Gurukula Application Test Report

Version 1.0

Prepared By: Anuj Kumar

Introduction:

The purpose of this document is to give high level overview about testing roadmap that will be followed for testing of **Gurukula** application. This document will list the detailed understanding about the Testing workflow that will be followed during Testing Life cycle.

System Description:

Gurukula application is a Java based application which is used to maintain Branches and Staffs information. Apart from existing users, a new user can also register and perform various operations like create, view, update and delete branches and staffs. Users can search for desired branches/staffs and can view paginated results . Also, users can update account details in account menu after logging into application.

How to launch Gurukula application:

1) Download application from below link:
https://github.com/PA-Reporting/staff

2) Launch the application using below command:
java –jar gurukula-0.0.1-SNAPSHOT.war

Note: *Java 1.8 is required to access the application.*

Scope of Testing:

In-Scope:

1) Functional Testing:

i. Manual and exploratory testing should be done in order to verify **Gurukula** application's functionalities.
ii. Automation testing should be done in order to reduce manual regression efforts for repetitive tests.

2) Non-Functional Testing:

i. UI testing should be done in order to verify that UI is not getting distorted after performing various actions and also to confirm that application is easy to use.
ii. Cross-browser testing should be done in order to verify that application works as intended on supported browsers.

Out of Scope:

1) Performance testing
2) Security testing

Test Strategy:

This section lists down the strategy to ensure the quality of Gurukula application via manual & automation testing.

1) Manually explore application to understand basic functionality.
2) Identify features in application.
3) Design test scenarios for testing of various features in application.
4) Identify automatable test scenarios and mark the test cases accordingly.
5) Design test scripts for automation testing
6) Execute both manual and automation tests against application in test.
7) Analyse test results to identify defects in application.

Tools used for automation:

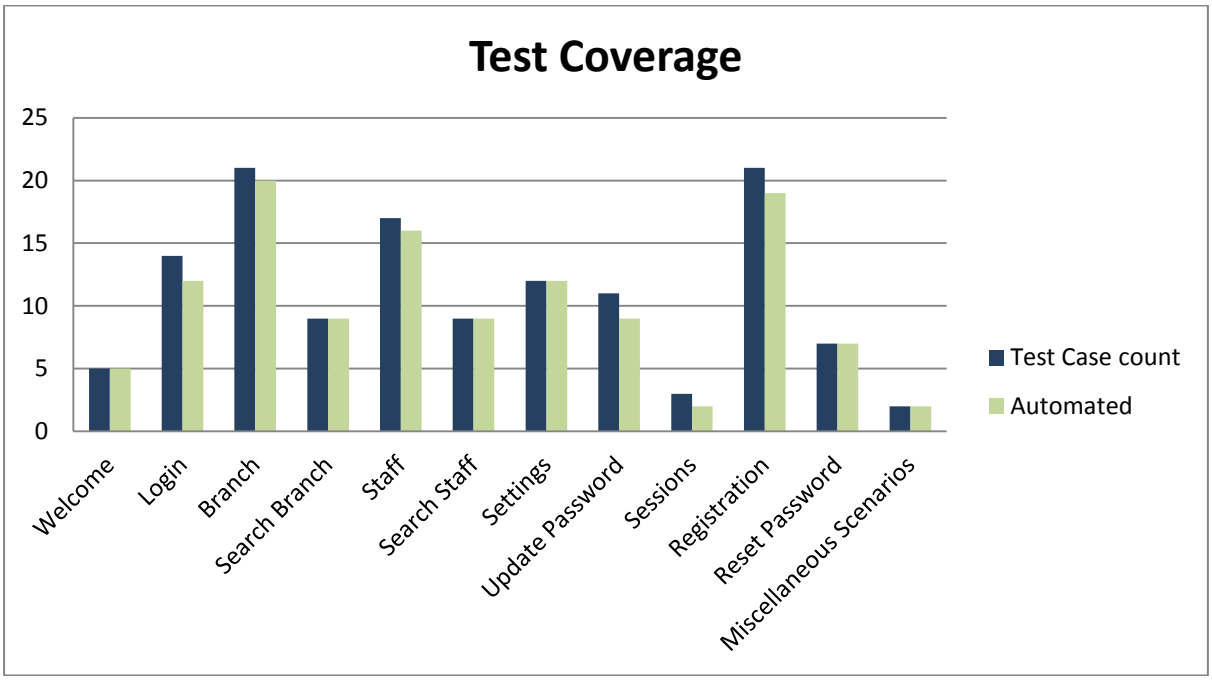
List of all the software and tools required for developing and executing automation framework.

