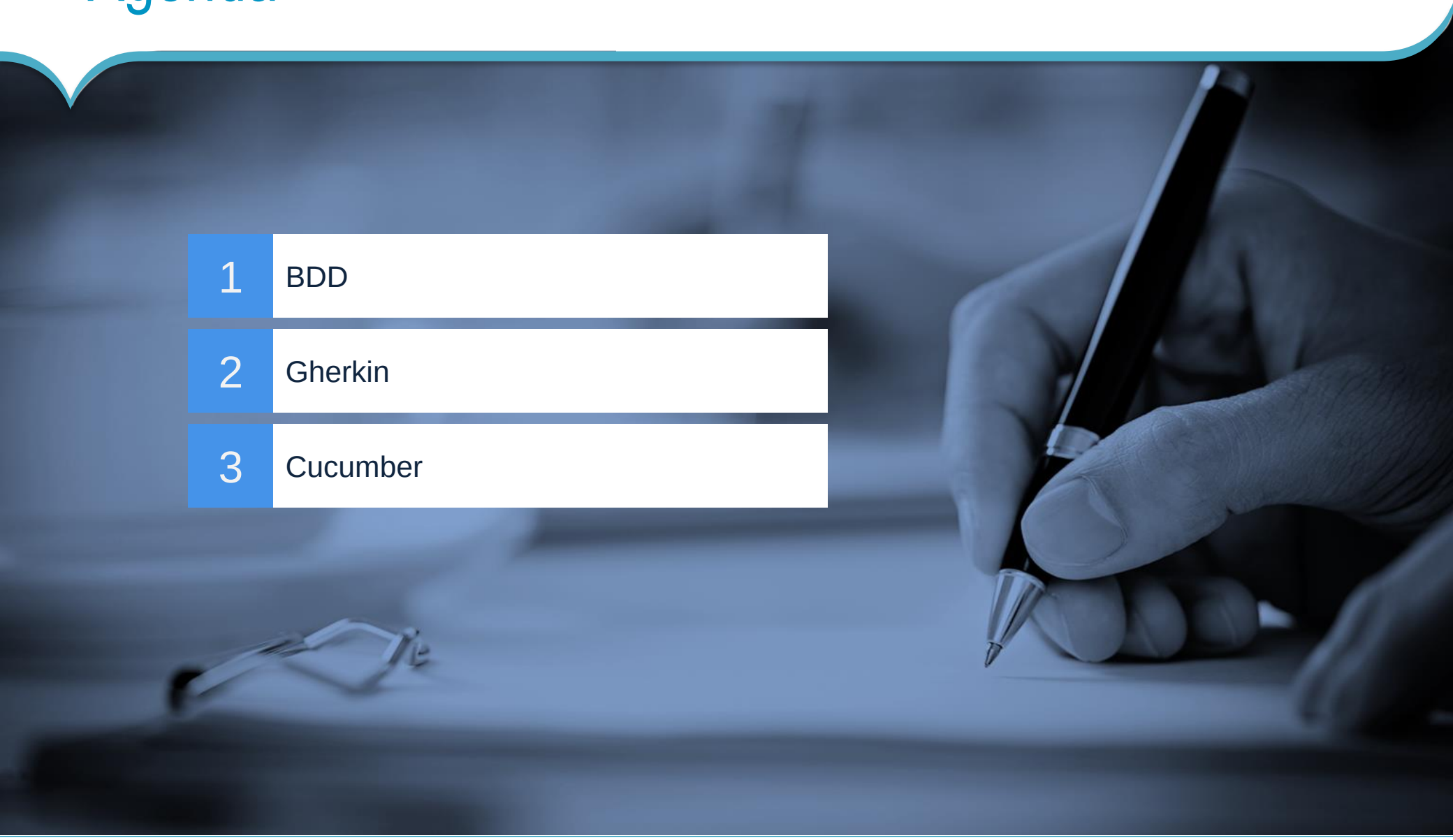
Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

BDD

2017

People matter, results count.

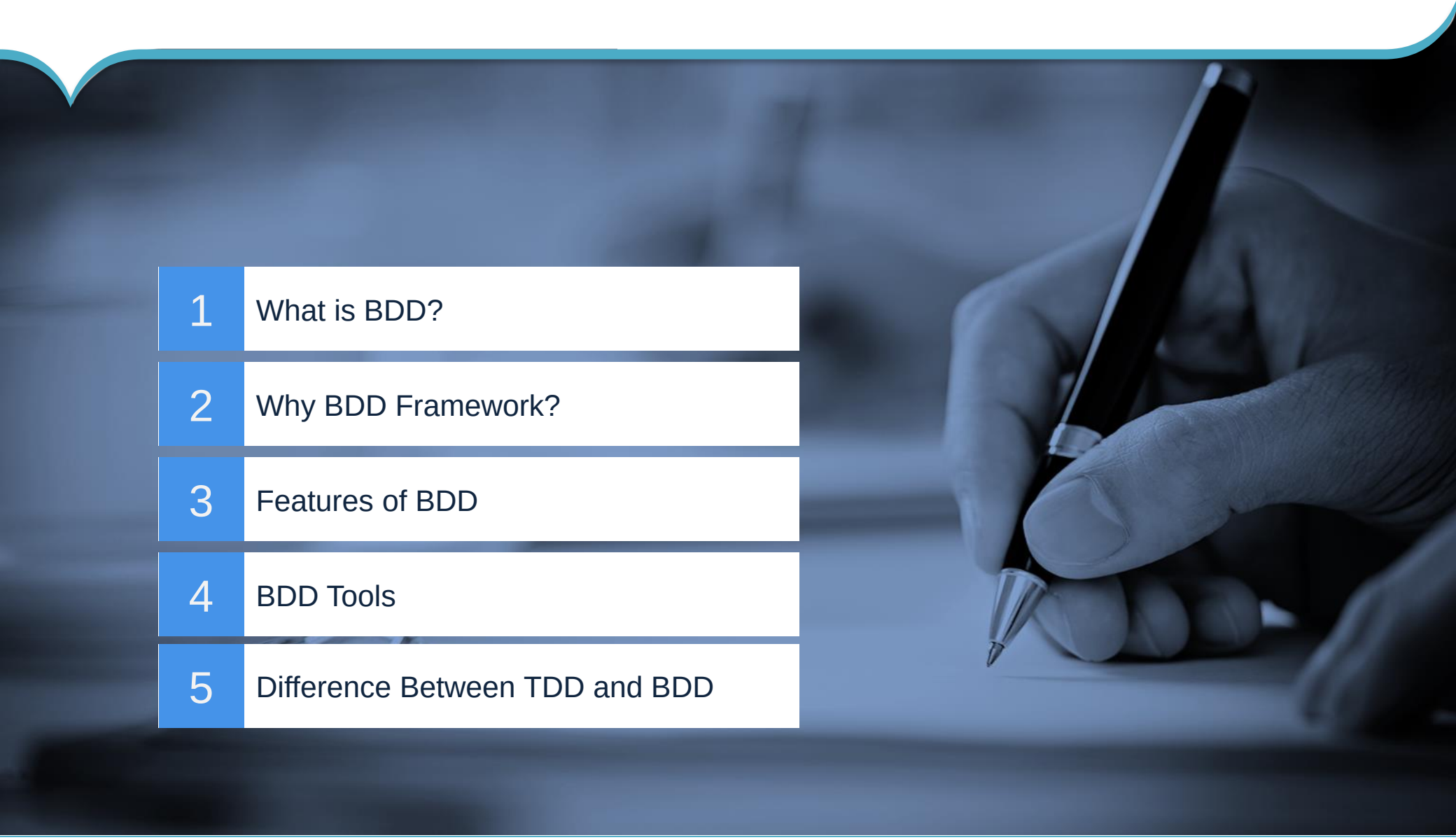
Agenda



1. BDD
2. Gherkin
3. Cucumber

Behavior Driven Development

Outline



1. What is BDD?
2. Why BDD Framework?
3. Features of BDD
4. BDD Tools
5. Difference Between TDD and BDD

What are BDD?

- Behavior-driven development (**BDD**) is a software development methodology in which an application is specified and designed by describing how its behavior should appear to an outside observer.
- Behavior Driven testing is an extension of TDD. Like in TDD in BDD also we write tests first and the add application code. The major difference that we get to see here are
  - *Tests are written in plain descriptive English type grammar*
  - *Tests are explained as behavior of application and are more user focused*
  - *Using examples to clarify requirements*

Why BDD Frameworks

- Consider : E-Commerce website -with implementing some new features on the website.
- Challenge: development team is to convert the client idea in to something that actually delivers the benefits to client.
- Person who is developing the idea is not the same person who has this idea.
- Idea needs to be communicated and has to travel from Business Owners(Client) to the development teams or many other people.
- High quality communication.
- Help of Gherkin language cucumber helps facilitate the discovery and use of a ubiquitous language within the team.
- Tests written in cucumber directly interact with the development code, but the tests are written in a language that is quite easy to understand by the business stakeholders.
- Cucumber test removes many misunderstandings long before they create any ambiguities in to the code.

Why BDD Frameworks

- **Example of a Cucumber/SpecFlow/BDD Test:**

Feature: Sign up

Sign up should be quick and friendly.

Scenario: Successful sign up

New users should get a confirmation email and be greeted personally by the site once signed in.

Given I have chosen to sign up
When I sign up with valid details
Then I should receive a confirmation email
And I should see a personalized greeting message

Scenario: Duplicate email

Where someone tries to create an account for an email address that already exists.

Given I have chosen to sign up
But I enter an email address that has already registered
Then I should be told that the email is already registered
And I should be offered the option to recover my password

Features of BDD

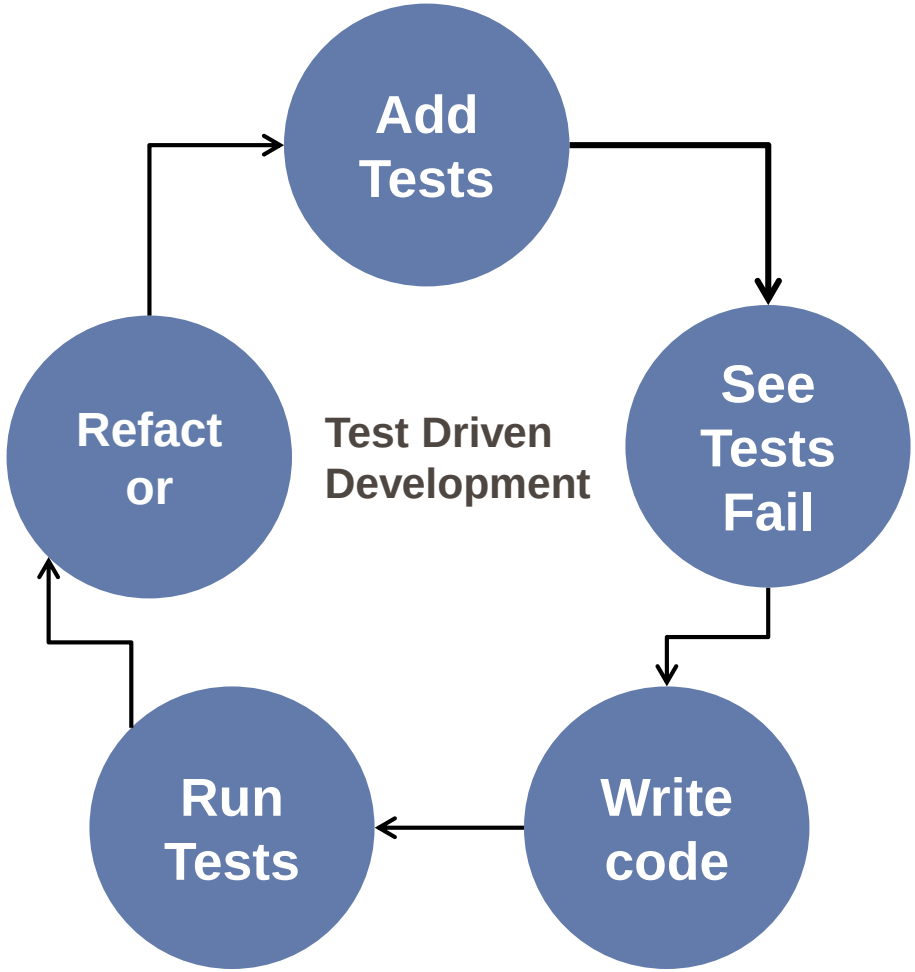
- Shifting from thinking in “tests” to thinking in “behavior”
- Collaboration between Business stakeholders, Business Analysts, QA Team and developers
- Ubiquitous language, it is easy to describe
- Driven by Business Value
- Extends Test Driven Development (TDD) by utilizing natural language that non technical stakeholders can understand
- BDD frameworks such as **Cucumbe**r or **JBehave** are an enabler, acting a “bridge” between Business & Technical Language

BDD Tools

- ***What is Cucumber?***
  - ***Cucumber*** is a testing framework which supports ***Behavior Driven Development (BDD).*** It lets us define application behavior in plain meaningful English text using a simple grammar defined by a language called ***Gherkin***. Cucumber itself is written in ***Ruby***, but it can be used to "test" code written in *Ruby* or other languages including but not limited to *Java*, *C#* and *Python.*
-
- ***What is SpecFlow?***
  - ***SpecFlow*** is inspired by *Cucumber* framework in the Ruby on Rails world. *Cucumber* uses plain English in the Gherkin format to express user stories. Once the user stories and their expectations are written, the Cucumber gem is used to execute those stores. ***SpecFlow brings the same concept to the .NET world*** and allows the developer to express the feature in plain English language. It also allows to write specification in human readable ***Gherkin format***.

Difference Between TDD and BDD

- **TDD:**
  - First the developer writes some tests.
  - The developer then runs those tests and (obviously) they fail because none of those features are actually implemented.
  - Next the developer actually implements those tests in code.
  - If the developer writes their code well, then the in next stage they will see that their tests pass.
  - The developer can then refactor their code, add comments and clean it up, as they wish because the developer knows that if the new code breaks something, then the tests will be an alert by failing.



Difference Between TDD and BDD

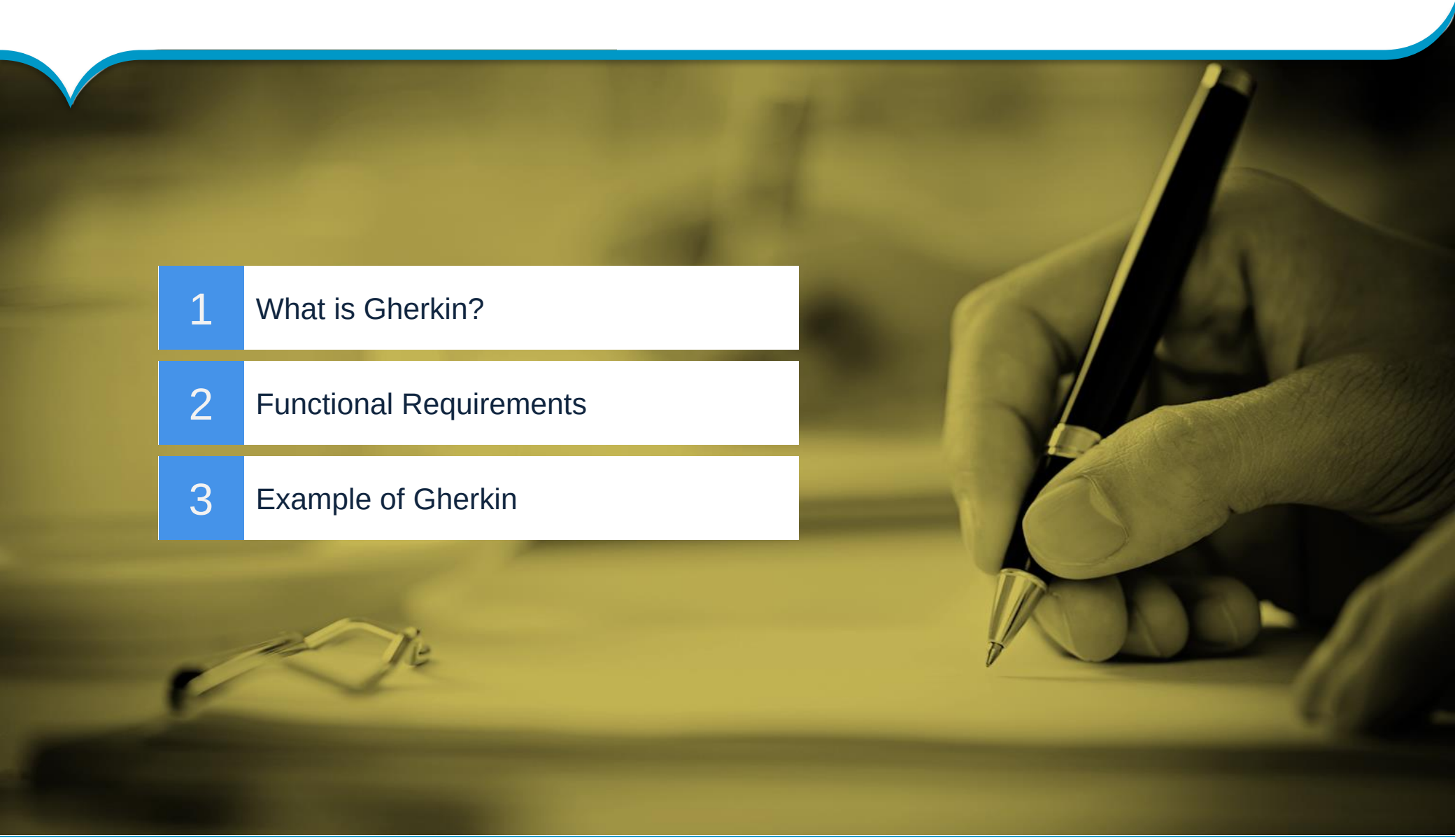
- **BDD:**
  - BDD is meant to eliminate issues that TDD might cause.
  - BDD is when we write *behavior & specification* that then drives our software development.
  - Behavior & specification might seem awfully similar to tests but the difference is very subtle and important.

