Network Security Tutor & Quiz Agent

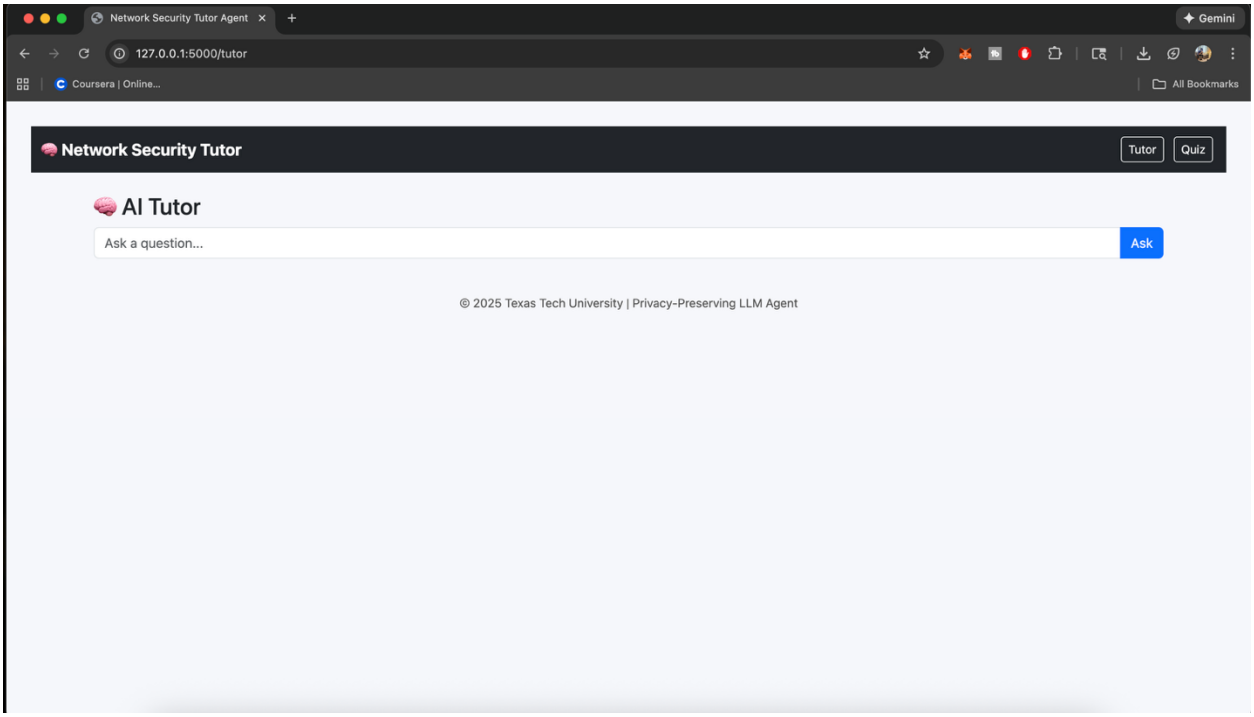
Name: Manideep Thogiti
R#: R11934465

This project demonstrates the design and implementation of an AI-powered Network Security Tutor, developed to help users understand and practice core concepts of network security. The system operates entirely in **a** local environment, where the Flask web application (running on port 5000) serves both the frontend interface and backend API.

When a user submits a question, the backend retrieves relevant study materials and contextual information from a Chroma vector database, forwards this data to the Llama 3.2 model hosted locally via Ollama (port 11434)**,** and returns the generated response to the user interface for display.

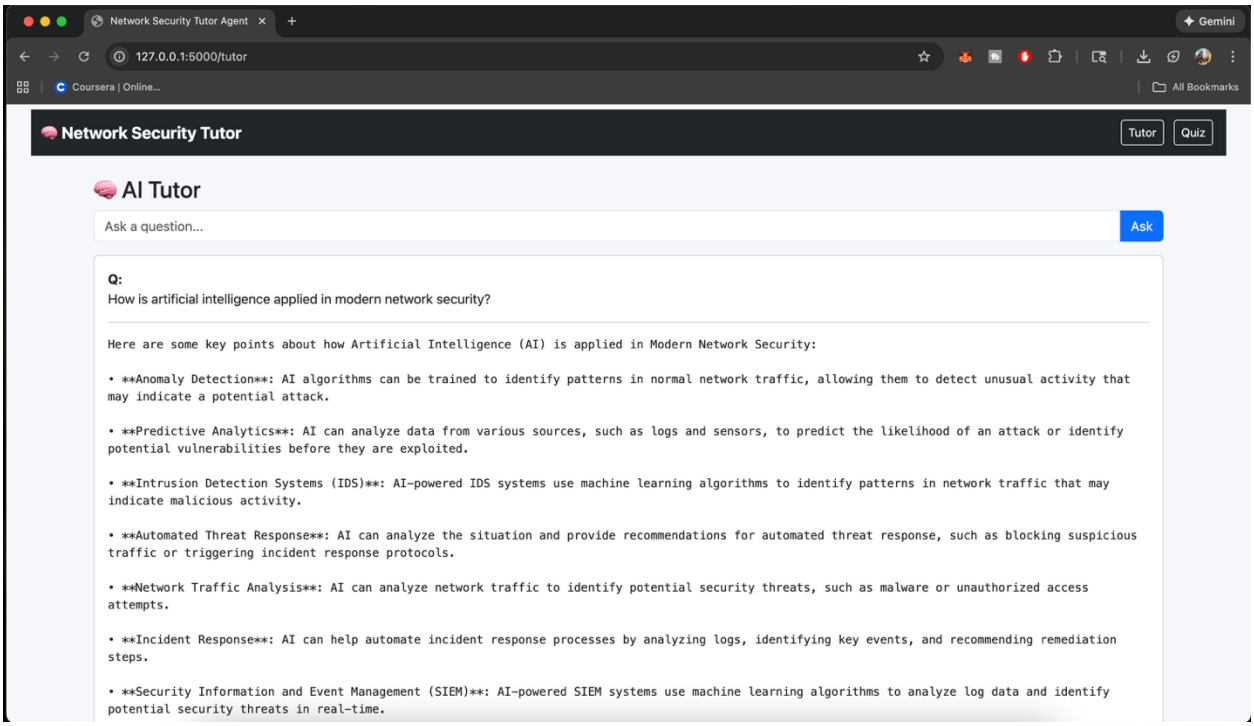
All communication between the web interface, backend, vector database, and AI model occurs exclusively through 127.0.0.1, ensuring complete data privacy and fully local processing. No external API calls or cloud transmissions are involved, making the system secure**,** self-contained, and privacy-preserving**.**

