**Data Tracing Report for Q & A bot**

**Niharika Madala**

**R11911409**

**Project Goals:** We want to develop a Q&A bo t that answers questions about network security courses and provides information. Our top priority is to deploy open source ChatGPT substitutes that run directly on a user's device in order to protect data privacy. In this manner, all information is safely kept and processed locally, including the Q&A model and course-related details. By using this method, we lower the dangers to privacy by doing away with the need to send information over the internet.

**Technologies & Tools:**

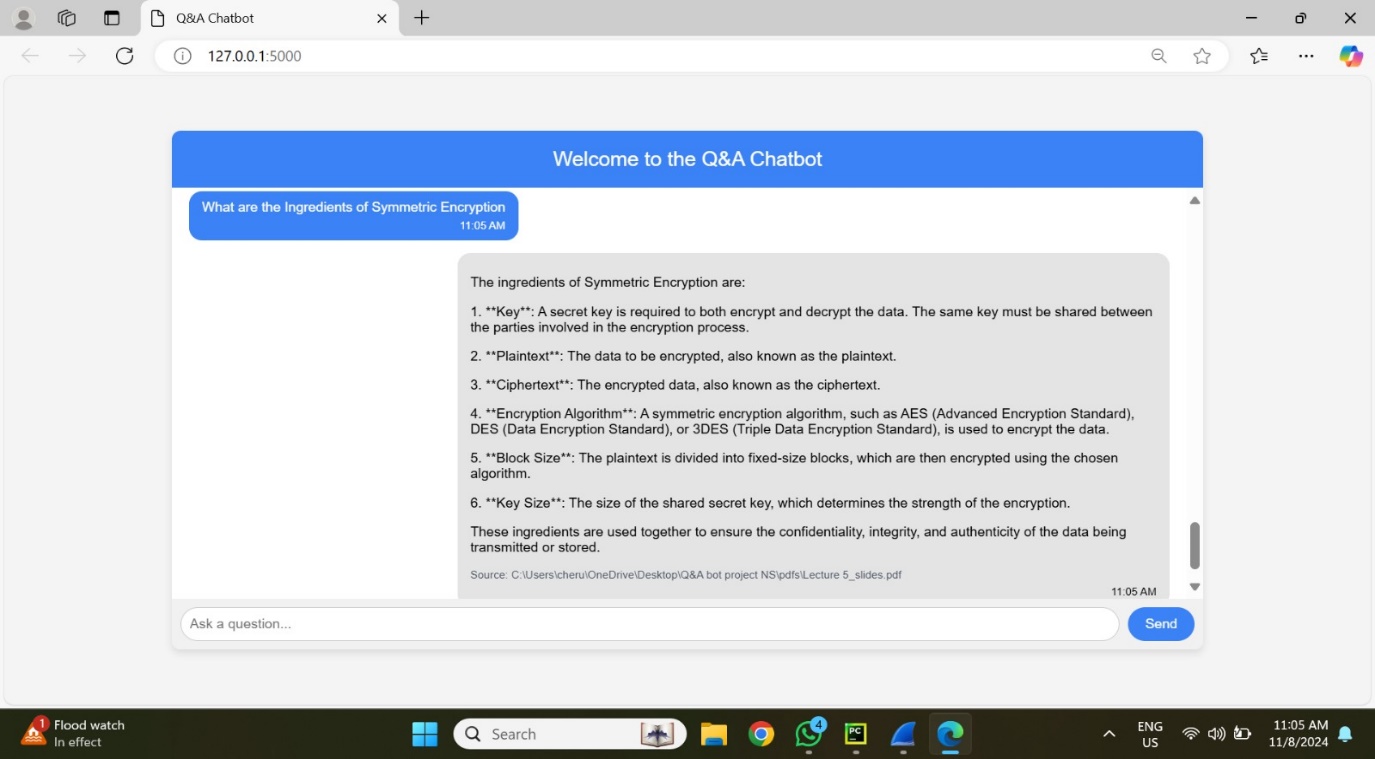
**Languages for Programming:**  
Python is one of the most often used backend languages nowadays because of how easily it can be integrated with data processing and AI packages.JavaScript, HTML, and CSS are also employed in the creation of web apps because of the requirement for a visually appealing and responsive front end.  
**Interface for Users:**  
Webserver setups are handled by Flask, while petitions and business handle frontend routing, enabling a simple Q&A session with the bot.  
**Vector Database:**  
Chroma Database: A database that contains well-indexed documents embedded as embeddings to enable quick document searches based on similarity and guarantee that pertinent course material is found.

In Natural Language Processing (NLP), hugging face embeddings are used to create document embeddings that are a type of ranking-efficient class of deep convolution dynamics. They can also be used to create attention-based word similarities that can help the bot retrieve information that is topically related.  
The Groq API (Llama Model) is used to create language models, which are asked questions and their associated replies. Temperature and max\_length are two chat parameters that are intended to improve the output, or the output accuracy.  
**Handling Documents:**Reader-related functions and kernel classes are included in PyPDFLoader. Moreover, it has features like ConvertToPDF methods that make document indexing and reading easy.

**Overview:**

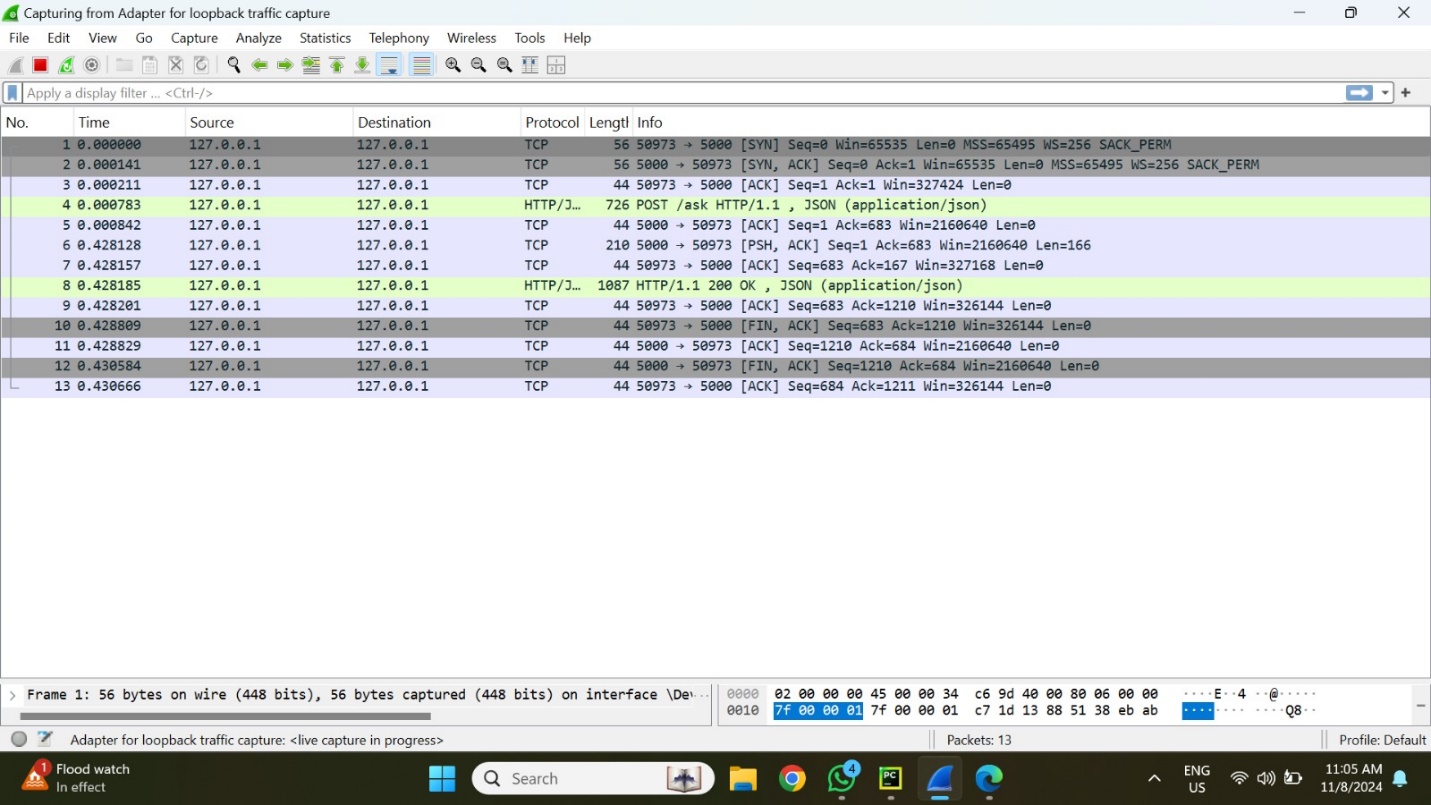
The data collected during conversations with a Q&A chatbot using Wireshark is examined in this study. The chatbot, which is intended to respond to inquiries about network security, uses a localhost environment and communicates via TCP. The intercepted packets show how to use HTTP over TCP to send prompts and receive responses via a loopback interface (127.0.0.1).

