**1. Behavior Driven Development (BDD) :**  
Behavior Driven Development (BDD) is a development technique which focuses more on a software application’s behavior. Mainly it creates an executable specification that fails because the respective feature doesn’t exist, then writing the simplest code that can make the specification pass and as a result we get the required behavior implemented in the system.  Actually it is a team methodology where Developers, Customer, QAs are involved in it.

**Process of BDD :**

1. Write the behavior of the application
2. Write the automated scripts
3. Then Implement the functional code
4. Check if the behavior is successful and if not success then fix it
5. Organize the code (Optional)
6. Repeat the steps for another behavior

Diagram

Description automatically generated

**2.Test Driven Development (TDD) :**  
[Test Driven Development (TDD)](https://www.geeksforgeeks.org/test-driven-development-tdd/) is a development technique which focuses more on the implementation of a feature of a software application/product.  Mainly it refers to write a test case that fails because the specified functionality doesn’t exist and after that update the code that can make the test case pass and as a result we get the feature implemented in the system. Actually it is a development practice where the developers are involved in it.

**Process of TDD :**

1. Add test case
2. Run the test cases and watch test fails
3. Update the code
4. Run the test cases again
5. Refactor the code (Optional)
6. Repeat the steps for another test case

Diagram

Description automatically generated

**Difference between BDD vs TDD :**

|  |  |  |
| --- | --- | --- |
| **S.NO.** | **Behavior Driven Development** | **Test Driven Development** |
| 01. | Behavior Driven Development is a development technique which focuses more on a software application’s behavior. | Test Driven Development is a development technique which focuses more on the implementation of a feature of a software application/product. |
| 02. | In BDD the participants are Developers, Customer, QAs. | In TDD the participants are developers. |
| 03. | Mainly it creates an executable specification that fails because the respective feature doesn’t exist, then writing the simplest code that can make the specification pass and as a result we get the required behavior implemented in the system. | Mainly it refers to write a test case that fails because the specified functionality doesn’t exist and after that update the code that can make the test case pass and as a result we get the feature implemented in the system. |
| 04. | Its main focus is on system requirements. | Its main focus is on unit test. |
| 05. | In BDD the starting point is a scenario. | In TDD the starting point is a test case. |
| 06. | It is a team methodology. | It is a development practice. |
| 07. | Here language used to write behavior/scenarios is simple English language. | Here language is used is similar to the one used for feature development like programming language. |
| 08. | In BDD collaboration is required between all the stakeholders. | In TDD collaboration is required only between the developers. |
| 09. | It is a good approach for project development which are driven by user actions. | It is a good approach for projects which involve API and third-party tools. |
| 10. | Some of the tools used are  Cucumber, Dave, JBehave, Spec Flow,  Concordian, BeanSpec etc. | Some of the tools used are  JBehave, JDave, Cucumber, Spec Flow, BeanSpec, FitNesse etc. |

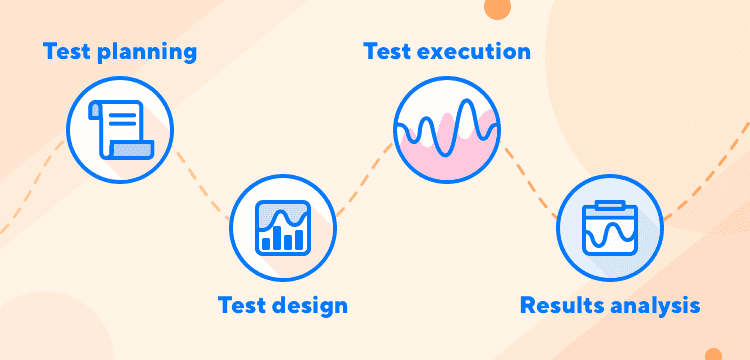
## 3.End-to-End Testing (E2E) :

End-to-end testing is a technique that tests the entire software product from beginning to end to ensure the application flow behaves as expected. It defines the product’s system dependencies and ensures all integrated pieces work together as expected.

The main purpose of End-to-end (E2E) testing is to test from the end user’s experience by simulating the real user scenario and validating the system under test and its components for integration and data integrity.

Software systems nowadays are complex and interconnected with numerous subsystems. If any of the subsystems fails, the whole software system could crash. This is a major risk and can be avoided by end-to-end testing.

## ****End-to-End Testing Lifecycle****



An end-to-end testing lifecycle consists of four components: test planning, test design, test execution, and results analysis.

* **Test planning**: Specifies key tasks, associated schedule, and resources
* **Test design**: Test specifications, test case generation, risk analysis, usage analysis, and scheduling tests
* **Test execution**: Executes test cases and documents testing results
* **Results analysis**: Analyzes test results, evaluate testing, and perform additional testing if necessary

## ****Benefits of End-to-End Testing****

End-to-end testing has been more reliable and widely adopted because of the following benefits:

* Expand test coverage
* Ensure the correctness of the application
* Reduce time to market
* Reduce cost
* Detect bugs

Modern software systems allow subsystem interactions through advancements in technology. Whether the subsystem is the same or different from the main system, within or outside the organization, subsystem failures can cause adverse effects throughout the entire system.

System risks can be avoided by performing the following:

* Verifying the system flow
* Increasing test coverage areas
* Detecting issues associated with the subsystem

E2E testing broadly appeals to many groups:

* **Developers** benefit from it because most of the testing is offloaded to others.
* **Testers** find it useful because writing tests that verify real-world behavior help them avoid problems and achieve a greater sense of accomplishment.
* **Managers** like it because user simulation tests let them know how a failing test would impact the user.