SE 3XA3: Test Plan
ScrumBot

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

Last Updated: April 6, 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, Software Description
February 28, 2020	Arkin Modi	Worked on Traceability Matrices and Tests for FRs
February 28, 2020	Leon So, Timothy Choy	Proofread Document
February 28, 2020	Everyone	Final Revisions
April 6, 2020	Arkin Modi, Leon So	Revised all test cases

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
Business Analyst	Communicates and coordinates project requirements and deadlines between the Product Owner(s), Scrum Master, and Development Team
Checklist	A method of testing through inspecting the code
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
Grooming	A meeting where the Business Analyst communicates and coordinates project requirements and deadlines with the Scrum Master and Development Team
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
Retrospective	A team meeting for reflecting on an Scrum sprint
ScrumBot	The Discord bot in development
Scrum Master	The facilitator for an agile development team who plans, leads and organizes Scrum meetings
Specification-based Testing	A method of testing where test cases are built based on the requirements specification

Term	Definition
Sprint	A set time period where specific work has to be completed and made ready for review
Stand-up	A daily coordination meeting used in the Scrum framework
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 specified 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. ScrumBot will provide features to add and manage Scrum meetings, as well as to store information relevant to those meetings. ScrumBot will also allow Scrum roles to be assigned to members of the Discord 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 partially 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. Manual testing will be done by the test team.

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

All previous test cases have been changed.

3.1.1 Installation

1. **FRT-BE1**
   Type: Functional, Dynamic, Manual
   Initial State: A Discord channel is active
   Input: ScrumBot is added into the channel
   Output: ScrumBot welcome message
   How test will be performed: ScrumBot will be manually added to a Discord channel.

3.1.2 Project Creation

1. **FRT-BE2-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Empty project list
   Input: Create New project, with description
   Output: New project created and added to list
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to create a project.

2. **FRT-BE2-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Empty project list
   Input: Create new project, without description
   Output: New project created and added to list
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to create a project.

3.1.3 Project Removal

1. **FRT-BE3-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project list containing one project with ID 0
   Input: Remove project by ID, ID is 0
   Output: Project list empty, project with ID 0 is removed
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a project.

2. **FRT-BE3-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Empty project list
   Input: Remove project by ID
   Output: KeyError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a project.

3.1.4 Sprint-Planning Meeting

1. **FRT-BE4-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and no tasks
   Input: Add a task without details
   Output: Task added to sprint
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add task to a sprint.

2. **FRT-BE4-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and no tasks
   Input: Add a task with details
   Output: Task added to sprint
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add task to a sprint.

3. **FRT-BE4-3**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no sprints and no tasks
   Input: Add a task
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add task to a sprint.

4. **FRT-BE4-4**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and tasks

Input: Get all tasks by sprint IDs
Output: A list of tasks
How test will be performed: Pytest will run the internal commands that ScrumBot will use to get all tasks from sprint.

5. **FRT-BE4-5**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with sprints and tasks, contains task with ID 0
   Input: Get a single task with ID 0
   Output: Task with ID 0
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a single task from sprint.

6. **FRT-BE4-6**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project no sprints and no tasks
   Input: Get a single task with ID 0
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a single task from a sprint.

7. **FRT-BE4-7**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project no sprints and no tasks
   Input: Get a list of tasks
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a list of tasks from a sprint.

8. **FRT-BE4-8**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and a task
   Input: Add feedback to a task by task ID
   Output: Feedback added to task of last sprint
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add feedback to a task in the last sprint.

9. **FRT-BE4-9**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no sprint and no task
   Input: Add feedback to a task by task ID
   Output: Index Error
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add feedback to a task in the last sprint.

10. **FRT-BE4-10**
    Type: Functional, Dynamic, Unit, Automated

Initial State: Project a sprint, a task, and a feedback
Input: Get feedback from task using sprint ID and task ID
Output: Feedback of sprint with inputted sprint ID and task with inputted task ID
How test will be performed: Pytest will run the internal commands that ScrumBot will use to get feedback from task and sprint.

11. **FRT-BE4-11**
    Type: Functional, Dynamic, Unit, Automated
    Initial State: Project with no sprint, no task
    Input: Get feedback from task using sprint ID and task ID
    Output: IndexError
    How test will be performed: Pytest will run the internal commands that ScrumBot will use to get feedback from task and sprint.

3.1.5 Stand-up Meeting

1. **FRT-BE5-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and a task
   Input: Remove task by task ID
   Output: Task removed from last sprint
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a task from the last sprint.

2. **FRT-BE5-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no sprint
   Input: Remove task by task ID
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a task from the last sprint.

3. **FRT-BE5-3**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and no tasks
   Input: Remove task by task ID
   Output: KeyError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a task from the last sprint.

4. **FRT-BE5-4**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint and a task with ID 0
   Input: Set details of task with ID 0
   Output: Task with ID 0 updates with new details
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to set description of a task from the last sprint.

5. **FRT-BE5-5**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no sprints
   Input: Set details of task with ID 0
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to set description of a task from the last sprint.

3.1.6 Retrospective Meeting

1. **FRT-BE6-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with a sprint, a task and feedback
   Input: Remove feedback by sprint and task IDs
   Output: Feedback removed
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove feedback from specified sprint and task.

2. **FRT-BE6-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no sprint
   Input: Remove feedback by sprint and task IDs
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove feedback from specified sprint and task.

3.1.7 Grooming Meeting

1. **FRT-BE7-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no requirements
   Input: Add requirement
   Output: Requirement added
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add a requirement to the project.

2. **FRT-BE7-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with requirements
   Input: Get a list of requirement
   Output: A list of requirements
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a list of requirements to the project.

3. **FRT-BE7-3**
   Type: Functional, Dynamic, Unit, Automated

Initial State: Project with requirements
Input: Remove a requirement by requirement ID
Output: Requirement removed
How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a requirement from the project.

4. **FRT-BE7-4**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no requirements
   Input: Remove a requirement by requirement ID
   Output: IndexError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a requirement from the project.

3.1.8 Add a Meeting

1. **FRT-BE8-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no meetings
   Input: Add a meeting with a description
   Output: Meeting added
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add a meeting to the project.

2. **FRT-BE8-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no meetings
   Input: Add a meeting with no description
   Output: Meeting added
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to add a meeting to the project.

3. **FRT-BE8-3**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings
   Input: Get a meeting by meeting ID
   Output: Meeting with specified ID
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a meeting from the project.

4. **FRT-BE8-4**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings that contains meeting with ID 0
   Input: Get meeting name with ID 0
   Output: Meeting name of meeting with ID 0
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the name of a meeting from the project.

5. **FRT-BE8-5**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings that contains meeting with ID 0
   Input: Get meeting description with ID 0
   Output: Meeting description of meeting with ID 0
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the description of a meeting from the project.

6. **FRT-BE8-6**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings that contains meeting with ID 0 and has no description
   Input: Get meeting description with ID 0
   Output: Meeting description of meeting with ID 0
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the description of a meeting from the project.

7. **FRT-BE8-7**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings that contains meeting with ID 0 and has a description
   Input: Get meeting description with ID 0
   Output: "No description"
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the description of a meeting from the project.

8. **FRT-BE8-8**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no meetings
   Input: Get meeting name with ID 0
   Output: KeyError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the name of a meeting from the project.

9. **FRT-BE8-8**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with no meetings
   Input: Get meeting description with ID 0
   Output: KeyError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get the description of a meeting from the project.

10. **FRT-BE8-9**
    Type: Functional, Dynamic, Unit, Automated
    Initial State: Project with no meetings
    Input: Add a meeting of every meeting type (4 meetings in total)

Output: Added meeting
How test will be performed: Pytest will run the internal commands that ScrumBot will use to add a meeting to the project.

11. **FRT-BE8-10**
    Type: Functional, Dynamic, Unit, Automated
    Initial State: Project with no meetings
    Input: Add a meeting of an invalid meeting type
    Output: TypeError
    How test will be performed: Pytest will run the internal commands that ScrumBot will use to add a meeting to the project.

3.1.9 Cancel a Meeting

1. **FRT-BE9-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings and contains meeting with ID 0
   Input: Remove meeting with ID 0
   Output: Meeting with ID 0 removed
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a meeting from the project.

2. **FRT-BE9-2**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with meetings and contains no meetings with ID 0
   Input: Remove meeting with ID 0
   Output: KeyError
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to remove a meeting from the project.

3.1.10 List Scheduled Meetings

1. **FRT-BE10-1**
   Type: Functional, Dynamic, Unit, Automated
   Initial State: Project with four meetings, each meeting of a different type
   Input: Get a list of all meetings
   Output: A list of all meetings
   How test will be performed: Pytest will run the internal commands that ScrumBot will use to get a list of meetings from the project.

3.2 Tests for Non-Functional Requirements

All previous test cases have been changed.

3.2.1 Look and Feel Requirements

Appearance Requirements

1. **NFRT-LF-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-LF-2**
   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. A sample survey is included in the appendix. Questions will be asked regarding their opinion on the clarity of role names, and how these roles are represented on Discord. A sample survey is included in the appendix.

2. **NFRT-LF-3**
   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 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-UH-1**
   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. A sample survey is included in the appendix.

Learning Requirements

1. **NFRT-UH-2**
   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-P-1**
   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. A helper method will record the time and report to the tester the total time taken.

3.2.4 Operational and Environmental Requirements

Expected Environment

1. **NFRT-OE-1**
   Type: Dynamic, Manual
   Initial State: New Discord channel without ScrumBot
   Input/Condition: Add ScrumBot to the Discord channel
   Output/Result: ScrumBot should be active 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

Installability Requirements

- **NFRT-OE-2**
  Type: Dynamic, Manual
  Initial State: ScrumBot not yet installed on Discord server
  Input/Condition: Establish connection between Scrumbot and Discord
  Output/Result: Scrumbot should connect to Discord
  How test will be performed: The test team will follow documentation provided by Discord to add ScrumBot to the Discord server

3.2.5 Maintainability and Support Requirements

Maintainability Requirements

1. **NFRT-MS-1**
   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-MS-2**
   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-MS-3**
   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-MS-4**
   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-MS-5**
   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-S-1**
   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

Test IDs correspond to the tests found in this document. Requirement IDs correspond to the requirements found in the SRS.

Tables were updated to reflect new test cases.

Table 4: Traceability Matrix: Functional Requirement

Test IDs	**Requirement IDs**										
	BE1	**BE2**	**BE3**	**BE4**	**BE5**	**BE6**	**BE7**	**BE8**	**BE9**	**BE10**	**BE11**
FRT-BE1	X										
FRT-BE2-1		X									
FRT-BE2-2		X									
FRT-BE3-1			X								
FRT-BE3-2			X								
FRT-BE4-1				X							
FRT-BE4-2				X							
FRT-BE4-3				X							
FRT-BE4-4				X							X
FRT-BE4-5				X							
FRT-BE4-6				X							
FRT-BE4-7				X							
FRT-BE4-8				X							
FRT-BE4-9				X							
FRT-BE4-10				X							
FRT-BE4-11				X							
FRT-BE5-1					X						
FRT-BE5-2					X						
FRT-BE5-3					X						
FRT-BE5-4					X						
FRT-BE5-5					X						
FRT-BE6-1						X					
FRT-BE6-2						X					
FRT-BE7-1							X				
FRT-BE7-2							X				
FRT-BE7-3							X				
FRT-BE7-4							X				
FRT-BE8-1								X			
FRT-BE8-2								X			
FRT-BE8-3								X			
FRT-BE8-4								X			
FRT-BE8-5								X			
FRT-BE8-6								X			
FRT-BE8-7								X			
FRT-BE8-8								X			
FRT-BE8-9								X			
FRT-BE8-10								X			
FRT-BE9-1									X		
FRT-BE9-2									X		
FRT-BE10-1										X	

Table 5: Traceability Matrix: Non-Functional Requirement

Test IDs	**Requirement IDs**																				
	LF1	**LF2**	**LF3**	**UH1**	**UH2**	**UH3**	**UH4**	**P1**	**P4**	**OE1**	**OE3**	**OE5**	**OE6**	**OE7**	**MS1**	**MS2**	**MS3**	**MS4**	**S1**	**C1**	**L1**	**HS1**
NFRT-LF-1	X																					
NFRT-LF-2		X	X																			
NFRT-LF-3		X																				
NFRT-UH-1				X	X																	
NFRT-UH-2							X															
NFRT-P-1								X														
NFRT-OE-1										X												
NFRT-OE-2											X											
NFRT-MS-1												X										
NFRT-MS-2													X									
NFRT-MS-3														X								
NFRT-MS-4															X							
NFRT-MS-5																X						
NFRT-S-1																	X					

4 Tests for Proof of Concept

4.1 Issues and Conflicts

The proof of concept demonstration for ScrumBot consisted of 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.

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? If so, explain.
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?
7. Are the role names clear and understandable?
8. Is the system and user commands intuitive and easy to use?