Difference Between TDD and BDD

- **TDD vs BDD**
  - The choice between TDD and BDD is a complicated one. It depends on if there is an appropriate testing framework for your given target language, what your coworkers are comfortable with, and sometimes other factors.
  - Some argue that BDD is always better than TDD because it has the possibility of eliminating issues that might arise when using TDD.
  - The key to BDD is that it **might** prevent issues; it isn't guaranteed to. Issues like poor code organization, bad design practices, etc. will still persist. You'll just have a lower likely hood of writing bad tests and thus have more robust features.

Gherkin Language

Outline



1. What is Gherkin?
2. Functional Requirements
3. Example of Gherkin

What is Gherkin?

- Common language across different domains of project.
  - ***Clients***, ***Developers***, ***Testers***, ***Business analysts*** and the ***Managerial*** team.
- Developer and Tester have task of collaborating with the business team (*Business owners and Business analysts*).
- Come up with the requirements of your project.
- These requirements will be what your development team will be implementing and test team will be testing.
- Requirement given by business team are very crude and basic.

Functional Requirements

- ***Search Functionality***
  - ***User should be able to search for a product***
  - ***Only the products related to search string should be displayed.***
- *Questions raised from the above requirements*
  - *– What is the maximum searchable length of search string?*
  - *– What should be the search results if user searches for an invalid product?*
  - *– What are the valid characters that can be used to search?*
  - ***and similarly a few more detailed behavior of the application.

- ***Question to Business Owner :*** *What should be the search results if user searches for an invalid product?*
- ***Reply from Business Owner :*** *Invalid product searches should show following text on the search page:* ***No product found***

Functional Requirements

- ***What Happens?***
  - *Business team and the technical teams are communicating at two different levels, business team being vague and technical team trying to be precise.*
  - *Ambiguity being introduced in the system, here by the definition of "invalid product".*
  - *Not enough insight being given to the Business team, so that they could have come up with new scenarios.*
  - *Some details of project being lost in emails and telephonic conversations.*

Functional Requirements

- ***How to improve the Requirements?***
  - *"When a user searches, without spelling mistake, for a product name present in the inventory. All the products with similar name should be displayed"*
  - *"When a user searches, without spelling mistake, for a product name present in the inventory. Search results should be displayed with exact matches first and then similar matches"*

Functional Requirements

- ***What have we achieved here?***
  - Forced the client to think in terms of details.
  - With this improved thinking Business teams are coming with more refined requirements.
  - This in turn with reduces the ambiguity in the project and will make developers and testers life easy by reducing the number of incorrect implementations.
  - Each requirement now documents one exact behavior of the application.
  - It can be considered as a requirement document in itself.

Functional Requirements

- ***What is the conclusion?***
  - Different teams in the project need a common language to express requirements. This language should be simple enough to be understood by Business team members and should be explicit enough to remove most of the ambiguities for developers and testers.
  - This language should open up the thinking of team members to come up with more scenarios. As you express more details you try to visualize the system more and hence you end up making more user scenarios.
  - This language should be good enough to be used as project documentation.

Example of Gherkin

Feature: Search feature for users

This feature is very important because it will allow users to filter products

Scenario: When a user searches, without spelling mistake, for a product name present in inventory. All the products with similar name should be displayed

