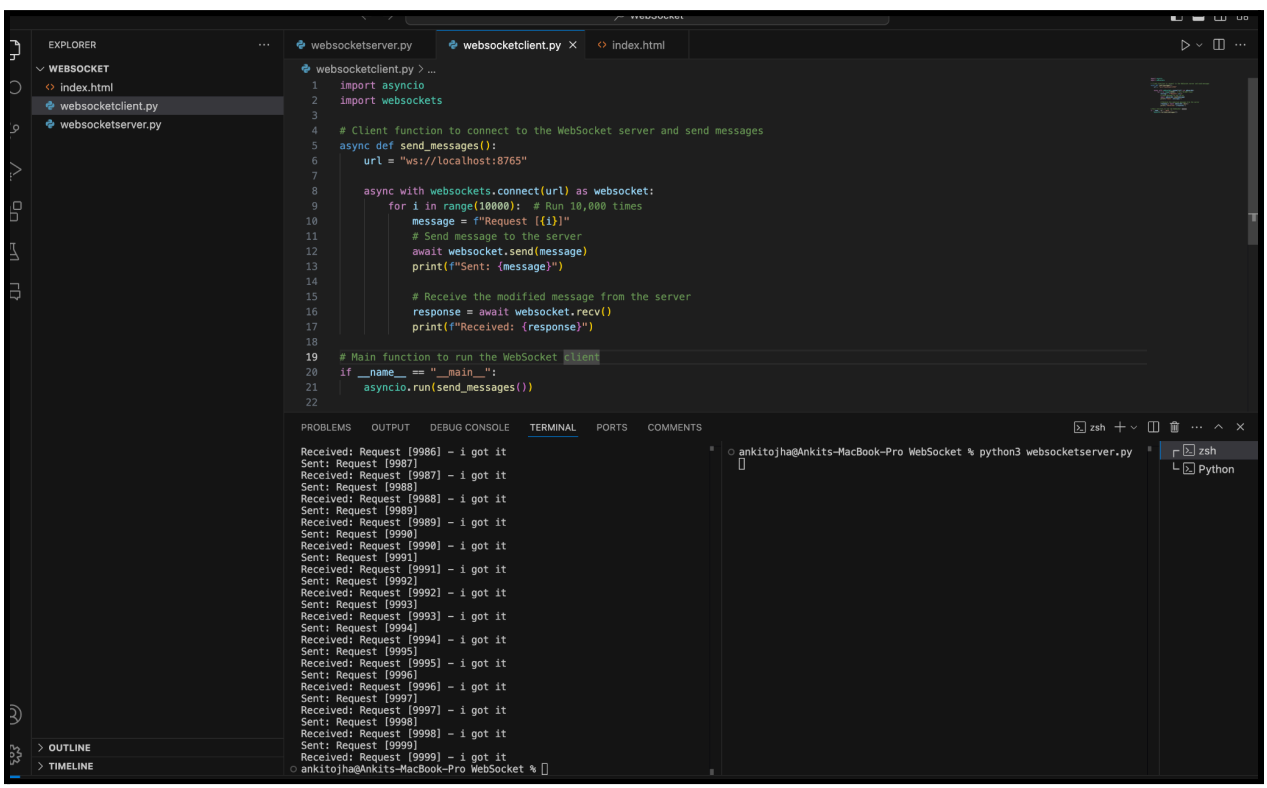
WebSocket Client-Server Assignment Report

1. Objective

The objective of this assignment was to build a full-duplex communication system using WebSockets, allowing real-time communication between a client (browser) and a server. The system should allow a user to send a message from the client, modify the message on the server, and return the modified message to the client without dropping any messages. Additionally, a user interface (UI) was required to allow message input and display the server's response.

2. ScreenShot



127.0.0.1:5500/index.html

WebSocket Client

hi my name is ankit Ojh | Send 10,000 Messages

hi my name is ankit Ojha [0] - i got it
hi my name is ankit Ojha [1] - i got it
hi my name is ankit Ojha [2] - i got it
hi my name is ankit Ojha [3] - i got it
hi my name is ankit Ojha [4] - i got it
hi my name is ankit Ojha [5] - i got it
hi my name is ankit Ojha [6] - i got it
hi my name is ankit Ojha [7] - i got it
hi my name is ankit Ojha [8] - i got it
hi my name is ankit Ojha [9] - i got it
hi my name is ankit Ojha [10] - i got it
hi my name is ankit Ojha [11] - i got it
hi my name is ankit Ojha [12] - i got it

