4th year project
Automated Drone Air Traffic Control System

Design Manual

Institute of Technology Carlow

Supervisor: Dr. Oisin Cawley
Author: James Hall
Submission Date: 27 November 2020

Abstract

The purpose of this document is to detail the design aspect of the project. This includes a class diagram and sequence diagrams for major functional aspects, as well as the projects file structure.

Table of Contents

Table of Figures

Introduction

This document will give an overview of the design of the Automated Drone Air Traffic Control System through UML diagrams, including a Class diagram and Sequence diagrams.A UI mockup will detail the layout of the User Interface. It will also detail the technologies used for the realisation of the project.

More information can be found at: https://github.com/Jamhougin/DroneTrafficControlSystem

UI Layout

The UI layout for the main screen contains the map and 3 colour coded sets of buttons relating to CRUD operations on Drones, Flights and Locations. Clicking on these buttons creates the popups detailed below..

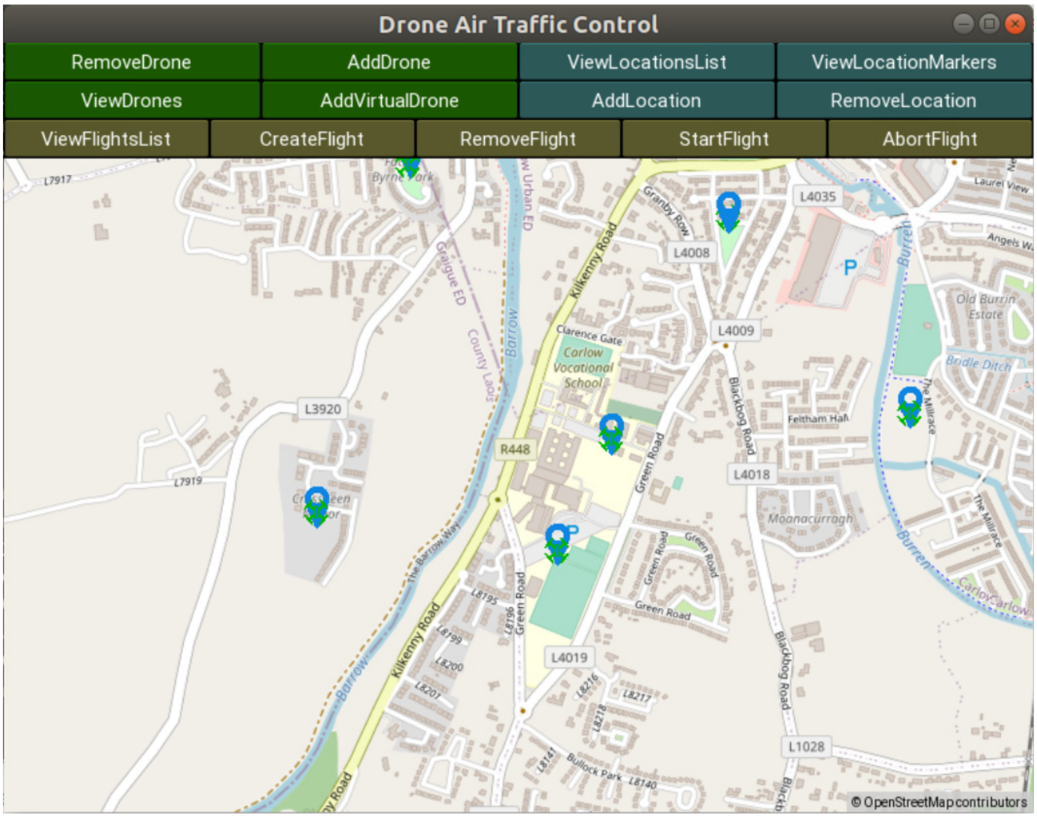


Figure 1 Main Screen

UML Diagrams

Class Diagram

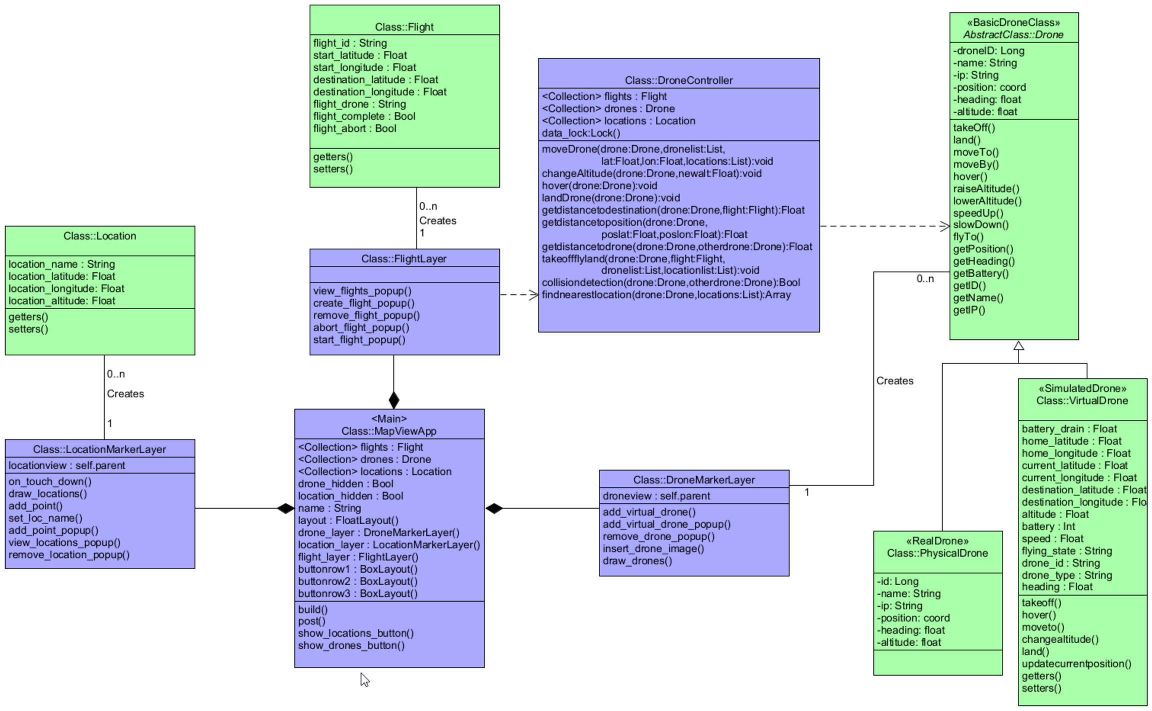
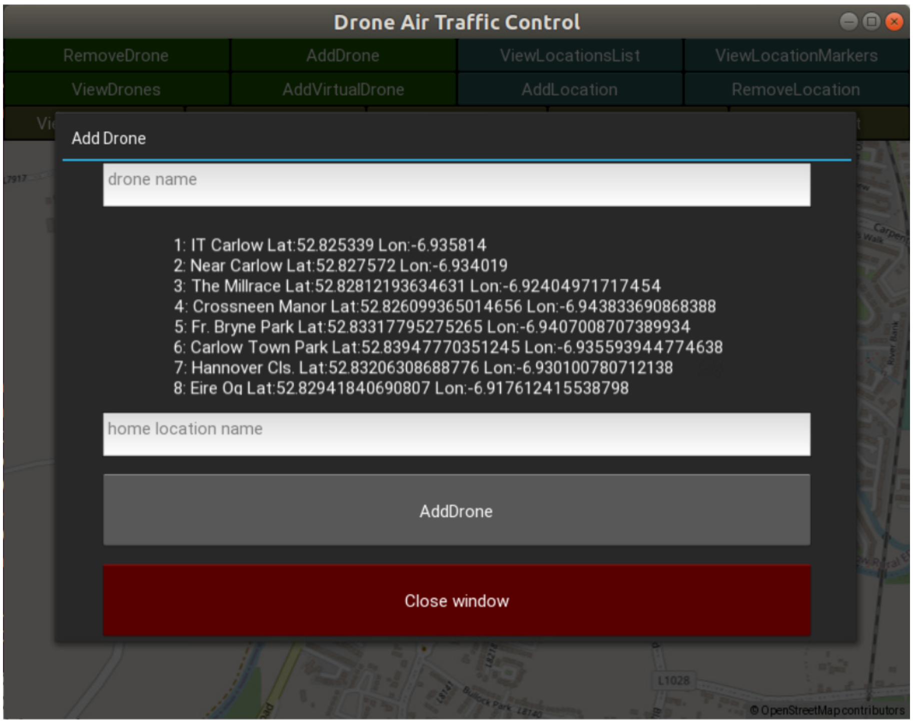


Figure 2 System Class Diagram

Sequence Diagrams

add_virtual_drone_popup()

Screen:



1: IT Carlow Lat:52.825339 Lon:-6.935814
2: Near Carlow Lat:52.827572 Lon:-6.934019
3: The Millrace Lat:52.82812193634631 Lon:-6.92404971717454
4: Crossneen Manor Lat:52.826099365014656 Lon:-6.943833690868388
5: Fr. Bryne Park Lat:52.83317795275265 Lon:-6.9407008707389934
6: Carlow Town Park Lat:52.83947770351245 Lon:-6.935593944774638
7: Hannover Cls. Lat:52.83206308688776 Lon:-6.930100780712138
8: Eire Og Lat:52.82941840690807 Lon:-6.917612415538798

Figure 3 Add drone popup

Description:
The Operator clicks on the "Add drone button" and is presented with a popup allowing them to enter the details needed to create the drone. They may click the "add drone" button within the popup which creates a Drone object and append this object to the system's list of drones.

Sequence Diagram:

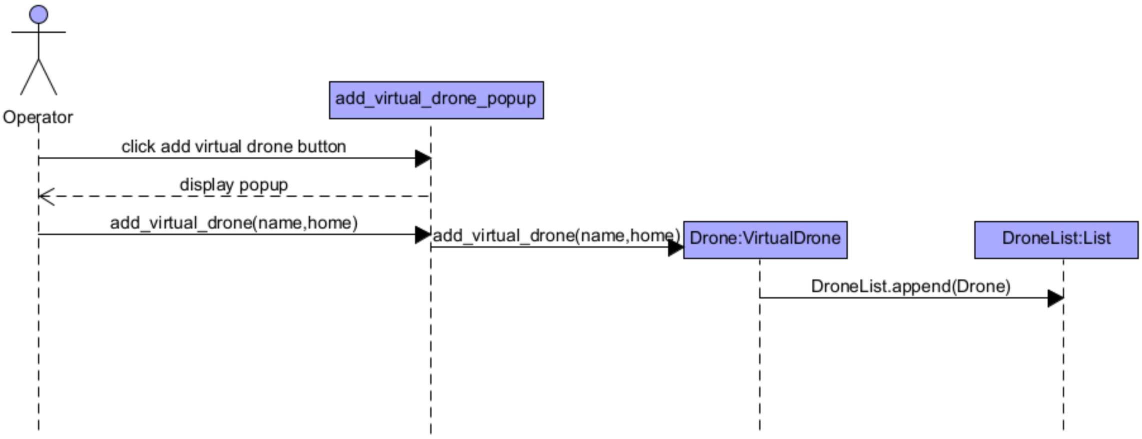
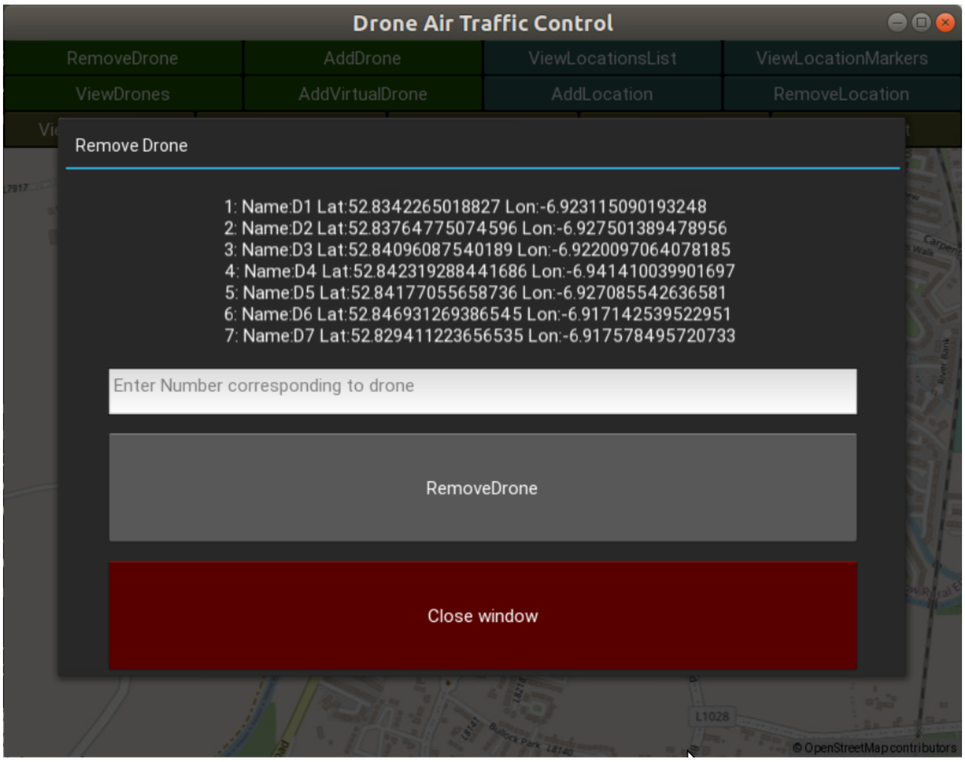


Figure 4 Add virtual drone sequence diagram

remove_drone_popup()

Screen:



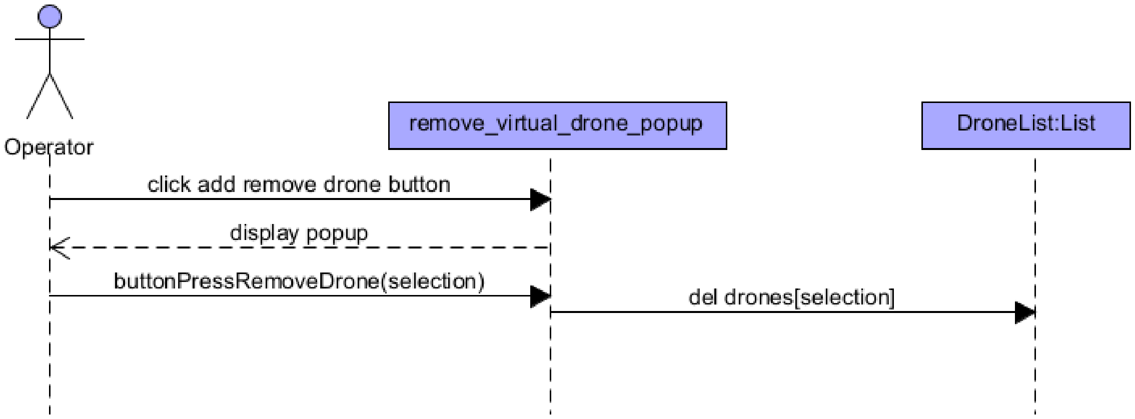
```
1: Name:D1 Lat:52.8342265018827 Lon:-6.923115090193248
2: Name:D2 Lat:52.83764775074596 Lon:-6.927501389478956
3: Name:D3 Lat:52.84096087540189 Lon:-6.9220097064078185
4: Name:D4 Lat:52.842319288441686 Lon:-6.941410039901697
5: Name:D5 Lat:52.84177055658736 Lon:-6.927085542636581
6: Name:D6 Lat:52.846931269386545 Lon:-6.917142539522951
7: Name:D7 Lat:52.829411223656535 Lon:-6.917578495720733
```

Enter Number corresponding to drone
RemoveDrone
Close window
© OpenStreetMap contributors

Figure 5 Remove drone popup

Description:
The operator clicks on the remove drone button and is presented with a popup menu allowing them to select a drone to be deleted. Upon pressing the remove drone button within the popup, the drone at the selected location in the drone list is deleted from the system.

Sequence Diagram:



Operator
remove_virtual_drone_popup
DroneList:List
click add remove drone button
display popup
buttonPressRemoveDrone(selection)
del drones[selection]

Figure 6 Remove drone sequence diagram

add_point_popup()

Screen:

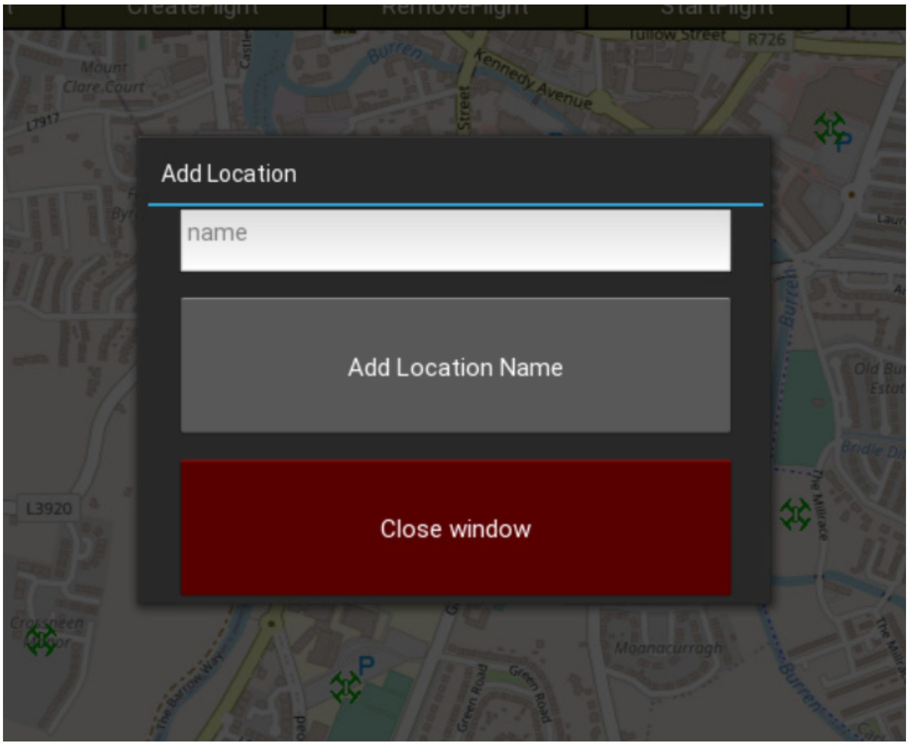


Figure 7 Add location popup

Description:
The operator clicks on the add location button and is presented with a popup enabling them to enter in the name for the new location. On pressing the “Add location name” button the popup closes and then next left click with the mouse on the map creates a new location object with the chosen name and the coordinates corresponding to the map position clicked. The new location object is then added to the location list.

Sequence Diagram:

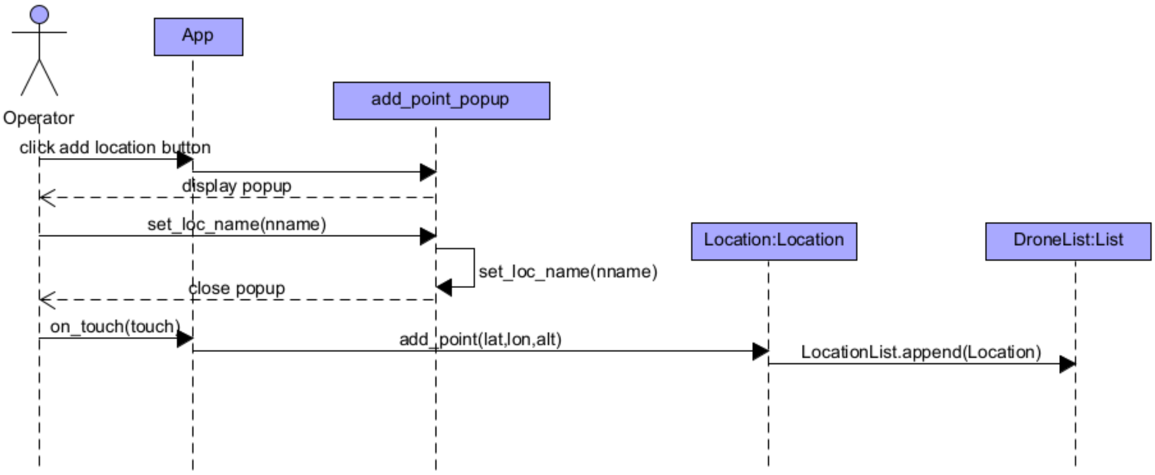
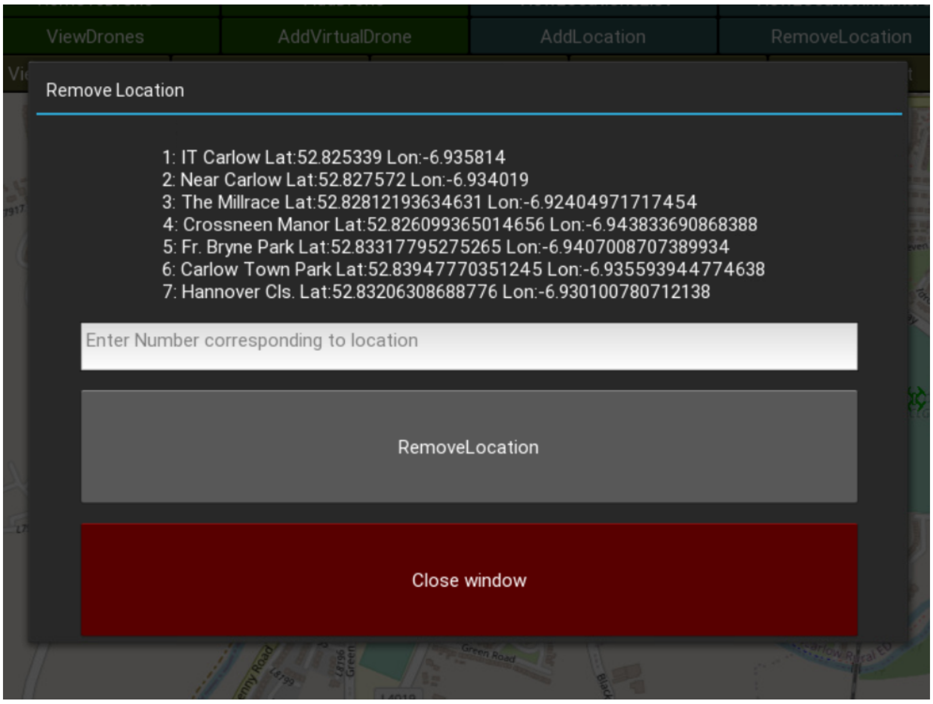


Figure 8 Add location sequence diagram

remove_location_popup()

Screen:



1: IT Carlow Lat:52.825339 Lon:-6.935814
2: Near Carlow Lat:52.827572 Lon:-6.934019
3: The Millrace Lat:52.82812193634631 Lon:-6.92404971717454
4: Crossneen Manor Lat:52.826099365014656 Lon:-6.943833690868388
5: Fr. Bryne Park Lat:52.83317795275265 Lon:-6.9407008707389934
6: Carlow Town Park Lat:52.83947770351245 Lon:-6.935593944774638
7: Hannover Cls. Lat:52.83206308688776 Lon:-6.930100780712138

Figure 9 Remove location popup

Description:
The operator clicks on the remove location button and is presented with a popup displaying a list of all locations on the system. They may then enter the number corresponding to the desired location to remove and upon clicking the “remove location” button, the location will be deleted from the system.

Sequence Diagram:

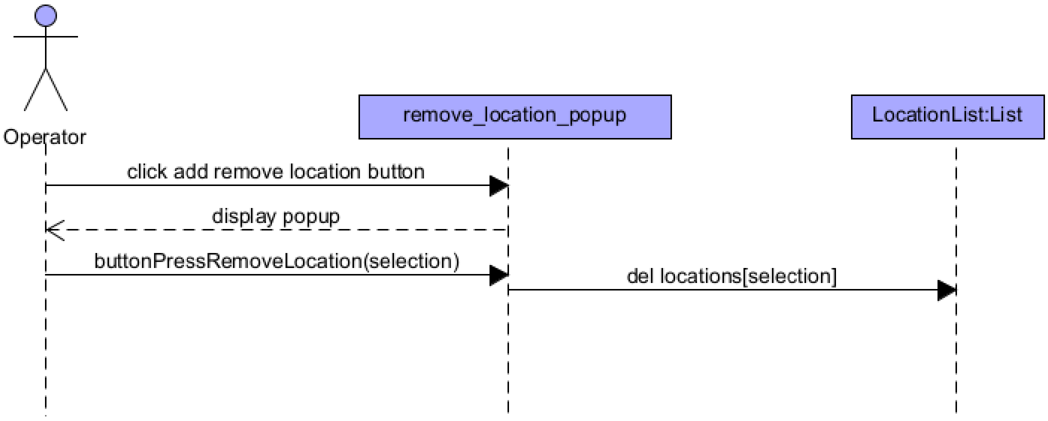
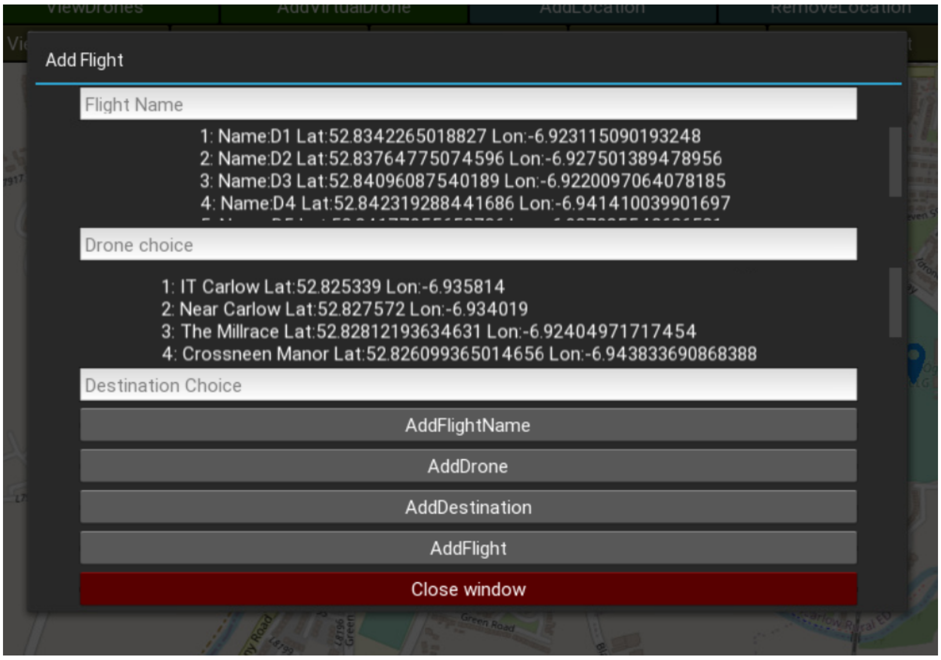


Figure 10 Remove location sequence diagram

create_flight_popup()

Screen:



```
1: Name:D1 Lat:52.8342265018827 Lon:-6.923115090193248
2: Name:D2 Lat:52.83764775074596 Lon:-6.927501389478956
3: Name:D3 Lat:52.84096087540189 Lon:-6.9220097064078185
4: Name:D4 Lat:52.842319288441686 Lon:-6.941410039901697
```

```
1: IT Carlow Lat:52.825339 Lon:-6.935814
2: Near Carlow Lat:52.827572 Lon:-6.934019
3: The Millrace Lat:52.82812193634631 Lon:-6.92404971717454
4: Crossneen Manor Lat:52.826099365014656 Lon:-6.943833690868388
```

Figure 11 Create flight popup

Description:
The operator clicks on the create flight button and is presented with a popup displaying a list of all drones and locations on the system. They may enter the name of the new flight, the number relating to the desired drone for the flight, and the number relating to the desired destination for the drone to fly to. There is a separate button to add the Flight name, drone and destination and a final button to confirm and create the new flight with the entered details. Upon clicking this button the new flight object is created and added to the flight list.

