- Description of system

My lift control system overcomes the unnecessary movement of going to the top floor and bottom floor in order to move the opposite direction. If the direction is up, the elevator will check if there are any customers above that of the elevator that have called. If there are customers above who want to go up, then the elevator will continue upward. If the elevator is full or there are no more people who are heading up the floors, the elevator will change direction, depending on its original direction.
This eliminates moved floors that the mechanical system takes in order to change direction. The proposal is therefore more efficient, especially if the customers are all close together. For example, if in the afternoon customers are all heading to the low to mid floors, the mechanical system must travel to the top most floor in order to change direction.
If the number of floors is high, then the mechanical system will waste time continuing upward.
The improved system however, once the customers have been dropped off, checks the customer list, will return to the lower floors to pick up the customers, and will complete the task much faster and efficiently.

When the number of customers waiting drops below a certain point the priority list is made. The elevator will select the floor with the most people to ensure that they are not kept waiting for too long, and maximise the capacity of the elevator.
The elevator will travel to the floor and pick the customers at the destination, then proceed to drop off and pick anyone else heading in the desired direction (as long as the elevator isn't already full). Once the elevator is empty, the next target floor will be selected.

This is repeated until the customer list is empty.

- All data structures and algorithms used in my implementation

Python classes:
Objects have member variables and have behaviour associated with them. In my implementation, I used a “variables” class, where the user will input the data for the lift system: the number of people to be generated, the number of floors and the elevator capacity.
This data is then used throughout the system.

This class will then call the “People” class, and generate an object as each unique person. Each person has a floor they are waiting on and a destination floor. Each customer object is represented as an ID.
Each person object has a value for how long it has waited until it is picked up, and how many floors it has moved in the elevator until it has reached its destination. This data is collectively used to compute the effectiveness of the system against the mechanical system.

The Gui “Frame” class is called for the functionality and display of the data and movement of the elevator components. In the initialisation the floors are generated and then the customers are generated in their identified waiting positions on the floors.

The “Elevator” class is called and keeps track of the current people in the elevator, its current floor, its destination and its capacity size. The Elevator class will call the movement function in the Frame class when the elevator movement is called. The elevator will then move up/down in the gui.

The “Building” class has the mechanical system and the proposed system that can be selected in the variable class. The Building class will call the frame class and the Elevator class within the logic and functions that run.

Object Array lists are used throughout the program, in order to identify the objects and values within, obtain system data and parse values used throughout the program and functionality.

Search algorithm - linear search

- This algorithm searches an array for object with correct matching data

Limits search algorithm - linear search

- This algorithm sets the upper and lower limits of the elevator, by checking the lift array to see the customers destination floor.
- And if the elevator doesn't need to pick anyone outside the limits (as it's full or not going in the direction) it will set the limit and change direction if it reaches that limit.

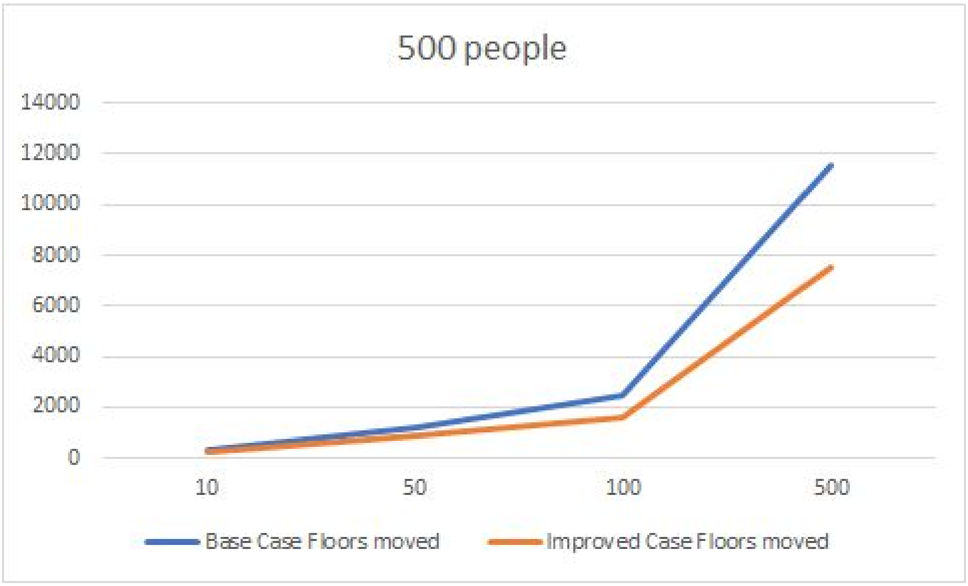
Priority check algorithm - linear search