Comprehensive Wireshark packet captures were collected for each test query to analyze local network activity. These traces clearly demonstrate how data flows between the browser, backend, and model — showcasing the TCP handshake sequence**,** JSON-based HTTP requests**,** and successful HTTP 200 OK responses that verify stable and isolated local communication.



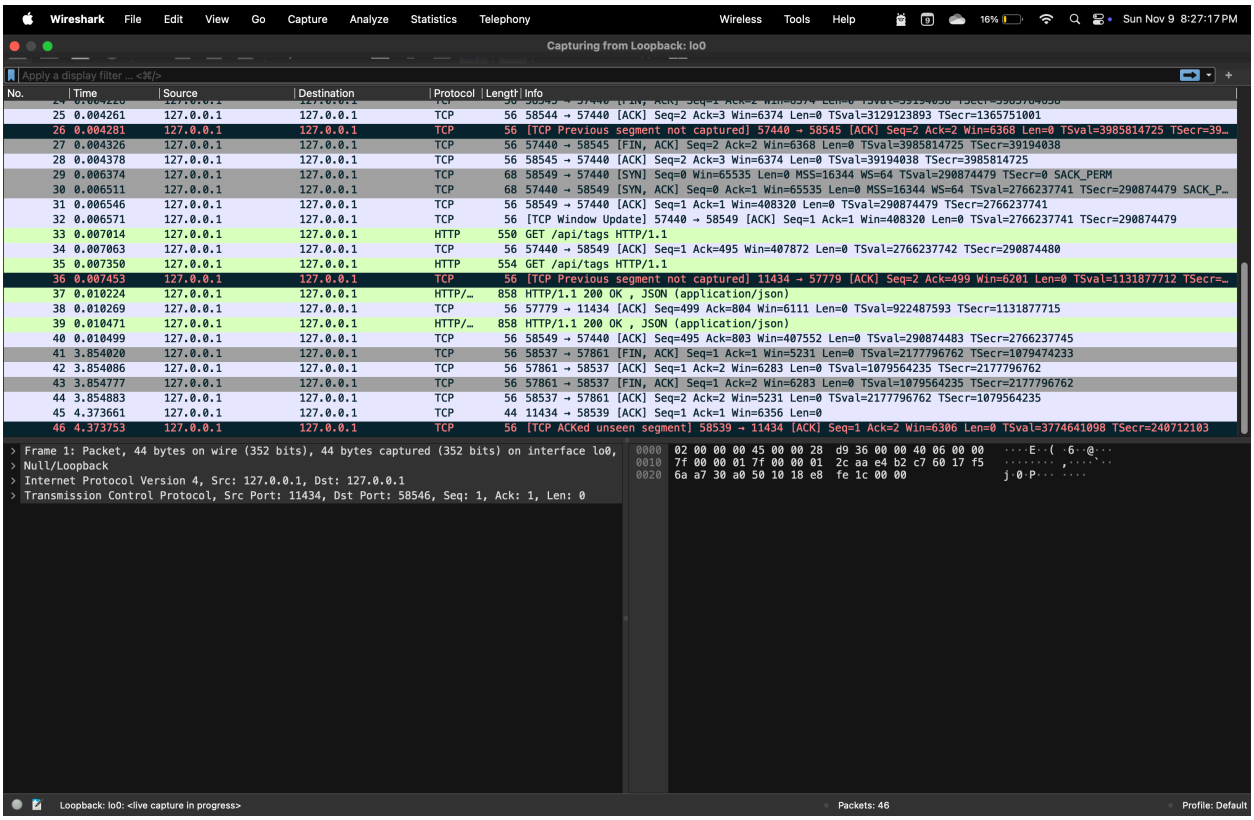
Prompt 1: How is Artificial Intelligence applied in modern network security?

