

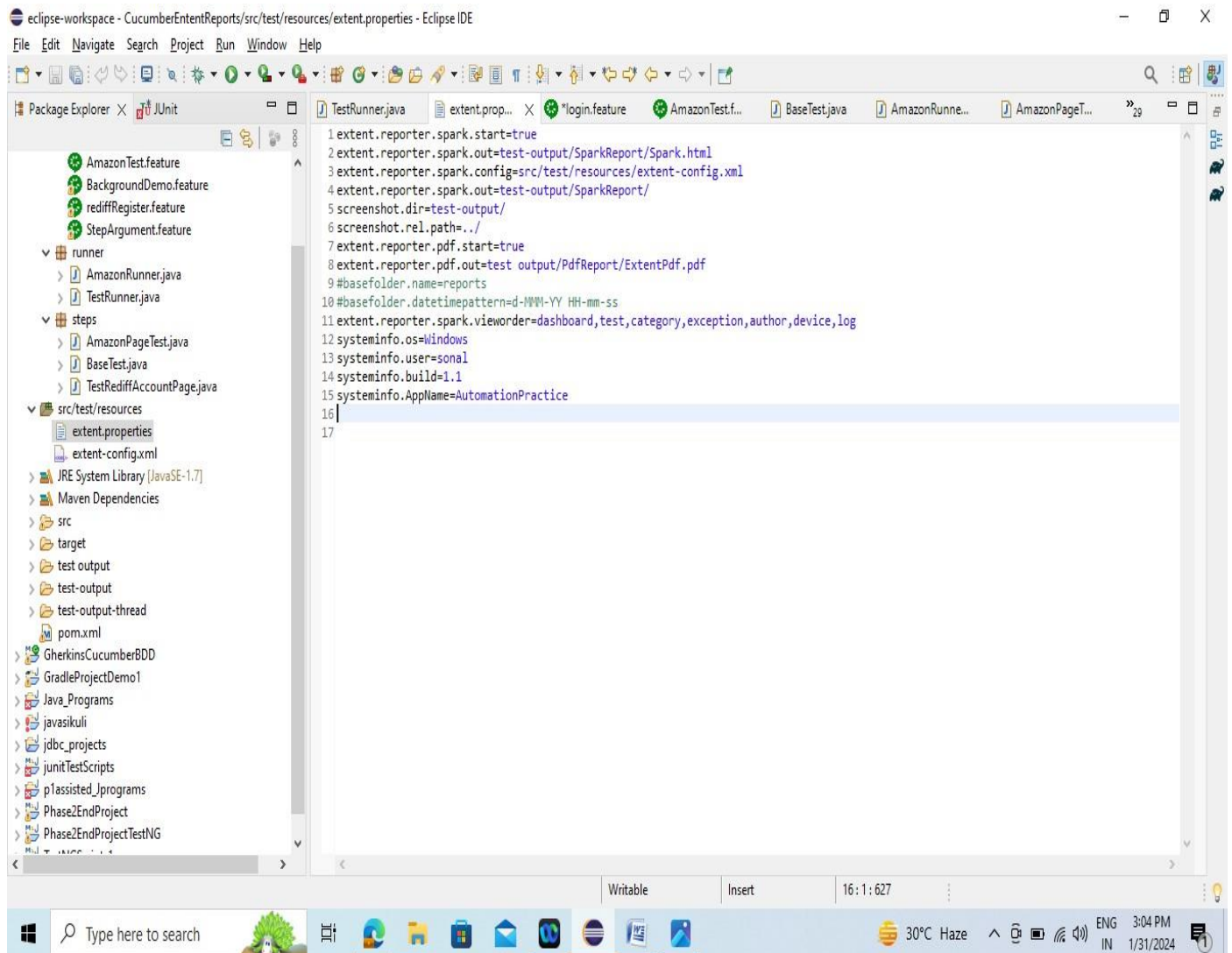
**PHASE\_2**

**EMP ID: 2587325**

**ASSISTED PRACTICE PROJECTS**

**NAME: HIRANMAI**

## 1. Cucumber Setup: Maven, Extent Reports, Cucumber and Eclipse Plugin



## 2. Gherkin Keywords

The following are the keywords used in Gherkin:

- **Feature:** This gives information on the high-level business functionality and the purpose of application under test. Everybody should be able to understand the intent of feature file by reading the first feature step.

Feature: Login Action Test

Description: This feature will test a LogIn and LogOut functionality

- **Scenario:** Basically, a scenario represents a particular functionality which is under test. By seeing the scenario, user should be able to understand the intent behind the scenario and test.