- The algorithm checks the current floors to find the floor with the most people. The elevator will then move toward that target floor, and will maximise capacity in the elevator to ensure the customers are not waiting for too long.
- Once at the target floor customers have enter the elevator, which will proceed to drop the customers off at their destinations
- Will then check the next target floor with the most people until there is no one to pick up

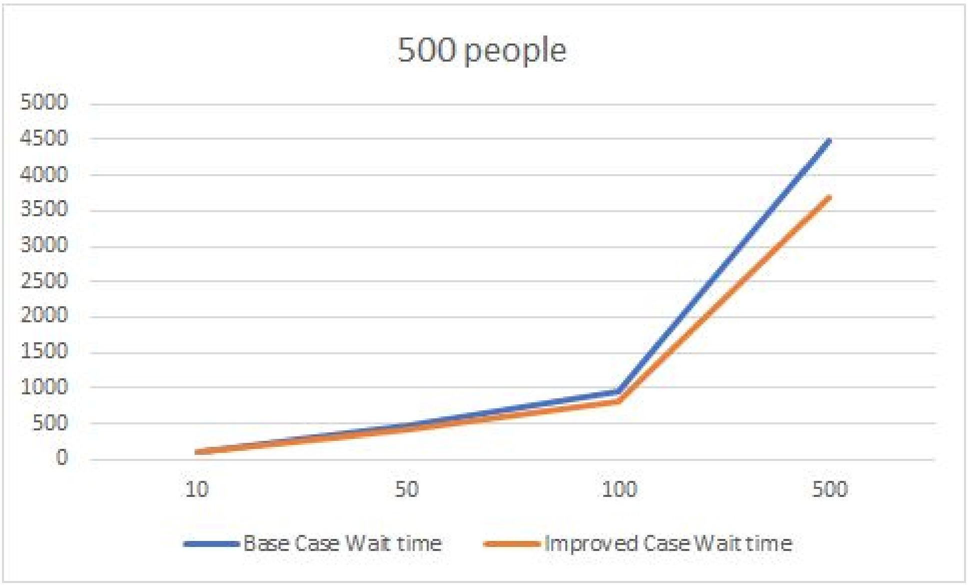
Performance analysis

In the testing of the system, I varied the number of people and floors to demonstrate the comparison against the mechanical system. For lower numbers of people and floors, the difference is much smaller.
Increasing the number of people and floors to a larger set shows the results more clearly.
In each test, 3 lots of simulations were processed to achieve an average

