PharmaDrone: Final Report

From Clinic to Home, in one Click.

Chelsea Davis Zakiyah Harris Shaniyah Jones Nikitha Malladi

04.17.2024
CS Software Engineering

Abstract

Introduction

For many, medication is essential for ease of living. At the same time, due to disability, illness, or time constraints, there are thousands of people each day who are unable to pick up their medication themselves. And when loss of medication is a matter of life and death for many, we must aim to ensure that everyone should get their medication in a timely manner regardless of circumstance. The PharmaDrone project aimed to apply drone technology to provide a delivery service to clinics and pharmacies to deliver medication.

Planned vs End Product + Differences:

Initially, the plan was to create a two-part system, broken down into Delivery and Archive.

In the Delivery section of planning, In addition to being able to pilot the drone to a given location, there would be a Monitoring System in place. A log detailing each movement of the drone, and a grid-like image that would update given the location of the drone. The drone would be able to detect its own starting point, and be able to return to it anytime if needed in case of faulty coordinates or misplaced medication. Unfortunately, due to the limitations of the drone-a lack of GPS, inability to detect its own starting point, and the movement capabilities of the drone-a lot had to be restricted or downsized. The grid was scrapped together entirely, due to the drone having difficulty detecting its current location.Instead, we opted for a 6-digit sequence detailing each movement direction to execute one-by-one. It does deliver, but movement in a rather grid-like movement.

```
82⊖        loginButton.addActionListener(new ActionListener() {
83⊖            public void actionPerformed(ActionEvent e) {
84                String username = "PharmaDroneUserIDNumber";
85                String password = "TestingAccount12345";
86                // Get the user input
87                String enteredUsername = usernameTextfield.getText();
88                String enteredPassword = passwordTextfield.getText();
89
90                // Check if the entered username and password match the correct ones
91                if (enteredUsername.equals(username) && enteredPassword.equals(password)) {
92                    // If correct, redirect to another page
93                    dispose(); // Close the current window
94                    PharmaDroneUI mainWin = new PharmaDroneUI();
95                    mainWin.NewFrame();
96                } else {
97                    // If the credentials are incorrect, clear the user input fields
98                    usernameTextfield.setText("");
99                    passwordTextfield.setText("");
00                }
```

Figure 1: *The actionListener event code for the "Log in" button on the Login screen on startup.*

The Archive feature was that after each delivery, the operator using the system would be able to create a report of the finished delivery after completion. Said Archive would clear itself upon logout. The end result of this feature works as planned, with an instance of the Report class being created when you press "Enter" on the Make Report(an instance of the ArchiveInfoInsertion class) page and stored in a Archive object. There is an additional safety measure in that the system records the last inputted coordinates of a delivery itself.
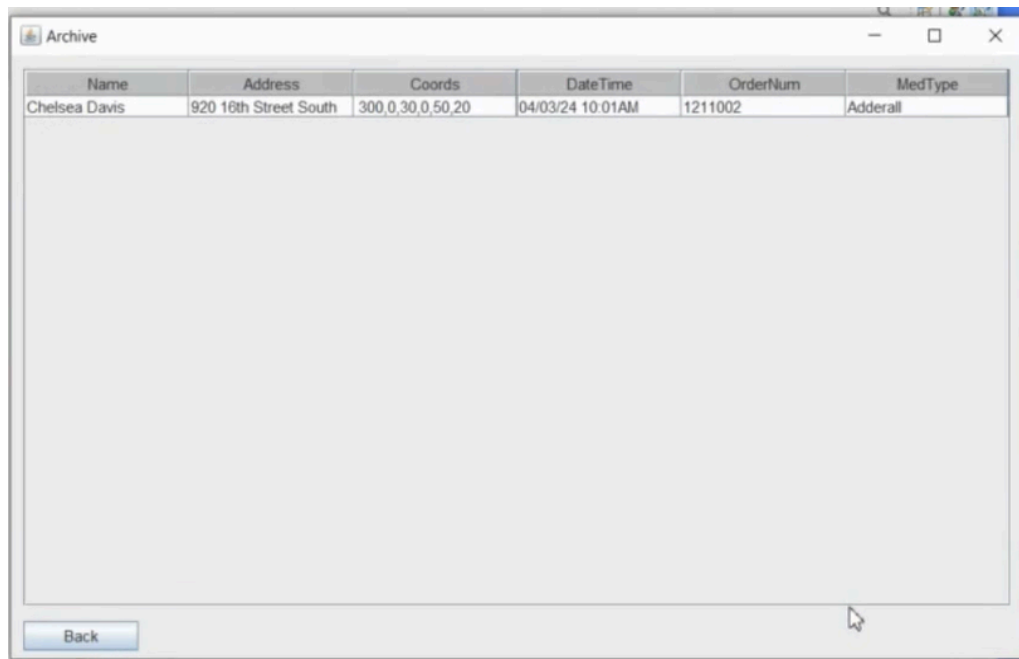
Figure 2: *An image of the Archive Page. The information of a Report object is displayed on screen.*

Figure 3: *The "Report Maker" page. Here is where after a finished delivery, information is inputted to create a Report object, which is stored in an Archive object, and its information is displayed in the Archive Page.*

Technical Documentation:

Software Design:

For the PharmaDrone, there are 3 main UI pages: the login(an instance of the AdminLogin class), the main page(an instance of the PharmaDroneUI class), and the Archive pages(instances of X and Y class) Upon startup, you are treated to the login page, and are required to enter a username and password. After typing them in and hitting enter, you are treated to the main page, where there are 4 buttons: Deliver, Return, Archive, and Logout. There is also a blank text field for a Log.
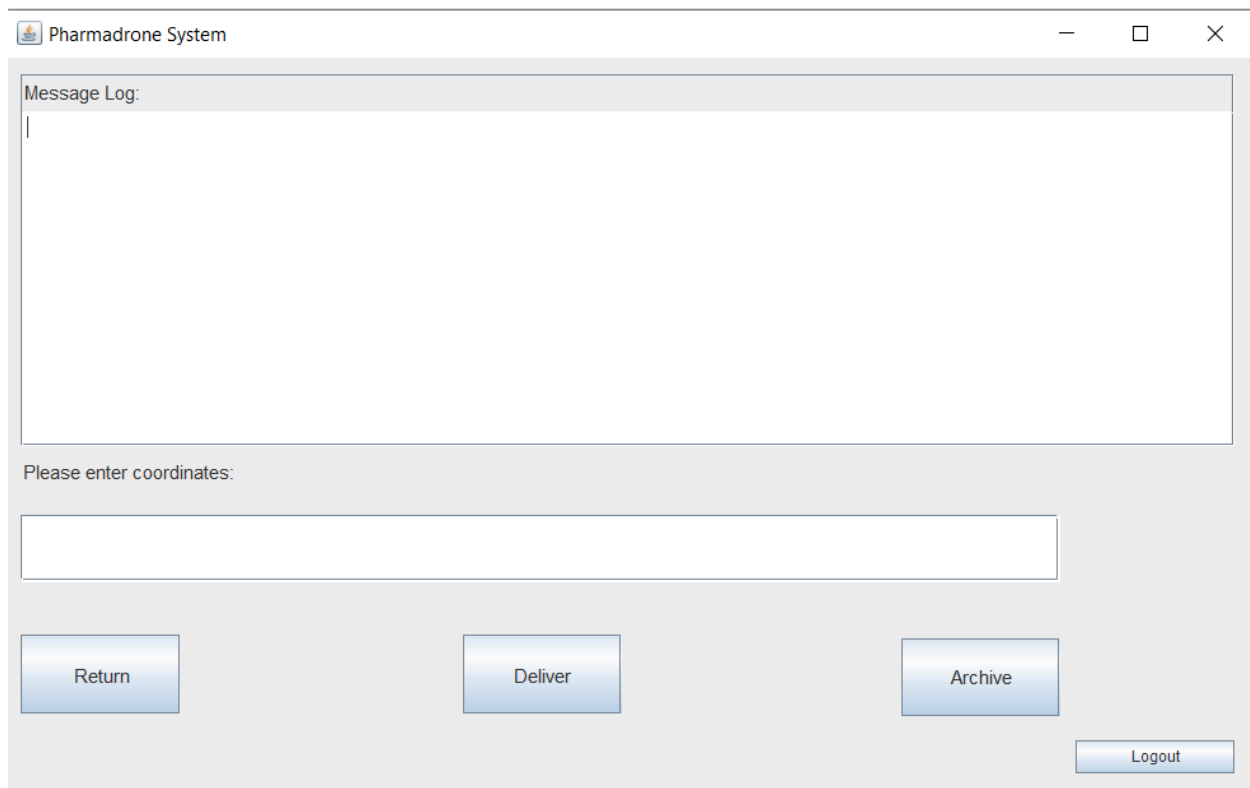

