**Introduction: -**

This project is all about getting industrial way to test applications either functionally or through automation. This explains, what is testing, what is need for testing and what are the different ways to do testing. It describes everything about testing such as types of testing, defects, functional test cases, need for automation, tools and technologies used in automation, test execution report generation etc.

**Software Testing -**

**Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps, or missing requirements in contrast to actual requirements.

**Why Software Testing is Important?**

**Software Testing is Important** because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction.

## What are the benefits of Software Testing?

Here are the benefits of using software testing:

* **Cost-Effective:**It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
* **Security:**It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
* **Product quality:**It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
* **Customer Satisfaction:**The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

## Testing in Software Engineering -

As per ANSI/IEEE 1059, **Testing in Software Engineering** is a process of evaluating a software product to find whether the current software product meets the required conditions or not. The testing process involves evaluating the features of the software product for requirements in terms of any missing requirements, bugs or errors, security, reliability, and performance.

**Static Testing: -**

Analysis of code, documentation without executing the program. Defects identified in static testing are less expensive to fix. It involves both developers and testers.

1. **Informal Reviews: -** Reviewing the documents and providing informal comments.
2. **Technical Reviews: -** Technical specification document such as test plan.
3. **Walkthrough: -** Explanation of a product by the developer to the whole team.
4. **Inspection: -** Formal reviews to find out defect. Reviewers find out the defects and inform the development team.
5. **Static Code Review: -** Code syntax and optimization.

**Dynamic Testing: -**

Execution of software and validating the output with expected result. Black & White box testing. It can be performed all the stages.

|  |  |
| --- | --- |
| **STATIC TESTING** | **DYNAMIC TESTING** |
| Without executing code | With executing code |
| Verification process | Validation process |
| Prevention of defects | Finding and fixing of defects |
| Before compilation | After compilation |
| Cost of defect is less | Cost of defect is high |
| ROI is very high at it occurs at early stage | ROI is low at it occurs at late stage |

**Black Box Testing: -**

Examines the application functionality and ignores the internal mechanism of system and concentrates on the generated output. Tester has no access of source code. The tester is conscious about “What the software is supposed to do” but not aware of “how it does it”.

**While Box Testing: -**

Analyze the internal structure and working of a program. The tester is required to have programming skills and knowledge of internal structure of a program.

**Unit Testing: -**

Done by developers. It comes under white box testing. Development of stubs and drivers. It separates each part of a program and verify proper functionality of individual modules. **Ex: - Calling functionality of google duo.**

**Integration Testing: -**

It tests the functionalities of various modules of an application combined together. It mostly used in distributed systems. It comes under white and black box testing. It helps in identifying defects in interfaces. Two types of integration testing: - Top-Down and Bottom-Up. **Ex: - Sign Up of Instagram using Facebook/Gmail.**

**Functional Testing: -**

It done on complete and integrated system to ensure that specified functionality required in the system works. It comes under black box testing. Focuses on testing the application against SRS, Test cases. It tests the functionality of each and every section of application. **Ex: - Message sending functionality of WhatsApp. It should be sent to the correct recipient without any dropping of messages etc.**

**System Testing: -**

It done on complete and integrated system. It comes under black box testing. Implemented by dedicated testing team which ensures that application meets functional and technical specification along with quality standards. Test environment is similar to production environment. **Ex: - How system is performing after giving various types of inputs in different conditions.**

**Smoke Testing: -**

Initial build of software to ensure that crucial functionalities of the system working fine. The main objective is to ensure that the major functionalities are working, and build is stable to go for functional testing. **Ex: - After deployment of any web application check whether URL is working fine or not before doing functional testing.**

**Sanity Testing: -**

It performs to ensure defects have been fixed and no further defects have been introduced due to minor changes in code. It is a subset of Regression testing. The main objective is to do quick checkup of software when defects are fixed before started detailed Regression Testing.

**Regression Testing: -**

There are various ways for introduction of defects in application.

1. Defects has been introduced when a new functionality interact with existing one.
2. Fixing of defect in application may introduced more defects.

It mainly focuses on detecting such defect and ensure application is functioning properly. It provides repetitive and consistence validation of every new product released. Automation is preferred during regression. **Ex: - After deployment of v1.1 on test environment check v1.0 is working as expected or not.**

**Acceptance Testing: -**

It ensures the software has met the required and performs the way customer expect in production. It helps in defect spelling mistake, cosmetic errors.

1. **Alpha Testing: -** Performed by an independent team within the company. Build confidence before goes for Beta Testing.
2. **Beta Testing: -** Performed by end users. It detects unexpected errors.

This testing helps in detecting real life issues in the product. Fixing these issues before releasing would improve quality of production and customer satisfaction.

**Performance Testing: -**

Non-functional requirements like response time, throughput is all accordance with SLA (Service Level Agreement).

1. **Load Testing: -** Ability of an application to work under a particular user load. It helps in identifying bottleneck in application.
2. **Stress Testing: -** Increase the load till it reaches to the saturation point called breakpoint. It ensures application handled load spikes when it goes live.
3. **Endurance Testing: -** Expected load over a long period of time. It helps in uncovering bottleneck like memory leak etc.
4. **Volume Testing: -** Large amount of data populated in DB. It helps in estimates impact of data on application such as response time and DB strength.

**Software Testing Life Cycle: -**

**Requirement Capture and Analysis**

**Test Cycle Closure**

**Test Closure**

Re

**Test Planning and Design**

**Test Execution**

1. **Requirement Capture and Analysis: -** It is the most significant stage in STLC. During this phase the QA team interacts with different stakeholders, developers, client etc. and tries to understand requirement from testing point of view. They identified testable requirements.
2. **Test Planning and Design: -** Test strategy for complete testing process. Cost and Efforts estimated. It specifies scope, approach, resources, roles and responsibilities of various testers.

**Test Design: -** Test cases/scripts ae designed, reviewed and reworked. Identification of test data and RTM Preparation also done.

1. **Test Execution: -** Test team test the readiness of environment which was set up in test design phase. Testers also execute the test cases as per test plan. The defects are reported back to development team. Once the defects are fixed the system is tested again to ensure software is defect free.
2. **Test Cycle Closure: -** The testing team meets and evaluate criteria for cycle completion based on time, scope, coverage, cost, quality etc.
3. **Test Closure: -** The testing process for the project is evaluated and lessons learnt from the process are documented. The process that are implemented in future is identified.

**Software Defect: -**

A software defect is an error or fault in software that leads to an unexpected behavior of system. The defects could be caused due to time pressure, miscommunication of requirements, Coding skills etc.

1. **Severity: -** It is the extent to which a defect can affect the software.
2. **Priority: -** The urgency with which defect should be addressed.

**Urgent** **Low**

**Critical** Key feature does not work Feature that is rarely used does not work

**Non-Critical** Company logo in wrong color Image caption is in wrong format

1. **New: -** Logged defect for the first time.
2. **Assigned: -** When defect is assigned to the development team to fix.
3. **Open: -** Developer started working on it.
4. **Fixed: -** Developer make code change to rectify defect.
5. **Retest: -** The tester rechecks the defect whether it has been fixed or not.
6. **Deferred: -** When the defect is expected to be fixed in upcoming releases.
7. **Duplicate: -** When 2 defects points to the same functionality.
8. **Rejected: -** If defect is not genuine then developer reject the defect.
9. **Reopen: -** If defect is not fixed then it re-opens.
10. **Closed: -** Defect does not exist in the software then status made as closed.

**Test Data: -**

1. **Static Data: -** It is permanent data and related to DB. The information that changes little over time. Ex: - Country
2. **Master Data: -** Rarely updated but often read. Ex: - Name/email
3. **Configurable Data: -** Data which drives the application. Ex: - User Credentials
4. **Transactional Data: -** Business transactions in Application Under Test. Ex: - Order ID.

**Challenges in Software Testing: -**

1. Difficult to test an entire application completely.
2. Regression Testing.
3. Lack of skilled testers.
4. Time constraint.
5. Understanding the requirement.
6. Which testes to execute first.

**Entry Criteria: -**

1. Business requirement should be clear and available.
2. Good understanding of requirement must be done.
3. Test environment must be up and stable for detailed testing.
4. Proper sign-off must be received for test cases from BA before starting detailed testing.

**Exit Criteria: -**

1. Acceptance test should be executed, and all the tests should be pass.
2. No critical/major defects left open. All the defects should be fixed and verified.
3. Test cases should be signed off.
4. Go/No-Go decision for the product.

**Login Page Test Cases: -**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case Scenario** | **Test Case ID** | **Test Case Description** | **Pre-Requisite** | **Steps** | **Expected Result** |
| Login Functionality of Facebook Page | **TC\_Login\_001** | User should be able to Login after valid credentials | User must present on Login Page | Enter valid USERID on Username field | USERID should be entered |
|  |  |  |  | Enter valid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | User must be on Homepage. |
|  | **TC\_Login\_002** | User should not be able to Login after entering in- valid credentials | User must present on Login Page | Enter invalid USERID on Username field | USERID should be entered |
|  |  |  |  | Enter invalid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | User should get alert as “Invalid User ID or Password” |
|  | **TC\_Login\_003** | User should get Single Sign On (SSO) Error | User’s previous session must be ON anywhere. | Enter valid USERID on Username field | USERID should be entered |
|  |  |  | User must present on Login Page | Enter valid PASSWORD on Password field | PASSWORD should be entered |
|  |  |  |  | Click on Login Button | Due to previous ON session. User must get SSO Error as “Invalid Session due to Single Sign on Policy”. |

**Automation Testing Life Cycle:** -

**Requirement Gathering and Automation Feasibility Analysis**

**Execution and Result Analysis**

**Scripting and Dry Run**

**Test Strategy and Design**

**Automation Life Cycle**

1. **Requirement Gathering and Automation Feasibility Analysis: -**
2. Understand the project requirement
3. Baseline the requirement
4. Automation Feasibility Analysis (as seen above)
5. Appropriate test automation tool

To finalize AFA we need to think on below factors also: -

1. Usage of module
2. Object Identification
3. Scope for automation
4. **Test Strategy and Design:** -
5. Prepare strategy document
6. Identify most suitable test automation framework
7. Setting up test environment

The automation framework combines coding standards, best practices, reporting mechanism. Due to this scripting will become more easy, efficient and less redundant.

1. **Scripting and Dry Run: -**
2. Create test scripts and associate them into library
3. Associate test script with test data
4. Perform dry run
5. Prepare execution plan
6. **Execution and Result Analysis: -**
7. Execute scripts and verify result
8. Log the defect if necessary
9. Rerun the scripts after fixing the defect
10. Share results with necessary stakeholders

**Advantages of Automation: -**

1. Very useful if set of test needs to execute repeatedly.
2. Cross browser testing is possible.
3. We can also run-on different machine simultaneously.

**Disadvantages of Automation: -**

1. Initial cost of creating automation framework and scripts is high.
2. Licensed tools are very expensive although they provide good support.
3. Skilled testers are required, and visual reference is not possible.

**Selenium** is an open source. There is no license overhead. Unlike other automation tools which need specific programming languages to create script. **Selenium** support 10 different language. It can easily integrate with Maven, Jenkins etc.

**Selenium WebDriver: -**

It can automate testing in all major browsers currently available in market. It can be integrated with other open-source tools like Cucumber, Jira, Jenkins etc.

1. You can run test on any of browser.
2. You can create test using any programming language.
3. You can also interact with complex UI code.

**Junit Framework: -**

It is a unit testing framework and a class library that contains definition of reusable classes, annotations, methods which we can use in our automation script. It is the simplest unit testing framework available for java called Junit.

**@Test:** - It tells Junit framework that the method attached to it can be run as a test case.

**@Before:** - It tells the method attached to it must be run once before executing every **@test** method for ex: - Create temporary variables, invoking test environment etc.

**@After:** - It tells that the method attached to it must be run once after executing every **@test** method for ex: - Release all resources etc.

**@BeforeClass: -** It tells that the method attached to it must be run only once before any other method in class is executed for ex: - Opening DB Connection.

**@AfterClass: -** It tells that the method attached to it must be run only once after all other methods in class are executed for ex: - Release resources.

