Application Testing Methods

Benjamin Roberts

Level 4 Software Development

# Table of Contents

[Table of Contents 2](#_Toc175317409)

[Table of Figures 3](#_Toc175317410)

[Introduction 4](#_Toc175317411)

[Theory Review – Testing 5](#_Toc175317412)

[Methodologies 7](#_Toc175317413)

[Functional Testing 7](#_Toc175317414)

[Non-Functional Testing 7](#_Toc175317415)

[Integration into Development 7](#_Toc175317416)

[Frameworks 9](#_Toc175317417)

[Software 9](#_Toc175317418)

[Language Libraries 9](#_Toc175317419)

[Workplace Findings 9](#_Toc175317420)

[Theory Reflections 10](#_Toc175317421)

[Testing – Applied 11](#_Toc175317422)

[Scenarios – Test Plan 12](#_Toc175317423)

[Test Code created 13](#_Toc175317424)

[Conclusion 14](#_Toc175317425)

[References 15](#_Toc175317426)

# Table of Figures

[Figure 1 - Internal Email Communication with IT Development & Testing Leads (names redacted for privacy) 5](#_Toc175317427)

[Figure 2- Email Response from IT with starting point suggestions 6](#_Toc175317428)

# Introduction

This report covers my research into Testing Frameworks and Methodologies, undertaken as an upskilling exercise so I can expand my knowledge and toolkit for ensuring the quality of software I work on. This will cover:

* A **theory review** of various Testing Frameworks and Methodologies accepted by the industry (with references)
* Provide details on which **Frameworks and Methodologies** are **used within my team**
* Expand on this via reaching out to our dedicated IT Testing teams, who I know engage in more testing methodologies then my team do, which will allow a view of testing in the wider organisation

Also covered are my efforts to apply these Testing Frameworks and Methodologies. To do this I put together a Test Plan for testing a deployed version of an implementation of our Credit Decisioning system. Within this plan I cover:

* **System Testing** - conducted via the Postman software, which makes use of JavaScript test cases via an implementation of the Chai library for running test cases on the HTTP response received from a web application
* **Integration testing** – for testing that our Credit Decisioning system integrates with 3rd party API's successfully (i.e. no errors returned for multiple real-world scenarios)
* **Performance Testing** - via testing peak volumes the system can handle

Possibly **Security testing** (not something we do in my team today, but I'm hopeful I could apply something with the support of our dedicated IT Testing teams)

(Note: would need to censor alot of sensitive information with this idea, possibly even specific parts of the written test code, but this would be a great idea to apply my learnings I feel.)

# Theory Review – Testing

To get started with this, I consulted our IT Development and Testing team leads for information on what methodologies and frameworks were used within the workplace. I approached these stakeholders as I knew they had a higher level of technical expertise then my team (who are a more hybrid team).

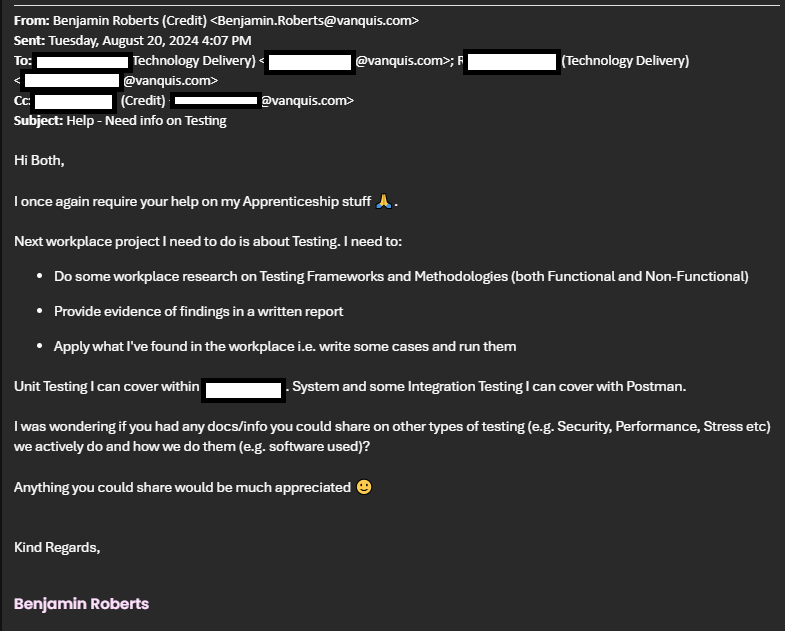


Figure 1 - Internal Email Communication with IT Development & Testing Leads (names redacted for privacy)

Below shows the response I got from the stakeholders. They provided suggestions of where to start with methodologies and frames for my theory review, along with suggestions of tools I could look at for conducting testing. Also was an invitation to discuss further in person.