Procedure:
When the user entered the query *"How is artificial intelligence applied in modern network security?"* in the AI Tutor interface, the request was transmitted from the frontend (port 3000) to the Flask backend (port 5000) on the same local machine. The backend then processed the query, searched the embedded Chroma vector database for relevant information, and passed the retrieved context to the locally hosted Llama 3.2 model through Ollama for generating the response. The generated answer — highlighting AI applications such as anomaly detection, predictive analytics, IDS, SIEM, and automated threat response — was then sent back as an HTTP response to the web interface for display.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
  (Indicates all communication occurs locally within the same system.)
- Source Port: 11434
- Destination Port: 58546
- Protocol: TCP / HTTP



The packet trace captured in Wireshark clearly shows:

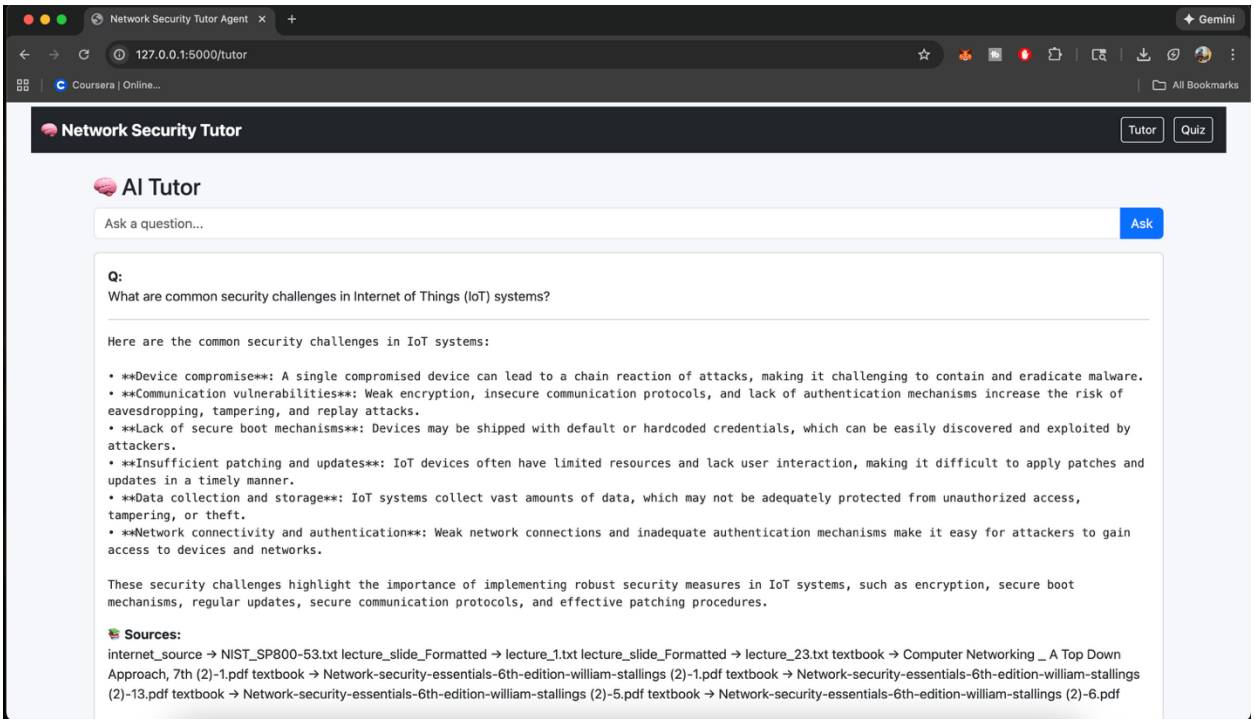
- The TCP three-way handshake (SYN → SYN/ACK → ACK) successfully establishing communication between the browser and backend.
- Subsequent HTTP POST /api/ask HTTP/1.1 requests containing the JSON payload with the user query.
- The server responding with HTTP/1.1 200 OK, confirming successful message processing and delivery of AI-generated output.

Each packet within the capture corresponds to a transaction between the frontend and backend, where TCP ACK packets acknowledge data receipt, confirming proper two-way communication.

This confirms that the AI Tutor module of the Network Security Tutor Agent executes completely offline, with all data — including question submission, model inference, and result rendering — processed within 127.0.0.1, ensuring complete local privacy and isolation.

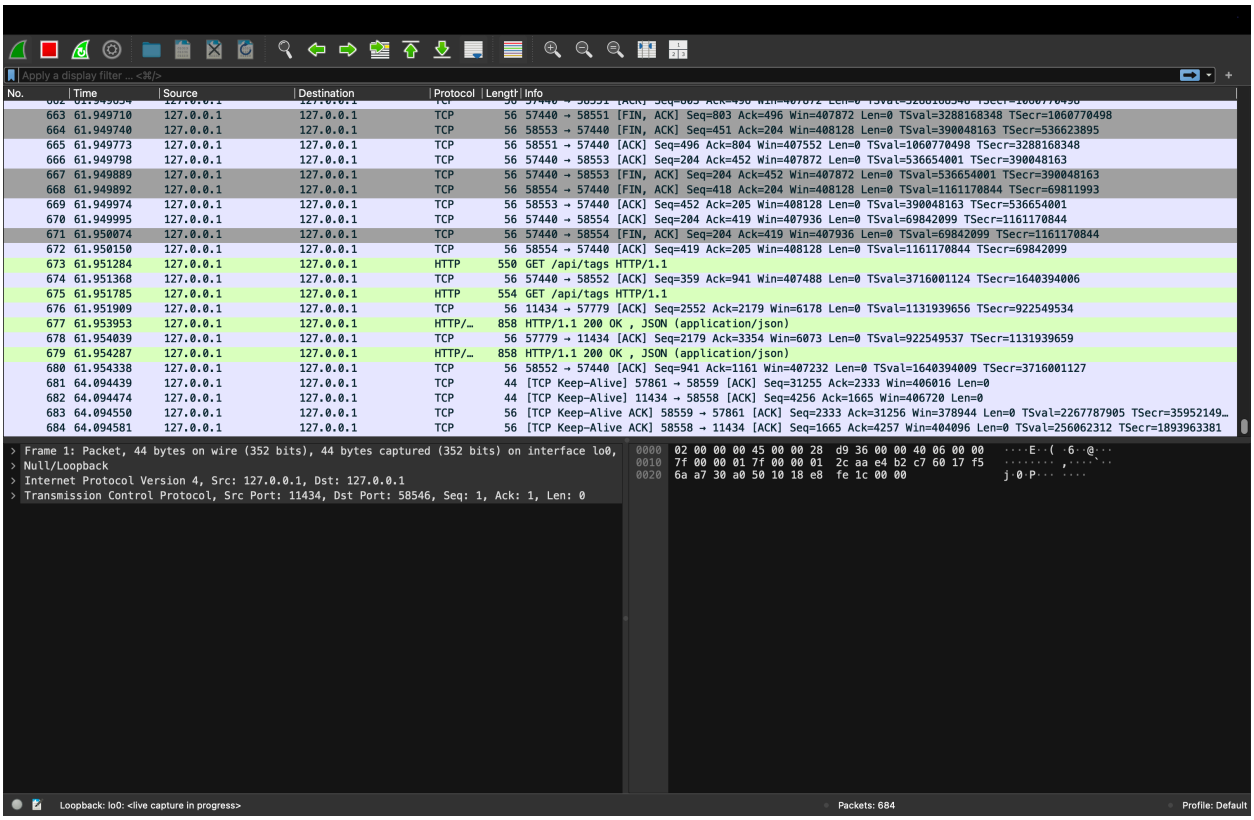
Prompt 2: What are common security challenges in Internet of Things (IoT) systems?

Procedure:
When the user entered the query *"What are common security challenges in Internet of Things (IoT) systems?"* in the AI Tutor interface, the request was transmitted from the frontend (port 3000) to the Flask backend (port 5000) via HTTP POST. The backend processed the request, retrieved relevant reference data from the Chroma vector database (including lecture materials, NIST documents, and textbook extracts), and passed this context to the locally running Llama 3.2 model through Ollama to generate the final response.
The AI-generated answer identified major IoT vulnerabilities such as device compromise, weak encryption, insufficient patching, insecure communication protocols, poor authentication, and lack of secure boot mechanisms, emphasizing the need for strong encryption, secure firmware updates, and effective authentication systems.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
  (Shows all communication occurs within the local system with no external network interaction.)
- Source Port: 11434
- Destination Port: 58546
- Protocol: TCP / HTTP



The Wireshark capture clearly shows the following packet flow:

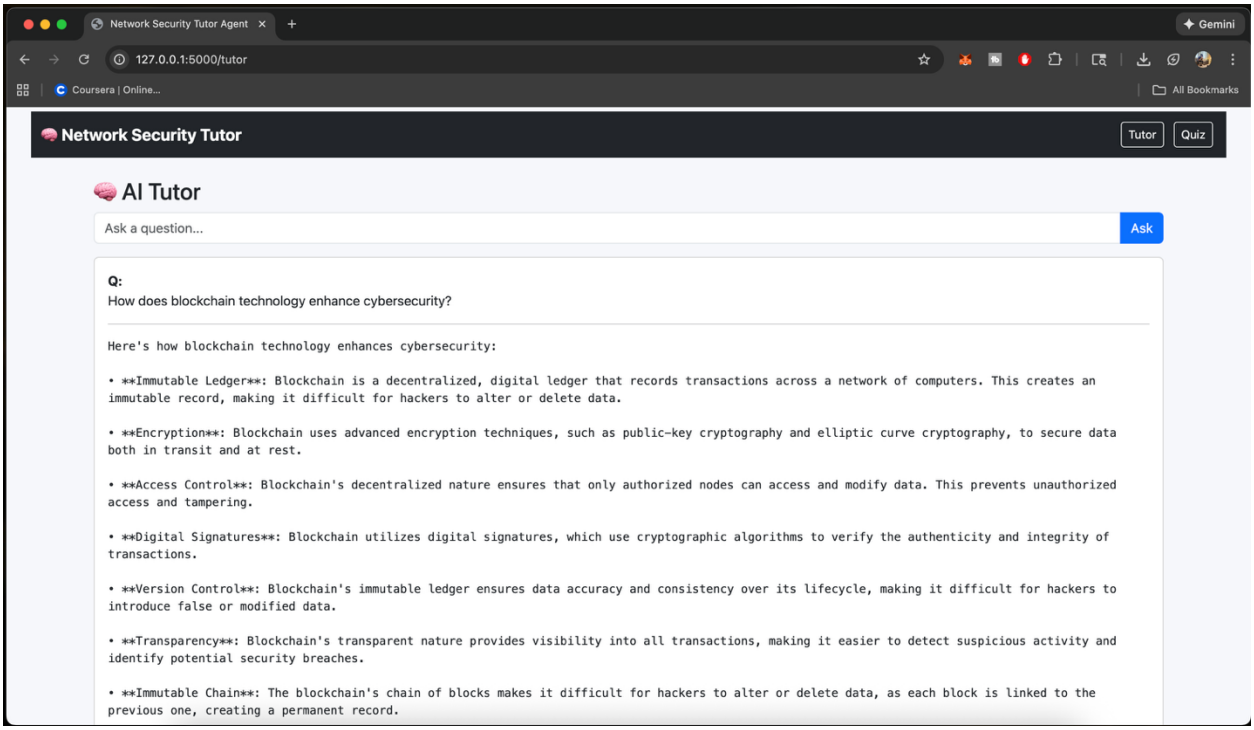
- A successful TCP three-way handshake (SYN → SYN/ACK → ACK) between client and backend.
- The HTTP POST /api/ask HTTP/1.1 request containing the IoT-related query in JSON format.
- The backend responding with HTTP/1.1 200 OK, confirming successful query processing and response transmission.