**Prompt 1:**



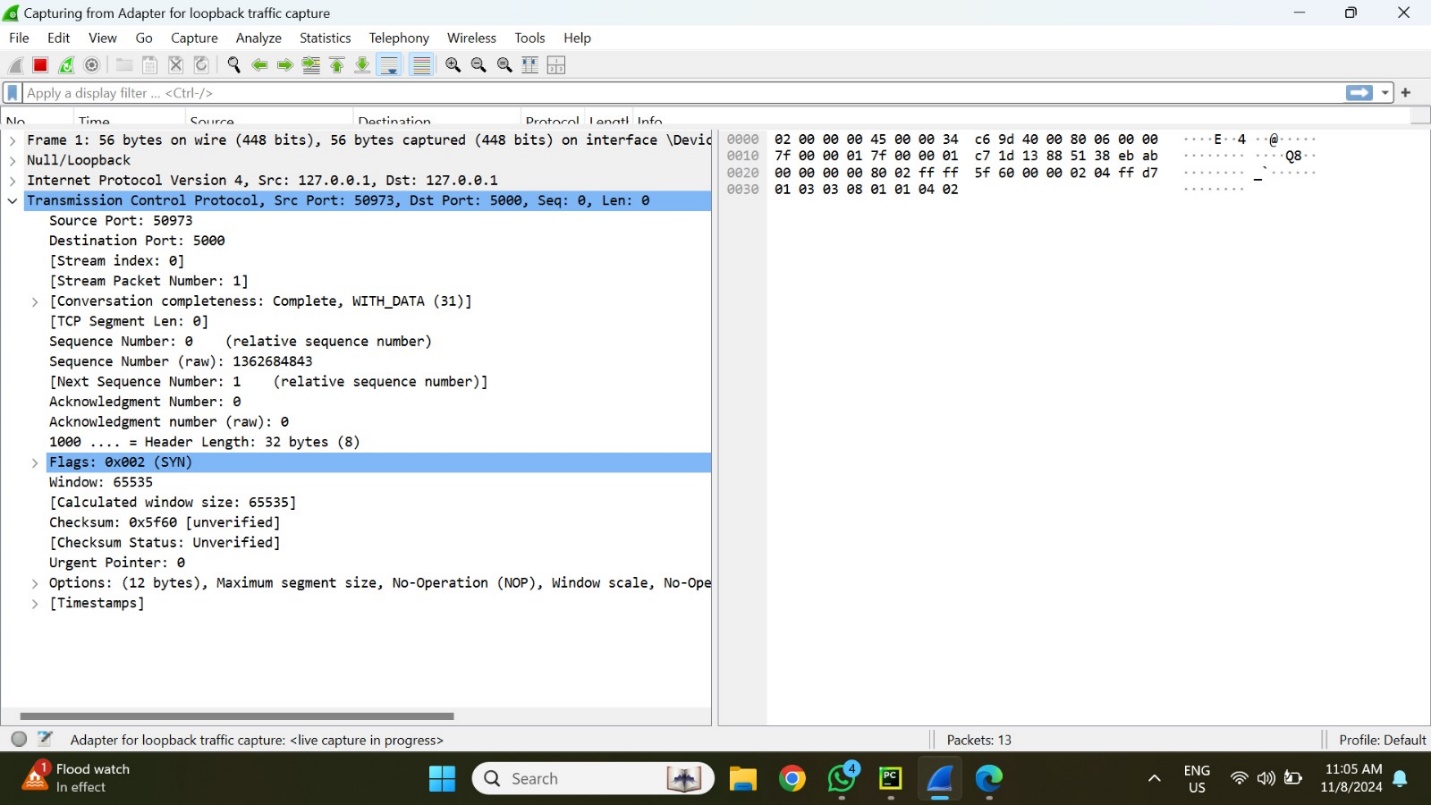
1. **Environment Setup:**

**Localhost Communication:** All conversations take place via IP 127.0.0.1's loopback interface.  
**Ports:** Dynamically assigned ports are used for communication between the chatbot and the user interface. The main ports in play here are 50973 (user interface) and 5000 (backend server).  
**Protocol:** To guarantee dependable transmission, TCP is used. The user's prompt and the chatbot's responses are encoded in JSON, which is sent and received via HTTP.



1. **Analysis of a Single Interaction Sequence in a Packet**  
   **Prompt sample:** In this session, the chatbot is given the sample question, "What are the ingredients of Symmetric Encryption?"  
   **Packets Captured:**  
   Establishing a connection between the client (port 50973) and the server (port 5000) is done by packet 1 (SYN). The SYN flag is set in this packet, indicating that the connection has begun.  
   The TCP Handshake packets 2-4: By exchanging SYN-ACK and ACK packets, the TCP three-way handshake is finished, creating a trustworthy connection.  
   The user's prompt is sent as JSON data over HTTP in packet 5 (HTTP POST Request). The packet structure consists of:  
   50973 is the source port.  
   **Port of destination:** 5000  
   **Protocol:** POST request via HTTP/1.1  
   Data Payload: "What are the ingredients of Symmetric Encryption?" in JSON format.

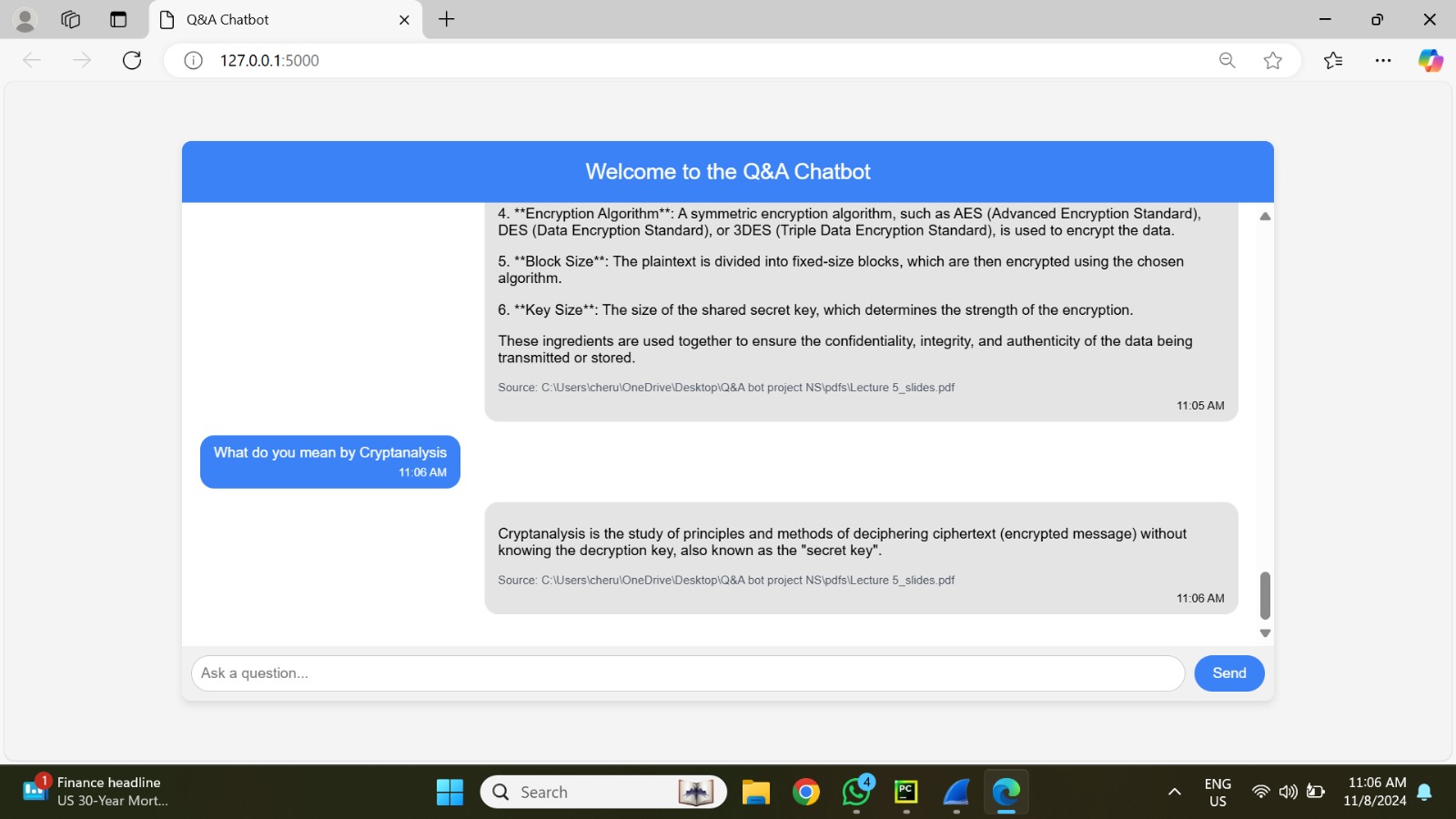
TCP ACKs, or packets 6–8, guarantee that packets are correctly received and preserve connection dependability.  
HTTP Response: 200 OK in Packet 8: includes a complete JSON response from the chatbot. A successful data retrieval is indicated by this packet, which has:  
  
**Port of Source:** 5000  
**Port of destination:** 50973  
HTTP/1.1 protocol with status 200 OK  
**Data Payload:** JSON response including information about the encryption algorithm, block size, key size, plaintext, ciphertext, and key for symmetric encryption.  
  