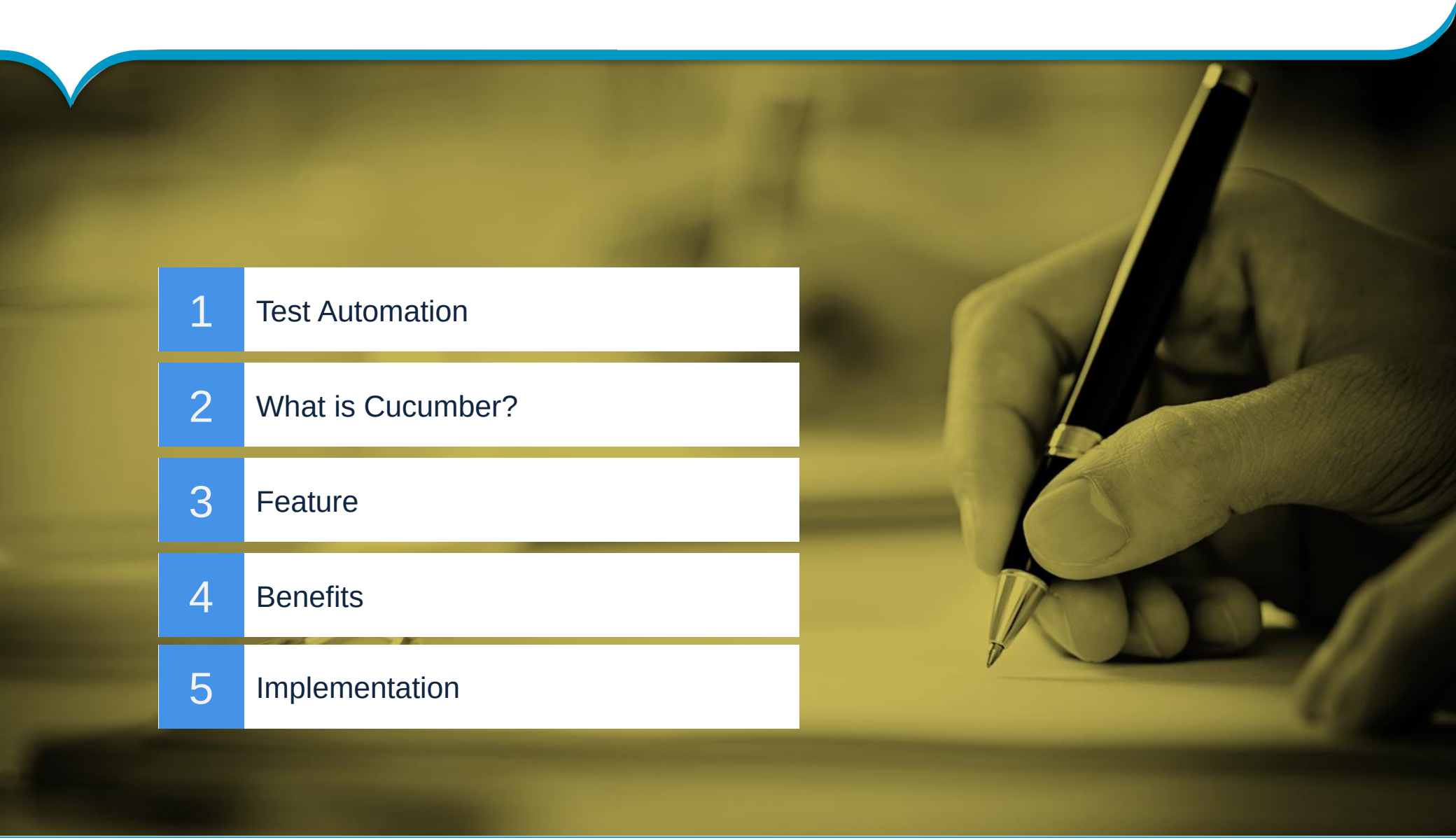
Given User is on the main page of www.myshopingsite.com

When User searches for laptops

Then search page should be updated with the lists of laptops

Cucumber

Outline



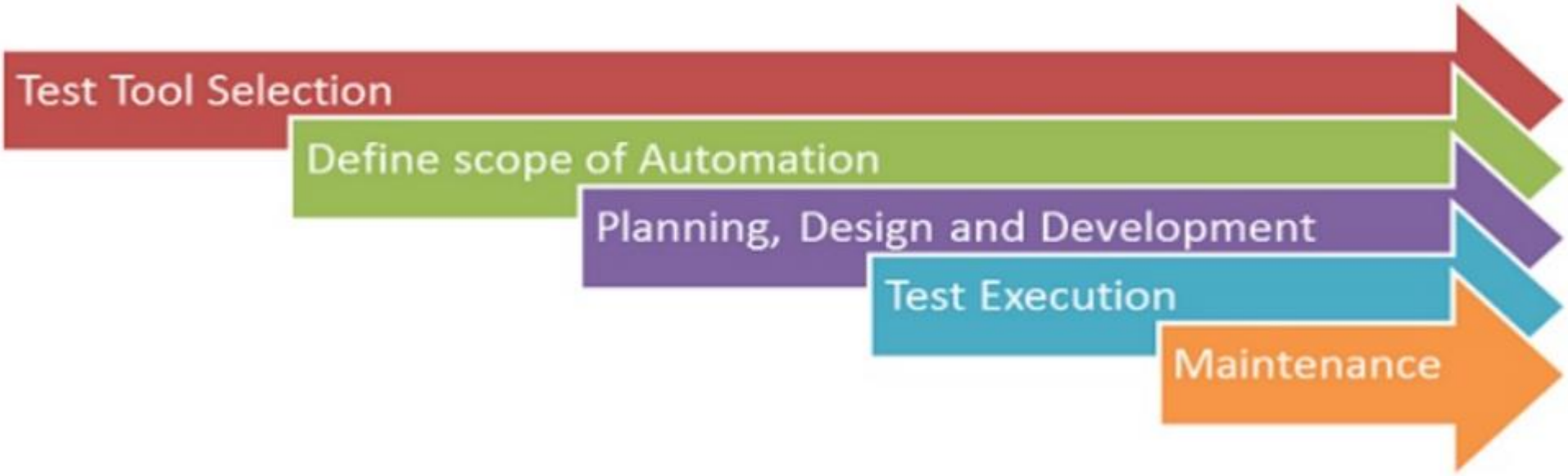
1. Test Automation
2. What is Cucumber?
3. Feature
4. Benefits
5. Implementation

Why Automation?

- Automated Testing is important due to the following reasons:
  - Manual Testing of all work flows, all fields, all negative scenarios is time and cost consuming.
  - Automation does not require Human intervention.
  - Automation increases speed of test execution.
  - Automation helps increase Test Coverage
  - Manual Testing can become boring and hence error prone.
- Benefits of Automation Testing:
  - Cost Reduction
  - The number of resources for regression test are reduced.
  - Automated tools run tests significantly faster than Human users.