1) Java (version 1.8)
2) Selenium (version 3.141.59)
3) TestNG (version 6.14.3)
4) Maven (version 3.5.3)
5) Log4j (version 1.2.17)
6) Google Chrome (version 74.0.3729.131)
7) Mozilla Firefox (version 66.0.3 (64-bit))

Test Deliverables:

1) ***Functionality Analysis:*** This section contains the details about total number of manual test cases per functionality. Below table contains total number of manual test cases per functionality. For details refer **'TestScenarios'** worksheet in **"Gurukula - Consolidated Report.xlsx".**

2) ***Automation feasibility :*** This section highlights the feasibility of automation against above identified manual test cases. For details refer **'TestScenarios'** worksheet in **"Gurukula - Consolidated Report.xlsx".**



Test Coverage				
S.No.	**Functionality**	**Test Case count**	**Automated**	**Percentage**
F1	Welcome	5	5	100
F2	Login	14	12	85.71
F3	Branch	21	20	95.24
F4	Search Branch	9	9	100
F5	Staff	17	16	94.12
F6	Search Staff	9	9	100
F7	Settings	12	12	100
F8	Update Password	11	9	81.82
F9	Sessions	3	2	66.67
F10	Registration	21	19	90.48
F11	Reset Password	7	7	100
F12	Miscellaneous Scenarios	2	2	100
	Total	**131**	**122**	**93.13**

3) ***Automated Details:*** *Automation test scripts are available at Github location* https://github.com/anujkumar21/gurukula-automation.git.

4) ***Defect List: List of defects*** found while executing manual & automated test scenarios is reported in **'Defects'** worksheet in **"Gurukula - Consolidated Report.xlsx".**

Automation Project structure:

This project is implemented in Page Object Model (POM) which helps in making the code more readable, maintainable, and reusable. Mentioned below are the description about different components of framework:

1) ***pom.xml:*** This file is used to perform build actions(clean, compile, install, test etc.) on project. All dependencies are included in pom.xml file.

2) ***drivers:*** This folder contains browser driver executables for chrome and firefox drivers.

3) ***log:*** This folder contains auto-generated log file.

4) ***common:*** This folder contains common constants and enums which are present throughout the application.

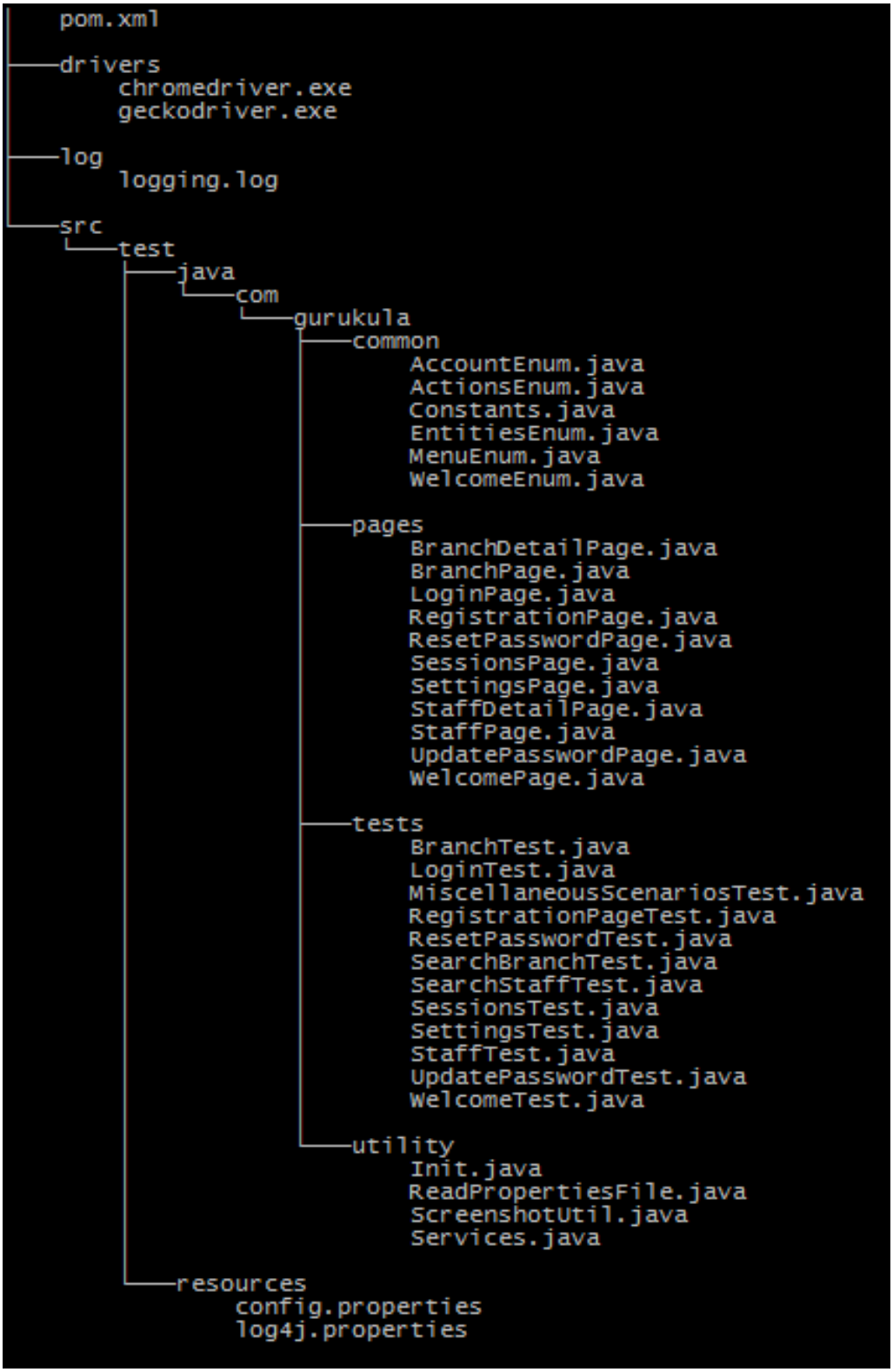
5) ***pages:*** This folder contains files corresponding to each functionality and has locators and methods to be used in the test methods.