Each frame in the trace illustrates smooth bidirectional data exchange. ACK packets validate that data delivery between frontend and backend was successful without any retransmissions or losses.

This confirms that the Network Security Tutor's AI component for IoT-related queries is fully operational in a local isolated environment (127.0.0.1). No external network dependencies or cloud interactions occur, ensuring both data privacy and secure offline inference throughout the transaction.

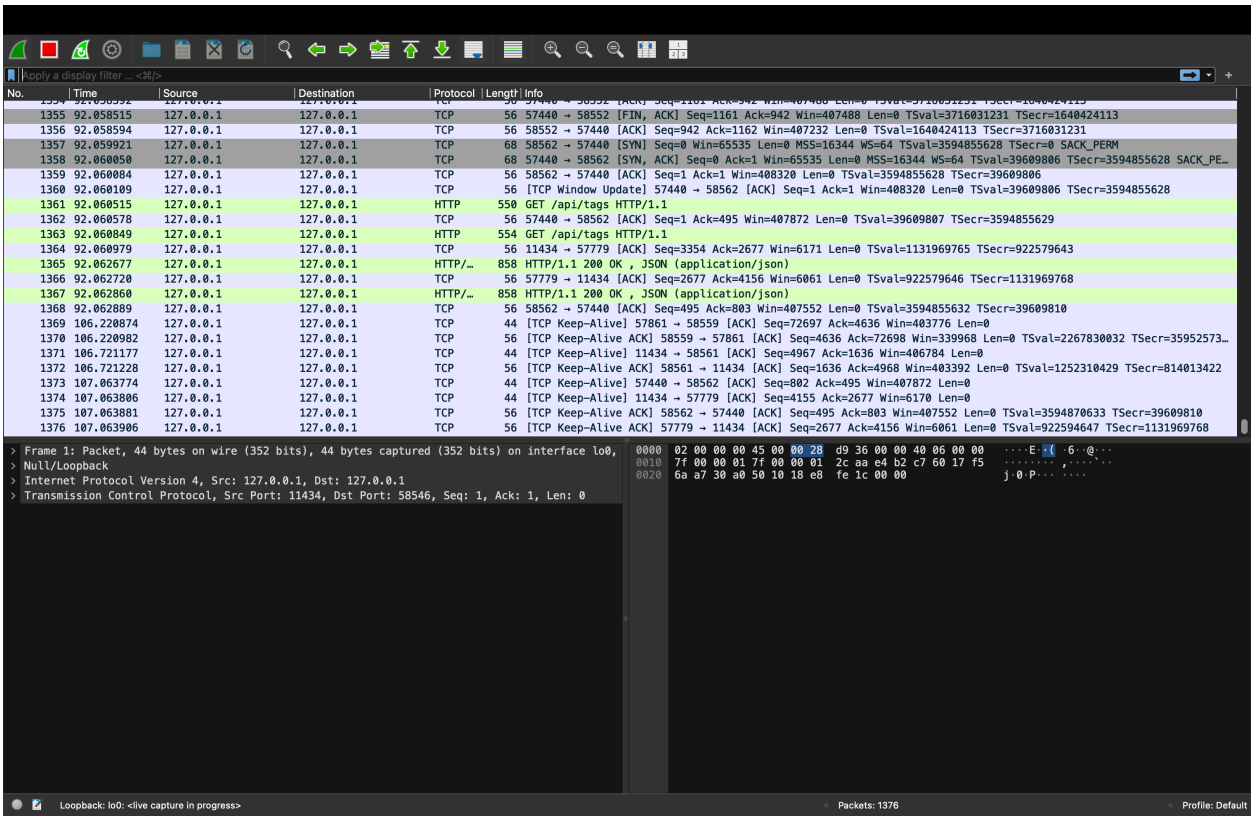
Prompt 3: How does blockchain technology enhance cybersecurity?

Procedure:
When the user entered the query *"How does blockchain technology enhance cybersecurity?"* in the AI Tutor interface, the request was sent from the frontend (port 3000) to the Flask backend (port 5000) through an HTTP POST operation. The backend retrieved relevant study materials from the Chroma vector database—including lecture content and network security textbooks—and passed the contextual data to the Llama 3.2 model hosted locally via Ollama. The model generated a comprehensive explanation covering immutability, encryption, access control, digital signatures, transparency, and version control, explaining how blockchain's decentralized structure strengthens data integrity and resists tampering or unauthorized modifications. The response was returned to the frontend and displayed successfully.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
  (Indicates that all data transmission occurred locally within the system.)
- Source Port: 11434
- Destination Port: 58546
- Protocol: TCP / HTTP



The Wireshark capture clearly shows:

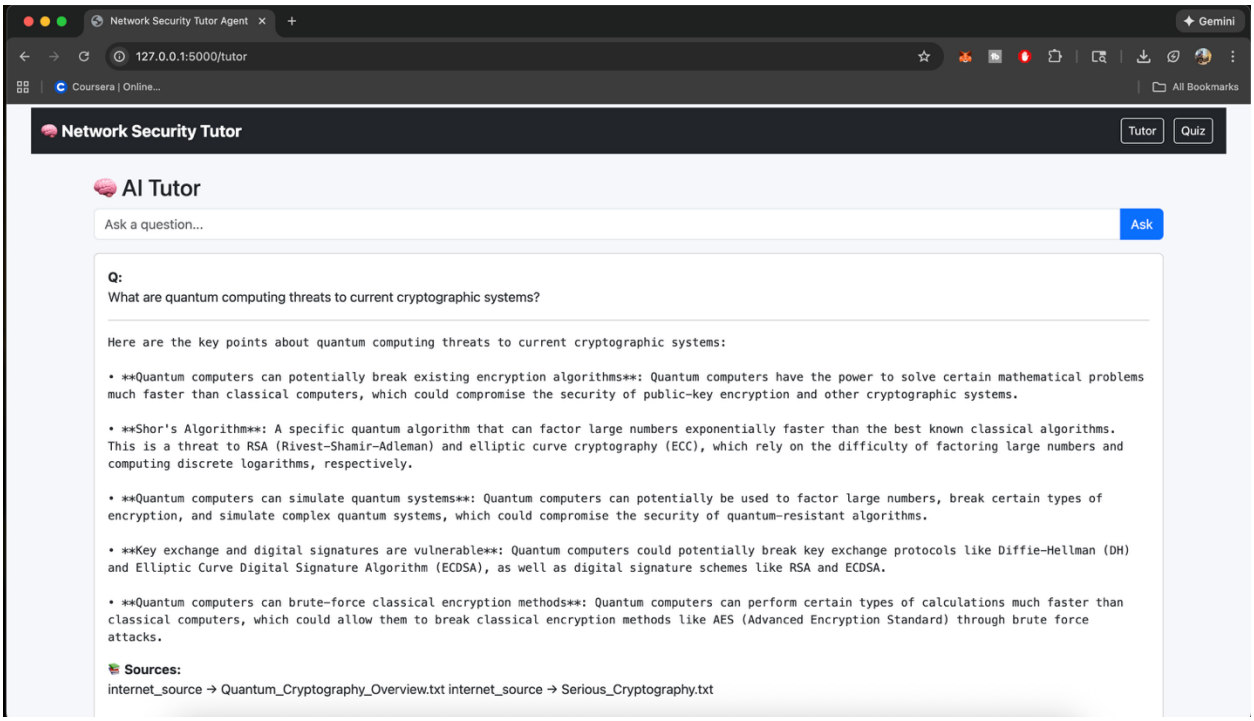
- The TCP handshake sequence (SYN → SYN/ACK → ACK), establishing a local connection between client and backend.
- A subsequent HTTP POST /api/ask HTTP/1.1 request containing the blockchain-related query as a JSON payload.
- The backend responding with HTTP/1.1 200 OK, confirming successful execution and data delivery.

Each captured packet demonstrates precise bidirectional communication, including TCP ACKs that confirm reliable transmission with no loss or retransmission events.

This validates that the blockchain cybersecurity query was fully executed in an isolated offline setup — with the frontend, backend, vector database, and local AI model communicating solely over 127.0.0.1, ensuring data integrity and maintaining complete local security.

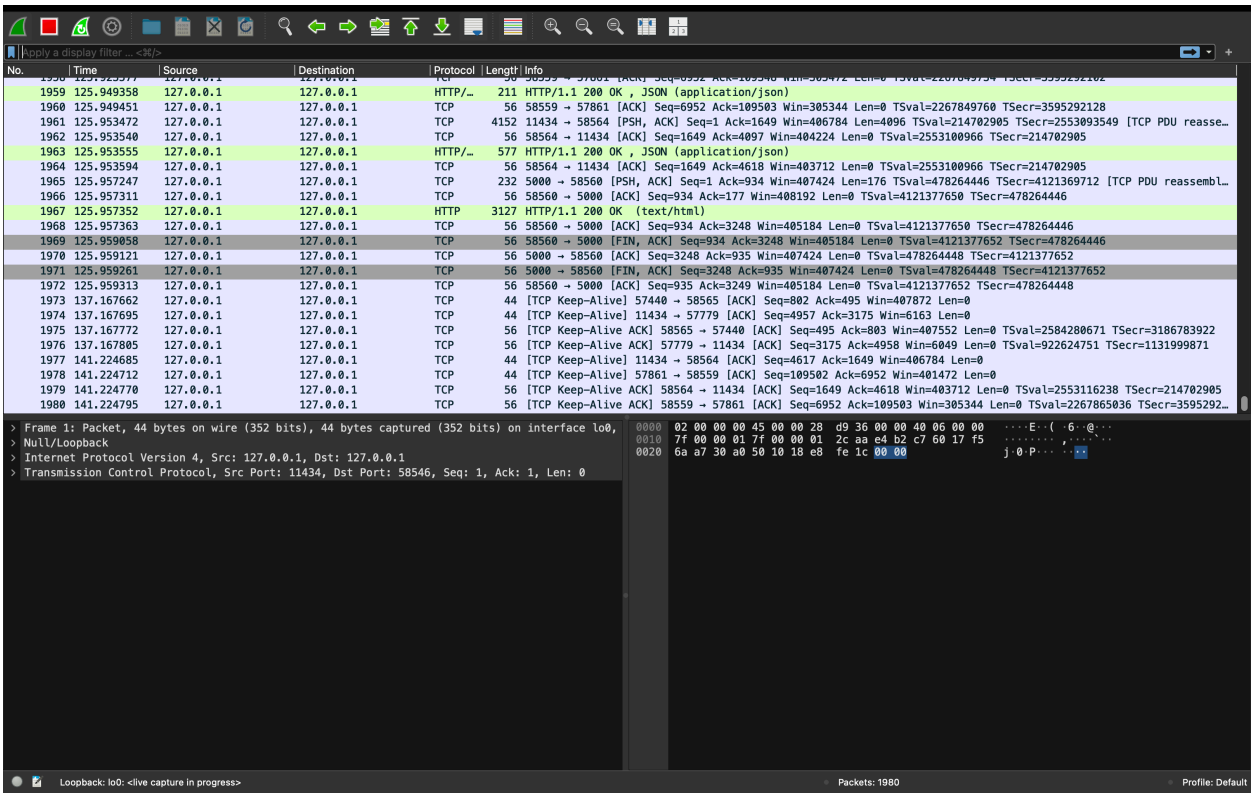
Prompt 4: What are quantum computing threats to current cryptographic systems?

