**OoeyGUI** AeroGotchi

**Software Design Document**

By:

Inderjit Singh

Keith Chua

Peter Hernandez

Jarrett McIntire

Zamir Barbosa

Elijah Lockett  
 **Lab Section: Comp 490/L**

**Workstation:**

**Date: (12/13/2023)**

# 

# **TABLE OF CONTENTS**

**1 INTRODUCTION 3**

1.1 Purpose **3**

1.2 Scope **3**

1.3 Overview **3**

1.4 Definitions and Acronyms **3**

**2 SYSTEM OVERVIEW 3**

**3 SYSTEM ARCHITECTURE [4](#_heading=h.1t3h5sf)**

3.1 Architectural Design **4**

3.2 Interface Design **5**

3.3 Decomposition Description **[5](#_heading=h.17dp8vu)**

3.4 Design Rationale **[5](#_heading=h.3rdcrjn)**

**4 COMPONENT DESIGN/DETAILED DESIGN 5**

4.1 Class Diagrams **5**

4.2 Sequence Diagrams **13**

**5 HUMAN INTERFACE DESIGN 16**

5.1 Overview of User Interface **16**

5.2 Screen Images **18**

5.3 Screen Objects and Actions **26**

**6.0 REQUIREMENTS MATRIX 26**

## 1 INTRODUCTION

### 1.1 Purpose

The purpose of this **Software Design Documen**t **(SDD)** is to **describe** and visualize the **design, structure** and **architecture of AeroGotchi**.

*The intended audience is our development team and Professor Edmund Dantes.*

### 1.2 Scope

**Aerogotchi** is an innovative **drone pet mobile application** designed for individuals in search of a *companion* without the associated risks and responsibilities of caring for a real pet.AeroGotchi seeks to offer users a unique human and drone bonding experience.

Beyond its role as a virtual companion, this project also strives to establish a groundwork for autonomous drone technology in a playful and engaging manner, contributing to the exploration of novel applications in the evolving and overgrowing landscape of drone technology.

### 1.3 Overview

This SDD highlights the structure and overall approach of this project.

It covers the AeroGotchi architecture, class structure, sequence diagrams, and our user interface design.

The following will include an example screenshot of the *Graphical User Interface (GUI)*, providing a visual representation of how the user interface is envisioned  
  
Additionally, within this Software Design Document (SDD), there will be a comprehensive explanation of how our project requirements will be fulfilled.

### 1.4 Definitions and Acronyms

***Controller*** *- a physical device that takes in user input that will communicate with the device.*

***OpenCV*** *- open source software for computer vision and machine learning related to Video Analysis. Frameworks available to use Augmented Reality (AR), gesture recognition, image tracking.*

***Flutter*** *- mobile development software that has user interfaces (UI) and user experiences (UX).*

**UI** *- User interface*

**UX** *- User experience*

## 

## 

## 2 SYSTEM OVERVIEW

AeroGotchi is an autonomous pet drone controlled via a mobile application or hardware.

This mobile application was inspired by the *Tamagotchi* style game where we can have digital pets without commitments. The *ability* to *feed*, *walk*, *play*, *clean* and *care* for digital pets provides a sense of companionship, responsibility, and accomplishment for our digital pets.

AeroGotchi’s design offers a unique experience for enthusiasts wanting a pet or companion that will create a bonding experience and entertainment for the users.

The system’s development relies on various programming languages such as Python and Flutter.

## 

## 3 SYSTEM ARCHITECTURE The following **section** describes the **architecture** and **design** for the AeroGotchi application.

### 3.1 Architectural Design

### 

### 

### 3.2 Interface Design

The Three Tier Style architecture is separated by Client Side/Front End, Server Side/Back End, and Database Design.

The Front End will connect to the Back End with a series of HTTP Requests.

The Back End is connected to the Database with a series of SQL Queries for necessary data.

### 3.3 Decomposition Description

Our *Front End* will be written in *Flutter* to develop a mobile app that could support current Android and iOS versions.

The *Back End* would be written in *Python* and use *OpenC*V and the *DJI Tello SDK* for several autonomous features.

Separating sections enables a modular program structure, simplifying development for both Client Side and Server Side.

### 3.4 Design Rationale

Our goal started with creating a Mobile Application for a Drone Pet Game.

As we wanted to include Autonomous Features and use *Image Recognition Software*, we found that the easiest language for those would be *Python*.

Therefore, we would need to have a Client, Server relationship to allow for those features to be added. The complex modular structure in the *Back End* is split up between simpler *Game Mechanics* and Specialized *Drone Controls*.

Personality *holds* the Status values and manipulation *as the User interacts with the Drone*. The Fun, FoodType, and Energy Classes would *hold* the different preset actions based on the *Random* *Personality values* in the Personality Class.

Drone System is the Computer Vision Center that calls for various other classes for extra input. Drone Actions and Drone Visuals are the subsystems that use the Tello SDK to manipulate drone movements and retrieve the Camera Feed from the Drone’s POV.

As for our Database, we wouldn’t need a complex design as the only values needed would be data for User Login and to allow the Pet Mechanics to continue in Real Time. A simple schema with Drone\_ID, Drone\_Name, User\_ID, Password, Food\_Status, Excitement\_Status should suffice.

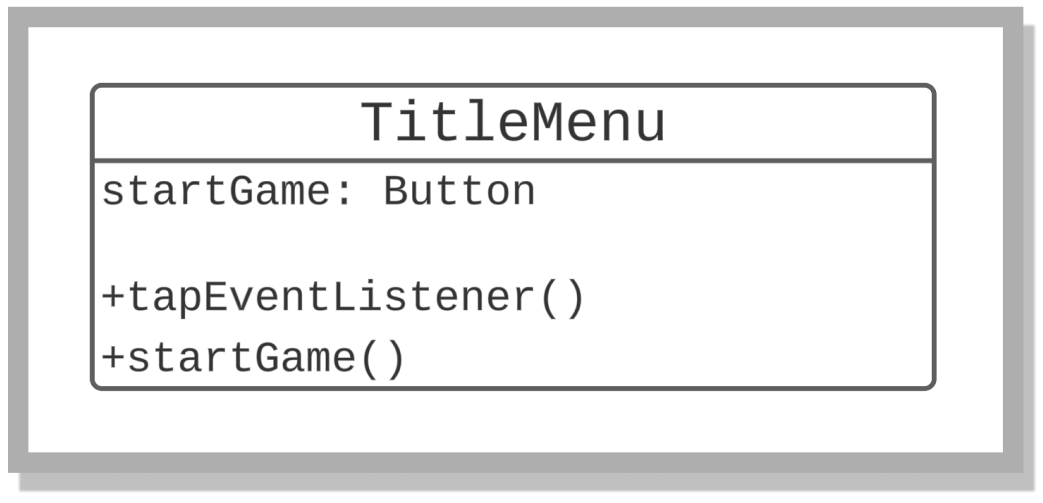
## 

## 4 COMPONENT DESIGN/DETAILED DESIGN

***The following******diagrams*** will depict **associations, attributes**, and **methods** for each **class**.

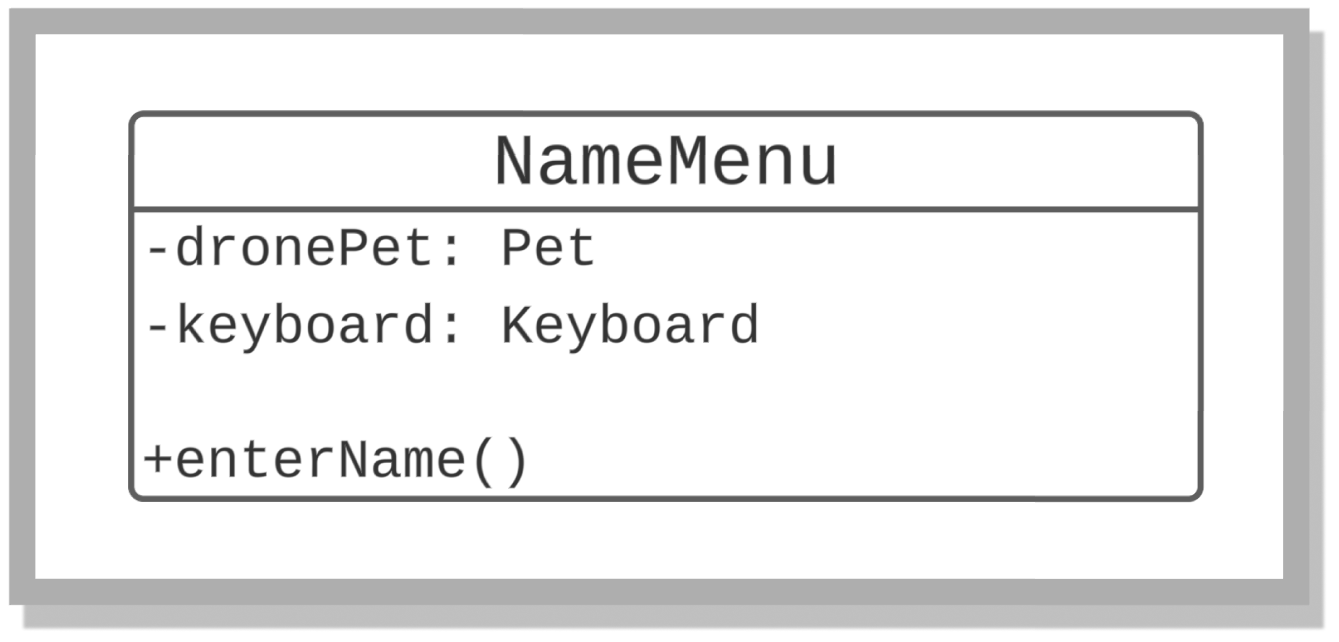
### 4.1 Class Diagrams

**4.1.1 Title Menu**



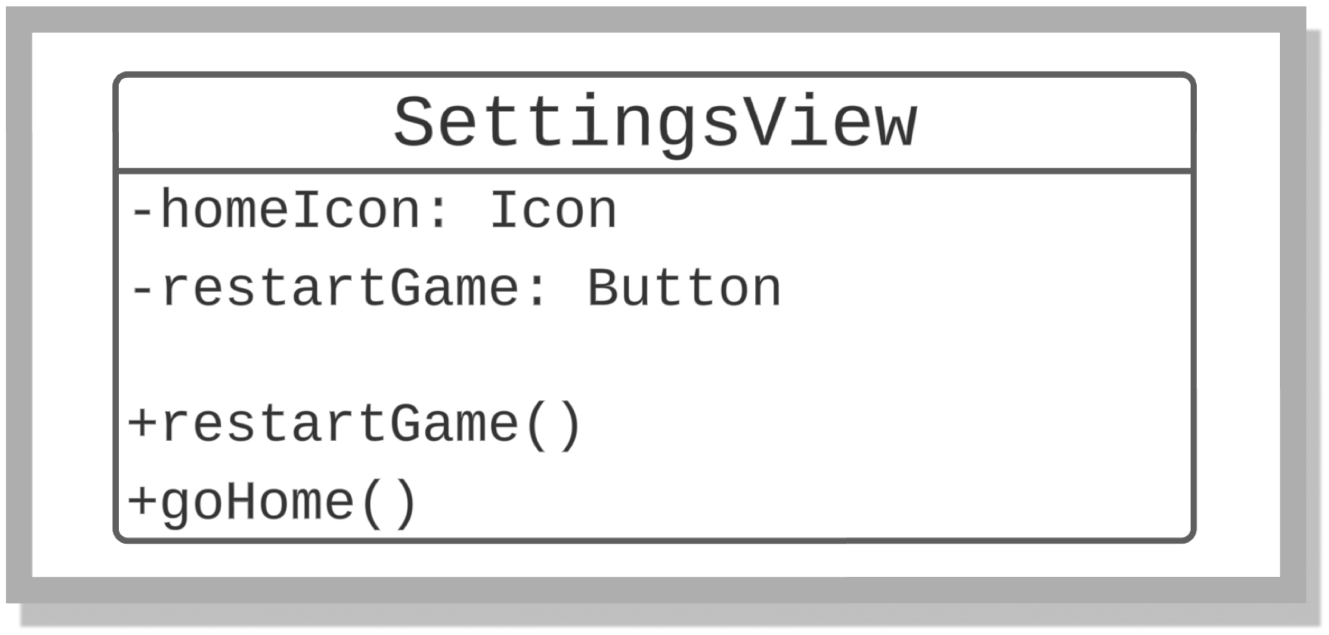
**TitleMenu** class is responsible for showing the title page when the application is initially launched.

**4.1.2 Name Menu**



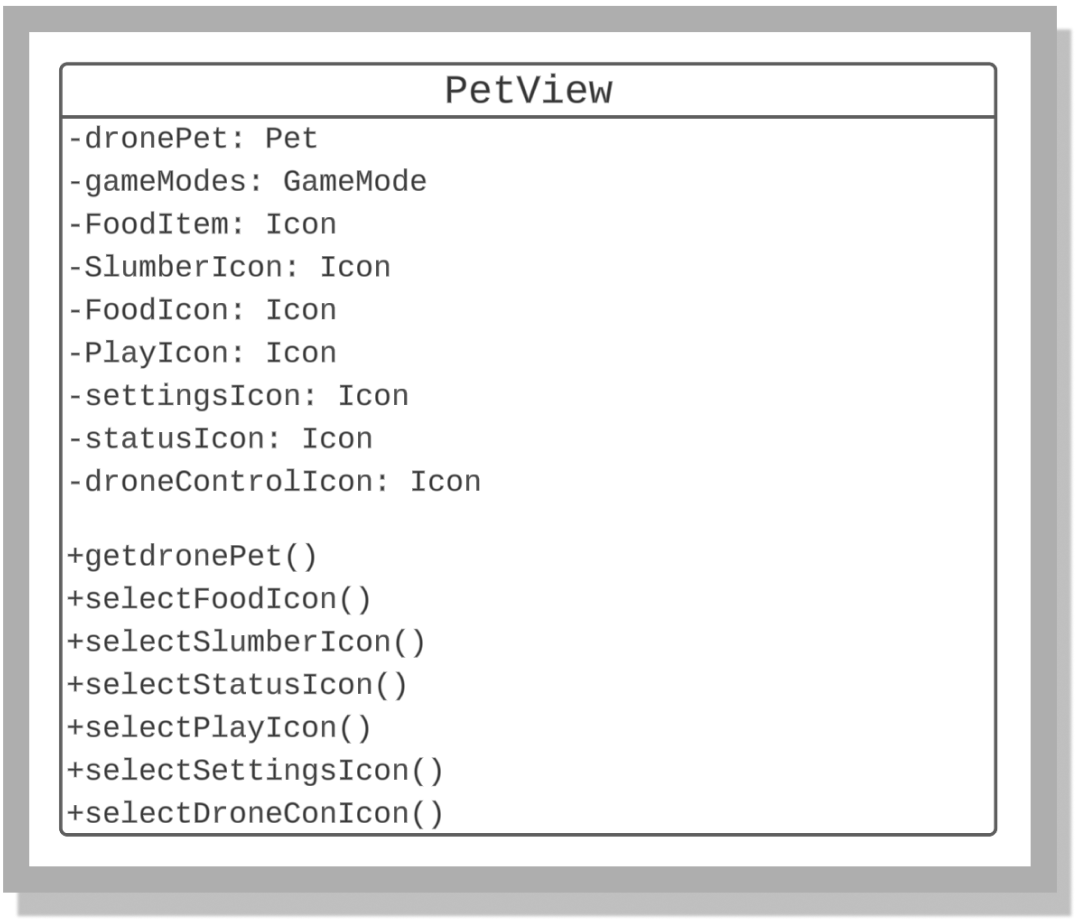
**NameMenu** class allows users to select the name they want for the pet.

**4.1.3 Settings View**



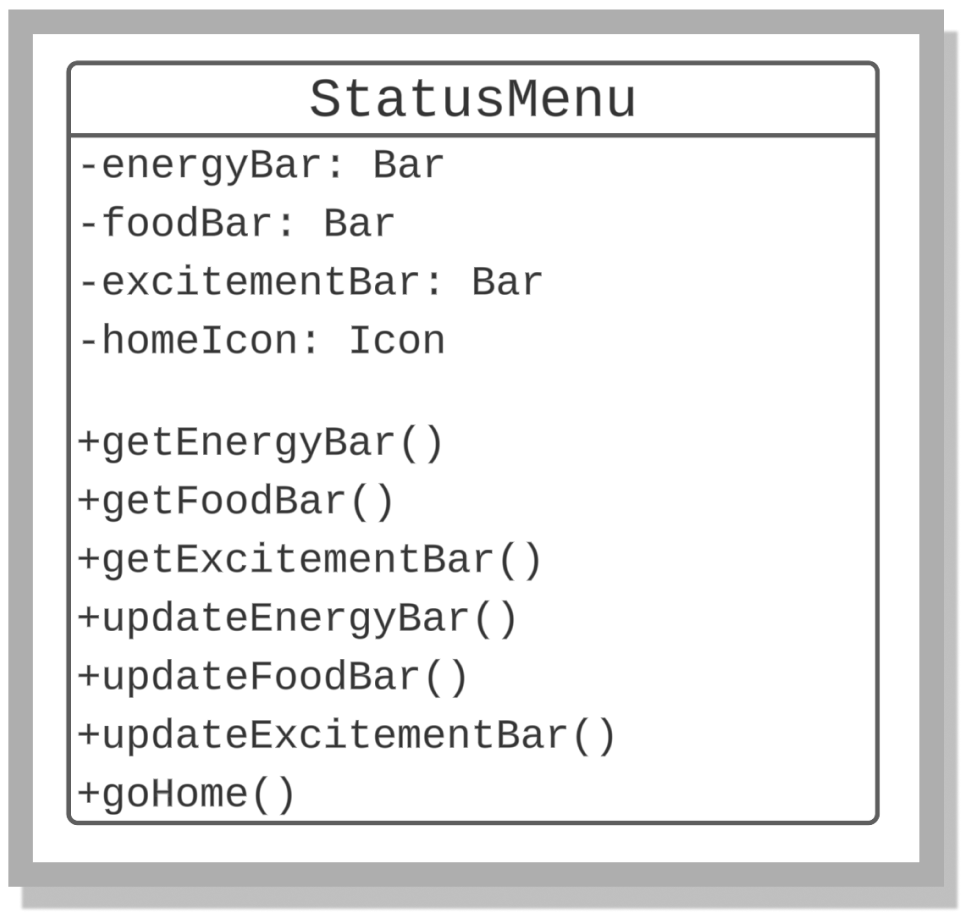
**Settings** view allows users to restart the game.

**4.1.4 Pet View**



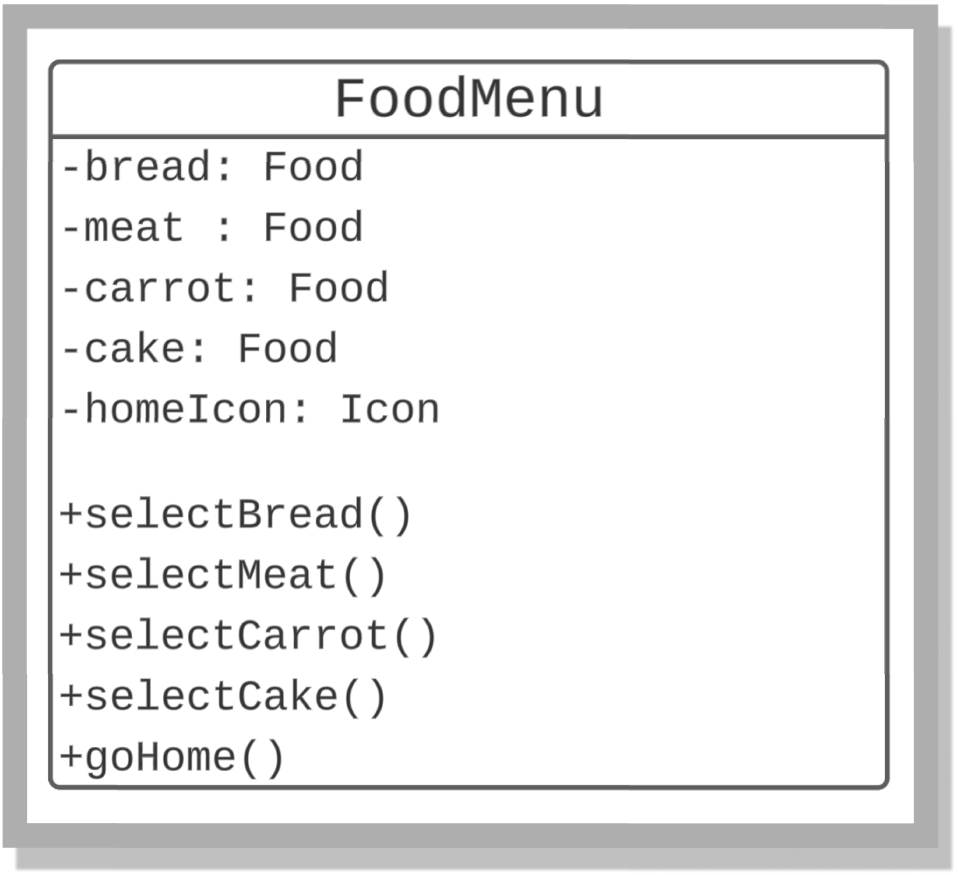
**PetView** class shows the pet within the application which is surrounded by multiple menu icons where users can choose from.

**4.1.5 Status Menu**



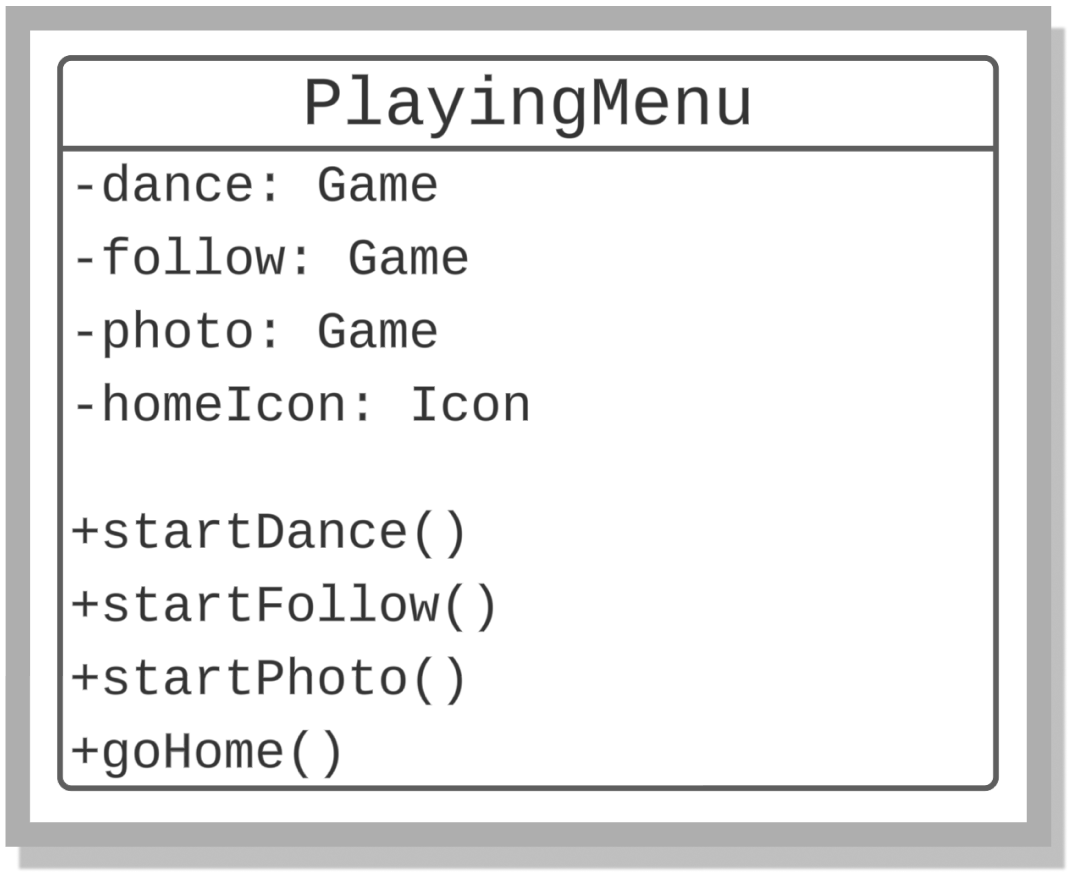
**StatusMenu** class shows the status of the pets Energy, Food, and Excitement levels which is *updated when certain events happen within the application*.

**4.1.6 Food Menu**



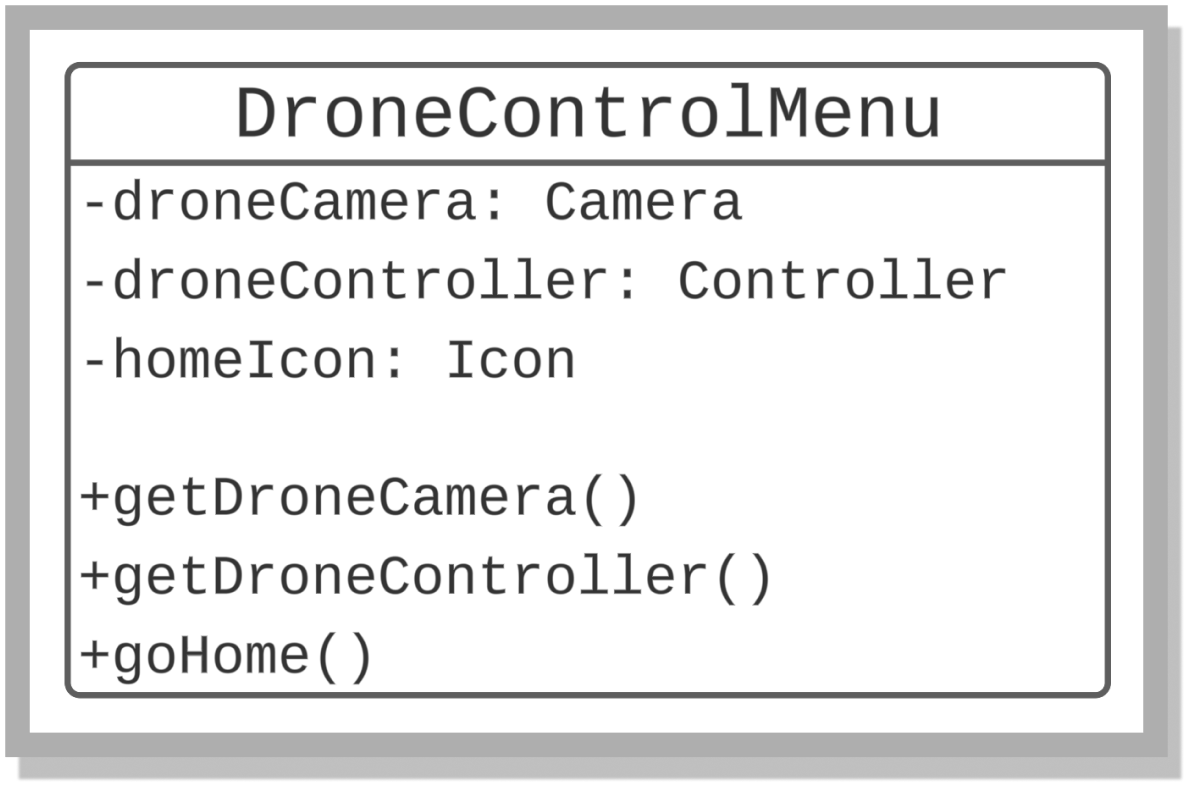
**FoodMenu** class is a class that allows the user to select different types of food to feed their pet.

**4.1.7 Playing Menu**



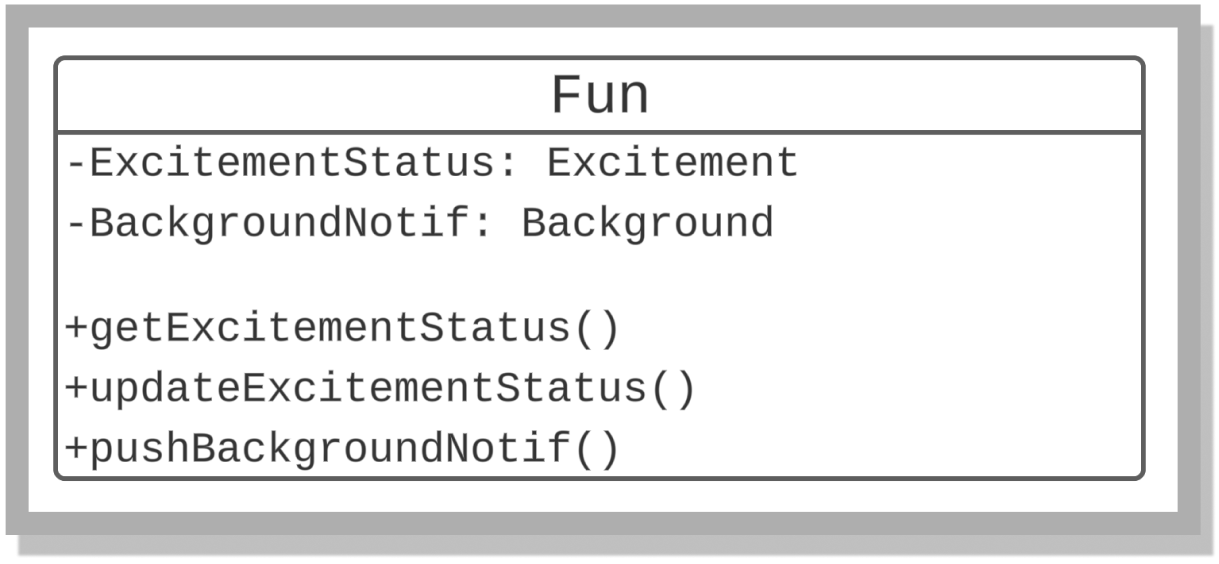
**PlayingMenu** class provides mini games with the drone where it could *dance, follow* the user or do a *photoshoot* upon selection.

**4.1.8 Drone Control Menu**



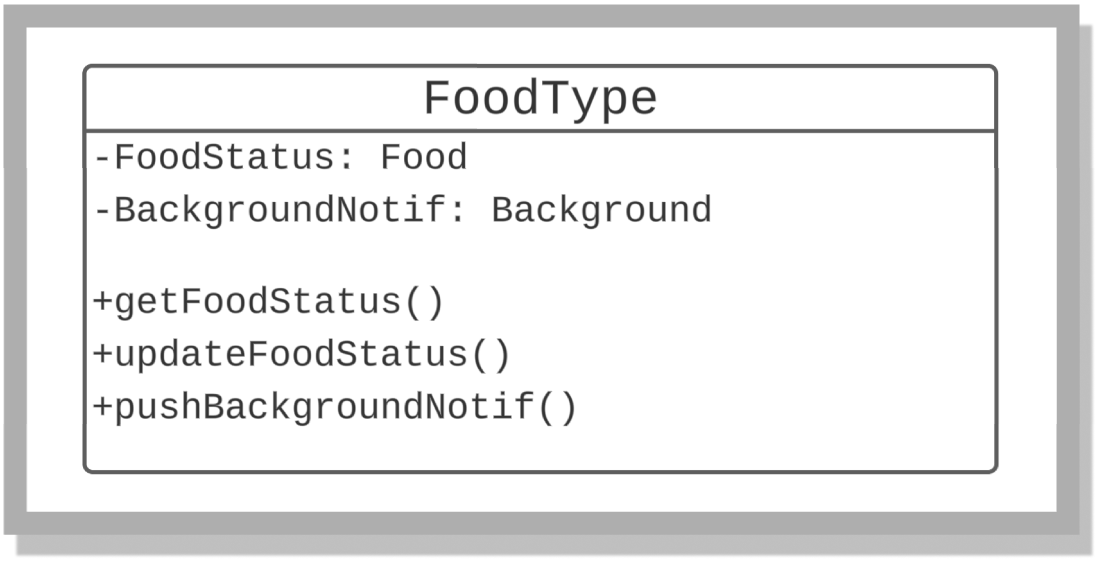
**DroneControlMenu** class that communicates with the drone. It can tell the drone to use its camera, have the user control the drone or to activate an autonomous feature within the drone to return to its home hub (location of its initial takeoff)

**4.1.9 Fun**



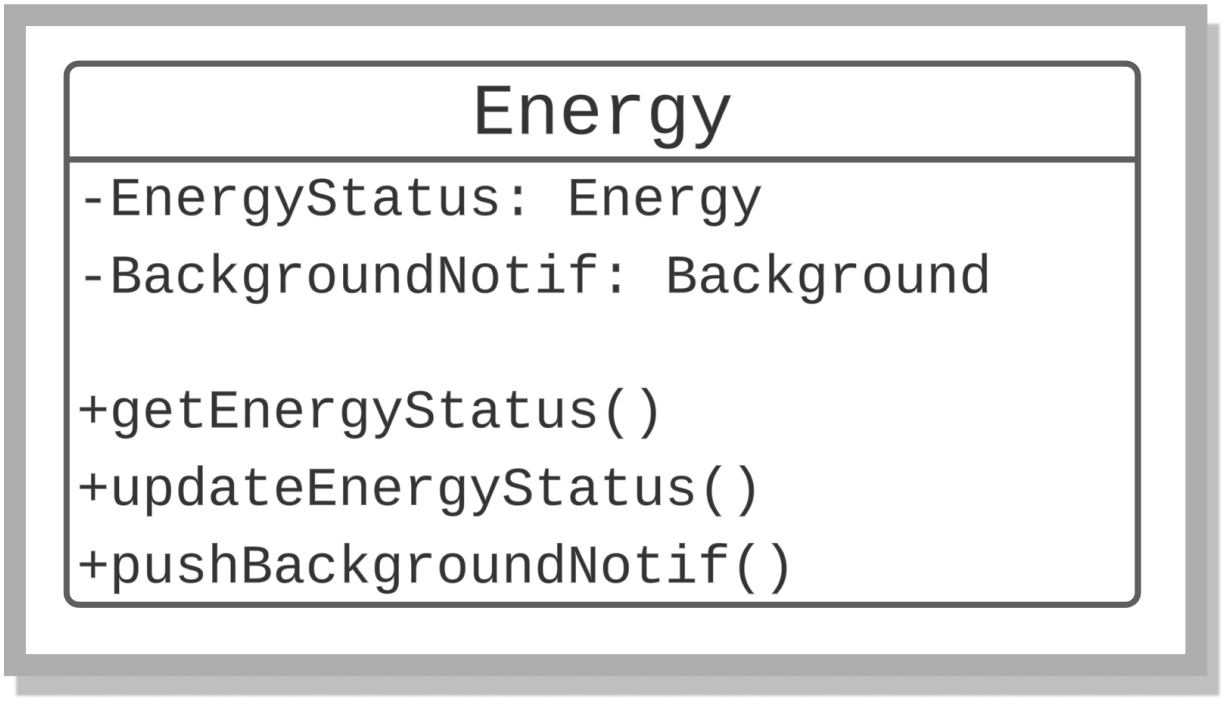
**Fun** class enables the excitement status of the pet in the application.

**4.1.10 Food Type**



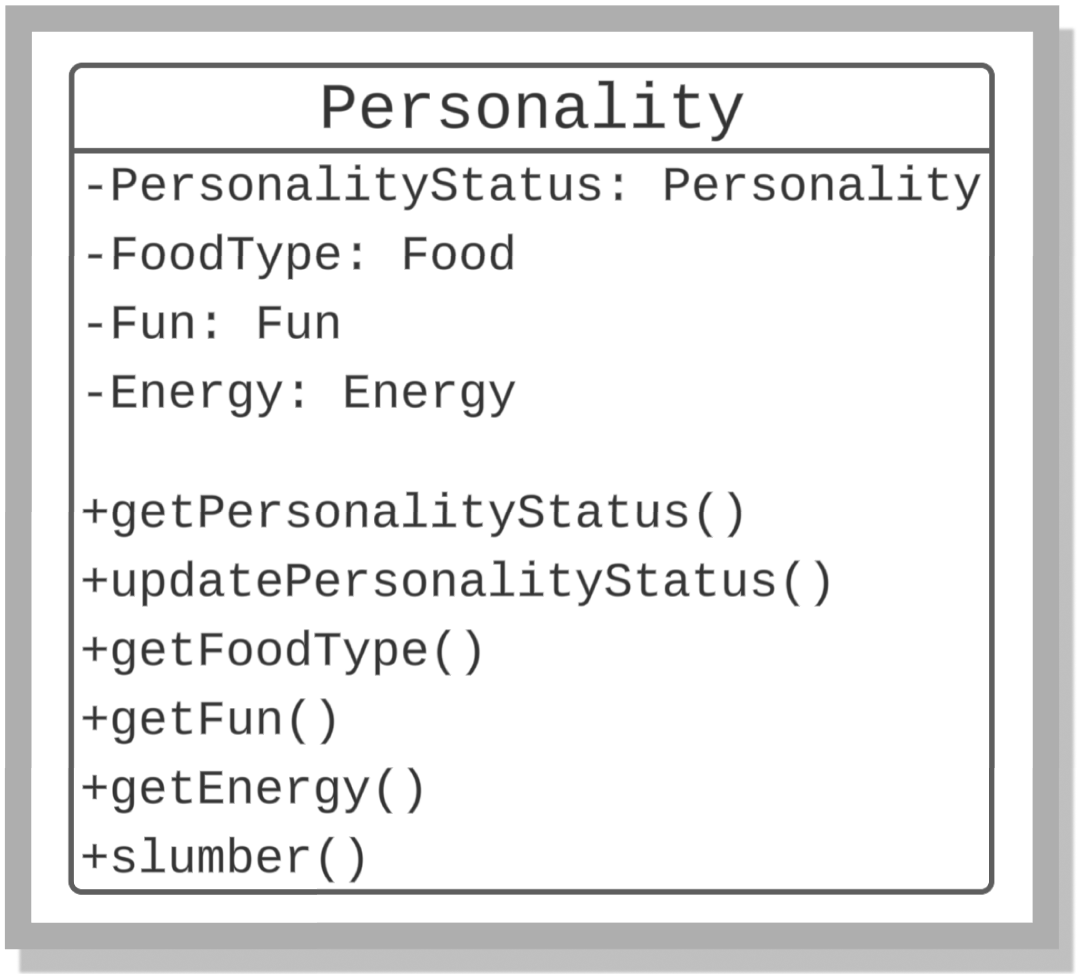
**Food Type** class shows us the hunger status of the pet and u*pdates periodically based on the status of the pet.*

**4.1.11 Energy**



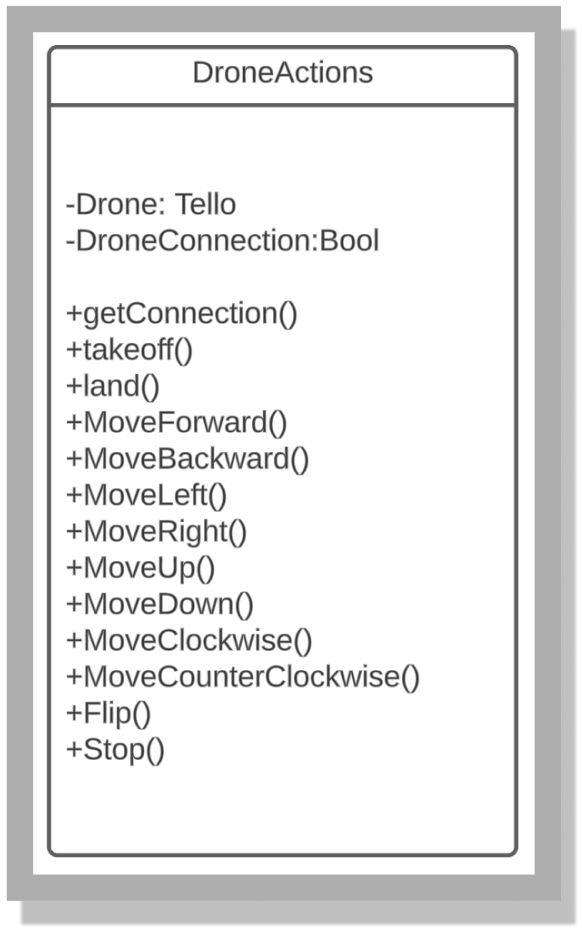
**Energy** Classshows us the energy status of the pet and *updates periodically based on its energy status.*

**4.1.12 Personality**



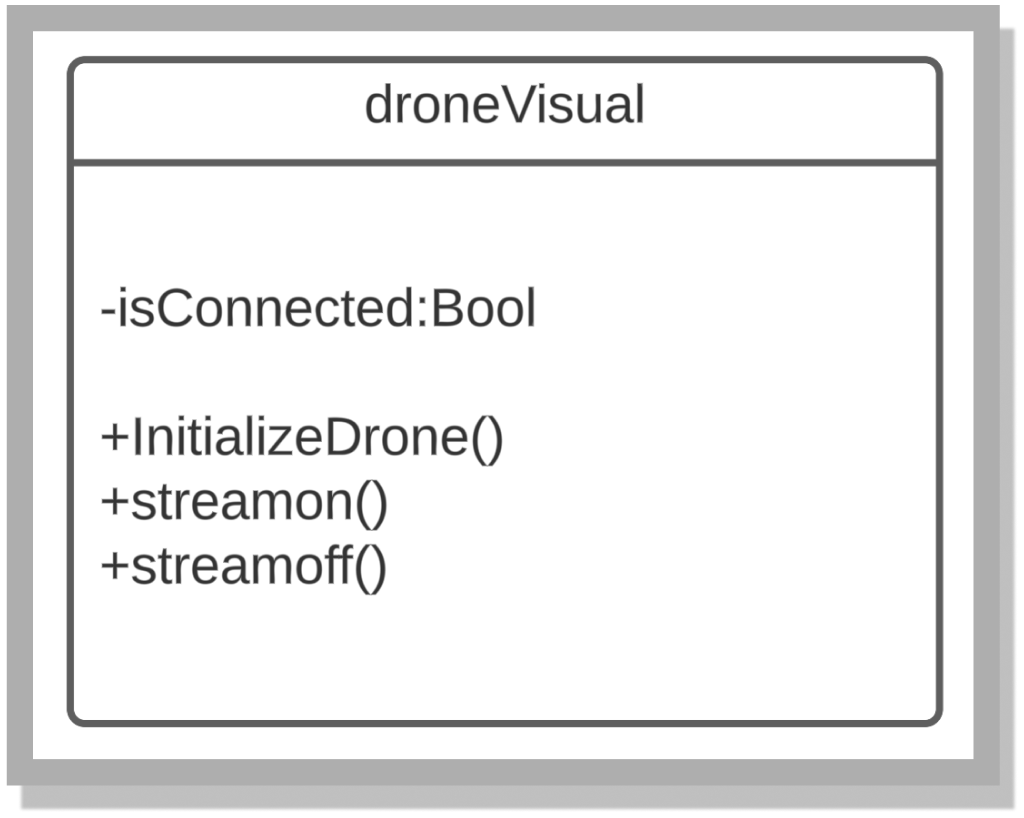
**Personality** class updates the pet personality depending on the Food it is fed, the amount of Fun the pet is having and the amount of energy it has.

**4.1.13 Drone Actions**



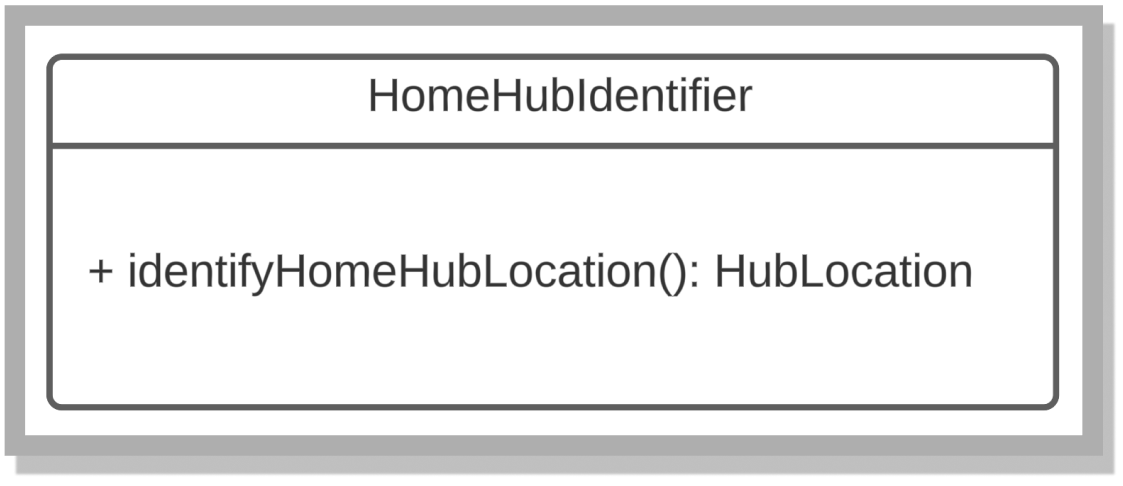
**Drone Action** class is home to the drone controls. This enables us to control the drone however we want and allow it to do certain actions as intended.

**4.1.14 Drone Visual**



**Drone Visual** is a class that turns the drone’s camera on and off where *we could see it in the application.*

**4.1.15 Home Hub Identifier**



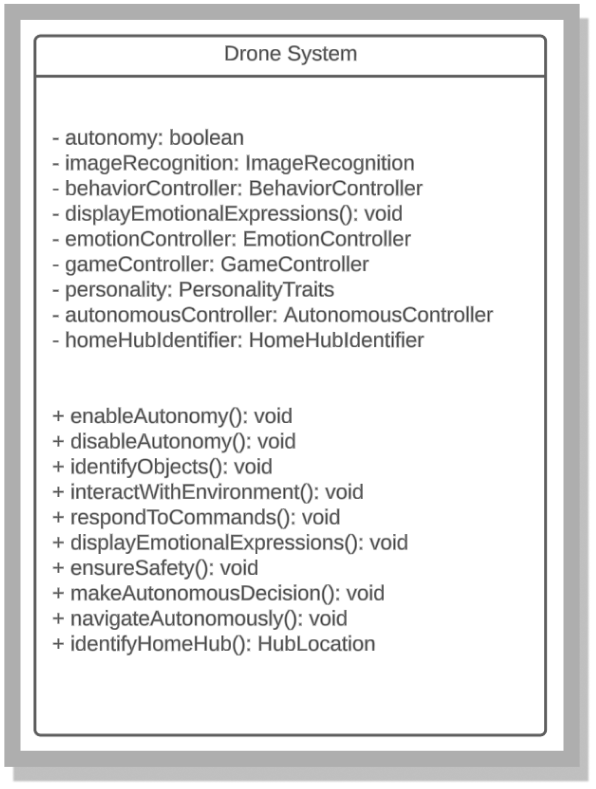
**Home Hub** Identifier saves the coordinates of the drone and considers it as home. This class is connected to Hub Location.

**4.1.16 Hub Location**



**Hub Location** class gathers the coordinates of the drone and sends it to Home Hub Identifier to be saved.

**4.1.17 Drone System**



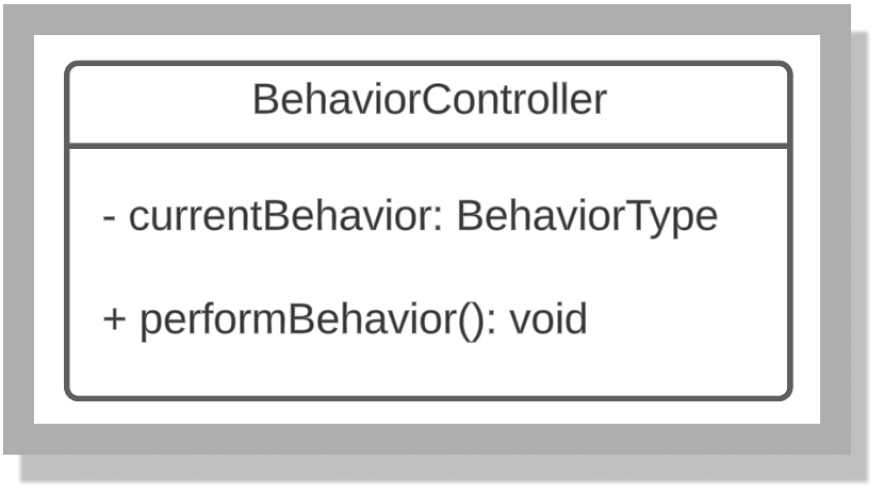
**Drone System** Class encapsulates the methods of controls with the autonomy, emotional expression, gaming, and identification functionalities.

**4.1.18 Emotion Controller**



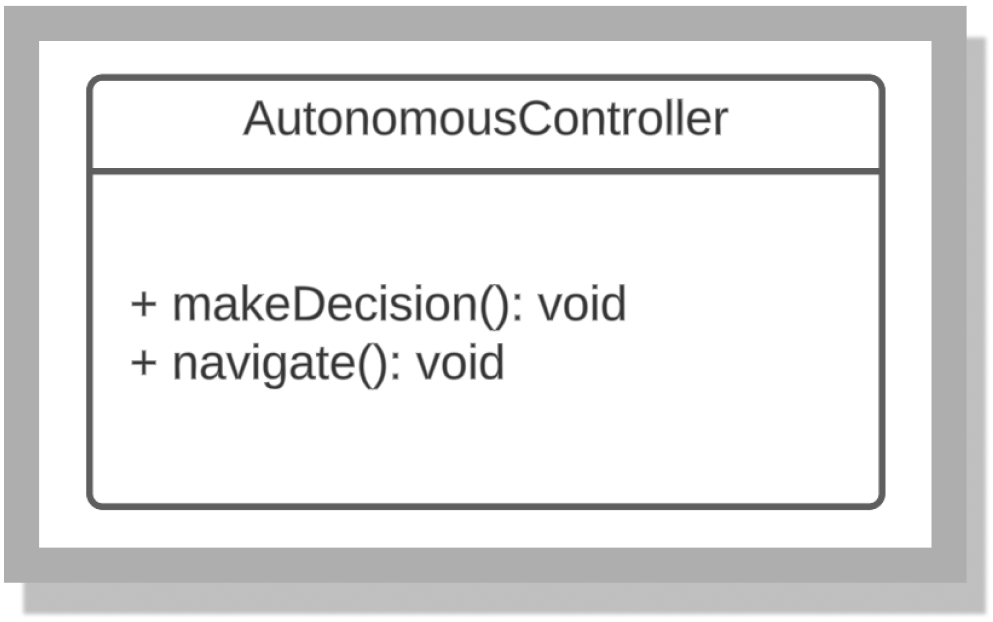
**Emotion Controlle**r class gets the current emotion of the pet and displays it on the drone. Certain emotions allow the drone to act in certain ways.

**4.1.19 Behavior Controller**



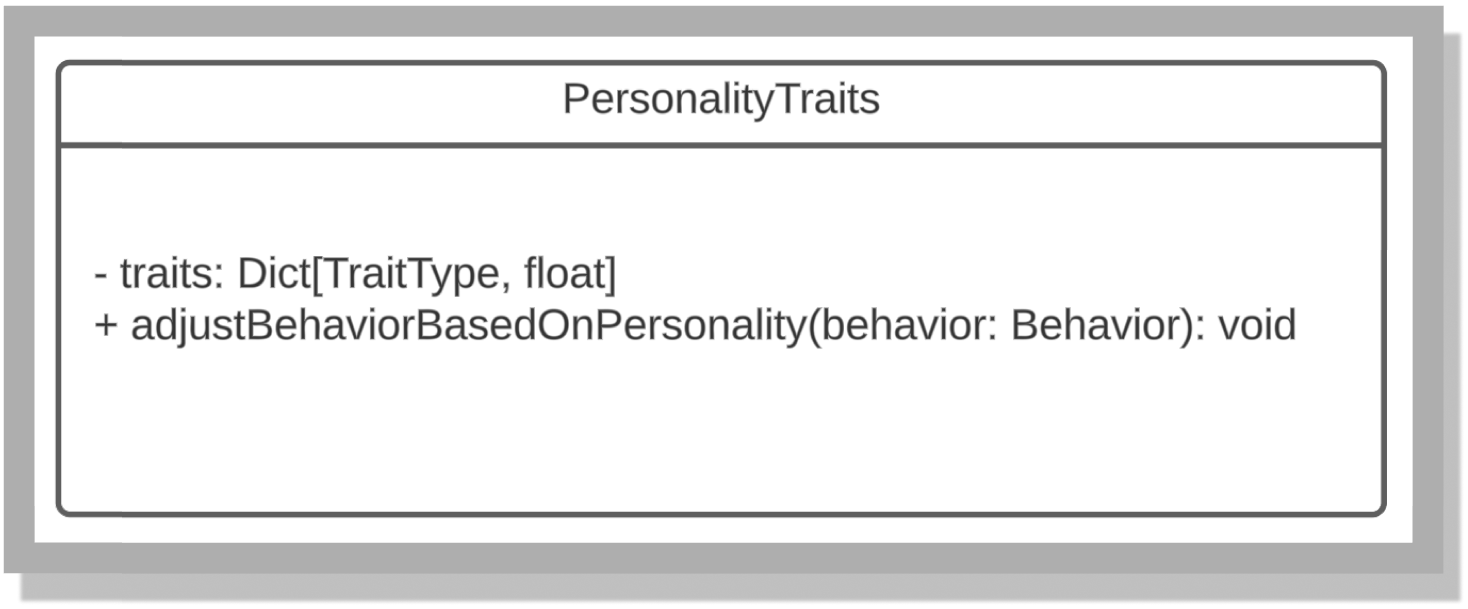
The **Behavior Controller** class controls the behavior of the pet.

**4.1.20 Autonomous Controller**



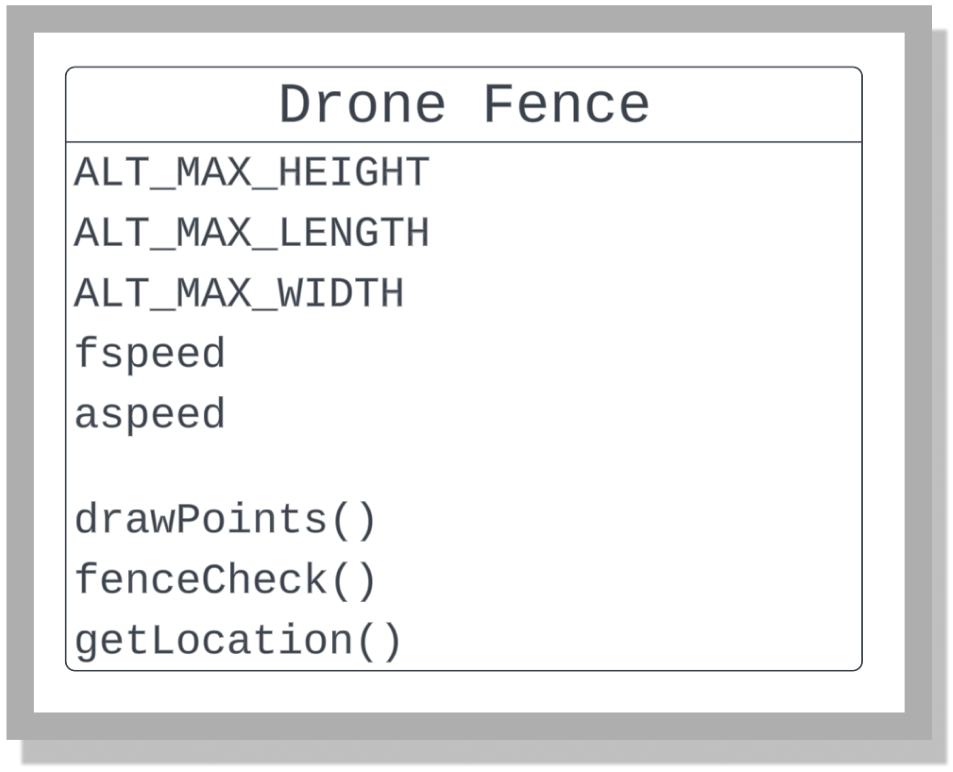
**Autonomous Controller** class allows the drone to navigate and *make decisions* to make it autonomous.

**4.1.21 Personality Traits**



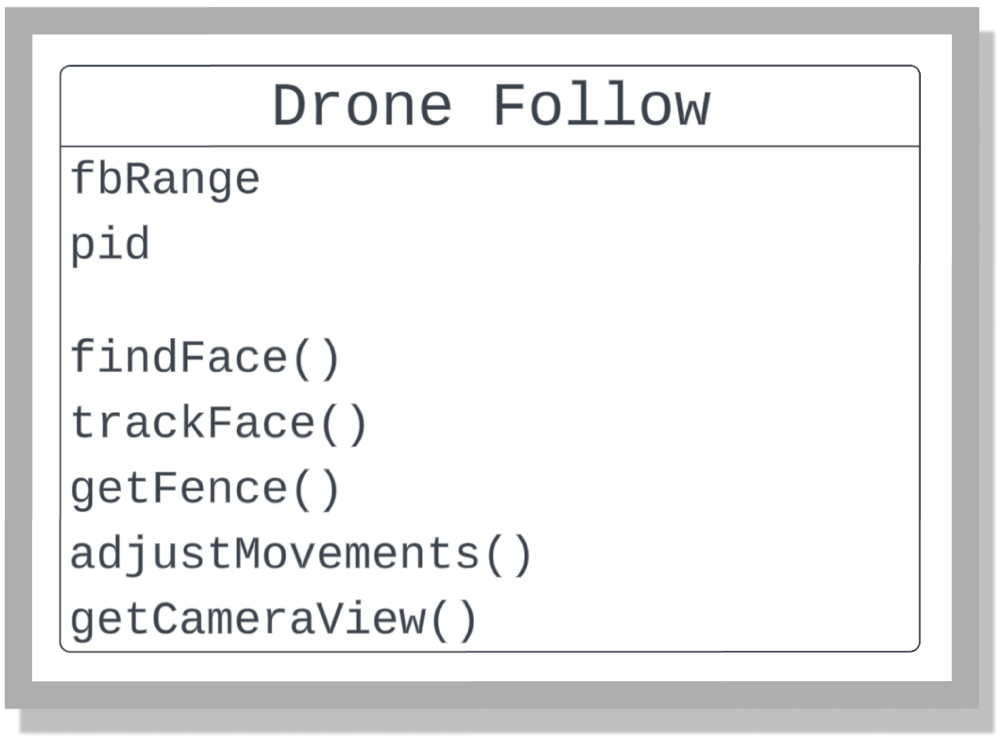
**Personality Traits** class would adjust the personality of the pet d*epending on its behavior.*

**4.1.22 Drone Fence**



**Drone Fence** class would re/draw the barriers that limit the drone’s autonomous distance it can travel.

**4.1.23 Drone Follow**

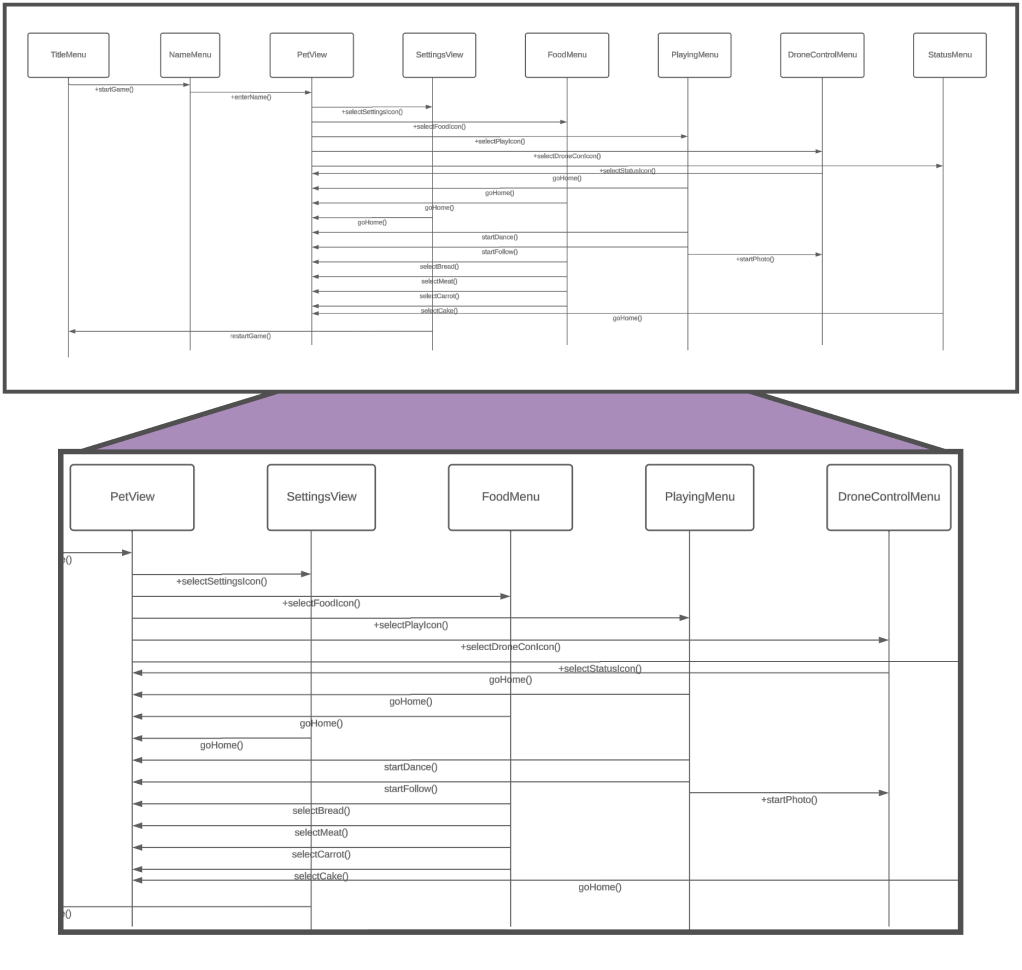


**Drone Follow** class implements Computer Vision with Autonomous Flight to have the AeroGotchi follow its user around until the game ends.

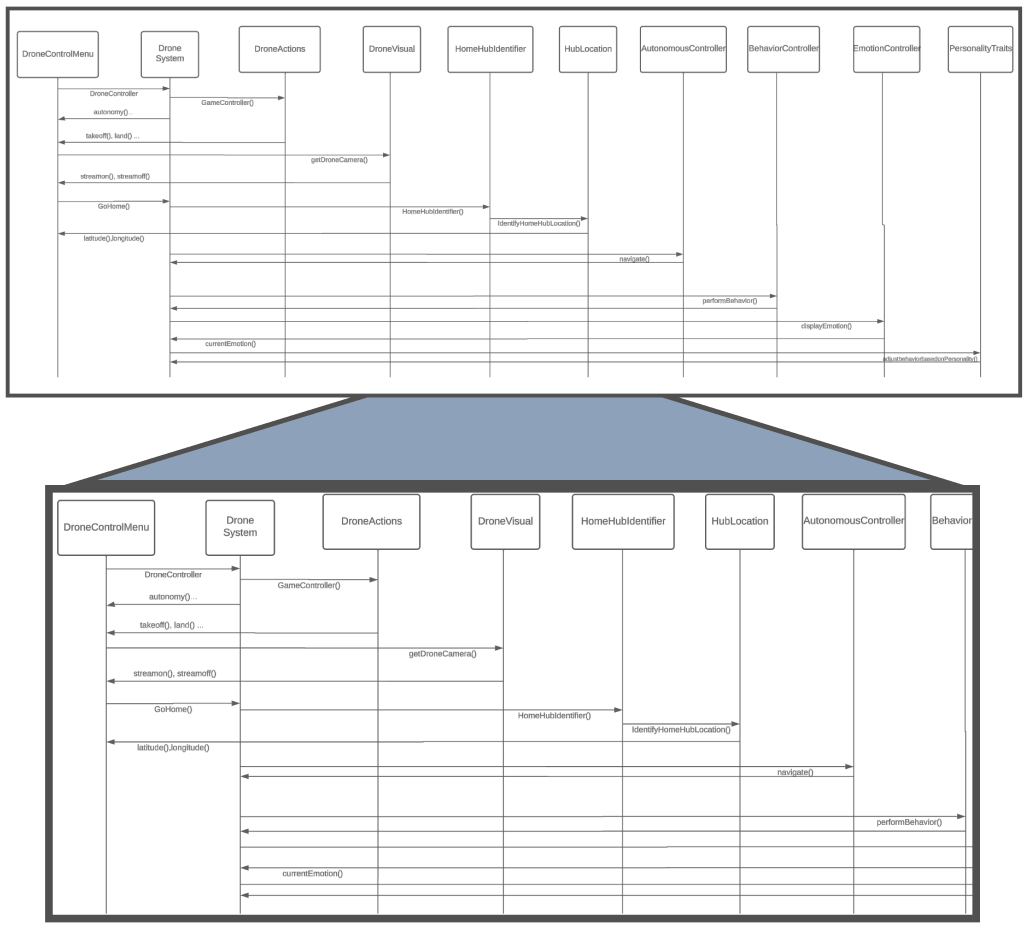
### 

### 4.2 Sequence Diagrams

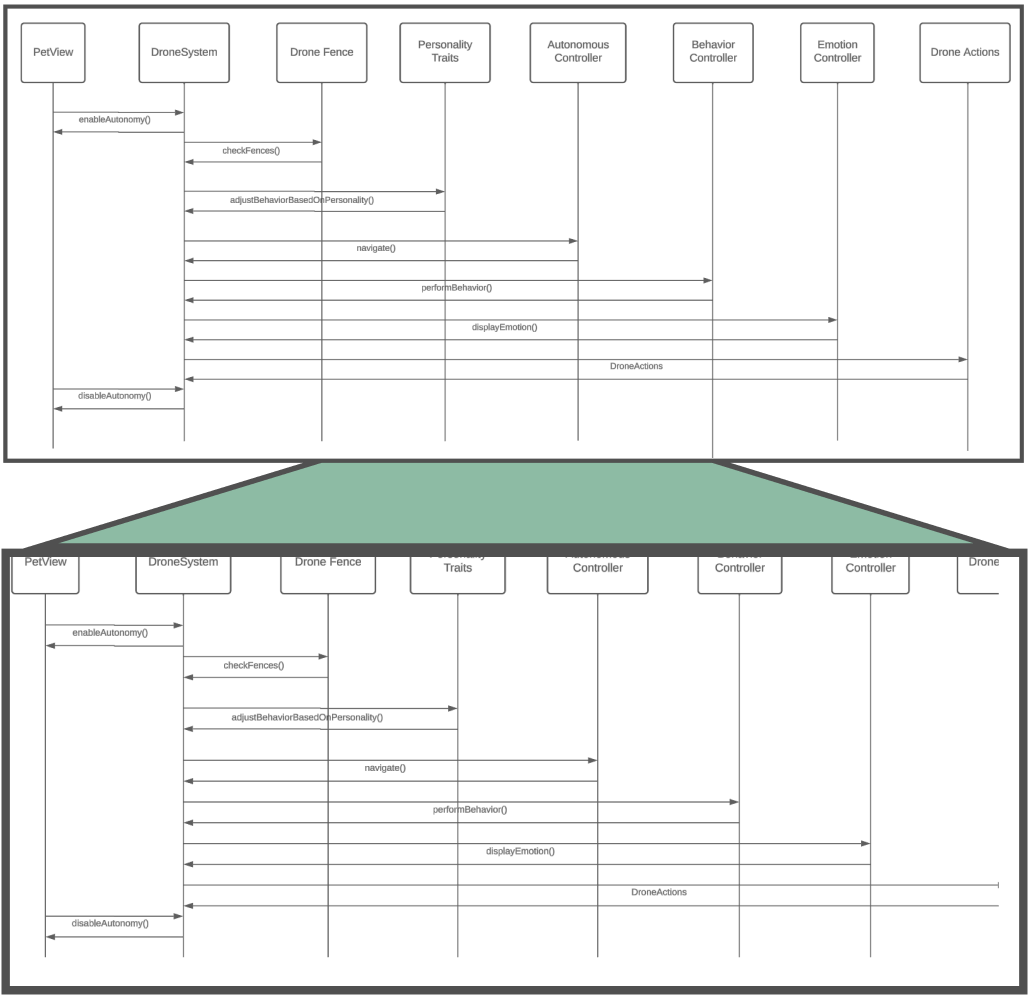
**The following are diagrams of the static and dynamic aspects of the subsystem components.**

**4.2.1 Front End SD**

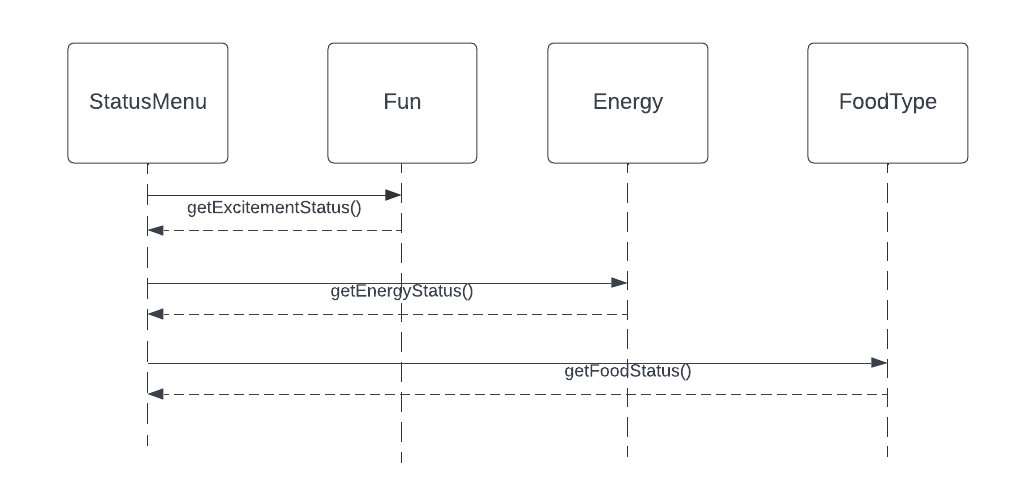
**4.2.2 Drone Control Menu SD**



**4.2.3 Pet View SD**

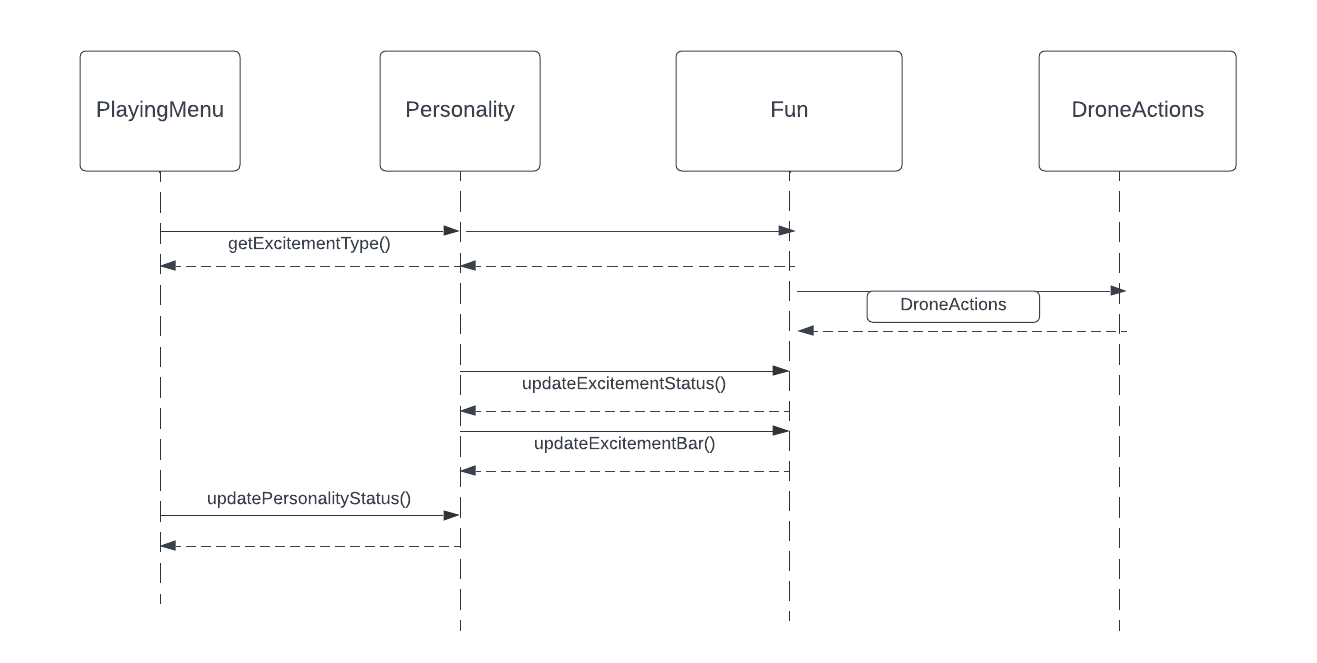


**4.2.4 Status Menu SD**

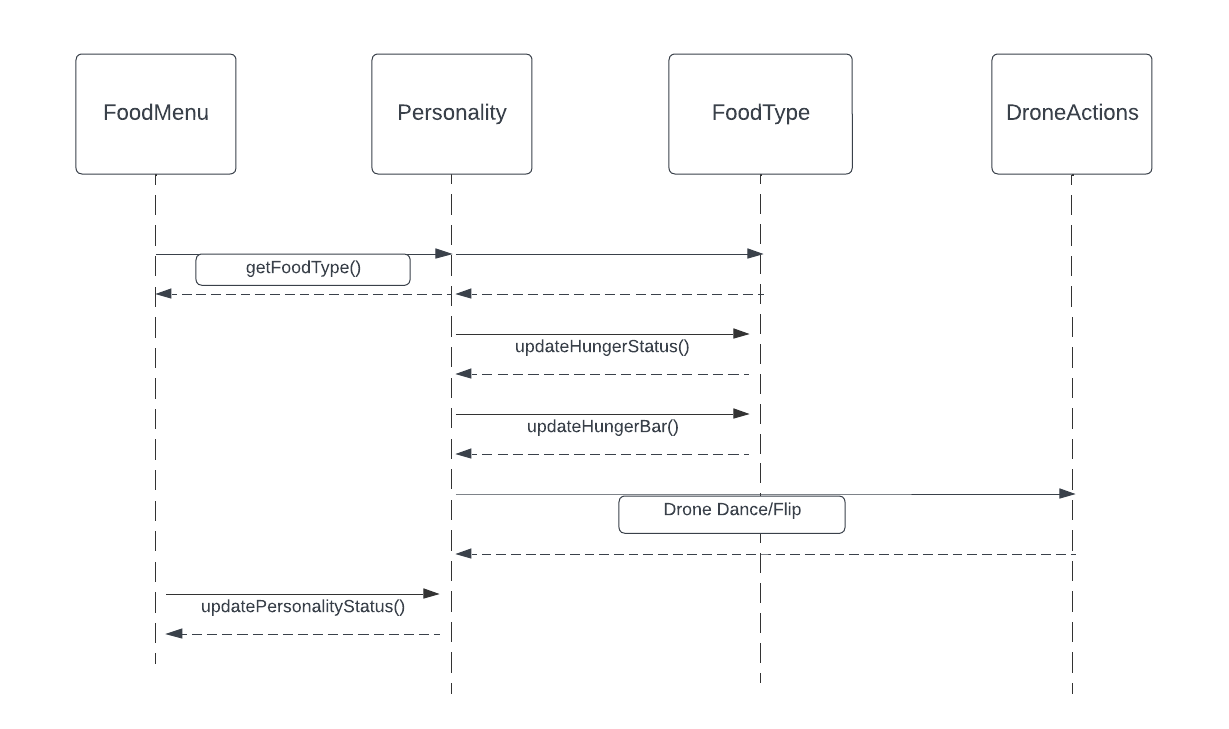


**4.2.5 Playing Menu SD**

**4.2.5 Playing Menu SD**



**4.2.6 Food Menu SD**



## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 5.0 HUMAN INTERFACE DESIGN

### 5.1 Overview of User Interface

From the *user’s perspective*, the following **UI subsystems** create the foundation **for the mobile application system**,

1. **Title Menu**
2. **Name Menu**
3. **Settings Menu**
4. **Pet View**
5. **Food Menu**
6. **Playing Menu**
7. **Status Menu**
8. **Drone Control Menu**

**5.1.1 Title Menu (reference 5.2.1)**

When opening the game, the user will be presented with the title screen and if it is the user’s first time opening the game, they will be asked to name their drone pet.

**5.1.2 Name Menu (reference 5.2.2)**

When starting the game for the first time or when restarting from the settings

menu, the user will be able to name their drone pet.

1. **Name Pet:** Allows the userto name their drone pet if it is the user’s first time on the application.

**5.1.3 Settings Menu (reference 5.2.3)**

In the Pet View, the user will be able to navigate to the settings menu to manage game settings.

1. **Restart game**: Allows the user to restart their game.

**5.1.4 Pet View (reference 5.2.4)**

This will function as the main screen the user will use to navigate to other menus.

1. **Food Menu**: Opens the Food Menu
2. **Playing Menu**: Opens the Playing Menu
3. **Status Menu**: Opens the Status Menu
4. **Drone Control Menu**: Opens the Drone Control Menu

**5.1.5 Food Menu (reference 5.2.6)**

This menu will allow the user to feed their drone pet by selecting from a choice of different foods.

1. **Select Food**: This will allow the user to select a specific *type of food* in the *Food Menu*.

**5.1.6 Playing Menu (reference 5.2.7)**

This menu will allow the user to select from three different games to play with their drone**.**

pet

1. **Dance Game**: The drone pet will perform dance moves
2. **Follow Game**: The drone pet will follow the user
3. **Photo Game**: The drone pet and the user will take pictures together

**5.1.7 Status Menu (reference 5.2.5)**

This menu will allow the user to view the status of their food, excitement, and energy.

1. **Food status**: The food status will be displayed through a food bar which shows how hungry or fed the drone pet is
2. **Excitement status**: The excitement status will be displayed through an excitement bar which shows whether the drone pet is having fun or not.
3. **Energy status**: The energy status will be displayed through an energy bar which shows the drone pet’s current energy.

**5.1.8 Drone Control Menu (reference 5.2.8)**

This menu will display drone information and allow the user control the drone through the *mobile application using joysticks* on the *screen* and also contain features, such as *takeoff/land, GoHome*, and allow the user to take a *picture.*

1. **GoHome**: This button will **tell** the drone pet to **go to** the designated **home location**.
2. **Takeoff/Land**: This button will **allow** the **drone** to **takeoff** or **land**.
3. **Take Picture:** This button will **take** a***picture*** of whatever the **drone screen** is displaying.
4. **Battery**: This will **display** the drone’s ***remaining battery***.
5. **Left Joystick**: This button will allow the user to manually **manage** the **drone’s** *altitude* and *rotation*.
6. **Right Joystick**: This button will allow the user to manually **control** the **drone** to **move** *forward*, *backward, left,* and *right.*

### 

### 

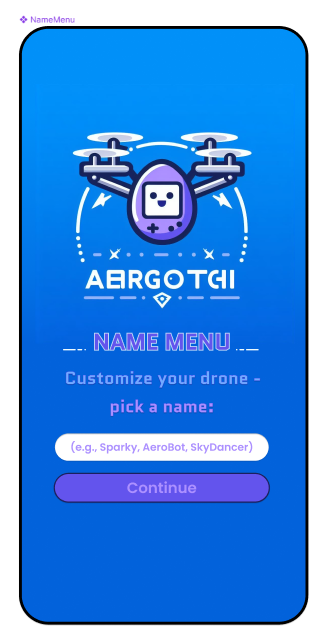
### 

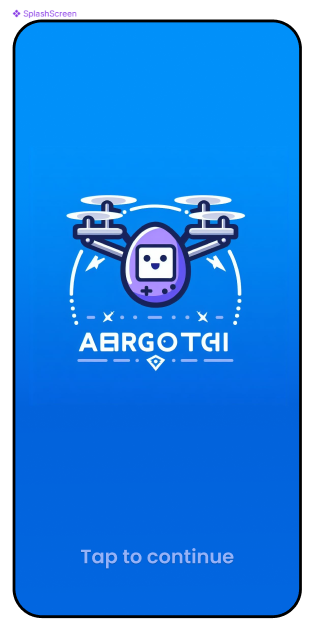
### 

### 

### 

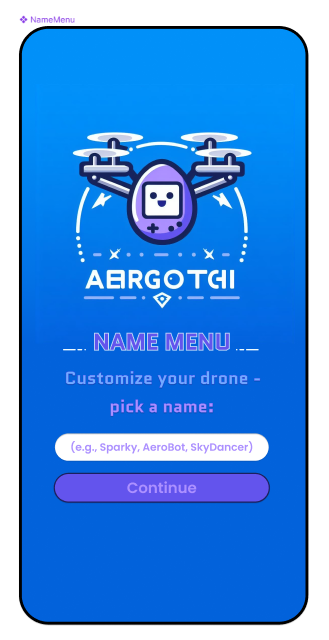
### 5.2 Screen Images



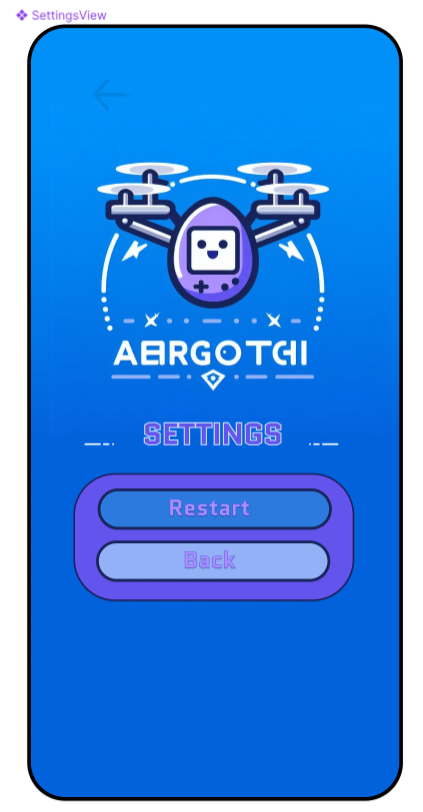
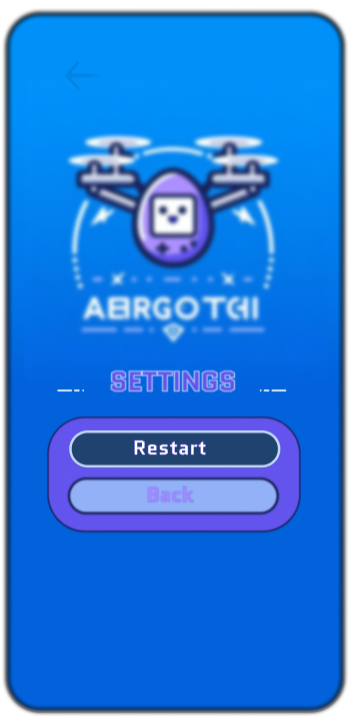
**5.2.1 TitleMenu**

### 5.2 Screen Images

**5.2.2 NameMenu**



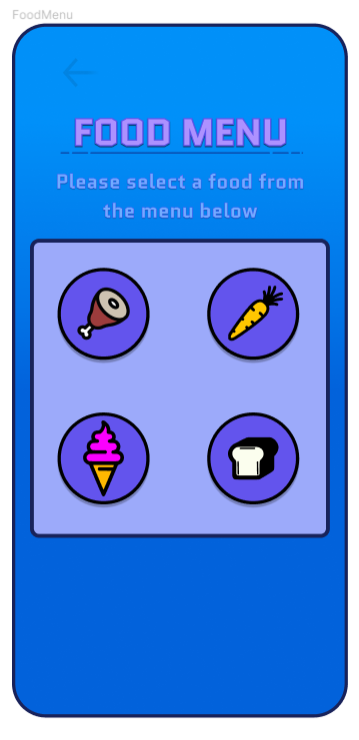
### 5.2 Screen Images

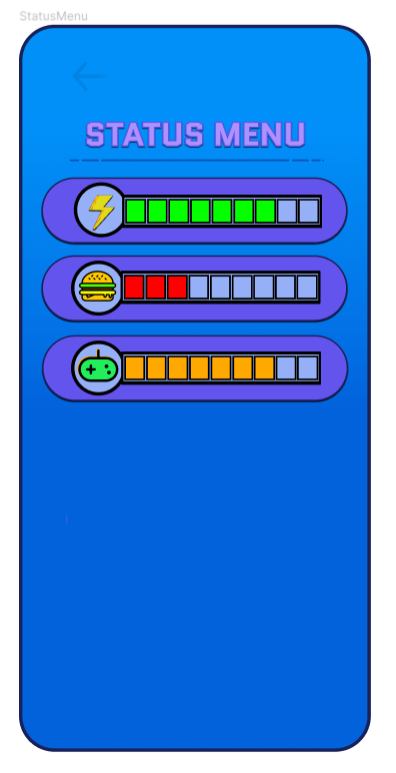
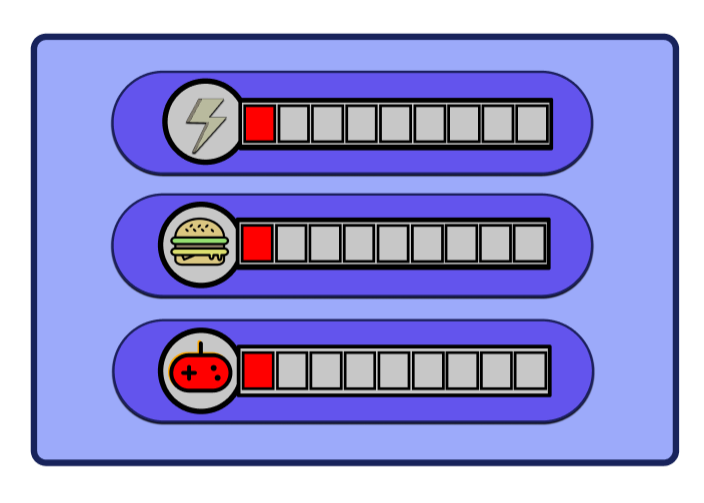
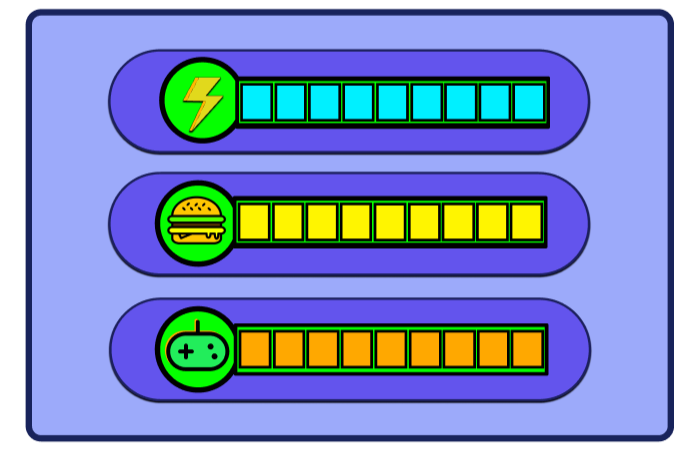
**5.2.3 SettingsMenu**