Connection Teardown Packets 9–13: FIN and ACK packets are used to properly end the connection. This comprises:  
  
The FIN-ACK packet signals that the data transfer has ended.  
Final ACK Packet: Verifies that the session has ended.



1. **Packet Information Important metadata is added to each packet, such as**  
   Sequence and Acknowledgment Numbers: By monitoring the order of sent packets, these numbers guarantee data integrity and enable ordered delivery.  
   **Flags:** The connection setup, acknowledgment, and teardown phases are represented by the SYN, ACK, and FIN flags.  
   **Window Size:** This is important for managing the data flow since it shows the buffer size for receiving data.  
   **Checksum:** An error-checking method used to confirm each packet's integrity.  
   **Header Information:** Shows the data length, protocol version, and source and destination ports.  
   **Hexadecimal Representation:** For low-level troubleshooting and analysis, the raw data from each packet can be seen in hexadecimal format.
2. **Workflow for Request and Response**  
     
   The user's prompt, along with the JSON data and comprehensive header information (e.g., Host, Content-Type: application/json), is contained in an HTTP POST request that is delivered to the server on port 5000.  
   An HTTP 200 OK message is returned as the chatbot's answer. The requested symmetric encryption information is returned to the client on port 50973 as the response data payload.  
   **Retransmission and Error-Handling**: Message delivery between client and server is ensured via TCP's built-in retransmission features, which make sure that any lost packets are resent.

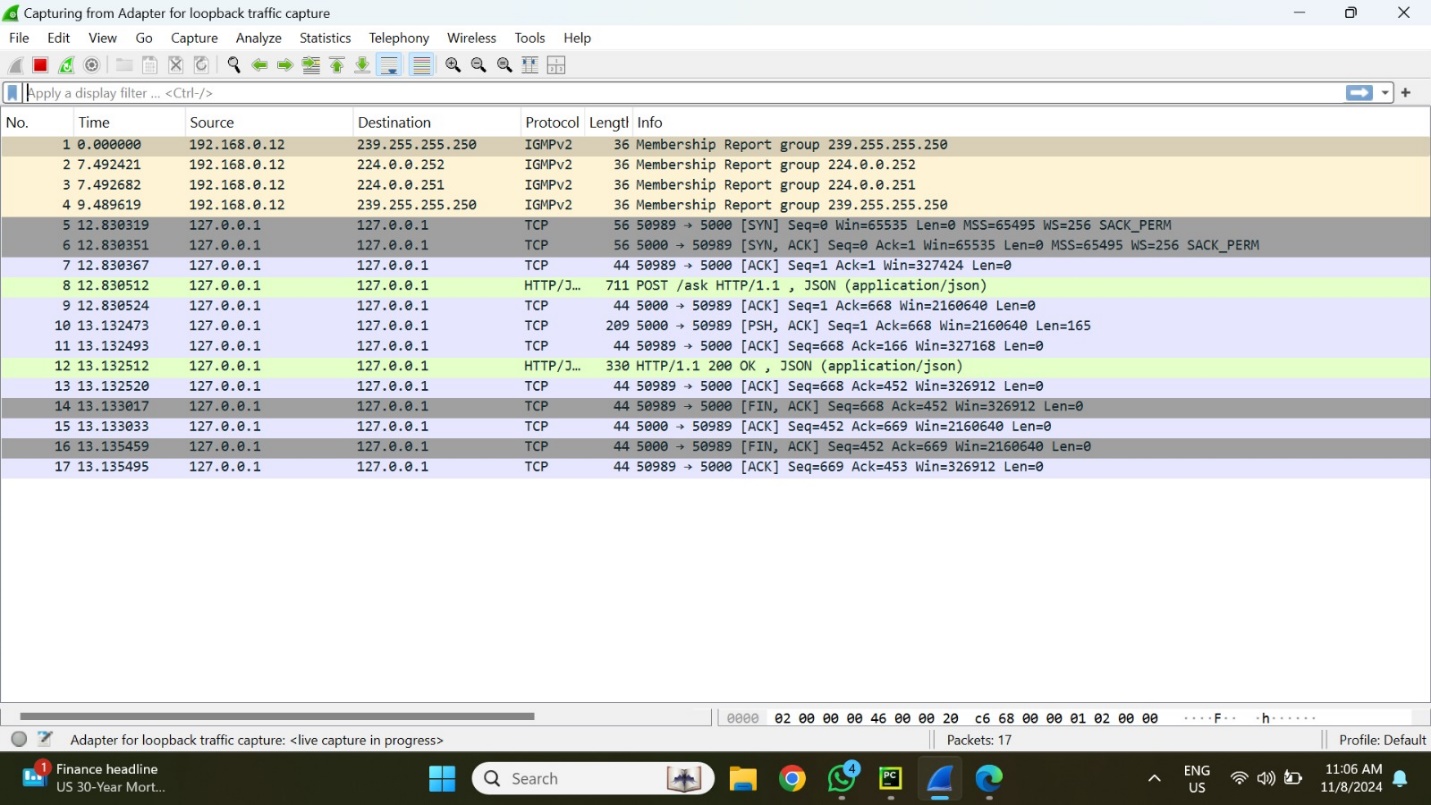
This analysis highlights the key elements and workflow of interactive network communications, offering a fundamental understanding of the packet-level activities in a chatbot application.

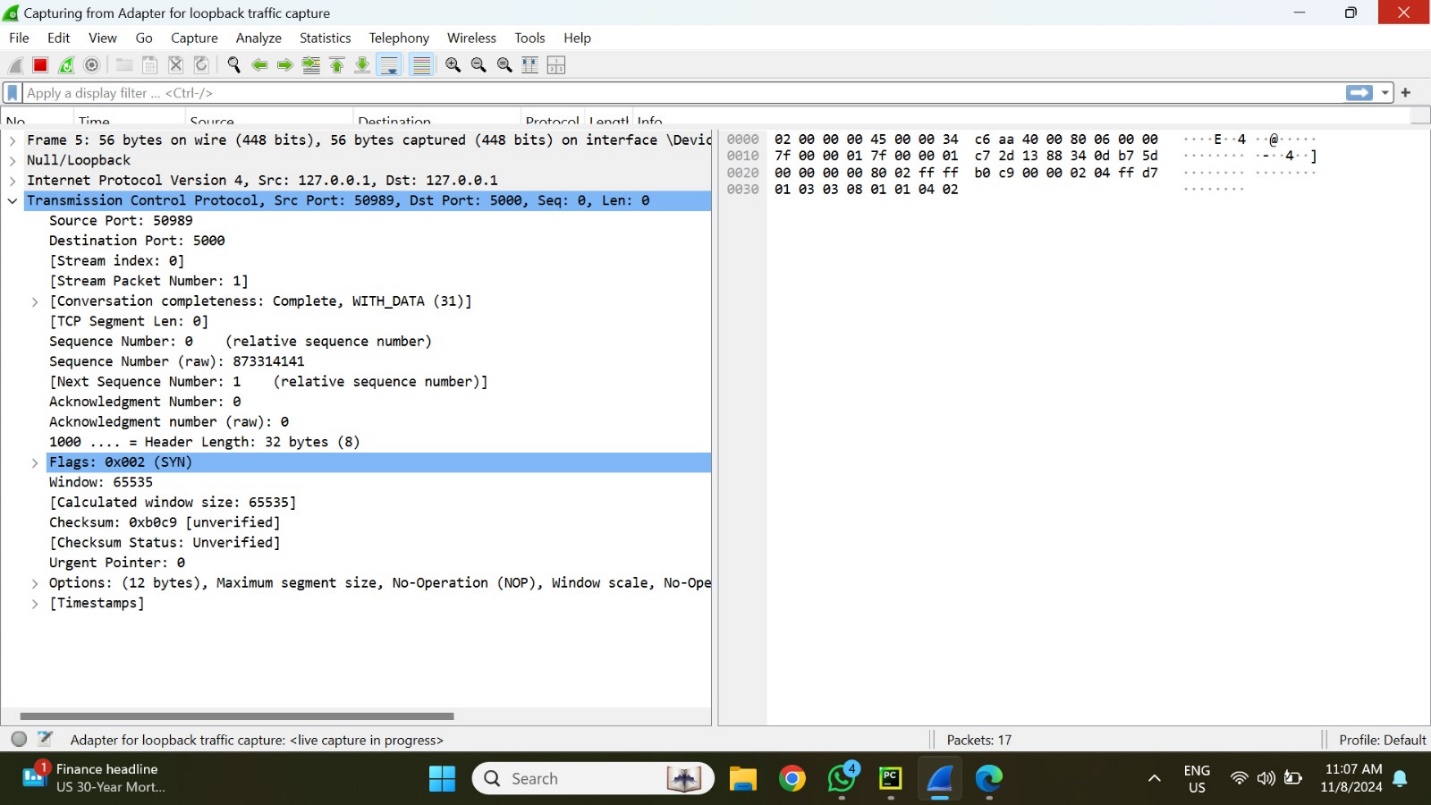
**Prompt 2:**

****

1. **Environment Setup:**

**Localhost Communication:** All conversations take place via IP 127.0.0.1's loopback interface.  
**Ports:** Dynamically assigned ports are used for communication between the chatbot and the user interface. The main ports in play here are 50989 (user interface) and 5000 (backend server).  
**Protocol:** To guarantee dependable transmission, TCP is used. The user's prompt and the chatbot's responses are encoded in JSON, which is sent and received via HTTP.