A screenshot of a computer

Description automatically generated

Figure 2- Email Response from IT with starting point suggestions

With this starting point, I’ve broken down the theory review into multiple sections below.

## Methodologies

Software Testing Methodologies are various strategies/approaches used for testing an application for ensuring it looks and behaves as per requirements (Smartbear, 2024a). These form the strategic approach to how developers/teams will test a piece of software. The approaches and strategies can be split into types:

1. **Functional**
2. **Non**-**Functional**

Testing methodologies have also found their way into the Software Development process directly, leading to development approaches with testing requirements at their core.

### Functional Testing

Functional Testing is about the application against the business requirements set e.g. does a specific feature perform per expectations (Smartbear, 2024a). Different forms of Functional testing are covered below:

#### Unit Testing

Unit Testing involved testing small pieces of code in isolation, allowing the behaviour of the component under test to be proven without any dependency on other parts of the software (AWS, 2024). This is the typical 1st level of testing and is often performed by the development team directly (Smartbear, 2024a). The advantage of this is that it allows a developer to confirm that their written code performs the function it’s expected to without having to design specific test data to handle other components that could block access to the function under test.

My team do this kind of testing within our Credit Decisioning platform as a standard part of our development process. Each piece of functionality has multiple unit tests created for it to demonstrate every possible outcome the component under test could return.

We use a 3rd party provided software to build and manage our decisioning platform, so the exact design how this work is considered trade secrets. As a compromise, the figure below shows a typical unit test we would create within the Java JUnit Framework (an open-source testing framework), which provides the same kind of functionality:

A screenshot of a computer program

Description automatically generated

Figure 3- Java JUnit representation of a Unit Test my team would write within our Credit Decisioning platform

Unit Tests work on the concept of Assertions, which define a TRUE/FALSE condition that must evaluate to TRUE for the Unit Test to pass. Different Frameworks provide various ways to define these Assertions, but a common one is “AssertEquals”, which tests if the output of the component under test equals an expected value. In this example, the 1st parameter passed into the “AssertEquals” method represents the expected value, while the 2nd parameter is the component under test. The method being passed in as the 2nd parameter returns a value back to “AssertEquals”, which can then determine if the test passes or fails by comparing the 2 parameters. Developer Tools the present the result back to the developer e.g. in our Credit Decisioning platform, this gets presented within a specific view available to the platform, seen below:

A screenshot of a computer

Description automatically generated

Figure 4- Unit Test Results as seen in our Credit Decisioning platform

Successful tests appear as regular text within this view, while failed tests will appear in red with a “FAIL” indicator next to them.

Once developed, Unit Tests can be run repeatedly. This makes them useful for confirming a component still works as intended if changes are make to it later.

#### Integration Testing

#### System Testing

#### Acceptance Testing (UAT)

### Non-Functional Testing

Non-Functional Testing focuses on the operational aspects of the software, rather than the features of the software e.g. testing how the software performs with multiple users in parallel (Smartbear, 2024a).

#### Performance Testing

#### Security Testing (Penetration)

#### Usability Testing

#### Compatibility Testing

#### Accessibility Testing

### Integration into Development

#### Test Driven Development

#### Behaviour Driven Development

#### Data Driven Development

## Frameworks

Software Testing Frameworks are the tools a developer/team actually use to conduct their testing.

### Software

* Selenium (Web Apps)
* Postman (System)
* JMeter (performance)

### Language Libraries

* JUnit (Java)
* NUnit (C#)
* Chai (JavaScript – Node)
* Pytest (python)

## Workplace Highlights

## Theory Reflections

# Applied Testing

## Scenarios – Test Plan

* System Testing (Postman)
* Integration Testing (test Delphi and TAC Integrations work)
* Performance (peak volumes and response times)
* Security (support from IT)

## Test Code created

# Conclusion

# References

<https://smartbear.com/learn/automated-testing/test-automation-frameworks/>

https://docs.apimatic.io/testing/testing-frameworks/

AWS. (2024). *What is Unit Testing?*. [Online]. . Available at: https://aws.amazon.com/what-is/unit-testing/#:~:text=Unit%20testing%20is%20the%20process,test%20for%20each%20code%20unit. [Accessed 23 August 2024].

Smartbear. (2024a). *Software Testing Methodologies*. [Online]. . Available at: https://smartbear.com/learn/automated-testing/software-testing-methodologies/ [Accessed 23 August 2024].