Test Strategy

**ChequeMeOut**

notsirkApps

**Prepared By**

|  |  |
| --- | --- |
| DOCUMENT OWNER(S) | Organization role |
| Kriston Sanders | Project Manager |

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Change description |
| 1.0 | 5/28/20 | Kriston Sanders | Document created |

**Document Reference**

|  |  |
| --- | --- |
| DOCUMENT Name | File location |
| Software Quality Assurance Plan | Documentation folder |
| Risk Matrix | Documentation folder |
| Test Metrics Report | Documentation folder |
| Requirements Testing Matrix | Documentation folder |

**Approvals**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Approving party | Signature |
| 1.0 | 6/11/20 | Kriston Sanders | Kriston Sanders |

| 1. Scope and overview |
| --- |
| **Testing Policies**   * All system functions that are accessed through menus should be tested * Where user input is provided, all functions must be tested with both correct and incorrect input * Where calls to external data sources are made, functions must be tested with alternate data sources * All functions calling the CMO API must be tested * Test behavior not implementation |

| 1. Test approach |
| --- |
| * 1. **Test Levels**      1. **Development Testing**   Project will utilize test-driven development (TDD) methodologies during development. A CI pipeline will be created to trigger automated unit tests as changes are made to the code base. Component and system tests will be added to the pipeline as objects are integrated. Regression testing will also be automatic as all tests rerun when new code is uploaded.   * + 1. **Release Testing**   It may not be possible for a separate team to run release testing has this is a personal project; however, requirements will be tested with unit tests. Scenario testing will be done with associates who can run through the various scenarios and ensure each piece of the process runs smoothly.  Performance/stress testing will be performed through Azure DevOps as a feature of Test Plans.   * + 1. **User Testing**   User testing shall be carried out; however, many details are yet to be determined. The app will be distributed to associates who can carry out a form of beta testing.   * 1. **Test Types** * Test-driven development * Automated CI pipelines/unit testing * Component/system testing * Scenario testing * Load testing * Beta testing * Acceptance testing   1. **Roles and Responsibilities**  |  |  | | --- | --- | | **Role** | **Responsibilities** | | Project Manger | Review testing reports; ensure testing strategy is adhered to; arrange scenario and beta testing participants | | QA Lead | Identify test activities; plan testing; validate test automation; prepare metrics report and maintain Requirements Testing Matrix | | QA Tester | Design/administer tests; identify needed resources; develop test documentation |  * 1. **Environment Requirements** * PC with Visual Studio and access to the internet for Azure DevOps * Windows tablet * Macintosh through MacinCloud * iPhone * iPad * Android phones * Additional devices through App Center * Cloud or local server to run API and SQL database |

| 1. Testing tools |
| --- |
| * Azure DevOps for test management * PBIs in DevOps for bug tracking * CI pipelines in DevOps for automated unit testing |

| 1. standards |
| --- |
| See Software Quality Assurance Plan for details on standards to follow. |

| 1. Test deliverables |
| --- |
| 1. Test Strategy 2. Test Plan 3. Test Metrics Report 4. Test Scenarios 5. Requirement Traceability Matrix (RTM) 6. Defect Report/Bug Report |

| 1. Testing metrics |
| --- |
| * 1. **Process Metrics**  1. **Test Case Preparation Productivity**   Test cases per hour. Formula:  (No. of test cases) / (Effort spent for test case preparation)     1. **Test Design Coverage**   Measurement of test case coverage against the number of requirements. Formula:  ((Total no. of requirements mapped to test cases) / (Total no. of requirements))\*100     1. **Test Execution Productivity**   Number of test cases that can be executed per hour. Formula:  (No. of test cases executed) / (Effort spent for execution of test cases)     1. **Test Execution Coverage**   Number of test cases executed against the number of test cases planed. Formula:  ((Total no. of test cases executed) / (Total no. of test cases planned to execute))\*100     1. **Test Cases Passed**   Formula:  ((Total no. of test cases passed) / (Total no. of test cases executed)) \* 100     1. **Test Cases Failed**   Formula:  ((Total no. of test cases failed) / (Total no. of test cases executed)) \* 100     1. **Test Cases Blocked**   Formula:  ((Total no. of test cases blocked) / (Total no. of test cases executed)) \* 100     * 1. **Product Metrics**  1. **Error Discovery Rate**   Determines the effectiveness of the test cases. Formula:  ((Total no. of defects found) / (Total no. of test cases executed)) \* 100     1. **Defective Fix Rate**   Quality of a build in terms of defect repair. Formula:  (((Total no. of defects reported as fixed) – (Total no. of defects reopened)) / ((Total no. of defects reported as fixed) + (Total no. of new bugs due to fix))) \* 100     1. **Defect Density**   Ratio of defects to requirements. Defect density determines the stability of the application. Formula:  (Total no. of defects identified) / (Actual size (requirements))     1. **Defect Leakage**   Efficiency of the testing process before UAT. Formula:  ((Total no. of defects found in UAT) / (Total no. of defects found before UAT)) \* 100     1. **Defect Removal Efficiency**   Formula:  ((Total no. of defects found pre-delivery) / ((Total no. of defects found pre-delivery) + (Total no. of defects found post-delivery))) \* 100 |

| 1. Risk and mitigation |
| --- |

| Risk | Probability/ Impact (1 low to 5 high) | Score (0.2 low to 5.0 high) | Strategy |
| --- | --- | --- | --- |
| Code coverage incomplete | 3/3 | 1.8 | Maintained the Requirements Testing Matrix and leverage DevOps reporting to expand coverage |
| Unexpected quantity of bugs or performance problems | 3/3 | 1.8 | Focus time on resolving problems before implementing new features |
| Large changes nullify previous testing | 1/4 | 0.8 | Continue operating under test-driven development |
| Uncertainty surrounding performance/stress testing options | 3/3 | 1.8 | Seek out new tools |
| No Macintosh available for iOS simulation/compilation | 5/2 | 2.0 | Leverage MacinCloud service, consider secondhand device purchase, or seek loner/ leased unit |
| Limited number of test devices | 4/2 | 1.6 | Leverage Visual Studio App Center to run tests through the cloud |
| Single tester does not provide enough perspective or expertise | 5/3 | 3.0 | Consult associates who may be able to assist in testing |
| Cloud server costs fall outside of budget | 1/2 | 0.4 | Install local server for testing |
| Not enough/diverse enough to beta testers available | 4/2 | 1.6 | Research additional options to seek out beta testers |

| 8.0 Test summary |
| --- |
| Development testing will primarily be tracked in Azure DevOps. The Requirements Traceability Matrix will be updated throughout every iteration.  As bugs are discovered, they will be added to the backlog as well as having a bug report completed to provide details of the defect.  At the end of each iteration, a Test Metrics Report will be completed and a master report will be maintained for the entire project.  Test scenarios will be developed when release testing begins. |