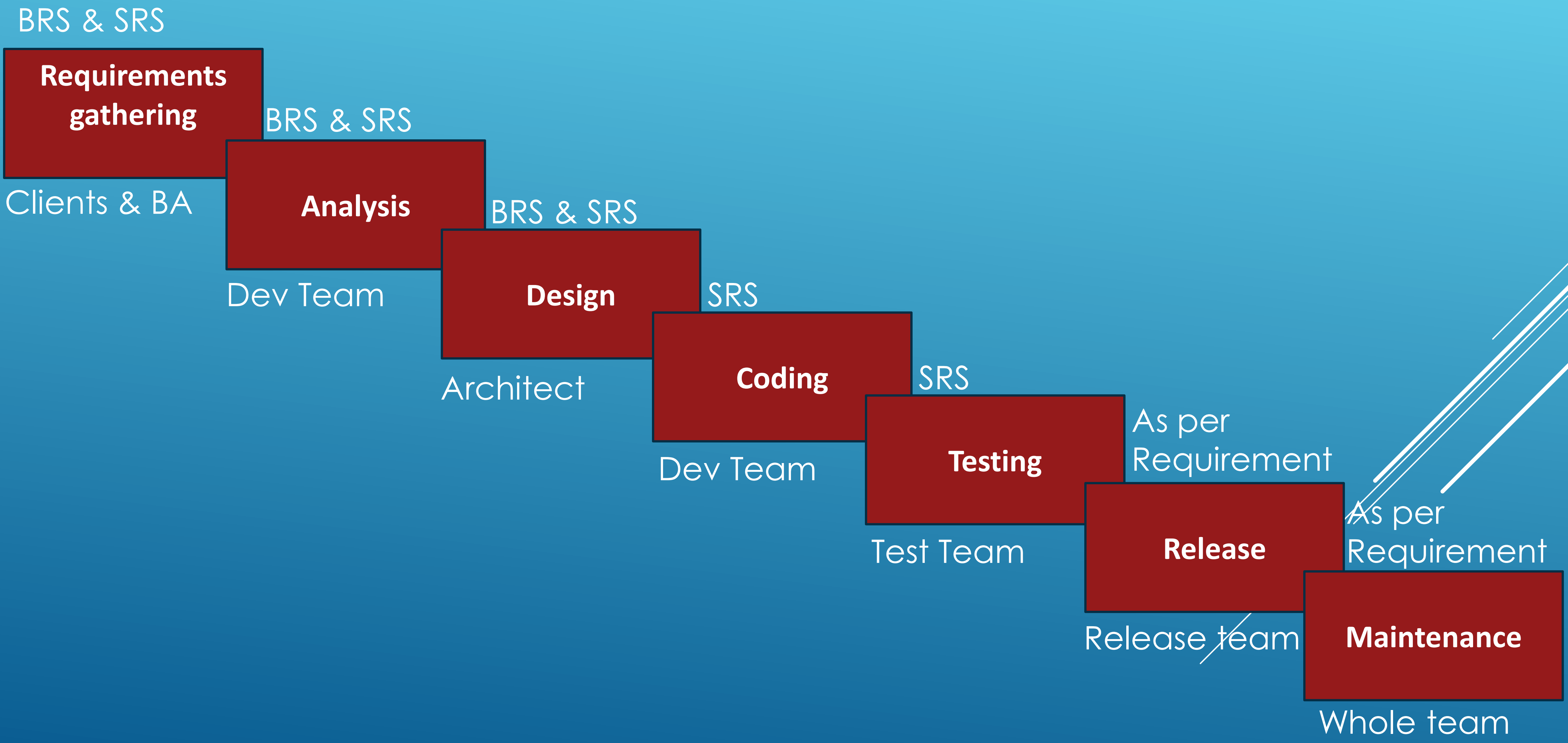
# TODAY'S AGENDA

- Agile
- Definition of Done
- TDD and BDD
- Exploratory and Ad-Hoc Testing
- Installations

# SDLC Software Development Life Cycle



BRS & SRS
**Requirements gathering**
Clients & BA

BRS & SRS
**Analysis**
Dev Team

BRS & SRS
**Design**
Architect

SRS
**Coding**
Dev Team

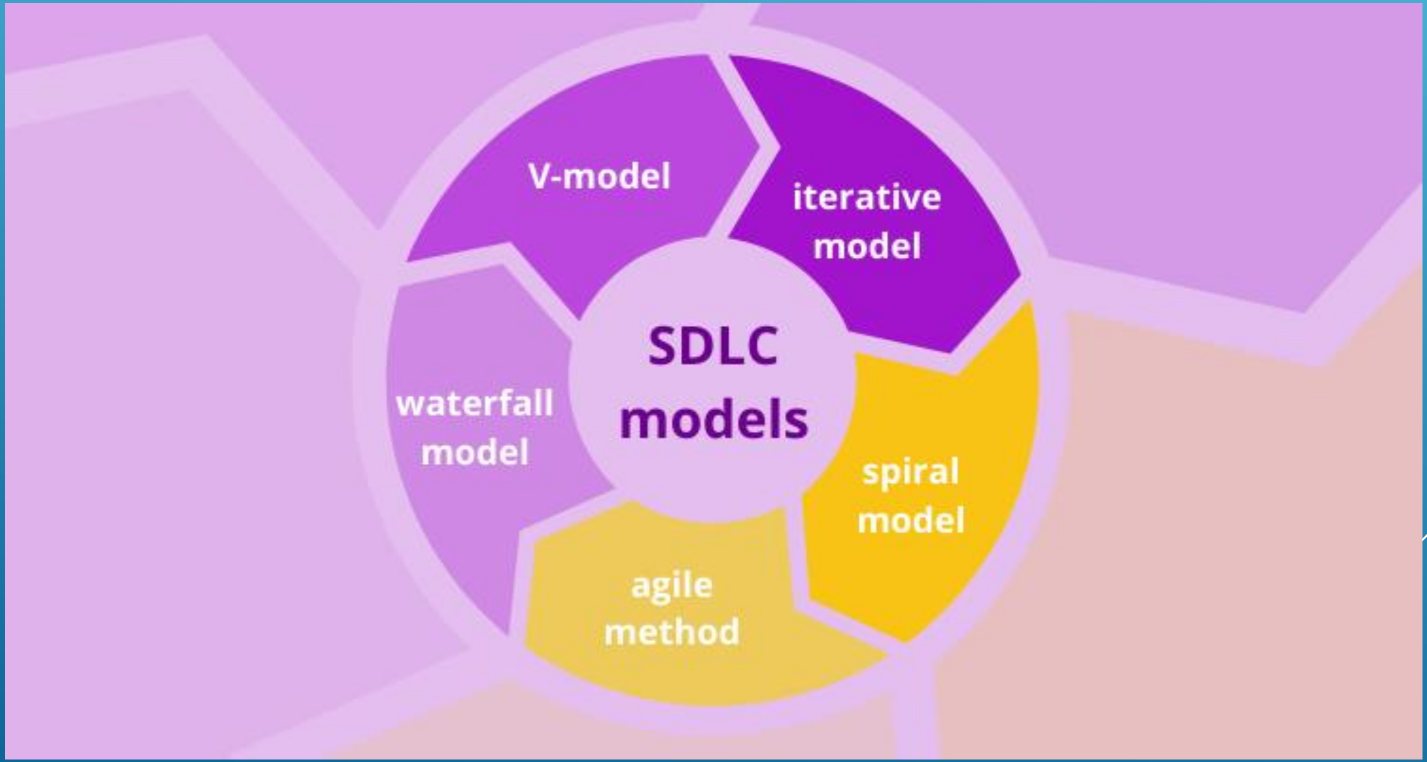
SRS
**Testing**
Test Team

As per Requirement
**Release**
Release team

As per Requirement
**Maintenance**
Whole team

# SOFTWARE DEVELOPMENT MODELS

Processes or methodologies that are being selected for the development of the project depending on the project's aims and goals.



# Agile Model

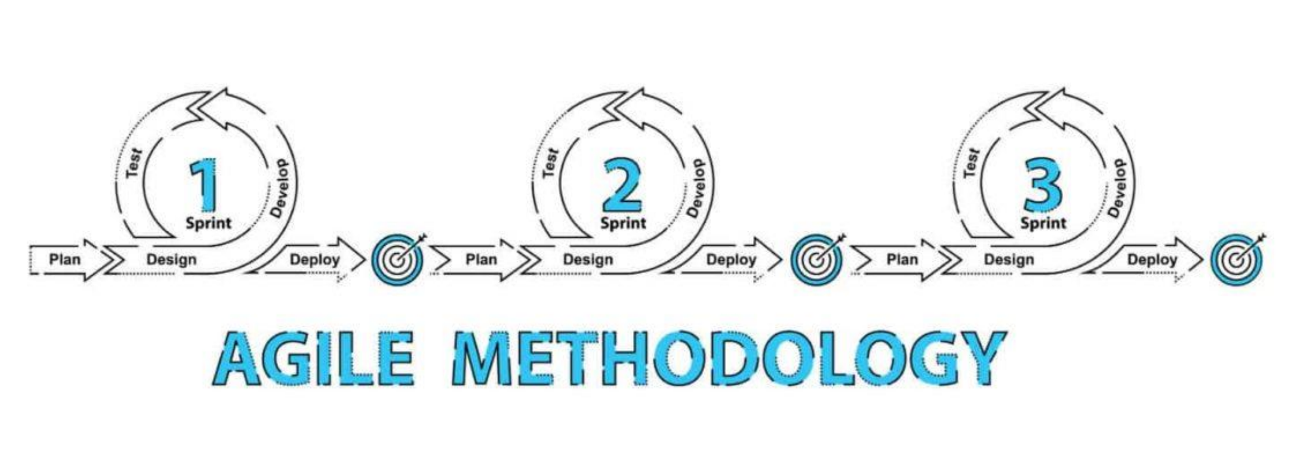
- Methodology that promotes continuous iteration and incremental of development and testing throughout the SDLC of the project.

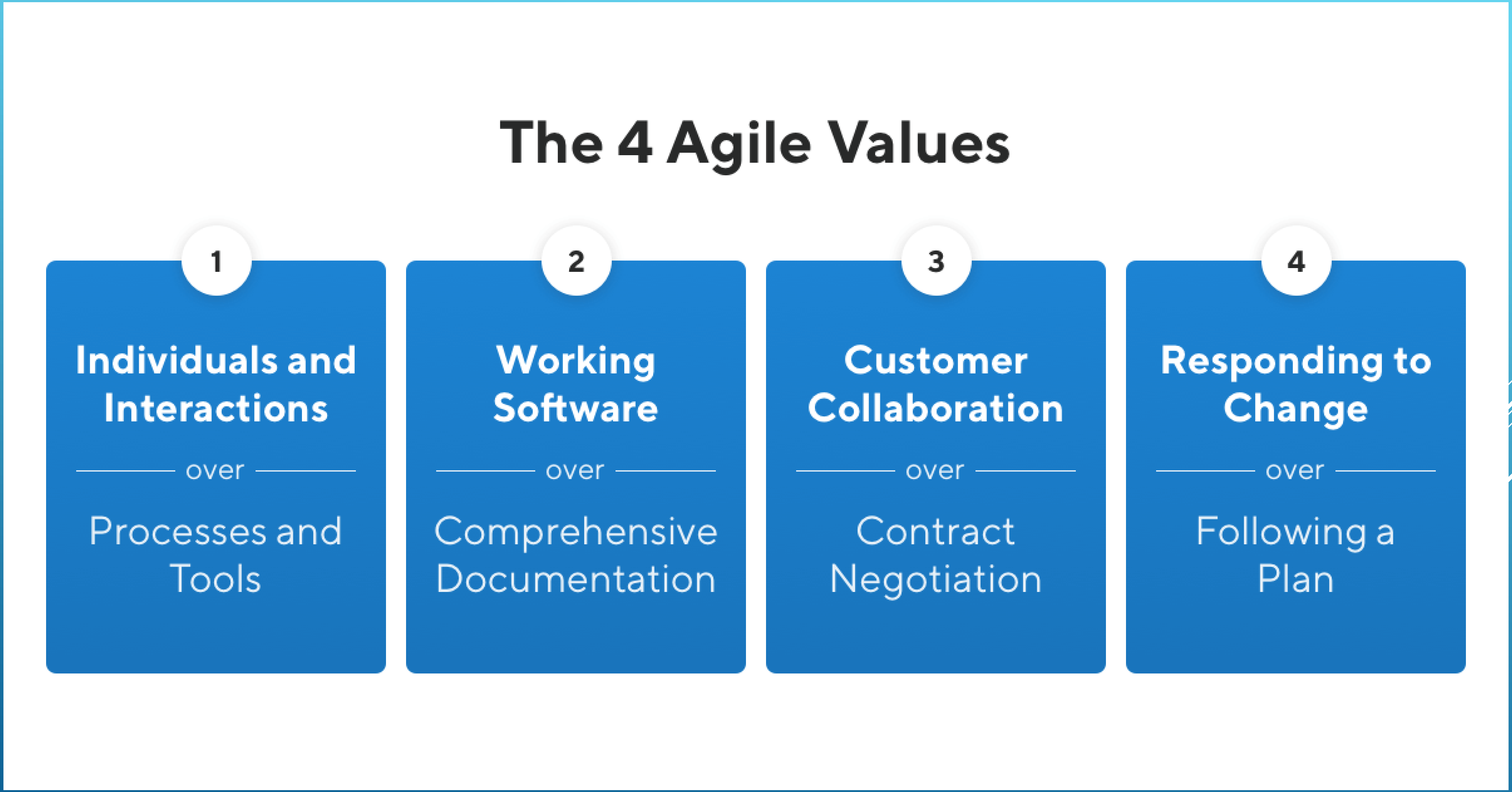
- Both development and testing activities are simultaneously and in one team.

- The tasks are divided into time boxes (small time frames) to deliver specific features for a release. This is called Sprint/Iteration.

**Scrum:**

- Framework through which we will develop and test the software in Agile process.





# The 4 Agile Values

1

**Individuals and Interactions**

over

Processes and Tools

2

**Working Software**

over

Comprehensive Documentation

3

**Customer Collaboration**

over

Contract Negotiation

4

**Responding to Change**

over

Following a Plan

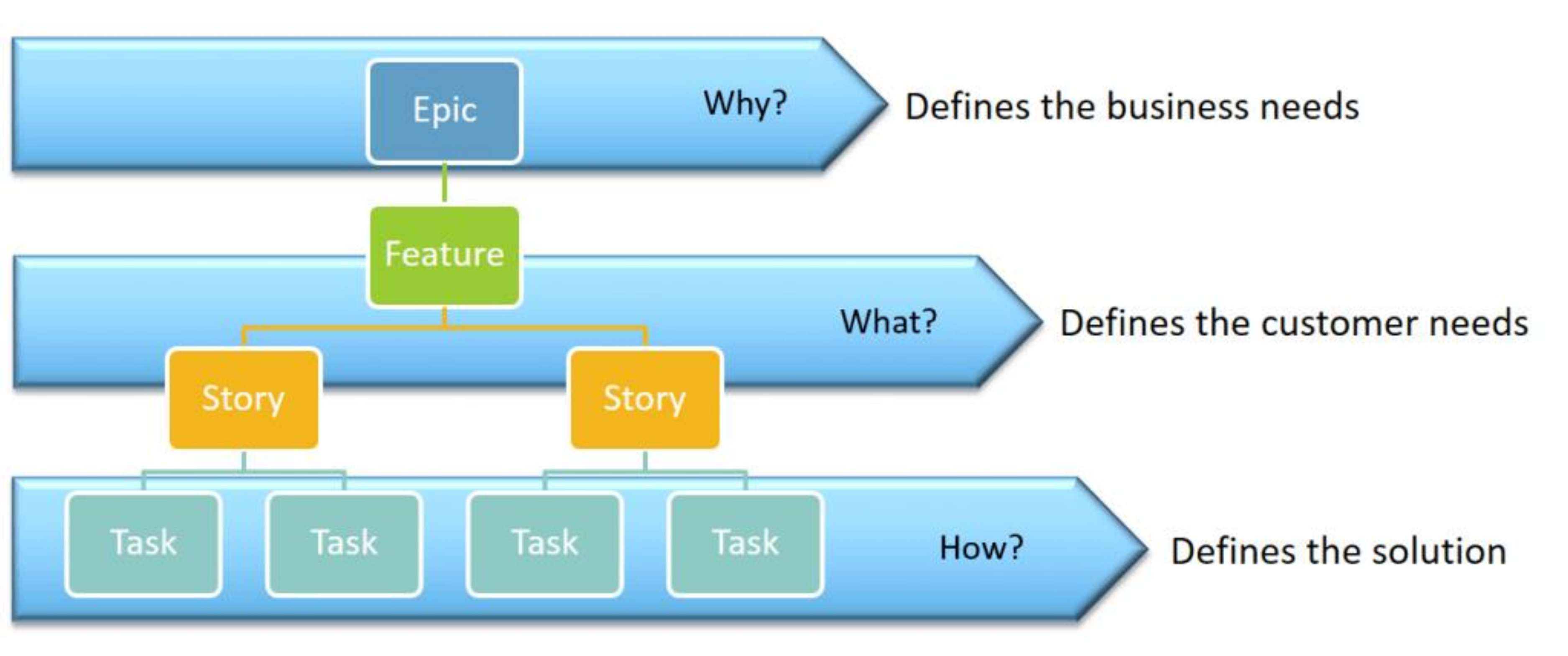
# Agile Model

**Epic:**

- Collection of user stories

**User Story:**

- A feature/module/functionality in a software.



# Scrum Terminology

**Product Backlog:**

- ➢ Defined by PO which contains the requirements from the customers in list of User Stories in order of importance.

**Sprint/Iteration:**

- ➢ Period of time to complete user stories, decided by PO and team, usually between 2-4 weeks.

**Sprint Backlog:**

- ➢ Contains list of committed stories by Dev/Testers for specific sprint.

**Burndown Chart:**

- ➢ Shows how much work is remaining in the sprint, which is maintained by the scrum master daily.

# Agile/Scrum Team

**Scrum Team:**

- Product Owner
- Scrum Master
- Developers
- Testers

"Working together with one goal – to deliver quality product/software to the customer on time."

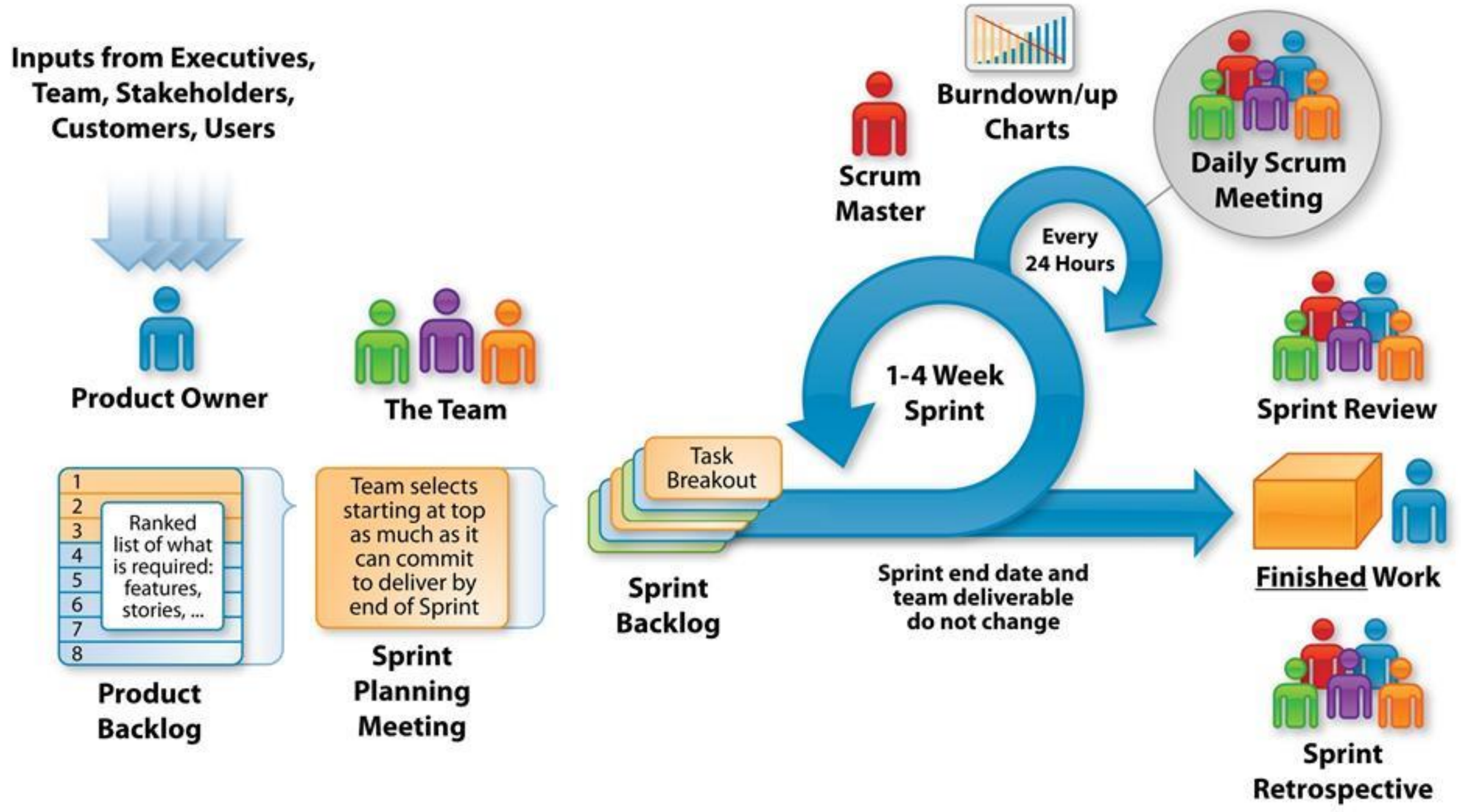
**Product Owner:**

- Define the features/functionality of the product.
- Prioritise features according to market value.
- Adjust features and priority every iteration/sprint.
- Accept or reject work.

**Scrum Master:**

- A specific role – Facilitating and driving the agile process.
- Responsible for conducting all the meeting, checking status of work, if process is going well or not, or any challenges being faced.

# The Agile - Scrum Framework



Inputs from Executives, Team, Stakeholders, Customers, Users

Product Owner

The Team

Ranked list of what is required: features, stories, ...

Product Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

Sprint Planning Meeting

Task Breakout

Sprint Backlog

Scrum Master

Burndown/up Charts

Daily Scrum Meeting

Every 24 Hours

1-4 Week Sprint

Sprint end date and team deliverable do not change

Sprint Review

Finished Work

Sprint Retrospective

# Agile Meetings

**Sprint Planning**

- A team planning meeting determines what user stories can be delivered in the coming sprint.
- Team review items on the backlog to ensure the backlog contains the appropriate items, that they are prioritised, and that the items at the top of the backlog are ready for delivery.
- Story point Estimation - Fibonacci sequence.

**Daily Stand up/Daily Scrum:**

- What we completed/achieved yesterday?
- Are there any impediments?
- What is the plan for today?

# Agile Meetings

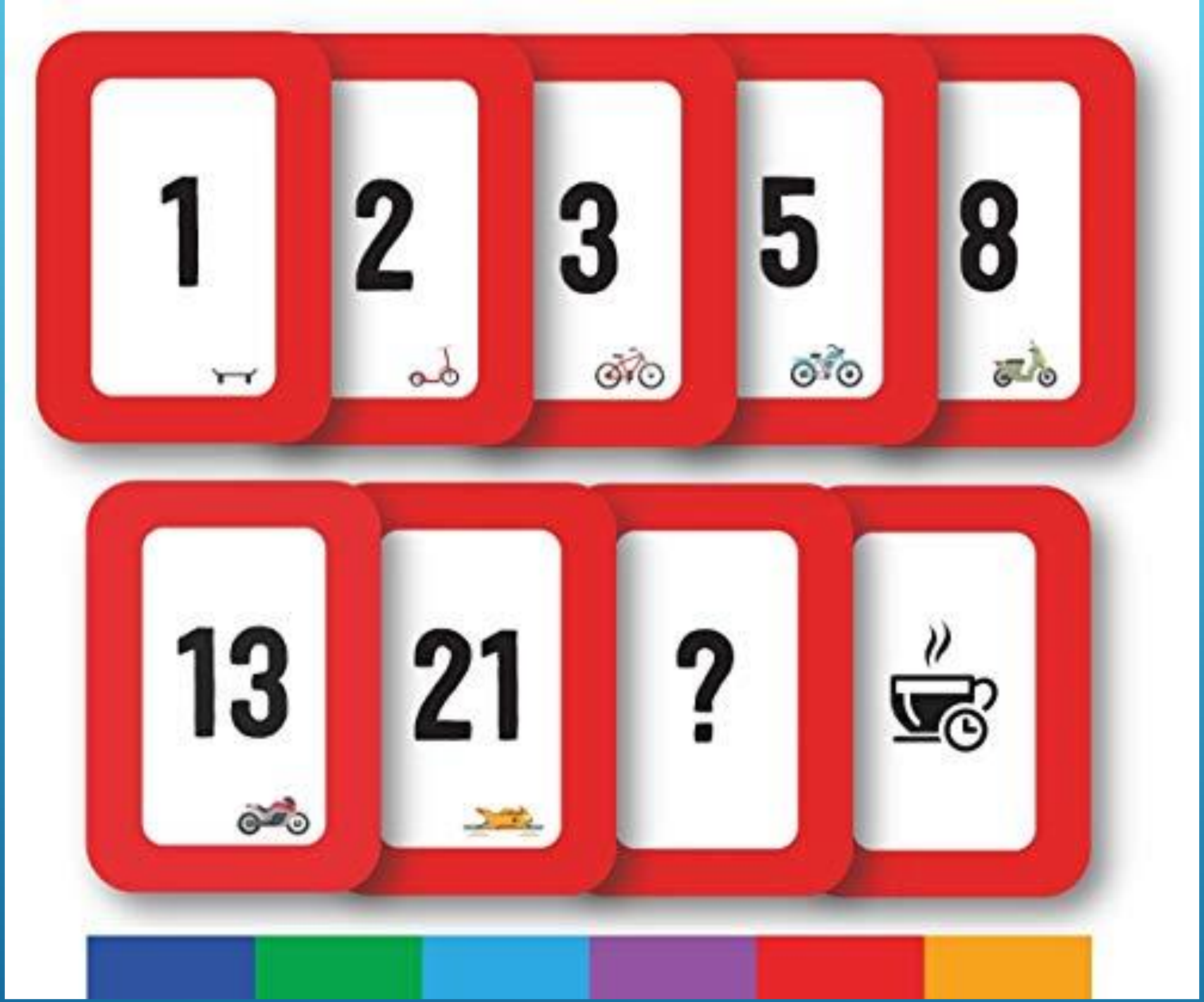
**Sprint Demo/Showcase**

- Demonstration to the client/stakeholders on the piece of work they have done.

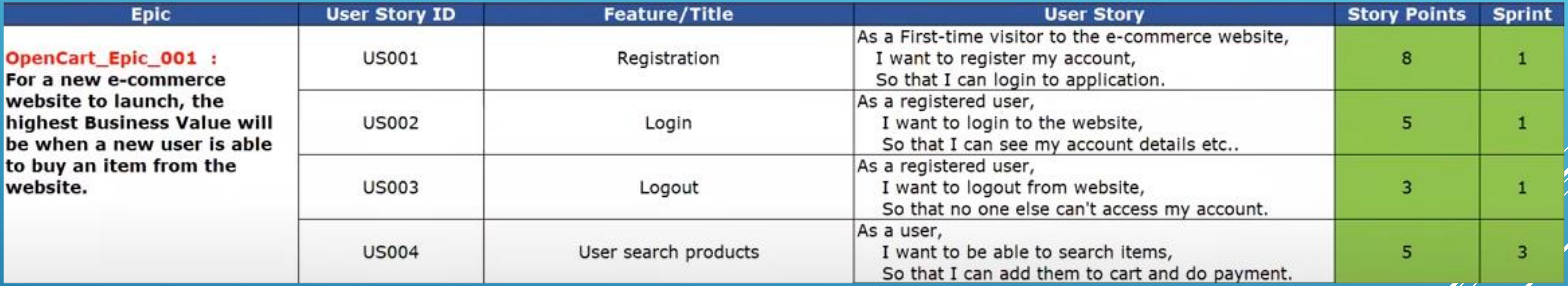
**Sprint Retrospective**

- What went well?
- What didn't go well?
- Any challenges faced?
- How to overcome challenges in the next sprint?

# Fibonacci Series

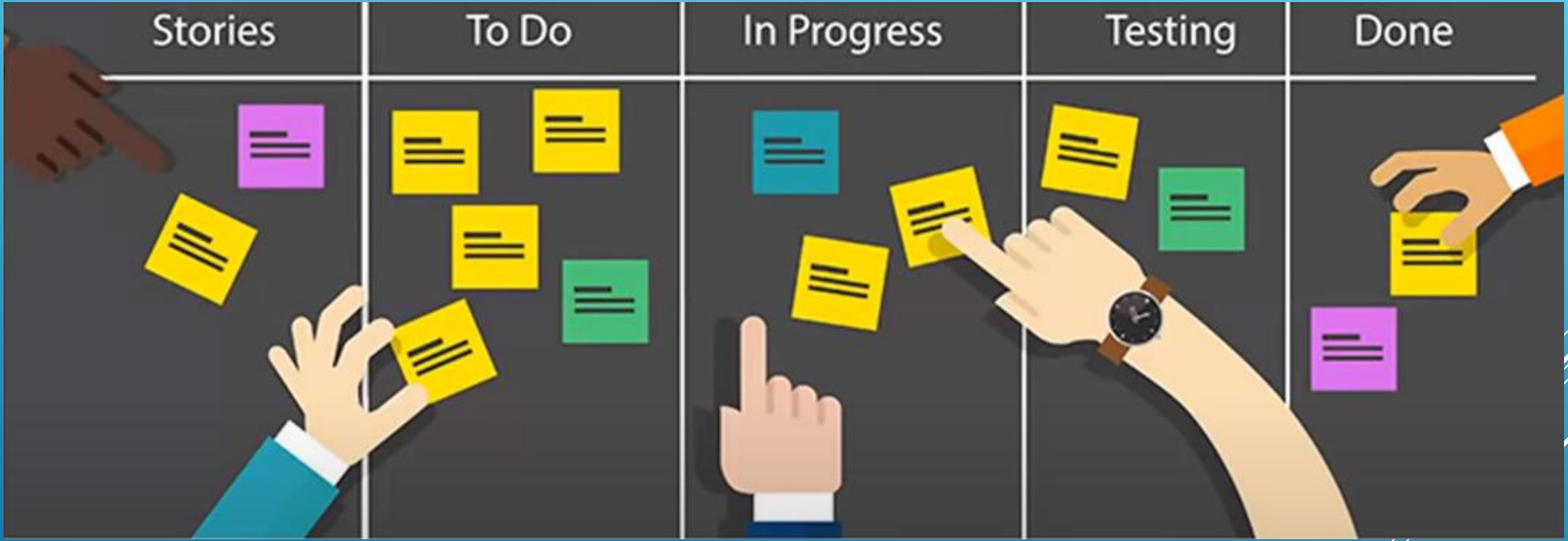


# Example



| Epic | User Story ID | Feature/Title | User Story | Story Points | Sprint |
|---|---|---|---|---|---|
| **OpenCart_Epic_001 :** **For a new e-commerce website to launch, the highest Business Value will be when a new user is able to buy an item from the website.** | US001 | Registration | As a First-time visitor to the e-commerce website,<br>I want to register my account,<br>So that I can login to application. | 8 | 1 |
| | US002 | Login | As a registered user,<br>I want to login to the website,<br>So that I can see my account details etc.. | 5 | 1 |
| | US003 | Logout | As a registered user,<br>I want to logout from website,<br>So that no one else can't access my account. | 3 | 1 |
| | US004 | User search products | As a user,<br>I want to be able to search items,<br>So that I can add them to cart and do payment. | 5 | 3 |

# Scrum Board



# Advantages of Agile Scrum

- Customer will get releases very frequently (continuous delivery).
- Save time and cost.
- Change requests/Requirement changes can be accepted anytime.
- Quality is ensured and customer satisfaction is maintained.
- All team is participating with each other so communication and interaction is transparent.

# Disadvantages of Agile Scrum

- Planning can be less concrete: It can sometimes be hard to pin down a solid delivery date. reprioritising tasks and additional sprints may be added at any time in the project, adding to the overall timeline.
- Team must be knowledgeable: Agile teams are usually small, so team members must be highly skilled in a variety of areas. They also must understand and feel comfortable with the chosen Agile methodology.
- Less focus on design and documentation: The Agile principle prefers working software over comprehensive documentation in order to provide faster delivery
- Final product can be very different: The initial Agile project might not have a definitive plan, so the final product can look much different than what was initially intended. Because Agile is so flexible, new iterations may be added based on evolving customer feedback, which can lead to a very different final deliverable.

# Agile Overview

- A product owner creates a prioritised wish list called a *product backlog*.
- During *sprint planning*, the team pulls a small chunk from the top of that wish list, a *sprint backlog*, and decides how to implement those pieces.
- The team has a certain amount of time — a *sprint* (usually two to four weeks) — to complete its work, but it meets each day to assess its progress (daily stand up).
- Along the way, the ScrumMaster keeps the team focused on its goal.
- At the end of the sprint, the work should be potentially shippable: ready to hand to a customer, put on a store shelf, or show to a stakeholder (Showcase).
- The sprint ends with a *sprint review and retrospective*.
- As the next sprint begins, the team chooses another chunk of the product backlog and begins working again.

# WHAT IS DEFINITION OF DONE?

- Definition of Done. The definition of done (DoD) is when all conditions, or acceptance criteria, that a software product must satisfy are met and ready to be accepted by a user, customer, team, or consuming system. ... It will prevent features that don't meet the definition from being delivered to the customer or user.
- Who creates definition of done? - Yes, The Definition of Done is created by the Scrum team. The Acceptance Criteria is created by the Product Owner. They are orthogonal concepts, but both need to be satisfied to finish a story.
- Why we need definition of done? - Definition of Done. ... We must meet the definition of done to ensure quality. It lowers rework, by preventing user stories that don't meet the definition from being promoted to higher level environments. It will prevent features that don't meet the definition from being delivered to the customer or user.

# DoR vs DoD

**Definition of Ready**

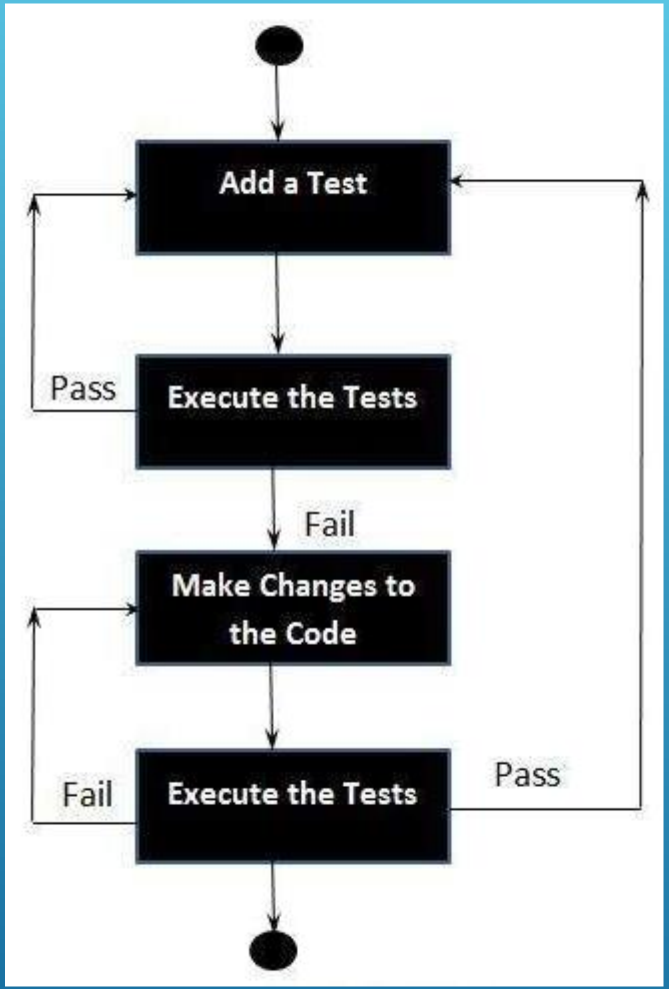
- User Story is clear
- User Story is testable
- User Story is feasible
- User Story defined
- User Story Acceptance Criteria defined
- User Story dependencies identified
- User Story sized by Development Team
- Scrum Team accepts User Experience artefacts
- Performance criteria identified, where appropriate.
- Scalability criteria identified, where appropriate
- Security criteria identified, where appropriate
- Person who will accept the User Story is identified
- Team has a good idea what it will mean to Demo the User Story.

**Definition of Done**

- Code produced (all 'to do' items in code completed).
- Code commented, checked in and run against current version in source control
- Peer reviewed (or produced with pair programming) and meeting development standards.
- Builds without errors.
- Unit tests written and passing.
- Deployed to system test environment and passed system tests.
- Passed UAT (User Acceptance Testing) and signed off as meeting requirements.
- Any build / deployment / configuration changes are implemented / documented / communicated.
- Relevant documentation / diagrams produced and / or updated.
- Remaining hours for task set to zero and task closed.

# TEST DRIVEN DEVELOPMENT (TDD)

- Test-driven development starts with developing test for each one of the features. The test might fail as the tests are developed even before the development. Development team then develops and refactors the code to pass the test.
- Test-driven development is related to the test-first programming evolved as part of extreme programming concepts.