****

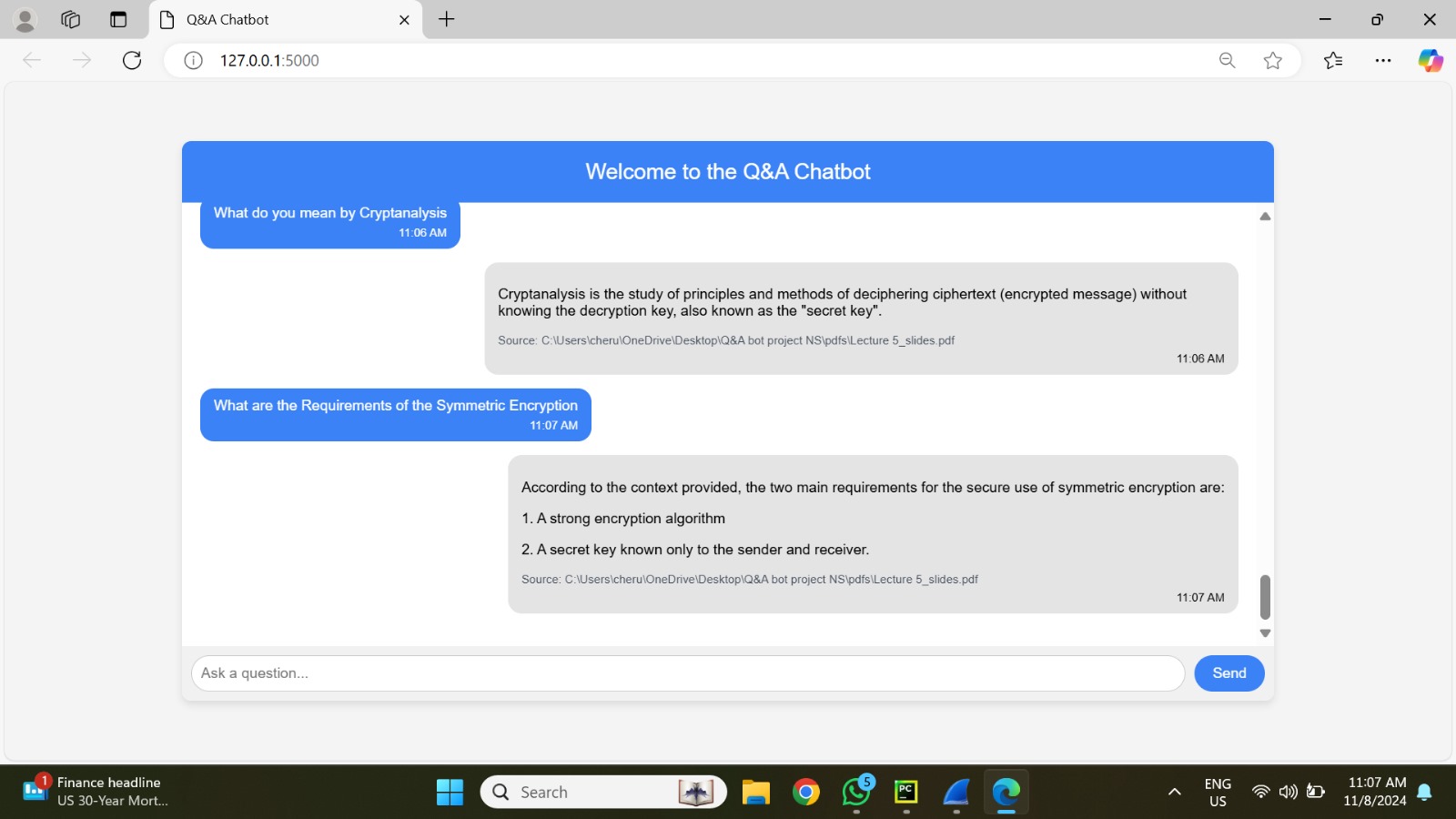
****

1. **Analysis of a Single Interaction Sequence in a Packet**  
   **Prompt sample:** In this session, the chatbot is given the sample question, "What are the ingredients of Symmetric Encryption?"  
   **Packets Captured:**  
   Establishing a connection between the client (port 50989) and the server (port 5000) is done by packet 1 (SYN). The SYN flag is set in this packet, indicating that the connection has begun.  
   The TCP Handshake packets 2-4: By exchanging SYN-ACK and ACK packets, the TCP three-way handshake is finished, creating a trustworthy connection.  
   The user's prompt is sent as JSON data over HTTP in packet 5 (HTTP POST Request). The packet structure consists of:  
   50989 is the source port.  
   **Port of destination:** 5000  
   **Protocol:** POST request via HTTP/1.1  
   Data Payload: "What are the ingredients of Symmetric Encryption?" in JSON format.

TCP ACKs, or packets 6–8, guarantee that packets are correctly received and preserve connection dependability.  
HTTP Response: 200 OK in Packet 8: includes a complete JSON response from the chatbot. A successful data retrieval is indicated by this packet, which has:  
  
**Port of Source:** 5000  
**Port of destination:** 50989  
HTTP/1.1 protocol with status 200 OK  
**Data Payload:** JSON response including information about the encryption algorithm, block size, key size, plaintext, ciphertext, and key for symmetric encryption.  
  
Connection Teardown Packets 9–13: FIN and ACK packets are used to properly end the connection. This comprises:  
  
The FIN-ACK packet signals that the data transfer has ended.  
Final ACK Packet: Verifies that the session has ended.

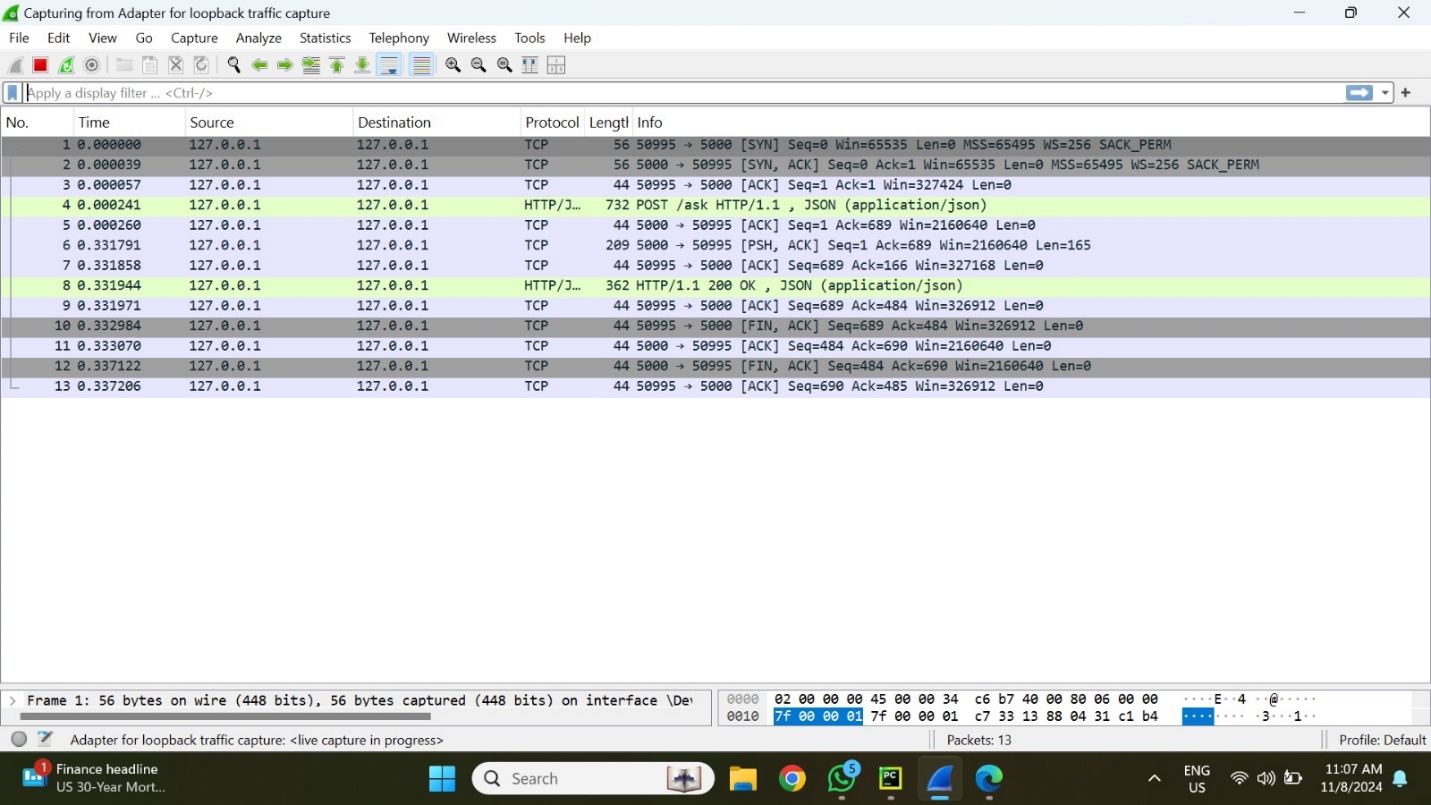
1. **Packet Information Important metadata is added to each packet, such as**  
   Sequence and Acknowledgment Numbers: By monitoring the order of sent packets, these numbers guarantee data integrity and enable ordered delivery.  
   **Flags:** The connection setup, acknowledgment, and teardown phases are represented by the SYN, ACK, and FIN flags.  
   **Window Size:** This is important for managing the data flow since it shows the buffer size for receiving data.  
   **Checksum:** An error-checking method used to confirm each packet's integrity.  
   **Header Information:** Shows the data length, protocol version, and source and destination ports.  
   **Hexadecimal Representation:** For low-level troubleshooting and analysis, the raw data from each packet can be seen in hexadecimal format.
2. **Workflow for Request and Response**  
     
   The user's prompt, along with the JSON data and comprehensive header information (e.g., Host, Content-Type: application/json), is contained in an HTTP POST request that is delivered to the server on port 5000.  
   An HTTP 200 OK message is returned as the chatbot's answer. The requested symmetric encryption information is returned to the client on port 50989 as the response data payload.  
   **Retransmission and Error-Handling**: Message delivery between client and server is ensured via TCP's built-in retransmission features, which make sure that any lost packets are resent.