6) ***tests:*** This folder contains test files corresponding to each functionality. Each file contains tests to test the functionality end to end.

7) ***utility:*** This folder contains classes to be utilized throughout the framework and is responsible to instantiate driver, read from properties file etc.

8) ***resources:*** This folder contains properties files which has details about the test application(url, username, password) and log4j properties.

Project Structure:



Automation Scripts Execution:

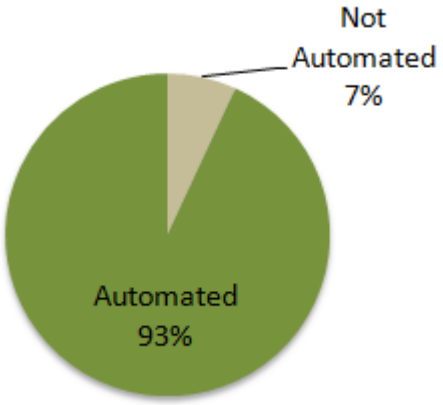
To execute test scripts please follow below steps:

1) Install Java & maven and set their respective paths in system variables.
2) Clone project (git clone https://github.com/)
3) Clean and compile project using
   mvn clean compile
4) To execute automation scripts, execute any of the following commands from command prompt:
   i. To execute all automation scripts (by default it will run on firefox)
      mvn test
   ii. To execute on particular browser, execute below command with parameter as 'firefox' or 'chrome'
      mvn test -Dbrowser=*<browser-name>*
      for e.g.: mvn test -Dbrowser=firefox
   iii. To execute test scripts for particular functionality execute below command with parameter as [branch, login, misc, password, register, resetPassword, searchBranch, searchStaff, sessions, settings, staff, welcome]
      mvn test -Dgroups=*<group-name>*
      for e.g.: mvn test -Dgroups=*branch*
   iv. To execute test scripts of any particular group on any particular browser
      mvn test -Dbrowser=<browser-name> -Dgroups=<group-name>
      for e.g.: mvn test -Dbrowser=firefox -Dgroups=branch

Automation Coverage:

Below table & pie chart contains the total automation coverage of **Gurukula** application.
Out of total 128 manual test cases, 93% test cases are automated.

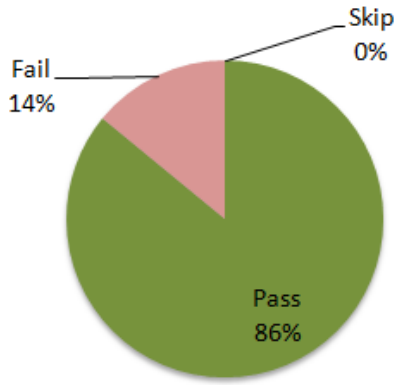
Automation Coverage	
Total Number of test cases	131
Not Automated	9
Automated	122



Automation Execution Status:

Below table & pie chart contains execution status of **Gurukula** automation.

Automation Execution Status	
Pass	105
Fail	17
Skip	0
Total:	**122**



Notes:

1) All **17 failures** are valid **product defects** and are logged in the **'Defects'** worksheet in **"Gurukula - Consolidated Report.xlsx".**

2) Framework has capability to **capture screenshot** for every failed test scripts. All the screenshots are stored at **". \target\screenshots".**

Automation Execution Status					
S No.	**Functionality**	**Total**	**Pass**	**Fail**	**%age**
F1	Welcome	5	5	0	100
F2	Login	12	12	0	100
F3	Branch	20	20	0	100
F4	Search Branch	9	9	0	100
F5	Staff	16	15	1	93.75
F6	Search Staff	9	8	1	88.89
F7	Settings	12	6	6	50
F8	Update Password	9	8	1	88.89
F9	Sessions	2	2	0	100
F10	Registration	19	14	5	73.68421
F11	Reset Password	7	5	2	71.43
F12	Miscellaneous Scenarios	2	1	1	50
	Total	**122**	**105**	**17**	**86.07**

TestNG Result: