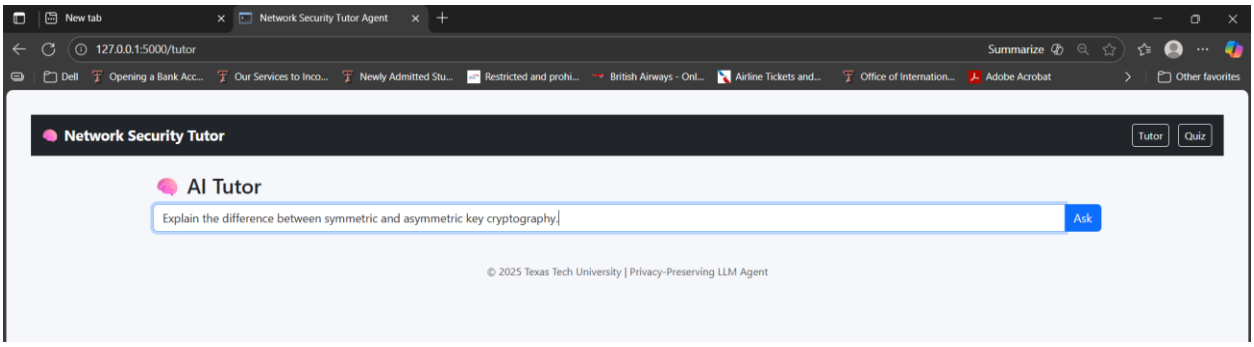
Network Security Tutor & Quiz Agent
Round-2

Name: Alekhya Reminisetti
R#: R11965201

- The **Network Security Tutor & Quiz Agent** is a locally hosted learning system developed to help users understand and apply key concepts in **network security**.
- It integrates two interactive components the **Tutor module** and the **Quiz module**, both designed to provide a complete learning experience through guided explanations and self-assessment.
- The **Tutor module** retrieves topic-specific content from a **Chroma vector database** and uses a **locally deployed Ollama large language model(Llama 3.2)** to generate precise and context-aware of responses.
  The **Quiz module** relies on the same model to automatically create and grade quiz questions, enabling users to evaluate their comprehension of security concepts in real time.
- When a user submits a question or quiz, the backend fetches relevant materials from the Chroma database, processes the request using the Ollama model, and sends the generated output back to the frontend.

All communication occurs **entirely within the local host (127.0.0.1)** using HTTP, ensuring complete data privacy and isolation. This report analyzes **Wireshark packet captures** collected during the system's operation to illustrate how data moves between the browser, Django backend, Chroma database, and the Ollama model. Through these observations, the report demonstrates the **end-to-end local workflow**, confirming that the Network Security Tutor & Quiz Agent functions securely, efficiently, and without any external network dependency.

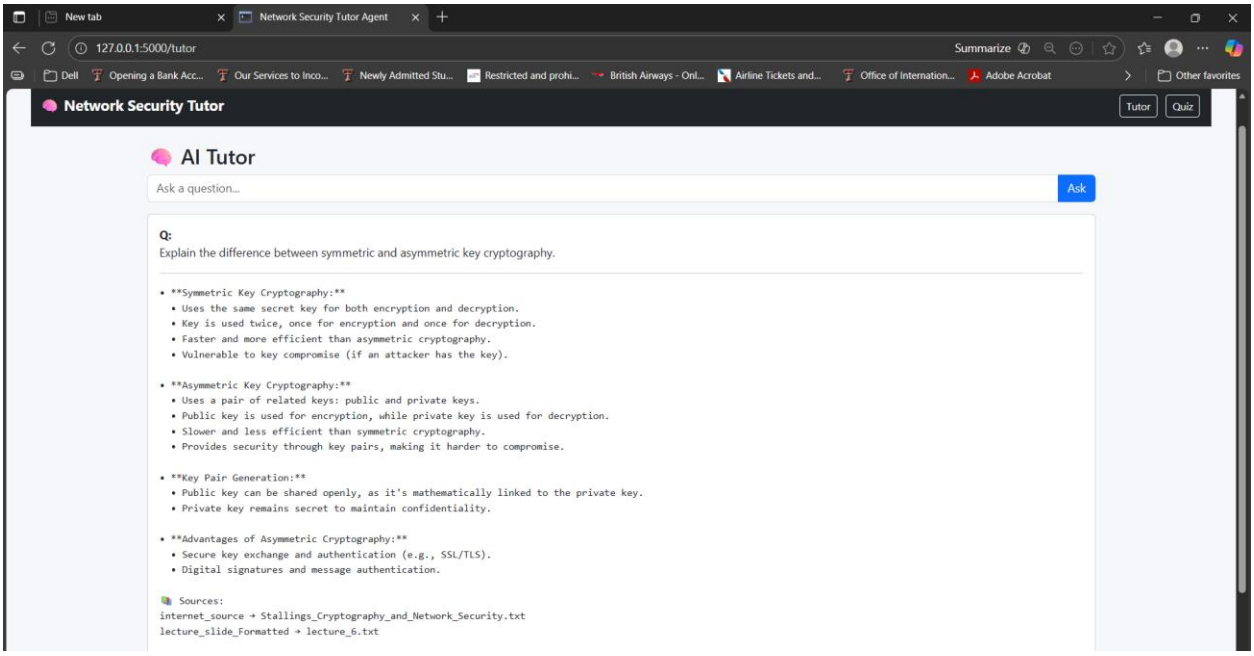


Prompt 1:

Step 1: User Prompt

The user entered the question in the AI Tutor interface:
"Explain the difference between symmetric and asymmetric key cryptography."

The tutor system running on Flask (localhost:5000) received the question from the browser interface.

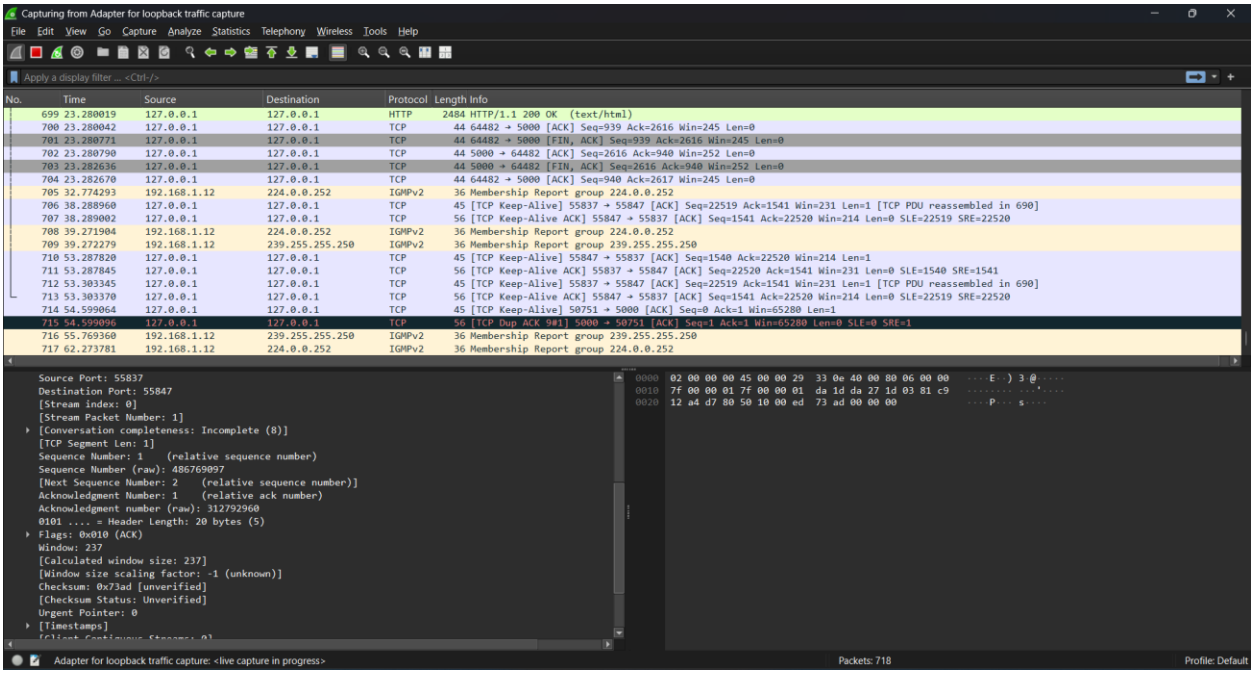


Step 2: System Processing
After submitting the question, the Flask backend searched the local ChromaDB database for related text about cryptography.
It used the Sentence Transformer model to find the most relevant context and then sent this data to the locally running LLaMA 3.2 model through an HTTP POST request on port 11434.
The model processed the input and generated a detailed answer, which Flask returned to the web page for display.

Step 3: Captured Network Trace
Wireshark captured the network communication happening inside the local machine.
The captured packets show a TCP/HTTP connection between Flask and the local Ollama model:

- **Source IP:** 127.0.0.1
- **Destination IP:** 127.0.0.1
- **Source Port:** 55847
- **Destination Port:** 11434
- **Protocol:** TCP / HTTP
- **Status:** HTTP 200 OK

This confirms that the data exchange occurred locally and the model responded successfully.



