Chatting Application Report

The protype I worked on was the chatting application. For the creation, I first installed Docker on my laptop that uses macOS and started a RabbitMQ container. I then chose VSCode as my chosen terminal and Python as my chosen programming language. As I am new to Docker and RabbitMQ, I had to research how to use it with Python and I was able to find various guides to help me get started. At first, I encountered errors when attempting to install the Pika module, I then learned that I had to create a virtual environment for it to work. Upon creation there were still errors with the pika module due to the wrong interpreter path being selected. From my research I was able to gain an idea of how I was going to code my chatting application. Upon completion of the code for the command line chatting application, there were issues connecting to the RabbitMQ server. Various methods to attempt to fix the error were undertaken. The first method attempted was double checking the code. The second method was checking the IP address, port and the credentials. From there code alterations became the third method and the Asyncore module was trialed. Another code alteration attempted was adding a line that specified the credentials and changed ‘localhost’ to my IP address. All these methods proved to be unsuccessful as there was still a connection error. Finally, the decision was made to attempt to make the code run without RabbitMQ. I was able to then create a chatting application that runs well using a command line terminal. On a terminal one can type python server.py and python user.py (python3 for macOS), all that is needed as an IP address that the user will be prompted for. Once connected everyone in the chatroom will be able to see each other’s messages and know who sent them.