In implementing my solution, I wanted to keep in mind a way to test my work during the process of its development. For this purpose, I made a UI that I had intended to interact with my control algorithm to provide a visual representation of the algorithm. Using PyQt5 I made My_App.py to house my UI for my algorithm.

Going into my solution to this problem I had a separate Plane class to contain the information that an individual plane would have including its position, heading, and speed. Within this class, there are functions to return the plane's current position, receive updates to update its heading, and update its position.

Within my main Control class I have worked on creating a system where new planes that spawn will be passed on information on whether to land if there are runways available or to go to a designated holding location in a queue to land. If we consider that planes are currently en route to land, once a plane has landed the first plane in the holding queue will proceed to calculate its landing trajectory and fly toward the runway. This will then update the holding queues and advance this plane to be in the landing queue to prevent other planes from landing ahead of it. The information my Control class uses would be the locations of the runways, the dimensions of each of them, a list of planes currently landing, and a list for planes currently in a holding pattern.

To decide where the planes would go I went with the case of spreading them out near the runway in a circular fashion such that any planes coming into the holding queue later would not interfere with the landing sequence of the preceding planes. Once a circular ring has been filled up, a new ring will be used 1km in radius further away than the previous ring. I had intended to implement a check for overlapping holding circles within the airspace between runaways as these would cause possible collisions to occur. By having holding patterns at each runway it would allow for each runway to handle its own set of planes independently and remove the reliance on directing planes to specific runways.

The holding locations for the planes were designed to take into account how many planes were already holding and place them in a holding pattern next to them in a ring. I have yet to implement the check to have new planes fill in spots for planes that have gone to land. For calculating the landing trajectory of the next plane in line I have simply used the direct distance to the closest runway and pass on the runway location and the heading to it for the plane.

I did not have time to implement a check that the flight trajectory of a plane going to land or going to its holding pattern location is going to intersect the holding pattern of another plane. I also did not implement the change in the plane's heading to adjust to the runway which would need to be adjusted in the Plane class. Lastly, I have yet to implement the UI connection to my control and plane classes.

For this challenge, I found it difficult to imagine how I might receive a continual stream of plane data while having to insert the data into my algorithm itself. The challenge of having developed a way to test the algorithm using a UI was also new to me which led to a large portion of my time being spent on this aspect as I valued the ability to see how my algorithm would work.