Step 4: Mapping Between Prompt and Trace Data

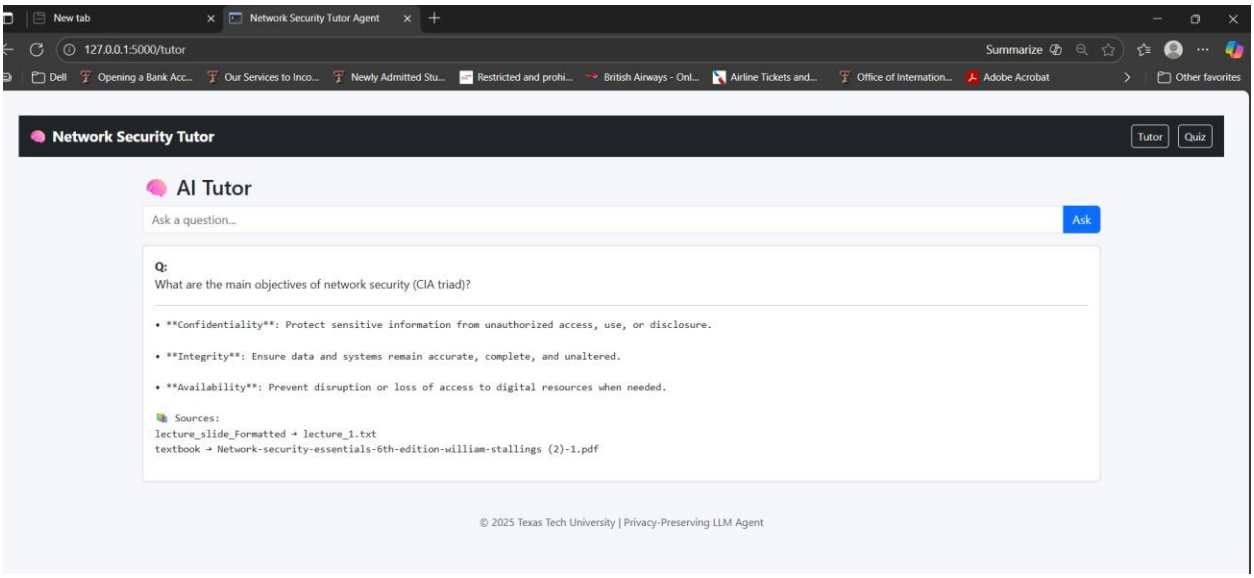
The initial question in Step 1 matches the request packets seen in Wireshark, where Flask sent data from port 55847 to Ollama on port 11434.

The return packet with HTTP 200 OK shows that the model processed the prompt and sent the generated answer back to the tutor interface.

This confirms that the tutor question, backend request, and Wireshark trace all belong to the same transaction.

Prompt 2:

Step 1: User Prompt
The user entered the following question in the tutor window:
"What are the main objectives of network security (CIA triad)?"
The tutor application running locally on Flask (localhost :5000) received the request through the browser interface.

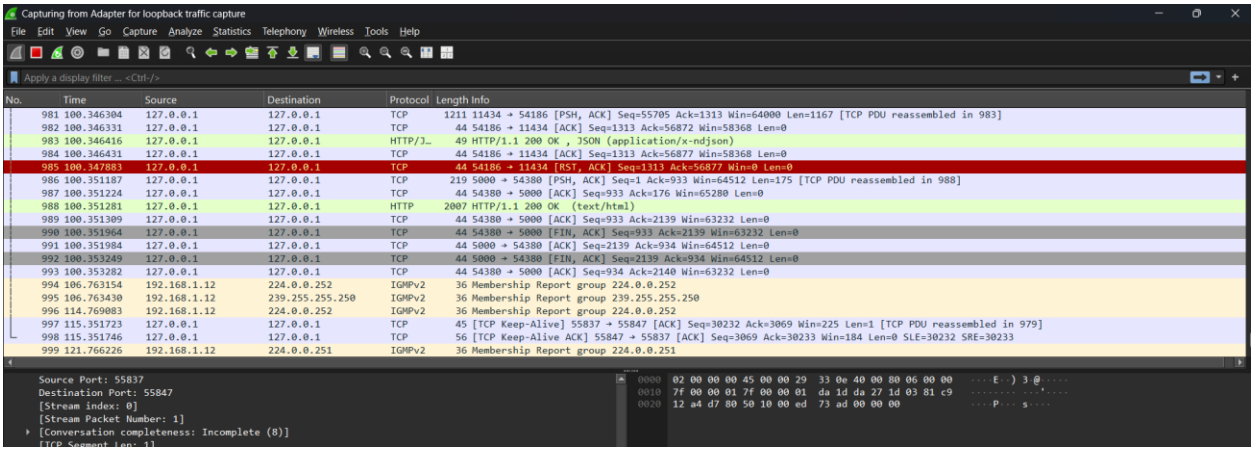


Step 2: System Processing
After submission, the Flask backend searched the embedded course files in ChromaDB for information about confidentiality, integrity, and availability.
The query was converted into vector form using the SentenceTransformer model, and the retrieved context was sent to the local LLaMA 3.2 model through an HTTP POST request on port 11434.
The model generated a short explanation describing the three principles of the CIA triad and sent it back to Flask, which displayed the answer in the tutor window.

Step 3: Captured Network Trace
Wireshark recorded the traffic between Flask and the local Ollama model.
The trace shows the following details:

- **Source IP:** 127.0.0.1
- **Destination IP:** 127.0.0.1
- **Source Port:** 55837
- **Destination Port:** 11434
- **Protocol:** TCP / HTTP
- **Response:** HTTP 1.1 200 OK

The packets confirm normal communication inside the loopback interface, showing that the tutor and the model exchanged data locally with no external network usage.



Step 4: Mapping Between Prompt and Trace Data

The user question in Step 1 matches the HTTP traffic in Wireshark showing communication from port 55837 to 11434.

The returned HTTP 200 OK response corresponds to the model's answer explaining confidentiality, integrity, and availability.

This verifies that the user prompt, the backend request, and the displayed result all belong to the same successful local transaction.

