SE 3XA3: Test Plan
ScrumBot

Team 304, ScrumBot
Arkin Modi, modia1
Leon So, sol4
Timothy Choy, choyt2

Last Updated: February 27, 2020

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	**Developer(s)**	**Change**
January 23, 2020	Arkin Modi	Copy template
February 20, 2020	Arkin Modi	Created the Purpose Section
February 27, 2020	Timothy Choy	Worked on Scope and Acronyms and Abbreviations
February 27, 2020	Leon So	Worked on Scope and Overview of Document
February 27, 2020	Leon So	Worked on Software Description
February 27, 2020	Arkin Modi	Worked on Test Team, Automated Testing Approach, and Testing Tools
February 27, 2020	Leon So	Worked on Tests for NFRs
February 27, 2020	Arkin Modi	Worked on the Unit Testing Plan for internal functions and output files

1 General Information

1.1 Purpose

The purpose of this document is to outline the testing, validation, and verification process of the functional and non-functional requirements, for the ScrumBot project. These test cases were conceived before the implementation and therefore will be used by the project members for future reference during the development process.

1.2 Scope

This test plan will provide a method to fully test ScrumBot by performing tests both at a modular level using unit tests created through Pytest, as well as at higher level, through using exploratory testing and specification-based testing. The unit tests cases will also cover partition testing, fuzz testing, and boundary testing.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: Table of Abbreviations

Abbreviation	**Definition**
CD	Continuous Delivery/Deployment
CI	Continuous Integration
POC	Proof of Concept
SRS	Software Requirements Specification

Table 3: Table of Definitions

Term	Definition
Acceptance Testing	A method of testing which is conducted to determine if the requirements of the specification are met
Boundary Testing	A method of testing where values are chosen on semantically significant boundaries
Code Inspection	A method of static testing where developers walk through the code
Dynamic Testing	A method of testing where code is executed
Exploratory Testing	A method of testing where the tester simultaneously learns the code while testing it. It approaches testing from a user's viewpoint
Fuzz Testing	A method of testing where random inputs are given to attempt to violate assertions
Integration Testing	A method of testing where individual software modules are combined and tested as a group
Partition Testing	A method of testing where the input domain is partitioned and input values are selected from the partitions
Pylint	A Python linter, used for static testing
Pytest	A unit testing framework for Python
ScrumBot	The Discord bot in development
Specification-based Testing	A method of testing where test cases are built based on the requirements specification
Static Testing	A method of testing where code is not executed
System Testing	A method of testing where the tests are performed on the system as a whole
Unit Testing	A method of testing focused on testing individual methods and functions

1.4 Overview of Document

This document outlines a test plan that fully encompasses all requirements of ScrumBot, as stated in the SRS. This document includes relevant information concerning: test team, automated testing, testing tools, testing schedule, unit-testing, and test cases.

2 Plan

2.1 Software Description

Scrum is an Agile process framework widely used in industry for managing and coordinating collaborative projects. Scrum being a process based on the agile development method, follows a highly iterative process and often has heavy customer involvement, therefore it can be often be complex. With Discord being a popular communication tool used by many teams of software developers today, ScrumBot provides a solution that directly integrates the management of a scrum development cycle into the communication channels. ScrumBot will allow for better management and organization of retrospectives, stand-ups, and other scrum/agile stages used by software teams within their routine communication channel.

2.2 Test Team

The test team will consist of all the members of the project: Arkin Modi, Leon So, and Timothy Choy.

2.3 Automated Testing Approach

Testing shall be automated with the use of the GitLab's CI/CD tool and the Pytest framework. The tests will be run every time a commit is pushed to the repository.

2.4 Testing Tools

The unit tests will be written using the Pytest framework. Static testing will be done with the use of Pylint.

2.5 Testing Schedule

See Gantt Chart at the following URL, https://gitlab.cas.mcmaster.ca/modia1/ScrumBot/-/blob/master/ProjectSchedule/.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Area of Testing1

Title for Test

1. test-id1
   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. test-id2
   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:

How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Non-Functional Requirements

3.2.1 Performance Requirements

Response Speed

1. NFRT-P1
   Type: Dynamic, Manual
   Initial State: No commands being made
   Input: Command entered
   Output: Response should be received within 2 seconds of the input being sent How test will be performed: The user will enter a command into the Discord channel with ScrumBot active. ScrumBot should provide a response within 2ms of the command being entered.

3.2.2 Security Requirements

HTTP Connections

1. NFRT-S1
   Type: Static, Manual, Code Inspection
   Initial State: N/A
   Input/Condition: N/A
   Output/Result: All connections between the system and the APIs use HTTPS requests
   How test will be performed: The test team will inspect the code and check if all connections between the system and APIs use HTTP requests

3.2.3 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1
   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. test-id2
   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

N/A

6 Unit Testing Plan

Unit testing will be performed through the use of the Pytest testing framework.

6.1 Unit testing of internal functions

Unit testing of internal functions will be performed for every function to ensure robustness. These tests will consist of a combination of partition testing and fuzz testing. Through this, verifying that the functions react in a predictable way. Following this, the functions will then undergo integration testing, to verify the compliance of the system with the SRS. During integration testing, stubs and drivers will be created as needed. For the testing of internal functions, the team will aim for a coverage of at minimum 85%.

6.2 Unit testing of output files

The only output file will be the application executable, which shall be tested for correctness as a whole as well as the fulfillment of the SRS. This testing will take the form of system testing and acceptance testing. The system testing will test the performance, the behavior under extreme/varying load, and scalability with the number of users. The acceptance testing will ensure the SRS has been fulfilled.

7 Appendix

Table 4: Traceability Matrix: Functional Requirement

	FRT-??	**FRT-??**
REQ?	X	
REQ?		X

Table 5: Traceability Matrix: Non-Functional Requirement

	NFRT-P1	**NFRT-S1**
P1	X	
S1		X