**Log4j: -**

Apache log4j is a java based reliable, fast and flexible logging utility which is used for logging information.

1. It is an open source and can be easily configured in selenium.
2. Using log4j we can store selenium project execution flow in a file.
3. It gives better status of project.

**Components of Log4J: -**

1. **Loggers: -** It is responsible for logging information provide by test script creator. There are 5 kinds of log level such as Fatal/Error/Warn/Info/Debug.
2. **Appendars: -** It is responsible for writing log messages to a file. We can use ConsoleAppender/FileAppender.
3. **Layout: -** It is responsible for formatting logging information in different style.

**Maven Overview: -**

It is a build automation tool for java projects. When a Maven project is created, it creates default project structure and developer is only required to place a file accordingly. It resolves following problems: -

1. Adding set of jar files.
2. Dependencies and Version.

POM in maven stands for Project Object Model. It contains information about the project and various config details used by Maven to build the project. It also contains the goals and plugins.

1. **Local Repository: -** Repository located in user’s local system.
2. **Central Repository: -** It is located on web. It has created by Maven itself.
3. **Remote Repository: -** It is located on web.

**Cucumber – BDD Framework: -**

**BDD** test are written in plain English language. They are driven by business values. BDD frameworks are Cucumber and JBehave.

**BDD (Behavior Driven Development)** is about implementing an application by describing its behavior from the perspective of its stakeholders.

Gherkin is the language that cucumber understands. It is simple, light weight and structured language. It uses **.feature** as file extension. Most lines in Gherkin starts with a special keyword.

Cucumber is a testing framework that supports BDD. It is itself written in RUBY. It can test code written in Ruby, Java or any programming language.

A step definition is a small bit of code with a pattern attached to it. The pattern is used to link the step definition to all matching steps in feature file. Step definition must be placed in a file where cucumber can find them.

**Given, When, Then, As, But, \*, Scenario, Scenario Outline, Example,** Feature used in feature file.

**Corresponding code of feature file**

**Step Definition  
  
We used Selenium + Java + Annotations**

**Test Runner  
Written in Junit (To run feature file and generate reports)**

**Feature File**

**Runner contains details of feature file and other cucumber options are specified as below**

**TestSauceDemo.feature**

**Cucumber Using POM:** -

**Feature File, Step Definition, Runner Class are all related to each other as shown in above diagram**

**Base Class**

dfadsf

**Will call all the java classes in Step Definition file**

**Runner Class**

**Step Definition**

**Feature File**

**All classes extendBASE Class**

**YourInformationPage.java**

**YourCartsPage.java**

**ProductsPage.java**

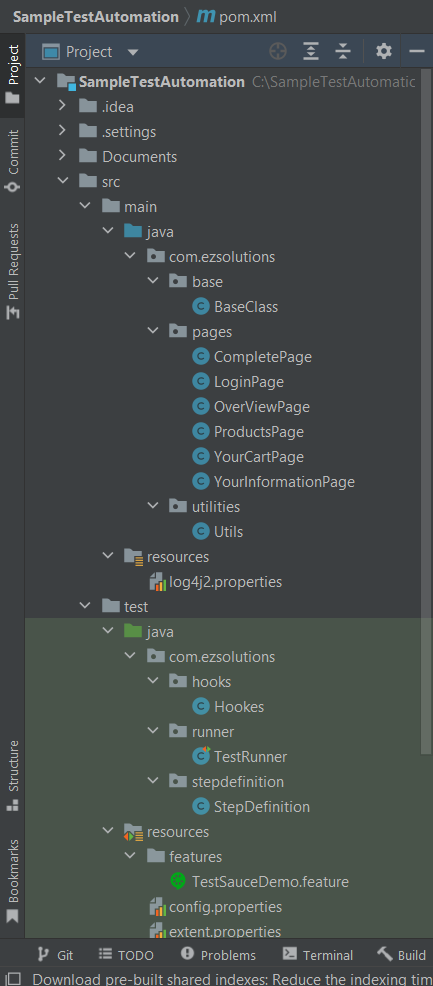
**LoginPage.java**

**BaseClass (Webdriver initialization, load properties file etc.)**

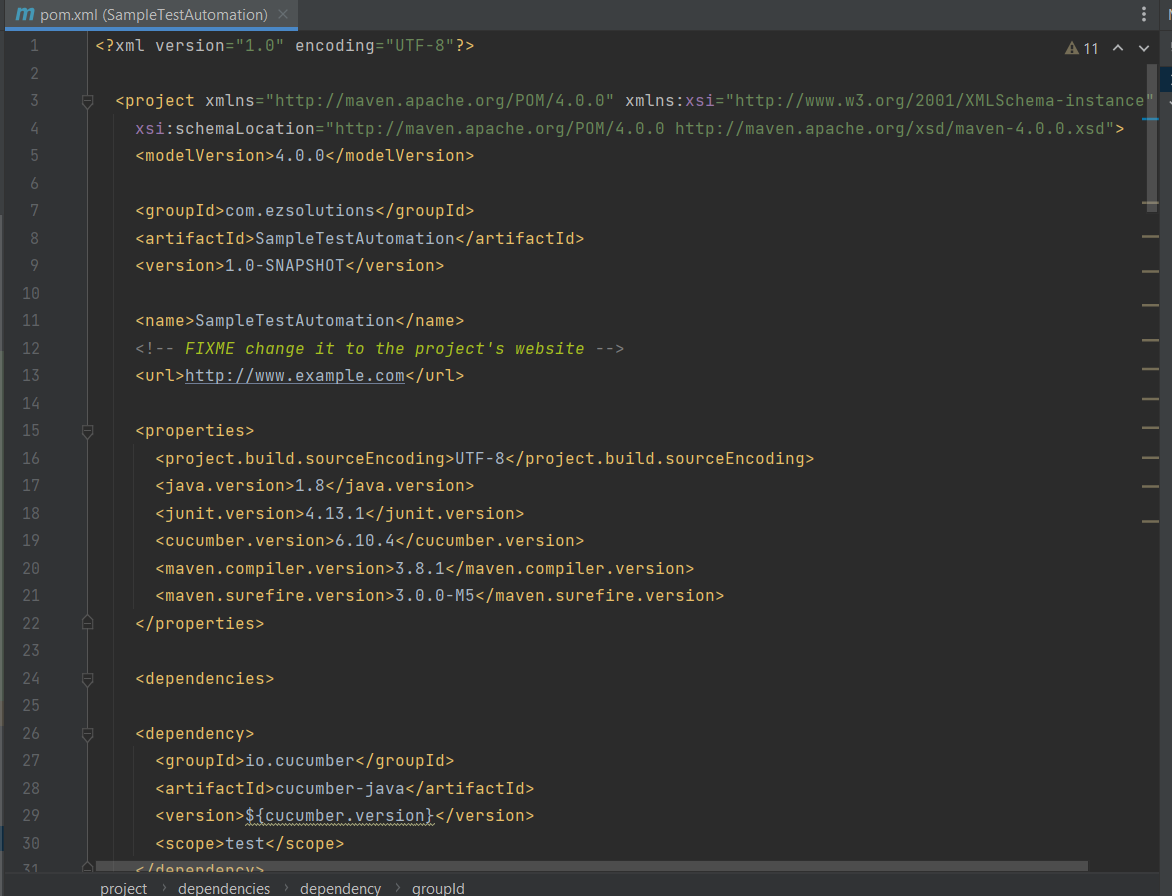
**Demo of Automation Testing –**

* **IDE** - IntelliJ
* **UI Automation Tool** – Selenium
* **Language** – Java
* **Framework Structure** – Maven
* **Test Cases** – Cucumber
* **Reporting** – PDF and Extent HTML Report

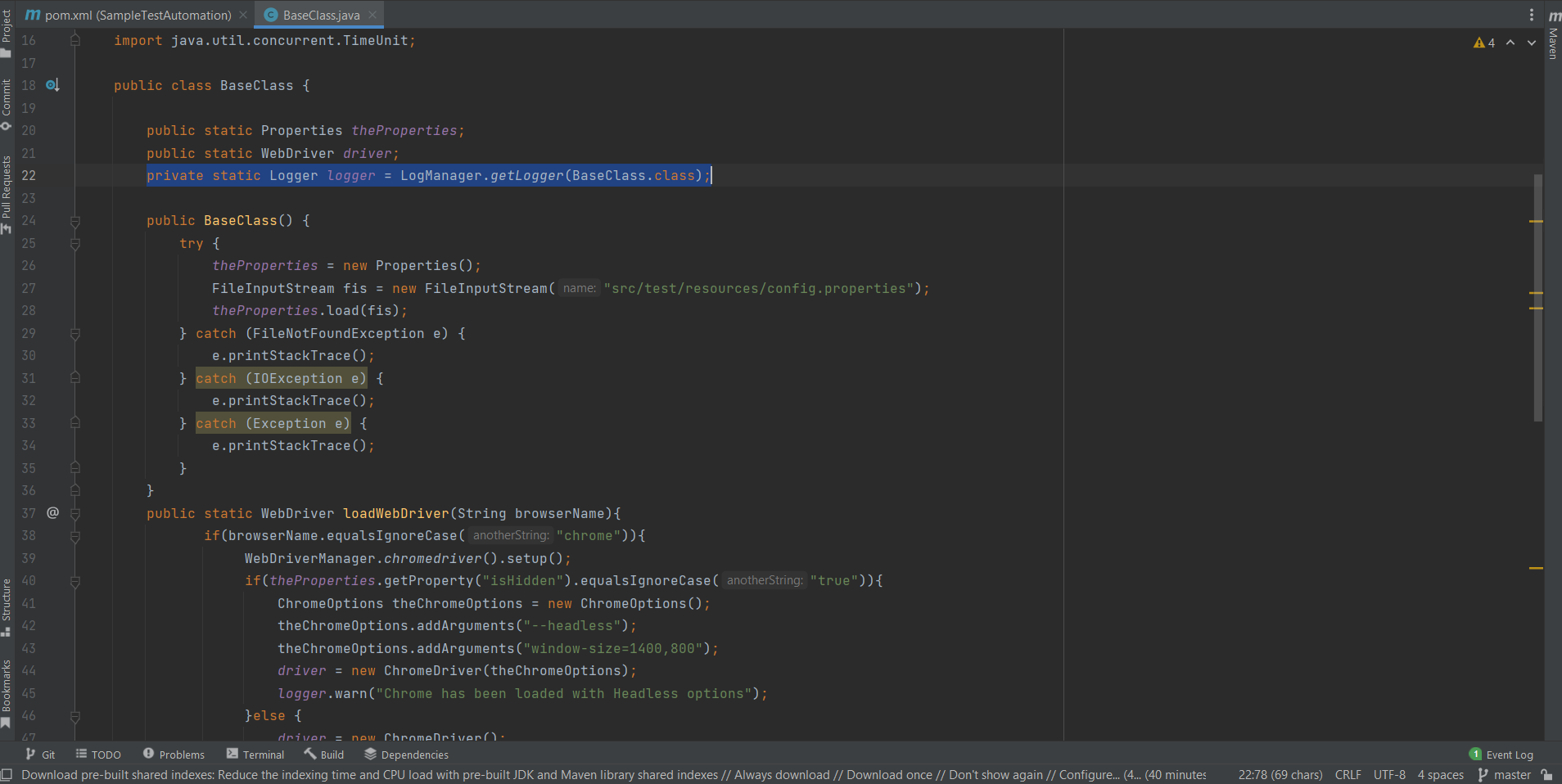
**Framework Structure -**

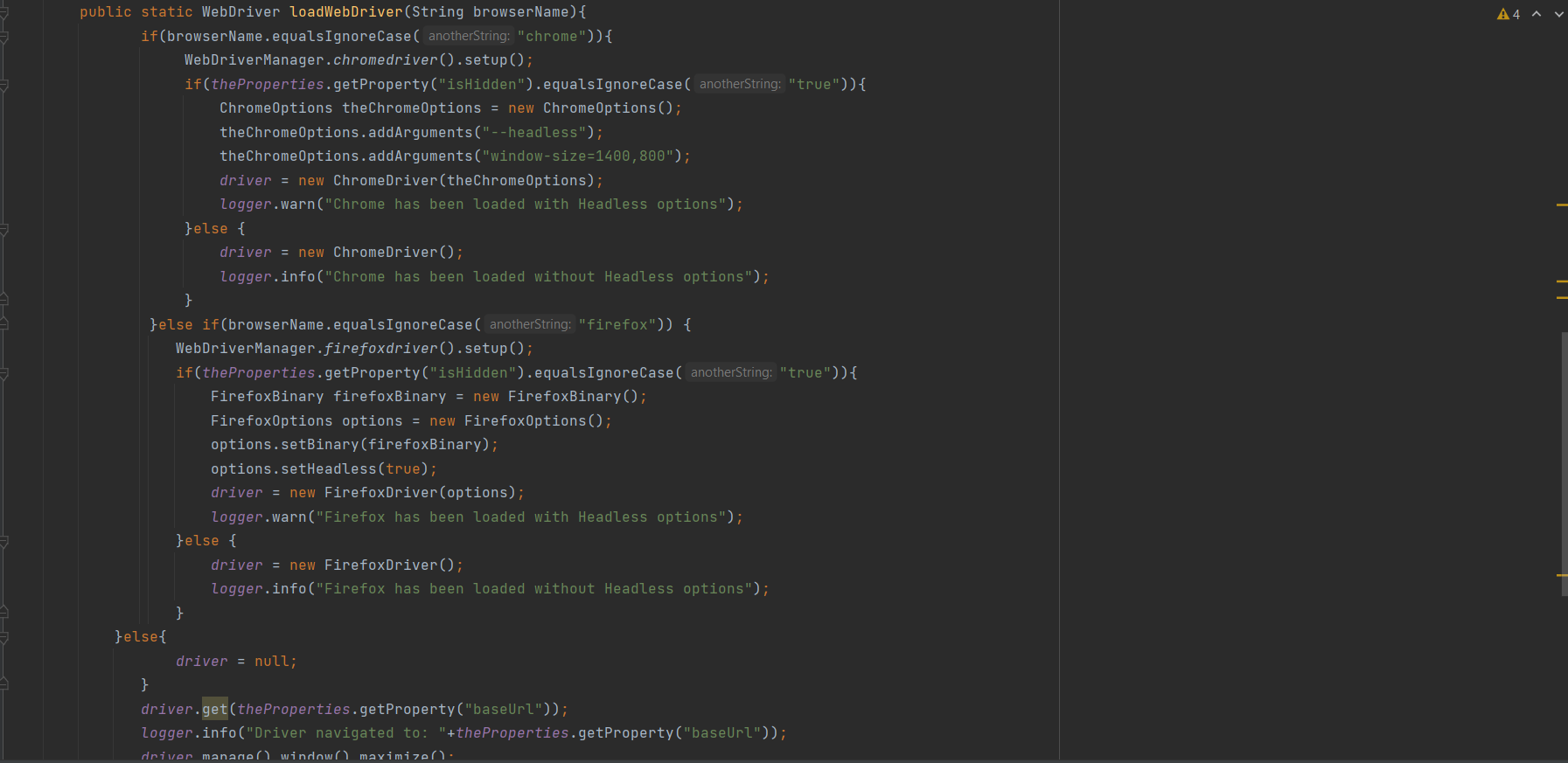


**Pom.xml – (Required Dependencies)**

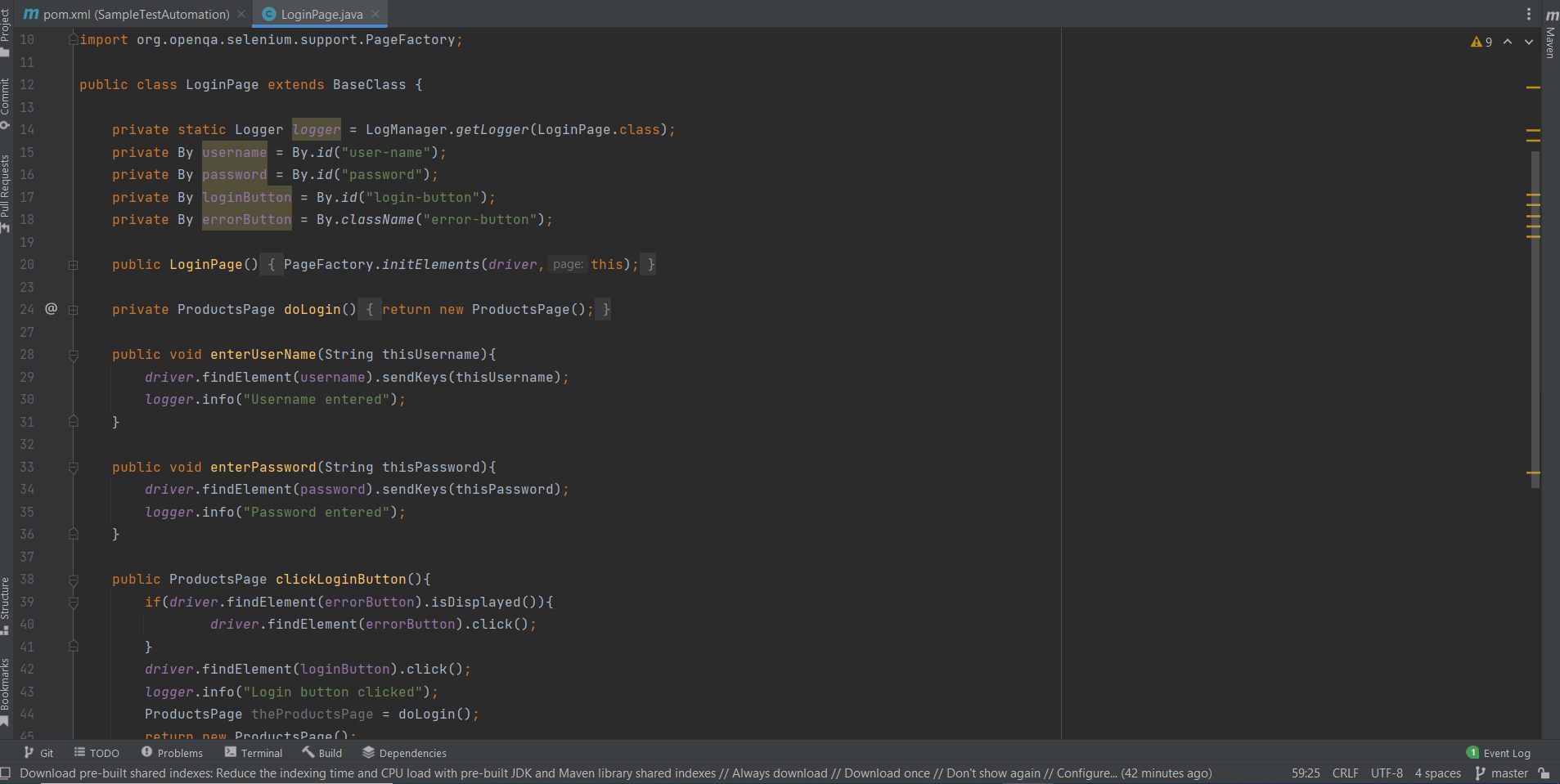