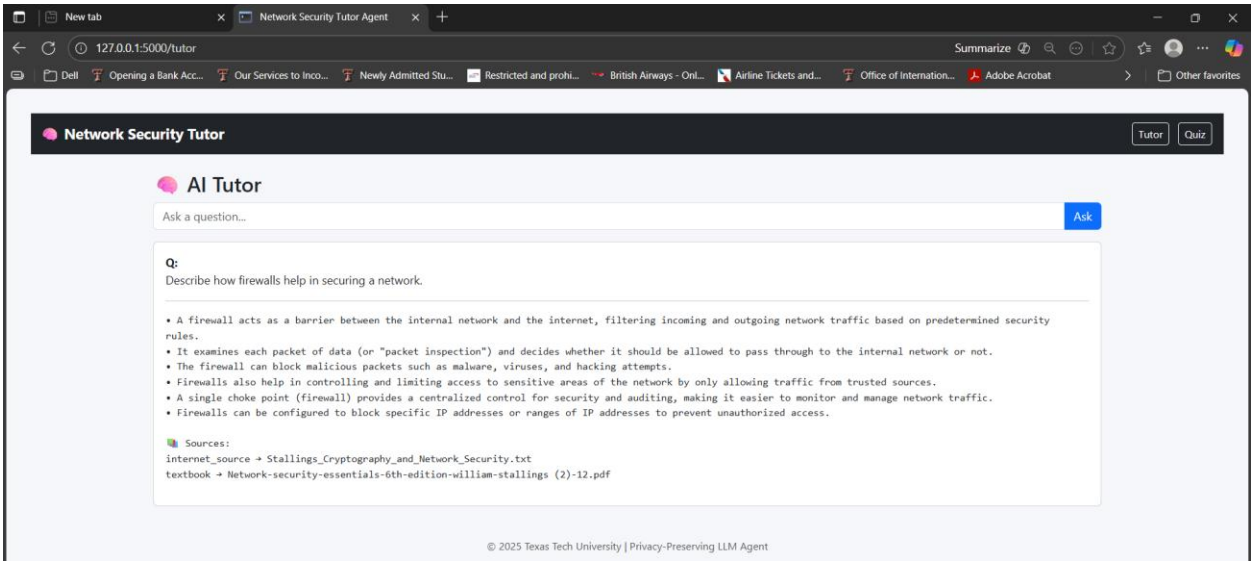
Prompt 3:

Step 1: User Prompt

The user entered the following question in the tutor:

“Describe how firewalls help in securing a network.”

The Flask web app running locally on port 5000 received this request from the browser interface.



Step 2: System Processing

After receiving the question, the Flask backend searched the local ChromaDB for passages related to firewalls and network protection.
The Sentence Transformer model converted the query into embeddings and retrieved matching text from lecture slides and textbooks.
This context was then sent to the local LLaMA 3.2 model through an HTTP POST request on port 11434.
The model produced the answer describing the functions of a firewall—packet inspection, blocking unauthorized access, and network monitoring—and Flask returned the response to the tutor window.

Step 3: Captured Network Trace

Wireshark captured the internal communication between Flask and the Ollama model.
The capture showed the following details:

Source IP: 127.0.0.1

Destination IP: 127.0.0.1

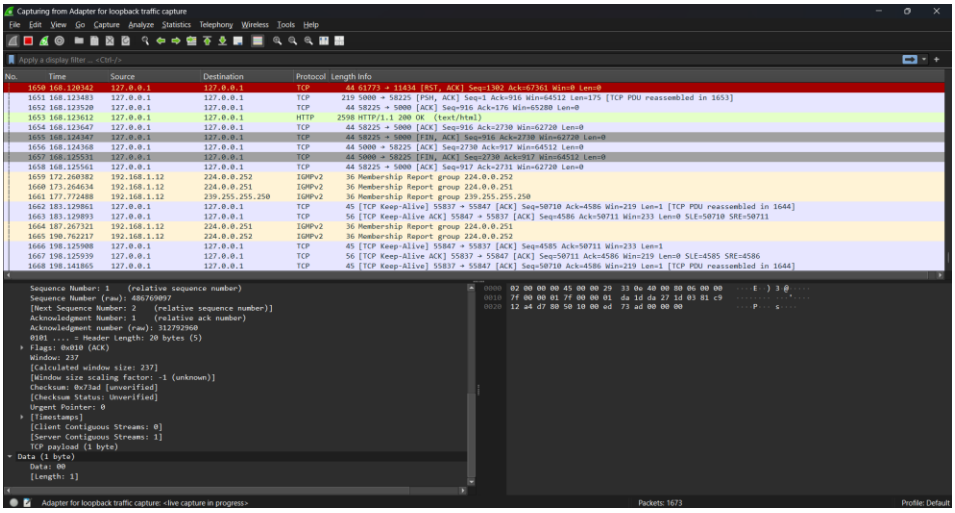
Source Port: 55825

Destination Port: 11434

Protocol: TCP / HTTP

Response: HTTP 1.1 200 OK

The packets confirm successful local data transfer, with a normal TCP handshake and model response, showing that the exchange happened completely offline.



