TeamVas Testplan

**Introduction**

I want to ensure software quality through various testing methods and continuous integration/continuous deployment (CI/CD) processes. The following testing methods will be applied:

* Unit testing
* Integration testing
* End-to-end testing
* Performance testing
* Code quality analysis.

**Test Environment**

For TeamVas I will be using .NET 7, even though it has a standard term support, converting to a higher .NET will be easier since it will include the latest features. I except TeamVas to grow, this is why I will be using a MySQL database since growth in data can be efficiently managed in MySQL. I will use a C# asp.net backend with a React framework-based frontend. I will use GitHub for version control and GitHub actions for my CI/CD.

**Test Strategy**

Unit Testing with MSUnit: Testing individual components for correctness.

Integration Testing: Testing combined parts of the application to ensure they work together.

End-to-End Testing: Testing the entire application flow from start to finish. To ensure that the application works on from start to finish.

Performance Testing: Assessing the application’s performance under load.

Static Code Analysis with SonarQube: Ensuring code quality and adherence to standards. Make sure high parts of code are covered by tests.

Secure web development using OWASP: By looking at the OWASP top 10, I will see what the most critical security risks are to web applications. I will then take these risks into account in my web application.

**CI/CD Integration**

Setup a build pipeline that compiles the code and runs unit tests on every check-in. Build a test pipeline which executes integration, end-to-end and performance tests. Integrate SonarQube analysis in the CI/CD pipeline for better software quality. Automate building docker containers inside the CI/CD pipeline.

**Test Cases**

Unit Test Cases: Specific cases for unit testing each component.

Integration Test Cases: Scenarios covering the integration of different components and distributed systems.

End-to-End Test Cases: Scenarios mimicking real-world user behavior.

Performance Test Scenarios: Define load and stress test cases.