Testing Plan for Sprint #2

Table of Content

1. Testing approach

During the development of the system, we will be writing tests at all levels:

i. Unit tests
ii. Integration tests
iii. System tests

i. Unit testing

Unit testing refers to the smallest testable part of software. The objective of unit testing is to verify that each unit of the software performs as designed.

ii. Integration testing

The goal of integration testing is to test multiple components of a system together, focusing on the interactions between them instead of testing the system.

iii. System Testing

System testing is a level of software testing that evaluates the complete and integrated software system. The purpose of system testing is to verify that the entire system functions according to the specified requirements.

To execute these tests, we must first create a form of automation. In our case, a Continuous Integration (CI) pipeline will be used to trigger automatically upon new commits or merges to the codebase. This will help fetch the latest code, build the software, run automated tests at various levels (unit, integration, and system), and generate reports on test results and code coverage. Examples of CI tools are Gitlab CI/CD, Azure DevOps, GitHub Actions, Jenkins, and Travis CI.

We will adopt a comprehensive testing approach that includes both automated unit testing and manual acceptance testing. Unit tests will focus on validating individual components and functionalities of the software, while acceptance tests will ensure that the overall system meets the specified requirements and user expectations. Future research will be conducted to decide which tools we will be using for automated and manual testing.

2. Testing tool

Best C# testing frameworks research in Sprint#1:

Adapted from JUnit, NUnit is an open-source unit testing framework for the .NET framework, and it is widely used in C# development. This framework would allow us to write and execute unit tests in a structured and organized manner. It also has fast testing and execution speeds. While NUnit does have documentation and resources available, it may not be as beginner-friendly as other testing frameworks for team members who haven't used it in past projects.

xUnit.Net is an open-source testing framework that is based on the .NET framework. It is popular for its intuitive terminology and structure for writing tests, making it easy for developers to understand and use. While xUnit.net has documentation and resources available, it may not be as extensive or well-established as NUnit.

MSTest (Microsoft Unit Testing Framework for .NET) is a unit testing framework developed by Microsoft for testing .NET applications. It is tightly integrated with Visual Studio and offers simplified test creation, execution, and debugging. Since MSTest is included with Visual Studio, there's no need to install additional packages or frameworks to start writing and executing tests. However, since it is primarily designed for Windows, it has limited cross-platform support. Furthermore, MSTest may be less extensible and more challenging to integrate with other tools or extend its functionality.

For sprint#2, we will continue using NUnit. NUnit provides seamless integration with the Uno Platform and offers robust features for writing and executing unit tests for our C# codebase.

https://avatars.githubusercontent.com/u/2678858?s=280&v=4

3. Testing Metrics and Coverage

We aim to achieve a minimum test coverage of 80% for our codebase by the end of Sprint #2. Test coverage will be measured using Visual Studio's built-in code coverage analysis tools.

We will track the number of defects identified during testing per thousand lines of code (KLOC) to assess the overall quality of our software.

We will monitor the time taken to execute our automated test suite to identify any performance bottlenecks or inefficiencies in our testing process.

4. Acceptance tests

Acceptance tests will be based on the acceptance criteria outlined in the user stories planned for Sprint #2. These criteria define the specific behaviors and functionalities that must be present in the software to meet user expectations. Once these acceptance tests are passed, we will be able to cross out

This testing plan outlines our approach, tools, metrics, and acceptance testing strategy for Sprint #2. By following this plan, we aim to ensure the quality, reliability, and compliance of our software with the specified requirements and user expectations.