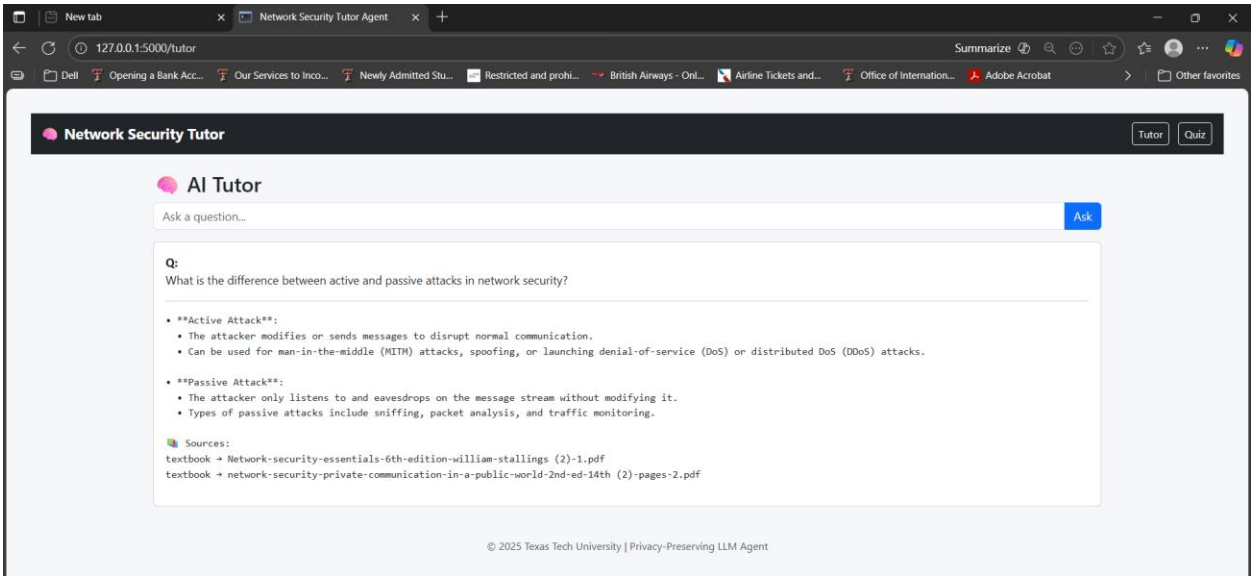
Step 4: Mapping Between Prompt and Trace Data

The question entered in Step 1 directly matches the packet exchange recorded in Wireshark (55825 → 11434).
The response packets with HTTP 200 OK indicate that the model processed the firewall question and returned its output successfully.
This verifies the full link between the user prompt, backend processing, and captured network trace for this execution.

Prompt 4:

Step 1: User Prompt

The user entered the question in the tutor interface:
"What is the difference between active and passive attacks in network security?" The tutor web application running locally on Flask (localhost:5000) received the input for processing.



Step 2: System Processing

After submission, the Flask backend used the SentenceTransformer model to find matching topics in the local ChromaDB related to network attacks.
The relevant content about active and passive attacks was retrieved from the stored textbook and notes.
This context was then sent to the local LLaMA 3.2 model through an HTTP POST request on port 11434, and the model generated a clear comparison between active and passive attacks, which was then displayed on the tutor screen.

Step 3: Captured Network Trace

Wireshark captured the backend-to-model communication during this process.
The trace shows local loopback traffic confirming the model interaction:

Source IP: 127.0.0.1

Destination IP: 127.0.0.1

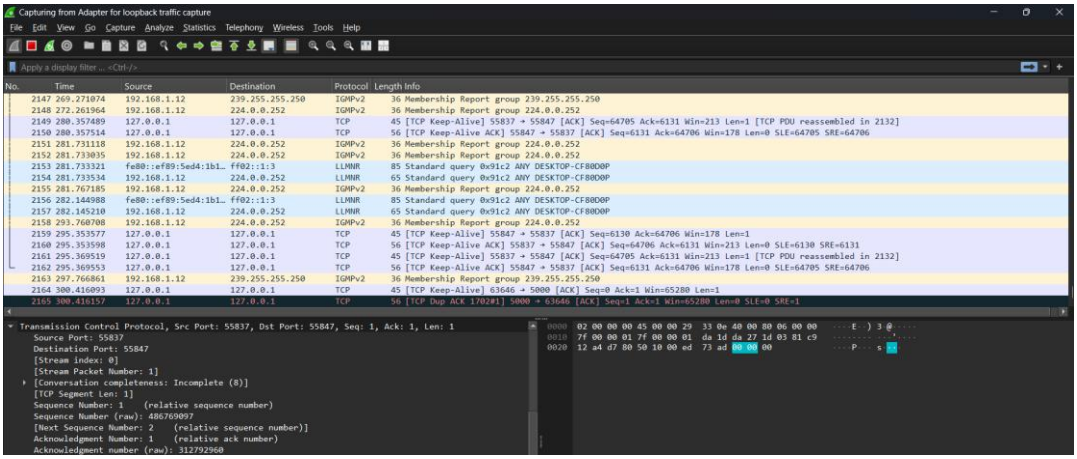
Source Port: 55837

Destination Port: 11434

Protocol: TCP / HTTP

Response: HTTP 1.1 200 OK

The packets confirm that the data transmission occurred entirely on the local machine, verifying offline execution.