**Benefits of TDD:**

- Much less debug time
- Code proven to meet requirements
- Tests become Safety Net
- Near zero defects
- Shorter development cycles

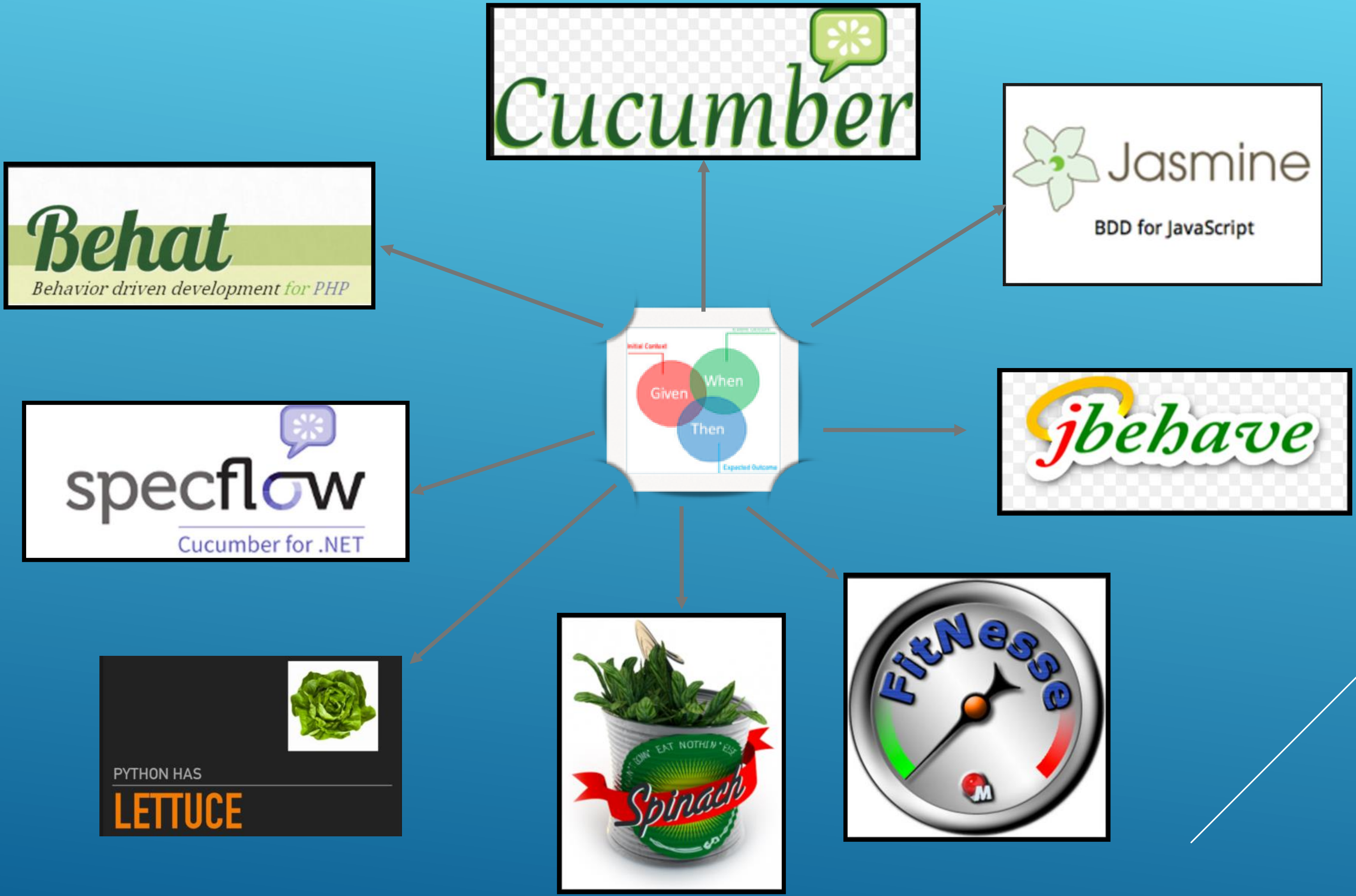
# BDD (BEHAVIOUR DRIVEN DEVELOPMENT)

- BDD is a subset of TDD.
- Also Known as Specification by Example.
- Dan North, who first formulated the BDD approach
- Behaviour-driven development (BDD) is a software development methodology in which an application is specified and designed by describing how its behaviour should appear to an outside observer.

# BDD BENEFITS

- BDD offers more precise guidance on organizing the conversation between developers, testers
- Given-When-Then format, are closer to everyday language
- Requirement becomes easy to understand and communicate
- Provides detailed (executable) specifications.
- Provides concrete evidence that your code works.

# BDD TOOLS


Cucumber
Jasmine
BDD for JavaScript
jbehave
FitNesse
Spinach
PYTHON HAS
LETTUCE
specflow
Cucumber for .NET
Behat
Behavior driven development for PHP
Initial Context
Given
When
Then
Expected Outcome

# WHAT IS GHERKIN?

- Gherkin is the **plain-text English** language that Cucumber understands.
- It is a **Business Readable**, Domain Specific Language that lets you **describe software's behaviour** without detailing how that behaviour is implemented.
- Gherkin serves two purposes — documentation and automated tests.
- Gherkin is designed to be **easy to learn by non-programmers**, yet structured enough to allow concise description of examples to illustrate business rules in most real-world domains.
- In Gherkin, each line has to start with a Gherkin keyword, followed by any text you like. The main keywords are:
  - Feature
  - Scenario
  - Given, When, Then, And, But (Steps)
  - Background
  - Scenario Outline
  - Examples

# BDD KEYWORDS

**Feature:** Keyword

- Each *Gherkin* file begins with a ***Feature*** keyword. *Feature* defines the logical test functionality you will test in this feature file. For e.g. if you are testing a payment gateway your *Feature* will become *Payment Gateway*

**Background:** Keyword

- ***Background*** keyword is used to define steps which are common to all the tests in the feature file.

**Scenario:** Keyword

- Each Feature will contain some number of tests to test the feature. Each test is called a ***Scenario*** and is described using the *Scenario:* keyword.

**Given** Keyword

- ***Given*** defines a precondition to the test. For e.g. In shopping website, assume that the *Login page link* is only present on the Home Page, so the precondition for clicking the *Login link* is that the user is at the Home Page.

**When** Keyword

- ***When*** keyword defines the test action that will be executed. By test action we mean the user input action.