Scenario: Successful Login with Valid Credentials

Given User is on Home Page

When User Navigate to Login Page

And User enters Username and Password

Then Message displayed Login Successfully

### 3. Gherkin: Given, When, Then, and Background Steps

Each scenario should follow **given**, **when**, and **then** format.

- Given: Specifies the preconditions. It is basically a known state.
- When: This is used when some action is to be performed. Then: The expected outcome or result should be placed here. For Instance, verify if the login and page navigation is successful.
- Background: Whenever any step is required to be performed in each scenario, then those steps are to be placed in the Background. For Instance, If user needs to clear the database before each scenario, then those steps can be put in the background.
- And: Used to combine two or more actions of the same type.
- But: Signifies logical OR condition between any two statements. OR can be used in conjunction with GIVEN, WHEN, and THEN statement.
- *This feature will test functionality of adding different products to the User basket from different flow*
- *Background: User is Logged In*
- *Scenario: Search a product and add the first result/product to the User basket*
- *Given User searched for Lenovo Laptop*
- *When Add the first laptop that appears in the search result to the basket*
- *Then User basket should display with 1 item*
- *Feature: Add to Cart*

#### 4. Gherkin: Step Arguments

Gherkin step argument

- Gherkin has Docstrings and Data tables.

### **Docstrings**

- If you need to specify information in a scenario that won't fit on a single line, you can use Docstrings.
- A Docstring follows a step. It starts and ends with three double quotes, like this:

When I ask to reset my password

Then I should receive an email with:

"""

Dear bozo,

Please click this link to reset your password

"""

### **Data Tables**

- Data Tables are handy for passing a list of values to a step definition
- |        |                    |                 |
|--------|--------------------|-----------------|
| name   | email              | twitter         |
| Aslak  | aslak@cucumber.io  | @aslak_hellesoy |
| Julien | julien@cucumber.io | @jbpros         |
| Matt   | matt@cucumber.io   | @mattwynne      |
- Just like Docstrings, Data Tables will be passed to the step definition as the last argument.

eclipse-workspace - CucumberEntentReports/src/test/java/features/StepArgument.feature - Eclipse IDE

File Edit Navigate Search Project Run Window Help

Package Explorer X JUnit

CucumberEntentReports

- src/main/java
- src/test/java
  - features
    - AmazonTest.feature
    - BackgroundDemo.feature
    - rediffRegister.feature
    - StepArgument.feature
  - runner
    - AmazonRunner.java
    - TestRunner.java
  - steps
    - AmazonPageTest.java
    - BaseTest.java
    - TestRediffAccountPage.java
- src/test/resources
  - extent.properties
  - extent-config.xml
- JRE System Library [JavaSE-1.7]
- Maven Dependencies
- src
- target
- test output
- test-output
- test-output-thread
- pom.xml

GherkinsCucumberBDD

GradleProjectDemo1

Java\_Programs

javasikuli

jdbc\_projects

TestRediffAc... TestRunner.java extent.prop... \*login.feature \*Background... \*StepArgumen... X BaseTest.java » 29

```
1 # doc strings is represented by ""
2 # these allow a user to pass multiple lines with a given gerkin keyword
3 Feature: Validate the error message on the webpage
4
5 Scenario: Test Error message when invalid data entered
6   Given User is on microsoft webpage
7   When User click on Next button
8   Then User gets an error message
9   ""
10  Enter a valid email address,
11  phone number,
12  or Skype name.
13  ""
14  Then User enter valid username
15  ""
16  username = sonal
17  ""
18
```

Writable Insert 1:1:0

Type here to search

30°C Haze ENG 4:47 PM 1/31/2024

## 5. Gherkin: Comments and Tags

### Gherkin comments

- Comment is basically a piece of code meant for documentation purposes and not for execution.
- Feature file: In case of a feature file, we just need to put # before beginning your comment.

#### **Example:**

Feature: annotation

#This is how background can be used to eliminate duplicate steps

Background:

User navigates to Facebook

Given I am on Facebook login page

- Step definition file: If you are using Java as a platform then mark your comments with //.