Test Automation

- Automation Process:
  - Following steps are followed in an Automation Process:

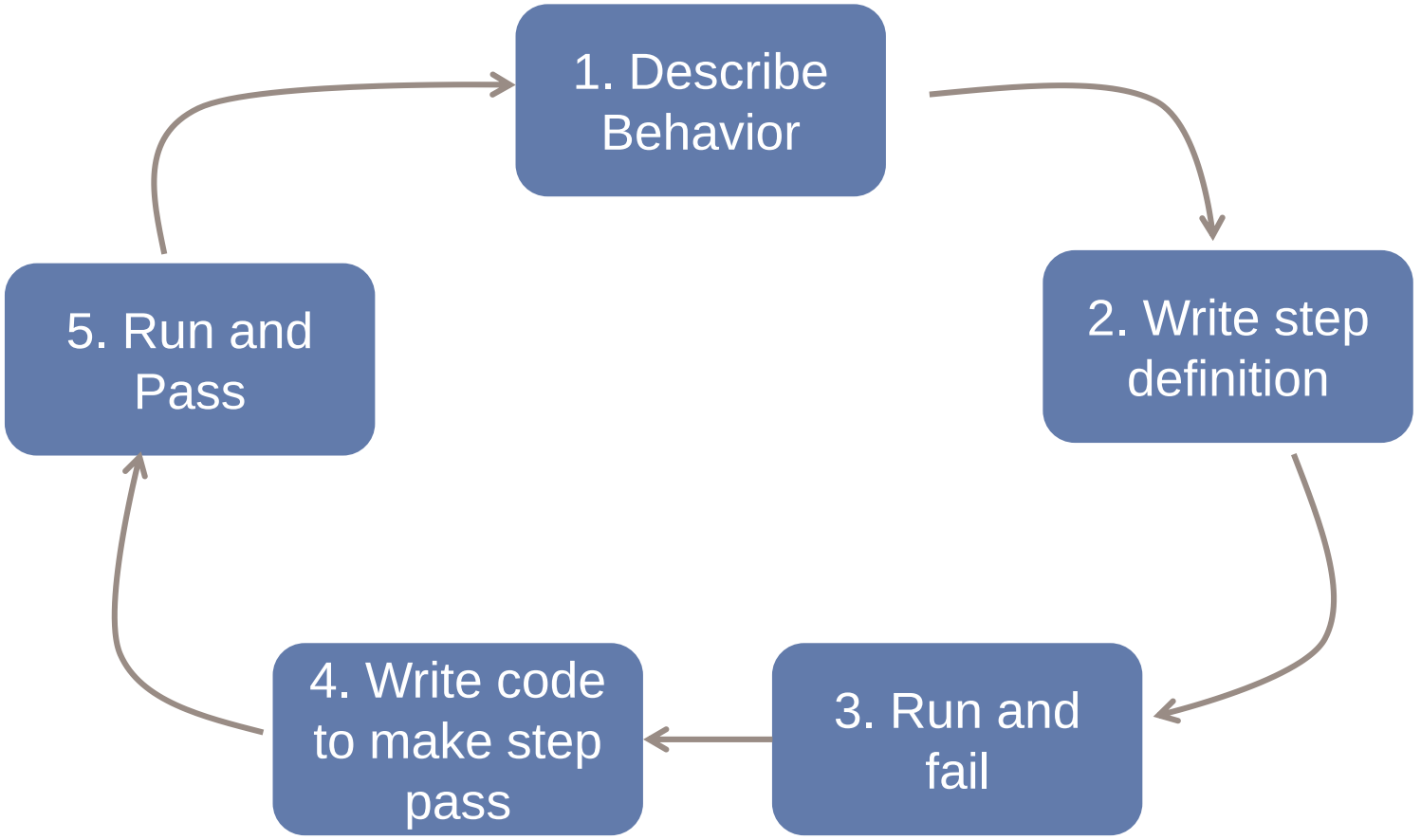


Cucumber

- Cucumber is a high-level testing framework that supports behavior driven development.
- It runs automated acceptance tests on web applications.
- Cucumber is a tool that executes plain-text functional description as automated tests.
- The language that Cucumber understands is called Gherkin.

*Describes the **behavior** of your system in a very understandable manner.*

Cucumber



Scenario:

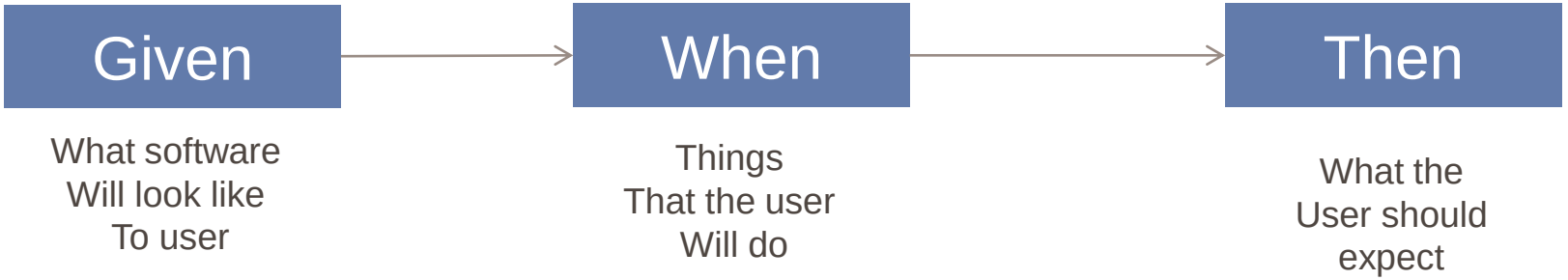
- Basically a scenario represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about. Each scenario should follow given, when and then format. This language is called as “gherkin”.