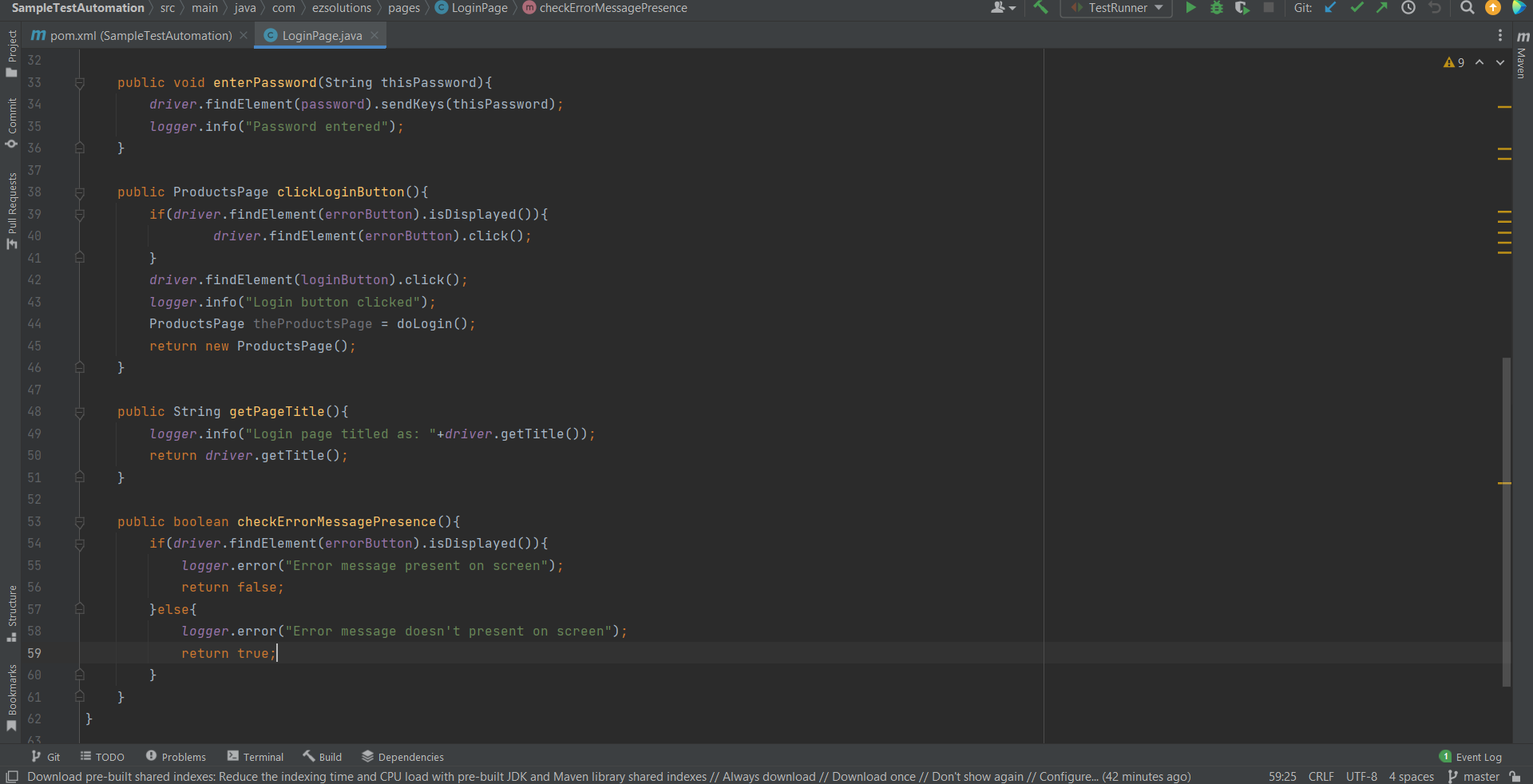
**Baseclass – (Parent class)**



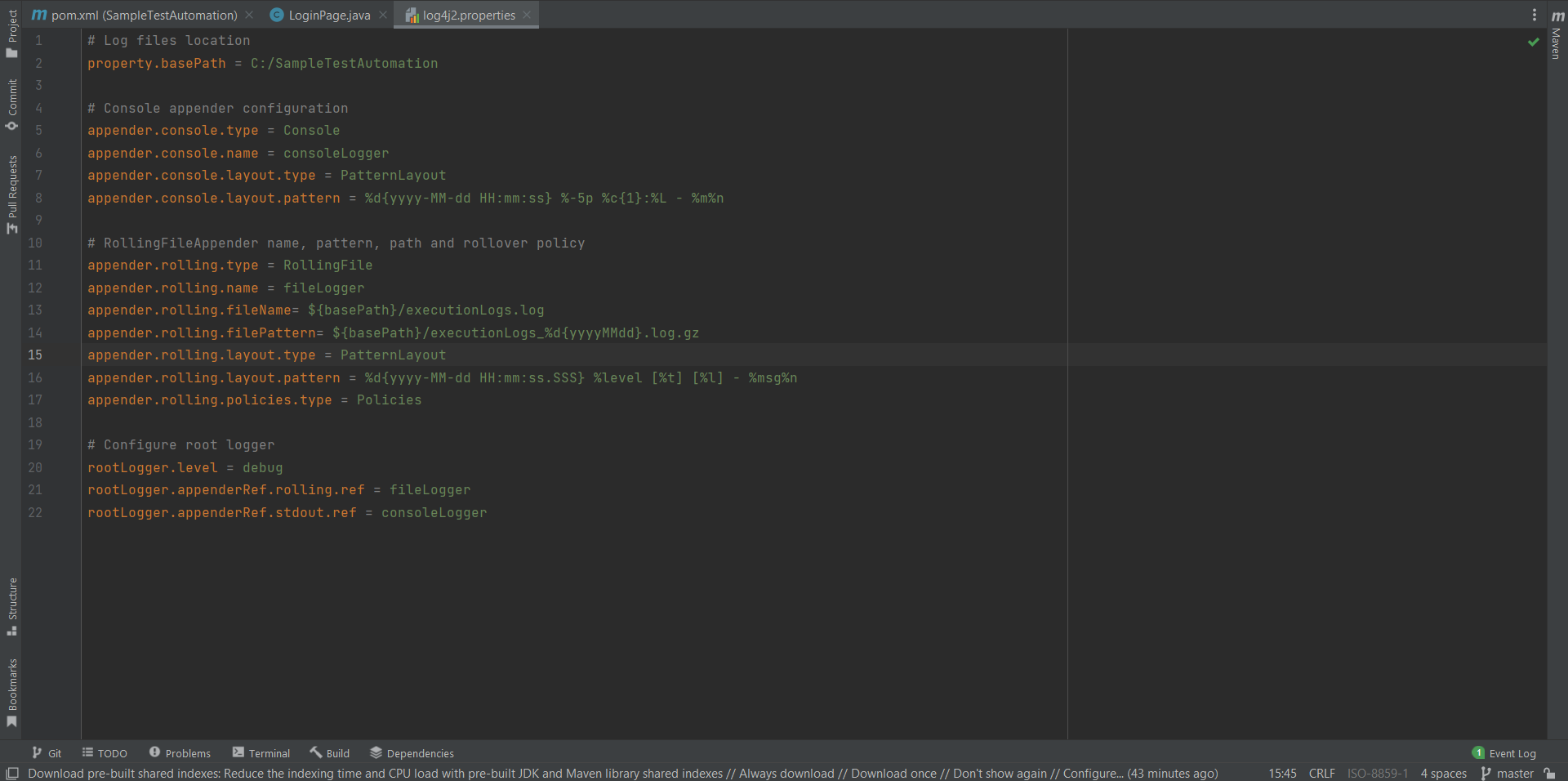


**LoginPage Class –**

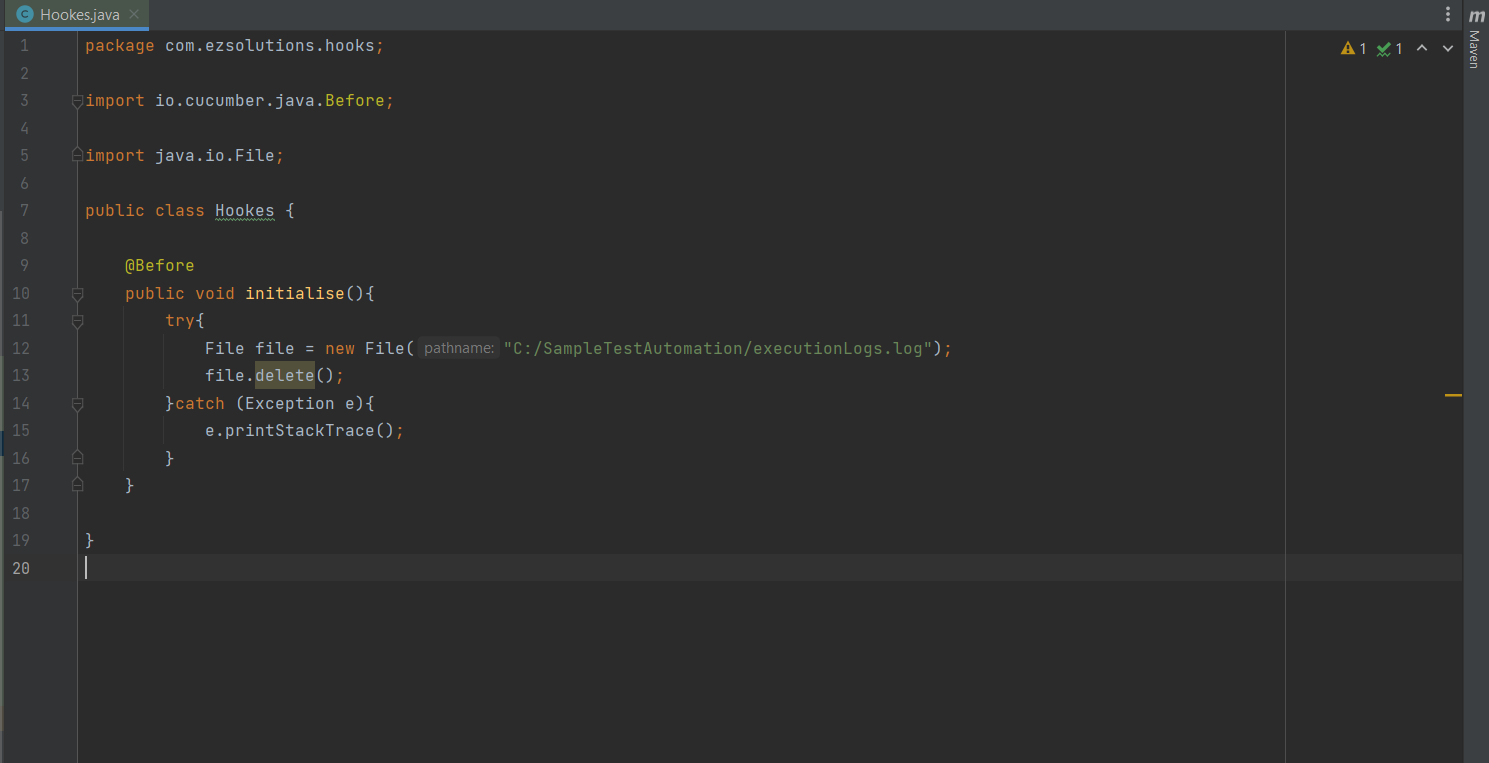




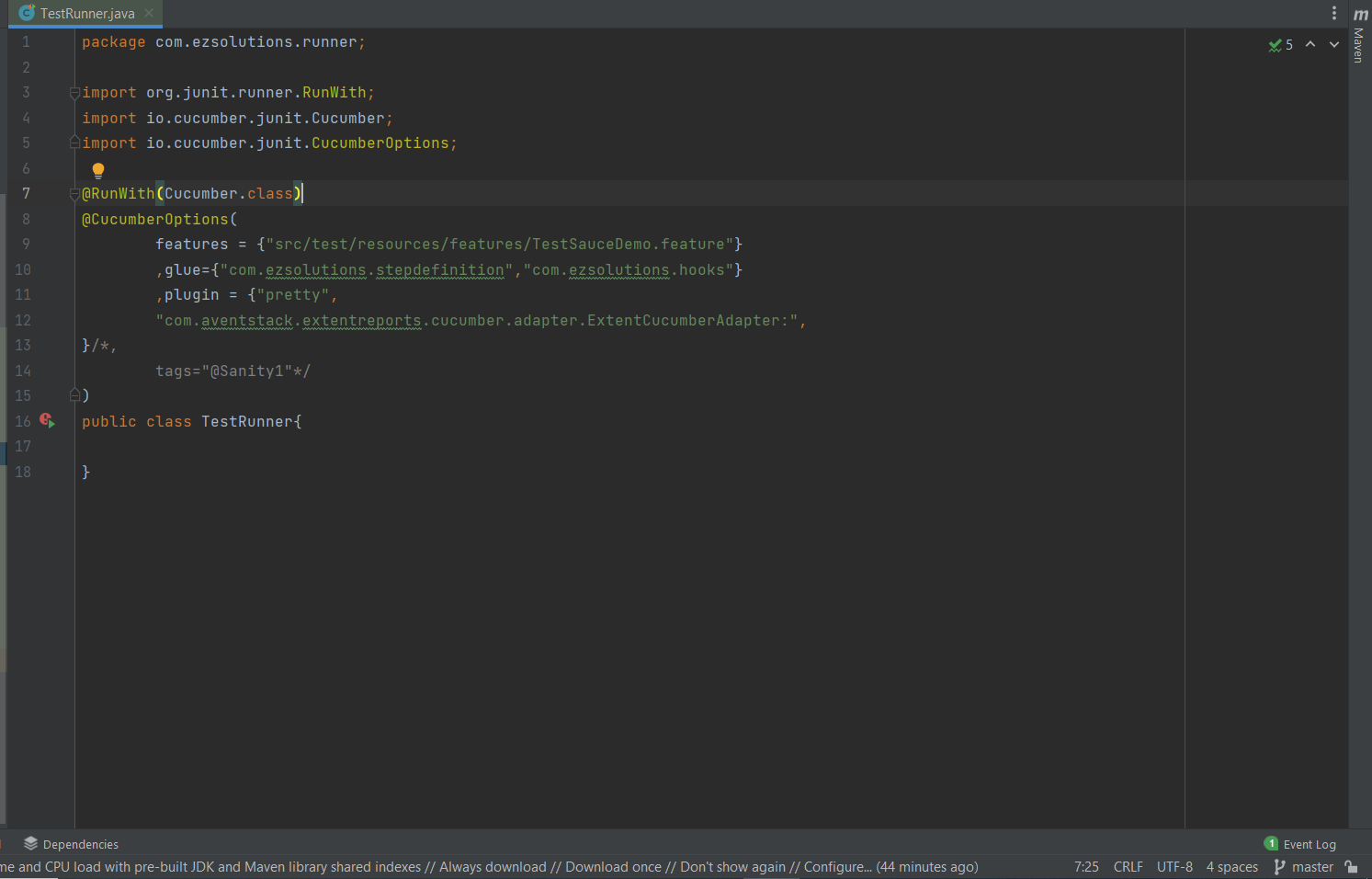
**Log4j properties file –**

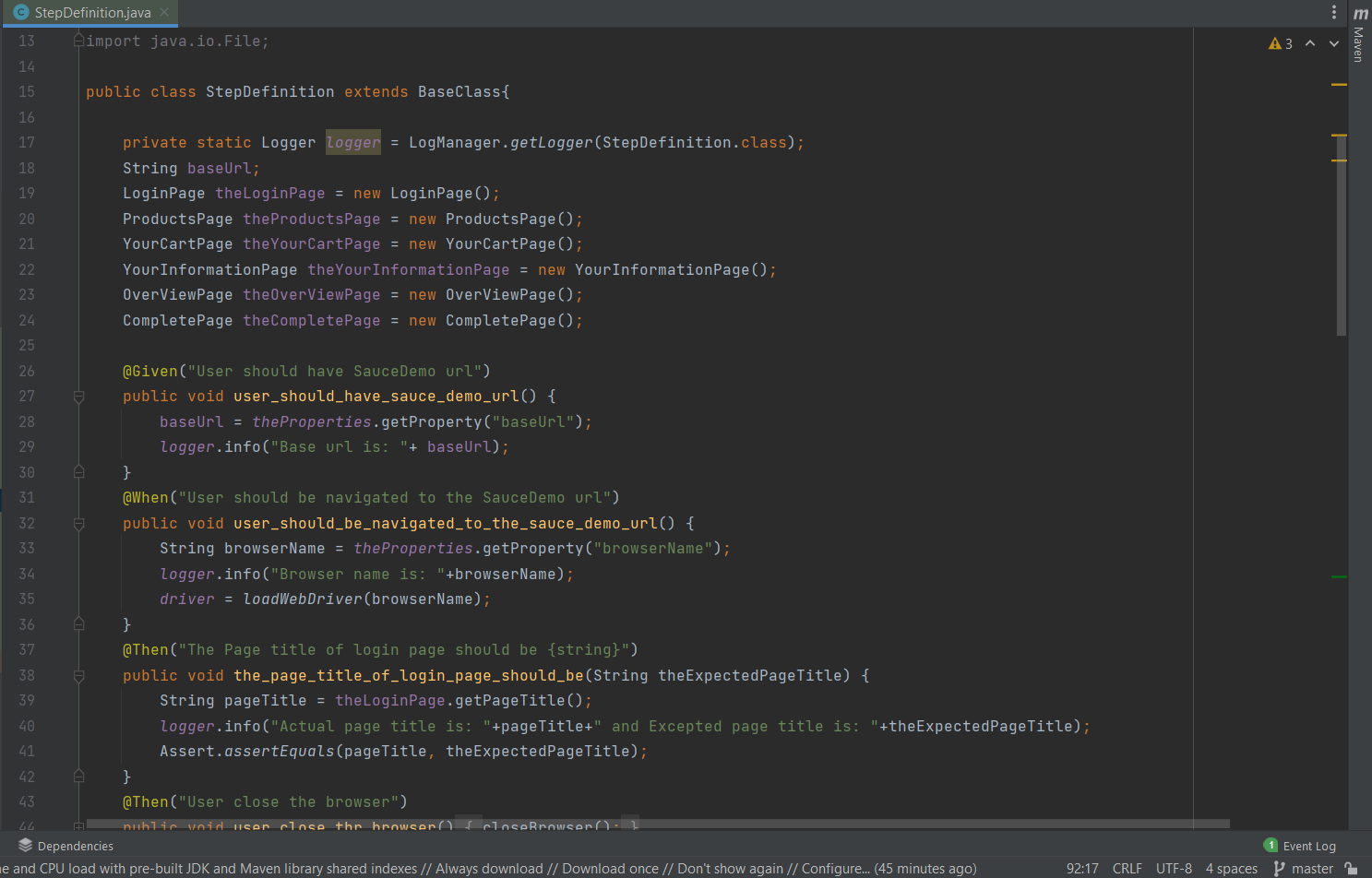
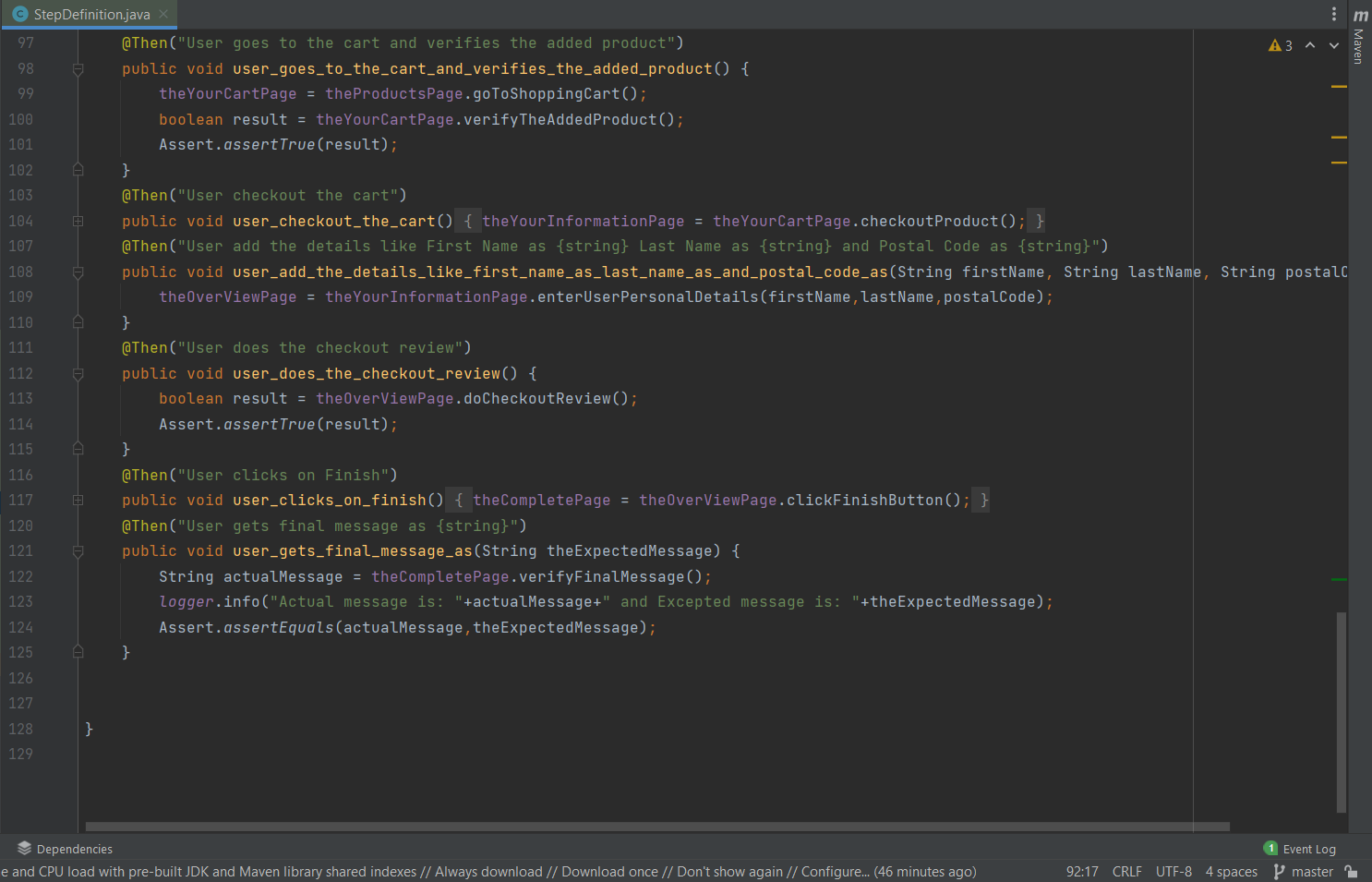