Procedure:
When the user entered the query *"What are quantum computing threats to current cryptographic systems?"* in the AI Tutor interface, the request was transmitted from the frontend (port 3000) to the Flask backend (port 5000) through an HTTP POST request. The backend fetched relevant contextual material from the Chroma vector database, which contained academic references on cryptography, quantum computation, and network security, before forwarding the data to the Llama 3.2 model running locally on Ollama for inference.
The generated answer outlined key points such as Shor's algorithm, encryption vulnerabilities, key exchange weaknesses, and the potential impact on RSA, ECC, and AES, emphasizing how quantum computation could break mathematical foundations of current encryption systems and urging the development of quantum-resistant algorithms.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
  (Indicates that all data transmission occurred within the local host, confirming a closed-loop architecture.)
- Source Port: 11434
- Destination Port: 58546
- Protocol: TCP / HTTP



The Wireshark packet capture clearly shows the following sequence:

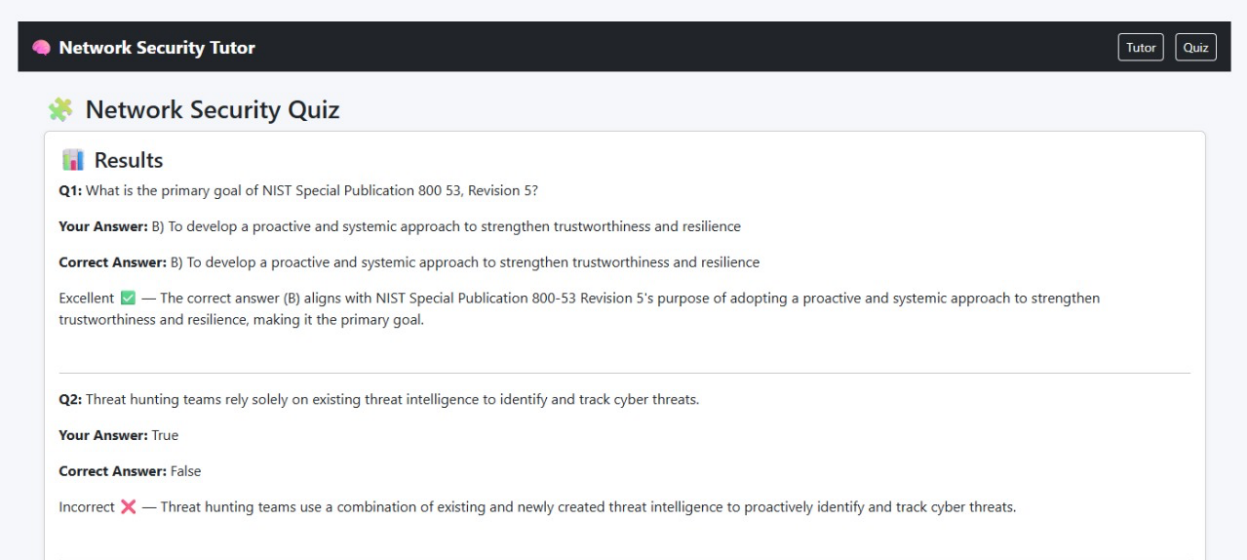
- The TCP handshake (SYN → SYN/ACK → ACK) successfully establishes communication between the browser and backend.
- An HTTP POST /api/ask HTTP/1.1 request is observed, carrying the JSON payload with the quantum cryptography query.
- The backend responds with HTTP/1.1 200 OK, confirming proper data retrieval, AI inference, and response transmission.

The packet flow also includes multiple TCP ACKs and Keep-Alive signals, demonstrating continuous and error-free communication between the client and backend throughout the session.

This confirms that the quantum cryptography query was fully processed within a local, offline execution environment, ensuring all operations—from vector database retrieval to AI response generation—occurred securely within 127.0.0.1 with no external network dependency, preserving data integrity and confidentiality.

