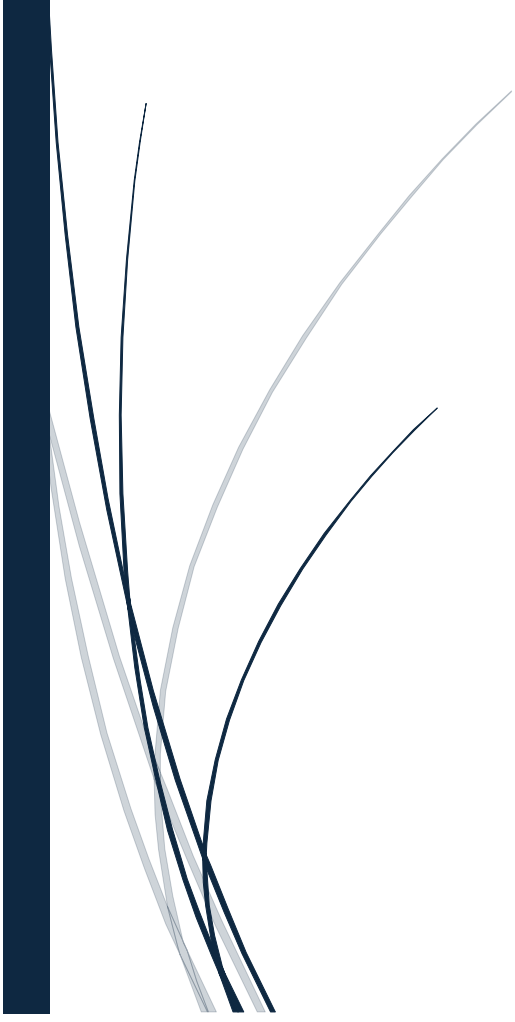
Acceptance Testplan

Individual Project: FestivalConnect



Name: Lucas Jacobs
Class: S-A-RB06
PCN: 490692
Student number: 4607368
Technical teachers: Felipe Ebert, Bartosz Paszkowski
Semester coach: Gerard Elbers

Table of Contents

Introduction

Within this document, the test methods that FestivalConnect is going to use are being covered. This includes why we are using them and what benefits it has. Furthermore, these tests are being automated inside the pipeline, as specified in (Jacobs, 2024, DevOps Report).

User Acceptance Tests

With the use of user acceptance tests (UAT), we will make sure that the end user will be pleased with the final solution. It is critical to confirm that the product that FestivalConnect delivers meets the expected outcomes. Furthermore, it will help identify where the problem could have been when a developer overlooked something. But we can also say with these tests when a product is ready for production, and quickly check if the project needs additional work. (All you need to know about user acceptance testing (UAT), n.d.)

Each UAT will be based on user stories that have been set in (Jacobs, 2024, User Stories). The tests consist of the following:

- Test description: Clear description of what the test is about to verify a certain requirement.
- Test Steps: The steps that the tester needs to walk through to successfully finish the test.
- Expected Results: What the user needs to expect from the test.
- User Stories: The stories that are linked to the test.

Moreover, before each test, there are certain prerequisites that the tester needs to meet to start the test.

Prerequisites

This is for each step mandatory except for the ones with IDs one and two.

Step	Test step	Expected Result
1	The user is logged in as a festival goer or festival organizer and sees the home page	The application is loaded in, the home page has been loaded in with the required information displayed.

Id	Test Description	Step	Test Steps	Expected Results	User Stories
1	Verify that a user can log in	1	The user opens the application as a non-logged-in user.	The user will be automatically redirected to the “Login” page.	U-E001
		2	The user fills in a registered user's credentials and clicks on the ‘Login’ button.	The user will be redirected to the “home page”	
2	Verify that a user can register	1	The user opens the application as a non-registered user.	The user will be redirected to the “Login” page	U-E002
		2	The user clicks on the “Register now” button	The user will be sent to the “Register Page”	
		3	The user fills in the required credentials that are unique and are not used yet and presses the “Register” button.	The user will receive a notification with “Successfully registered”.	

Test Pyramid

The concept originated and was introduced by Mike Cohn in his book “Succeeding with Agile”. It is a term that will serve as a visual metaphor to separate the different kind of layers of testing. Two main principles can be taken out from the pyramid:

1. Different levels of testing: Tests should be written at various levels, from testing a single unit of code to testing the whole flow of the application.
2. Distribution of the tests: This means that the lower level of the pyramid should have more tests than the higher levels.

(Vocke, 2018)

This can be further clarified in the following picture.

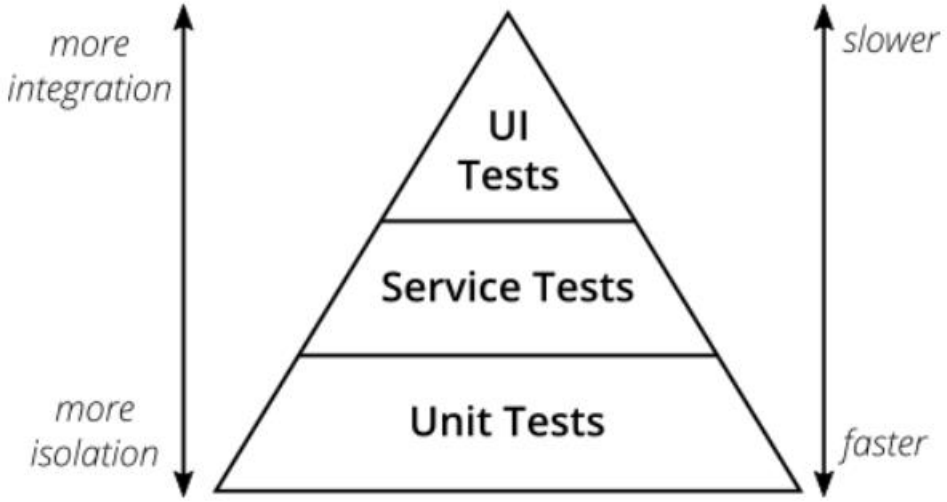


Figure 1: The Test Pyramid (Vocke, 2018)

The Layers of the Test Pyramid

Unit Tests (Base of the Pyramid)

The main goal of this test is to test an individual unit or components of the software with isolation. For the services that are written in C# using the framework .NET, I will test my application in isolation using MSTest and Moq libraries, which easily let me isolate and mock external dependencies to focus on the single unit that needs to be tested. The tests are fast to execute, small, and high in amount, to verify that each small piece of code is working as intended. The unit tests are all written in a AAA pattern, which means Arrange, Act, Assert.

- Arrange: mocking all the external dependencies, the objects that are needed, and the potential expectations when applicable.
- Act: This part contains where the method is being called and tested.
- Assert: To check if the expectations that are set in Arrange are met.

(Vocke, 2018)

Service Tests (Middle Layer)

The main purpose of this is to test the interaction between units and endpoints, such as APIs and services. These are slower than unit tests, have a broader perspective in testing, and are less needed compared to unit tests. The main goal for this is to make sure that different components of the system work together as expected. These are also called integration tests, they test the whole flow and all the parts from outside of the application. So when I have a database connected to my service, this needs to be running to run the test. (Vocke, 2018)

User Interface (UI) Tests (Top Layer)

The focus of these tests is to test the whole application from the user's perspective. These are the slowest to execute and have the least number of tests. This is to make sure that the system will work as a whole and meet the user's requirements. For the frontend tests, FestivalConnect uses Cypress to execute and write tests or E2E tests, which are made to test the whole flow of the application. These tests are executed in different browsers to check the functionalities for each platform and if it is the same. Furthermore, we have user acceptance tests (UAT) which we manually check if real-world scenarios are handled correctly. The main difference between these two tests is that E2E tests are there to simulate user interactions with the UI to validate the end-to-end functionality. Whereas UAT includes the E2E in the scope it uses real users and focuses on business requirements. (Vocke, 2018)

Modern Adjustments and Criticisms

The first thing is to say that the test pyramid is a value guideline, but with the modern usage of software practices, it can lead to limitations. To begin with the service tests termilogy. This is vaguely described. It is better to call it integration tests or API tests. Moreover, With the usage of React in FestivalConnect, UI tests can often be conducted on various levels of the pyramid, exclusively the top. For example, in React there are UI components that can be unit tested to make sure that individual elements are working correctly, so this implies that UI tests do not necessarily mean the "top of the pyramid" (Vocke, 2018)

Conclusion

The testing pyramid and various other testing methods, it is a fundamental concept for automated testing, and guiding an agile project to create a balanced and effective test environment. While the original pyramid needs some adjustments to fit modern software development practices, it is still a core principle that continues to be essential for maintaining FestivalConnect to be a robust and reliable application.

References

All you need to know about user acceptance testing (UAT). (n.d.). Retrieved from userback.io/: https://userback.io/blog/user-acceptance-testing-explained/#:~:text=The%20%E2%80%9CUser%20Acceptance%20Testing%E2%80%9D%20phase,by%20you%20or%20your%20team.

Vocke, H. (2018, 02 26). *The Practical Test Pyramid*. Retrieved from martinfowler: https://martinfowler.com/articles/practical-test-pyramid.html

Jacobs, L. (2024). User Stories (Unpublished manuscript), FontysICT.
Jacobs, L. (2024). DevOps Report (Unpublished manuscript), FontysICT.