Scenario:

- **Given:** As mentioned above, given specifies the pre-conditions. It is basically a known state.
- **When**: This is used when some action is to be performed. As in above example we have seen when the user tries to log in using username and password, it becomes an action**.**
- **Then:** The expected outcome or result should be placed here. For Instance: verify the login is successful, successful page navigation.
- **Background:** Whenever any step is required to perform in each scenario then those steps needs to be placed in Background. For Instance: If user needs to clear database before each scenario then those steps can be put in background.
- **And**: And is used to combine two or more same type of action.

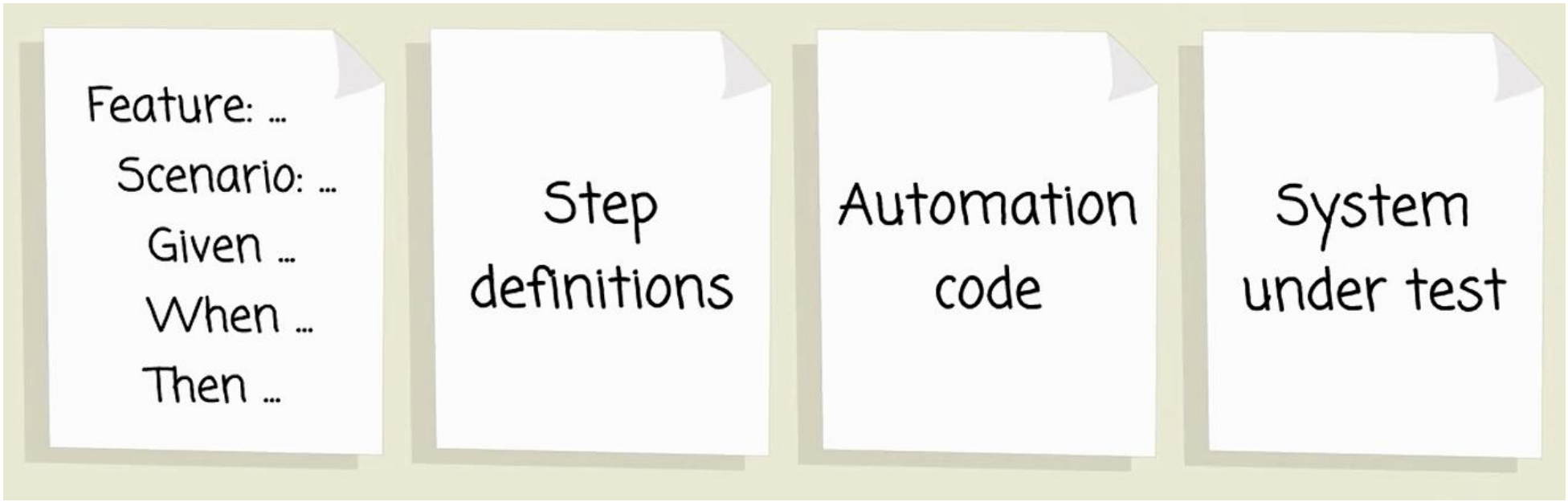
Feature

- Every .feature file conventionally consists of single feature. A line starting with the keyword Feature followed by free indented text starts a feature. A feature usually contains a list of scenarios. Scenarios together independent of your file and directory structure.



- **Given** The login page is opening
- **When** I input username, password and click Login button
- **Then** I am on the Home page

Feature

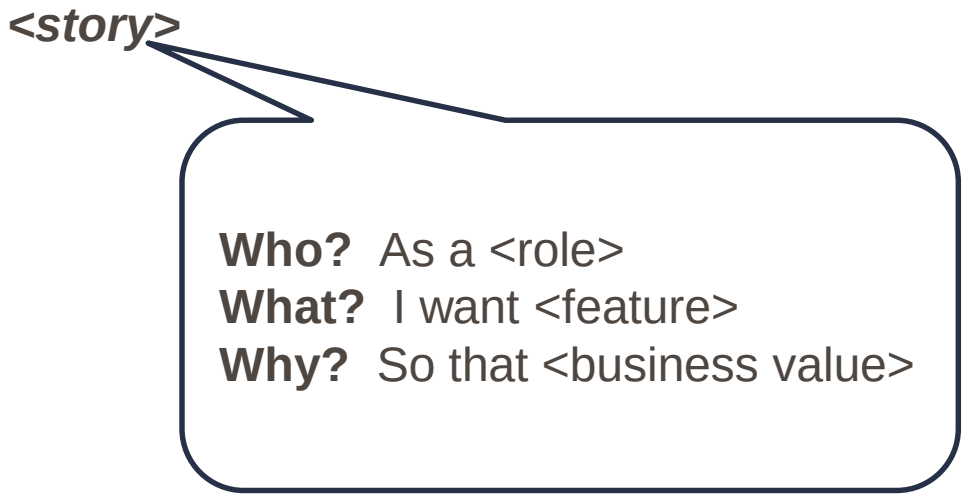


Cucumber

- **Cucumber Nomenclature**
  - **Feature** : Single file, ideally describing single feature.
  - **Scenario**: A test case
  - **Given-When-Then:** Test preconditions, Execution and Post conditions
  - **And**: Additional test constructs

