Secure AI Chatbot Documentation

Secure AI Chatbot Documentation

Flowchart Overview:

The system consists of multiple layers that work together to process user inputs securely, generate appropriate responses, and ensure data privacy. Here is the detailed explanation of each component, following the flow:

AES - Adversarial Attack Check - NLP Context Classification - Content Filtering - LLM Guard - NLP Processing - Query Processing (LawGPT or General LLM) - Hosting

---

1. AES (Advanced Encryption Standard)

- **Purpose:** AES is used to encrypt the user input before sending it through the pipeline, ensuring data security during transit.
- **Implementation:** The data is encrypted at the beginning of the flow and decrypted once it reaches the processing layers. Use AES-256 for strong encryption.

Code Explanation:
python
Copy code

```
from Crypto.Cipher import AES

# Key and IV setup

key = b'YourAESKeyHere1234'  # 16 bytes key

iv = b'YourAESIVHere123456'  # 16 bytes IV



# Encryption

cipher = AES.new(key, AES.MODE_CBC, iv)

ciphertext = cipher.encrypt(pad(user_input.encode(),
AES.block_size))



# Decryption when processing

decipher = AES.new(key, AES.MODE_CBC, iv)

decrypted_data = unpad(decipher.decrypt(ciphertext), AES.block_size)
```

Secure AI Chatbot Documentation

-

2. Adversarial Attack Check

- **Purpose:** This component uses ReBuff.AI or similar frameworks to identify and mitigate adversarial attacks in user input.
- **Implementation:** Runs checks on the encrypted input to identify potentially malicious or adversarial patterns.

Code Explanation:
python
Copy code

```
import re

from rebuff import AdversarialChecker



# Initialize Adversarial Checker

checker = AdversarialChecker()



# Check for adversarial attacks

if checker.is_adversarial(decrypted_data):

    raise ValueError("Adversarial input detected")
```

-

3. NLP Context Classification

- **Purpose:** Classifies the context of the input as either legal (1) or general (0). This classification guides the LLM Layer on which model to use for query processing.
- **Implementation:** Uses NLP models to extract keywords and determine the context of the conversation.

Code Explanation:
python
Copy code

```
from nlp_module import ContextClassifier



# Initialize context classifier
```

```
classifier = ContextClassifier()



# Classify the context

context_type = classifier.classify_context(decrypted_data)

# context_type = 1 if legal, 0 if general
```

-

4. Content Filtering

- **Purpose:** Filters out inappropriate or irrelevant content based on predefined rules and guidelines.
- **Implementation:** Uses filtering frameworks like LLM Guard to moderate and secure input.

Code Explanation:
python
Copy code

```
from llm_guard import ContentFilter



# Initialize content filter

content_filter = ContentFilter()



# Filter the input

filtered_content = content_filter.filter(decrypted_data)

if not filtered_content:

    raise ValueError("Content rejected by filter")
```

-

5. LLM Guard

- **Purpose:** LLM Guard serves as an additional layer of protection, ensuring that inputs to the LLM do not violate security or policy guidelines.
- **Implementation:** Validates the input against advanced criteria and checks for any unauthorized content.

Secure AI Chatbot Documentation

Code Explanation:
python
Copy code

```
from llm_guard import LLMGuard



# Initialize LLM Guard

llm_guard = LLMGuard()



# Guard validation

if not llm_guard.validate(filtered_content):

    raise ValueError("Input failed LLM Guard validation")
```

-

6. NLP Processing

- **Purpose:** Further processes the validated content, including keyword extraction, sentiment analysis, and embedding generation.
- **Implementation:** This module extracts relevant data and prepares it for LLM processing.

Code Explanation:
python
Copy code

```
from nlp_module import NLPProcessor



# Initialize NLP Processor

nlp_processor = NLPProcessor()



# Extract keywords and other NLP elements

keywords, sentiment = nlp_processor.process(filtered_content)
```

-

7. Query Processing

Secure AI Chatbot Documentation

- **Purpose:** Depending on the context classification (legal or general), the system directs the query to the appropriate model for response generation.
    - **If Legal Context (1):** Uses LawGPT for specialized legal or cybersecurity queries.
    - **If General Context (0):** Uses a general-purpose LLM like LLaMA 2.
- **Implementation:** This decision-making is crucial for accurate and context-aware response generation.