Figure 4: A fresh main page (instance of the PharmaDroneUI class) that the client sees upon logging in.

Logout takes you back to the login screen, and will erase anything in the Archive. Upon entering a six-digit string of coordinates separated by commas, the client will simply have to press the Deliver button, and the drone will take flight, land to deliver, and then fly back home. The drone's movements are documented on a Log for the client to survey the drone's behavior. If it is needed, the "Return" button can initiate an emergency return, where it will land, and then immediately head back to base. Once finished, the

5

Log will remind you to create a Report, so the client will hit the Archive page. It will prompt the client to input various information about the client. Once Enter is hit, a Report object is made, and is viewable in the Archive page. Hit back to return to the main page.
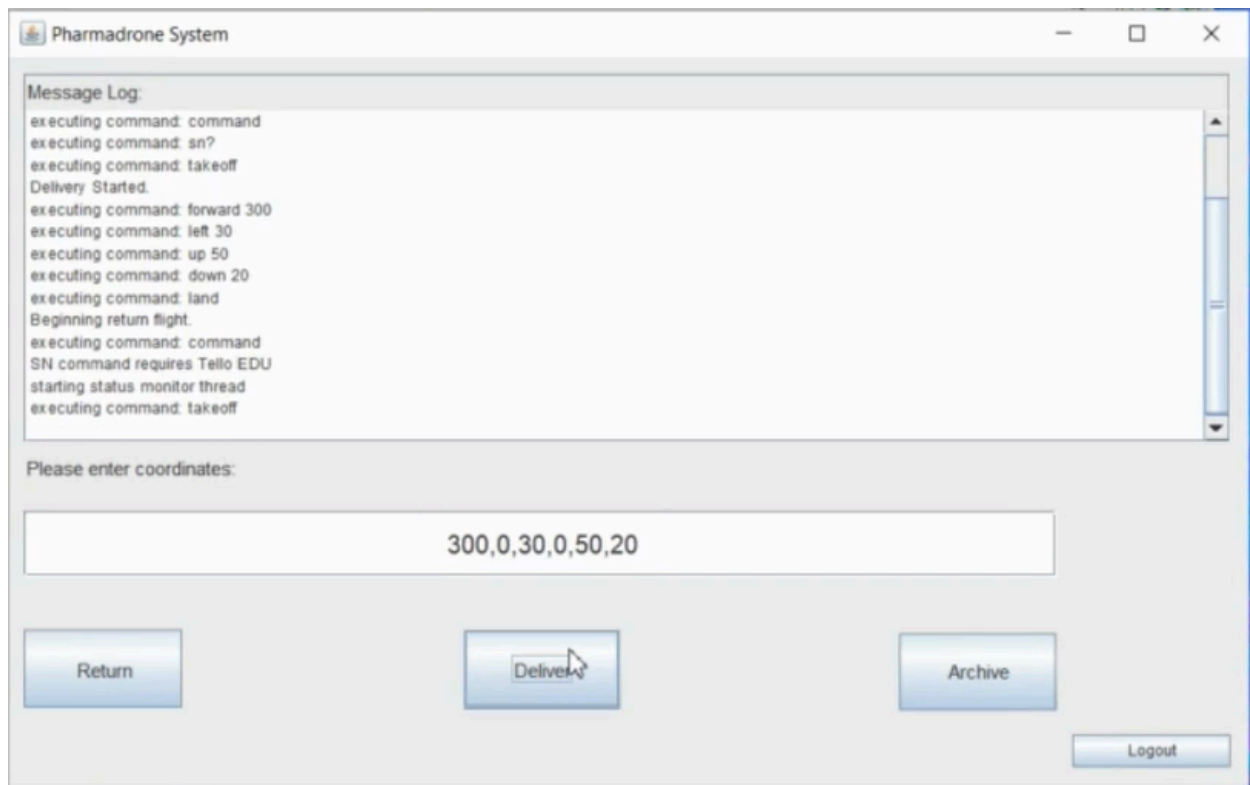

Figure 5: The log prints off information as the drone is on a delivery. Here, we're notified that the drone has just lifted back up after landing, preparing to make its return trip.

```
Message Log:
monitor thread constructor
status monitor thread start
executing command: command
executing command: sn?
executing command: takeoff
Delivery Started.
executing command: forward 300
Emergency Stop Called! Stopping.....
executing command: land
Beginning return flight.
executing command: command
SN command requires Tello EDU
starting status monitor thread
executing command: takeoff
```

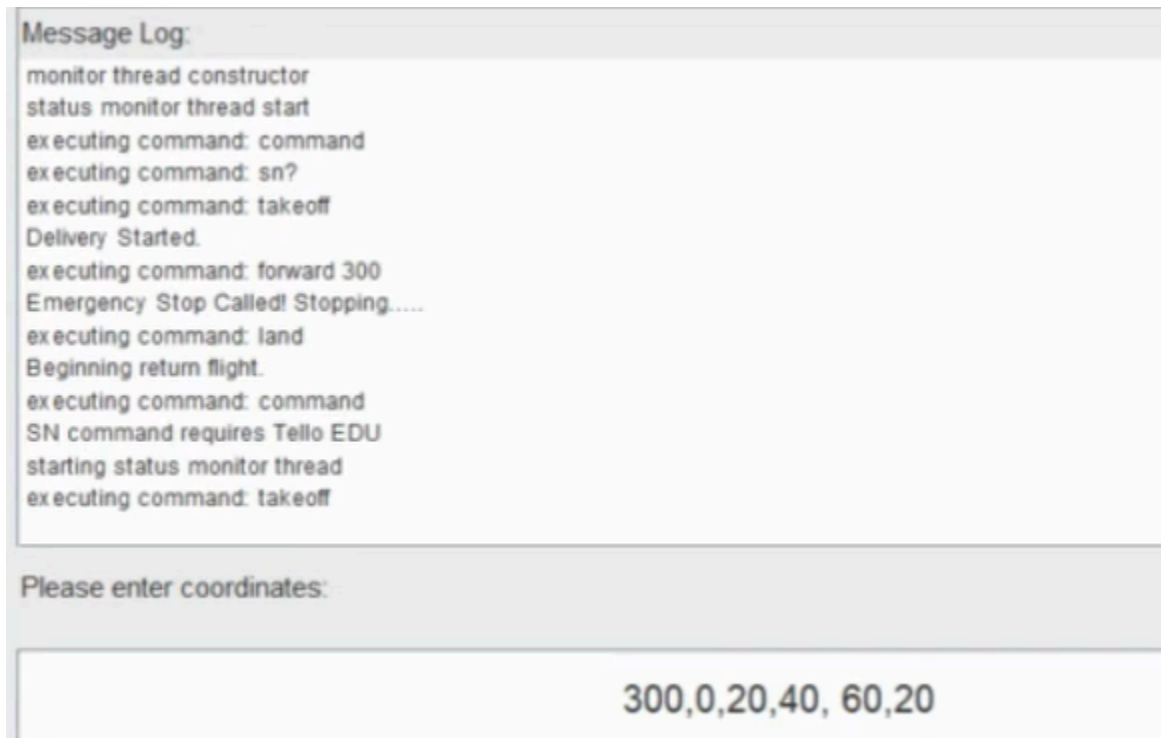Please enter coordinates:

300,0,20,40, 60,20

Figure 7: An example of the return feature in action. The drone does not execute the other 5 coordinates in favor of immediately landing, and is about to return home.

Tools Used:

The project was completed entirely in Java, and the tools we used include the Tello-SDK package for Java, Eclipse's WindowBuilder function, and Java Swing.

Assumptions and Dependencies:

A major assumption of the project in terms of applied use is that the customer has already ordered their medication through the required means first before it is delivered, and the user is relying on their knowledge of said order for the report. Another related assumption is that all medications delivered through the PharmaDrone system are approved to do so through the Food and Drug Administration's regulations regarding transportation and storage of medication. If neither assumptions are true, the client is using the Drone to engage in illegal activities, transporting medication in a dubious manner to an unapproved client. It is also assumed an alternative record (presumably either paper documentation, or screenshots of the Archive page) are made and stored

safely by the operator of the drone before they log out. It is also assumed that the drone is consistently taking off from the exact same spot with each delivery, and that coordinates from any given address are calculated by third-party means.

Evaluation

Test Plan

The test plan for this project was as followed

1. Do ample research on the legality and efficiency of drone usage in reference to medication.
2. Gaining an intimate understanding of the drone's limitations, strengths, and quirks to prepare for the design phase of the project.
3. Split into two groups: one concerning the UI, and Archive functions (UI Team) and one concerning the controlling the drone via tight-instruction (Drone Team)
4. Drone Team develops a way to manipulate the drone.
5. UI Team creates the pages that make up the GUI, as well as the Archive System to manage records of flights.
6. Test subprojects individually.
7. Initiate the integration process, where the Drone class is connected to the GUI and Archive system.
8. Determine aspects of the project that need to be cut for time, or elaborated upon.

Testing Conducted:

While aspects of the UI and the Archive could be comfortably tested in the UI Team's own homes, for monitoring the drone's flight path and ability to return correctly, drone testing was conducted at Blazer Hall's Movie Room, a large spacious area with a tile floor perfect for monitoring inconsistencies in travels.

Results of Testing:

The UI and Archive System worked flawlessly. As for the Drone's delivery functionality, it could get up, head to a location, and return not quite in the same position, but in the relatively the same vicinity, never more than half a foot away from its initial starting point. This is due to how the drone's takeoff often caused it to fly up at an angle, rather than up directly. For the return feature, it would finish its last executed movement before

stopping. Harmless in short travel, but could prove troublesome on long travels depending on the urgency level of stopping.

Future Work

The Projected Future:

In the projected future, those in remote or underdeveloped areas will have as much access to their medication as those who live in populated cities. Ultimately, PharmaDrone will ensure timely and efficient delivery of medications to those in need, transcending barriers of distance, disability, and time constraints if implemented on a nationwide scale. Even where weather disasters destroy pharmacies nearby, a person can still receive life-saving medicine by having the next nearest clinic deliver medication to a recovering zone. And if implemented on a worldwide scale, we could potentially send drones across borders to deliver humanitarian aid.

Next Steps:

The most important thing is we were to continue updating and improving the PharmaDrone System would be switching to a bigger, faster drone that can fly farther and has a higher carry load.A big feature that could be implemented in the future is the ability to assume control of the drone and fly it manually, switching the log to video footage in order to pilot it. In addition, another future feature would be pulling information from the order made online directly so that Report making is entirely autonomous and done as soon as you hit Archive. Lastly, exporting the day's Archive into a csv file upon Logout.