**System Test Plan**

**For**

**UAV Swarm**

Team Members:

* Samantha Ballistreri
* Tyler Wise
* Joseph Moran
* Michael Fornito
* William Edwards
* Daniela Regueira

|  |  |
| --- | --- |
| Version/Author | Date |
| 1.0 Daniela Regueira, Tyler Wise | 11/08/2020 |
| 2.0 Tyler Wise, Michael Fornito | 11/29/2020 |
| 3.0 The whole team | 11/30/2020 |
|  |  |

Table of Contents

[**1. Introduction**](#_heading=h.3c8xpneikfu) **1**

[1.1 Purpose](#_heading=h.1fob9te) 2

[1.2 Objectives](#_heading=h.3znysh7) 2

[**2. Functional Scope**](#_heading=h.2et92p0) **2**

[**3. Overall Strategy and Approach**](#_heading=h.zc2kw9gr9m7p) **3**

[3.1 Testing Strategy](#_heading=h.3dy6vkm) 3

[3.2 System Testing Entrance Criteria](#_heading=h.1t3h5sf) 3

[3.3 Testing Types](#_heading=h.4d34og8) 4

[3.3.1 Usability Testing](#_heading=h.7dag15bmh7mn) 4

[3.3.2 Functional Testing](#_heading=h.ir8p0x1vhvp) 4

[3.4 Suspension Criteria and Resumption Requirements](#_heading=h.2s8eyo1) 4

[3.4.1 Suspension Criteria](#_heading=h.6xkgmloh2by) 4

[3.4.2 Resumption Requirements](#_heading=h.6xda0s8och0) 4

[**4. Execution Plan**](#_heading=h.17dp8vu) **4**

[**5. Traceability Matrix & Defect Tracking**](#_heading=h.26in1rg) **5**

[5.1 Traceability Matrix](#_heading=h.lnxbz9) 5

[5.2 Defect Severity Definitions](#_heading=h.35nkun2) 7

[**6. Environment**](#_heading=h.8buikr4hkyc8) **8**

[**7. Assumptions**](#_heading=h.z337ya) **8**

[**8. Risks and Contingencies**](#_heading=h.3j2qqm3) **8**

# 1. Introduction

## 1.1 Purpose

This document is a test plan for the UAV Swarm System Testing, produced by the System Testing team. It describes the testing strategy and approach to testing the team will use to verify that the application meets the established requirements of the customer prior to release.

## 1.2 Objectives

The objectives for testing the simulation are:

* Test all acceptance criteria specified in the project’s SRS document.
* Test integration of all system modules.
* Satisfies the criteria outlined in the project vision statement.

# 2. Functional Scope

The requirements in the scope of testing for the UAV Swarm System include:

|  |  |
| --- | --- |
| Requirement | Description |
| F1 | The python script, StartUp, must connect to the Unreal Engine |
| F2 | The simulation with the selected number of drones will start. |
| F3 | The simulation will end. |
| F4 | The simulation will run the formation selected. |
| F5 | Run the python test script that puts a non int or float data type as the parameter for adding drones. An error message,can only input a number, shall appear. |
| F6 | Run the python test script that puts a non int or float data type as the parameter for removing drones. An error message, can only input a number, shall appear. |
| F7 | Run the python test script that removes a drone, but there are only 1 drones in the settings file. An error, must have at least one drone message, shall appear. |
| F8 | Run the python test script that adds more than 10 drones. An error, cannot have more than 10 drones, message shall appear. |
| F9 | Run the python test script that puts a non int or float data type as the parameter for drone distribution size. An error message, can only input a number, shall appear. |
| F10 | Run the python test script that removes a drone at an index larger than the amount of drones in the settings files. An error, out of bounds, message shall appear |
| HF1 | The user clicks on the Unreal start button. The user clicks the run button on the python script called StartUp. |
| HF2 | The user has followed HF1 procedure, the user shall visually see that the swarm has started |
| HF3 | The user clicks pause and the swarm will halt within the Unreal World |
| HF4 | The user clicks the stop button and the swarm will disappear from the Unreal World. |
| HF5 | The user shall be notified which drone(s) have failed during the simulation. |
| HF6 | The user shall be notified which drone(s) have successfully finished the simulation. |
| HF7 | The user shall be able to see the perspective of the selected drone flying one at a time. |
| HF8 | The user types in the terminal to add a distribution size resulting a modification of the settings files |
| HF9 | The user types in the terminal to add the drone with its position |
| HF10 | The user types in the terminal to remove the drone based on its index |
| HF11 | The user types in the terminal to display the drones |

# 3. Overall Strategy and Approach

## 3.1 Testing Strategy

UAV Swarm System Testing will include testing of all functionalities that are in scope (Refer to Functional Scope Section) identified. System testing activities will include the testing of any new or modified functionalities. This includes, but is not limited to, Unreal Engine integrations, AirSim integrations, user interfaces,

## 3.2 System Testing Entrance Criteria

In order to start system testing, certain requirements must be met for testing readiness. The readiness can be classified into:

* Module compliance with specified system design criteria as outlined in the project’s System Design Document
* Module evaluation as complete from all developers involved in work
* Code review and approval from all members of the team

## 3.3 Testing Types

### 3.3.1 Usability Testing

User interface attributes, cosmetic presentation and content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is functional and provides the user with consistent and appropriate access and navigation through the functions of the application (e.g., Python manipulation of settings files, Unreal Engine Editor, etc.)

### 3.3.2 Functional Testing

The objective of this testing is to ensure that each element of the component meets the functional requirements of the customer as outlined in the following sections of the project’s SRS:

* [Functional Requirements](https://docs.google.com/document/d/1OrtbVfPCt8Wtusz_93U02ICh25k6fr7iR1zvU9yq8hY/edit#heading=h.hiwtdbcsidte)
* [User and Human Factors Requirements](https://docs.google.com/document/d/1OrtbVfPCt8Wtusz_93U02ICh25k6fr7iR1zvU9yq8hY/edit#heading=h.24xogj971hg5)

## 3.4 Suspension Criteria and Resumption Requirements

### 3.4.1 Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the software, it is up to the Testing Manager to determine whether the test plan will be re-executed or part of the plan will be re-executed.

### 3.4.2 Resumption Requirements

Resumption of testing will be possible when the functionality that caused the suspension of testing has been retested successfully or the functionality under test has been removed from the iteration.

# 4. Execution Plan

|  |  |  |  |
| --- | --- | --- | --- |
| Activities | Task | Owner | Date |
| Plan the test process and map test to requirements | F1-F10 | Tyler Wise, Michael Fornito | 11/07/2020,11/29/30 |
| Plan the test process and map test to requirements | HF1-HF11 | Tyler Wise, Michael Fornito | 11/07/2020, 11/29/30 |
| Execute tests | F1-F10 | Will Edwards | 11/30/2020 |
| Execute tests | HF1-HF11 | Michael Fornito | 11/30/2020 |
| Track the tests and test results | F1-F10 | Will Edwards | 11/30/2020 |
| Track the tests and test results | HF1-HF11 | Michael Fornito | 11/30/2020 |

# 5. Traceability Matrix & Defect Tracking

## 5.1 Traceability Matrix

This list of requirements and corresponding test cases are enumerated in greater detail in the project’s SRS document. Inclusion of this list as part of the test plan is for

|  |  |  |  |
| --- | --- | --- | --- |
| Requirement/Test Case Number | Description | Pass/Fail | Notes |
| F1/FTC1 | Click the start button in Unreal Engine. Click the start button in the user’s choice of IDE to execute the StartUp python script. If successful it will say “connected”, if not it will say “not connected” in the python console. | Pass |  |
| F2/FTC2 | The simulation with the at least 1 drone in the settings file. The simulation with the mode in the settings file set as Multirotor. Click the run button in the Unreal Engine. Execute the python script called StartUp | Pass |  |
| F3/FTC3 | The simulation will end. | Pass |  |
| F4/FTC4 | The simulation will run the formation | Pass |  |
| F5/FTC5 | Run the python test script that puts a non int or float data type as the parameter for adding drones. An error message,can only input a number, shall appear. | Pass |  |
| F6/FTC6 | Run the python test script that puts a non int or float data type as the parameter for removing drones. An error message, can only input a number, shall appear. | Pass |  |
| F7/FTC7 | Run the python test script that removes a drone, but there are only 1 drones in the settings file. An error, must have at least one drone message, shall appear. | Pass |  |
| F8/FTC8 | Run the python test script that adds more than 10 drones. An error, cannot have more than 10 drones, message shall appear. | Pass |  |
| F9/FTC9 | Run the python test script that puts a non int or float data type as the parameter for drone distribution size. An error message, can only input a number, shall appear. | Pass |  |
| F10/FTC10 | Run the python test script that removes a drone at an index larger than the amount of drones in the settings files. An error, out of bounds, message shall appear | Pass |  |
| HF1/HFTC1 | The user clicks on the Unreal start button. The user clicks the run button on the python script called StartUp. | Pass |  |
| HF2/HFTC2 | The user has followed HF1 procedure, the user shall visually see that the swarm has started | Pass |  |
| HF3/HFTC3 | The user clicks pause and the swarm will halt within the Unreal World | Pass |  |
| HF4/HFTC4 | The user clicks stop button and the swarm will disappear from the Unreal World | Pass |  |
| HF5/HFTC5 | The user shall be notified which drone(s) have failed during the simulation. | Fail | Next Iteration |
| HF6/HFTC6 | The user shall be notified which drone(s) have successfully finished the simulation. | Fail | Next Iteration |
| HF7/HFTC7 | The user shall be able to see the perspective of the selected drone flying one at a time. | Fail | Next Iteration |
| HF8/HFTC8 | User types in the terminal to add a distribution size resulting a modification of the settings files | Pass |  |
| HF9/HFTC9 | User types in the terminal to add the drone with its position | Pass |  |
| HF10/HFTC10 | User types in the terminal to remove the drone based on its index | Pass |  |
| HF11/HFTC11 | User types in the terminal to display the drones | Pass |  |

## 5.2 Defect Severity Definitions

|  |  |
| --- | --- |
| **Critical** | The defect causes a catastrophic or severe error that results in major problems and the functionality rendered is unavailable to the user. A manual procedure cannot be either implemented or a high effort is required to remedy the defect. Examples of a critical defect are as follows:   * Image recognition failure * Swarm operation failure |
| **Medium** | A defect that does not seriously impair system function can be categorized as a medium Defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of a medium defect are as follows:   * User interface for swarm configuration behaves unexpectedly * Drone position log inaccuracy |
| **Low** | The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of a low defect are as follows:   * Improper naming in configuration user interface parameters * Documentation does not match intended functionality |

# 

# 6. Environment

* The System Testing Environment shall be used for System Testing.
* The System Testing Environment shall meet all minimum system/resource requirements.
* The System Testing Environment shall be observable by all System Testing Team Members.
* All development team members shall be a member of the System Testing Team for requirements they have participated in.

# 7. Assumptions

* The testing of the System assumes that all system requirements are satisfied.
* The testing of the System assumes that the Unreal Engine and AirSim environments are correctly configured.

# 8. Risks and Contingencies

There are no risks of physical damage to the System or System Testing Environment equipment. There is a risk of system modules failing to meet testing criteria. The contingency for this is to redirect development resources to the most prioritized action items as defined by the Testing Manager and Scrum Master.