Writing BDD Test Scenarios

Before we start to understand about the BDD test scenarios. Let’s basic idea about the BDD and TDD and different between them.

The difference between BDD and TDD:

Both BDD and TDD refer to the methods of software development employed by your engineering team. There are broadly 2 mainstream approaches to development: test driven development is one and behaviour driven development is the other.

BDD stands for behaviour driven development	TDD stands for test driven development.
The primary goal of behaviour driven development is to solve the problem of communication between the business (including the product manager), the engineering team and the machines.	Test driven development is primarily concerned with the principle of unit testing
The aim of behaviour driven development is to reduce the cost of translation.	TDD perform testing specific, individual units of code
Broadly speaking, BDD is meant to eliminate many of the issues that TDD introduces. BDD tests use a more verbose language so that they can be read almost like you would read a sentence.	The primary focus in test driven development is to ensure that the unit tests pass and that the ‘build’ is green.
BDD principles can help you and your team shift your mindset towards the behaviour of your product so that testing isn’t from a purely technical perspective.	TDD is focusing exclusively on the code itself i.e. the technical aspects of a system, disregards the human, *or* behavioural aspect of your application.

What are the principles of BDD?

1. BDD encourages simple languages to be used across teams, known as ubiquitous languages.
2. The simple and easy to use language should be used in the way the tests themselves are written, so that in theory, a business person can read a test and understand what it is testing.
3. Tests are often written from the customer’s point of view; the focus is on the customers and the users who are interacting with the product

What are benefits of BDD?

1. **Simple language**: - the straightforward language is usable/understandable, not only by domain experts, but also by every member of the team.
2. **Focus:** – BDD helps teams focus on a product's behavioural elements rather than focusing on testing the technical implementation in isolation through individual units. This subtle, but important shift, means that everyone is focused on what the behaviour of the product should be.
3. **Using Scenarios**: – BDD is designed to speed up the development process. Everyone involved in development relies upon the same scenarios. Scenarios are requirements, acceptance criteria, test cases, and test scripts all in one; there is no need to write any other artifact.
4. **Efficiency:** – BDD frameworks make it easy to turn scenarios into automated tests. The steps are already given by the scenarios – the automation engineer simply needs to write a method/function to perform each step's operations.

How to write BDD scenarios

Your first question is probably 'what are BDD scenarios?. That's a fair question!

A BDD scenario is a written description of your product's behaviour from one or more users' perspectives.

Scenarios are designed to reduce the cost of translation and make it easier for your engineers to understand the requirements and for your QA.

Using BDD scenarios means requirements and tests can be combined into 1 specification. In some cases, the scenarios that are written can then be easily converted into automated tests.

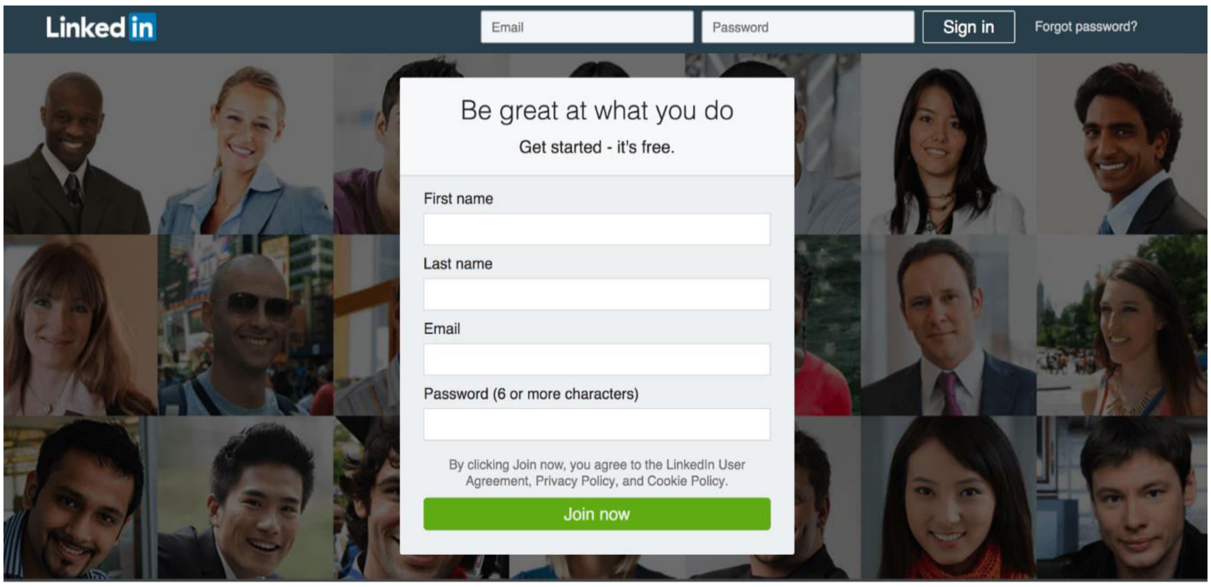
BDD scenarios tend to follow a specific format. The format is fairly straight forward, and with a little practice you'll be able to write your own.

In its simplest format, there are 3 key elements in any BDD scenario:

GIVEN (describing the context)
WHEN (describing the action)
THEN (describing the outcome)

In order to explain how BDD and scenarios work in practice, let's take a look at the example of a user signing up to LinkedIn.

Example – signing up for a LinkedIn account


Linked in
Email
Password
Sign in
Forgot password?
Be great at what you do
Get started - it's free.
First name
Last name
Email
Password (6 or more characters)
By clicking Join now, you agree to the LinkedIn User Agreement, Privacy Policy, and Cookie Policy.
Join now

Here's a basic BDD scenario which describes the LinkedIn signup process:

Accepted Criteria	GIVEN	WHEN	THEN	QA STATUS
ABC successfully creates a LinkedIn Account	ABC is on the LinkedIn Registration page	he enters all required registration fields	LinkedIn account is created successfully	PASS/FAIL
User gets the proper validation messages when submit the registration form with blank fields.	The user has not entered any data on the form	They click the 'join now' button.	User should show the proper validation messages.	

If you have more than one or you require more information than this, you can add them with AND.

Using ANDs

When you require more information in a scenario, an 'AND' can be used after any of the descriptors:

GIVEN (context),
AND (further context),
WHEN (action/event),
AND (further action/event),
THEN (outcome)
AND (further outcome)

Let’s revisit our LinkedIn registration scenario and flesh it out in more detail with a few ANDs cases.

Accepted Criteria	GIVEN	WHEN	THEN	QA STATUS
ABC successfully creates a LinkedIn Account	ABC is on the LinkedIn Registration page.	He enters all required registration fields. **AND** He click on ‘join now’ button.	His LinkedIn account is created successfully. **AND** He is directed to the profile creation page. **AND** His confirmation email is sent.	PASS/FAIL

How do BDD scenarios work with user stories?

Your BDD scenarios should form part of 1 specification. This includes both the user story and the scenarios. So, for example, if you’re writing a user story in **JIRA** or some other **beloved backlog management tool** you could structure your specification in JIRA in the following order:

1. **User story** – start with the user story describing the requirements from a user’s perspective
2. **BDD scenarios** – then include your scenarios describing the behaviour of the system to assist with testing

A user story has a slightly different structure to your BDD. Your user story should comprise the following:

As an X
I can / want Y
So that Z

Using our LinkedIn example:

As a new user (ABC),
I can register a new account on the homepage
So that I can access LinkedIn.

When should BDD scenarios be used?

BDD scenarios are not necessarily mandatory across all of your product specifications. If they were you'd spend most of your life writing BDD scenarios which would clearly be very unpleasant.

BDD scenarios are best suited to specifications where you think there is a likelihood that the requirements may be misunderstood without them or that a more thorough testing approach needs to be adopted. If you're working on a small color change, text change or a technical chore / bug, there will clearly be no case for using BDD scenarios as this would be a waste of everyone's time.

It may be that you as a team decide to write them for all major new feature stories or that you only focus on a specific type of specification. BDD scenarios can assist you in your development process but as with all things product, you and your team should decide on what works best for you.

BDD Tools

Cucumber:
Cucumber is a test framework that supports BDD. In Cucumber, the BDD specifications are written in plain, simple English which is defined by the Gherkin language. In other words, Gherkin is a language that Cucumber understands. Gherkin presents the behaviour of the application used, from which Cucumber can generate the acceptance test cases. Cucumber is a framework developed by Ruby that can work across different technologies.

Spec flow:
Spec flow evolved from the Cucumber framework using Ruby on Rails, and is used mainly for .Net projects. Spec Flow also uses the Gherkin language.

Some benefits to using BDD

If you plan to implement BDD, here are a few points that will benefit the software team.

1. You are no longer defining 'test', but are defining 'behaviour'.
2. Better communication between developers, testers and product owners.
3. Because BDD is explained using simple language, the learning curve will be much shorter.
4. Being non-technical in nature, it can reach a wider audience.
5. The behavioural approach defines acceptance criteria prior to development.

Disadvantages of BDD

Even the best development approaches can have pitfalls and BDD is no exception. Some of them are:

1. To work in BDD, prior experience of **TDD** is required.
2. BDD is incompatible with the waterfall approach.
3. If the requirements are not properly specified, BDD may not be effective.
4. Testers using BDD need to have sufficient technical skills.