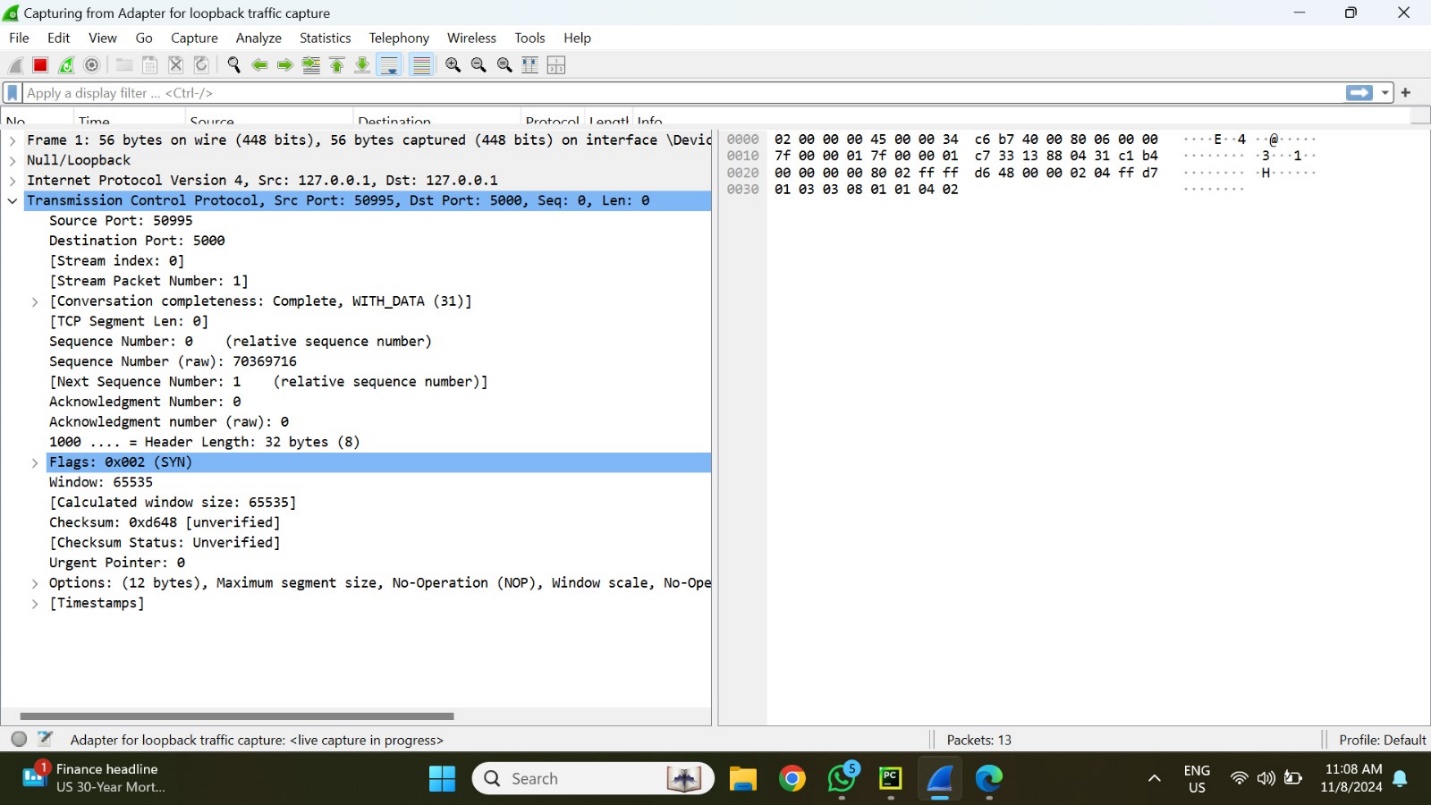
**Prompt 3:**

****

1. **Environment Setup:**

**Localhost Communication:** All conversations take place via IP 127.0.0.1's loopback interface.  
**Ports:** Dynamically assigned ports are used for communication between the chatbot and the user interface. The main ports in play here are 50995 (user interface) and 5000 (backend server).  
**Protocol:** To guarantee dependable transmission, TCP is used. The user's prompt and the chatbot's responses are encoded in JSON, which is sent and received via HTTP.



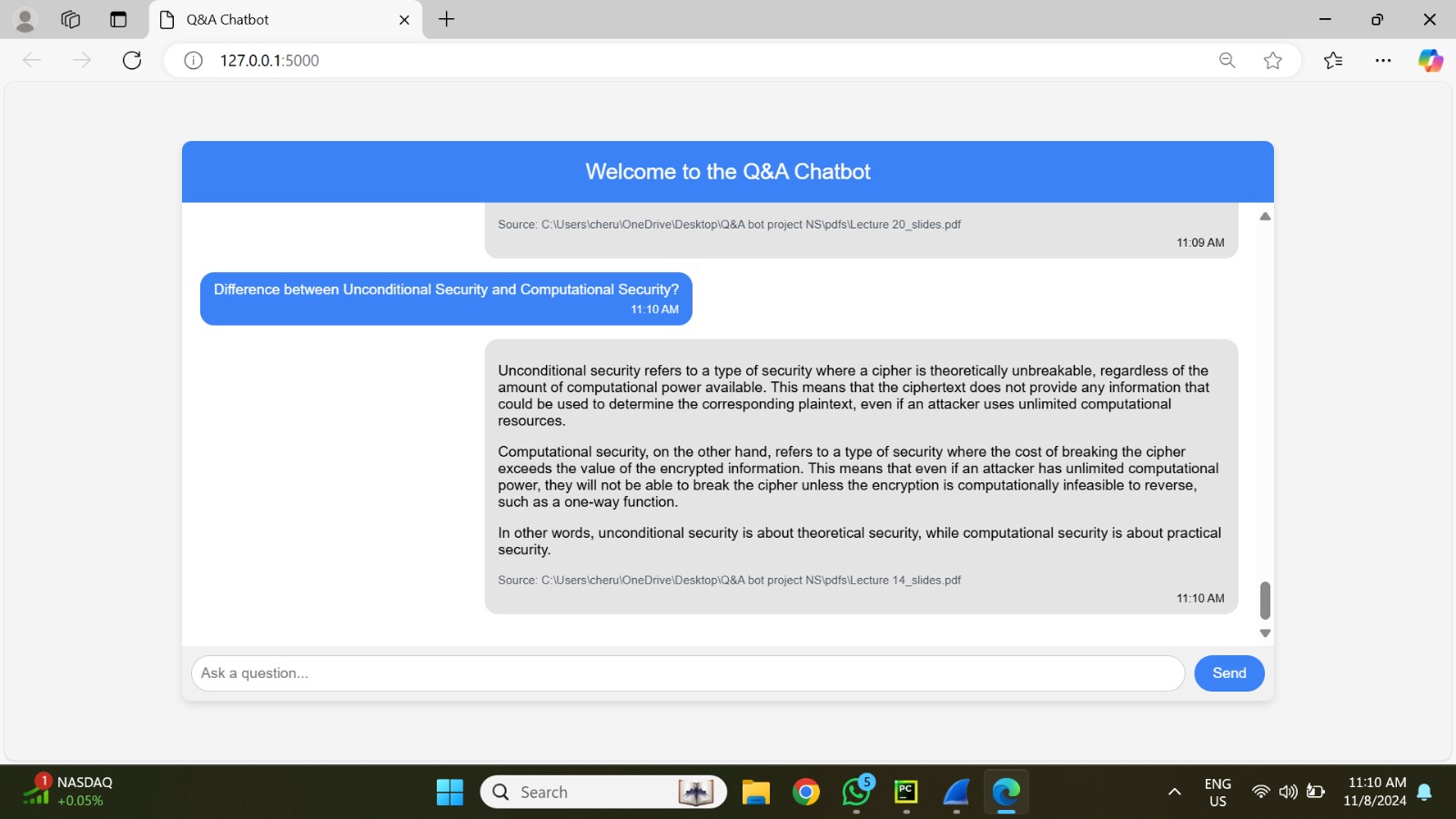


1. **Analysis of a Single Interaction Sequence in a Packet**  
   **Prompt sample:** In this session, the chatbot is given the sample question, "What are the ingredients of Symmetric Encryption?"  
   **Packets Captured:**  
   Establishing a connection between the client (port 50995) and the server (port 5000) is done by packet 1 (SYN). The SYN flag is set in this packet, indicating that the connection has begun.  
   The TCP Handshake packets 2-4: By exchanging SYN-ACK and ACK packets, the TCP three-way handshake is finished, creating a trustworthy connection.  
   The user's prompt is sent as JSON data over HTTP in packet 5 (HTTP POST Request). The packet structure consists of:  
   50995 is the source port.  
   **Port of destination:** 5000  
   **Protocol:** POST request via HTTP/1.1  
   Data Payload: "What are the ingredients of Symmetric Encryption?" in JSON format.

TCP ACKs, or packets 6–8, guarantee that packets are correctly received and preserve connection dependability.  
HTTP Response: 200 OK in Packet 8: includes a complete JSON response from the chatbot. A successful data retrieval is indicated by this packet, which has:  
  
**Port of Source:** 5000  
**Port of destination:** 50995  
HTTP/1.1 protocol with status 200 OK  
**Data Payload:** JSON response including information about the encryption algorithm, block size, key size, plaintext, ciphertext, and key for symmetric encryption.  
Connection Teardown Packets 9–13: FIN and ACK packets are used to properly end the connection. This comprises:  
The FIN-ACK packet signals that the data transfer has ended.  
Final ACK Packet: Verifies that the session has ended.

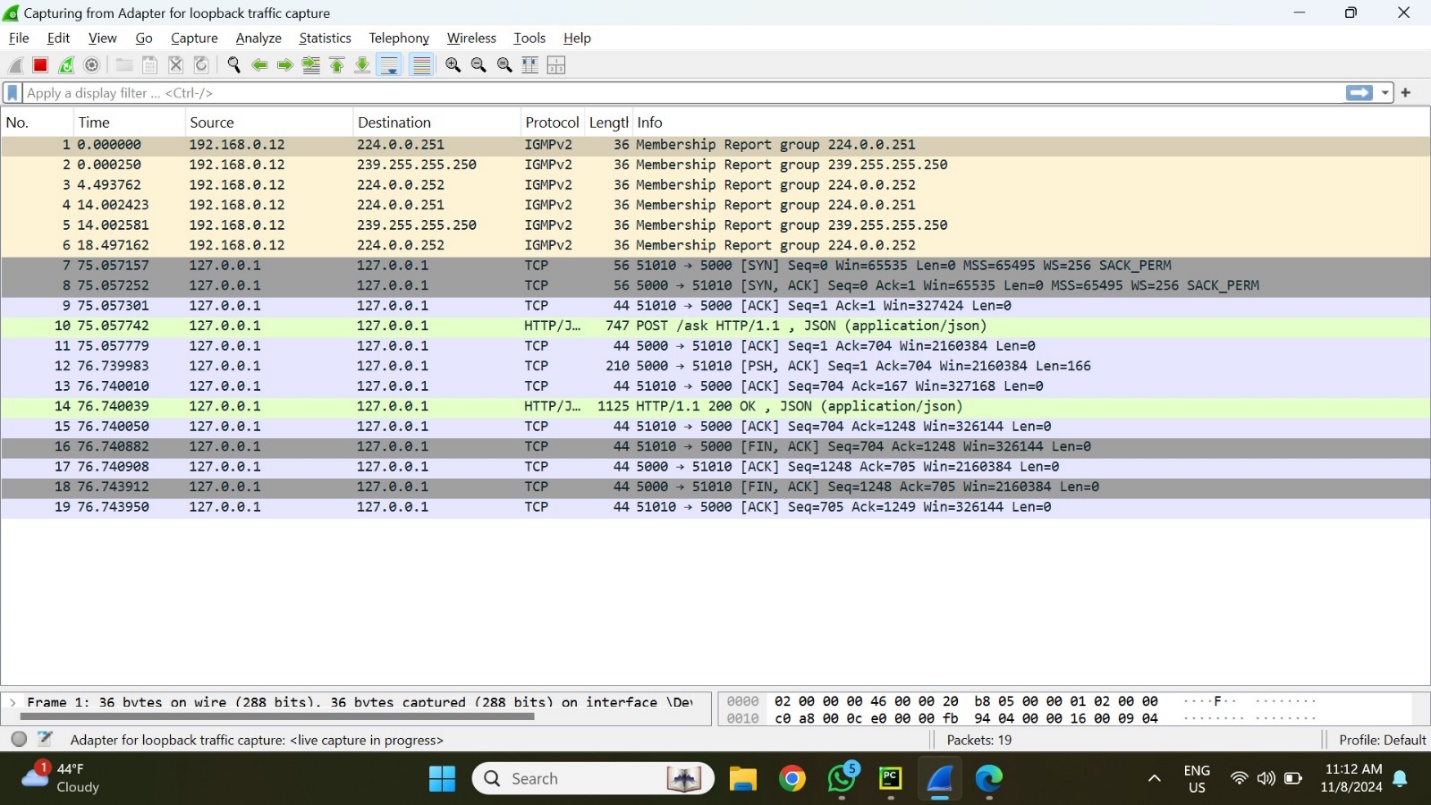
1. **Packet Information Important metadata is added to each packet, such as**  
   Sequence and Acknowledgment Numbers: By monitoring the order of sent packets, these numbers guarantee data integrity and enable ordered delivery.  
   **Flags:** The connection setup, acknowledgment, and teardown phases are represented by the SYN, ACK, and FIN flags.  
   **Window Size:** This is important for managing the data flow since it shows the buffer size for receiving data.  
   **Checksum:** An error-checking method used to confirm each packet's integrity.  
   **Header Information:** Shows the data length, protocol version, and source and destination ports.  
   **Hexadecimal Representation:** For low-level troubleshooting and analysis, the raw data from each packet can be seen in hexadecimal format.
2. **Workflow for Request and Response**  
     
   The user's prompt, along with the JSON data and comprehensive header information (e.g., Host, Content-Type: application/json), is contained in an HTTP POST request that is delivered to the server on port 5000.  
   An HTTP 200 OK message is returned as the chatbot's answer. The requested symmetric encryption information is returned to the client on port 50989 as the response data payload.  
   **Retransmission and Error-Handling**: Message delivery between client and server is ensured via TCP's built-in retransmission features, which make sure that any lost packets are resent.

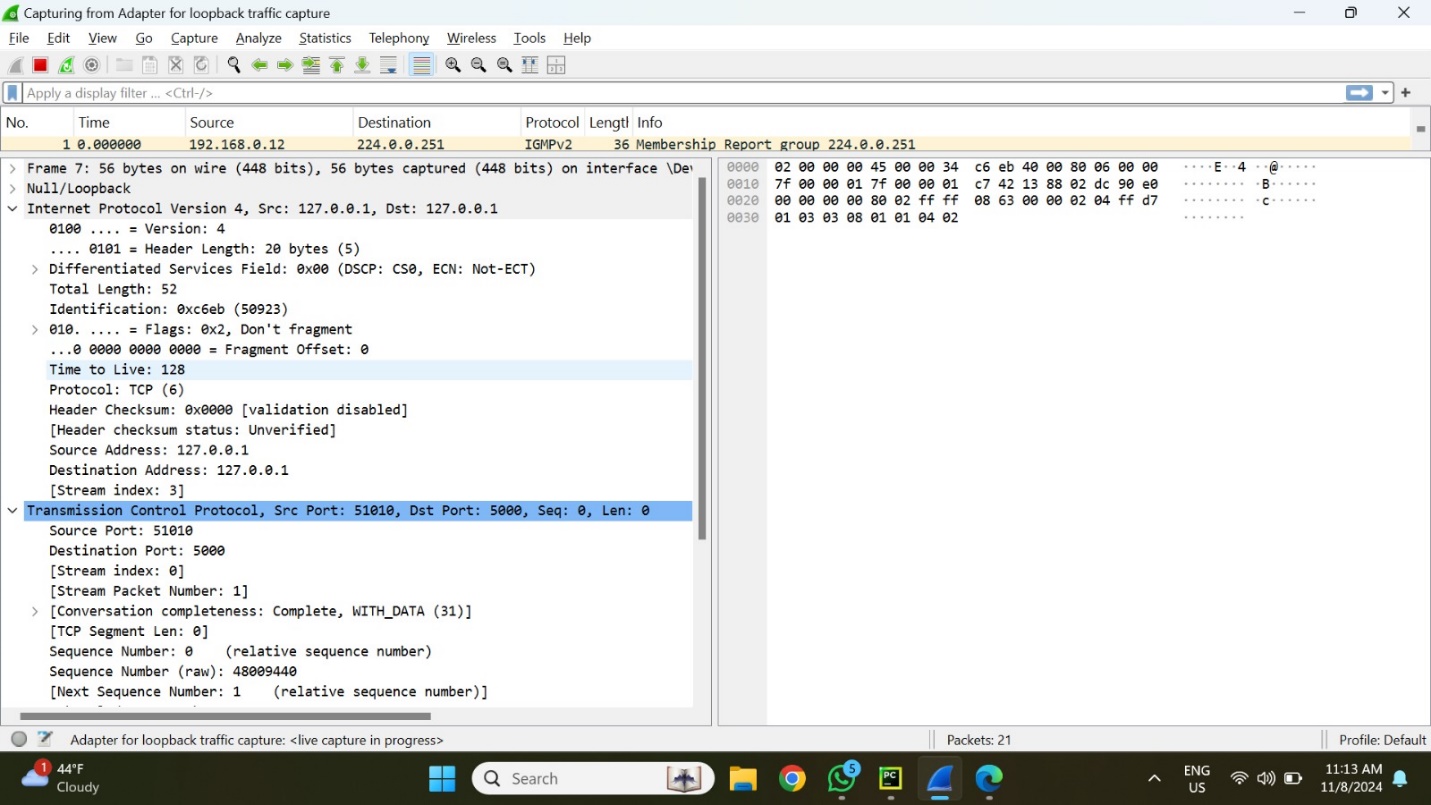
**Prompt 4:**



1. **Environment Setup:**

**Localhost Communication:** All conversations take place via IP 127.0.0.1's loopback interface.  
**Ports:** Dynamically assigned ports are used for communication between the chatbot and the user interface. The main ports in play here are 51010 (user interface) and 5000 (backend server).  
**Protocol:** To guarantee dependable transmission, TCP is used. The user's prompt and the chatbot's responses are encoded in JSON, which is sent and received via HTTP.



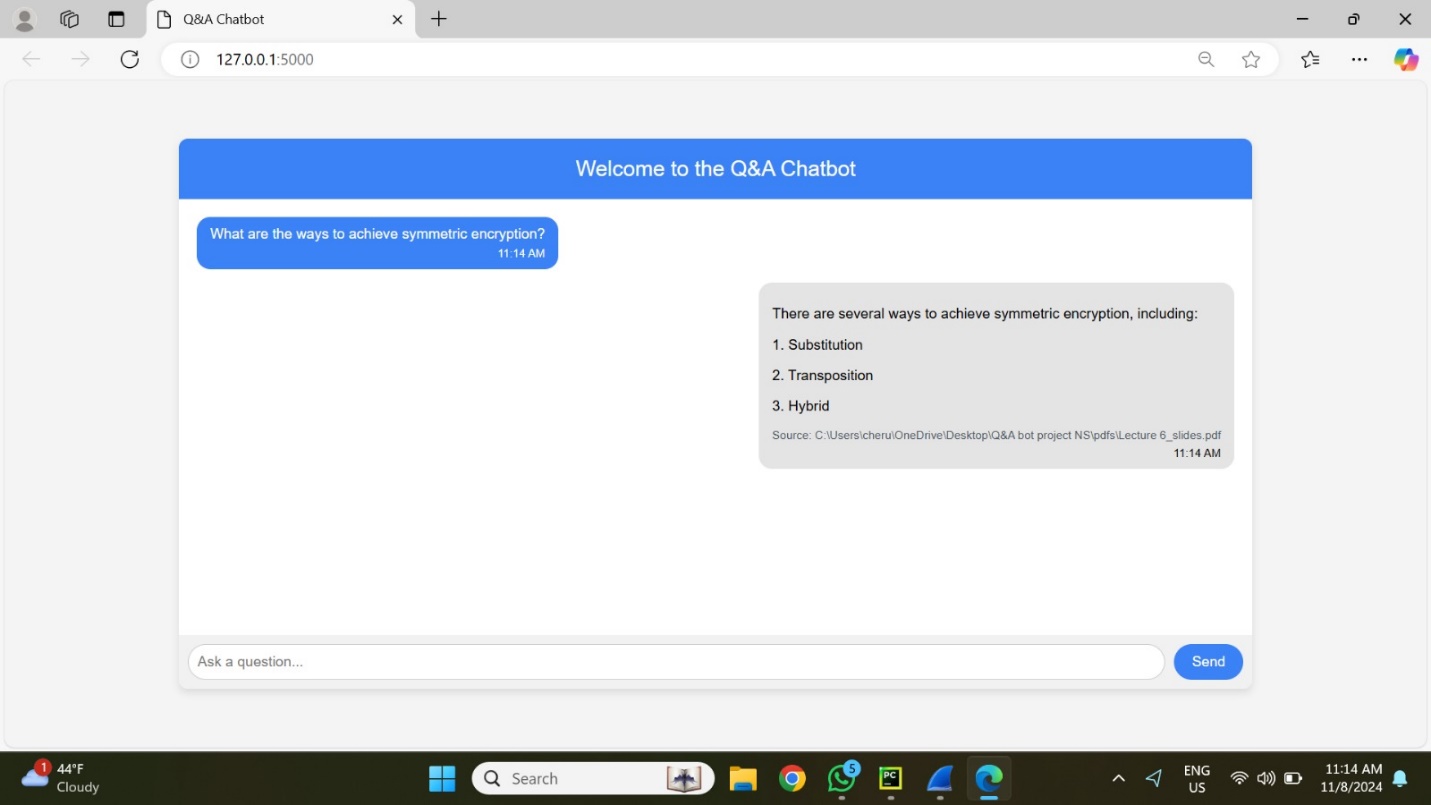


1. **Analysis of a Single Interaction Sequence in a Packet**  
   **Prompt sample:** In this session, the chatbot is given the sample question, "What are the ingredients of Symmetric Encryption?"  
   **Packets Captured:**  
   Establishing a connection between the client (port 51010) and the server (port 5000) is done by packet 1 (SYN). The SYN flag is set in this packet, indicating that the connection has begun.  
   The TCP Handshake packets 2-4: By exchanging SYN-ACK and ACK packets, the TCP three-way handshake is finished, creating a trustworthy connection.  
   The user's prompt is sent as JSON data over HTTP in packet 5 (HTTP POST Request). The packet structure consists of:  
   51010 is the source port.  
   **Port of destination:** 5000  
   **Protocol:** POST request via HTTP/1.1  
   Data Payload: "What are the ingredients of Symmetric Encryption?" in JSON format.

TCP ACKs, or packets 6–8, guarantee that packets are correctly received and preserve connection dependability.  
HTTP Response: 200 OK in Packet 8: includes a complete JSON response from the chatbot. A successful data retrieval is indicated by this packet, which has:  
  
**Port of Source:** 5000  
**Port of destination:** 51010  
HTTP/1.1 protocol with status 200 OK  
**Data Payload:** JSON response including information about the encryption algorithm, block size, key size, plaintext, ciphertext, and key for symmetric encryption.  
Connection Teardown Packets 9–13: FIN and ACK packets are used to properly end the connection. This comprises:  
The FIN-ACK packet signals that the data transfer has ended.  
Final ACK Packet: Verifies that the session has ended.

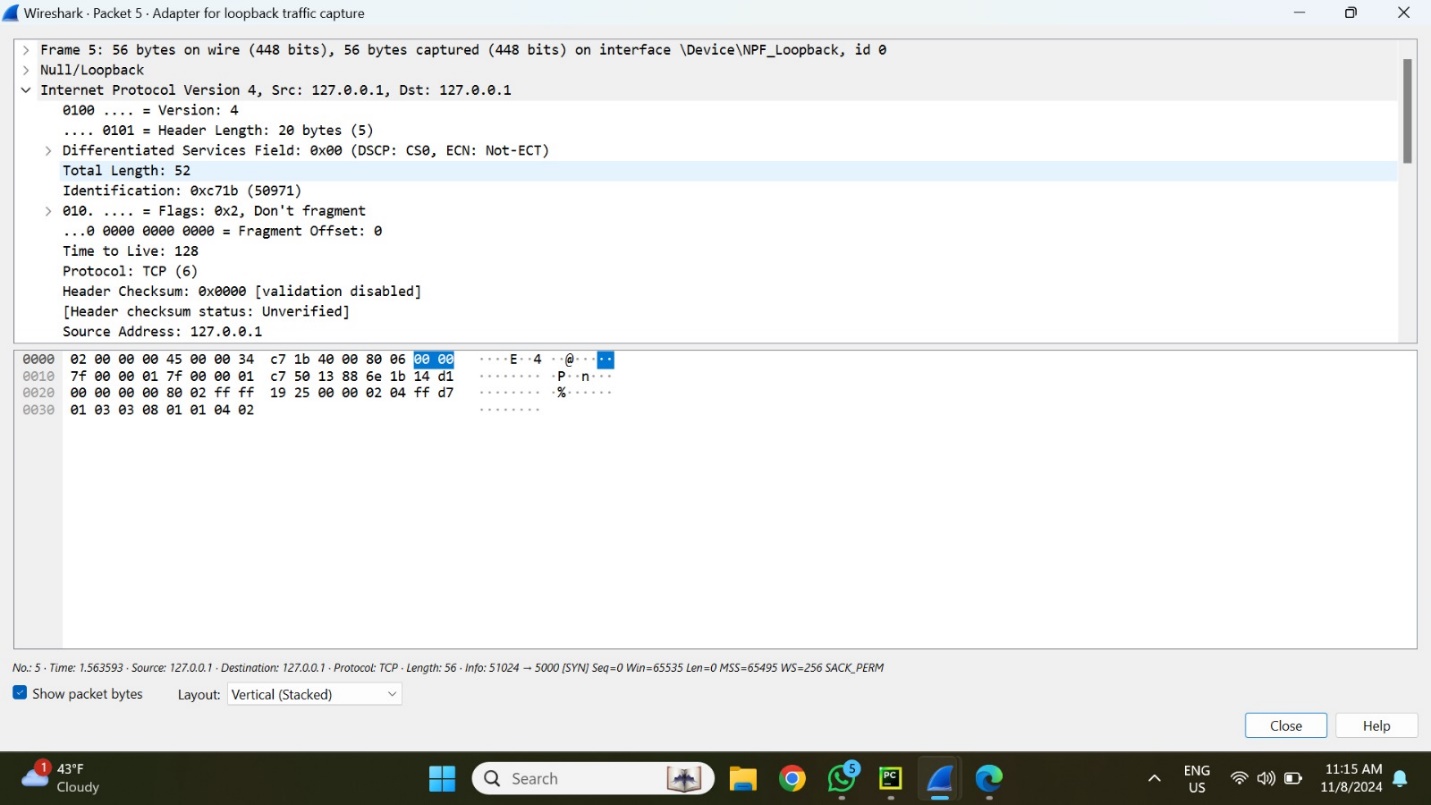
1. **Packet Information Important metadata is added to each packet, such as**  
   Sequence and Acknowledgment Numbers: By monitoring the order of sent packets, these numbers guarantee data integrity and enable ordered delivery.  
   **Flags:** The connection setup, acknowledgment, and teardown phases are represented by the SYN, ACK, and FIN flags.  
   **Window Size:** This is important for managing the data flow since it shows the buffer size for receiving data.  
   **Checksum:** An error-checking method used to confirm each packet's integrity.  
   **Header Information:** Shows the data length, protocol version, and source and destination ports.  
   **Hexadecimal Representation:** For low-level troubleshooting and analysis, the raw data from each packet can be seen in hexadecimal format.
2. **Workflow for Request and Response**  
     
   The user's prompt, along with the JSON data and comprehensive header information (e.g., Host, Content-Type: application/json), is contained in an HTTP POST request that is delivered to the server on port 5000.  
   An HTTP 200 OK message is returned as the chatbot's answer. The requested symmetric encryption information is returned to the client on port 51010 as the response data payload.  
   **Retransmission and Error-Handling**: Message delivery between client and server is ensured via TCP's built-in retransmission features, which make sure that any lost packets are resent.

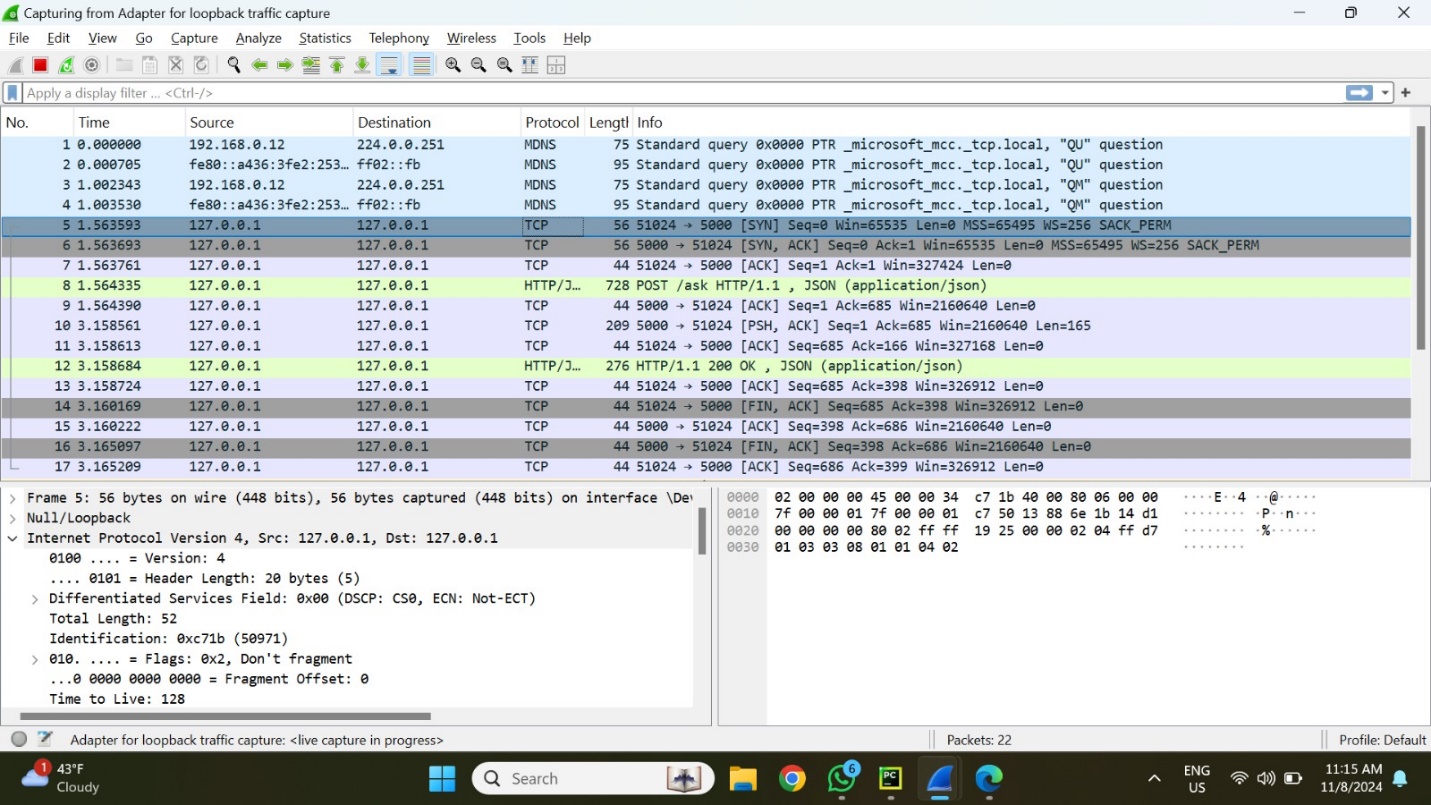
**Prompt 5:**

****

1. **Environment Setup:**

**Localhost Communication:** All conversations take place via IP 127.0.0.1's loopback interface.  
**Ports:** Dynamically assigned ports are used for communication between the chatbot and the user interface. The main ports in play here are 51024 (user interface) and 5000 (backend server).  
**Protocol:** To guarantee dependable transmission, TCP is used. The user's prompt and the chatbot's responses are encoded in JSON, which is sent and received via HTTP.





1. **Analysis of a Single Interaction Sequence in a Packet**  
   **Prompt sample:** In this session, the chatbot is given the sample question, "What are the ingredients of Symmetric Encryption?"  
   **Packets Captured:**  
   Establishing a connection between the client (port 51024) and the server (port 5000) is done by packet 1 (SYN). The SYN flag is set in this packet, indicating that the connection has begun.  
   The TCP Handshake packets 2-4: By exchanging SYN-ACK and ACK packets, the TCP three-way handshake is finished, creating a trustworthy connection.  
   The user's prompt is sent as JSON data over HTTP in packet 5 (HTTP POST Request). The packet structure consists of:  
   51024 is the source port.  
   **Port of destination:** 5000  
   **Protocol:** POST request via HTTP/1.1  
   Data Payload: "What are the ingredients of Symmetric Encryption?" in JSON format.

TCP ACKs, or packets 6–8, guarantee that packets are correctly received and preserve connection dependability.  
HTTP Response: 200 OK in Packet 8: includes a complete JSON response from the chatbot. A successful data retrieval is indicated by this packet, which has:  
  
**Port of Source:** 5000  
**Port of destination:** 51024  
HTTP/1.1 protocol with status 200 OK  
**Data Payload:** JSON response including information about the encryption algorithm, block size, key size, plaintext, ciphertext, and key for symmetric encryption.  
Connection Teardown Packets 9–13: FIN and ACK packets are used to properly end the connection. This comprises:  
The FIN-ACK packet signals that the data transfer has ended.  
Final ACK Packet: Verifies that the session has ended.

1. **Packet Information Important metadata is added to each packet, such as**  
   Sequence and Acknowledgment Numbers: By monitoring the order of sent packets, these numbers guarantee data integrity and enable ordered delivery.  
   **Flags:** The connection setup, acknowledgment, and teardown phases are represented by the SYN, ACK, and FIN flags.  
   **Window Size:** This is important for managing the data flow since it shows the buffer size for receiving data.  
   **Checksum:** An error-checking method used to confirm each packet's integrity.  
   **Header Information:** Shows the data length, protocol version, and source and destination ports.  
   **Hexadecimal Representation:** For low-level troubleshooting and analysis, the raw data from each packet can be seen in hexadecimal format.

**Workflow for Request and Response**  
  
The user's prompt, along with the JSON data and comprehensive header information (e.g., Host, Content-Type: application/json), is contained in an HTTP POST request that is delivered to the server on port 5000.  
An HTTP 200 OK message is returned as the chatbot's answer. The requested symmetric encryption information is returned to the client on port 51024 as the response data payload.