Home Assignment Report - Guy Green - NeuroBrave

Part 1:
For this part, I've programmed a python code that reads a json file that stores coordinates and a sender email and a receiver email addresses (using a dummy email address for example). The program monitors the mouse's location and sends an email stating that the cursor has entered or exited the box's coordinates.
I used the "pyautogui" library to monitor the mouse's location, and the "smtplib" library to automatically send emails(Gmail by code's default).
The json format I assumed is:

```
{ email :
    { sender :  ,
      receiver: ,
      Password:
    },
 rect :
   { x1 : ,
     y1 : ,
     x2 : ,
     y2 :
   }
}
```

Part 2:
In this part, I've created in Python a simple server and a sender-client, using UDP protocol. The decision to use UDP over TCP was made after consideration of the application's requirements. While TCP provides reliability and guaranteed delivery of data, UDP offers greater speed and efficiency. Given that the task involves streaming mouse positions and CPU loads continuously, occasional packet loss or out-of-order delivery is acceptable. Therefore, UDP was chosen to optimise speed and minimise latency.
I also added the functionality of sending CPU load (using "psutil" library) in addition to the mouse positions.
The sender-client is built with two threads, one thread for generating the data: mouse positions and CPU loads, and the second thread for sending the data to the server, both threads respect the parameters received.
The server is also built with two threads, one thread for receiving initial messages and then receiving data from the sender client, and the second thread for sending the data to the receiver client.
In order to determine the minimum generate_period for a system, one must consider the system's CPU abilities such as speed and memory and network bandwidth. There is a tradeoff between the generate_period of the mouse, and the CPU load, and since we record the CPU load, we can implement a client that changes the generate_period of the mouse as a function of the CPU load readings, such that when the CPU load is low we can decrease the generate_period, and increase it when the CPU load is getting higher.
For the send_period of the mouse, one might begin with a small value to experience continuous stream, and a higher value of send_period for the CPU load, as it is less necessary from the mouse data. We can also make sure that the send_periods intervals won't overlap, such that they will not try to send at the same time.

In conclusion, the UDP-based server and sender-client application provide a lightweight and efficient solution for streaming mouse positions and CPU loads over a network. Further improvements in the code might include as stated before, dynamic adjustments of the parameters. In hindsight, I should've asked more questions such as how the json is supposed to look like, what emails to use, or how the server is supposed to receive and recognise the clients. These insights highlight the importance of communication and exploration of requirements during the development process.