GTS EDUTECH

Cucumber with Java

Shreyata Sugandhi

Corporate Trainer & Consultant

shreyata@goldentouchsolutions.tech

https://www.linkedin.com/in/shreyatasugandhi/

Contents

GTS EDUTECH

- Introduction
- Cucumber Architecture
- Cucumber Project Structure
- Components of Cucumber – Feature, Scenarios, Steps, etc.
- Keywords
- Tags
- Hooks
- Installation
- Practical Implementation of Cucumber Tests

Introduction

GTS EDUTECH

- Cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for web application.
- It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, Product Owner, etc.
- Cucumber can be used along with Selenium, Watir and many more.
- Cucumber supports many other languages like Perl, PHP, Python, .Net etc.

Introduction

GTS EDUTECH

- It is helpful to involve business stakeholders who can't easily read code
- Cucumber focuses on end-user experience
- Style of writing tests allow for easier reuse of code in the tests
- Quick and easy set up and execution
- Efficient tool for testing
- Can be easily plugged/integrated with various tool
- Cucumber is developed in Ruby

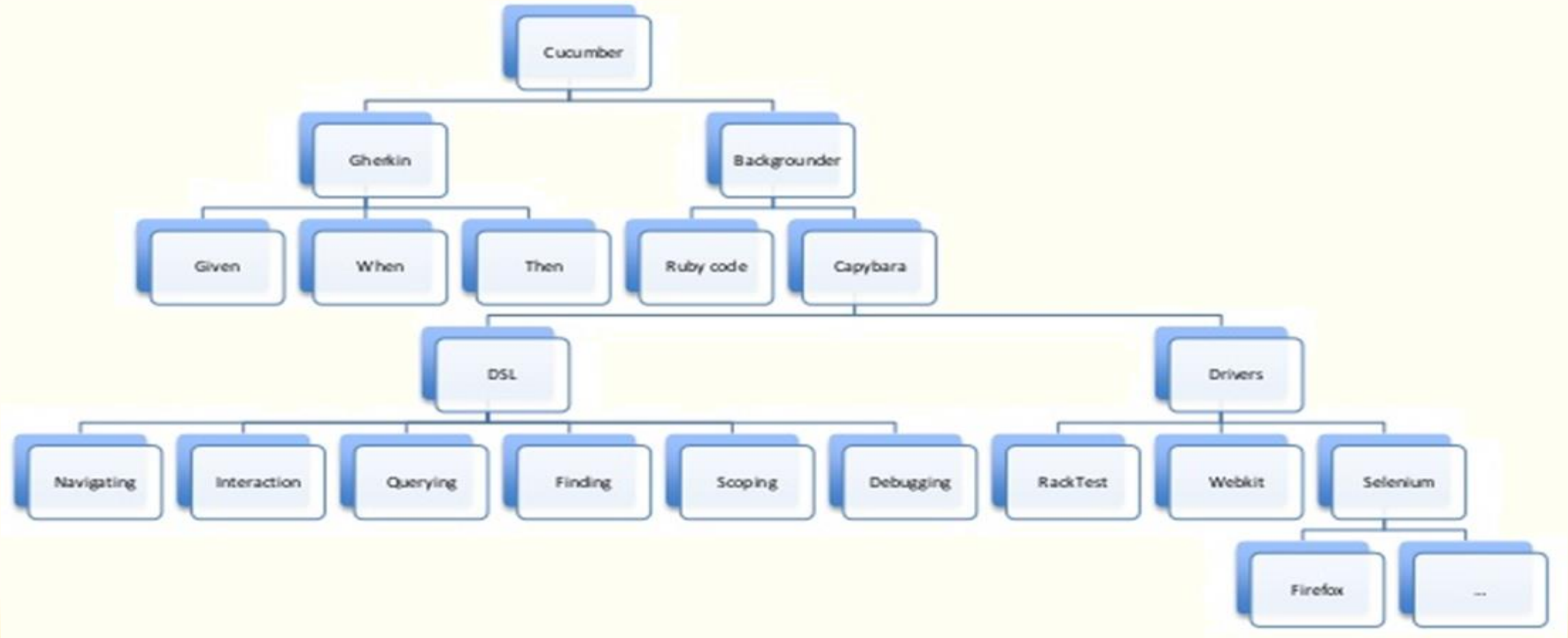
Cucumber & Languages

GTS EDUTECH

	Ruby/JRuby
	JRuby (using Cucumber-JVM)
	Java
	Groovy
JS	JavaScript
JS	JavaScript (using Cucumber-JVM and Rhino)
	Clojure
GOSU	Gosu
	Lua
.NET	.NET (using SpecFlow)
php	PHP (using Behat)
python	Jython
	C++
	Tcl

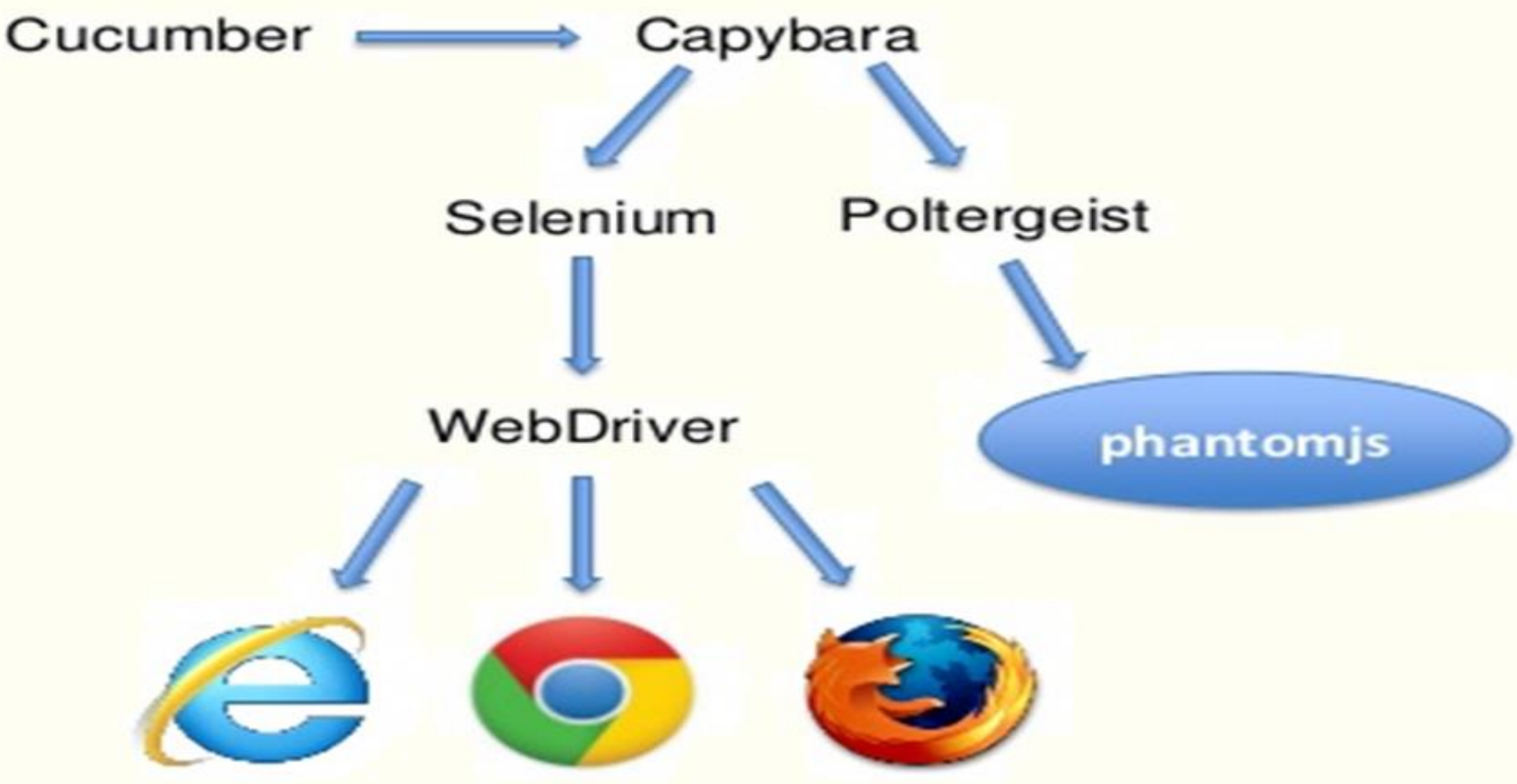
Cucumber Architecture

GTS EDUTECH



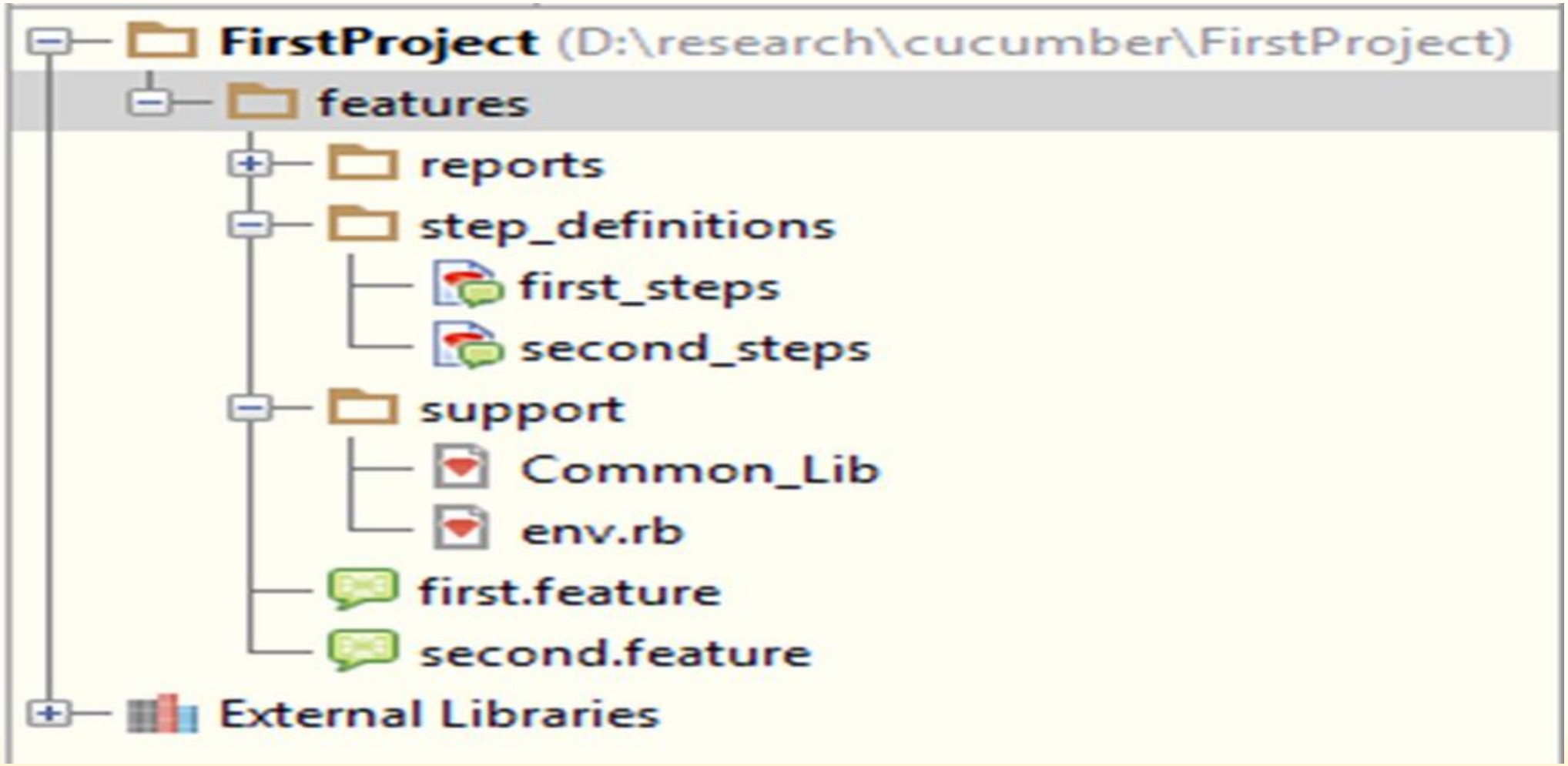
Cucumber with Browser

GTS EDUTECH

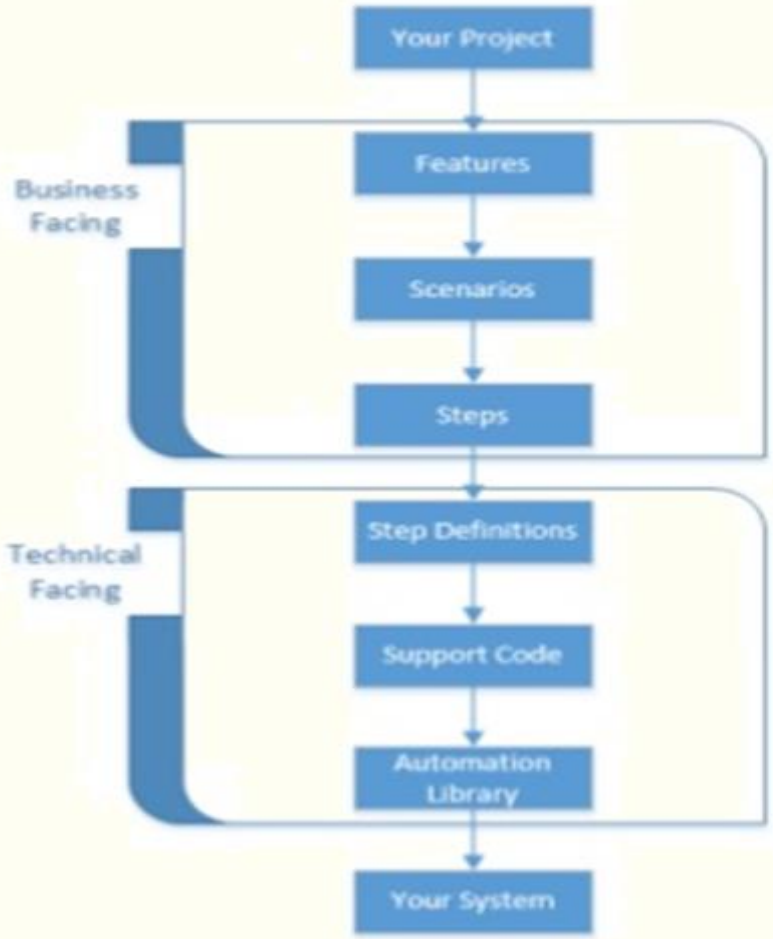


Cucumber Project Structure

GTS EDUTECH



Cucumber Project Structure

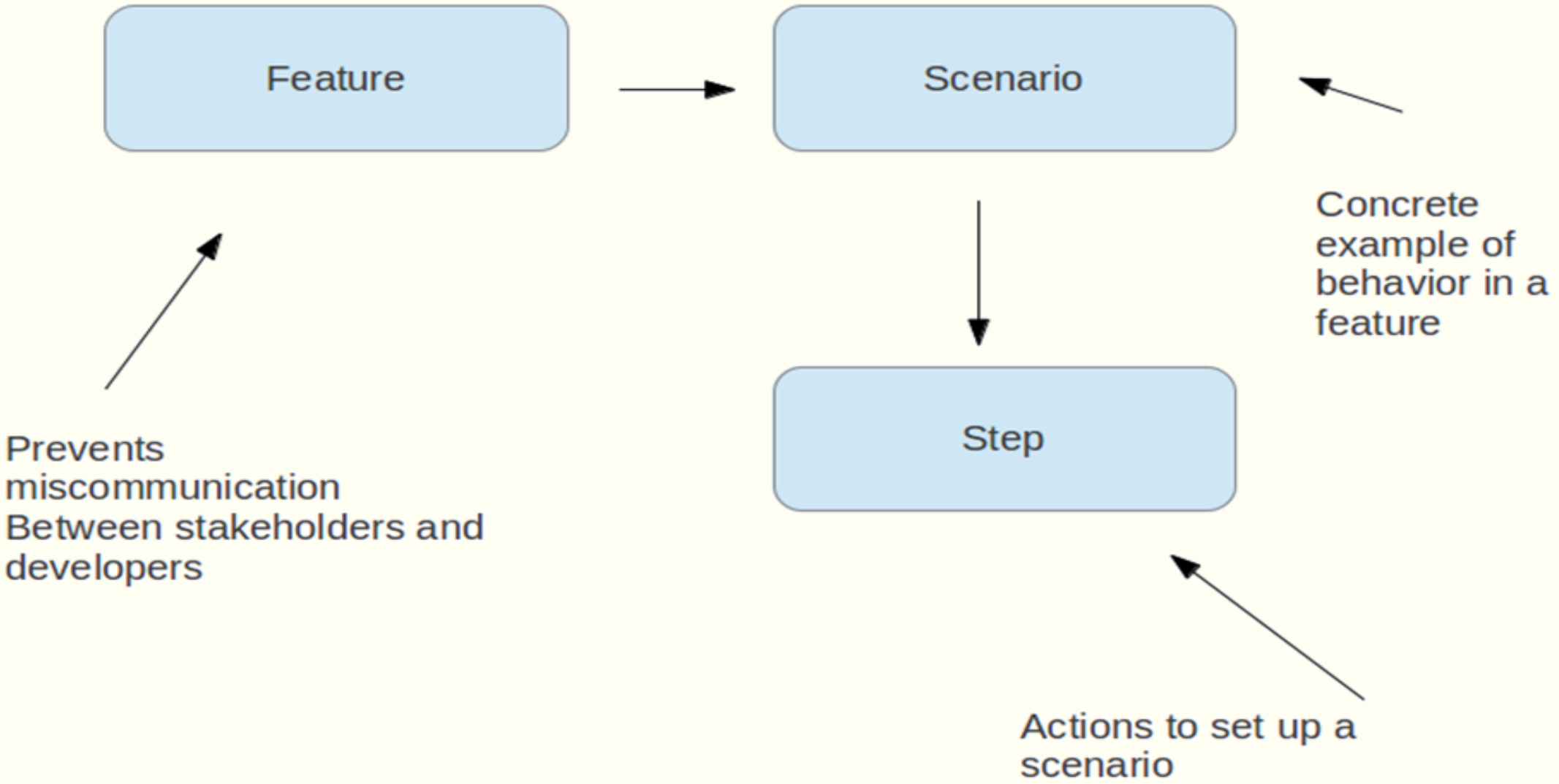



```
Feature: login to the system.
  As a user, I want to login into the system when I provide username and
  password.

  Scenario: Verify that user can login gmail
  Given     I launch "https://accounts.google.com" page
  When      I fill in "Email " with "kms.automation.2013@gmail.com"