Code Explanation:
python
Copy code

```
if context_type == 1:

    # Process with LawGPT for legal context

    from law_gpt import LawGPT

    law_model = LawGPT()

    response = law_model.generate_response(keywords, sentiment)

else:

    # Process with General LLM for non-legal context

    from llama_model import LLaMA2

    general_model = LLaMA2()

    response = general_model.generate_response(keywords, sentiment)
```

-

8. Hosting on Azure

- **Purpose:** Host the entire application on Azure for scalable, secure, and accessible deployment.
- **Implementation:**
    - **Azure Setup:** Configure an Azure VM or App Service with enough resources (CPU, GPU, RAM) to handle concurrent requests.
    - **API Integration:** Use Azure API Management for secure and rate-limited API access.

Code Explanation:
bash
Copy code

```
# Azure CLI commands to set up the environment
```

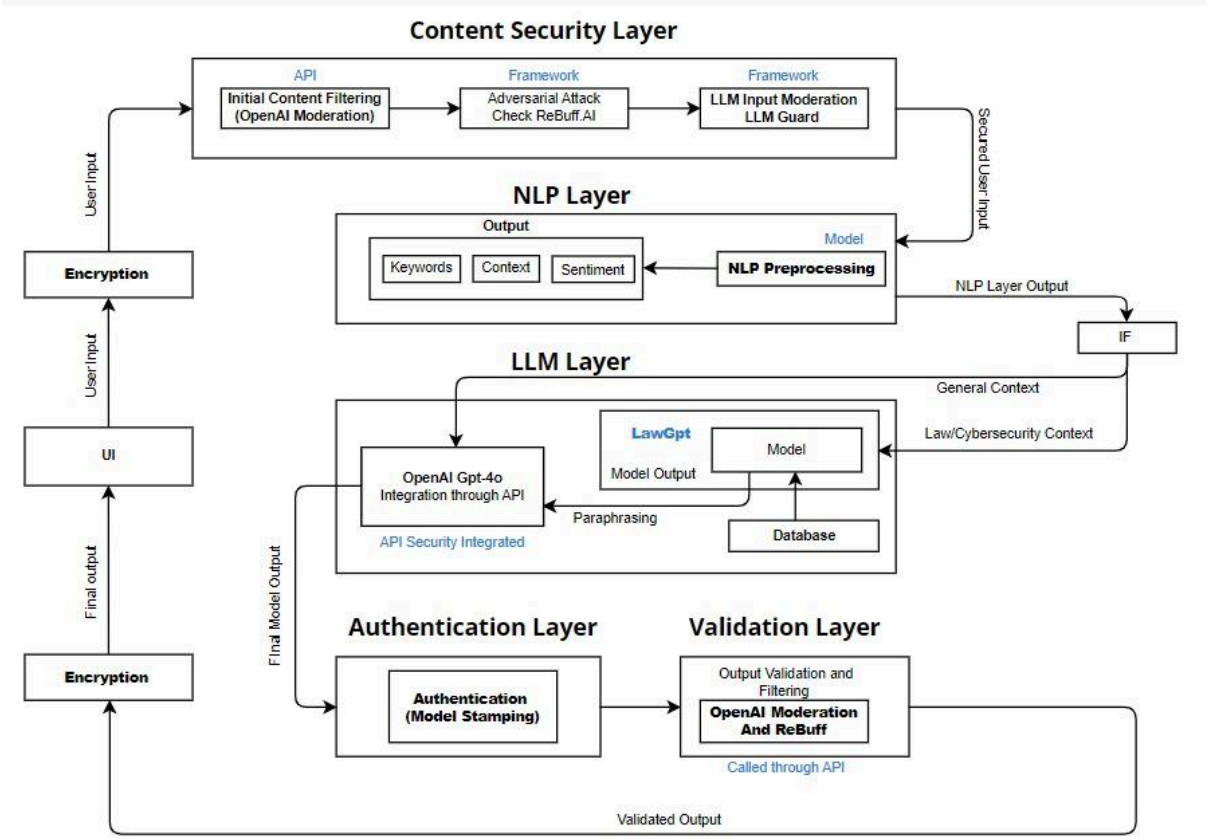
Secure AI Chatbot Documentation

```
az group create --name SecureAIResourceGroup --location eastus

az vm create --resource-group SecureAIResourceGroup --name
SecureAIChatbotVM --image UbuntuLTS --admin-username azureuser
--generate-ssh-keys

az vm open-port --resource-group SecureAIResourceGroup --name
SecureAIChatbotVM --port 80 --priority 1001



# Deploy the application using Docker or direct code upload

az webapp up --name secure-ai-chatbot --resource-group
SecureAIResourceGroup --plan AppServicePlan
```