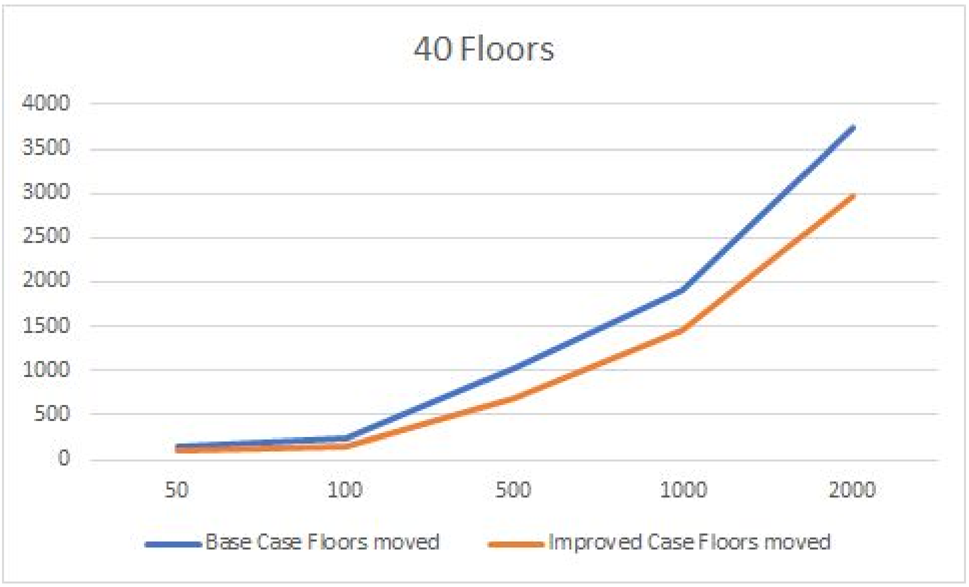
Using Random distribution:
Total floors moved by the elevator for each number of floors specified.
Constant elevator capacity = 12
Constant number of people = 500



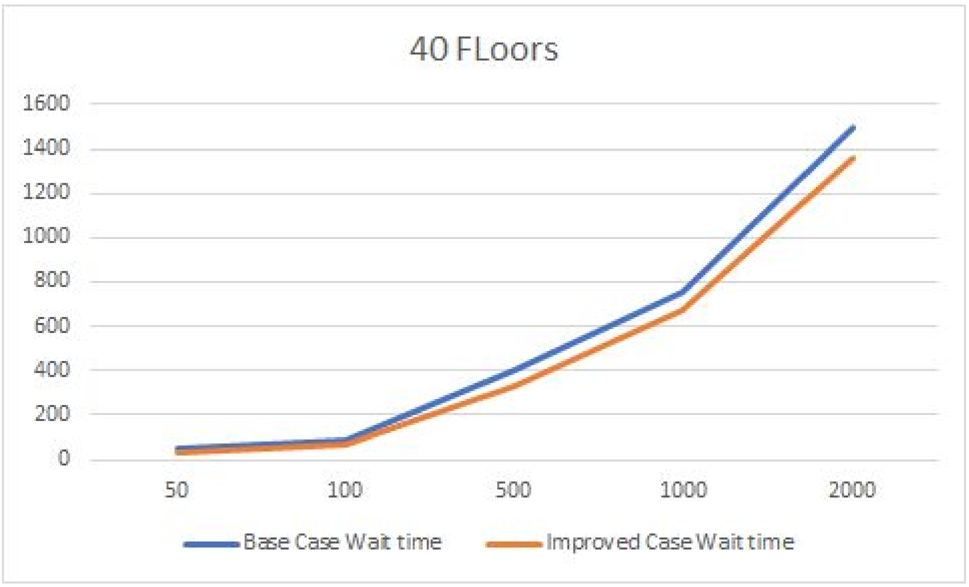
Total average wait time for elevator against increasing number of floors



Total Floors moved for increasing people population
Constant floor number = 40
Constant elevator capacity = 12

