AUTOMATION TESTING FRAMEWORK FOR LUMINOUS LMS

CONTENT

INTRODUCTION

WHAT LIBRARIES ARE USED

- Java v.1.8.0_45
- Maven v.3.5.0
- Selenium WebDriver v.2.0
- JUnit v.4.12
- JBehave v.1.6.0
- Serenity BDD v.1.1.26

INTRODUCTION

MAVEN

- Maven is a tool that can be used for building and managing any Java-based project

- To run project, Maven uses pom.xml file which describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plug-ins

- Maven dynamically downloads Java libraries and Maven plug-ins from one or more repositories such as the Maven 2 Central Repository, and stores them in a local cache

- To accurately build, test and install Maven project the following command is used:

mvn install

- Full list of Maven commands is available on Maven official website

INTRODUCTION

SELENIUM WEB DRIVER

Selenium WebDriver:

- Is a suite of tools to automate web browsers across many platforms
- Allows Java code to interact with browser and perform basic actions like click, type text to a text field, swipe etc. This is implemented through a browser-specific browser driver, which sends commands to a browser, and retrieves results
- Supports multiple browsers such as Firefox, Chrome, Internet Explorer, or Microsoft Edge
- Deploys on Windows, Linux, and OS X platforms
- Is fully implemented and supported in Java, C#, Ruby, JavaScript and Python

INTRODUCTION

JUNIT

- JUnit is a unit testing framework for the Java programming language
- JUnit provides set of methods that allows to compare expected result with actual result after test execution
- For example, assertEquals() method can be used to compare actual page header text with expected one. If expected and actual results are different, JUnit marks test as failed and passes results of test execution to reporting system.
- Also JUnit has variety of special methods that makes test script code easy-readable. For example, if you simply mark your java method with @BeforeTest annotation, your method will be executed before every test. You don’t have to instantiate your method before each test manually:

```
@BeforeTest

public void beforeTest() {

    browser.clearCookies();

    System.out.println(“Browser has started”); }
```

INTRODUCTION

JBEHAVE

- JBehave is a framework for Behaviour-Driven Development (BDD)
- Behaviour-Driven Development (BDD) is about implementing an application by describing it from the point of view of its stakeholders
- JBehave library allows to create easy-readable test scenarios in **Given-When-Then** format and link these scenarios with Java code
- In this way JBehave makes development practices more accessible and intuitive to newcomers and experts alike

INTRODUCTION

JBEHAVE (CONTINUATION)

This is the example of story and its test scenario that will be parsed by JBehave library

Narrative:

In order to communicate effectively to the business some functionality

As a development team

I want to use Behaviour-Driven Development

Scenario: A scenario is a collection of executable steps of different type

Given step represents a precondition to an event

When step represents the occurrence of the event

Then step represents the outcome of the event

INTRODUCTION

SERENITY BDD

Serenity BDD is a library that:

- Produces rich meaningful test reports (or "living documentation") that not only report on the test results, but also what features have been tested

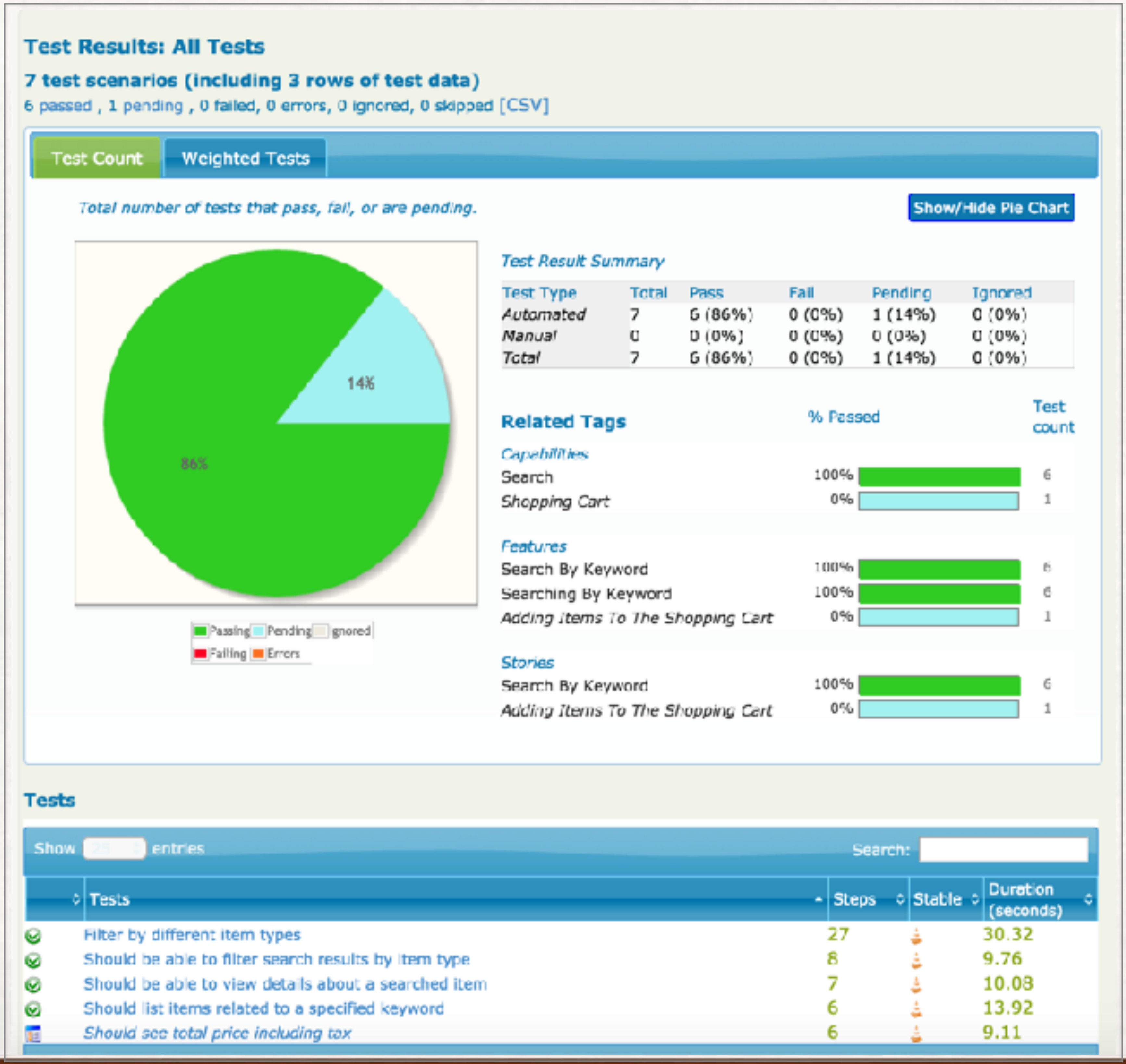
- Maps your automated tests back to your requirements

- Shows how much of your application is actually being tested

- Takes screenshot after each test step and links screenshots to the test execution report

INTRODUCTION

SERENITY BDD TEST EXECUTION REPORT



Test Results: All Tests

7 test scenarios (including 3 rows of test data)

6 passed , 1 pending , 0 failed, 0 errors, 0 ignored, 0 skipped [CSV]

Total number of tests that pass, fail, or are pending.