127.0.0.1:5500/index.html

WebSocket Client

hi my name is ankit Ojh | Send 10,000 Messages

hi my name is ankit Ojha [9987] - i got it
hi my name is ankit Ojha [9988] - i got it
hi my name is ankit Ojha [9989] - i got it
hi my name is ankit Ojha [9990] - i got it
hi my name is ankit Ojha [9991] - i got it
hi my name is ankit Ojha [9992] - i got it
hi my name is ankit Ojha [9993] - i got it
hi my name is ankit Ojha [9994] - i got it
hi my name is ankit Ojha [9995] - i got it
hi my name is ankit Ojha [9996] - i got it
hi my name is ankit Ojha [9997] - i got it
hi my name is ankit Ojha [9998] - i got it
hi my name is ankit Ojha [9999] - i got it

2. Technologies Used

- **Backend**: Python with websockets library for WebSocket server implementation.
- **Frontend**: HTML, JavaScript for WebSocket client functionality and UI.
- **WebSocket Library**: websockets (installed via pip).
- **Browser**: Used the WebSocket API to handle client-server communication in real-time.
- **Tools**: Python, WebSocket API, JavaScript, HTML5, CSS3 (basic).

3. WebSocket Functionality Overview

- **WebSocket** is a protocol that provides full-duplex communication over a single, long-lived connection. It allows real-time communication between a web server and a browser or any client application.

Key Features Implemented:

- **Full-Duplex Communication**: Both the server and the client can send and receive messages simultaneously without reopening the connection each time.
- **Message Modification**: The server appends a random number to each client-sent message and returns it to the client.
- **Real-Time UI Updates**: The client displays server responses in real-time using the WebSocket onmessage event.

4. Implementation Details

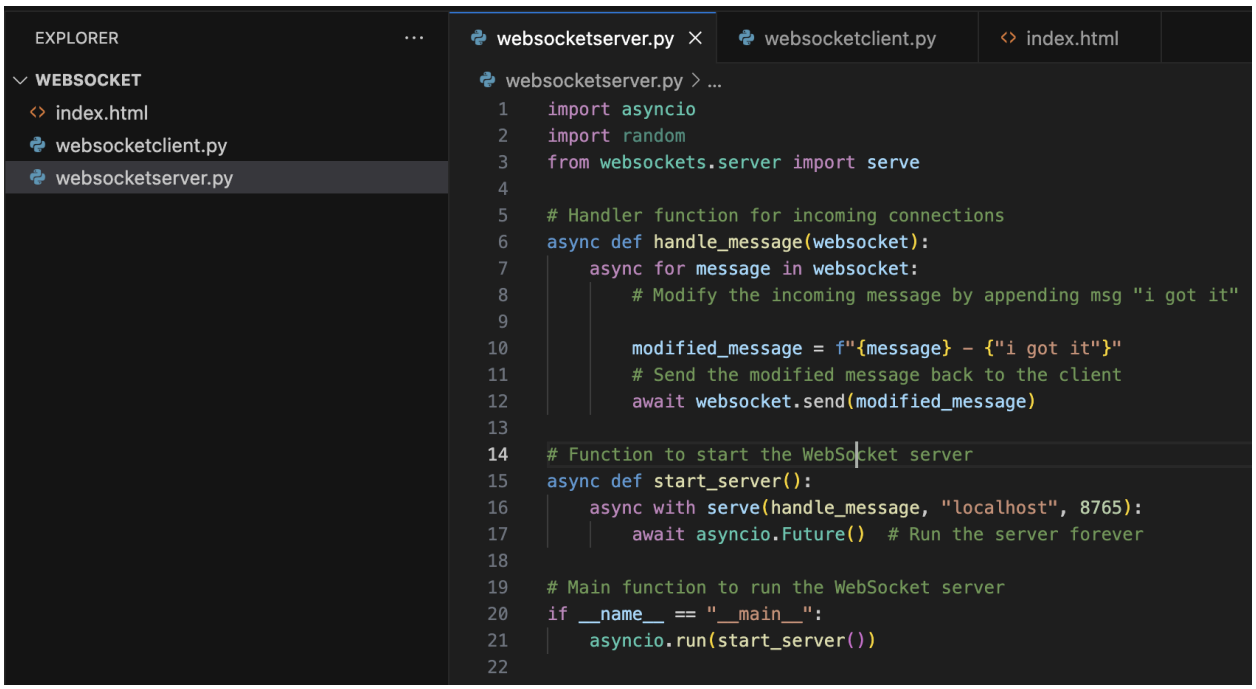
4.1 WebSocket Server (Python)

The WebSocket server is implemented in Python using the websockets library. It listens for messages from the client, modifies the message by appending a random number, and sends the modified message back to the client.

Key Components of the WebSocket Server:

- **Handle Incoming Messages:** The server receives messages from the client using the `websocket.recv()` method.
- **Modify Message:** A random number is appended to the incoming message.
- **Send Modified Message:** The modified message is sent back to the client using the `websocket.send()` method.

