## 

**OoeyGUI** AeroGotchi

**Software\_Test\_Plan**

Prepared By:

Inderjit Singh

Keith Chua

Peter Hernandez

Jarrett McIntire

Zamir Barbosa

Elijah Lockett

**Section: Comp 490/L**

**Date: (05/01/2024)**

## 

## TABLE OF CONTENTS

**1. INTRODUCTION 1**

**2. OBJECTIVES AND TASKS 2**

**2.1 Objectives 2**

**2.2 Tasks 2**

**3. SCOPE 2**

**General 2**

**4. TESTING STRATEGY 3**

**4.1 Unit Testing Definition: 3**

**4.2 System and Integration Testing Definition: 4**

**4.3 Performance and Stress Testing 5**

**5. ENVIRONMENT REQUIREMENTS (TOOLS) 6**

**6. TEST SCHEDULE 6**

**7. CONTROL PROCEDURES 6**

**Problem Reporting 6**

**Change Requests 6**

**8. RESOURCES/ROLES & RESPONSIBILITIES 7**

**9. RISKS/ASSUMPTIONS 7**

## 

## 1\_INTRODUCTION

Aero Gotchi is an autonomous drone companion mobile application, designed to provide users with a virtual companion experience with an autonomous drone. This test plan outlines the high-level functions of Aero Gotchi and the testing approach to ensure its functionality and usability.   
  
The main functions include, Mobile Application Interface, Backend Integration, Autonomous Drone Tech, Virtual Companion Interaction.

The test objectives for Aero Gotchi will be based on the functional validation of its virtual companion interaction, drone navigation, and mobile application interface.

## 

## 

## 2\_OBJECTIVES\_AND\_TASKS

## 

## 2.1 Objectives

The Software Test Plan for the Aerogotchi application outlines testing objectives, responsibilities, and communication channels among team members. It serves as a service level agreement, defining quality standards and acceptance criteria. Additionally, it addresses risk management, resource planning, and test coverage, ensuring comprehensive testing of the Flutter app's features and functionalities.

The Test Plan specifies test execution procedures, reporting formats, and compliance with quality assurance standards to deliver a high-quality software product.

## 

## 

## 2.2 Tasks

The tasks of this Test Plan include:

* Provide the scope of the testing plan and explain what is being tested and how.
* Provide hardware requirements for our product.
* Provide environment requirements for our product.
* Provide the tools used for the testing process.
* Provide roles and responsibilities of each team member.
* Provide risks and assumptions of the product.

# 

## 

## 3 SCOPE

## 

## 3.1 General

In this section, we will be explaining the features being tested and their integration. We will verify all our functions are working as expected based on what they are returning.

* Mobile Application
  + Testing Widgets
  + Testing UI
  + Testing API & Backend
  + Testing in Emulators & Devices
* Python Function Testing
  + Verify all functions are running without error
  + Verify correct response is returned
  + Verify that code includes checks to ensure the drone will perform the function. Ex. Flipping the drone will need 50% battery so we need to make sure we have checks on our functions

## 

## 

## 

## 4 TESTING STRATEGY

## 4.1 TESTING STRATEGY

## Unit Testing Definition:

Given the nature of our development process and the structure of our project, we have adopted a tailored testing approach that best fits our specific needs. Since we’re doing a mobile application and Python functions/programs in order to give our drone “life,” our testing will mostly be done by running our programs individually and then trying to link them together to run the features continuously. The mobile application side of the code will be tested with integration testing using IntelliJ built in testing method. With the Python side of things, we will just be conducting function testing. We will be running each function to verify that the function runs as expected and and without error.

Testing the front end required us to do the general approach of testing individual components specifically. We needed to confirm the API’s connected to our database worked with ensuring the get request and confirming username and passwords worked. We did come into issues with posting requests of the database when testing on the IDE but it was a different case when it was tested on the emulator and actual device. We tested each screen one by one per their use case.

## Participants:

* Keith Chua
* Inderjit Singh
* Peter Hernandez

## Methodology:

We conducted unit testing for each individual UI screen that we have created for the front-end portion of our application. We verified the functionality of the screens and how it would respond as if we were using the application. It worked as intended and routed to the specific pages that we set it to be. We created some scripts specifically for each screen. The testing activity was done built in intellij testing

## 

## 

## 

## 

## 

## 

## System and Integration Testing Definition:

Testing folders are marked as test sources to allow the IDE to differentiate what needs to be tested. Running the test in the IDE provides results where testing has passed or failed.

**Frontend/Backend Flutter**

* + Widgets / UI
  + API & Backend
  + Emulator

**Python Functions**

* + Face Tracking Function
  + Keyboard Control Function
  + Sky Shuffle Function
  + Photo Pilot Function

List what is your understanding of System and Integration Testing for your project.

System and integration testing for our project will be running various parts of our functions in the mobile application as well as python functions in order to ensure everything is running smoothly and properly.

## Participants:

* Keith Chua
* Inderjit Singh
* Peter Hernandez

## Methodology:

Describe how System & Integration testing will be conducted. Who will write the test scripts for the unit testing, what would be the sequence of events of System & Integration Testing, and how will the testing activity take place?

We conducted integration testing for our application. We tested the application as a whole to see if it works properly and as intended. For the UI/UX side of testing, we created scripts for the IDE to do the testing based on what we required. The scripts would run in accordance to how the application is set up. The testing verifies the get requests and usability of the application as a whole. For the API side of the application, the test cases we provided were data from our database to ensure workability. The get aspect of data worked as intended but posting was an issue.

## Performance and Stress Testing:

**N/A**

## Definition:

**N/A**

## Participants:

**N/A**

## Methodology: **N/A**

## 5. ENVIRONMENT REQUIREMENTS (TOOLS)

# 5.1 ENVIRONMENT REQUIREMENTS (TOOLS)

We used the **IntelliJ IDE** for integration testing

The testing for the drone with aspects of flutter code and firebase required a bluetooth device and an ethernet connection to be able to connect to 2 separate networks for testing, **Postman, PyTest.**

The testing environment is a system running the Windows operating system. The testing system must have Flutter and Python installed as well as Visual Studio Code.

## 

## 6. TEST SCHEDULE

**N/A**

## 

## 

## 

## 7. CONTROL PROCEDURES

# CONTROL PROCEDURES

## Problem Reporting

Any errors that occur during the testing process will be reported and logged during the runtime.

## Change Requests Lead Inderjit Singh is responsible for all sign offs on the changes while Keith Chua, Zamir Barbosa, and Peter Hernandez will be responsible for requesting changes to the current product.

## 8. RESOURCES/ROLES & RESPONSIBILITIES

# 

# RESOURCES/ROLES & RESPONSIBILITIES

Specify the staff members who are involved in the test project and what their roles are going to be (for example, Mary Brown (User) compiles Test Cases for Acceptance Testing). Identify groups responsible for managing, designing, preparing, executing, and resolving the test activities as well as related issues. Also, identify groups responsible for providing the test environment. These groups may include developers, testers, operations staff, testing services, etc.

Inderjit Singh (Tester) and Peter Hernandez (Tester): In charge of testing the python functions to make sure they are working.

Keith Chua (Tester) : Will be in charge of conducting unit/integration testing for the flutter side of the code

## 

## 

## 

## 

## 9. RISKS/ASSUMPTIONS

# RISKS/ASSUMPTIONS

**Risk**: Delay in the delivery of test items

**Assumption**: Test items, including drone prototypes and application versions will be delivered on schedule.

**Risk**: Inadequate test coverage due to evolving requirements

**Assumption**: The initial test coverage plan is sufficient throughout the project lifecycle.

**Risk**: Hardware limitations affecting test execution

**Assumption**: The specified hardware is adequate to support all testing scenarios.

**Risk**: Software and hardware integration issues

**Assumption**: Software components will seamlessly integrate with hardware without major issues.

**Risk**: Resource availability (human and technical)

**Assumption**: All project resources, including team members and technical tools, are available as planned.

**Risk**: Technological advancements rendering current testing tools obsolete

**Assumption**: Current testing tools will remain adequate throughout the project duration.

**Risk**: Regulatory changes affecting drone operations

**Assumptio**n: Current regulations will remain stable.

**Contingency Plan**: Increase the allocation of resources such as additional weekend work to accelerate testing phases.   
**Contingency Plan**: Implement a flexible testing framework that allows easy integration of additional tests. Regularly review and adjust the test coverage to accommodate new or changed requirements.

**Contingency Pla**n: Plan for scalable hardware solutions and have additional or alternative hardware resources ready for deployment. Conduct early hardware stress tests to identify potential bottlenecks.

**Contingency Plan**: Establish a dedicated integration testing phase early in the cycle to detect integration issues sooner. Set up a rapid response team to address integration failures.

**Contingency Plan**: Develop a resource management plan that includes backup resources and cross-training team members to handle critical tasks. Engage with contractors or temporary hires if necessary.

**Contingency Plan**: Regularly review and potentially upgrade testing tools to ensure compatibility with the latest technological advancements. Maintain flexibility in the budget for tool upgrades.

**Contingency Plan**: Monitor regulatory environments closely and have a legal team ready to adapt the test plans and product design to meet new regulatory requirements.