Server file:

Importing necessary libraries, setting up variables for rabbitmq and defining a function for publishing the activity log using rabbitmq.

A screen shot of a computer code

Description automatically generated

Hardcoding the rough employee database

A screenshot of a computer program

Description automatically generated

Creating a data lock to allow safe multi-threaded access to the employee database and defining the handle\_client function which will run each time a client connects to the server.

The server waits for the data from the client which will be the employee id, then it locks the employee database and checks if the employee exists within it. Sending back an appropriate response.

A computer screen shot of a code

Description automatically generated

Then the server waits for the client to input the first option and the second option which is resembled by the letters.A computer screen shot of a program code

Description automatically generated

Depending on the option combination and using the data lock, the program grabs the data that was requested by the client.

A computer screen shot of a program

Description automatically generated

And at the end of any final data to the client, the server publishes the activity log to the rabbitmq client and resumes listening to the socket.

A screen shot of a computer program

Description automatically generated

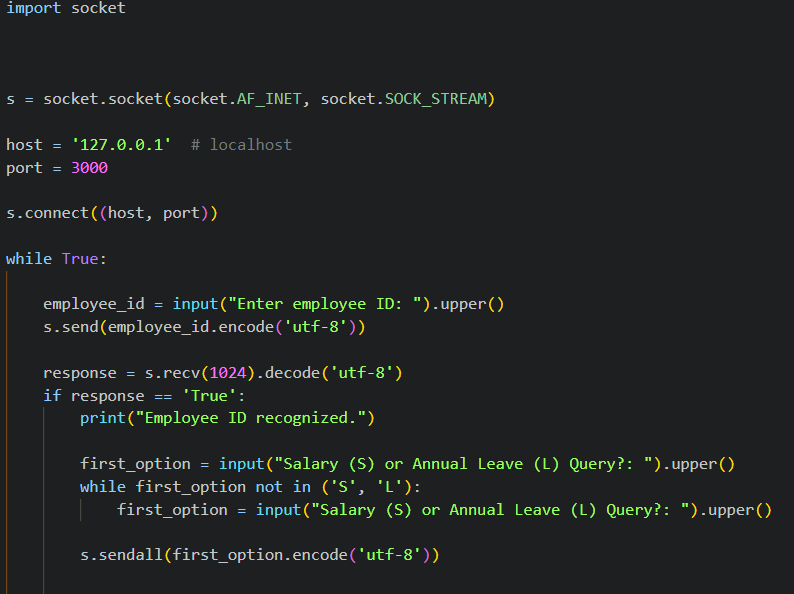
Declaring the start code, we create a socket object, declare the host and port, bind the socket to them and set the socket to listen to connections.

We loop over the socket accepting connections, and for each connection it gets it creates a new thread, that will instantly run our previously defined handle\_client function.

Client file (client.py)

We start the while loop by asking our client to enter the employee id, the result of that will be sent to the server, and the response being “True” or “False” will be brought back to the client depending on if the employee exists in the database.

If the employee does exist, the program follows with options to choose from and sends the choices to the server.



A computer screen shot of a program

Description automatically generated

The second option is requested after the first option is recognized, then it is sent to the server, depending on the second option result, the program might additionally ask for the year. After sending the data to the server, the client awaits the response in the form of accurate data and prints out the results.

A computer screen shot of code

Description automatically generated

A screen shot of a computer program

Description automatically generated

At the end of the program, the user is asked to choose if they would like to continue or exit, upon exiting, the socket is closed, and the program quits.

Rabbit\_client

A screen shot of a computer program

Description automatically generated

Importing the pika dependency, declaring necessary variables and creating a callback function that will print out a suitable message whenever it receives a message from the server.

Then defining a function to accept the published activity logs that was sent from the server whenever it is sent.

Lastly,  
dockerfile:  
A screenshot of a computer

Description automatically generated

And requirements.txt:

