser 2. Perform various operations on the elements in different parts of the screen	
<b>Input</b>		None	
<b>Expected Result</b>		Pass	
<b>Actual Result</b>		As the various sections on the screen were working properly so the test passed.	
<b>Status</b>		Pass	
<b>Remarks</b>		Test Case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		3 November	
<b>Executed By</b>		Harshal Shinde	



<b>Date of Execution</b>	3 November
--------------------------	------------

## 89.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU89
<b>Test case description</b>	To verify whether the fonts in web-page are readable or not	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>		1. Open the website in a compatible browser 2. Verify if the font is of a minimum specified size	
<b>Input</b>		None	
<b>Expected Result</b>		Pass	
<b>Actual Result</b>		The minimum font fixed by us for text to be readable is 11px, so as all the text was above 11px the test passed	
<b>Status</b>		Pass	
<b>Remarks</b>		Test Case verified	
<b>Created By</b>		Harshal Shinde	
<b>Date of creation</b>		3 November	

<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 90.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU90
<b>Test case description</b>	To verify the screen in different resolutions with the help of zooming in and out like 620 x 460, 580 x 780, etc.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Zoom in and then verify the resolution 3. Zoom out and then verify the resolution		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The web elements got aligned properly on zooming in or out of the page and so the test passed		
<b>Status</b>	Pass		
<b>Remarks</b>	This test cannot be entirely automated and requires interaction with real users to be		

	passed
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 91.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU91
<b>Test case description</b>	To verify that alignment of text and other icons like buttons is proper or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Verify if the elements are in proper position		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The alignment was proper and so the test passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 92.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU92
<b>Test case description</b>	To test the colors of different fonts on web page.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Get the id of the text 2. Verify the color using assert		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The colors of various text elements were as per the expected ones and so the test passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

### 93.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU93
<b>Test case description</b>	To test the color of error and warning messages.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Generate the error/warning message 3. Get the id of the text 4. Verify the color using assert		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The error message of errors and warning message is red, the rgb values we received		

	in testing were the same, so the test passed
<b>Status</b>	Pass
<b>Remarks</b>	Test case passed
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 94.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU94
<b>Test case description</b>	Testing whether the image has good clarity or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Browse to the page with the image 3. Verify the clarity of the image by inspecting its resolution		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The person in the image were easily recognisable, so the test passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 95.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU95
<b>Test case description</b>	Testing the alignment of the images.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Browse to the page with the image 3. Verify the image alignment		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The alignment of the images were proper and so the test passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 96.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU96
<b>Test case description</b>	Testing of the scrollbars according to the size of the page if any.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Use the scrollbar to scroll through the page		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The page could be scrolled if it did not fit in one screen and so the test passed		



<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 97.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU97
<b>Test case description</b>	Testing of the color of the hyperlink.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Generate the error/warning message 3. Get the id of the text 4. Verify the color using assert		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The color of all hyperlinks was blue, so the test case passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 98.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU98
<b>Test case description</b>	Testing of the headings whether it is properly aligned or not.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Browse to the page with the image 3. Verify the headings' alignment		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	The headings were aligned properly, the test case passed		

<b>Status</b>	Pass
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November

## 99.

<b>Test scenario ID</b>	GUI Test	<b>Test case ID</b>	TU99
<b>Test case description</b>	Testing of the size of the images.	<b>Test priority</b>	High
<b>Prerequisite</b>	The website must be opened on a compatible browser	<b>Post-requirement</b>	NA
<b>Test Procedure</b>	1. Open the website in a compatible browser 2. Browse to the page with the image 3. Verify the size of the image		
<b>Input</b>	None		
<b>Expected Result</b>	Pass		
<b>Actual Result</b>	As the height and width of the image were as specified, the test case passed		
<b>Status</b>	Pass		

Remarks	Test case verified
Created By	Harshal Shinde
Date of creation	3 November
Executed By	Harshal Shinde
Date of Execution	3 November

## 100.

Test scenario ID	GUI Test	Test case ID	TU100
Test case description	Verify that after updating any field a proper confirmation message must be displayed.	Test priority	High
Prerequisite	The website must be opened on a compatible browser	Post-requirement	NA
Test Procedure	1. Open the website in a compatible browser 2. Update the field 3. Verify the confirmation message displayed		
Input	None		

<b>Expected Result</b>	Pass
<b>Actual Result</b>	No confirmation message is shown that the field has been updated. Test failed
<b>Status</b>	Failed
<b>Remarks</b>	Test case verified
<b>Created By</b>	Harshal Shinde
<b>Date of creation</b>	3 November
<b>Executed By</b>	Harshal Shinde
<b>Date of Execution</b>	3 November