🟩 Passing	Pending	Ignored
🟥 Failing	Errors	

Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	7	6 (86%)	0 (0%)	1 (14%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	7	6 (86%)	0 (0%)	1 (14%)	0 (0%)

Related Tags	% Passed	Test count
Capabilities		
Search	100%	6
Shopping Cart	0%	1
Features		
Search By Keyword	100%	6
Searching By Keyword	100%	6
Adding Items To The Shopping Cart	0%	1
Stories		
Search By Keyword	100%	6
Adding Items To The Shopping Cart	0%	1

Tests

Show [illegible] entries   Search:

	Tests	Steps	Stable	Duration (seconds)
✓	Filter by different item types	27		30.32
✓	Should be able to filter search results by item type	8		9.76
✓	Should be able to view details about a searched item	7		10.08
✓	Should list items related to a specified keyword	6		13.92
	Should see total price including tax	6		9.11

HOW LUMINOUS LMS APPLICATION

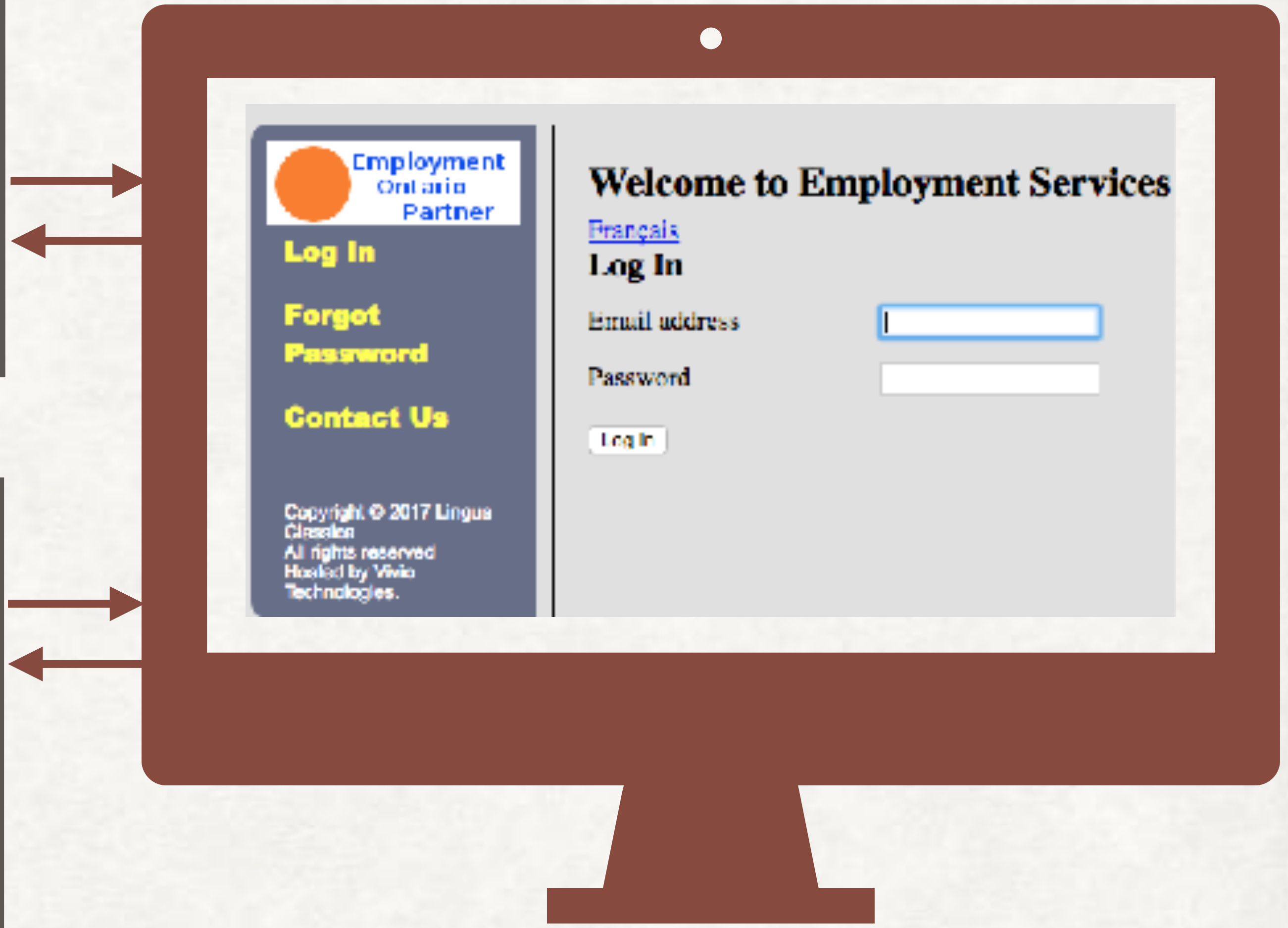
WORKS

Tomcat Server

With deployed Luminous LMS application

MySQL Server

With database that contains tables with application information such as user credentials, messages, training materials, links to courses etc.



Copyright © 2017 Lingua Glossion
All rights reserved
Hosted by Vivio Technologies.

Welcome to Employment Services

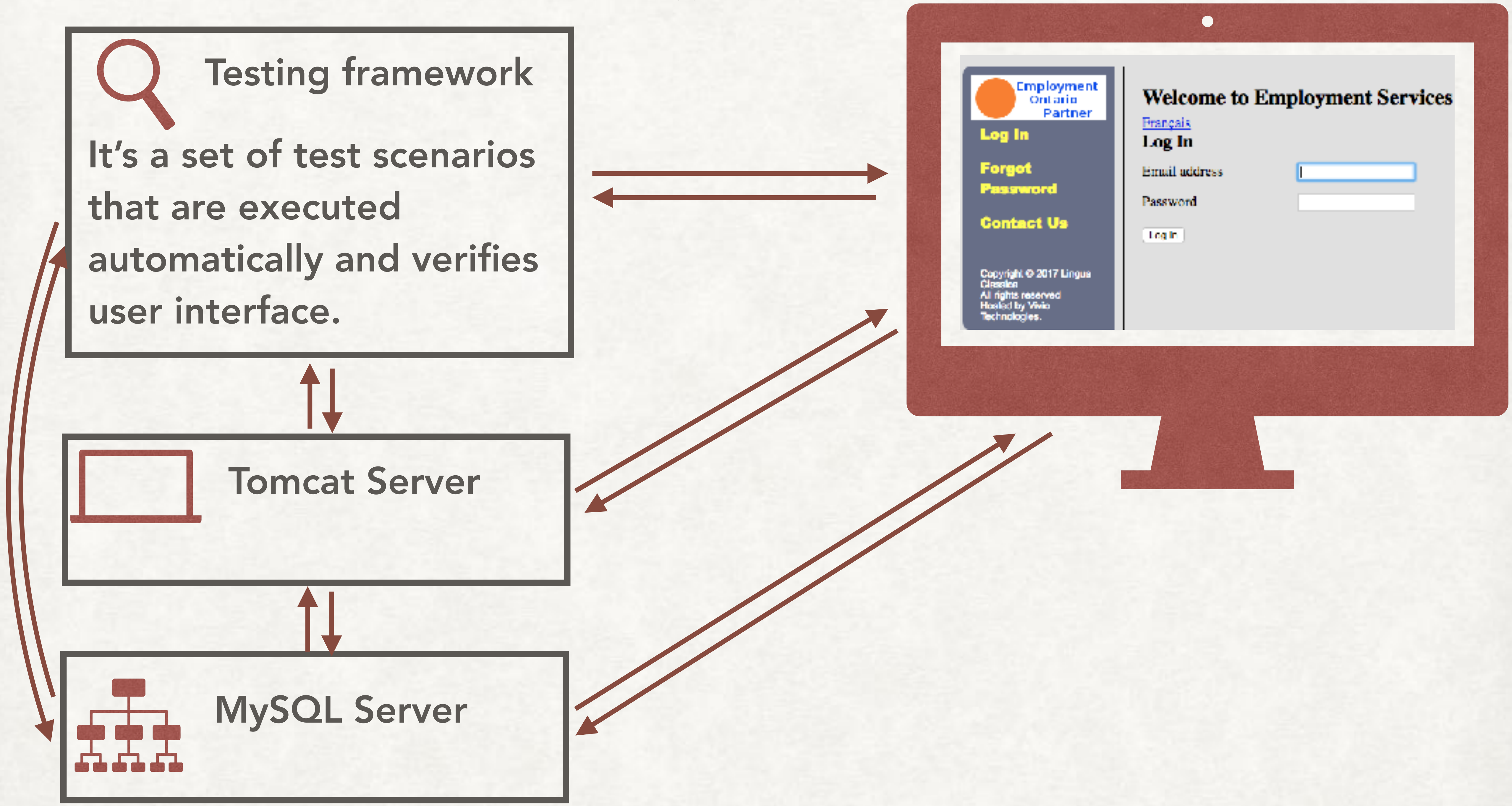
Français

Log In

Email address

Password

HOW TESTING FRAMEWORK WORKS



It’s a set of test scenarios that are executed automatically and verifies user interface.

Copyright © 2017 Lingua Classica
All rights reserved
Hosted by Vivio Technologies.

Welcome to Employment Services

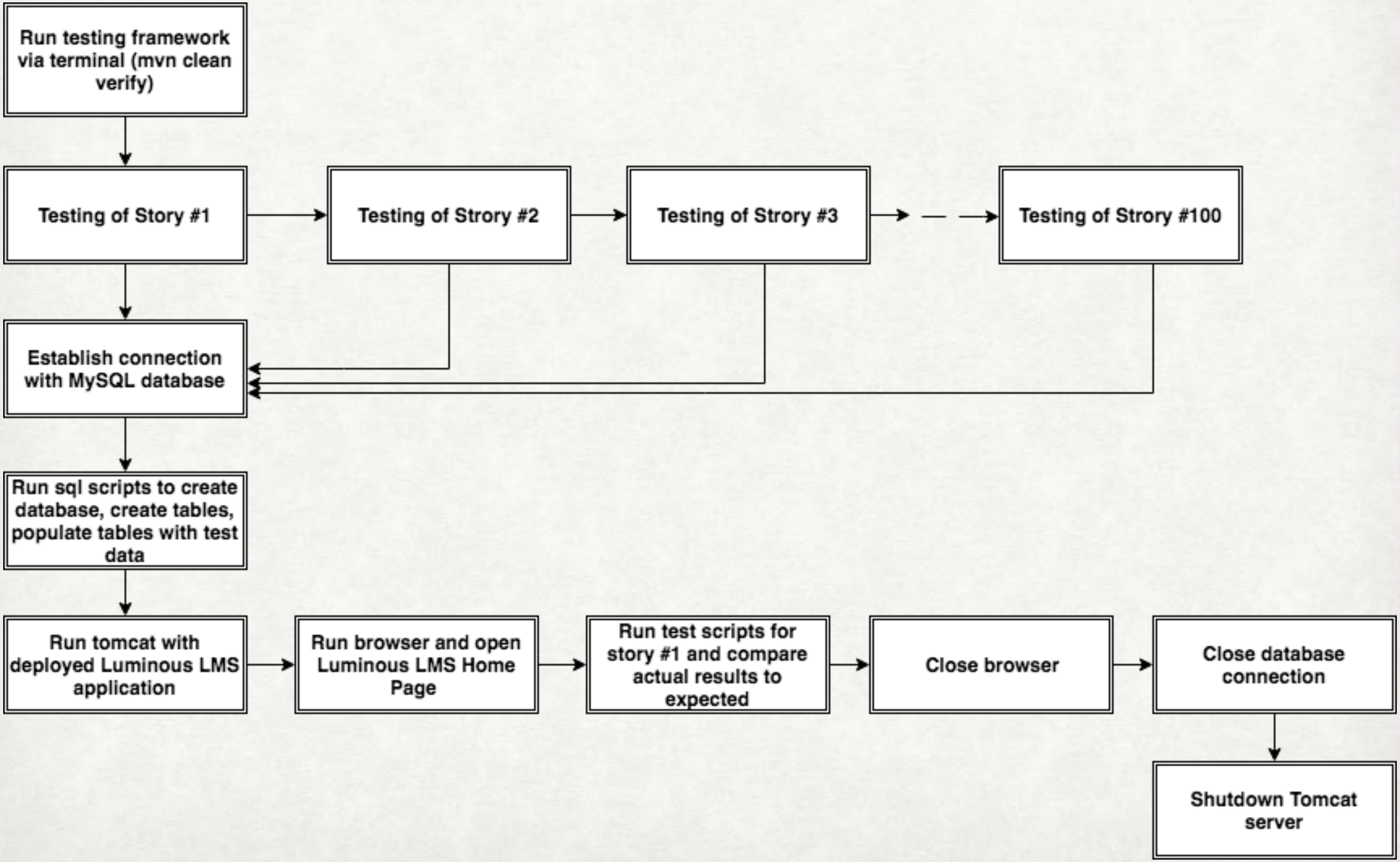
Français

Log In

Email address

Password

STORY TESTING LIFECYCLE


Run testing framework via terminal (mvn clean verify)
Testing of Story #1
Testing of Strory #2
Testing of Strory #3
Testing of Strory #100
Establish connection with MySQL database
Run sql scripts to create database, create tables, populate tables with test data
Run tomcat with deployed Luminous LMS application
Run browser and open Luminous LMS Home Page
Run test scripts for story #1 and compare actual results to expected
Close browser
Close database connection
Shutdown Tomcat server

INSTALLATION INSTRUCTIONS FOR TESTING FRAMEWORK

REQUIRED INFORMATION

- Before starting setting up the test environment make sure that you:
- installed Java, Maven, MySQL server, Tomcat server
- checked out main application code from repository by link:

  svn://linguaclassicacloud.com/root/repos/LuminousLMS/trunk

- and are able to run main application in Eclipse
- checked out testing framework code from repository by link:

  svn://linguaclassicacloud.com/root/repos/LuminousLMSTests/trunk

- imported testing framework code to Eclipse in the same way as you did it for main application code
- know path to Tomcat installation folder
- know path to LuminousLMS and LuminousLMSTests projects

INSTALLATION INSTRUCTIONS FOR TESTING FRAMEWORK

DEPLOYING THE MAIN APPLICATION TO TOMCAT SERVER FOR TESTING PURPOSES

- *Create web archive file of the main application project. In Eclipse in left menu select LuminousLMS project. In menu '**File**' choose **Export**. In the list expand Web and select WAR file. Click '**Next**'. On the next screen set Destination to any folder. Check **'Optimize for a specific server runtime'** and choose version of Tomcat server that you have. Click Finish. LuminousLMS.war file was saved to the folder you chose. Copy LuminousLMS.war file to webapps folder in Tomcat installation directory*

- *Every time you run Tomcat server LuminousLMS application will be deployed to the server automatically*

- *To check that main application was deployed correctly open command line and execute the following command:*

```
<your path to tomcat installation folder>/bin/startup.sh (for Mac OS)

<your path to tomcat installation folder>//bin//startup.bat (for Windows)
```

- Open browser and check that Luminous LMS application Home Page is loaded

- To shutdown Tomcat server in command line execute the following command:

```
<your path to tomcat installation folder>/bin/shutdown.sh (for Mac OS)

<your path to tomcat installation folder>//bin//shutdown.bat (for Windows)
```

It is very important to create and copy LuminousLMS.war file to Tomcat folder every time when you check out the main application code

INSTALLATION INSTRUCTIONS FOR TESTING FRAMEWORK

SET UP TESTING FRAMEWORK PROPERTIES

- There are three files that contain testing framework properties:

  tomcat.properties, serenity.properties, database.properties

- Property files are located in root folder of testing framework

- Before running test framework change property file in testing framework project depending on your operating system and project location:

  - in tomcat.properties file

    for Windows set **tomcat.run** and **tomcat.shutdown** properties to path to startup.bat and shutdown.bat files in tomcat installation folder, for example

    C://<your path to tomcat folder here>//bin//startup.bat

    Comment **tomcat.run** and **tomcat.shutdown** properties for Mac OS if you use Windows

  - do not change database.properties file

  - in serenity.properties file set webdriver.driver property to firefox. Note that Firefox browser version should be 39.0 or lower to be compatible with testing framework

INSTALLATION INSTRUCTIONS FOR TESTING FRAMEWORK

HOW TO RUN PROJECT

- Open Terminal (or Command Prompt on Windows) and navigate to testing framework project folder that contains pom.xml file. Command may look like this:

```
cd <your path to testing framework folder>/LuminousLMSTest/trunk
```

- Then type

```
mvn clean verify
```

- *This command will delete some files from previous test execution, e.g. previous test report data, and run all test scripts*

- *You should see new browser window launching LMS web application. Terminal will show you current test progress*

- *Test execution is completed when you see "BUILD SUCCESS" in Terminal*

TEST FRAMEWORK STRUCTURE

PROJECT FOLDER

Project folder contains

- src (source) folder - all code is stored here
- target folder - test execution reports are stored here
- pom.xml - configuration file that Maven library uses in order to install and run test framework
- serenity.property - configuration file with parameters for test execution such us browser name, browser settings, default application URL etc.
- tomcat.property - configuration file with parameters for Tomcat server
- database.property - configuration file with database settings and path to sql scripts for creating and populating testing database

TEST FRAMEWORK STRUCTURE

SRC FOLDER

Stories with their test scenarios are stored in '*resources*' folder

/src/test/resources/stories/

Source code

/src/test/java/com/luminous/

'luminous' folder contains:

- ***'pages'*** with list of application pages
- ***'steps'*** folder with list of test scenarios that implemented on Java. Each java class in, for instance, 'authorization' folder corresponds to scenario in authorization.story file
- ***'serenity'*** folder (/luminous/steps/) with list of users. Each user class contains steps, that can be performed only by this type of user

TEST FRAMEWORK STRUCTURE

TARGET FOLDER

- ‘**Target**’ folder is used by different maven plugins to place their reports here

- Our test framework is built by **serenity-maven-plugin** (you can find it in pom.xml). Serenity-maven-plugin stores its report in

  trunk/target/site/serenity/

- After the test execution when you see in Terminal (or Command Prompt) “BUILD SUCCESS”, navigate to ***trunk/target/site/serenity/*** and open ***index.html*** in your browser. You should see test execution report

- If after test execution project build failed, report will not be generated

TEST FRAMEWORK STRUCTURE

POM.XML

- pom.xml keeps links to all libraries and plugins that needed to run test framework
- Project setting are stored in xml format between tags such us <build></build>, <dependency></dependency>, <plugin></plugin>, <repository> </repository>, etc.
- Build plugins will be executed during the build and they should be configured in the <build> element from the POM
    - *maven-failsafe-plugin - aggregates report after test execution*
    - *maven-surefire-plugin - runs test framework*
    - *maven-compiler-plugin - compiles code before running*
- Dependency tag is used to add library to the project. For instance, JUnit library is linked to our project through pom.xml:

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

TEST FRAMEWORK STRUCTURE

SERENITY.PROPERTIES

- In serenity.properties file we can declare default browser

```
webdriver.driver=chrome,
```

or set up taking screenshots after scenario failed

```
serenity.take.screenshots=AFTER_EACH_STEP
```

- List of all properties that can be placed in serenity.properties file are listed in official Serenity BDD documentation under section 22. Serenity System Properties and Configuration

INSTRUCTIONS

HOW TO RUN TEST SCRIPTS IN DIFFERENT BROWSERS

- In order to run test scenarios in different browsers, webdriver.driver property in serenity.properties file should be set to browser name. For instance, firefox, chrome, fantomjs or iexplorer:

    webdriver.driver=chrome

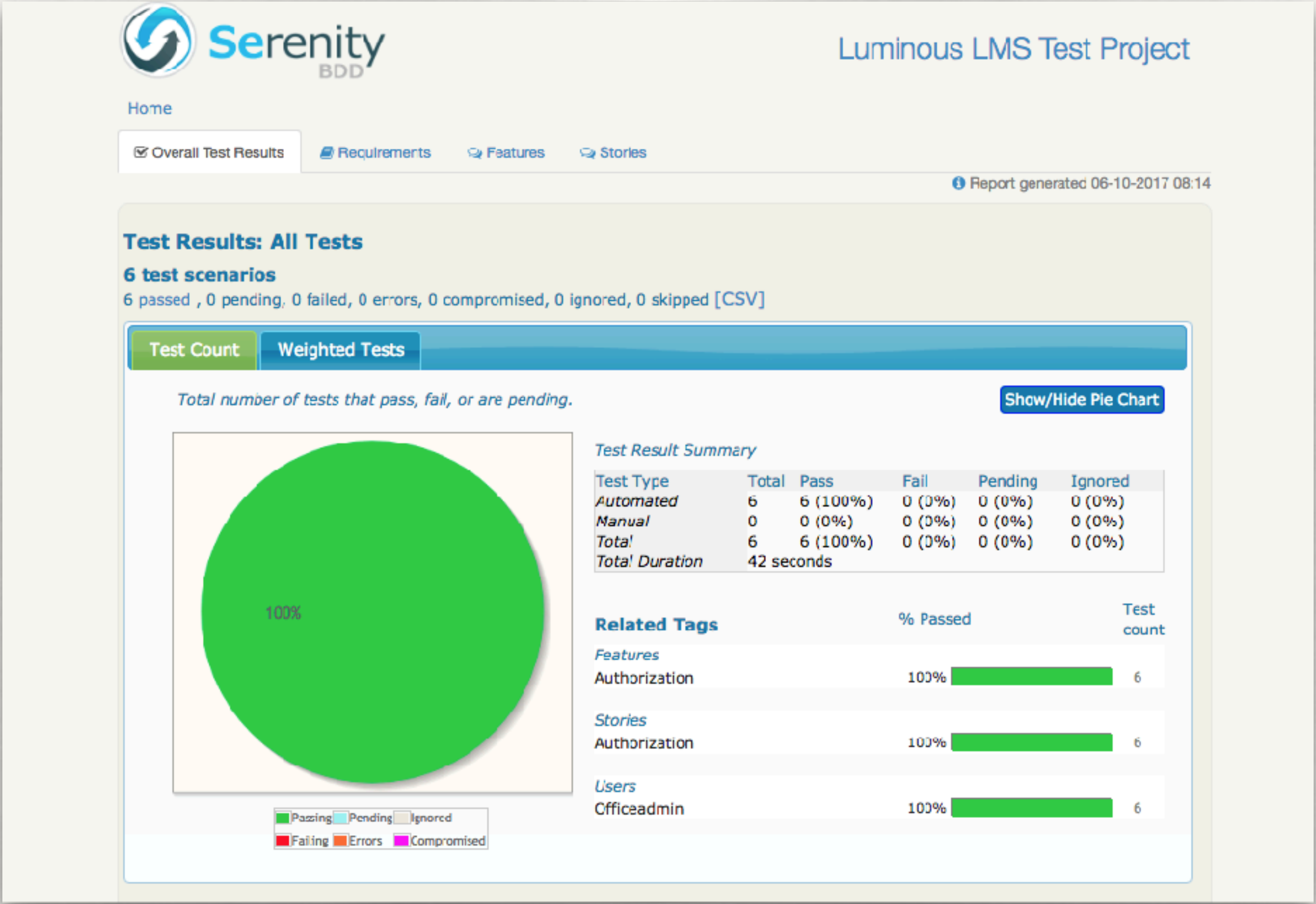
- To allow test code to interact with browser, browser-specific driver is needed. Driver for Firefox is embedded into Selenium WebDriver.

- For Chrome browser download browser driver. Unzip and put it in your project folder (for instance, */trunk/driver/chromedriver.exe*)

- Add property to serenity.properties:

    webdriver.chrome.driver = <path to your project>/trunk/driver/chromedriver.exe

TEST EXECUTION REPORT



Luminous LMS Test Project

Report generated 06-10-2017 08:14

Test Results: All Tests

6 test scenarios

6 passed , 0 pending, 0 failed, 0 errors, 0 compromised, 0 ignored, 0 skipped [CSV]

Total number of tests that pass, fail, or are pending.

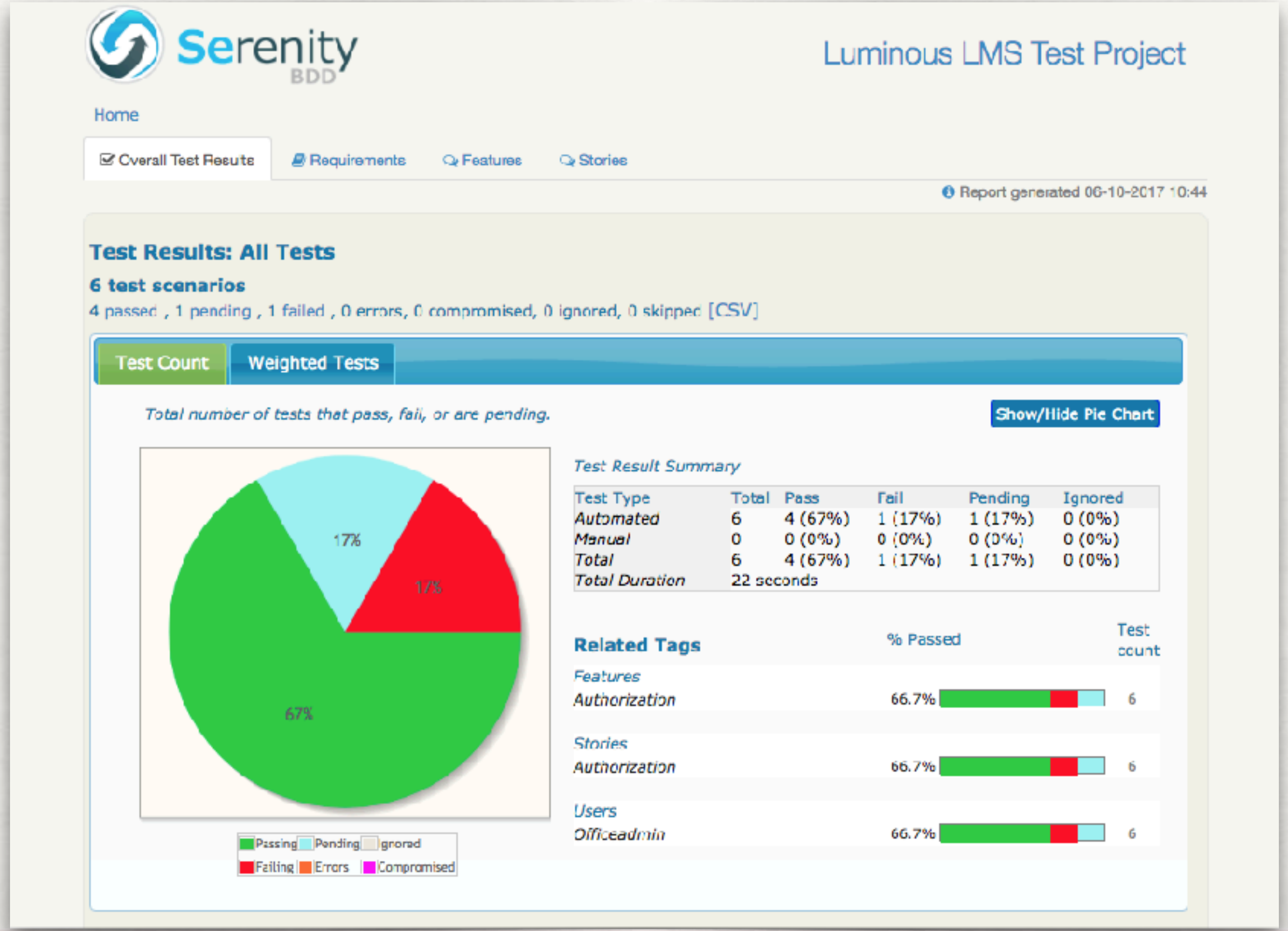
Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	6	6 (100%)	0 (0%)	0 (0%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	6	6 (100%)	0 (0%)	0 (0%)	0 (0%)
Total Duration	42 seconds				