Sequence Diagram:

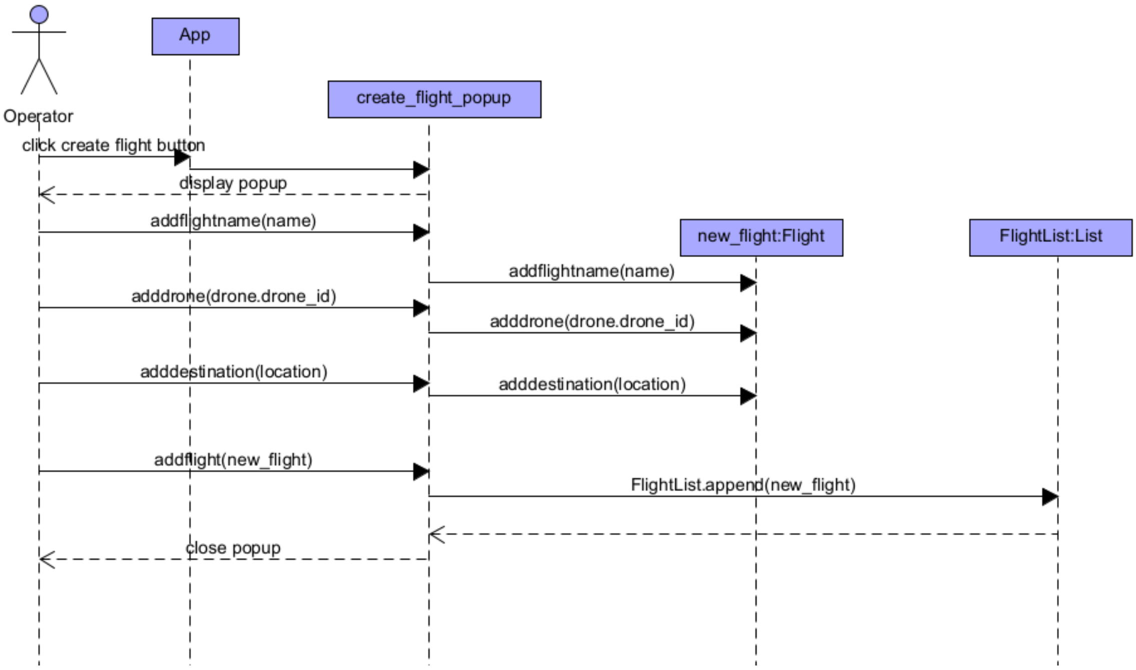
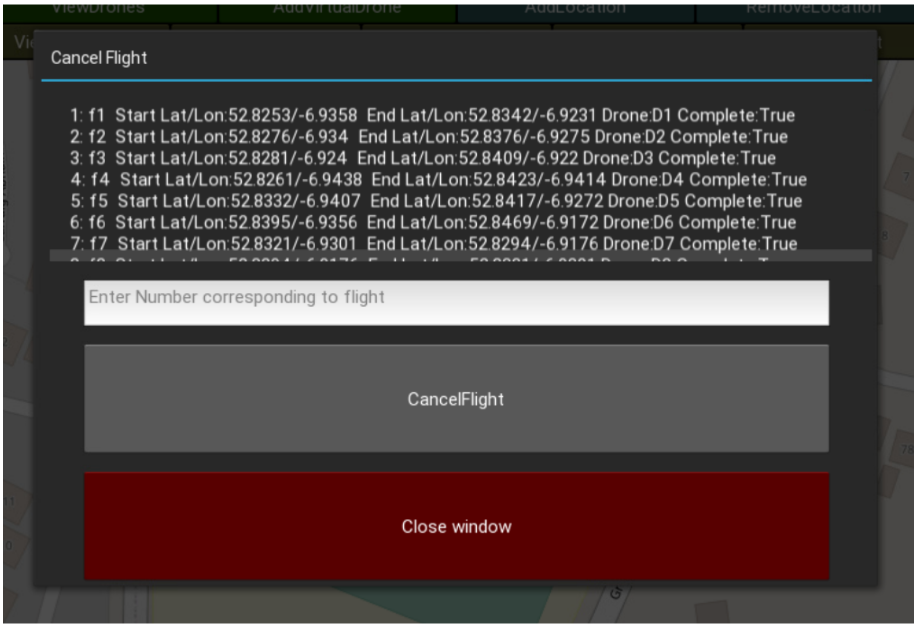


Figure 12 Create flight sequence diagram

remove_flight_popup()

Screen:



Cancel Flight

```
1: f1  Start Lat/Lon:52.8253/-6.9358  End Lat/Lon:52.8342/-6.9231 Drone:D1 Complete:True
2: f2  Start Lat/Lon:52.8276/-6.934  End Lat/Lon:52.8376/-6.9275 Drone:D2 Complete:True
3: f3  Start Lat/Lon:52.8281/-6.924  End Lat/Lon:52.8409/-6.922 Drone:D3 Complete:True
4: f4  Start Lat/Lon:52.8261/-6.9438  End Lat/Lon:52.8423/-6.9414 Drone:D4 Complete:True
5: f5  Start Lat/Lon:52.8332/-6.9407  End Lat/Lon:52.8417/-6.9272 Drone:D5 Complete:True
6: f6  Start Lat/Lon:52.8395/-6.9356  End Lat/Lon:52.8469/-6.9172 Drone:D6 Complete:True
7: f7  Start Lat/Lon:52.8321/-6.9301  End Lat/Lon:52.8294/-6.9176 Drone:D7 Complete:True
```

Figure 13 Remove flight popup

Description:

When the operator clicks the remove flight button, they are presented with a popup displaying a list of all flights on the system. They may enter a number corresponding to the desired flight to remove and upon clicking the “cancel flight” button, the desired flight is deleted from the system.

Sequence Diagram:

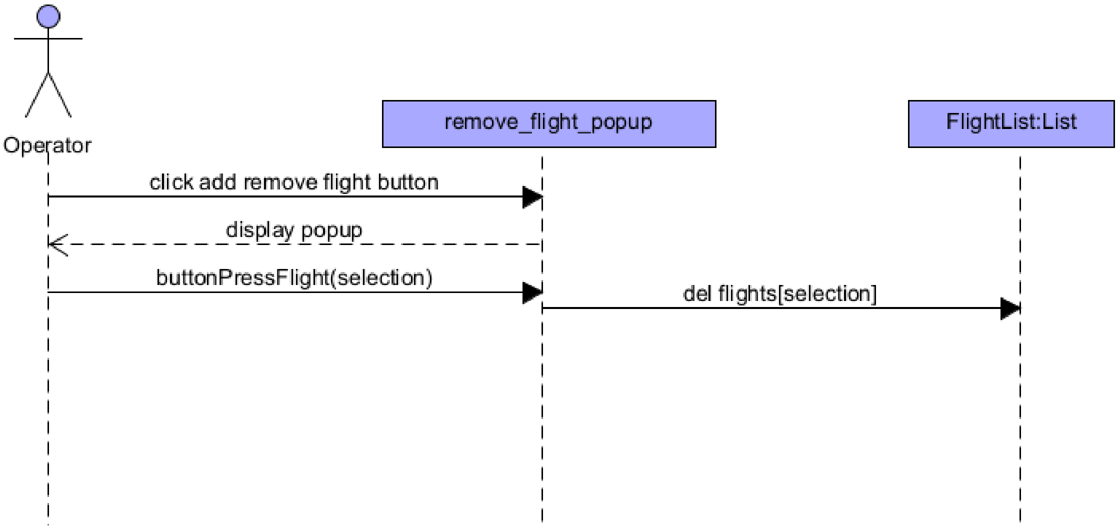
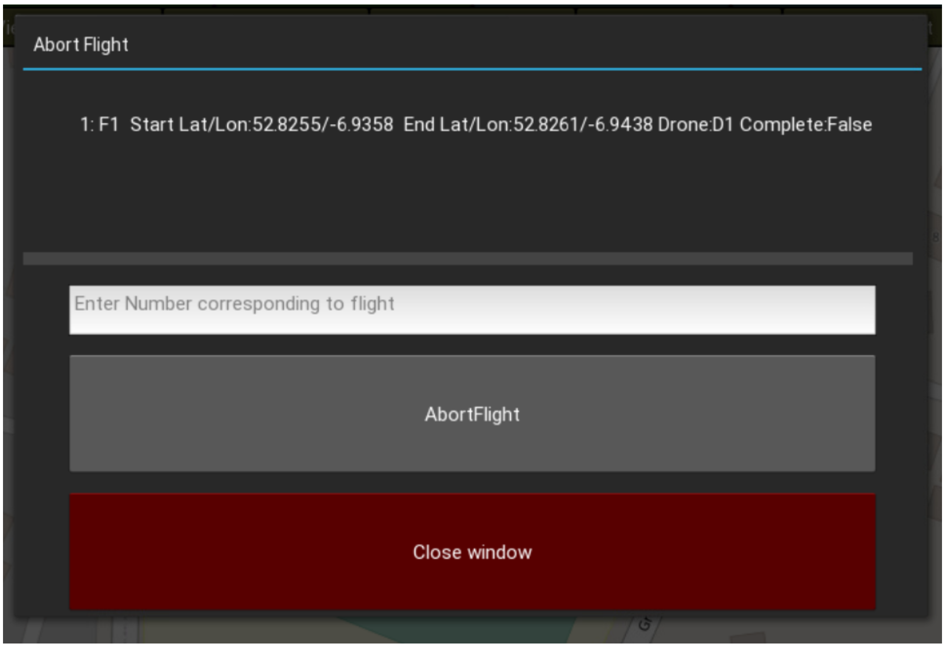


Figure 14 Remove flight sequence diagram

abort_flight_popup()

Screen:



Abort Flight

1: F1 Start Lat/Lon:52.8255/-6.9358 End Lat/Lon:52.8261/-6.9438 Drone:D1 Complete:False

Enter Number corresponding to flight

Figure 15 Abort flight popup

Description:

When the operator clicks on the Abort flight button, they are presented with a popup displaying a list of all flights on the system. They may choose a number using a numerical input box that corresponds to the flight they wish to abort. Upon pressing “abort flight” the abort_flight flag in the chosen flight is set to true. The current MoveDrone() function being executed checks the current flight object’s abort_flight flag and if true, a function is called which checks the position of all locations in relation to the drone and changes the drones destination to the closest location. The MoveDrone() function then returns to its standard loop until completion.

Sequence Diagram:

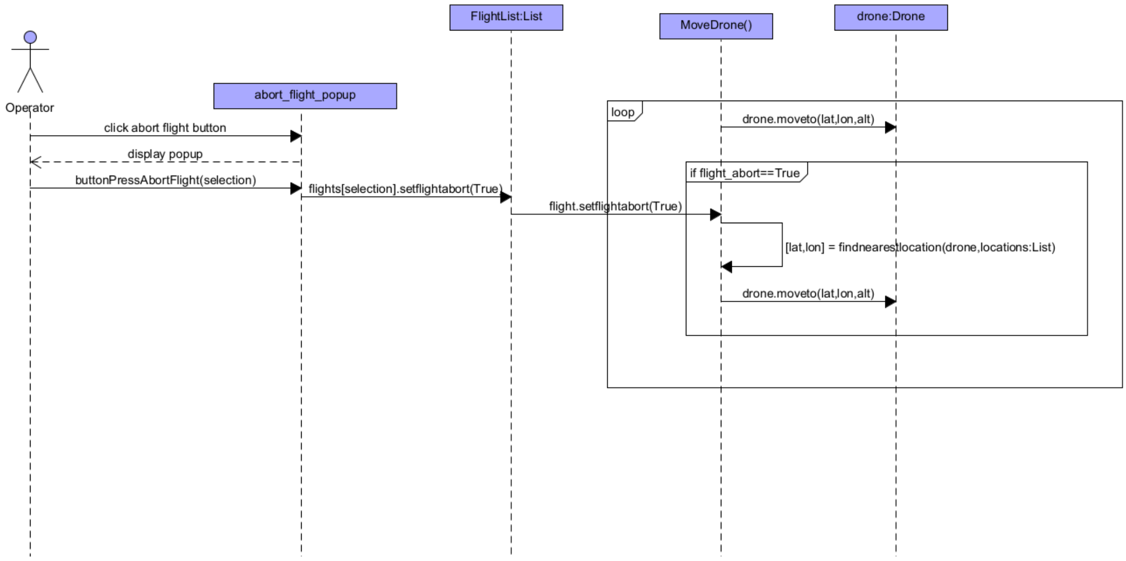
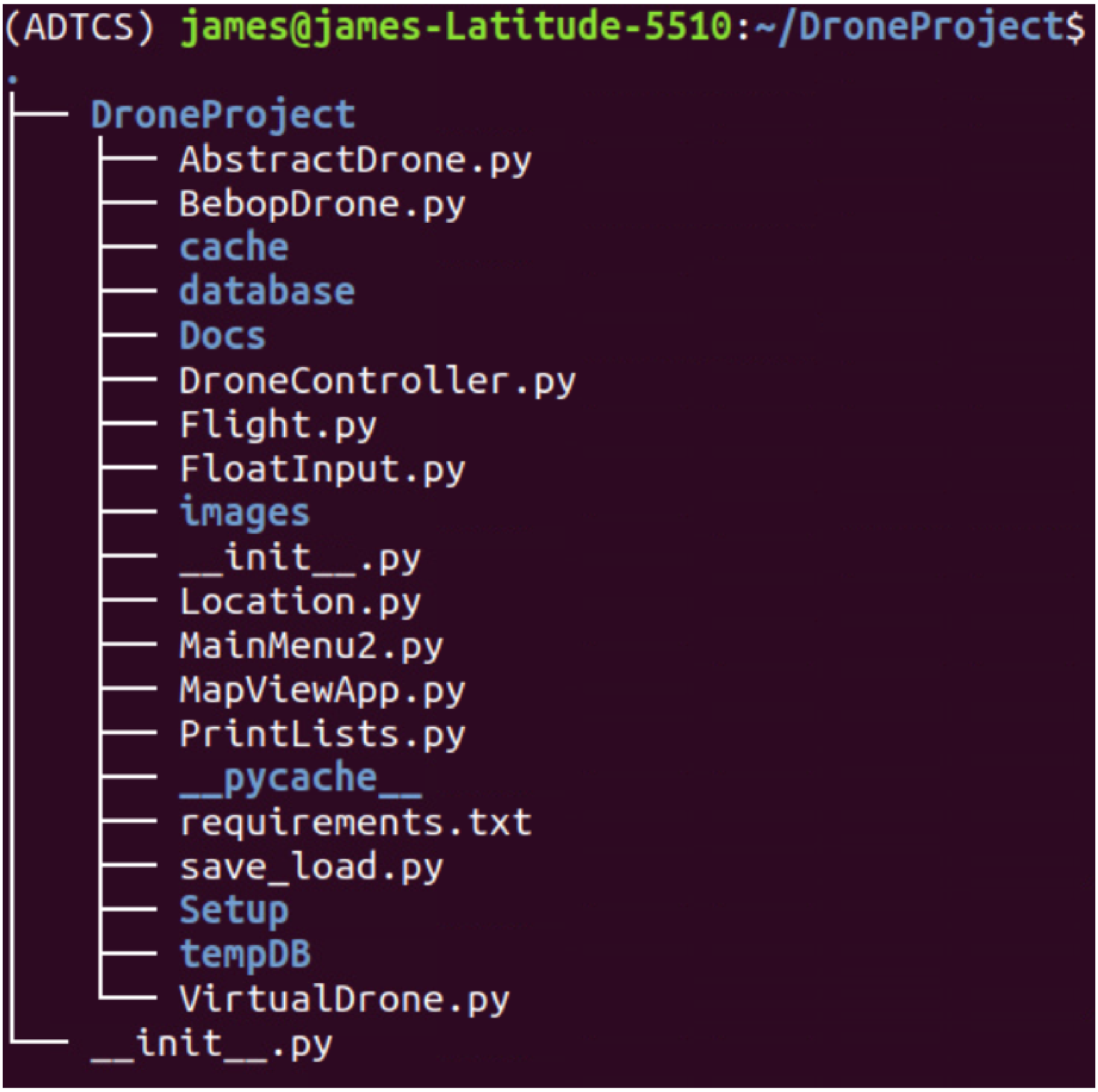

FlightList:List
MoveDrone()
drone:Drone
Operator
abort_flight_popup
loop
drone.moveto(lat,lon,alt)
click abort flight button
display popup
if flight_abort==True
buttonPressAbortFlight(selection)
flights[selection].setflightabort(True)
flight.setflightabort(True)
[lat,lon] = findnearestlocation(drone,locations:List)
drone.moveto(lat,lon,alt)

Figure 16 Abort flight sequence diagram

File Structure



```
(ADTCS) james@james-Latitude-5510:~/DroneProject$
.
├── DroneProject
│   ├── AbstractDrone.py
│   ├── BebopDrone.py
│   ├── cache
│   ├── database
│   ├── Docs
│   ├── DroneController.py
│   ├── Flight.py
│   ├── FloatInput.py
│   ├── images
│   ├── __init__.py
│   ├── Location.py
│   ├── MainMenu2.py
│   ├── MapViewApp.py
│   ├── PrintLists.py
│   ├── __pycache__
│   ├── requirements.txt
│   ├── save_load.py
│   ├── Setup
│   ├── tempDB
│   └── VirtualDrone.py
└── __init__.py
```

Figure 17 File structure

The project's .py files are stored in the DroneProject folder along with a database folder containing the pickle config files and an images folder containing all the marker images. The cache folder is created by Kivy MapView and contains cached images from the mapview tiles.