**Cucumber Hooks –**

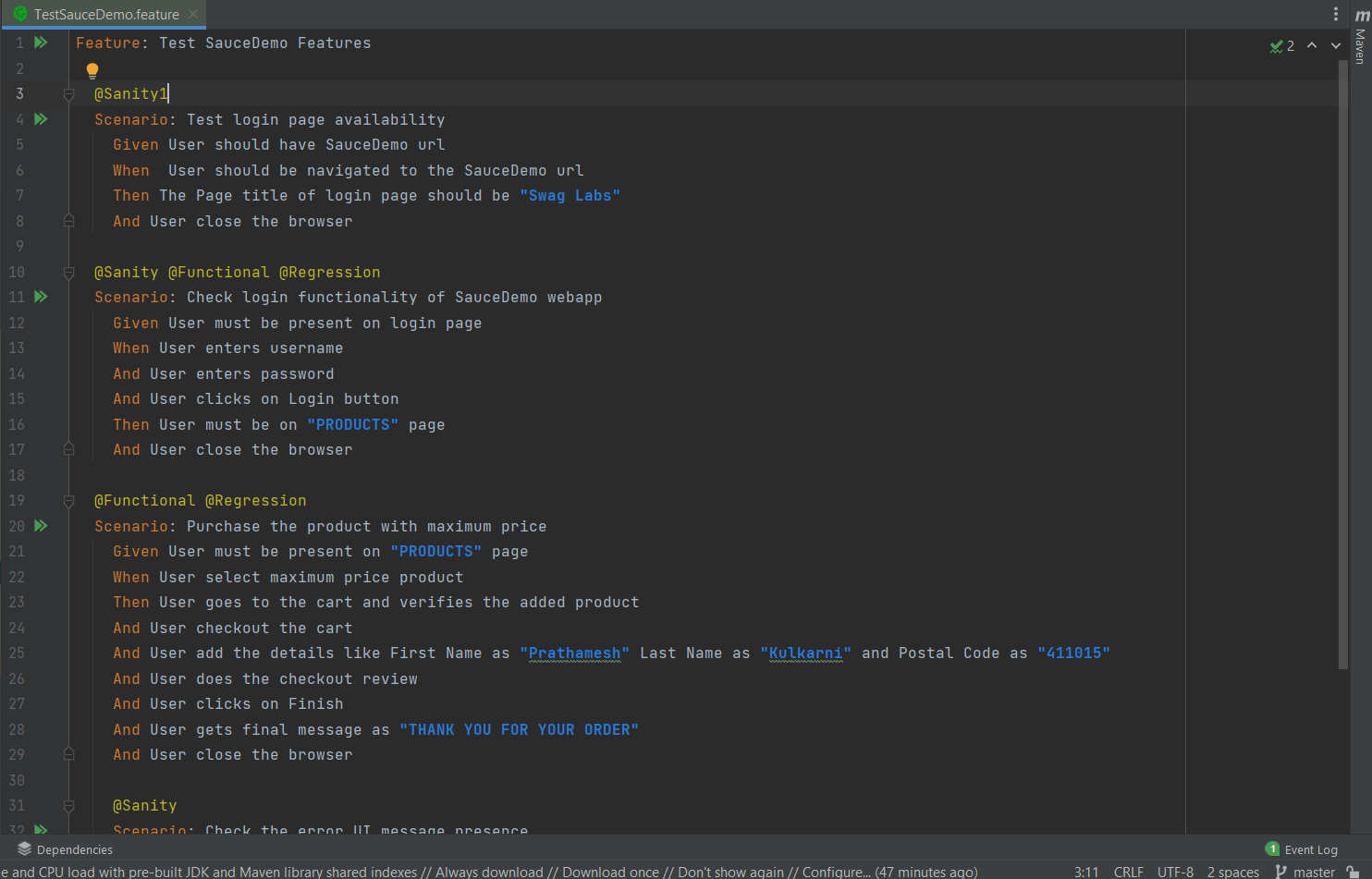


**Cucumber Test Runner –**

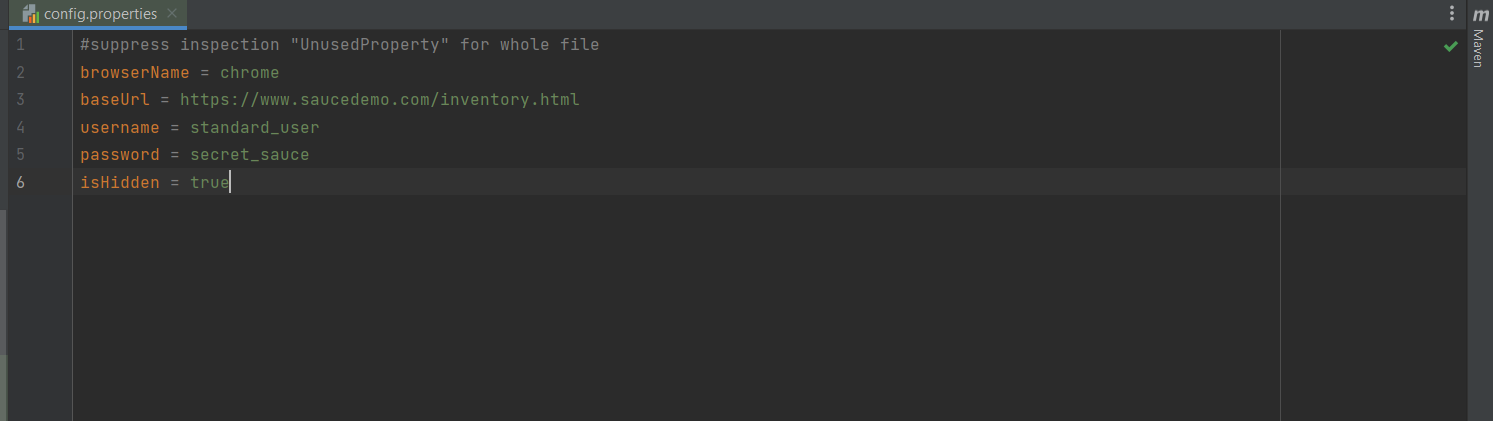


**Cucumber Step Definition –**

**Automation Test Cases –**

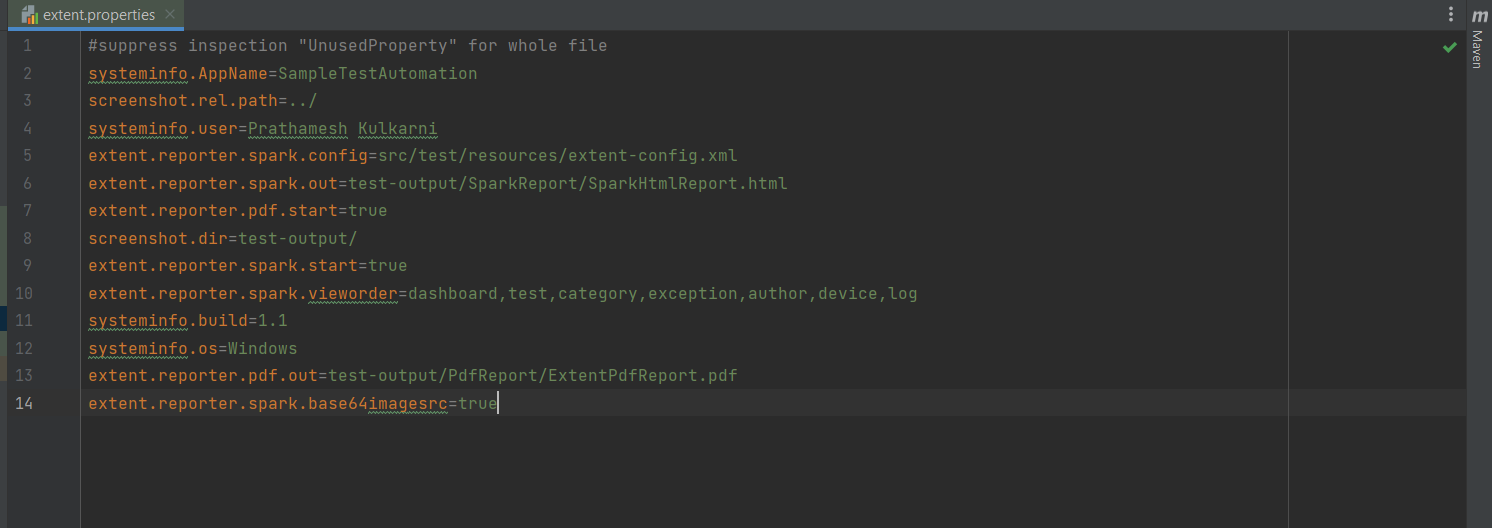


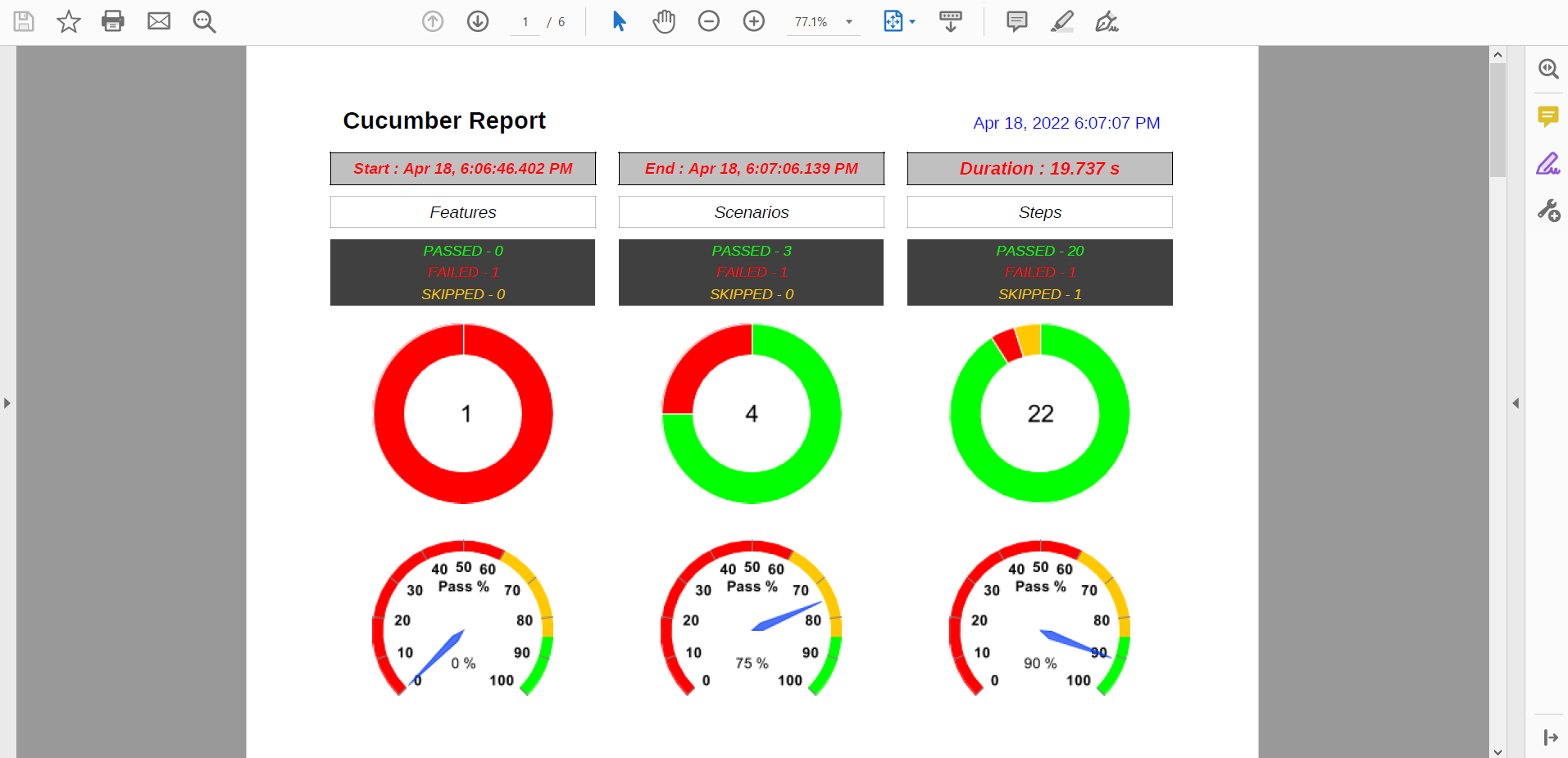
**Project configuration properties –**

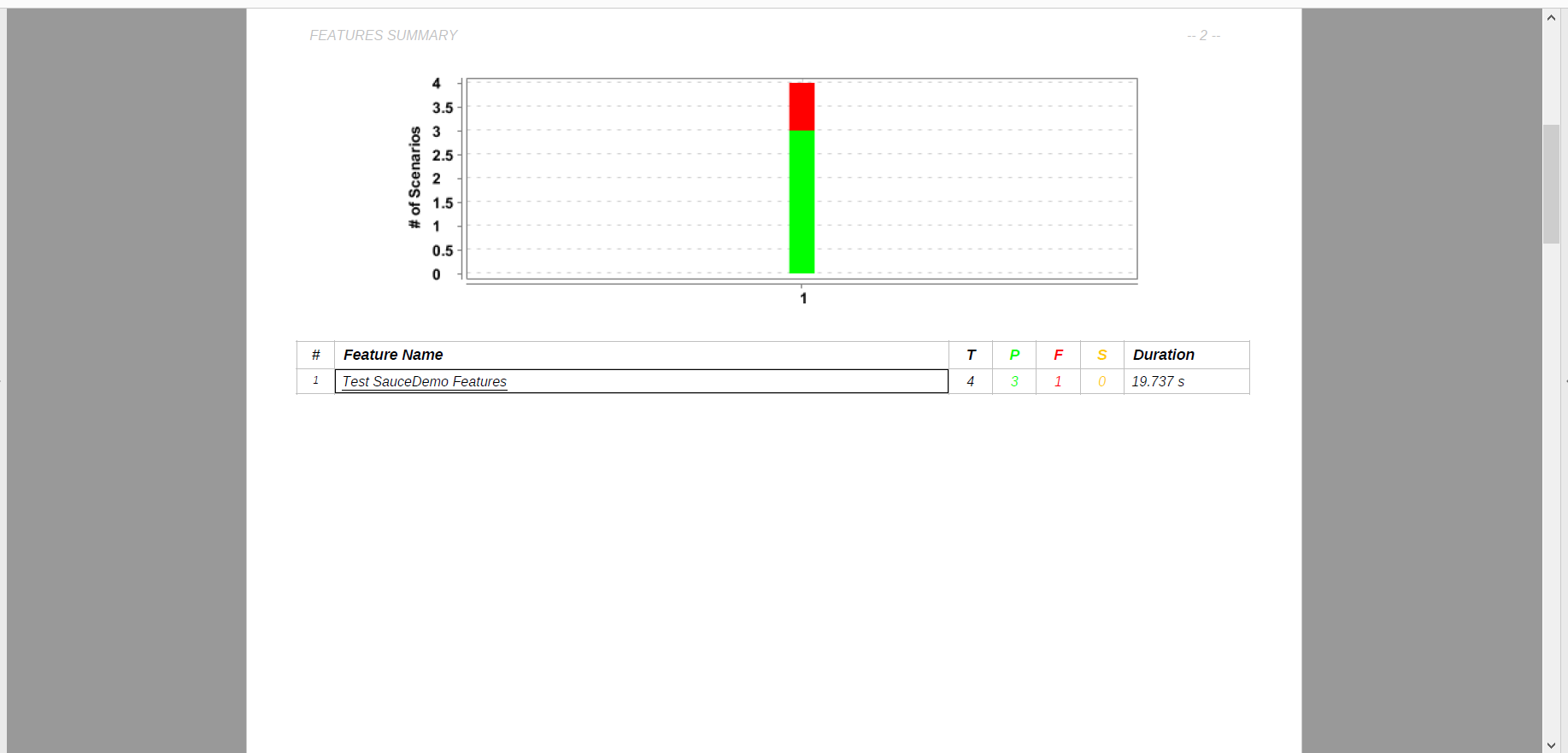


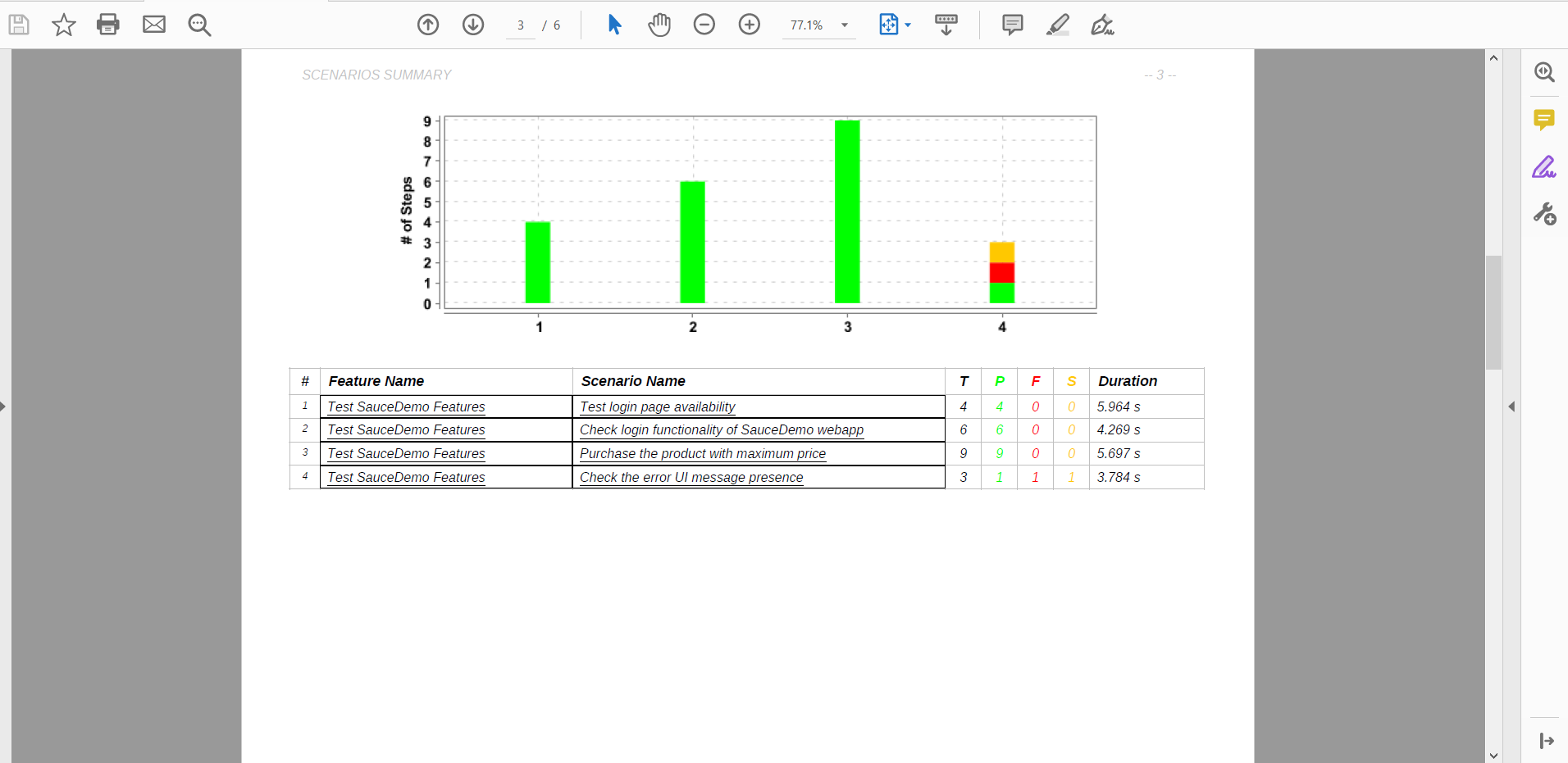
**Project execution reports –**

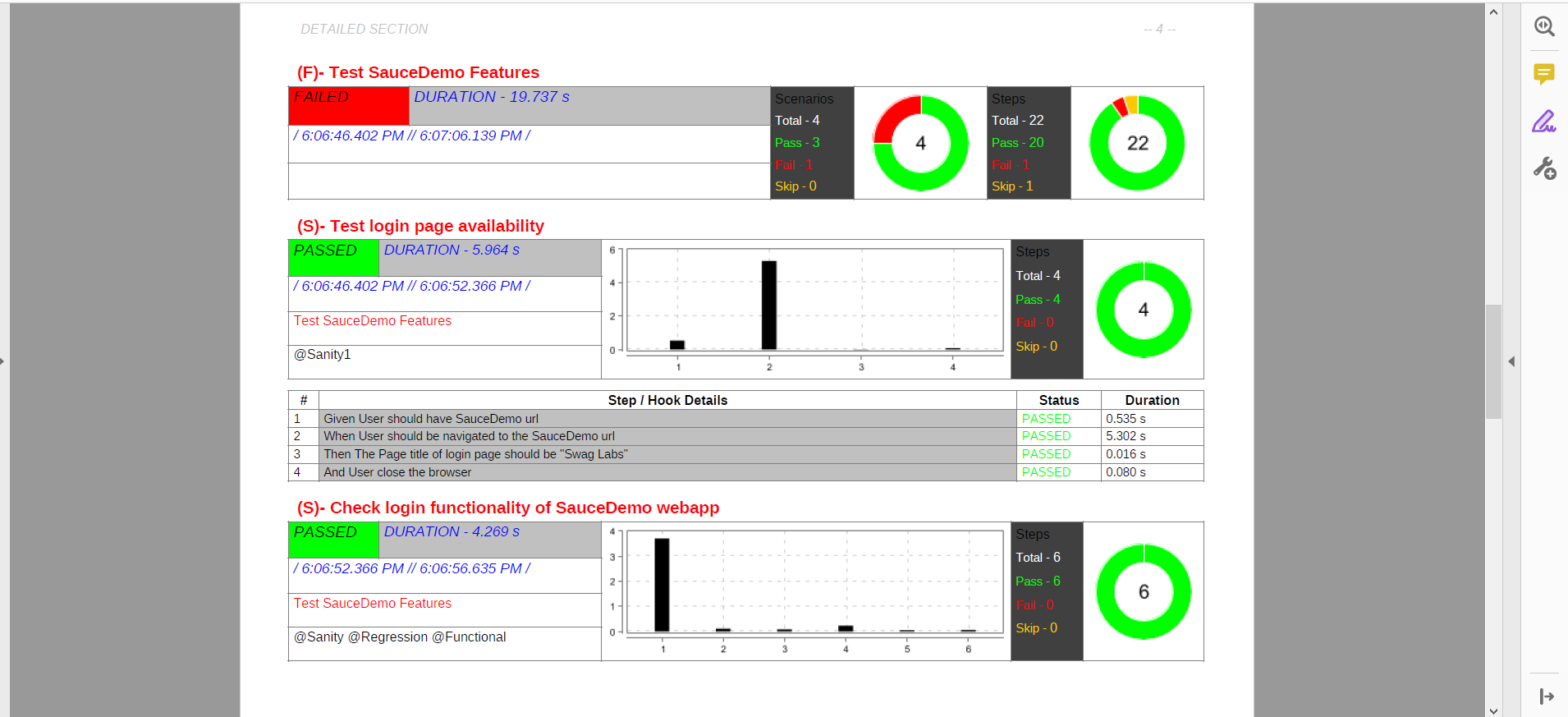
1. **PDF Report –**

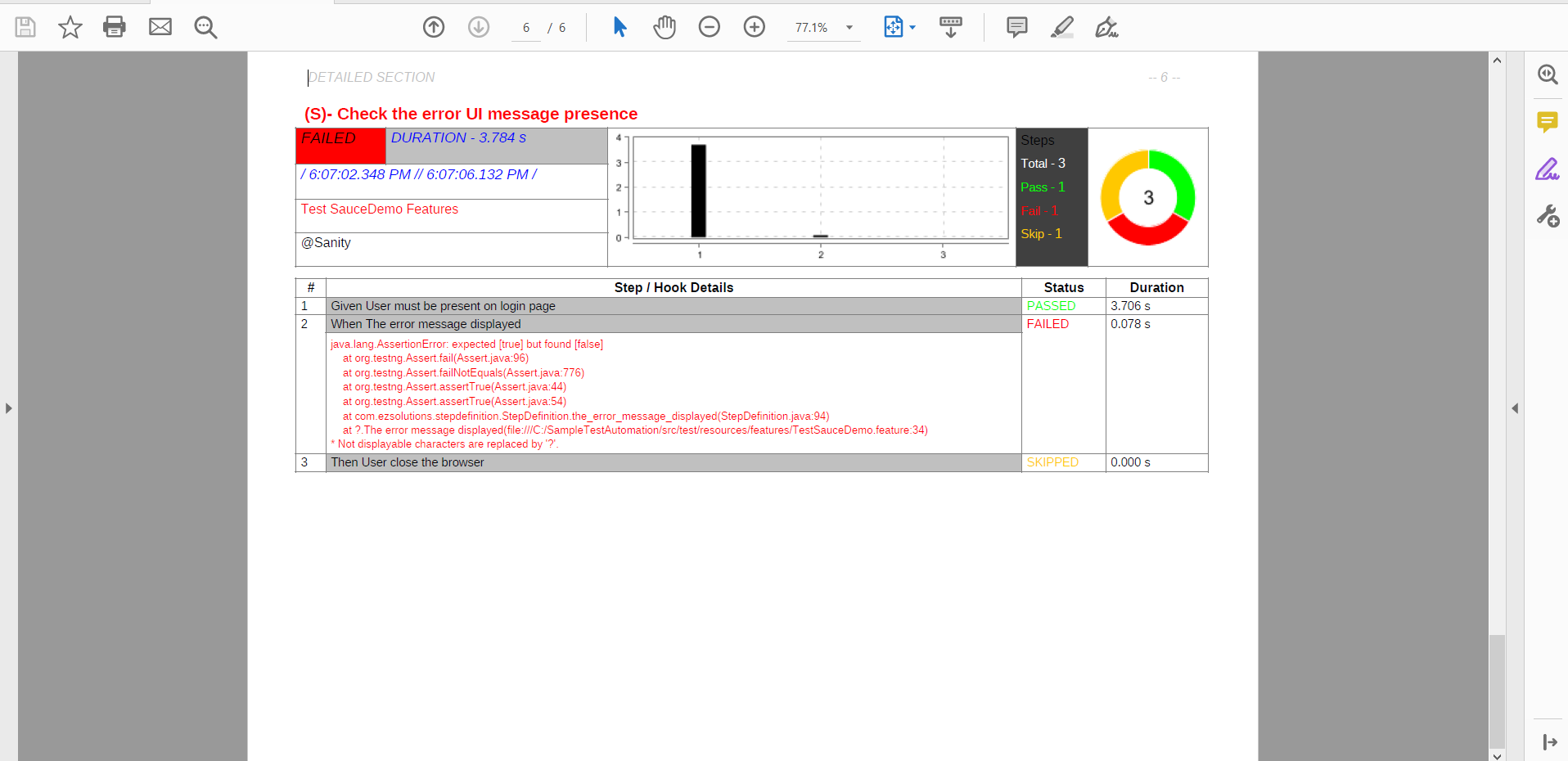




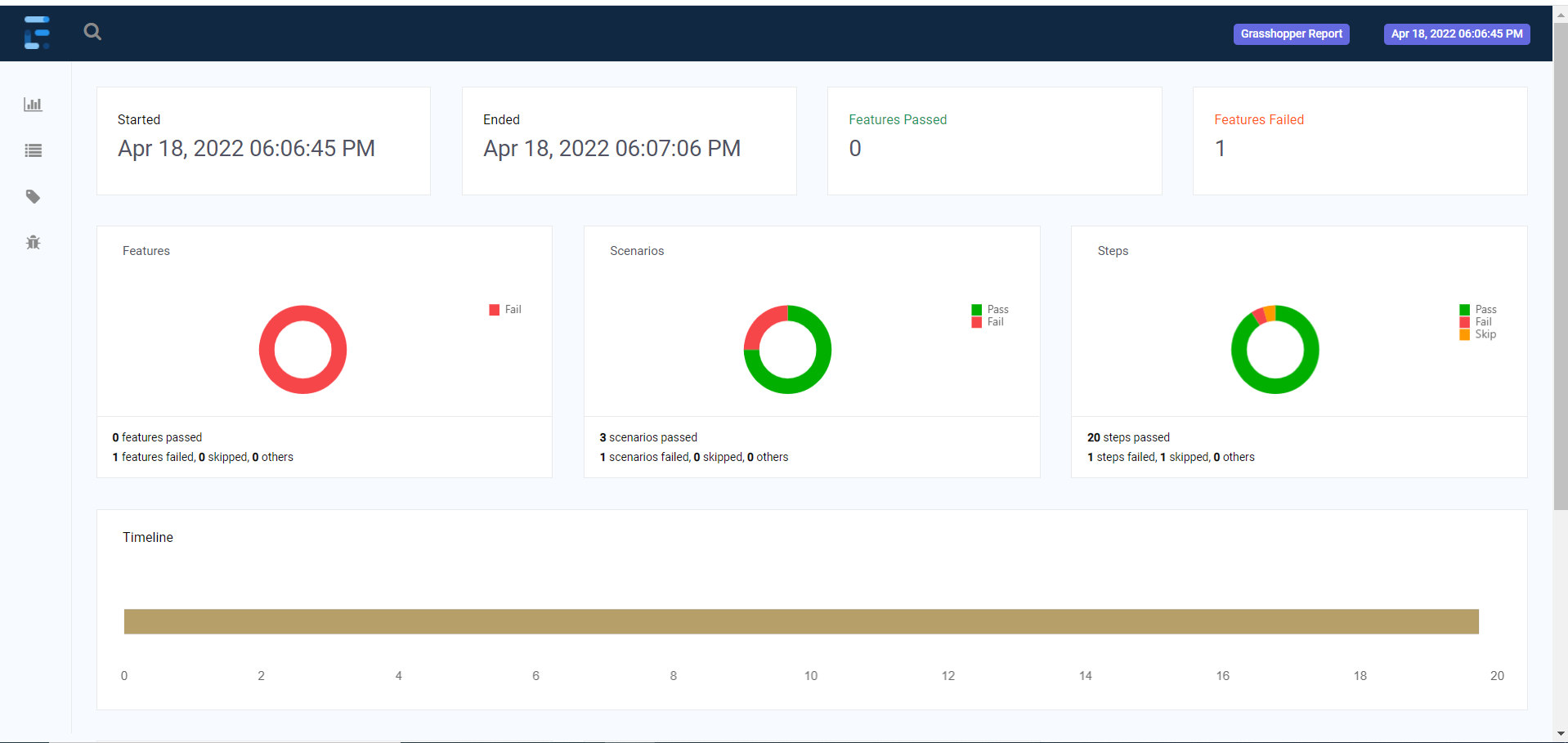


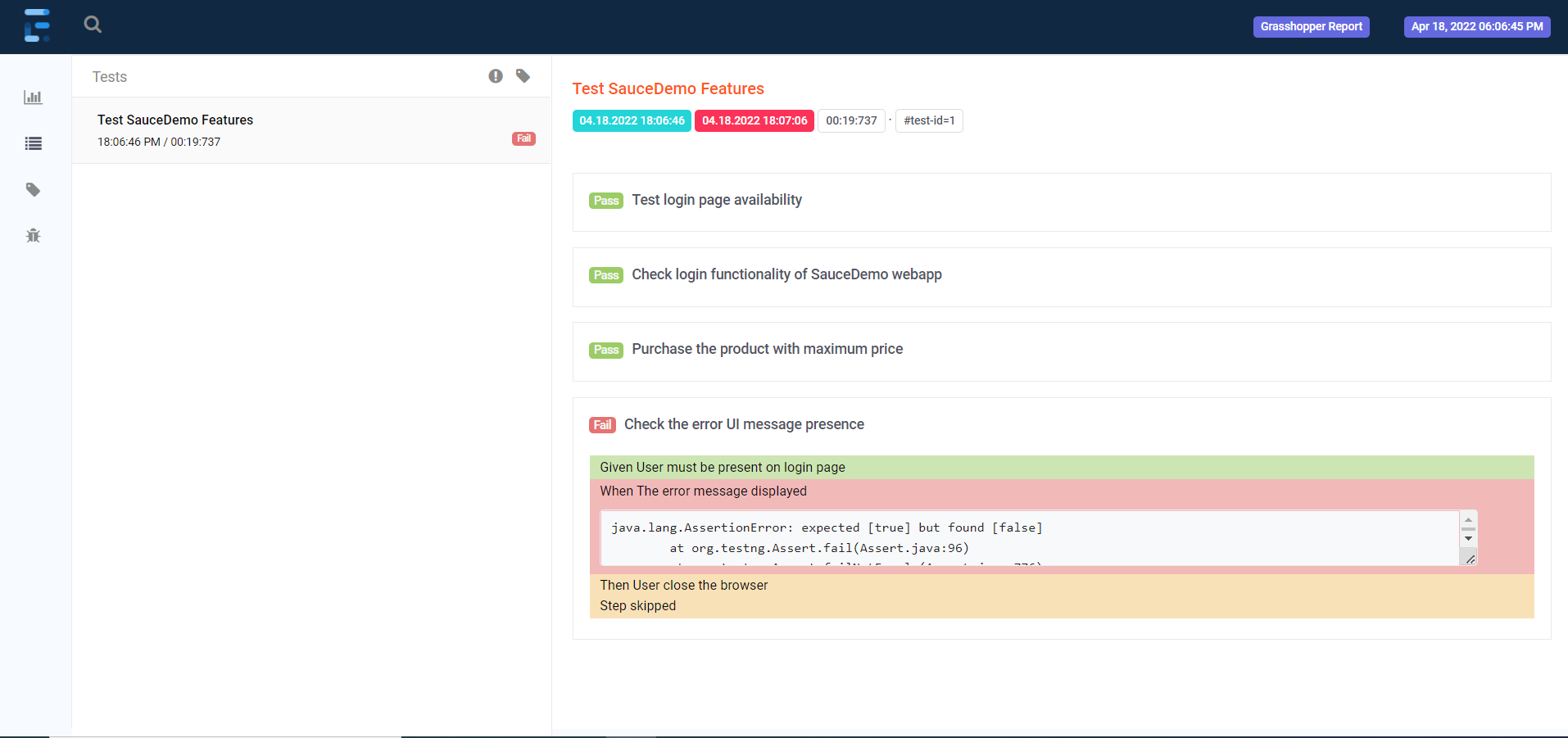


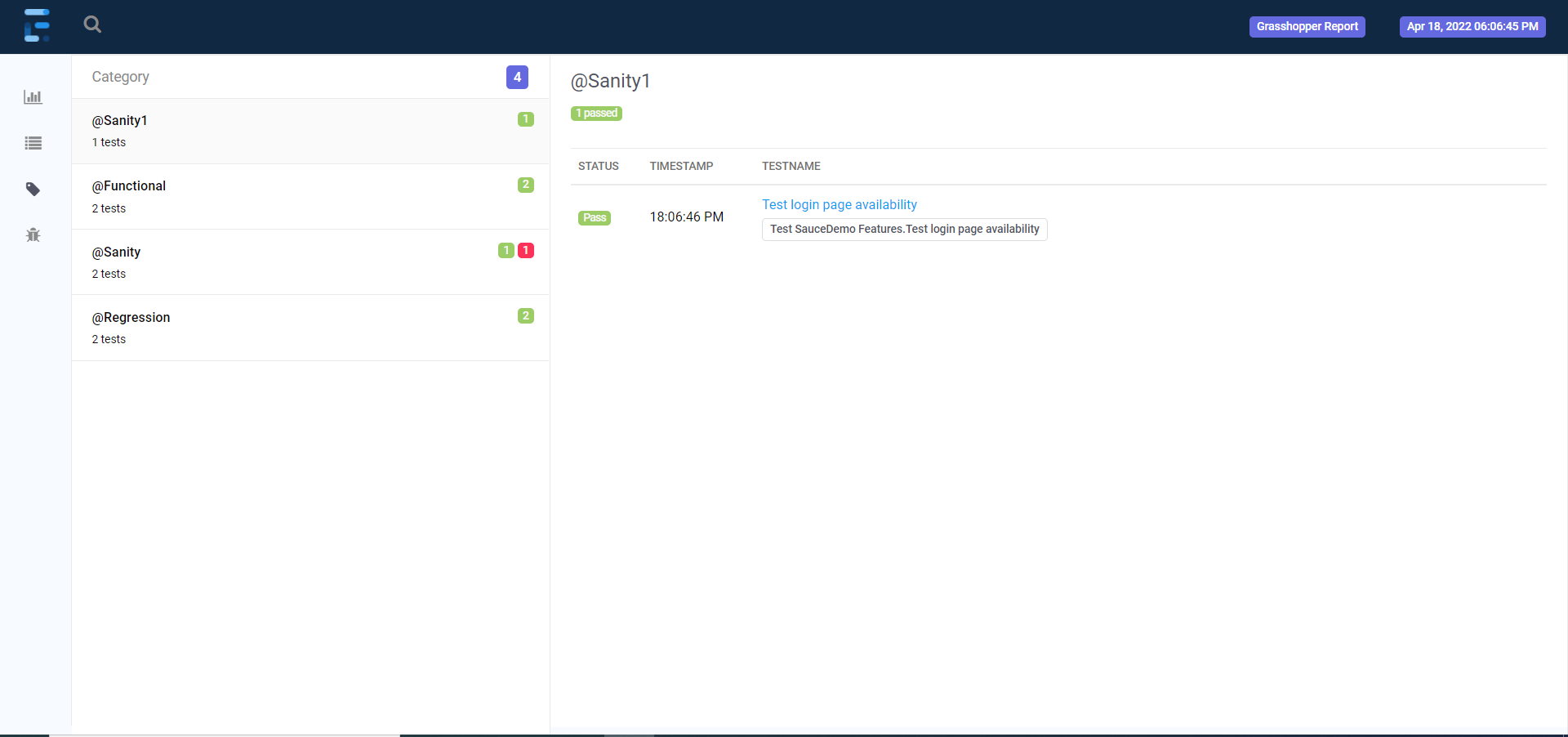


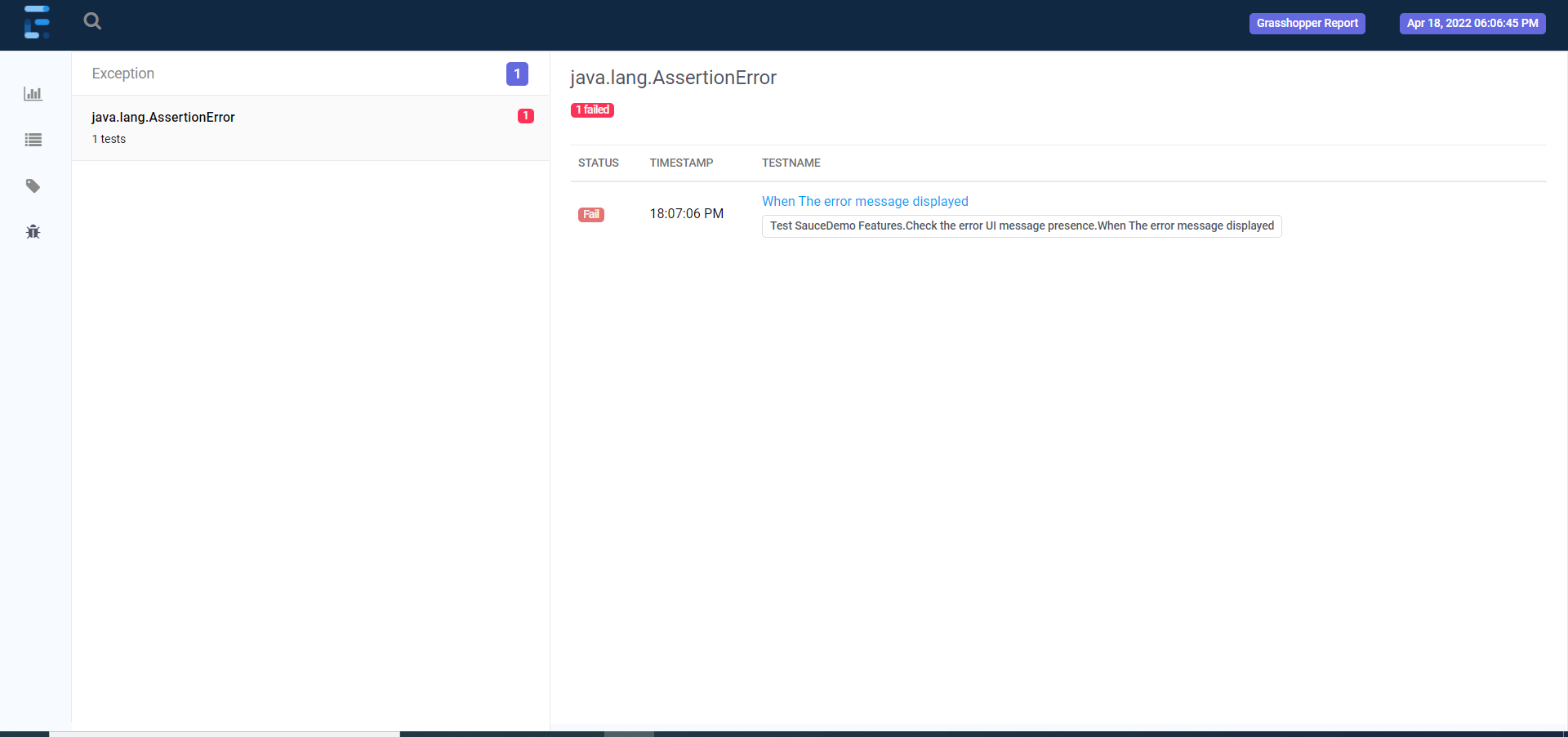


1. **Extent HTML Report –**









**Summary: -**

This was a small try to explain the need of hour for testing in industry. It clearly explains, what is testing, need of testing and different ways industry use to test applications. This project explains, the definitions of functional testing, automation testing etc. It demonstrates, the need of automation testing over functional testing. This is really a need of industry and it will definitely help to achieve a defect free application.