#### **Example:**

//scroll to the bottom of the page

```
((JavascriptExecutor) driver).executeScript("window.scrollTo(0, document.body.scrollHeight)");
```

### Gherkin tags

- if we have many scenarios in a feature file, to put them under a single umbrella, we use tags to generate reports for specific scenarios under the same tag.
- Tags are defined in our runner class like this:

```
@RunWith(Cucumber.class)
```

```
@CucumberOptions{
```

```
format= {"Pretty" ,"json:target/output.json", "html:targer/html"},
```

```
feature={"src/functional-test/resources"},
```

```
tags={"@tag", "@tag1" }
```

- When we define multiple tags in runner class in below form, it will be defined with the use of logical operator:

- tags = {"@tag", "@tag1"}: means AND condition. It says that scenarios matching both these tags need to be executed.

tags = {"@tag1, @tag2"}: means OR condition. It says that scenarios matching any of this tag need to be executed

## 6. Step Definition

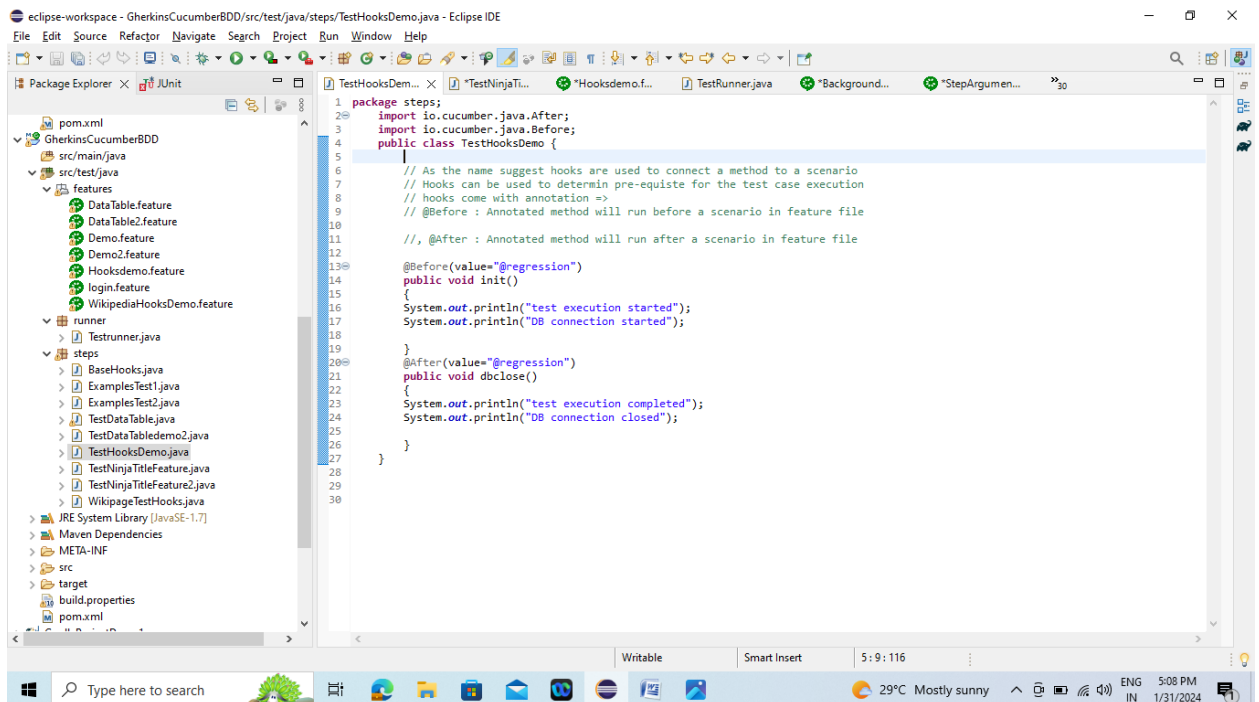
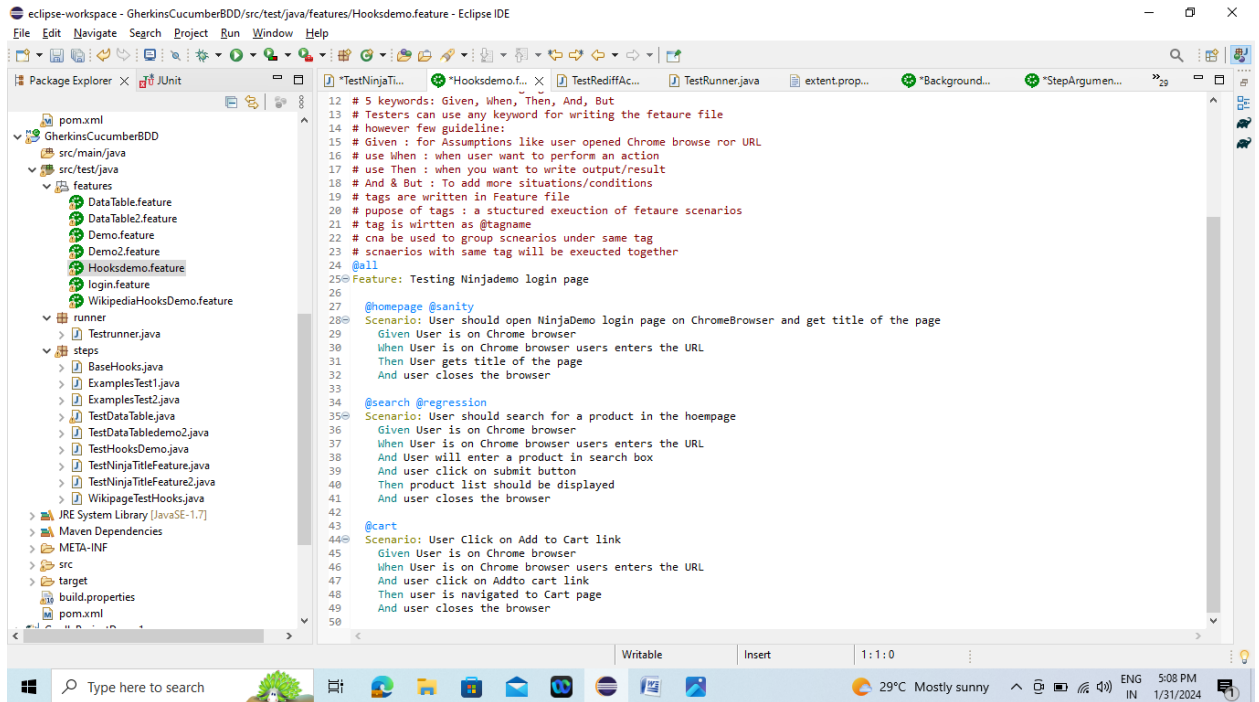
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure for 'GherkinsCucumberBDD'. The main editor window shows the code for 'TestNinjaTitleFeature.java'. The code defines a public class with several methods: 'OpenBrowser()', 'EnterURL()', 'testpageTitle()', and 'teardown()'. The code uses Selenium WebDriver and Cucumber annotations like @Given, @When, @Then, and @And. The IDE's status bar at the bottom shows 'Writable', 'Smart Insert', and the time '13:9:415'. The Windows taskbar at the very bottom shows the search bar, taskbar icons, and system tray information including '29°C Mostly sunny' and 'ENG IN 5:06 PM 1/31/2024'.

```
1 package steps;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import io.cucumber.java.en.And;
5 import io.cucumber.java.en.Given;
6 import io.cucumber.java.en.Then;
7 import io.cucumber.java.en.When;
8 public class TestNinjaTitleFeature {
9
10     // In cucumber we dont use Junit Annotations
11     // we use cucumber annotation:
12     // @Given , @Then, @When, @And, @But
13
14     // In this file we create method for every feature step
15     WebDriver driver;
16
17     @Given("User is on Chrome browser")
18     public void OpenBrowser()
19     {
20         driver = new ChromeDriver();
21         driver.manage().deleteAllCookies();
22         driver.manage().window().maximize();
23     }
24
25     @When("User is on Chrome browser users enters the URL")
26     public void EnterURL()
27     {
28         driver.get("https://tutorialsninja.com/demo/");
29     }
30
31     @Then("User gets title of the page")
32     public void testpageTitle()
33     {
34         String title = driver.getTitle();
35         System.out.println(title);
36     }
37
38     @And("user closes the browser")
39     public void teardown()
```



## 7. Hooks

Hooks are blocks of code that run **before** or **after** each scenario. You can define them anywhere in your project or step definition layers using the methods **@Before** and **@After**.



## 9. Execute Multiple Scenarios.

## Multiple scenarios

- Feature file can have more than one scenario or scenario outline. You can write your all possible requirement or scenarios for a particular feature in a feature file.

Feature: Registration, Login and MyAccount

Background:

Given I am on the homepage

And I follow "Sign in"

@regression @smoke

Scenario: Verify Login Functionality

And I fill "email address" with "goswami.tarun77@gmail.com"

And I fill "password" with "Test@1234"

And I click "sign in"

Then I should see "MY ACCOUNT" heading

@regression

Scenario: Create New User

When I fill "registration email text box" with "goswami.tarun77+1@gmail.com"

Then I click "create an account button"

And I enter following details

| First Name | Tarun |

| Last Name | Goswami |

| Password | Test1234 |

| Date | 13 |

| Year | 1989 |

And I click "register button"

## 10. Tagged Hooks

eclipse-workspace - GherkinsCucumberBDD/src/test/java/steps/BaseHooks.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer x JUnit

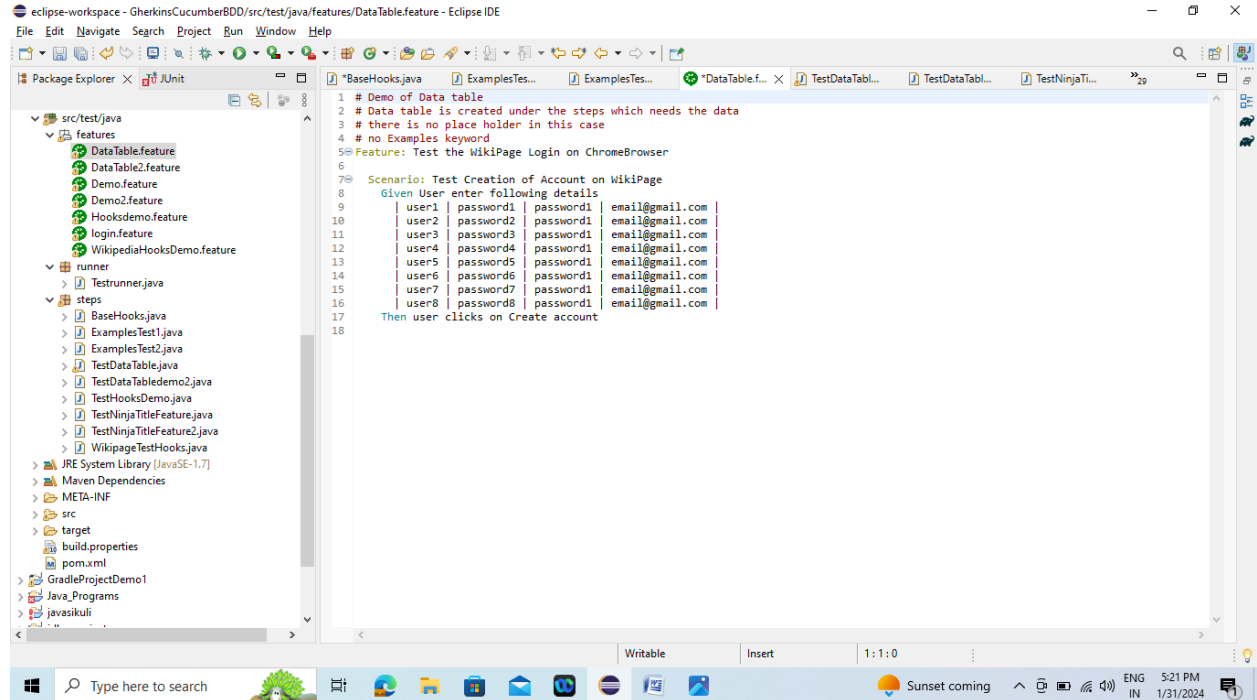
- Hooksdemo.feature
- login.feature
- WikipediaHooksDemo.feature
- runner
  - Testrunner.java
- steps
  - BaseHooks.java
  - ExamplesTest1.java
  - ExamplesTest2.java
  - TestDataTable.java
  - TestDataTabledemo2.java
  - TestHooksDemo.java
  - TestNinjaTitleFeature.java
  - TestNinjaTitleFeature2.java
  - WikipediaTestHooks.java
- JRE System Library [JavaSE-1.7]
- Maven Dependencies
- META-INF
- src
- target
- build.properties
- pom.xml
- GradleProjectDemo1
- Java\_Programs
- javasikuli
- jdbc\_projects
- JUnitTestScripts
- p1assisted\_Jprograms
- Phase2EndProject
- Phase2EndProjectTestNG
- TestNGScripts1

```
1 package steps;
2 import org.openqa.selenium.WebDriver;
3 import org.openqa.selenium.chrome.ChromeDriver;
4 import io.cucumber.java.After;
5 import io.cucumber.java.AfterAll;
6 import io.cucumber.java.AfterStep;
7 import io.cucumber.java.Before;
8 import io.cucumber.java.BeforeAll;
9 import io.cucumber.java.BeforeStep;
10 public class BaseHooks {
11     // In this we will write the methods that are annotated with Hooks annotation:
12     // @Before , @After, @BeforeAll, @AfterAll, @BeforeStep, @AfterStep
13     // Global Hooks
14     @BeforeAll
15     public static void featureStart() // this annotated method will execute before any of the scenario in feature begins
16     {
17         System.out.println("The feature started Execution");
18     }
19     @AfterAll
20     public static void featureCompleted()
21     // this annotated method will execute after all the scenarios in feature completed
22     {
23         System.out.println("The feature Execution Completed");
24     }
25     public static WebDriver driver;
26     @Before(order=1)
27     public void openBrowser()
28     {
29         System.out.println("task 1. Open Browser window");
30         driver = new ChromeDriver();
31     }
32     @Before(order=2)
33     public void ManageBrowser()
34     {
35         System.out.println("task 2. Manage Browser window");
36         driver.manage().window().maximize();
37         driver.manage().deleteAllCookies();
38     }
39     @Before(order=3)
```