  And       I fill in "Passwd" with "kms@2013"
  And       I click on "signin" button
  Then      I am on the "Home" page
```

```
Given /^I launch "([^\"]*)" page$/ do |page|
  visit(page)
End

When /^I fill in "([^\"]*)" with "([^\"]*)"$/ do |field, value|
  fill_in field, :with => value
end
```



Feature & Feature File

GTS EDUTECH

- **Feature** gives information about 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.
- This part is basically kept brief.
- Every '.feature' file conventionally consists of a single feature.
- A feature usually contains a list of scenarios.
- You can write whatever you want up until the first scenario, which starts with the word Scenario on a new line.

What is Scenario?

GTS EDUTECH

- Basically a scenario represents a particular functionality under test.
- By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about.
- Each scenario should follow given, when and then format.
- Scenario is one of the core Gherkin structures.
- Each feature can have one or more scenarios, and every scenario consists of one or more steps.

Example

```
# feature/hello_cucumber.feature
Feature: Hello Cucumber
As a product manager
I want our users to be greeted when they visit our site
So that they have a better experience

Scenario: User sees the welcome message
When I go to the homepage
Then I should see the welcome message
```

Actions to set up a scenario

GTS EDUTECH

The following scenarios each have 3 steps:

Scenario: Wilson posts to his own blog
Given I am logged in as Wilson
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

Scenario: Wilson fails to post to somebody else's blog
Given I am logged in as Wilson
When I try to post to "Greg's anti-tax rants"
Then I should see "Hey! That's not your blog!"

Scenario: Greg posts to a client's blog
Given I am logged in as Greg
When I try to post to "Expensive Therapy"
Then I should see "Your article was published."

Steps & Step Definition

GTS EDUTECH

- Scenario instructions are called **Steps**
- **Step definition** maps the test case steps in the feature files (introduced by Given/When/Then) to code, which executes and checks the outcomes from the system under test.
- For a step definition to be executed, it must match the given compoent in a feature. Step definition is defined in ruby files under "features/step_definitions/*_steps.java".

Installation

GTS EDUTECH

- Install Java 1.8 and above
- Install Eclipse
- Install Cucumber and Naturals plugin into Eclipse
- Download Selenium Standalone Server
- Set below path
  - JAVA_HOME
  - JRE_HOME
  - M2

Installation

GTS EDUTECH

- Download jar files
  - Cucumber-core
  - Cucumber-html
  - Cucumber-java
  - Cucumber-junit
  - Cucumber-jvm-deps
  - Cucumber-reporting
  - Hemcrest-core
  - Gherkin
  - Junit/TestNG
  - mockito-all
  - cobertura code coverage

Cucumber Keywords

GTS EDUTECH

- ***Background*** in cucumber is a concept that allows you to specify steps that are pre-requisite to all the scenarios in a given feature file.
- ***Feature*** A feature would describe the current test script which has to be executed.
- ***Scenario*** Scenario describes the steps and expected outcome for a particular test case.
- ***Scenario Outline*** Same scenario can be executed for multiple sets of data using scenario outline. The data is provided by a tabular structure separated by (||).
- ***Given*** It specifies the context of the text to be executed. By using data tables "Given", step can also be parameterized.
- ***When*** "When" specifies the test action that has to performed
- ***Then*** The expected outcome of the test can be represented by "Then"

```
Feature:
  As a user
  I want to be able to add new clients in the system
  So that i can add accounting data for that client

  Background:
    Given the user is on landing page
    When she chooses to sign up

  Scenario: Sign up a new user
    And she provides the first name as Sukesh
    And she provides the last name as Kumar
    And she provides the email as validemail@aq.com
    And she provides the password as password
    And she provides the confirm password again as password
    And she signs-up
    Then she should be logged in to the application


  Scenario Outline: Data driving new user sign-up
    And she provides the first name as <firstName>
    And she provides the last name as <lastName>
    And she provides the email as <email>
    And she provides the password as <password>
    And she provides the confirm password again as <password>
    And she signs-up
    Then she should be logged in to the application
  Examples:
    | firstName | lastName | email              | password |
    | Sukesh    | Kumar    | validemail@aq.com | password |
```

GTS EDUTECH

Cucumber Keywords

GTS EDUTECH

- ***Data Tables*** is a set of test data written in the form of table (rows and columns). "|" used as column data separator in Data Table. Data tables is used to pass more than one set of test data into the tests in order to achieve data driven test. The table can easily be converted to a list or a map that you can use in your step.

```
Given the following users exist:
  | name   | email               | twitter         |
  | Aslak  | aslak@cucumber.io   | @aslak_hellesoy |
  | Julien | julien@cucumber.io  | @jbpros         |
  | Matt   | matt@cucumber.io    | @mattwynne      |
```

Cucumber Keywords

GTS EDUTECH

- ***Doc Strings*** are handy for passing a larger piece of text to a step definition. The syntax is inspired from Python's Doc String syntax. The text should be offset by delimiters consisting of three double-quote marks on lines of their own.

```
Given a blog post named "Random" with Markdown body
  """
  Some Title, Eh?
  ===============
  Here is the first paragraph of my blog post. Lorem ipsum dolor sit amet,
  consectetur adipiscing elit.
  """
```

- ***Step Arguments*** - In some cases you might want to pass a larger chunk of text or a table of data to a step—something that doesn't fit on a single line.

```
Given /^I have (.*) cucumbers in my belly$/ do |cukes|
```