-

Screenshots and Diagrams

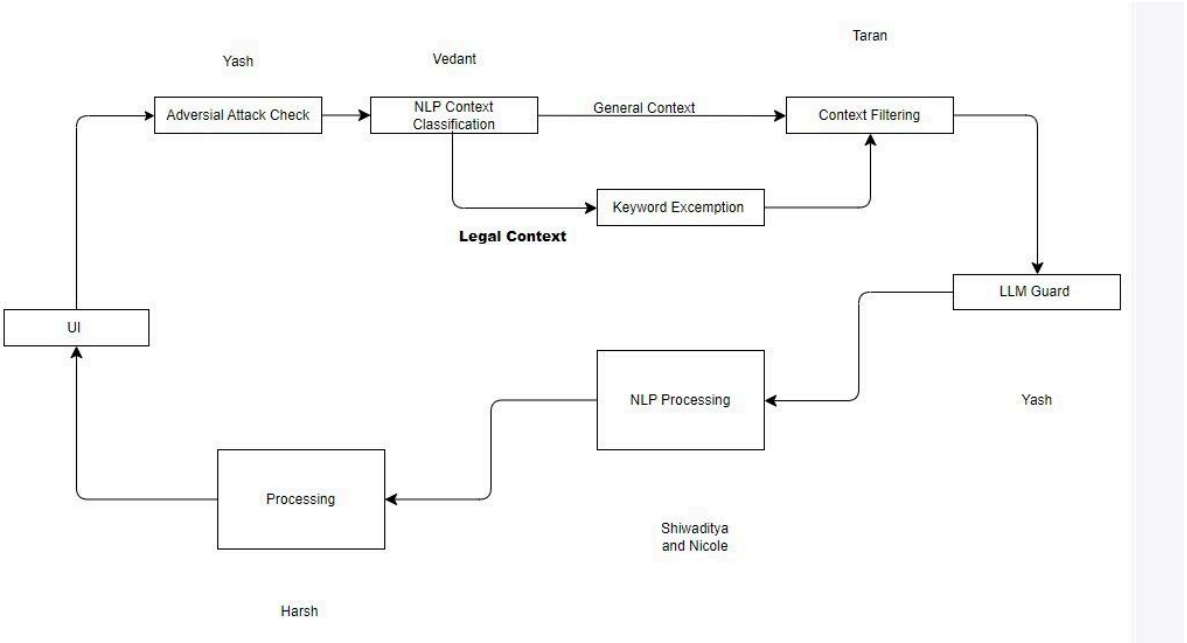
- **Flowchart 1:** Complete Project Flowchart - Shows how data moves through each layer from user input to



final output.

Secure AI Chatbot Documentation

- **Flowchart 2:** Updated Flowchart with Legal Context Focus - Highlights adjustments in the flow for better handling of legal contexts.



9. Standard Operating Procedures (SOP)

- **Pre-Deployment Checklist:**
  - Ensure Azure VMs are configured and secure.
  - Validate all API keys and access controls are correctly set.
  - Test all LLM models locally with dummy data to verify responses.
- **Deployment Steps:**
  - Set up the Azure environment following the configuration steps outlined above.
  - Deploy the code using Docker, ensuring all dependencies are met (`requirements.txt`).
  - Monitor system performance and adjust VM resources as necessary for scaling.
- **Post-Deployment:**
  - Regularly update LLM models and NLP pipelines as needed.
  - Conduct security audits periodically to ensure compliance with data privacy laws.
  - Set up logging and monitoring using Azure Monitor for real-time insights and issue tracking.