Writable Smart Insert 38:10:1311

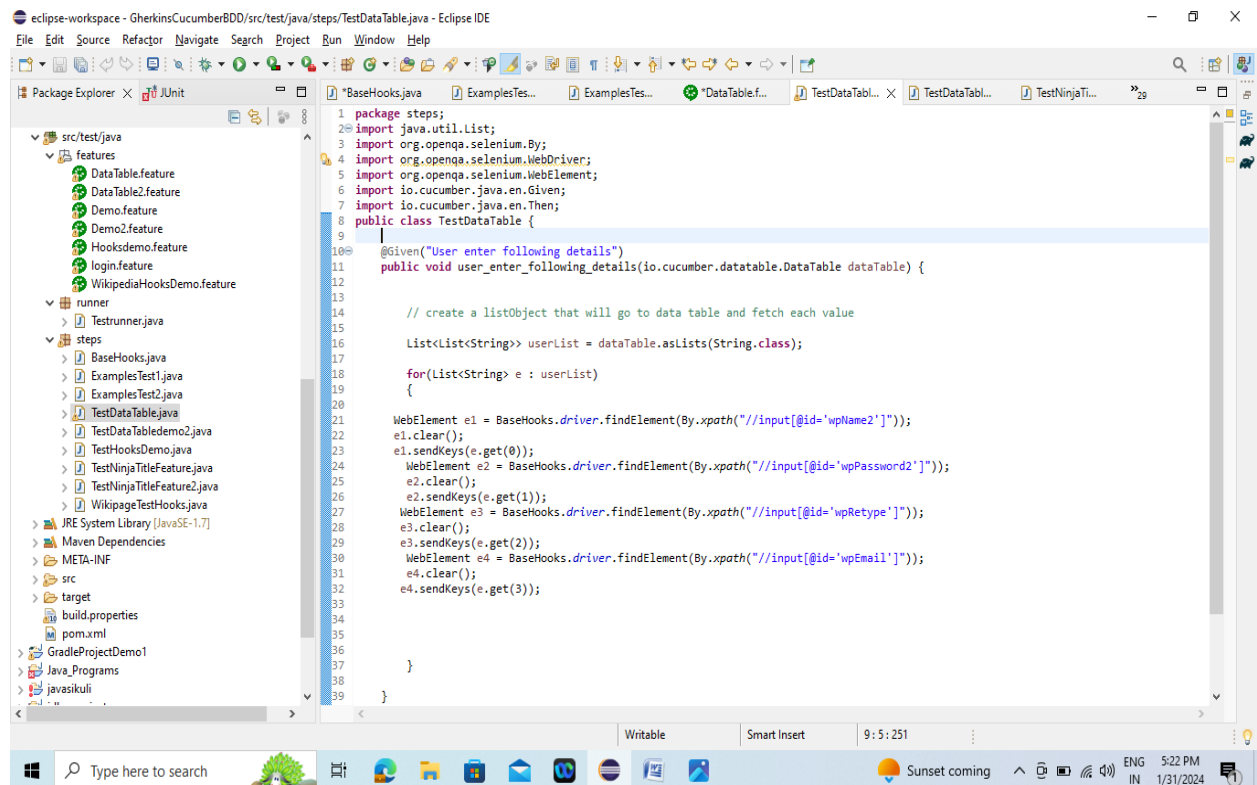
Type here to search 29°C Mostly sunny ENG IN 5:17 PM 1/31/2024

## 11. Cucumber Data Table



The screenshot shows the Eclipse IDE with the file `src/test/java/features/DataTable.feature` open. The Package Explorer on the left shows the project structure. The main editor displays the following Gherkin code:

```
1 # Demo of Data table
2 # Data table is created under the steps which needs the data
3 # there is no place holder in this case
4 # no Examples keyword
5 Feature: Test the WikiPage Login on ChromeBrowser
6
7 Scenario: Test Creation of Account on WikiPage
8   Given User enter following details
9     | user1 | password1 | password1 | email@gmail.com |
10    | user2 | password2 | password1 | email@gmail.com |
11    | user3 | password3 | password1 | email@gmail.com |
12    | user4 | password4 | password1 | email@gmail.com |
13    | user5 | password5 | password1 | email@gmail.com |
14    | user6 | password6 | password1 | email@gmail.com |
15    | user7 | password7 | password1 | email@gmail.com |
16    | user8 | password8 | password1 | email@gmail.com |
17   Then user clicks on Create account
```



The screenshot shows the Eclipse IDE with the file `src/test/java/steps/TestDataTable.java` open. The Package Explorer on the left shows the project structure. The main editor displays the following Java code:

```
1 package steps;
2 import java.util.List;
3 import org.openqa.selenium.By;
4 import org.openqa.selenium.WebDriver;
5 import org.openqa.selenium.WebElement;
6 import io.cucumber.java.en.Given;
7 import io.cucumber.java.en.Then;
8 public class TestDataTable {
9
10    @Given("User enter following details")
11    public void user_enter_following_details(io.cucumber.datatable.DataTable dataTable) {
12
13        // create a listObject that will go to data table and fetch each value
14
15        List<List<String>> userList = dataTable.asLists(String.class);
16
17        for(List<String> e : userList)
18        {
19
20            WebElement e1 = BaseHooks.driver.findElement(By.xpath("//*[@id='wpName2']"));
21            e1.clear();
22            e1.sendKeys(e.get(0));
23            WebElement e2 = BaseHooks.driver.findElement(By.xpath("//*[@id='wpPassword2']"));
24            e2.clear();
25            e2.sendKeys(e.get(1));
26            WebElement e3 = BaseHooks.driver.findElement(By.xpath("//*[@id='wpRetype']"));
27            e3.clear();
28            e3.sendKeys(e.get(2));
29            WebElement e4 = BaseHooks.driver.findElement(By.xpath("//*[@id='wpEmail']"));
30            e4.clear();
31            e4.sendKeys(e.get(3));
32
33        }
34    }
35
36    }
37
38    }
39 }
```

## 12. Cucumber Integration with Extent Report

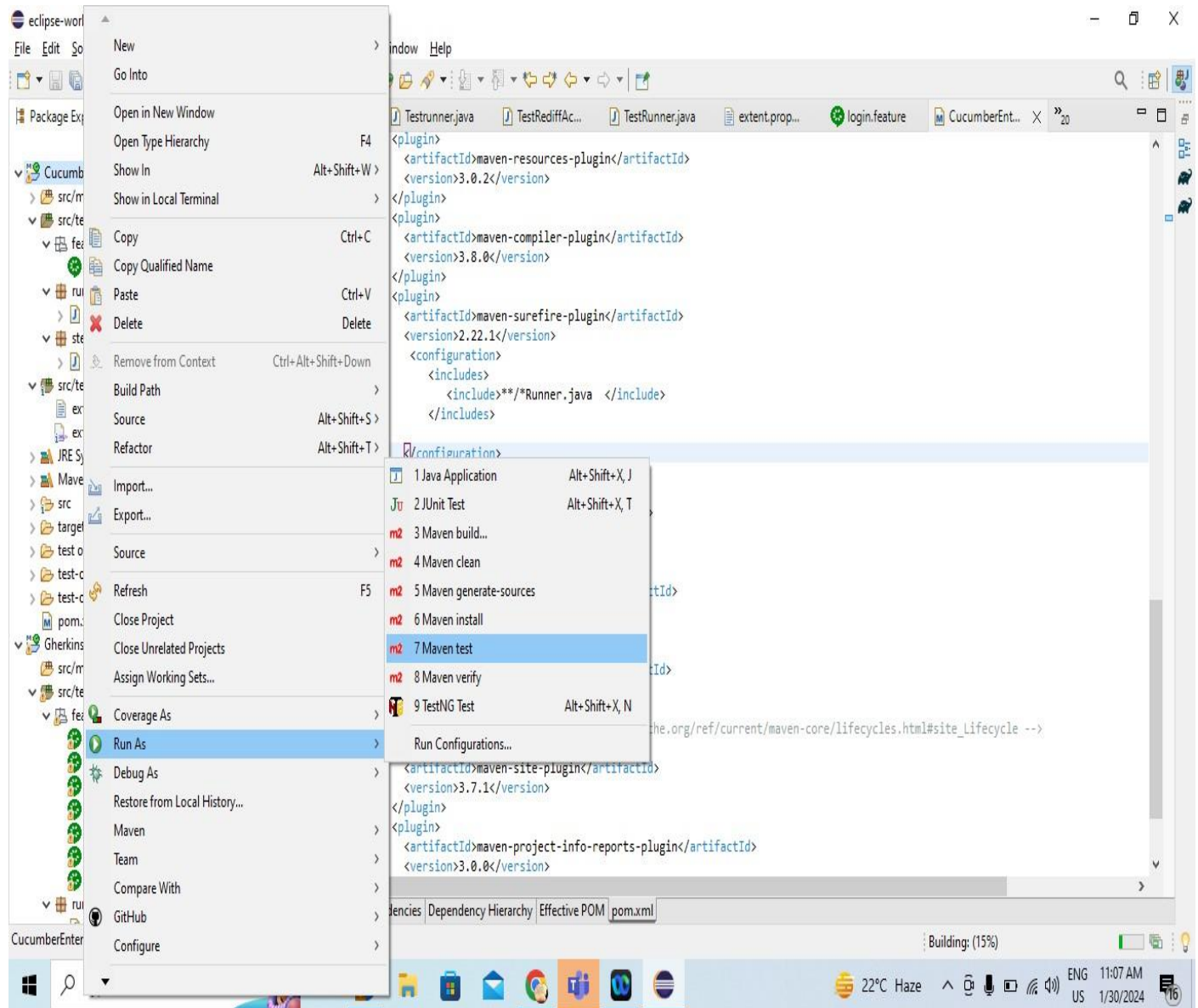
The top screenshot shows the Eclipse IDE with the 'extent.properties' file open. The file contains the following properties:

```
1 extent.reporter.spark.start=true
2 extent.reporter.spark.out=test-output/SparkReport/Spark.html
3 extent.reporter.spark.config=src/test/resources/extent-config.xml
4 extent.reporter.spark.out=test-output/SparkReport/
5 screenshot.dir=test-output/
6 screenshot.rel.path=../
7 extent.reporter.pdf.start=true
8 extent.reporter.pdf.out=test_output/PdfReport/ExtentPdf.pdf
9 #basefolder.name=reports
10 #basefolder.datetimepattern=d-MM-YY HH-mm-ss
11 extent.reporter.spark.vieworder=dashboard,test,category,exception,author,device,log
12 systeminfo.os=Windows
13 systeminfo.user=sonal
14 systeminfo.build=1.1
15 systeminfo.AppName=AutomationPractice
16
17
```

The bottom screenshot shows the Eclipse IDE with the 'extent-config.xml' file open. The file contains the following XML configuration:


```
1 <?xml version="1.0" encoding="UTF-8"?>
2   <!-- Bind to grammar/schema -->
3   <extentreports>
4     <configuration>
5       <!-- report theme -->
6       <!-- standard, dark -->
7       <theme>dark</theme>
8       <!-- document encoding -->
9       <!-- defaults to UTF-8 -->
10      <encoding>UTF-8</encoding>
11      <!-- protocol for script and stylesheets -->
12      <!-- defaults to https -->
13      <protocol>http</protocol>
14      <!-- title of the document -->
15      <documentTitle>Extent</documentTitle>
16      <!-- report name - displayed at top-nav -->
17      <reportName>Grasshopper Report</reportName>
18      <!-- location of charts in the test view -->
19      <!-- top, bottom -->
20      <testViewChartLocation>bottom</testViewChartLocation>
21      <!-- custom javascript -->
22      <scripts>
23        <![CDATA[
24          $(document).ready(function() {
25          });
26        ]]>
27      </scripts>
28      <!-- custom styles -->
29      <styles>
30        <![CDATA[
31        ]]>
32      </styles>
33    </configuration>
34  </extentreports>
35
36
37
```

## 13. Cucumber Execution with Maven





## 14. Build (Execute) Jenkins Job


localhost:8080/view/All/newJob


 **Jenkins**


Jenkins > All >


 **New Item**

 People

 Build History

 Manage Jenkins

 Credentials

 My Views

**Build Queue**

No builds in the queue.

**Build Executor Status**

master

1 Idle

2 Idle

Jenkins Slave

(offline)

Item name

☐ Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system.

☒ Maven project

Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

☐ External Job

This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine [details](#).

☐ Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environmen

☐ Copy existing Item

Copy from

OK

Results :

Tests run: 8, Failures: 0, Errors: 0, Skipped: 0

[JENKINS] Recording test results

[INFO] -----

[INFO] BUILD SUCCESS

[INFO] -----

[INFO] Total time: 13.848 s

[INFO] Finished at: 2017-01-28T22:26:26+05:30

[INFO] Final Memory: 15M/36M

[INFO] -----

[JENKINS] Archiving F:\Selenium\mavenparameterize\pom.xml to com.easy/mavenparameterize/0.0.1-SNAPSHOT/mavenparameterize-0.0.1-SNAPSHOT.pom

channel stopped

Finished: SUCCESS



## 15. Jenkins Integration with Extent Report

**Publish HTML reports**

Reports

HTML directory to archive: D:\Test\_PP\_POM\test-output

Index page[s]: STMExtentReport9.html

Index page title[s] (Optional):

Report title: Extent Report

Publishing options...

Add