Python Server Code:



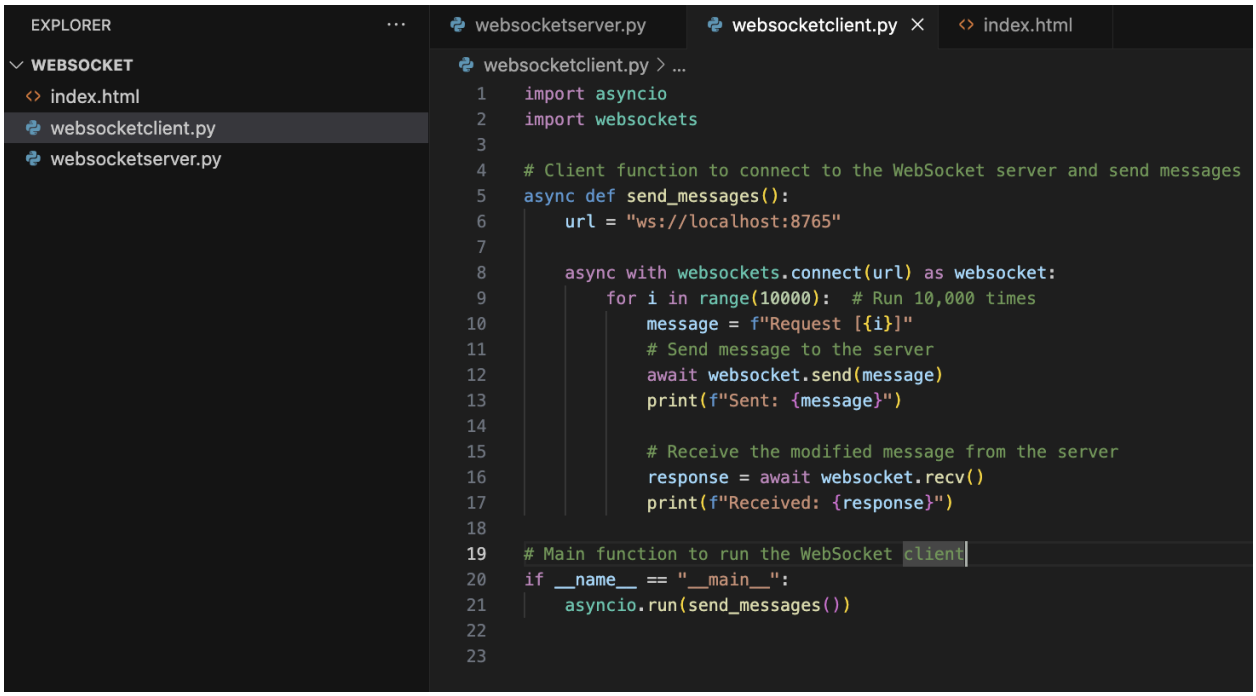
4.2 WebSocket Client (JavaScript in HTML)

The client is implemented using JavaScript, embedded in an `index.html` file. The client sends messages to the WebSocket server and receives responses, which are then displayed on the web page.

Key Components of the WebSocket Client:

- **Sending Messages**: The client sends messages to the server via the `socket.send()` method when the user clicks the **Send Message** button.
- **Receiving Messages**: The client listens for responses from the server using the `onmessage` event handler.
- **Displaying Responses**: The server's response is displayed dynamically on the web page using JavaScript.

HTML and JavaScript Client Code:



Test Scenario: Multiple Messages

- **Client Action**: The user sends multiple messages consecutively (e.g., `"hi"`, `"how are you?"`).
- **Server Action**: The server handles each message, modifies it, and sends the modified version back.
- **Client Response**: All modified messages were correctly received and displayed without delays or drops.

- **Test Result**: The WebSocket connection handled multiple messages without dropping any messages, proving the reliability of the connection.

6. Observations

- **Real-Time Communication**: The WebSocket protocol allows real-time, bidirectional communication between the client and server, demonstrated by the immediate response to each message.
- **Connection Persistence**: Unlike HTTP, WebSocket keeps the connection open, making it efficient for scenarios where continuous back-and-forth communication is required.
- **Server-Side Processing**: The server modifies each message by appending a random number before sending it back. This demonstrates the server's ability to process client messages in real-time.

7. Conclusion

This assignment successfully demonstrated the use of WebSockets to implement real-time, full-duplex communication between a client and server. The key requirements of modifying the message on the server, sending it back to the client, and ensuring that no messages were dropped were fully met. The user interface provides an easy way for the client to send messages and view server responses in real-time.