### 5.2 Screen Images

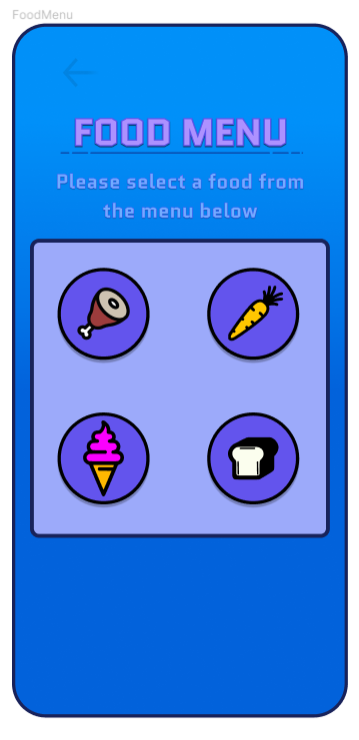
**5.2.4 PetView**



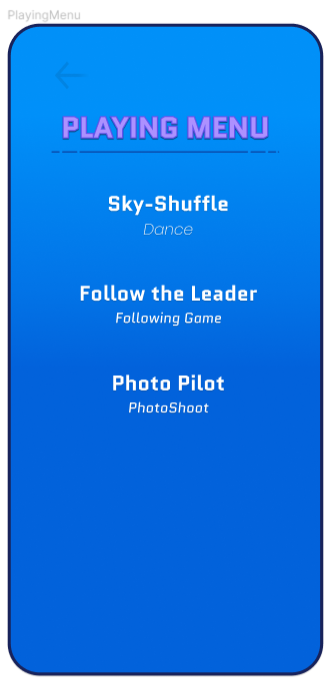
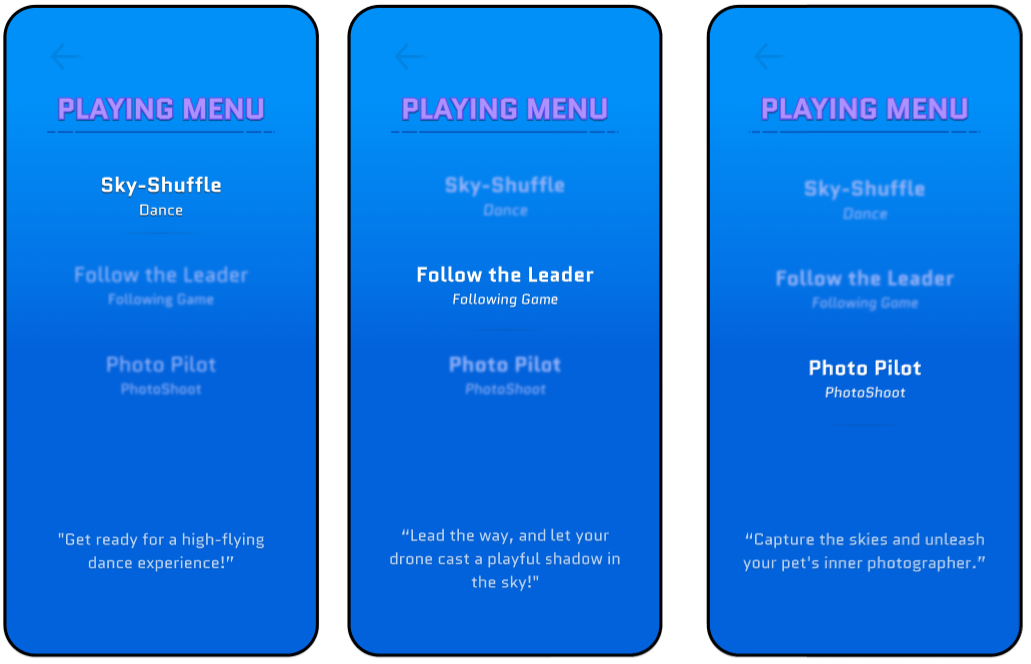
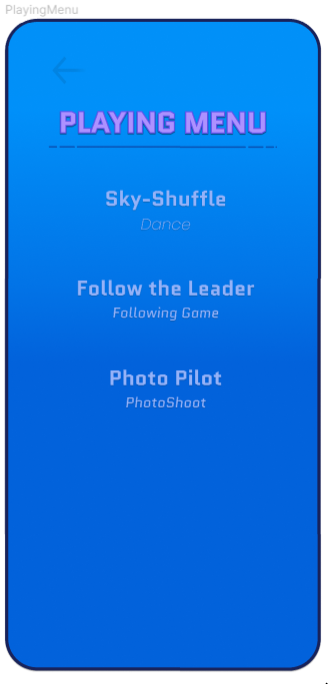
**5.2 Screen Images**

**5.2.5 StatusMenu**

**5.2 Screen Images**

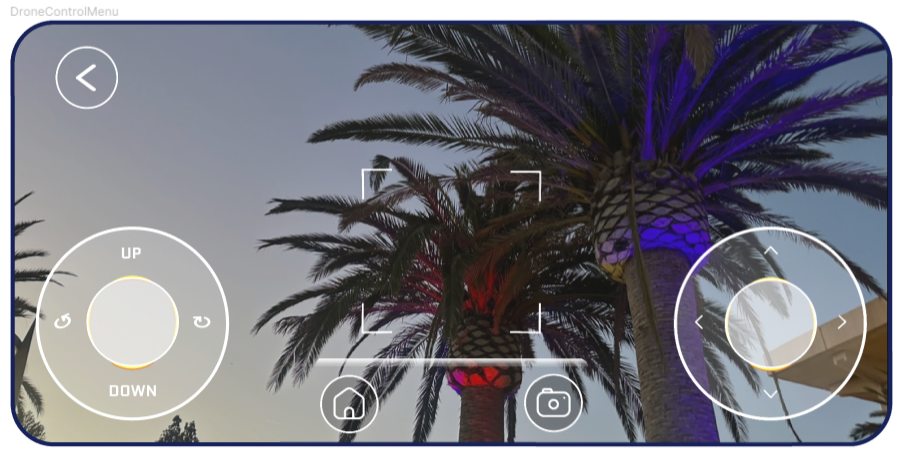
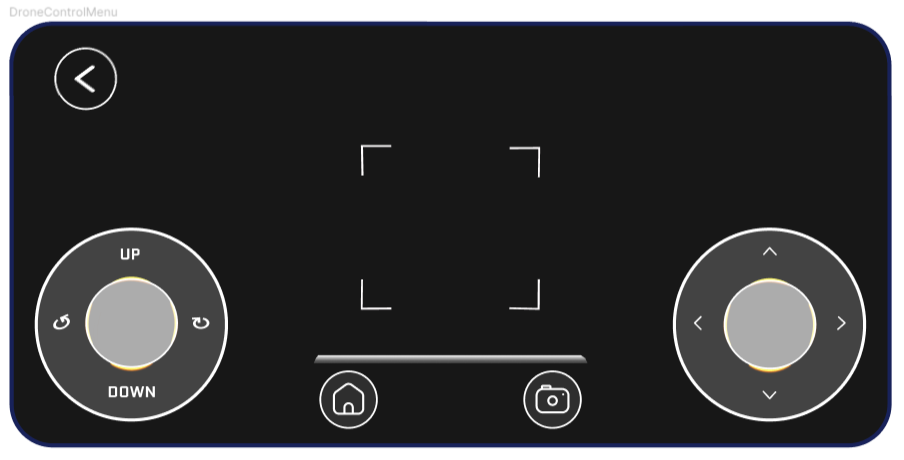
**5.2.6 FoodMenu**

**5.2 Screen Images**

**5.2.7 PlayingMenu**

**5.2 Screen Images**

**5.2.8 DroneControlMenu**



### 5.3 Screen Objects and Actions

The following is an **overview** of the **screen objects** and **action.**

1. **Go to Title: Allows** the **user** to **go** back to the **title screen from** the **settings menu**
2. **Panels: UI elements** that take up **sections of** the **screen** to **act as backdrop menus.**
3. **Food Bar: Displays food status**
4. **Fun Bar: Displays excitement status**
5. **Energy Bar: Displays energy status**

## 

## 

## 

## 6.0 REQUIREMENTS MATRIX

| **SRS Requirement ID** | **Paragraph Title** | **Component/Proof Num** | **Component**/  **Proof Name** |
| --- | --- | --- | --- |
| FUNC\_SRS\_001.1 | The drone shall be autonomous, but will have the option to be controlled manually using a physical controller and a smartphone application. | 4.1.17  4.1.13  4.1.8 | DroneSystem  DroneAction  DroneControlMenu |
| FUNC\_SRS\_001.2 | The drone shall use image recognition to enable the drone pet to identify and interact with specific objects or patterns in its environment. | 4.1.20 | AutonomousController |
| FUNC\_SRS\_001.3 | The drone shall display interactive behaviors, including responding to user commands. | 4.1.13  4.1.17 | DroneActions  DroneSystem |
| FUNC\_SRS\_001.4 | The drone shall display emotional expressions and engage in playful activities. | 4.1.9  4.1.12  4.1.17 | Fun  Personality  DroneSystem |
| FUNC\_SRS\_001.5 | The application shall allow the user to manually control the drone using a controller and have access to the game that is included for entertainment. | 4.1.8  4.1.9  4.1.17 | DroneControlMenu  Fun  DroneSystem |
| FUNC\_SRS\_001.6 | The game shall present periodic in-game challenges to maintain user engagement. | 4.1.7 | PlayingMenu |
| FUNC\_SRS\_003.2 | The drone shall have safety mechanics such as collision avoidance, distance to prevent accidents and safety concerns. | 4.1.20  4.1.17 | AutonomousController  DroneSystem |
| UIR\_SRS\_001 | The AeroGotchi game shall have a title screen that shows off the OoeyGUI logo and AeroGotchi design | 4.1.1 | TitleMenu |
| UIR\_SRS\_001.1 | After tapping the screen, the game will make sure it’s connected to the drone or ask to be connected. | 4.1.1 | TitleMenu |
| UIR\_SRS\_001.2 | If this is the first time connecting to the drone, the game will ask if you would like  to provide a name for the drone pet. | 4.1.2 | NameMenu |
| UIR\_SRS\_002 | The Aerogotchi game shall have a Name Menu that allows the User to name their pet drone and will also display if the User restarts the game. | 4.1.2  5.2.2 | NameMenu  NameMenu Screen Image |
| UIR\_SRS\_003 | The game shall have a Pet View where you can see the pet character with several Menu Icons surrounding the border to provide room for the main visuals | 4.1.4 | PetView |
| UIR\_SRS\_003.1 | Pet View shall have 8-bit visuals of drone pet. | 5.2.4 | PetView Screen Image |
| UIR\_SRS\_003.2 | Drone Pet shall have features based on game status and mechanics. | 4.2.3 | Pet View SD |
| UIR\_SRS\_004 | The Settings Menu shall be accessible from the Pet View screen. | 4.2.1  5.1.3  5.2.4 | Front End SD  Pet View UI Overview  Pet View Screen Image |
| UIR\_SRS\_004.1 | Settings Menu shall have a “Restart” option that resets the app, forces the User to the Title Menu, and allows the User to reconnect the drone with a new name. | 5.2.3 | Settings Menu Screen Image |
| UIR\_SRS\_005 | The Drone Control Menu shall be accessible from the Pet View screen. | 4.2.1  5.1.3  5.2.4 | Front End SD  Pet View UI Overview  Pet View Screen Image |
| UIR\_SRS\_005.1 | Drone Control Menu shall connect to the drone camera for viewing and flying purposes. | 4.1.8  4.1.14 | DroneControlMenu  DroneVisual |
| UIR\_SRS\_005.2 | Drone Control Menu shall allow the user to fly the drone with a controller directly connected to the mobile device. | 5.2.8 | Drone Control Menu Screen Image |
| UIR\_SRS\_005.3 | Drone Control Menu shall have a button titled “Go Home” that tells the drone to fly to its “Home” spot. | 4.1.8  4.1.15  4.1.16  5.2.8 | DroneControlMenu  HomeHubIdentifier  HubLocation  DroneControlMenu Screen Image |
| UIR\_SRS\_006 | The Status Menu shall be accessible from the Pet View screen. | 4.2.1  5.1.3  5.2.4 | Front End SD  Pet View UI Overview  Pet View Screen Image |
| UIR\_SRS\_006.1 | Status Menu shall have an Energy Bar to display the drone pet’s “energy” or actual battery percentage. | 4.1.5  4.1.11 | StatusMenu  Energy |
| UIR\_SRS\_006.2 | Status Menu shall have a Food Bar to display the drone pet’s “hunger.” | 4.1.5  4.1.0 | StatusMenu  FoodType |
| UIR\_SRS\_006.3 | Status Menu shall have a Fun Bar to display the drone pet’s level of “excitement”. | 4.1.5  4.1.9 | StatusMenu  Fun |
| UIR\_SRS\_007 | The Food Menu shall be accessible from the Pet View screen. | 4.2.1  5.1.3  5.2.4 | Front End SD  Pet View UI Overview  Pet View Screen Image |
| UIR\_SRS\_007.1 | Food Menu shall have options of different Food to serve the drone pet depending on Drone Personality and other game factors. | 4.1.6 | FoodMenu |
| UIR\_SRS\_007.2 | Food Menu shall have different foods that affect Drone Statuses differently based on Drone Personality | 5.2.6 | FoodMenu Screen Image |
| UIR\_SRS\_008 | The Playing Menu shall be accessible from the Pet View screen. | 4.2.1  5.1.3  5.2.4 | Front End SD  Pet View UI Overview  Pet View Screen Image |
| UIR\_SRS\_008.1 | Playing Menu shall have access to different predefined game modes that the User and drone pet can interact with | 4.1.7 | PlayingMenu |
| UIR\_SRS\_008.2 | Playing Menu shall have different activities that affect Drone Statuses differently based on Drone Personality. | 5.2.7 | Playing Menu Screen Image |
| UIR\_SRS\_009 | The application shall have background notifications to notify the User of certain mechanics, such as the food bar, fun bar, or energy bar | 4.1.9  4.1.10  4.1.11 | Fun  FoodType  Energy |