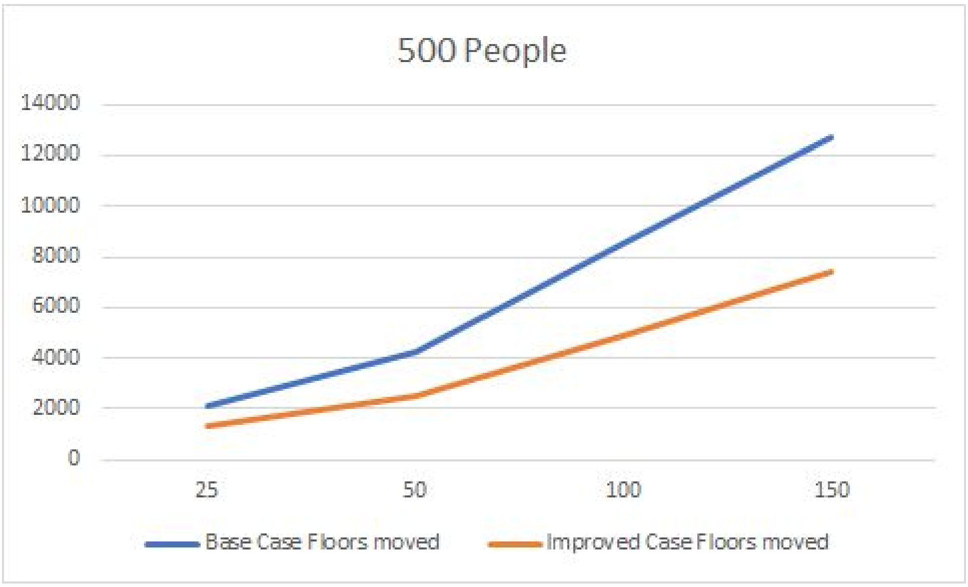


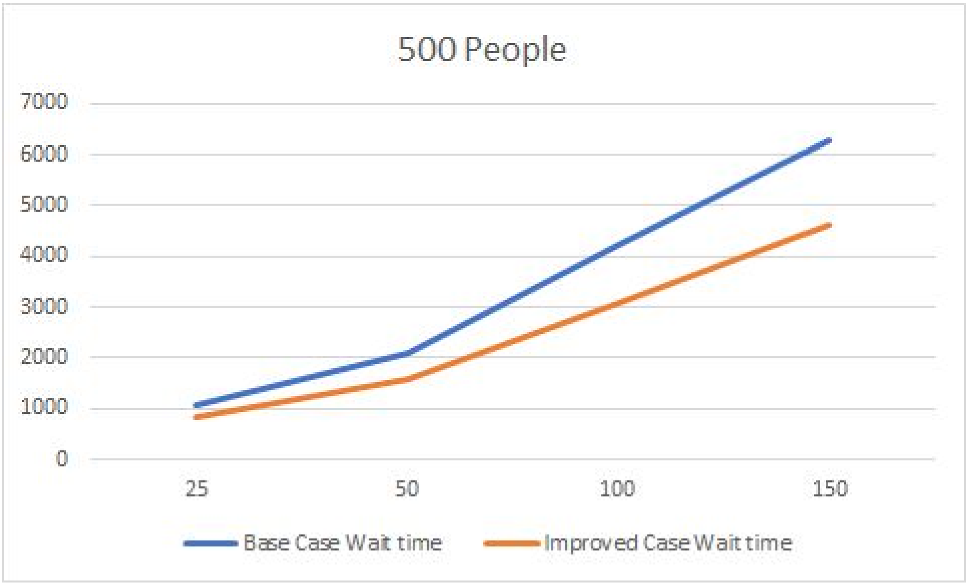
Total average wait time for increasing people population



In this test, people are starting on the top floors(above the mid floor), and being dropped off at the lower floors(below the mid floor).