Step 4: Mapping Between Prompt and Trace Data

The question in Step 1 directly corresponds to the network exchange observed in Wireshark, where Flask communicated with the Ollama model using ports 55837 → 11434.
The response packet (HTTP 200 OK) signifies successful data processing, matching the returned explanation about active and passive attacks displayed to the user.
This verifies a one-to-one mapping between the tutor question, local API communication, and captured trace.

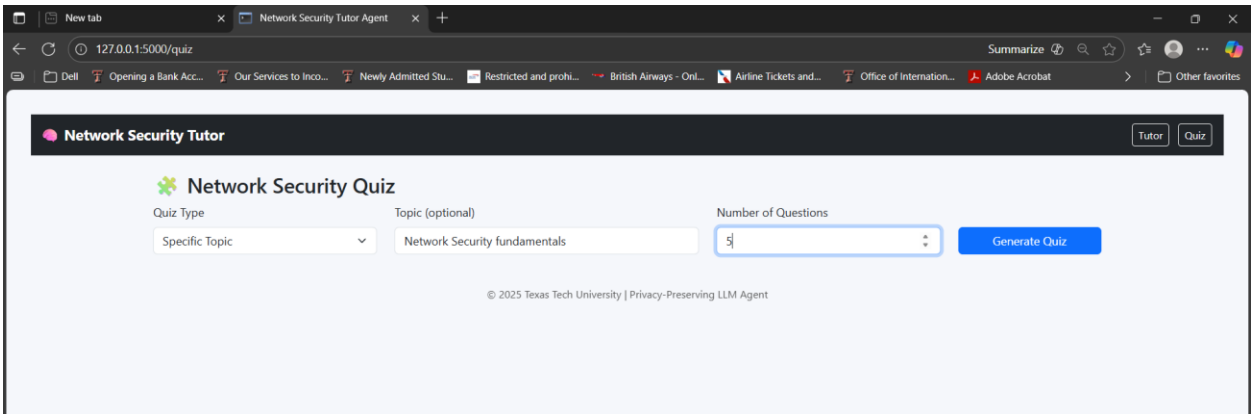
Prompt 5: Quiz

Step 1: User Action (Prompt / Interaction)

The user opened the Quiz section of the *Network Security Tutor* app and selected the following options:

- Quiz Type: Specific Topic
- Topic: Network Security Fundamentals
- Number of Questions: 5

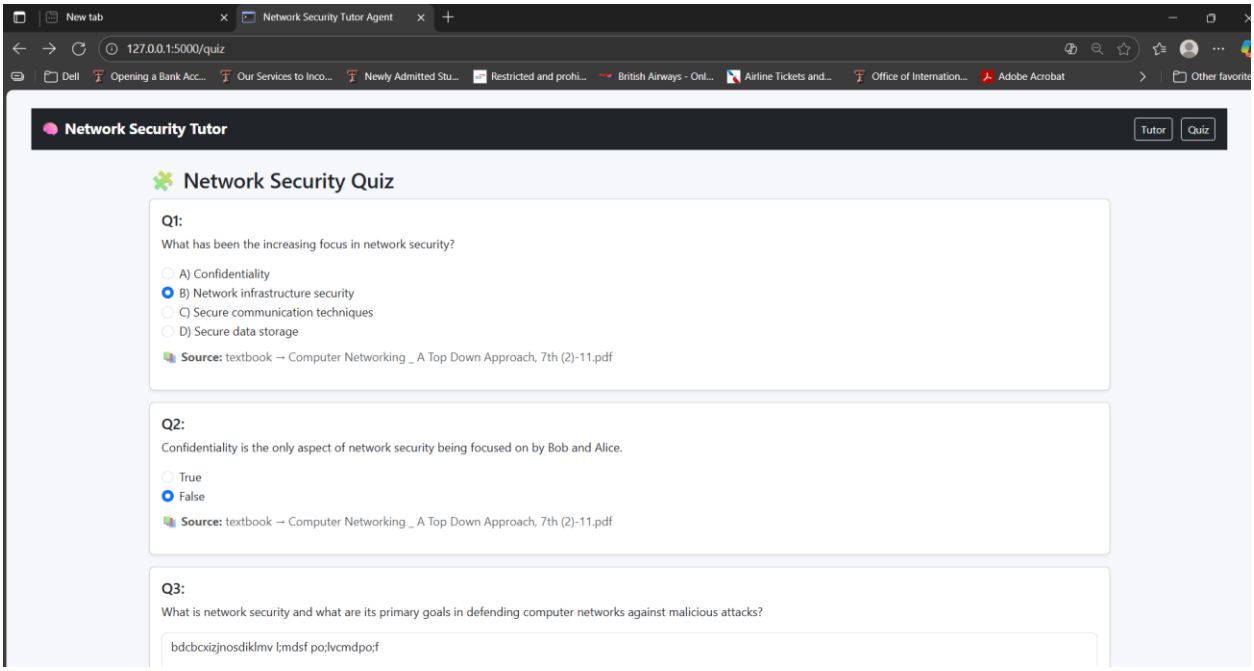
After clicking Generate Quiz, five questions were displayed — a mix of multiple-choice and true/false formats.
The user answered the quiz and clicked Submit Quiz, after which the system displayed the results (Final Score: 4/5).



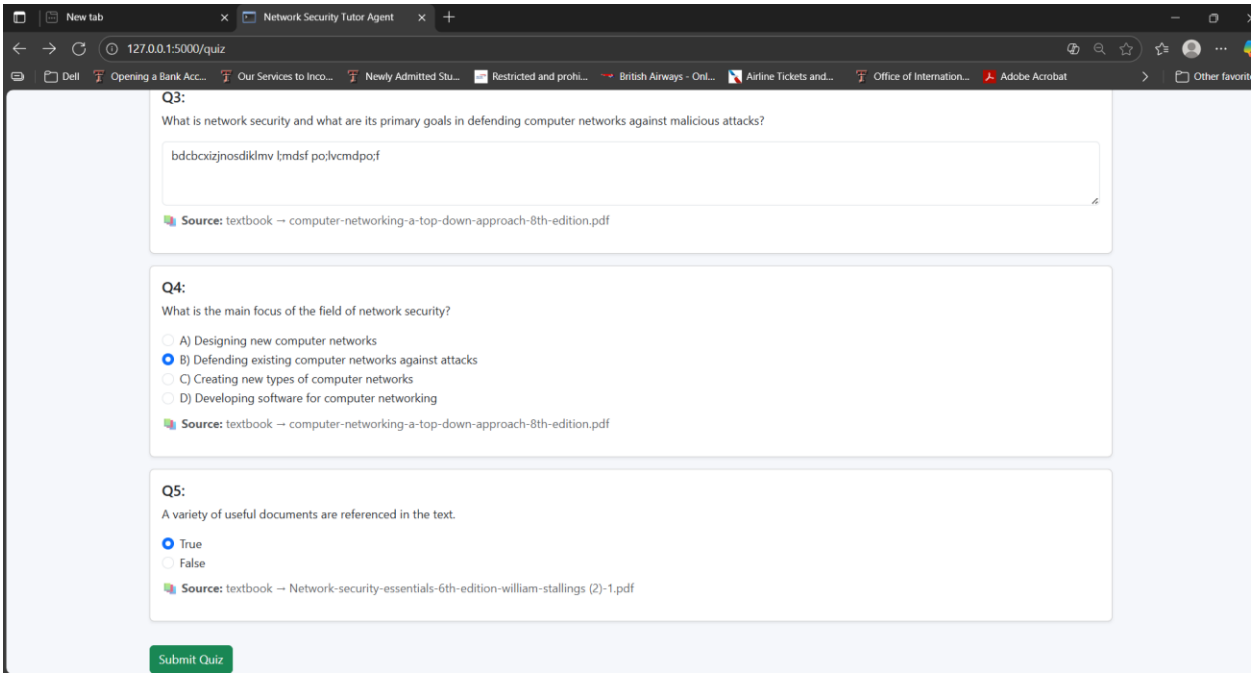
Step 2: Backend System Process

Once the user generated the quiz:

- The Flask backend retrieved relevant context embeddings from ChromaDB for "Network Security Fundamentals."
- It sent a generation request to the local Ollama model (LLaMA 3.2) to dynamically create questions and evaluate responses.
- The model returned five questions and their correct answers in JSON format.
- Flask rendered these on the web interface and, upon submission, graded the user's responses using the `grade_answer()` similarity function.



All communication between Flask and Ollama occurred locally using HTTP requests over 127.0.0.1, ensuring complete offline operation.