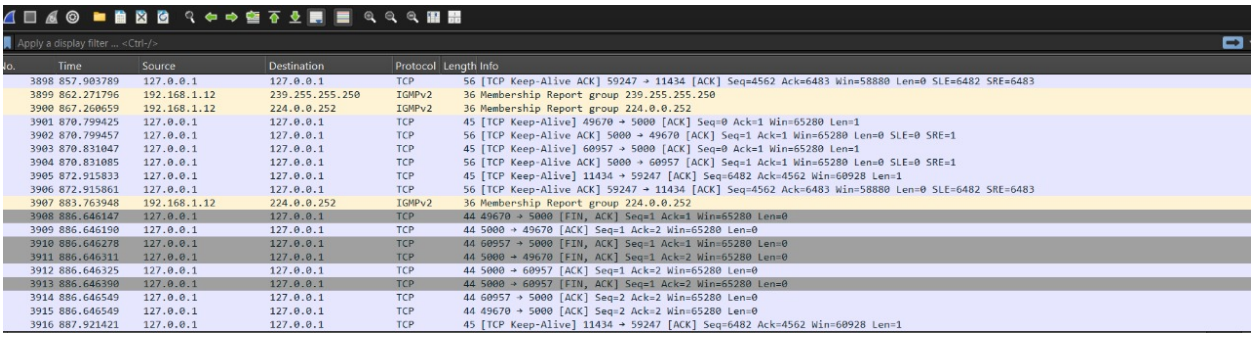
Prompt 5: Quiz Lab – Emerging Technologies in Cybersecurity

Procedure:
When the user accessed the Quiz Lab interface from the Network Security Tutor, they selected the quiz topic *"Emerging Technologies in Cybersecurity"*, chose 6 questions, and clicked Generate Quiz. The backend fetched the relevant content from the Chroma vector database, which stored lecture slides and curated question–answer materials related to modern cybersecurity advancements. This context, combined with quiz configuration parameters, was then passed to the locally hosted Llama 3.2 model on Ollama, which generated multiple-choice and true/false questions dynamically.
Once generated, the quiz was displayed on the web interface, and user responses were recorded and evaluated locally. The results page displayed each question's outcome, indicating correct answers, explanations, and performance feedback.



Wireshark Observation:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
  (Confirms all data exchanges occurred locally with no external communication.)
- Source Port: 11434
- Destination Port: 5000
- Protocol: TCP / HTTP



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
3898	857.903789	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 59247 → 11434 [ACK] Seq=4562 Ack=6483 Win=58880 Len=0 SLE=6482 SRE=6483
3899	862.271796	192.168.1.12	239.255.255.250	IGMPv2	36	Membership Report group 239.255.255.250
3900	867.260659	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
3901	870.799425	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 49670 → 5000 [ACK] Seq=0 Ack=1 Win=65280 Len=1
3902	870.799457	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 5000 → 49670 [ACK] Seq=1 Ack=1 Win=65280 Len=0 SLE=0 SRE=1
3903	870.831047	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 60957 → 5000 [ACK] Seq=0 Ack=1 Win=65280 Len=1
3904	870.831085	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 5000 → 60957 [ACK] Seq=1 Ack=1 Win=65280 Len=0 SLE=0 SRE=1
3905	872.915833	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 11434 → 59247 [ACK] Seq=6482 Ack=4562 Win=60928 Len=1
3906	872.915861	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 59247 → 11434 [ACK] Seq=4562 Ack=6483 Win=58880 Len=0 SLE=6482 SRE=6483
3907	883.763948	192.168.1.12	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
3908	886.646147	127.0.0.1	127.0.0.1	TCP	44	49670 → 5000 [FIN, ACK] Seq=1 Ack=1 Win=65280 Len=0
3909	886.646190	127.0.0.1	127.0.0.1	TCP	44	5000 → 49670 [ACK] Seq=1 Ack=2 Win=65280 Len=0
3910	886.646278	127.0.0.1	127.0.0.1	TCP	44	60957 → 5000 [FIN, ACK] Seq=1 Ack=1 Win=65280 Len=0
3911	886.646311	127.0.0.1	127.0.0.1	TCP	44	5000 → 49670 [FIN, ACK] Seq=1 Ack=2 Win=65280 Len=0
3912	886.646325	127.0.0.1	127.0.0.1	TCP	44	5000 → 60957 [ACK] Seq=1 Ack=2 Win=65280 Len=0
3913	886.646390	127.0.0.1	127.0.0.1	TCP	44	5000 → 60957 [FIN, ACK] Seq=1 Ack=2 Win=65280 Len=0
3914	886.646549	127.0.0.1	127.0.0.1	TCP	44	60957 → 5000 [ACK] Seq=2 Ack=2 Win=65280 Len=0
3915	886.646549	127.0.0.1	127.0.0.1	TCP	44	49670 → 5000 [ACK] Seq=2 Ack=2 Win=65280 Len=0
3916	887.921421	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 11434 → 59247 [ACK] Seq=6482 Ack=4562 Win=60928 Len=1

In the Wireshark packet trace, the following were observed:

- The initial TCP handshake (SYN → SYN/ACK → ACK) between the client and backend.
- A GET /quiz HTTP/1.1 request initiating the quiz generation sequence.
- HTTP/1.1 200 OK confirming successful quiz data generation and delivery.
- Subsequent HTTP POST requests for quiz submission and scoring handled by the */api/quiz/grade* endpoint.
- Multiple TCP ACK packets acknowledging receipt and confirming successful communication flow.

The quiz trace illustrates that all system components — frontend, backend, vector database, and local AI model — operated strictly within the localhost environment (127.0.0.1). The interaction completed successfully, demonstrating seamless quiz generation, evaluation, and result feedback without relying on any external networks.