**Challenging Dom – Test Plan**

**Overview**

The front end should be testable through automation with development practices implemented to enable this. Integration tests will be primarily automated with a core set of manual regression tests completed. The manual element will be developed within a strict set of boundaries and executable in a defined period (Hours not days). As we continue to work in an Agile environment code and tests will be developed together with both reviewed under the same scrutiny. Automation tests to be executable locally and as part of CI. Metrics will be implemented to ensure that tests are executable within a specific period of time. Automation tests to be developed for each component (user interface in this instance) as well as overall integration tests.

## ****Key Principles****

* Application developed to allow for Automation
* Application code and tests are developed at the same time
* Test code reviewed with the same scrutiny as development code
* Automation tests are executable locally and via CI (execute in a defined time period)
* Automation tests exist for each component (user interface) and for integration
* Automation to be executed pre and post application deployment
* Non-functional requirements tested from the outset

**Development Pipeline (sample)**

| **STAGE** | **TEST ACTION** |
| --- | --- |
| Build | Unit Tests, Static code analysis |
| Deploy | Deployment, Automated regression, Automated performance |
| Post Deployment | Manual, Exploratory, Performance |

**Test Phases (Per Environment)**

**Local Environment (Single Feature)**

| **For Feature Under Test** | **For Product** |
| --- | --- |
| * Unit Testing * Static Code Analysis * Automation Testing * Manual Exploratory Testing * Code reviews and refactoring | * Impact based regression testing (As applicable) |

**Test Environment (One or more features)**

| **For Feature Under Test** | **For Product** |
| --- | --- |
| * Automation Testing * Manual Testing * Exploratory Testing | * Regression Testing * Performance Testing * Security Testing |

**Production Environment**

| **For Product** |
| --- |
| * Deployment Test (Automated if possible) * Production Monitoring (Including usage statics gathering) |

**Initial Test Approaches**

| **Manual** | **Automation** | **Non-functional** |
| --- | --- | --- |
| Test Scenarios   * Test scenarios should be light weight consisting of a mix of a small number of traditional test cases for major features and an exploratory approach for smaller stories (Checklist or Tour approach)   Tools  Test rail could be used to store manual test cases. Although to follow an automation first approach we could view the Feature file provided in BDD design as our overall test plan. | Server   * API Testing/Not Applicable for this task   UI   * C# Selenium, Web driver   Approach   * UI automated tests will be developed using BDD design principles. This utilises the use of the gherkin **Given**, **When**, **Then** syntax in test creation. * Requirements will lead the way in distinguishing what to automate, and priorities made for designated areas if required. * The automation framework must control data creation as part of the initial test setup, and also any test clean-up if required.   Tools   * C# with the Selenium Web driver bindings will be used for automated test development at the UI level. * Azure DevOps will be the CI tool that builds and executes automated tests in the cloud. * Automated tests will be executed as part of the release build process. * All test code will be stored in GitHub with QA and developers working closely in PR code reviews. | Security   * Not Applicable for this task   Performance   * JMeter/Not Applicable for this task |