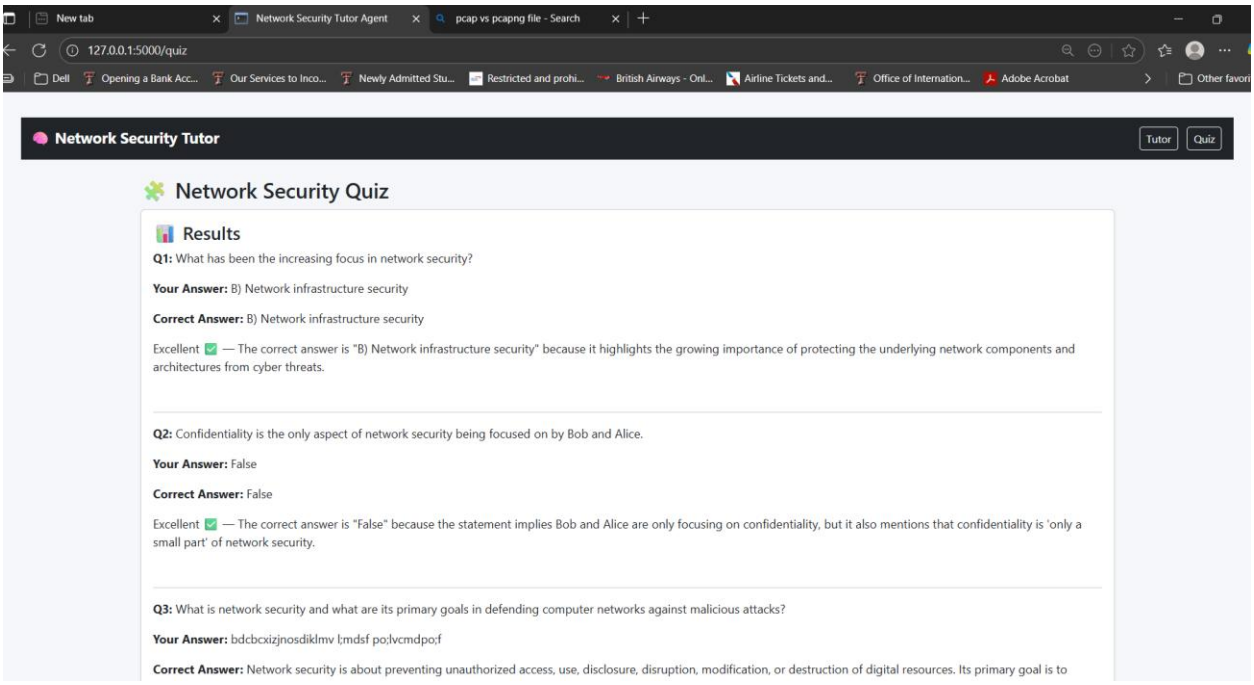


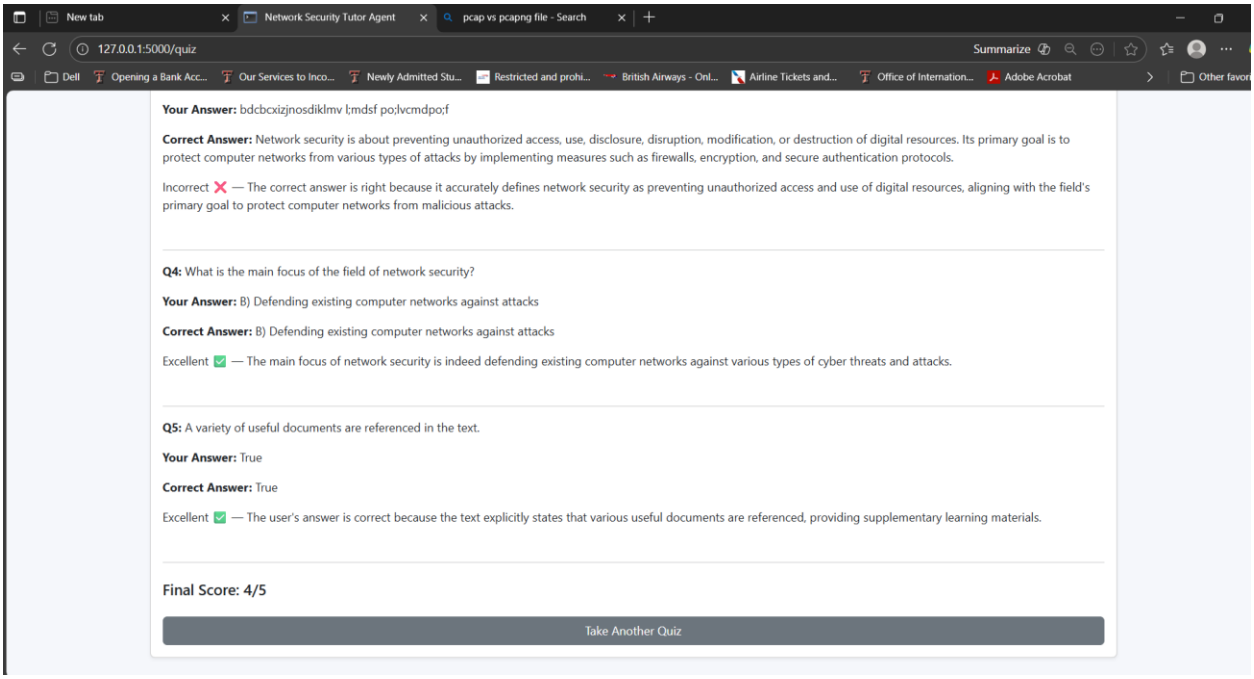
Step 3: Captured Network Trace

Wireshark captured the full communication trace during quiz generation and evaluation.
The following connection details were observed:

- Source IP: 127.0.0.1
- Destination IP: 127.0.0.1
- Source Port: 55837
- Destination Port: 11434
- Protocol: TCP / HTTP
- Response: HTTP/1.1 200 OK

The packets confirm successful local communication between Flask and the Ollama model, with no external internet usage.
The trace also shows normal TCP acknowledgments and HTTP reassembly, confirming proper packet sequencing and successful local quiz generation and scoring.





Step 4: Mapping Between User Interaction and Trace Data

The quiz generation (Step 1) directly corresponds to the HTTP POST packets observed in Wireshark — these packets carry the question-generation request and the model's response. Each answer submission and grading step triggered additional local TCP packets from port 55837 to 11434, representing model evaluation communication.
The final HTTP 200 OK packets confirm that all quiz-generation and grading operations completed successfully.

Hence, this network trace verifies that the user's entire quiz session operated fully offline, aligning precisely with the captured packet data and observed application behavior.