Written by Dylan Coon.

Assessment 1 Report on Chat Application

1. Demonstrate Knowledge and Understanding of Functional Middleware

The chat application uses RabbitMQ as its middleware for message brokering. RabbitMQ is a robust, flexible messaging platform that facilitates asynchronous communication between distributed systems. In this project, RabbitMQ is utilized to manage the communication between multiple chat clients by setting up an exchange of type topic, which enables the clients to publish and subscribe to messages within a specific chat room.

Key functionalities of RabbitMQ in the application:

- **Connection Management**: Establishes a connection to the RabbitMQ server using a ConnectionFactory.
- **Exchange and Queue Declaration**: Declares an exchange of type topic and binds unique queues for each client to this exchange.
- **Message Publishing and Consumption**: Publishes messages to the exchange and consumes messages from the queue, ensuring that each client receives the messages intended for the chat room.

2. Knowledge and Understanding of Compiled Code

The chat application is written in C# and utilizes Windows Forms for its user interface. The project is structured within a namespace PBT_205_A1, with a class ChatApp that inherits from Form.

Key components of the compiled code:

- **UI Initialization**: The InitializeComponent method sets up the form controls, including text boxes, list boxes, and buttons, and configures their properties.
- **RabbitMQ Initialization**: The InitRabbitMQ method establishes a connection and sets up the necessary exchange and queue for message communication.
- **Message Handling**: The application defines methods for sending (SendMessage) and receiving (StartChatroom) messages, utilizing RabbitMQ's publishing and consumption mechanisms.

3. Knowledge and Understanding of Main Topics, Arguments Implemented, and Outputs

The main topics covered in the chat application include:

- **Message Queueing**: Implementation of a message queue using RabbitMQ, ensuring reliable message delivery between clients.
- **Event-Driven Architecture**: Use of event handlers (EventingBasicConsumer) to process incoming messages asynchronously.
- **User Interface Design**: Customization of the Windows Forms interface to display chat messages and manage user inputs.

Arguments implemented:

- **ConnectionFactory Configuration**: Setting up the RabbitMQ ConnectionFactory with the host name, username, and password.

Written by Dylan Coon.

- **Exchange and Queue Binding**: Configuring the exchange and queue with specific routing keys to manage chat room communications.
- **Message Encoding**: Encoding messages in UTF-8 format for transmission and decoding them upon reception.

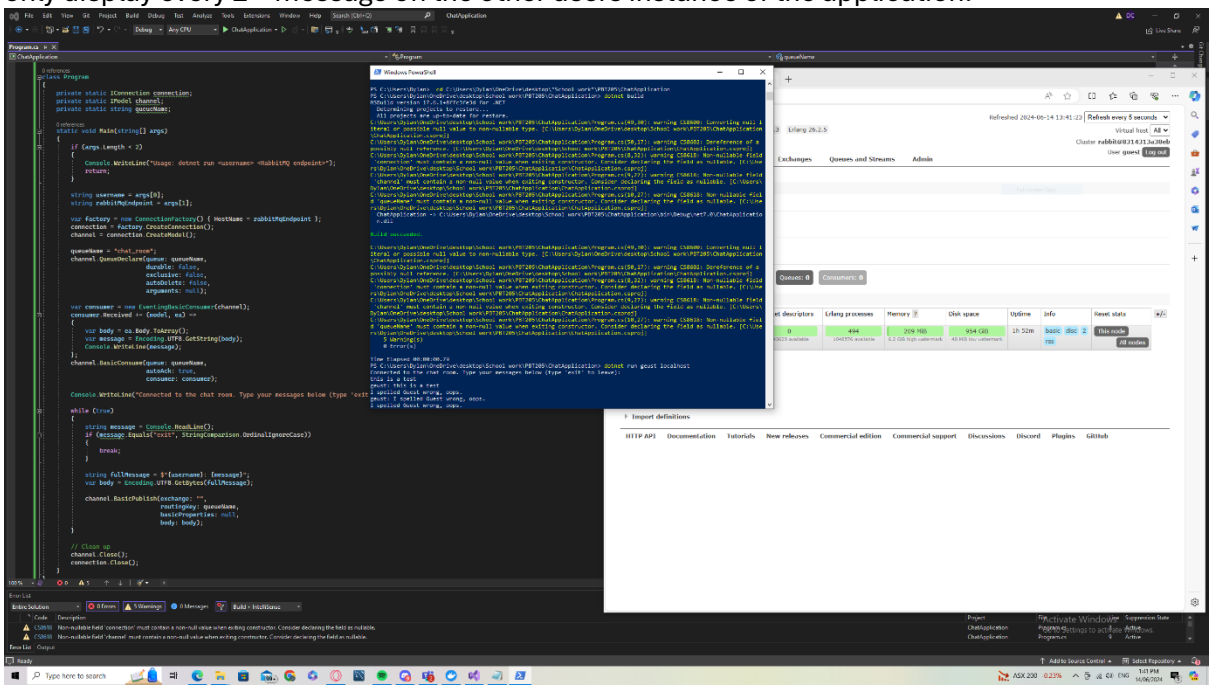
Outputs:

- **Chat Messages**: Display of sent and received chat messages in the chatListBox.
- **User Status Updates**: Display of online and offline users in the usersTextBox.

4. Teamwork and Collaboration Between Members

The original version of the application was developed by **Dylan Coon** and ran in Windows PowerShell.

In this original version multiple people were able to communicate across a RabbitMQ server, however we noticed an issue where in the same user sends multiple messages in a row it would only display every 2nd message on the other users instance of the application.



After sending this version of the chatting application to the rest of the team, **Dalton Christopher** created a GitHub repository and altered the code to fit within a Windows form application to better suit users, after some discussion regarding the issue with the messages we can to the conclusion that the message queue was causing the problem.

We came to the conclusion that as long a queue name isn't unique this line of code only ran once, so we made it that users have unique queue names.

```
channel.BasicConsume(queue: queueName, autoAck: true, consumer: consumer);
```

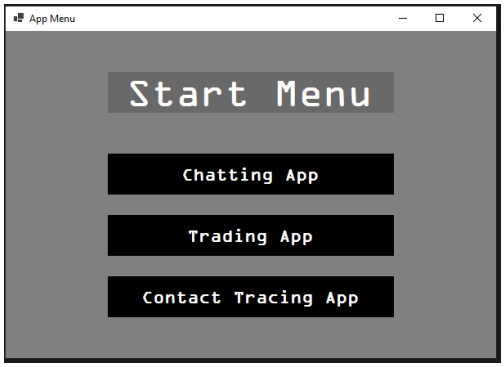
Written by Dylan Coon.

5. Windows Forms implementation.

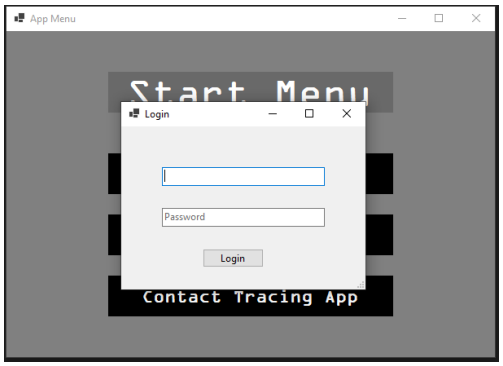
The chatting application is displayed in a windows forms format for ease of use as well as allowing the option to test the other prototypes through the menu, originally the application ran in Windows PowerShell but was moved to a windows forms application.

Below I have attached some images demonstrating the application in Windows Forms.

Menu:



Login:



Application: