BDD	BDD or behavior-driven development, in which an application is specified and designed by describing how it behaves. BDD offers the ability to enlarge the pool of input and feedback to include business stakeholders and end users like Scrum Master, Product Owner, Business Analyst etc., who may not even have software development knowledge.
Cucumber	Software tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for the web application. The tests are written in easily readable and understandable format for Business Analysts, Developers and Testers.
Features of BDD	Shifting from thinking in "tests" to thinking in "behavior" Collaboration between Business stakeholders, Business Analysts, QA Team and developers Driven by Business Value Extends Test-Driven Development (TDD) by utilizing natural language that non-technical stakeholders can understand BDD frameworks such as Cucumber or JBehave are an enabler, acting a "bridge" between Business & Technical Language BDD is popular and can be utilized for Unit level test cases and for UI level test cases.
Feature	The Feature keyword's aim is to collect relevant scenarios and provide a high-level description of a software feature.
Scenario	The scenarios are written based on the expected behavior of the software and it is tested to check if it matches said scenarios.
Step	Each line in a scenario is called a step
Given	Describes the initial steps of pre-condition before the start of a test
When	Describes user actions during a test or steps performed
Then	Describes test results or outcome from When actions
And	Between any two statements, it gives the logical AND condition. AND can be combined with the GIVEN, WHEN, and THEN statements
But	It denotes a logical OR relationship between two propositions. OR can be combined with the GIVEN, WHEN, and THEN statements
Background	The Background section describes any common context to be established before each scenario.
Step Definitions	Cucumber scenarios become automated tests with the addition of what are called step definitions. A step definition is a block of code associated with one or more steps by a regular expression
Example	This is a practical illustration of a business rule. It comprises a series of steps.
Scenario Outline	The scenario outline is similar to scenario, with the exception that several inputs are provided.
tags	Tags in cucumber provide a way to run scenarios in a specific sequence from a runner file. Each situation can be labeled with a useful tag. Later, in the runner file, user may specify which tag (and hence which scenario(s)) Cucumber should run. “@” is the first character in a tag. Any relevant content after "@" can be used to define your tag. Example - "@SmokeTest"

hooks	Hooks are code blocks that execute before or after each Cucumber scenario in the execution cycle. This enables us to better control the development workflow and decrease code redundancy. Setting up the web driver and terminating the web driver session resembles a test setup. The methods @Before and @After can be used to define hooks anywhere in the project or step definition layers. Before hook is executed before any other test situations, and after the hook is executed after all test scenarios have been completed.
Feature File	It has plain text descriptions of single or numerous test situations. Keywords like Then, When, Background, Scenario Outline, Feature, And, But, and so on are used in the tests. As a result, it's a file that keeps track of features and their descriptions.
Step Definition File	It essentially acts as a translator between the test scenario steps provided in the feature file and the automation code. Cucumber searches the step definition file and executes the relevant functions that are assigned to that step when it runs a step described in the feature file.
TestRunner	It connects the feature file and the step definition file. It allows the user to run one or more feature files at the same time. It contains the locations of the step definition and feature files
Prerequisite required for using Cucumber	Step 1: Download and install the Java platform on user machine Step 2: Download and install Eclipse IDE Step 3: Download Cucumber Eclipse Plugin: a. In the eclipse, navigate to Help > Install New Software. Copy the URL "http://cucumber.github.io/cucumber-eclipse/update-site/" and press Enter. b. User would see a checkbox named "Cucumber Eclipse Plugin", Select the checkbox ‘Cucumber Eclipse Plugin’. c. Click ‘Next’ d. Again click ‘Next’ and Accept the license terms. e. Click Finish f. Click ‘Install anyway’ g. Click ‘Restart Now’ Step 4: Create a Maven Project in Eclipse Step 5: Open the pom.xml file in eclipse and the below dependency after navigating to Maven Repository "https://mvnrepository.com/" a. cucumber-java b. cucumber-core c. cucumber-junit

Cucumber Maven Dependency	<dependency>
	   <groupId>io.cucumber</groupId>
	   <artifactId>cucumber-core</artifactId>
	   <version>7.0.0</version>
	</dependency>
	