Constant people = 500
Constant elevator capacity = 12





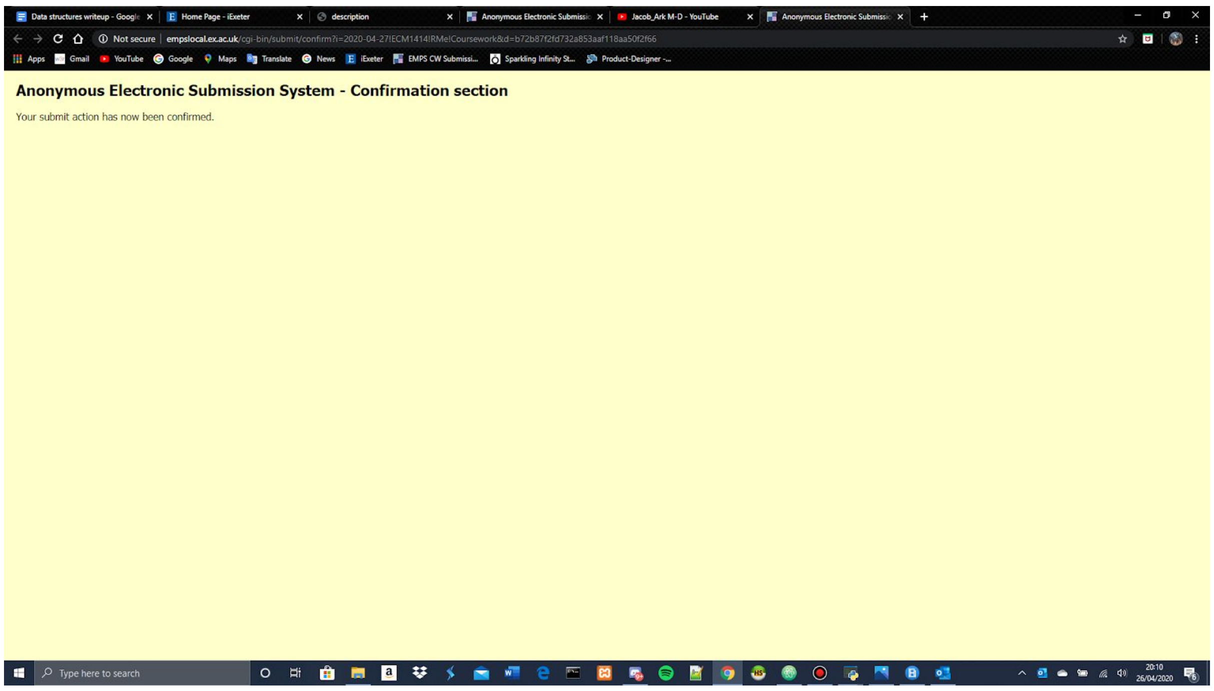
Weekly log of progress.

Date of development	Description
3rd - 9th FEB	Designing the initialization of the lift elements. What variable will be needed during development, and what data will be parsed through. Design of People class, Building class, and gui class. Generation array list of people. People have a waiting floor that they can be picked up from and a predefined destination floor that the person will be dropped off from. No gui implementation, setup of development in console window. Floor counter incrementing to max floor number (top) and returning loop will increment. Array check if person is on the current elevator floor. Person added to lift array. Lift array checks if the person's destination floor is reached. Hard coded lift person limit, floors and amount of people.
10th - 16th FEB	Gui implementation. Generation of lift, and movement in accordance to the current floor. Generation of floors, starting at input coordinates. Programming the movement of the life, which will move in between the floors.
17th - 23rd FEB	Generation of customers on the floor. Development of dropping the customers off at the floor number. Initialise gui will have a predefined array where the people will be shown waiting on the floor. When picked up the person is removed from the floor and “enters” the lift (into the lift array). In gui the person is removed at their ID current coordinate as they enter the lift. When the person exits the lift at the destination floor, the person is “dropped off”, and they are shown in the gui to exit at the floor.
24th - 1st MAR	Gui refinement. Bug fixing with array errors. Refinement of the mechanical system. Gui input selection development,, to input the number of customers, floors and elevator capacity. Base case selection, and data inputs are processed and created.
2nd - 8th MAR	Bug fixing. Development of the algorithm. Implementing the selection of the secondary system selection in the input gui. Validation of the inputs as whole integers only.

9th -15th MAR	Fixing array errors. Object removal of all arrays in class fixed. Gui improvement, display colours of the elevator when full, when a customer gets on and drops off.
16th - 22nd MAR	Development of algorithm
23rd - 29th MAR	Development of algorithm
6th - 12th APR	Data distribution of people on floors selection: Random, Normal, Exponential (up), Exponential (down). Generation so that people are distributed accordingly.

Video recording demo

https://youtu.be/v5o7JSi_6jc



Anonymous Electronic Submission System - Confirmation section

Your submit action has now been confirmed.