Cucumber

- **Behaviour**
  - ***Feature: <short description>***



Cucumber

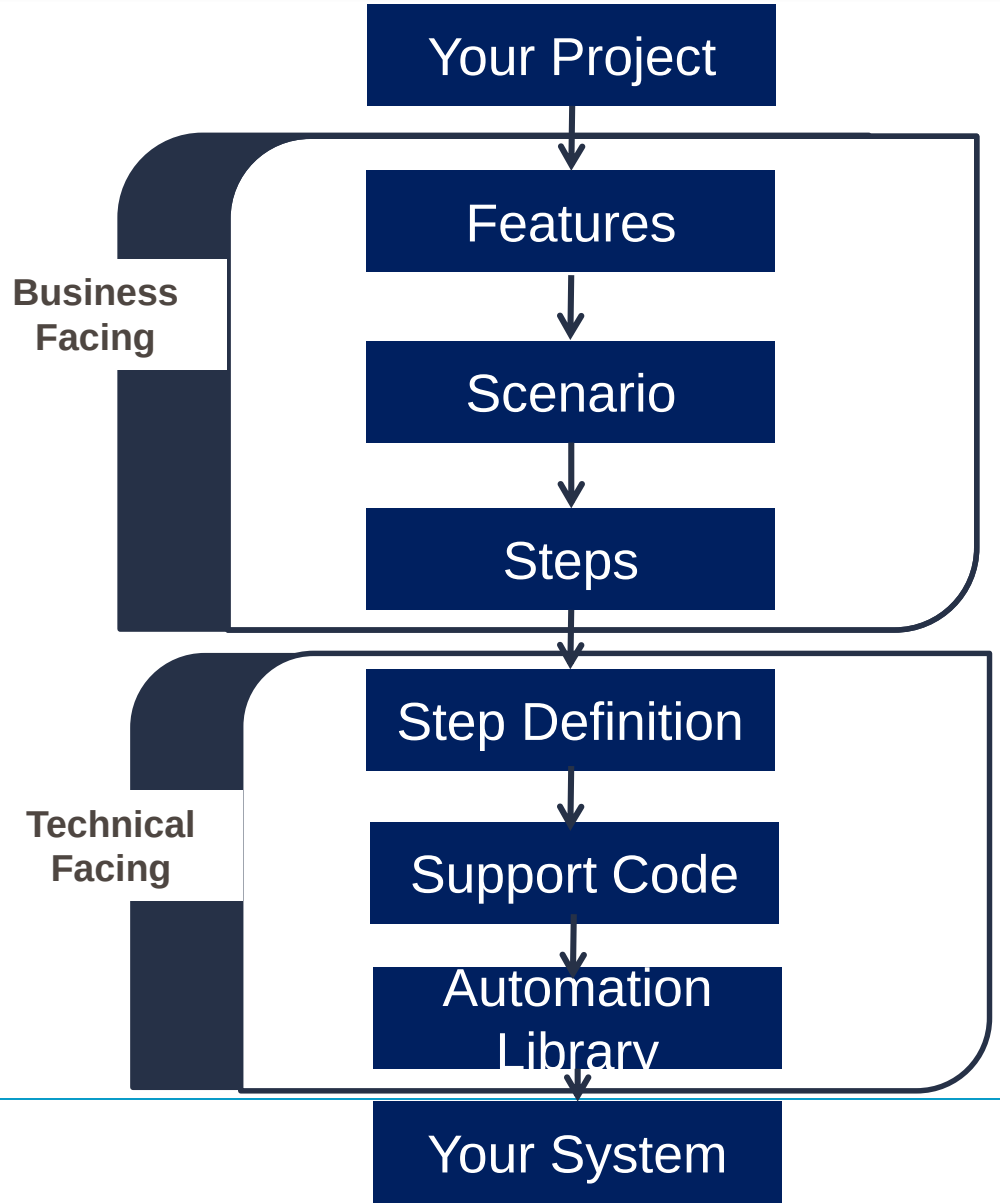
- **Scenario : <description>**

  <Scenario 1>

  <Scenario 2>

Given <preconditions, context>
[And] <additional preconditions>
When <action, behavior>
Then <postConditions>
[And] <additional postConditions>

Cucumber



Cucumber

- **Benefits of Cucumber**
  - **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 setup and execution**
  - **Efficient tool for testing**
  - **Instead of writing your tests purely in code, with Cucumber you start by writing a human-readable user story. Then, you write code to run the story and perform test(S) based on it.**

Cucumber

- Human Language Support

@admin_login

- Feature: Landing page
  - As a Admin
  - I want to see the landing page
  - So that I can login to the application

@smoke_login

Scenario: Click on customer login button for the first time
Given I am on Home page
When I click on Customer Login button
Then Login page for the user should be displayed
Given I login as a admin
When i click on Sign in button
then List view page for the user should be displayed
List view page is displayed

Cucumber

- **Summary**
  - ***Cucumber is a communication tool***
  - ***Works great for some classes of problems***
  - ***Doesn't work for some problems***

Lab Assignments for Cucumber

THANK YOU

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

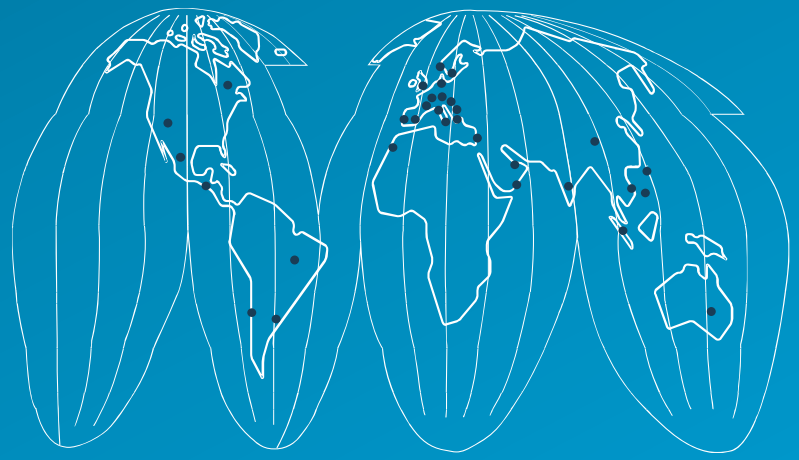
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com