	         <dependency>
	   <groupId>io.cucumber</groupId>
	   <artifactId>cucumber-java</artifactId>
	   <version>7.0.0</version>
	</dependency>
	
	      <dependency>
	   <groupId>io.cucumber</groupId>
	   <artifactId>cucumber-junit</artifactId>
	   <version>7.0.0</version>
	    <scope>test</scope>
	</dependency>

Sample Feature File	Feature: Search in Google Home Page
	
	  Scenario: Search Cucumber Tutorial
	
	   Given Google Page open
	   And Search Text Box should be present in the Google Home Page
	   When User Search a Course with keyword Cucumber Tutorial
	   And Hit Enter Button
	   Then All Courses related to Cucumber Tutorial should be displayed

Sample Step Definitions Class	
	
	public class GoogleSearchEngine {
	@Given("Google Page open")
	public void google_page_open() {
	}
	@Given("Search Text Box should be present in the Google Home Page")
	public void search_text_box_should_be_present_in_the_google_home_page() {
	 }
	@When("User Search a Course with keyword Cucumber Tutorial")
	public void user_search_a_course_with_keyword_cucumber_tutorial() {
	  }
	@When("Hit Enter Button")
	public void hit_enter_button() {
	  }
	@Then("All Courses related to Cucumber Tutorial should be displayed")
	public void all_courses_related_to_cucumber_tutorial_should_be_displayed() {
	 }

Sample Runner Class	@RunWith(Cucumber.class)
	@CucumberOptions(
	features = ("src/test/java/Features"),
	glue = ("StepDefinitions"),
	plugin = ("pretty"),
	monochrome = true
	)
	public class Runner {
	}

Sample Cucumber Hooks	@Before Hook: It will execute before every scenario. Example
	@Before
	public void setUp() {
	      System.out.println("Starting the test");
	}
	@After Hook: It will execute after every scenario.
	@After
	Public void tearDown () {
	        System.out.println("Closing the test");
	}

Sample Scenario Outline Example	Feature: Facebook Login
	Scenario Outline: To check the login functionality for the facebook site
	   Given User Navigates to the Facebook Login Page
	   When User Enter <username> as UserName and <password> as Password
	   Then Login should be <status> for Facebook
	
	   Examples:
	     | username | password | status |
	     | username1 | password1 | Successful |
	     | username1 | password2 | Unsuccessful |
	     | username2 | password3 | Unsuccessful |

Sample Cucumber Tags and Background	Feature: Search in Google Home Page
	
	  Background:
	    Given Google Home Page Open
	    And Search Text Box is visible and Enabled
	
	@Smoke @Regression
	  Scenario: Search Cucumber Tutorial in Google Home Page
	    When User Search a Course with Keyword Cucumber Tutorial
	    And Hit Enter
	    Then All Courses related to Cucumber Tutorial should be displayed
	
	@Regression @Integration
	  Scenario: Search Java Tutorial in Google Home Page
	    When User Search a Course with Keyword Java Tutorial
	    And Hit Enter
	    Then All Courses related to Java Tutorial should be displayed

Sample Runner Class to accommodate tags	@RunWith(Cucumber.class)
	@CucumberOptions(
	features = ("src/test/java/Features"),
	glue = ("StepDefinitions"),
	//tags = ("@Integration"),
	//tags = ("not @Integration"),
	tags = ("@Integration or @UAT"),
	//tags = ("@Integration and @Regression"),
	
	publish = true,
	plugin = ("pretty"),
	monochrome = true
	)
	
	public class Runner {
	
	}

Sample Runner Class to accommodate reports	@RunWith(Cucumber.class)
	@CucumberOptions(
	features = ("src/test/java/Features"),
	glue = ("StepDefinitions"),
	tags = ("@UAT"),
	publish = true,
	plugin = ("pretty"),
	//plugin = {"pretty", "html:target/html-reports/report.html", "junit:target/junit-reports/", "junit:target/xml-reports/report.xml", "json:target/json-reports/report.json"},
	//plugin = {"pretty", "html:target/html-reports/report.html"},
	monochrome = true
	)
	public class Runner {
	}