SE 3XA3: Test Plan
ScrumBot

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

Last Updated: February 28, 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
February 28, 2020	Timothy Choy	Worked on the Proof of Concept Testing
February 28, 2020	Leon So	Worked on Tests for NFRs
February 28, 2020	Timothy Choy	Updated Definitions, Worked on Tests for FRs
February 28, 2020	Leon So	Worked on Usability Survey, Worked on Tests for FRs
February 28, 2020	Arkin Modi	Worked on Traceability Matrices and Tests for FRs

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 and testing 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
EDT	Eastern Daylight Time (UTC-4)
EST	Eastern Standard Time (UTC-5)
FR	Functional Requirement
HTTP	HyperText Transfer Protocol
MVC	Model View Controller
NFR	Non-functional Requirement
POC	Proof of Concept
SRS	Software Requirements Specification
UTC	Coordinated Universal Time

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
Discord	A chat application. The platform in which ScrumBot will be implemented.
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
Kanban Board	A method of scheduling tasks through categorizing tasks to improve efficiency
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
Trello	A web based Kanban project management system
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 Installation

1. **FRT-BE1**
   Type: Specification-based, Dynamic, Manual
   Initial State: A Discord channel is active
   Input: ScrumBot is added into the channel
   Output: Various notifications, such as a link to documentation and basic commands
   How test will be performed: ScrumBot will be manually added to a Discord channel

3.1.2 Project Creation

1. **FRT-BE2-1**
   Type: Specification-based, Dynamic, Manual, Boundary
   Initial State: ScrumBot active in Discord channel
   Input: Add project command is entered, with specified details
   Output: Confirmation message
   How test will be performed: The test will be performed manually through commands entered in Discord. Manual testing will focus if the notifications are sent to the correct people as well as the interactions between ScrumBot and the tester

2. **FRT-BE2-2** Type: Specification-based, Dynamic, Unit, Boundary
   Initial State: ScrumBot active in Discord channel
   Input: Pytest file is run
   Output: Pytest output
   How test will be performed: The test will be performed through Pytest, where tests will run the command to add a project, focusing on the parameters entered to create projects

3.1.3 Project Removal

1. **FRT-BE3-1**
   Type: Specification-based, Dynamic, Manual, Boundary
   Initial State: ScrumBot active in Discord channel
   Input: project ID
   Output: The system shall as the Business Analyst of the project for confirmation. If confirmed, the project should be removed from project list. The system shall then ask if the project has been completed or cancelled. All project collaborators should be notified that the project has been removed.
   How test will be performed: The test team will attempt to remove a project in the Discord channel using an chat member account that is assigned with the role of Business Analyst. This test will be done manually in the chat channel where commands to remove a project will be entered. Boundary cases will also be tested, such as attempting to remove a project from an empty project list

2. **FRT-BE3-2**
   Type: Dynamic, Unit, Boundary
   Initial State: ScrumBot active in Discord channel
   Input: Pytest file is run
   Output: Pytest output
   How test will be performed: The test will be performed through Pytest, where the test will run the command to add a project, focusing on the parameters entered to remove projects. The Pytest file will also include tests which tests boundary conditions, such

as attempting to remove a project from an empty project list

3.1.4 Sprint-Planning Meeting

1. **FRT-BE4-1**
   Type: Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Sprint-planning meeting command is run
   Output: ScrumBot outputs message stating that a sprint-planning meeting has started with relevant commands for sprint-planning meetings
   How test will be performed: A tester will execute the command to start the meeting from a user account assigned to the role of Scrum Master

2. **FRT-BE4-2**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel, Sprint-planning meeting in progress
   Input: Progress information is entered
   Output: ScrumBot stores or shows information based on the input
   How test will be performed: A tester with the Scrum Master role will run commands to see if they can add goals, feedback, backlog tasks, tasks, as well as update the Trello board. A tester with the Development Team role will run commands to check whether or not they can see new feedback, updated backlog tasks, new goals, as well as assign tasks and update the Trello board

3.1.5 Stand-up Meeting

1. **FRT-BE5-1**
   Type: Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Stand-up meeting command is run
   Output: ScrumBot outputs message stating the stand-up meeting has started with relevant commands for stand-up meetings
   How test will be performed: A tester will execute the command to start the meeting from an user account assigned the role of Scrum Master

2. **FRT-BE5-2**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel, Stand-up meeting in progress
   Input: Progress information is entered
   Output: ScrumBot stores progress information and the corresponding member
   How test will be performed: A tester with Scrum Master role will start a stand-up meeting in the channel. The members of the test team will test commands to add and

update progress from various user accounts with the Development Team member role and Scrum Master role

3.1.6 Retrospective Meeting

1. **FRT-BE6-1**
   Type: Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Retrospective meeting command is run
   Output: ScrumBot outputs message stating the stand-up meeting has started with relevant commands for retrospective meetings
   How test will be performed: A tester will execute the command to start the meeting from an user account assigned the role of Scrum Master

2. **FRT-BE6-2**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel, Retrospective meeting in progress
   Input: Feedback and details are entered using command
   Output: ScrumBot stores the feedback and corresponding details and feedback is visible to all members of the development team
   How test will be performed: A tester with Scrum Master role will start a retrospective meeting in the channel. A tester with the role of Scrum Master will add feedback. Another member of the test team with the role of Development Team member will see if the feedback and feedback details are visible

3.1.7 Grooming Meeting

1. **FRT-BE7-1**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Grooming meeting command is run
   Output: ScrumBot outputs message stating the grooming meeting has started with relevant commands for grooming meetings
   How test will be performed:

2. **FRT-BE7-2**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel, Grooming meeting in progress
   Input: Progress information is entered
   Output: ScrumBot stores the information and shows any necessary information
   How test will be performed: A tester with the role of Scrum Master or Development Team will see all updates regarding project requirements, tasks and deadlines. A tester

with the role of Business Analyst will see the same, excluding tasks.

3.1.8 Add a Meeting

1. **FRT-BE8-1**
   Type: Dynamic, Unit
   Initial State: ScrumBot active in Discord channel
   Input: Pytest file is run
   Output: Pytest output
   How test will be performed: The test will be performed through Pytest, where tests will run the command to add meetings, focusing on the parameters entered to create meetings

2. **FRT-BE8-2**
   Type: Specification-based, Dynamic, Boundary, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Cancel meeting command is entered, with a specified meeting
   Output: The specified meeting will be removed, otherwise an error message is raised
   How test will be performed: The test will be performed manually through commands entered in Discord. Manual testing primarily will focus on if notifications are sent to the proper people (development team members, business analysts and scrum masters), as well as the interactions between ScrumBot and the tester

3.1.9 Cancel a Meeting

1. **FRT-BE9-1**
   Type: Dynamic, Boundary, Unit
   Initial State: ScrumBot active in Discord channel
   Input: Pytest file is run
   Output: Pytest output
   How test will be performed: The test will be performed through Pytest, where tests will run the command to cancel meetings, in situations where the meeting exists and where the meeting does not exist

2. **FRT-BE9-2**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Cancel meeting command is entered, with a specified meeting
   Output: The specified meeting will be removed, otherwise an error message is raised
   How test will be performed: The test will be performed manually through commands entered in Discord. Manual testing primarily will focus on if notifications are sent to

the proper people (development team members, business analysts and scrum masters)

3.1.10 List Scheduled Meetings

1. **FRT-BE10-1**
   Type: Specification-based, Dynamic, Boundary, Unit
   Initial State: ScrumBot active in Discord channel
   Input: Pytest file is run
   Output: Pytest output
   How test will be performed: The test will be performed through Pytest, where tests will run the command to list meetings, in situations where there are meetings scheduled and when there are no meetings

2. **FRT-BE10-2**
   Type: Specification-based, Dynamic, Boundary, Manual
   Initial State: ScrumBot active in Discord channel
   Input: The list scheduled meetings command is entered into Discord
   Output: ScrumBot should list out the list of scheduled meetings
   How test will be performed: The test will be performed manually through commands entered in Discord. Test cases be similar to the unit tests above

3.1.11 See Tasks

1. **FRT-BE11**
   Type: Specification-based, Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: The see tasks command is entered into Discord
   Output: A link to the Trello Kanban board, if integrated
   How test will be performed: The test will be performed manually through commands entered in Discord

3.2 Tests for Non-Functional Requirements

3.2.1 Look and Feel Requirements

Appearance Requirements

1. **NFRT-UH13-1**
   Type: Manual, Dynamic, Checklist
   Initial State: ScrumBot active in Discord channel
   Input: Set of all ScrumBot commands entered
   Output: ScrumBot outputs text formatted according to the text format of ScrumBot
   How test will be performed: The test team will run commands in the Discord chat

with ScrumBot active and check if the text format is appropriate and in accordance with Discord's text format

Style Requirements

1. **NFRT-UH13-2**
   Type: Manual, Dynamic, Checklist
   Initial State: ScrumBot active in Discord channel
   Input: One of each role available is added
   Output: ScrumBot adds the roles and permissions to the assigned users. The role names should be short, clear, and descriptive. The roles should be colour coded
   How test will be performed: The test team will add roles in the Discord channel. One of each role available will be assigned

3.2.2 Usability and Humanity Requirements

Ease of Use & Personalization and Internationalization Requirements

1. **NFRT-UH13-3**
   Type: Manual, Usability Survey
   How test will be performed: The test team will ask a sample set of users to answer questions on a usability survey after using ScrumBot for the first time

Learning Requirements

1. **NFRT-UH4**
   Type: Dynamic, Manual
   Initial State: ScrumBot active in Discord channel
   Input: Help command entered
   Output: ScrumBot should output help menu
   How test will be performed: The test team will enter the help command into Discord chat with ScrumBot active

3.2.3 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 test team will enter a command into the Discord channel with ScrumBot active. ScrumBot should provide a response within 2ms of the command being entered

Meeting Schedule Accuracy

1. **NFRT-P2**
   Type: Dynamic, Manual
   Initial State: No meetings schedules
   Input: Schedule meeting command with meeting details
   Output: The meeting is added to the meeting list with the correct location
   How test will be performed: The test team will enter the command to schedule a meeting and the meeting details into the Discord channel with ScrumBot active

2. **NFRT-P3**
   Type: Dynamic, Manual
   Initial State: No meetings schedules
   Input: Schedule meeting command with meeting details
   Output: The meeting is added to the meeting list with the correct time
   How test will be performed: The test team will enter the command to schedule a meeting and the meeting details into the Discord channel with ScrumBot active

3.2.4 Operational and Environmental Requirements

Expected Environment

1. **NFRT-OE1**
   Type: Dynamic, Manual
   Initial State: New Discord channel without ScrumBot
   Input/Condition: Add ScrumBot to the Discord channel
   Output/Result: ScrumBot should be fully functional in the Discord channel once added
   How test will be performed: The test team will follow the provided installation documentation and add ScrumBot to a brand new Discord channel

Requirements for Interfacing with Adjacent Systems

1. **NFRT-OE2**
   Type: Dynamic, Manual
   Initial State: ScrumBot added to Discord channel not yet connected to users' Google services
   Input/Condition: User wants to connect their Google services to ScrumBot
   Output/Result: ScrumBot should be connect to the User's Google account through Google API services
   How test will be performed: The test team will attempt to connect their Google services to ScrumBot

2. **NFRT-OE3**
   Type: Dynamic, Manual
   Initial State: ScrumBot not yet registered as a public bot
   Input/Condition: ScrumBot register with Discord API services and deployed publically
   Output/Result: ScrumBot should be a public bot registered with Discord

How test will be performed: The test team will follow documentation provided by Discord to add ScrumBot as a public bot

3. **NFRT-OE4**
   Type: Dynamic, Manual
   Initial State: ScrumBot added to Discord channel not yet connected to users' Trello services
   Input/Condition: User wants to connect their Trello services to ScrumBot
   Output/Result: ScrumBot should be connect to the User's Trello account through Trello API services
   How test will be performed: The test team will attempt to connect their Trello services to ScrumBot

Installability Requirements

1. **NFRT-OE5**
   Type: Dynamic, Manual
   Initial State: ScrumBot not yet added to Discord channel
   Input/Condition: User without expertise in Discord bots wants to install ScrumBot
   Output/Result: ScrumBot added to Discord channel
   How test will be performed: An inexperienced user will attempt to add ScrumBot to a Discord channel by following the provided documentation

2. **NFRT-OE6**
   Type: Dynamic, Manual
   Initial State: ScrumBot not yet added to Discord channel
   Input/Condition: Start installation of ScrumBot
   Output/Result: ScrumBot takes less than 5 minutes to install
   How test will be performed: Test team will install ScrumBot and time the installation

3. **NFRT-OE7**
   Type: Dynamic, Manual
   Initial State: ScrumBot installed
   Input/Condition: Start uninstalling Scrumbot
   Output/Result: ScrumBot is uninstalled
   How test will be performed: The test team will attempt to uninstall ScrumBot

3.2.5 Maintainability and Support Requirements

Maintainability Requirements

1. **NFRT-MS1**
   Type: Static, Manual, Code Inspection
   Initial State: N/A
   Input/Condition: N/A
   Output/Result: The code is well documented with comments

How test will be performed: The test team will inspect the code and check if the code is adequately documented using comments

2. **NFRT-MS2**
   Type: Static, Manual
   Initial State: No Doxygen documents generated yet
   Input/Condition: Generate Doxygen HTML and pdf
   Output/Result: The Doxygen documentation is successfully compiled and generated
   How test will be performed: The test team will try to generate Doxygen HTML and pdf

3. **NFRT-MS3**
   Type: Static, Manual, Code Inspection
   Initial State: N/A
   Input/Condition: N/A
   Output/Result: The code documentation is easy to understand
   How test will be performed: The test team will review the code documentation and make sure it is easily understandable. The test team will survey other developers and verify that they too can easily understand the documentation

Supportability Requirements

1. **NFRT-MS4-1**
   Type: Dynamic, Manual
   Initial State: ScrumBot is active on the Discord channel
   Input/Condition: User wants to open help menu
   Output/Result: Help menu is displayed in the Discord chat
   How test will be performed: The test team will attempt to open the help menu in the Discord chat

Longevity Requirements

1. **NFRT-MS4-2**
   Type: Static, Manual, Code Inspection
   Initial State: N/A
   Input/Condition: N/A
   Output/Result: System is modularized into classes
   How test will be performed: The test team will perform a code inspection to ensure that the system is separated into modules

3.2.6 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

3.3 Traceability Between Test Cases and Requirements

For the Traceability Matrix: Functional Requirement, the “FRT-” has been drop from the test cases names (top row in the table) due to page space limitations.

Table 4: Traceability Matrix: Functional Requirement

Test IDs	**Requirement IDs**										
	BE1	**BE2**	**BE3**	**BE4**	**BE5**	**BE6**	**BE7**	**BE8**	**BE9**	**BE10**	**BE11**

For the Traceability Matrix: Non-Functional Requirement, the "NFRT-" has been drop from the test cases names (top row in the table) due to page space limitations.

Table 5: Traceability Matrix: Non-Functional Requirement

Req.	**Test IDs**																			
	UH13-1	**UH13-2**	**UH13-3**	**UH4**	**P1**	**P2**	**P3**	**OE1**	**OE2**	**OE3**	**OE4**	**OE5**	**OE6**	**OE7**	**MS1**	**MS2**	**MS3**	**MS4-1**	**MS4-2**	**S1**
UH1	X	X	X																	
UH2	X	X	X																	
UH3	X	X	X																	
UH4				X																
P1					X															
P2						X														
P3							X													
OE1								X												
OE2									X											
OE3										X										
OE4											X									
OE5												X								
OE6													X							
OE7														X						
MS1															X					
MS2																X				
MS3																	X			
MS4																		X	X	
S1																				X

4 Tests for Proof of Concept

4.1 Issues and Conflicts

The proof of concept demonstration for ScrumBot consisted on a simple Python discord bot performing basic front end tasks such as creating, listing, and deleting meeting times. There was no database connected to the proof of concept, so memory was not stored from instance to instance.
Issues that were found with the proof of concept were:

1. The lack of a priority list for features
2. The lack of a defined software architectural style
3. The need for time zones, as people could be connecting to meetings from around the world

4.2 Resolution

To resolve these issues, we have taken the list of business events from our SRS and have assigned priority to the events. In the case where ScrumBot will not be able to fulfill all the requirements in the given timeframe, ScrumBot will still be able to function as the prioritized functionalities will be implemented.

In regards to the chosen architectural style, we have decided on using MVC as our primary architectural style. This best suits ScrumBot as the view module will be all the input and output from Discord, our controller module will be the commands run, and our model module will be the databases containing all the meeting information.

To tackle the issue regarding time zones, we plan on writing our times in EST or EDT, based on the date scheduled for the meeting. The choice is simply because of our current location, and makes it simpler for us to test and create meetings. However, a new feature we plan on implementing is the ability to convert between time zones given a meeting.

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.

References

7 Appendix

7.1 Usability Survey Questions

These survey questions will be used to test usability requirements.

1. Are you 13+ years old?
2. (On a scale from 1-10) How intuitive did you find ScrumBot's commands?
3. Did you find ScrumBot commands unnecessarily complex?
4. Did you find ScrumBot easy to use?
5. Would you use ScrumBot on a regular basis?
6. Do you agree that the language used by ScrumBot is appropriate for English speakers?