**Then** Keyword

- ***Then*** keyword defines the Outcome of previous steps.

**And** Keyword

- ***And*** keyword is used to add conditions to your steps.

**But** Keyword

- ***But*** keyword is used to add negative type comments. It is not a hard & fast rule to use but only for negative conditions.

**Scenario Outline:** Keyword

- ***Scenario Outline*** keyword is used to remove repeated given steps from different scenario(s) in same feature.

# BDD EXAMPLE

**Feature: Refund item**

**Scenario**: Jeff returns a faulty microwave

- **Given** Jeff has bought a microwave for $100
- **And** he has a receipt
- **When** he returns the microwave
- **Then** Jeff should be refunded $100

**Feature:**

- Feedback when entering invalid credit card details (**Feature Definition)**

**Background:**

- **Given** I have chosen an item to buy,
- **And** I am about to enter my credit card number

**Scenario:** Credit card number too short (**Scenario Definition)**

- **When** I enter a card number that is less than 16 digits long
- **And** all the other details are correct
- **And** I submit the form (**Steps)**
- **Then** the form should be redisplayed
- **And** I should see a message advising me of the correct number of digits

# What is Exploratory Testing?

- Exploratory testing is a type of software testing where Test cases are not created in advance but testers check system on the fly. They may note down ideas about what to test before test execution. The focus of exploratory testing is more on testing as a "thinking" activity.
- Exploratory testing is widely used in Agile models and is all about discovery, investigation, and learning. It emphasizes personal freedom and responsibility of the individual tester.

# When to use exploratory testing?

Exploratory testing can be used extensively when

- The testing team has experienced testers
- Early iteration is required
- There is a critical application
- New testers entered into the team

# What is Adhoc Testing?

- Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage.
- Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.
- Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the Software testing technique called Error Guessing.

# When to execute Adhoc Testing?

- Ad hoc testing can be performed when there is limited time to do elaborative testing.
- Usually adhoc testing is performed after the formal test execution.
- And if time permits, ad hoc testing can be done on the system.
- Ad hoc testing will be effective only if the tester is knowledgeable of the System Under Test.

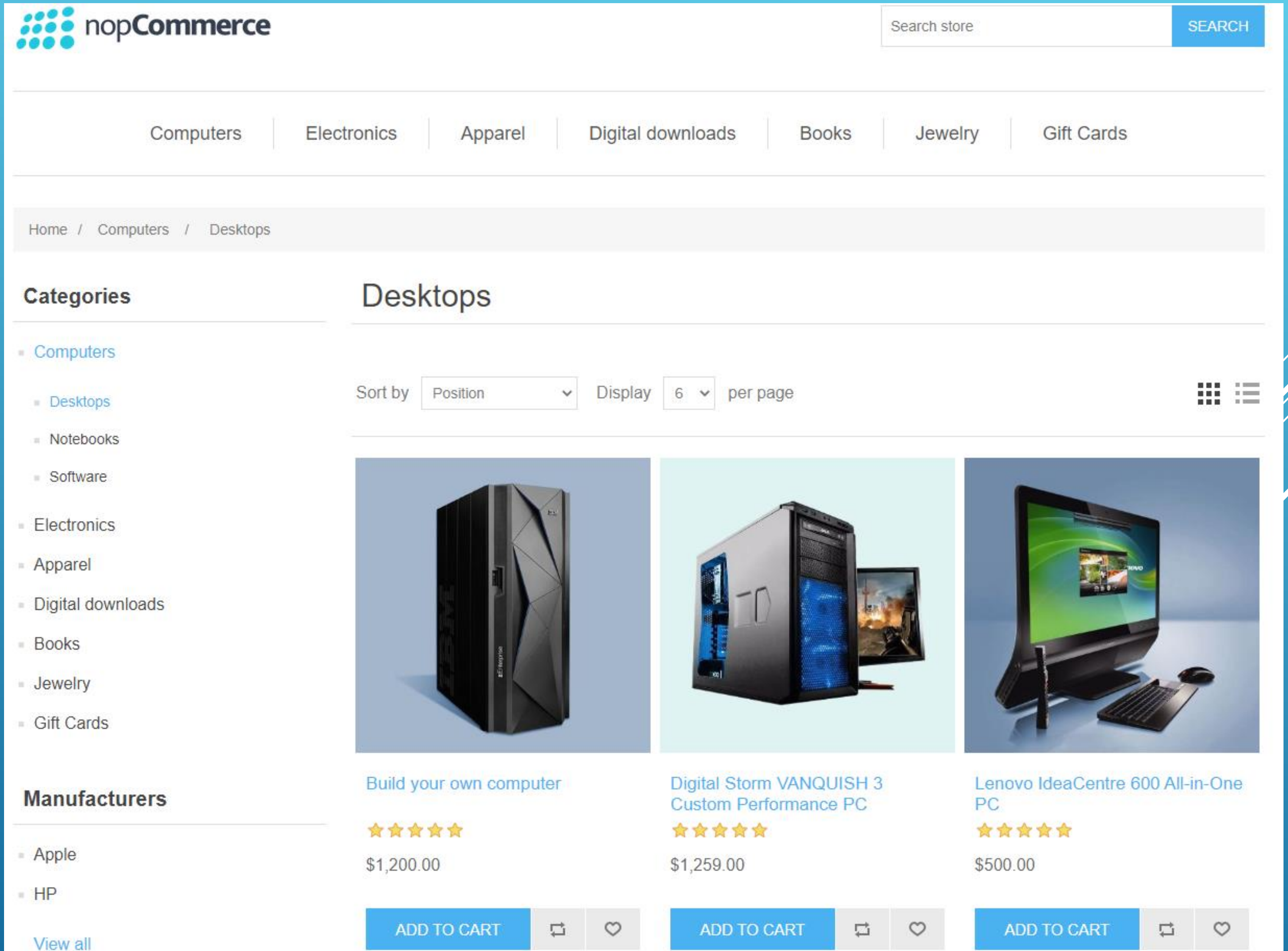
## Adhoc Testing

- No Documentation
- No Test Design
- No Test Case

# HOMEWORK

- Test cases in BDD format for shown wireframe.
- Interview preparation including today topics.

Deadline:
Wednesday Midnight Latest



Desktops

Sort by Position Display 6 per page

Build your own computer
★★★★★
$1,200.00

Digital Storm VANQUISH 3 Custom Performance PC
★★★★★
$1,259.00

Lenovo IdeaCentre 600 All-in-One PC
★★★★★
$500.00