Related Tags	% Passed	Test count
Features		
Authorization	100%	6
Stories		
Authorization	100%	6
Users		
Officeadmin	100%	6

TEST EXECUTION REPORT

POSSIBLE TEST EXECUTION RESULTS



Serenity BDD

Luminous LMS Test Project

Home

Overall Test Results | Requirements | Features | Stories

Report generated 06-10-2017 10:44

Test Results: All Tests

6 test scenarios

4 passed , 1 pending , 1 failed , 0 errors, 0 compromised, 0 ignored, 0 skipped [CSV]

Test Count | Weighted Tests

Total number of tests that pass, fail, or are pending.

Show/Hide Pie Chart

17%
17%
67%
🟩 Passing 🟦 Pending ⬜ Ignored
🟥 Failing 🟧 Errors 🟪 Compromised

Test Result Summary

Test Type	Total	Pass	Fail	Pending	Ignored
Automated	6	4 (67%)	1 (17%)	1 (17%)	0 (0%)
Manual	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Total	6	4 (67%)	1 (17%)	1 (17%)	0 (0%)
Total Duration	22 seconds				

Related Tags	% Passed	Test count
Features		
Authorization	66.7%	6
Stories		
Authorization	66.7%	6
Users		
Officeadmin	66.7%	6

HOW TO WRITE TEST SCRIPT. EXAMPLE

STORY FILE

- To start with test script creation, add new story folder to *trunk/src/test/resources and put there new story file:*
    - *trunk/src/test/resources/calendar/calendar.story*
- *Put in calendar.story file story that should be tested:*

Meta:

@officeadmin @client @consultant @login

Narrative:

As an Office Admin, or Consultant, or Client

I want to be able to navigate to Landing Page

So that I am able to see calendar

HOW TO WRITE TEST SCRIPT. EXAMPLE

STORY FILE (CONTINUATION)

- Put test scenarios in Given-When-Then format below the story

  Scenario: Login as Office Admin and verify

  that calendar is displayed on Landing Page

  Given user is on Home Page

  When user is logged in with username

  ‘admin@100.com' and password '12345'

  Then user should see calendar on Landing Page

  - Parameters in single quotes (‘admin@100.com' and ‘12345') will be passed to Java code and will be used for user authorization during scenario execution

HOW TO WRITE TEST SCRIPT. EXAMPLE

STORY REPRESENTATION IN JAVA CLASS

- The second step is to create Java class for each scenario
- Navigate to ***trunk/src/test/java/com/luminous/steps/*** and create folder for your scenario classes. Put there empty Java class with name that consists of user role and short scenario description:

  trunk/src/test/java/com/luminous/steps/calendar/OfficeAdminViewCalendar.java

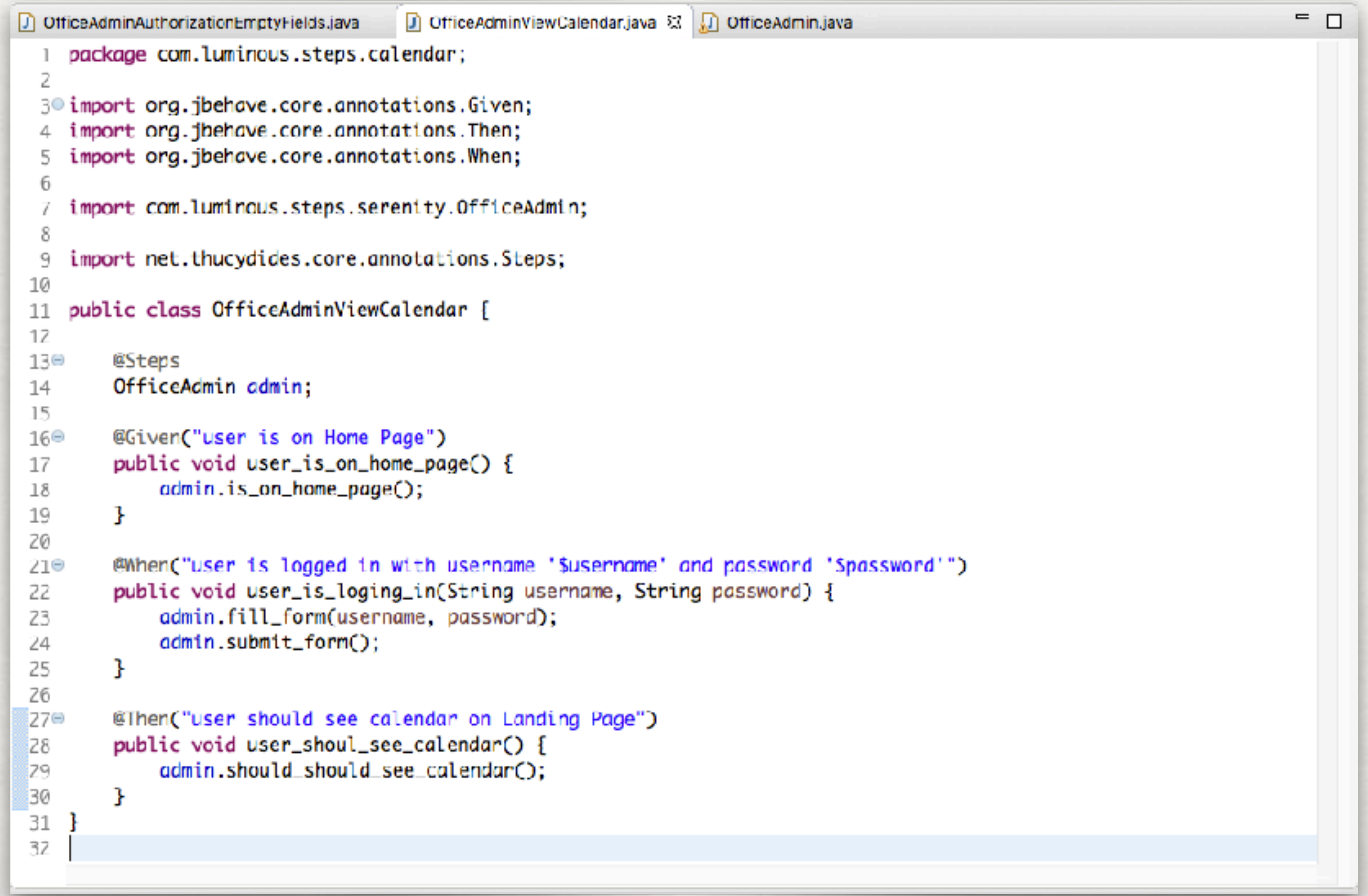
  or

  trunk/src/test/java/com/luminous/steps/calendar/ClientViewCalendar.java

HOW TO WRITE TEST SCRIPT. EXAMPLE

STORY REPRESENTATION IN JAVA CLASS (CONTINUATION)

- *OfficeAdminViewCalendar.java should look like this*



OfficeAdminAuthorizationEmptyFields.java
OfficeAdminViewCalendar.java
OfficeAdmin.java

```java
package com.luminous.steps.calendar;

import org.jbehave.core.annotations.Given;
import org.jbehave.core.annotations.Then;
import org.jbehave.core.annotations.When;

import com.luminous.steps.serenity.OfficeAdmin;

import net.thucydides.core.annotations.Steps;

public class OfficeAdminViewCalendar {

    @Steps
    OfficeAdmin admin;

    @Given("user is on Home Page")
    public void user_is_on_home_page() {
        admin.is_on_home_page();
    }

    @When("user is logged in with username '$username' and password '$password'")
    public void user_is_loging_in(String username, String password) {
        admin.fill_form(username, password);
        admin.submit_form();
    }

    @Then("user should see calendar on Landing Page")
    public void user_shoul_see_calendar() {
        admin.should_should_see_calendar();
    }
}
```

HOW TO WRITE TEST SCRIPT. EXAMPLE

STORY REPRESENTATION IN JAVA CLASS (CONTINUATION)

- Each step in appropriate scenario should be placed under correct section:
  - in calendar.story
    - **Given user is on Home Page**
  - in OfficeAdminViewCalendar.java

```
@Given("user is on Home Page")
    public void user_is_on_home_page() {
        admin.is_on_home_page(); }
```

HOW TO WRITE TEST SCRIPT. EXAMPLE

USER STEPS

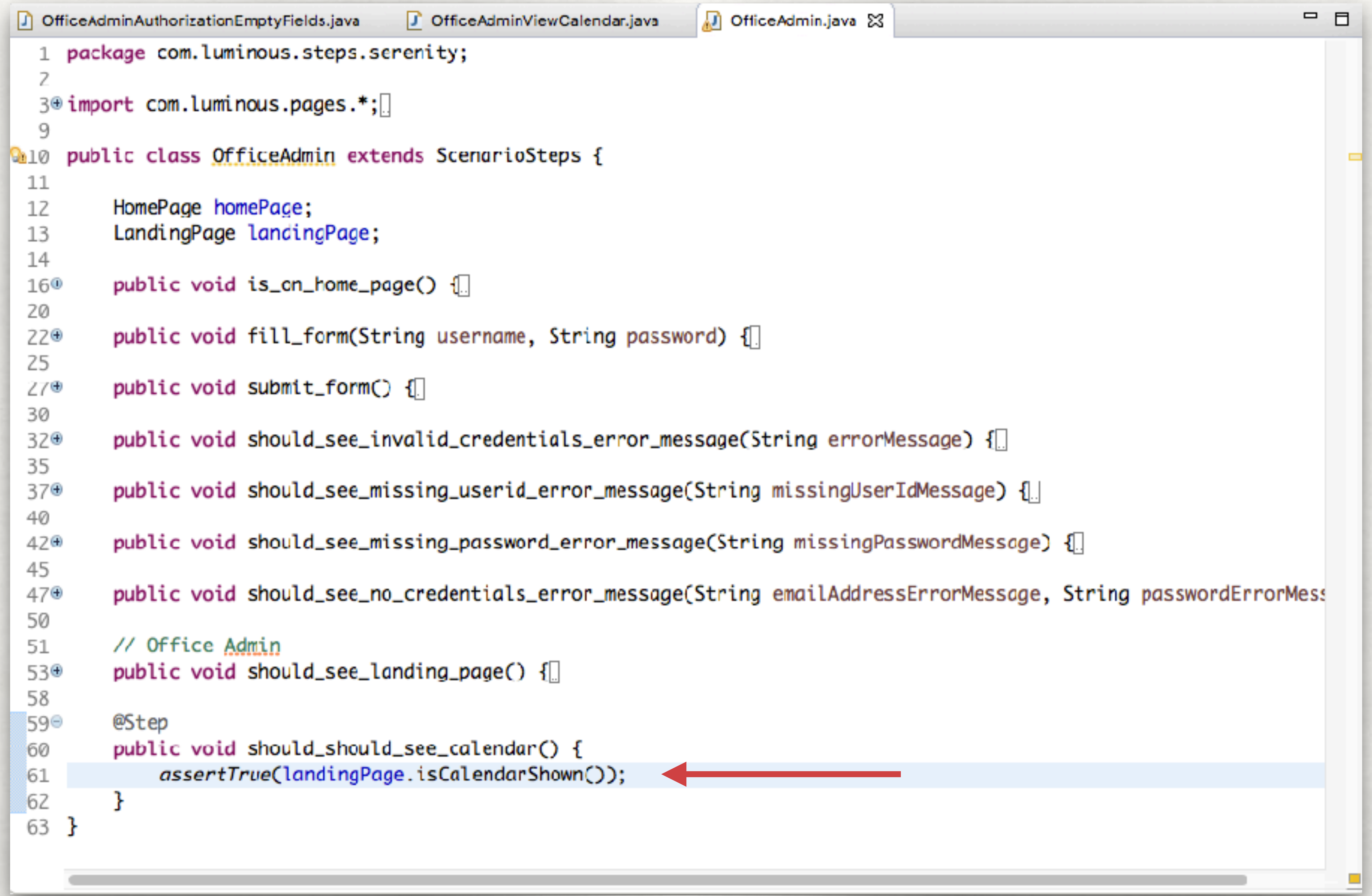
- Each scenario step is executed by particular user. So user step should be implemented in user class. For Office Admin this class is called OfficeAdmin.java

trunk/src/test/java/com/luminous/steps/serenity/OfficeAdmin.java

- should_should_see_calendar() method for our scenario is implemented in this class and calls JUnit method assertTrue() (please, see the next slide)

