UML Design

Documentation

for

<PharmaDrone Project>

Prepared by <Zakiyyah Harris, Chelsea Davis, Shaniyah, Nikitha>

<UAB>

<3.22.24>

Table of Contents

Table of Contents..... 2
1. Introduction .....
1 1.1 Purpose 1
1.2 Document Conventions..... 1 **2.**
Overall Description 1
2.1 Project Purpose and System Components 1
2.2 Relevant UML Documentation 1
2.3 UML Introduction..... 1
2.4 Diagrams of UML ..... 2 **3.**
Specified Mobile Application UML Diagrams..... 3
3.1 Use Case Diagram 3
3.2 Class Diagram 4
3.3 State Diagram..... 4

1. Introduction

1.1 Purpose

This UML Design Document details a state diagram, a UML diagram, and a wireframe as well as brief explanations of how they relate to our Pharma Drone concept.

2. Overall Description

2.1 Project Purpose and System Components

A client, an employee of the store pharmacy, will log into a store computer, and activate the drone program. The user will input the relevant coordinates, and after the program checks that the drone is capable of doing a full round-trip journey, will fly into the air to its destination. After it lands at its destination, the drone returns to the storefront. After a successful delivery, a Report receipt document important client information is created and sent to the Archive. The Archive is only accessible to the manager, and should be cleared out daily .

2.3 UML Introduction

UML or Unified Modeling Language is used in object oriented software engineering as a tool for modeling application structures, behavior, and some processes and states. The two basic categories of UML diagrams are structure diagrams, which show static structure of the system, and

behavior diagrams, which show dynamic behavior between objects within the system. Class diagrams fall into the structure category while use case and state diagrams fall into the behavior category.

2.4 Diagrams of UML

We have 3 parts to our UML showcase: a class diagram showcasing the major relevant classes in action within the Pharma Drone and their relation to one another, a wireframe showcasing the UI that the client will interact with during usage to connect with the drone and initiate deliveries, and the state diagram regarding the current state of the Pharma Drone during usage on a delivery.

Relevant commentary will be added underneath each diagram.

3. Specified Mobile Application UML Diagrams

3.1 Class Diagram

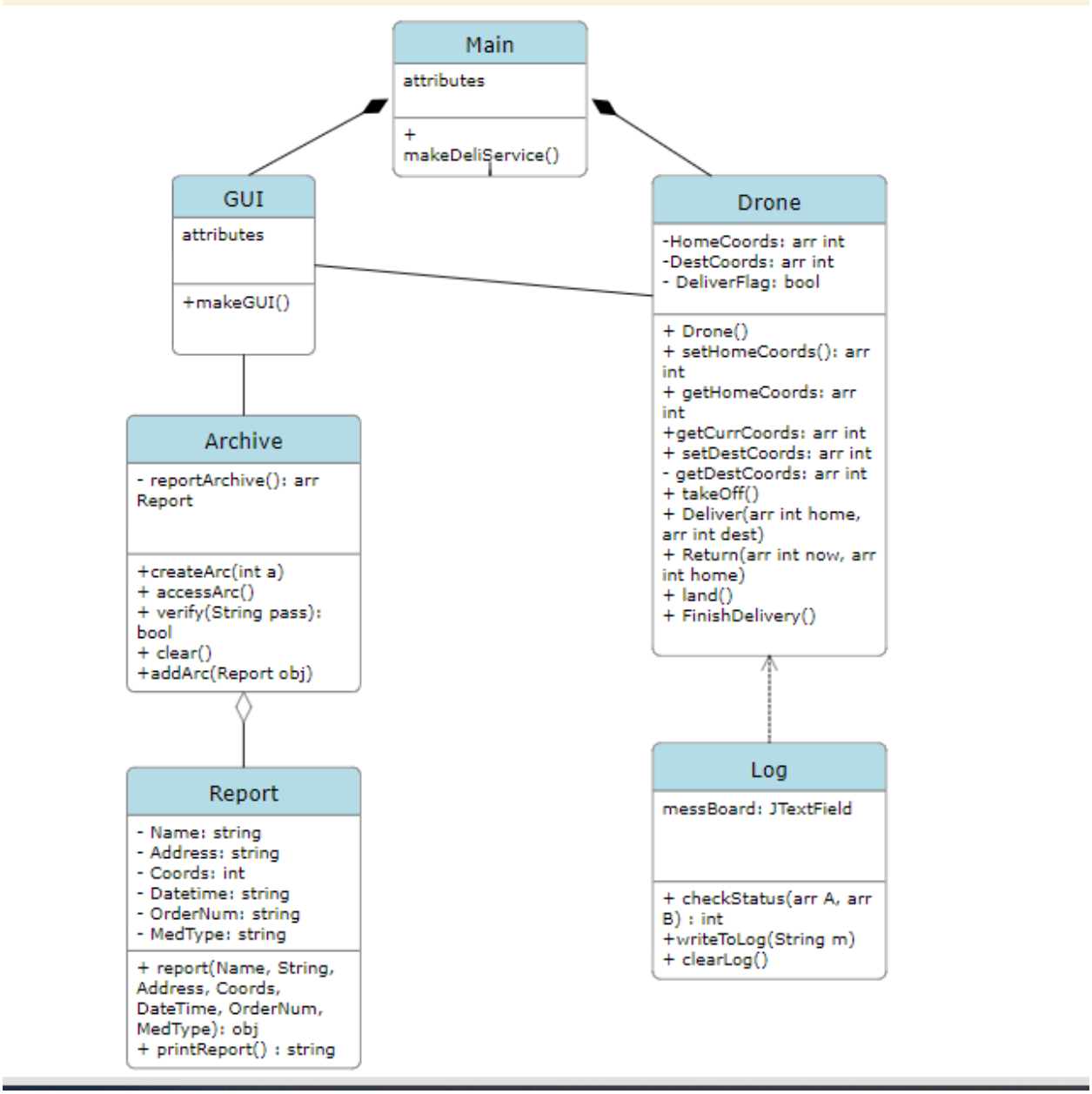

Figure 1: Use Case Diagram for PharmaDrone project

The class diagram above details the major classes that make up the Pharma Drone Project, and how they interact with each other. Save for Report, there should be exactly one instance of each class created when the program is running.

Main:

- makeDeliveryService(): calls for a new GUI and Drone to be made, also allows functionality to shut the program down.

GUI:

- makeGUI: creates the GUI for the project

Report:

- Creates a record of a finished delivery:
- When initialized (report()), all attributes must be inputted as parameters
- printReport reports a “polished” version of the attribute info in a easy to read format

Archive:

- Attribute ReportArc() stores all Report objects made from a finished Delivery
- access() calls verify() to confirm the user has rightful access, if returned true, displays full log using a JTextPanel attribute.
- clear() will wipe all the day’s reports
- addArc() adds a new Report to ReportArc().

Drone:

- Attribute HomeCoords should always point towards the location of the client’s place of operation, and DestCoords a destination for a certain delivery.
- DeliverFlag ensures that only one delivery is carried at a time (If True = I’m busy on a delivery, else I’m free for a new delivery)
- Drone() should call the necessary methods to ensure connection to the Drone.
- takeOff() is when the drone lifts off and hovers in the air in a stationary position, but only after checking its connection, battery level, and if DeliverFlag() is set.
- Deliver() is the actual routing from Home to Destination of Delivery, where the drone flies.
- Return() should be able to be called at any time in case of emergency, but should typically be called after land().
- FinishDelivery resets the DeliverFlag to false and causes a JPanel popup to occur to ensure a Report object is created and slotted into the Archive.

Log

- Operates separately from the main GUI because it is purely concerning information taken from Drone.
- checkStatus() calls getCurrCodes from Drone, compares them to a call of getHomeDrone to determine what % of the delivery is done.
- writetoLog() writes a particular message to the messBoard depending on checkStatus
- clearLog() clears the log for the next delivery after checkStatus reaches 100%.

3.2 Wireframe Diagram

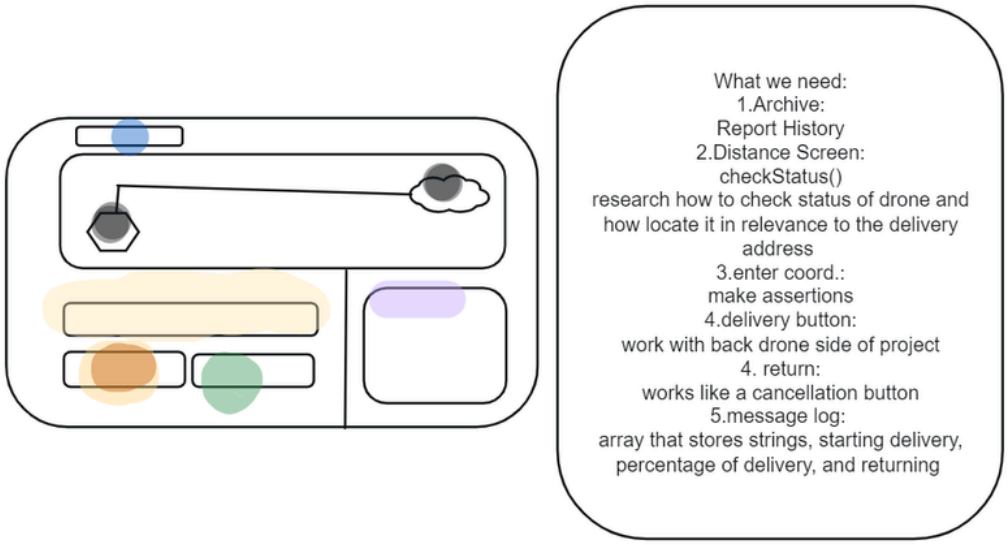

Figure 2: Wireframe for PharmaDrone

The wireframe made above describes and details the user interface that will be used by the client. In addition to this we have added an additional button to the UI, which is a logout button. The yellow highlighted area on the wireframe is where the client will input coordinates of their clients to be inputted into the drone's destination coordinates. The orange highlighted button is used to return the drone and cancel drone routes/trips. The green highlighted button is used to start drone deliveries. The purple highlighted box is the message log box, where all the messages concerning the drone will be printed out such as percentages, and any errors. The box with black highlights is the box where the drone's location in comparison with the destination of the drone.

3.3 State Diagram

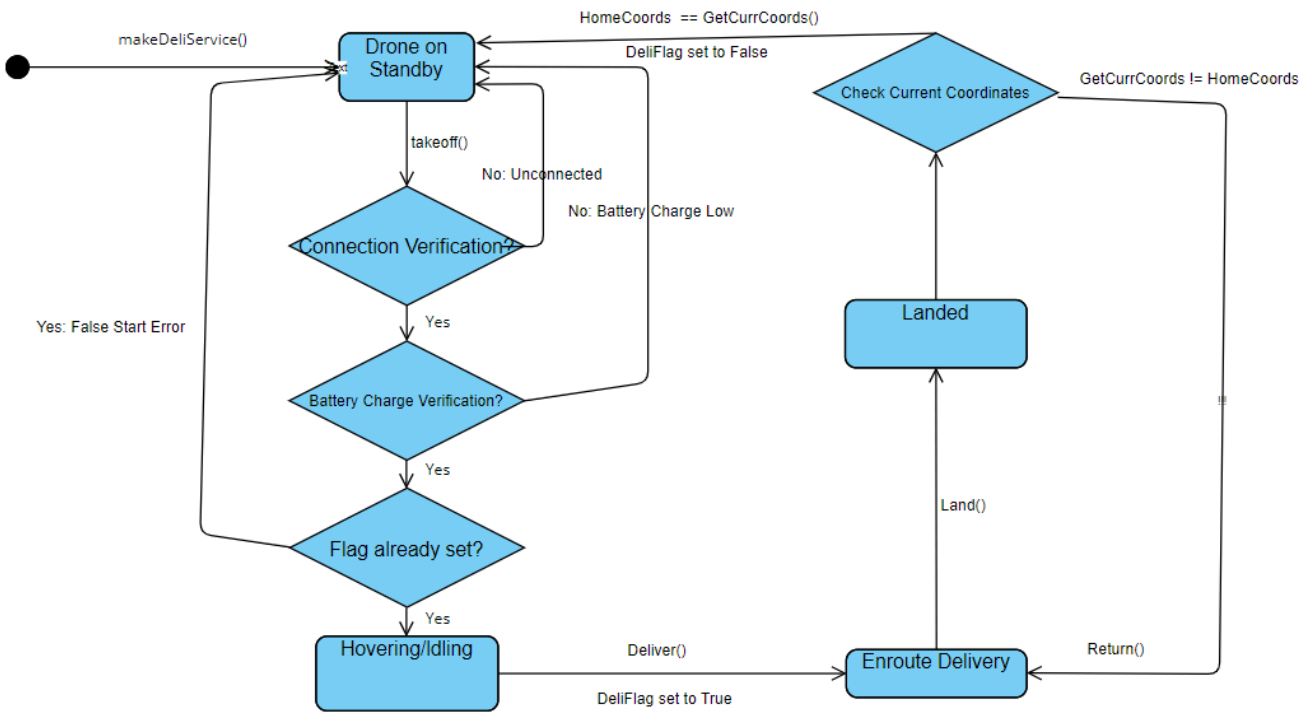

Figure 1: Use Case Diagram for PharmaDrone project

The state diagram above details the events that happen to the Drone upon the program being activated. Our Drone has four major states represented in the state chart: Standby, Hovering, Enroute,and Landed. When takeoff() is called, the drone checks for several factors(connectivity, battery, If another delivery is already in motion if the Deliflag is set to true), and if they are all true, it will fly straight up and idly hover. If not, it returns to standby for someone to fix the issues. Then deliver() is called, and the drone files to its destination, in the Enroute. When land() is called, it enters in Landed() state: if it is landed at base, the drone re-enters standby, but if it's not, return() is called.