Cucumber Keywords

GTS EDUTECH

- ***Comments*** Gherkin provides lots of places to document your features and scenarios. The preferred place is descriptions. Choosing good names is also useful. If none of these places suit you, you can start a line with a # to tell Cucumber that the remainder of the line is a comment, and shouldn't be executed.

```
# Hi.
Feature: some feature
  Comments can be (almost) everywhere.
  # And another comment

  Scenario: some scenario
    # Yet another one.
    When a step runs
    Then a step runs
    # Oh my, comments are everywhere.
# Thank you for reading.
```

Tags

GTS EDUTECH

- Tags are a great way to organize your features and scenarios. Consider this example:

```
@billing Feature: Verify billing
@important Scenario: Missing product description
Scenario: Several products
```

- A Scenario or feature can have as many tags as you like. Just separate them with spaces:

```
@billing @bicker @annoy
Feature: Verify billing
```

- Tags are also a great way to “link” your Cucumber features to other documents such as excel, word, html, etc.

Hooks

GTS EDUTECH

- Cucumber provides a number of hooks which allow you to configure the environment for your application. By default hooks are run for each scenario, but you can use tagged hooks if you want more fine grained control.
- **Scenario hooks**
  - @Before
  - @After

```
public class StartingSteps extends DriverFactory {


    @Given("^the user is on landing page$")
    public void setup() throws Throwable {
        driver.get("http://accountsdemo.herokuapp.com");
        driver.manage().window().maximize();
    }

    @Before
    public void beforeScenario(){
        System.out.println("this will run before the actual scenario");
    }

    @After
    public void afterScenario(){
        System.out.println("this will run after scneario is finished, even if it failed");
    }

}
```

Hooks

GTS EDUTECH

- @Around

```
Around('@fast') do |scenario, block|
  Timeout.timeout(0.5) do
    block.call
  end
end
```

- **Step hooks**
  - After Step hook does not work with scenarios which have backgrounds

```
AfterStep do |scenario|
  # Do something after each step.
end
```

Hooks

GTS EDUTECH

- **Tagged hooks**
  - Sometimes you may want a certain hook to run only for certain scenarios. This can be achieved by associating a Before, After, Around or AfterStep hook with one or more tags. You can OR and AND tags in much the same way as you can when running Cucumber from the command line.

  - For OR tags, pass the tags in a single string comma separated:

```
Before('@cucumis, @sativus') do
  # This will only run before scenarios tagged
  # with @cucumis OR @sativus.
end
```

Hooks

GTS EDUTECH

- For AND tags, pass the tags as separate tag strings:

```
Before('@cucumis', '~@sativus') do
  # This will only run before scenarios tagged
  # with @cucumis AND NOT @sativus.
end
```

- Complex tag conditions using both OR and AND on tags:

```
Before('@cucumis, @sativus', '@aqua') do
  # This will only run before scenarios tagged
  # with (@cucumis OR @sativus) AND @aqua
end
```

Hooks

GTS EDUTECH

- After Step example:

```
AfterStep('@cucumis', '@sativus') do
  # This will only run after steps within scenarios tagged
  # with @cucumis AND @sativus.
end
```

- **Global hooks**
  - If you want something to happen once before any scenario is run - just put that code at the top-level in your env.rb file (or any other file in your features/support directory. Use Kernel#at_exit for global teardown.

```
my_heavy_object = HeavyObject.new
my_heavy_object.do_it

at_exit do
  my_heavy_object.undo_it
end
```

Hooks

GTS EDUTECH

- **Running a Before hook only once**
  - If you have a hook you only want to run once, use a global variable:

```
Before do
  $dunit ||= false  # have to define a variable before we can reference its value
  return $dunit if $dunit                    # bail if $dunit TRUE
  step "run the really slow log in method" # otherwise do it.
  $dunit = true                              # don't do it again.
end
```

- **AfterConfiguration**
  - You may also provide an AfterConfiguration hook that will be run after Cucumber has been configured. The block you provide will be passed the cucumber configuration (an instance of Cucumber::Cli::Configuration).
  - This hook will run only once; after support has been loaded but before features are loaded.

```
AfterConfiguration do |config|
  puts "Features dwell in #{config.feature_dirs}"
end
```

Non GUI mode

GTS EDUTECH

- mvn test -Dcucumber.options='--tags "@smoke and @fast"

GTS EDUTECH