COURSEWARE

Acceptance Testing

Contents

Overview

Below we will be covering the basics of Acceptance testing, exploring how and why we use this testing approach.

What is Acceptance Testing

Our testing here focuses on the original requirements of the project and whether we've achieved our goal.

As described in System testing, there are two approaches to testing; `White/Clear box` and `Black box` testing. Acceptance testing is primarily performed from the highest level available on our project, typically targeting the front-end of the service. As such, Acceptance testing is primarily a form of Black box testing because it doesn't expose or require knowledge about the code that forms the back-end service; we are only testing the front-end to see if we have achieved our client's request.

Acceptance testing allows us to replicate/emulate the behaviour of a potential customer/client and their interactions with our product, interacting with it and determining that the functionality that is provided and implied.
Checking our product this way allows us to check whether the functionality matches the original requirements of the product.

Why do we perform Acceptance Testing

As part of testing and following the `v-model` software development lifecycle, acceptance testing helps us target the original requirements of the project specification. Acceptance testing in conjunction with `Unit`, `Integration`, and `System` testing helps provide a proof model to verify that the software solution developed meets the clients' original request.

Developing any product needs to meet the `minimum viable requirements` expected by the end-user, as such, we use Acceptance testing against user acceptance criteria to prove we have achieved our intended `MVP - minimum viable product`.

Whilst the goal is to produce a project that meets the goals originally set out, the MVP should be prioritised above other features; this is, however, not an end goal as there may be other requirements that are part of the overall project which should not be neglected if possible.

Tutorial

Below we'll investigate the practical steps of performing acceptance testing.

How to perform Acceptance Testing

Acceptance testing is unlike many of the testing approaches prior i.e `Unit`, `Integration`, `System`.

Because the tangible metric is defined by the client's requirements we cannot simply target code and line coverage, we instead focus on targeting user stories to determine if we've achieved the Minimum Viable Project.

As such we follow a premise that builds upon `TDD`, using a process known as BDD or Behaviour Driven Development we can attempt to perceive and fulfill our user requirement of the software we develop.

BDD - Behavioural Driven Development

We often break down user stories into steps that we can perform to check our initial circumstances, the actions we wish to take, and then the consequences of our collective steps.

Using the following `Given`, `When`, and `Then` we can take a scenario and create tests that reflect user interaction:

- GIVEN - A certain circumstance or initial starting position
- WHEN - We perform an action
- THEN - An observable result should emerge

Due to the simple language of this approach, it is arguable that these can be created by anyone with access to the original outline of the project specification, in a typical IT environment we have tools that can be used in conjunction with this approach to aid in the process testing.

Technologies

Cucumber and Gherkin

Test technologies are built around behavioural driven development practices, as mention above we use the given, when, and then approach.

Cucumber provides us with a technology framework that provides a language parser known as Gherkin, using Gherkin to create `.feature` files that define scenario outlines using `given`, `when`, `then` as well as additional keywords to semantically tie together additional steps of the same type i.e `AND`, `BUT` and `*`.

You can read more on Cucumber and Gherkin on the respective dedicated modules.

Exercises

There are no Exercises with this module.