HOW TO WRITE TEST SCRIPT. EXAMPLE

USER STEPS (CONTINUATION)



OfficeAdminAuthorizationEmptyFields.java
OfficeAdminViewCalendar.java
OfficeAdmin.java

```java
package com.luminous.steps.serenity;

import com.luminous.pages.*;

public class OfficeAdmin extends ScenarioSteps {

    HomePage homePage;
    LandingPage landingPage;

    public void is_on_home_page() {

    public void fill_form(String username, String password) {

    public void submit_form() {

    public void should_see_invalid_credentials_error_message(String errorMessage) {

    public void should_see_missing_userid_error_message(String missingUserIdMessage) {

    public void should_see_missing_password_error_message(String missingPasswordMessage) {

    public void should_see_no_credentials_error_message(String emailAddressErrorMessage, String passwordErrorMess

    // Office Admin
    public void should_see_landing_page() {

    @Step
    public void should_should_see_calendar() {
        assertTrue(landingPage.isCalendarShown());
    }
}
```

HOW TO WRITE TEST SCRIPT. EXAMPLE

PAGE CLASS

- Each page class contains set of actions that can be performed on this particular page by any user. Page class also contains locators of elements that can be displayed on this page
- On previous slide isCalendarShown() method is called from Landing Page.
- Landing Page class contains locator to calendar element

```
@FindBy(id = “calendar") WebElement calendar;
```

and isCalendarShown() method implementation

```
public boolean isCalendarShown() {
    return calendar.isDisplayed(); }
```

- isDisplayed() is Selenium WebDriver method, that finds element by locator and check if it is displayed on the web page

HOW TO WRITE TEST SCRIPT. EXAMPLE

HOW IT WORKS

Let’s summarize:

- When you type mvn clean verify into Command Prompt, serenity-maven-plugin parses calendar.story file
- serenity-maven-plugin reads each scenario step in calendar.story and calls appropriate piece of java code in OfficeAdminViewCalendar class
- To execute methods in @Given, @When and @Then, OfficeAdmin class methods are called. Because all methods that can Office Admin perform, are stored in this class
- To perform Office Admin steps, first HomePage class methods are called to execute authorization. Then LandingPage class methods are called to find calendar by Selenium method, and execute verification method by JUnit
- Depending on calendar visibility JUnit will mark scenario as failed or passed. When all scenarios are executed, Serenity will receive results from JUnit and aggregate test execution report

HOW TO WRITE TEST SCRIPT

RUNNING TEST SCRIPTS MARKED WITH @TAG

Simply add 'Meta' key word and @your_label after scenario description as following:

Scenario: Consultant can perform log in and log out

Meta: @undertest

Given user is logged with username 'consultant@100.com' and valid password 'aaaaaa' and is on Landing Page

When user click logout link

Then user should see Home Page

and run **mvn clean verify -Dmetafilter="+undertest"**

Only this one scenario will be executed.

'+' means add scenario that is marked as 'undertest' to the test execution

Label name should not contain spaces and can be any word. You may use underscore in label name, e.g. my_label

HOW TO WRITE TEST SCRIPT

RUNNING TEST SCRIPTS MARKED WITH @TAG

You can mark as many scenarios as you want and run them all. For example:

Scenario: Consultant can perform log in and log out

Meta: @tested

Given user is logged with username 'consultant@100.com' and valid password 'aaaaaa' and is on Landing Page

When user click logout link

Then user should see Home Page

Scenario: Consultant is not able to access Landing Page by URL if log out was performed

Meta: @donebyme

Given user is logged with username 'consultant@100.com' and password 'aaaaaa' and is on Landing Page

When user copies page url

And user performs log out

And user tries to access Landing Page by URL

Then user should see Session Expired Page

run **mvn clean verify -Dmetafilter="+undertest +donebyme"**

Only these two scenarios will be executed.

HOW TO WRITE TEST SCRIPT

RUNNING TEST SCRIPTS MARKED WITH @TAG

It is also possible to mark whole story by label

Meta: @logout

Narrative:

As a Consultant, Client, Office Admin

I want to be able to see logout link and be able to logout

Scenario: Consultant can perform log in and log out

Given user is logged with username 'consultant@100.com' and valid password 'aaaaaa' and is on Landing Page

When user click logout link

Then user should see Home Page

Scenario: Consultant is not able to access Landing Page by URL if log out was performed

Given user is logged with username 'consultant@100.com' and password 'aaaaaa' and is on Landing Page

When user copies page url

And user performs log out

And user tries to access Landing Page by URL

Then user should see Session Expired Page

and run **mvn clean verify -Dmetafilter="+logout"**

Only scenarios that belong to this story will be executed

HOW TO WRITE TEST SCRIPT

RUNNING TEST SCRIPTS MARKED WITH @TAG

You may need the following command

```
mvn clean verify -Dmetafilter="-logout"
```

to exclude some labeled scenarios or stories from the test execution.

With this command all scenarios or stories that you marked as @logout will be excluded from test execution

COMMIT YOUR TEST SCRIPTS

TO SVN REPOSITORY

- Use Tortoise SVN to commit your scripts
- Before commit make sure that implemented new scenarios are executed correctly and project build has status “success”
- Before committing test code add comments about changes you did and test scenarios you are about to commit

RESOURCES

USEFUL LINKS

Maven

- https://en.wikipedia.org/wiki/Apache_Maven
- https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Selenium WebDriver

- http://www.seleniumhq.org/projects/webdriver/
- https://en.wikipedia.org/wiki/Selenium_(software)

Serenity BDD

- **http://www.thucydides.info/#/**
- **https://dzone.com/articles/introduction-bdd-test-1**

JUnit

- https://en.wikipedia.org/wiki/JUnit
- http://www.vogella.com/tutorials/JUnit/article.html#usingjuni4

JBehave

- http://jbehave.org/
- https://en.wikipedia.org/wiki/Behavior-driven_development

RESOURCES

ONLINE COURSES

Java: Testing with JUnit

- https://www.linkedin.com/learning/using-junit-for-testing-in-java

Introduction to Programming with Java - Part 1: Starting to Program in Java

- https://courses.edx.org/courses/UC3Mx/IT.1.1x/1T2015/course/