a)Functional Description: Compliance Monitoring and PDF Report Generation-

Large language models (LLMs) are used in the provided code to construct a system for log analysis-based compliance monitoring and enforcement. It uses ReportLab to create PDF reports, the Streamlit framework for user interaction, and Spacy for natural language processing. Users of the system can upload log files, compliance requirements, and rule definitions. As a result of processing, it examines the logs, looks for compliance infringements, and creates a thorough PDF report with both compliant and non-compliant logs.

1.*User Interface:-* A Streamlit web interface will be used by users to communicate with the system.
They are prompted to upload the following files:
Rule definitions file: Specifies rules for compliance checking.
Compliance standards file: Defines compliance standards for entities.
Log text file: Contains logs to be analyzed for compliance.

2.*ComplianceMonitor Class:-* The main compliance monitoring logic is contained in this class.It loads rule definitions, compliance requirements, and an NLP model for entity extraction upon initialization.The extract_entities method extracts named entities from text using *Spacy's NLP model*.A stated rule's compliance with an entity is checked using the *check_compliance* method.

3.*Log Analysis:-* The ***_analyze_logs*** method processes log files and extracts entities from each log entry.
For each entity, the corresponding rule is retrieved from the rule definitions.
If the entity complies with the rule, it is marked as compliant; otherwise, it's marked as non-compliant.
A comprehensive compliance report is generated, including details of compliant and non-compliant entities for each log entry.

4.PDF Report Generation:- The ***process_data_and_generate_pdf*** function orchestrates the entire process.
It instantiates a ComplianceMonitor object with the provided rule definitions and compliance standards.
The ***analyze_logs*** method of the ***ComplianceMonitor class*** is called to process the uploaded log file and generate compliance results.
The compliant and non-compliant entities for each log entry are organized into tables.
A landscape-oriented PDF document is created using *ReportLab's SimpleDocTemplate*.
If compliant entities are present, a table for compliant logs is generated and styled with appropriate formatting.
If non-compliant entities are present, a table for non-compliant logs is generated and styled.
The PDF document is built, and its binary content is returned.

5.Streamlit UI Interaction:The main function initializes the Streamlit web application.Users upload rule definitions, compliance standards, and log files using file upload widgets.When the *"Process and Generate PDF"* button is clicked, the following steps are executed:Uploaded files are loaded and processed.The ***process_data_and_generate_pdf*** function is called to generate the PDF report.
A download button is displayed to allow users to download the generated PDF report.

b)Code Overview:-

ComplianceMonitor Class:
The *ComplianceMonitor class* is at the heart of the compliance monitoring system. It plays a crucial role in analyzing logs, checking compliance, and extracting relevant information.

Initialization:
Upon initialization, the class will takes two important inputs:
rule_definitions: A dictionary that defines rules for different entities.
compliance_standards: A dictionary that defines compliance standards for entities.
It also initializes an instance of the *Spacy NLP model (en_core_web_lg)* for natural language processing.

Entity Extraction:
The *extract_entities* method takes a text input and utilizes ***Spacy's NLP*** model to extract named entities from the text. Named entities can include things like dates, people's names, locations, and more.

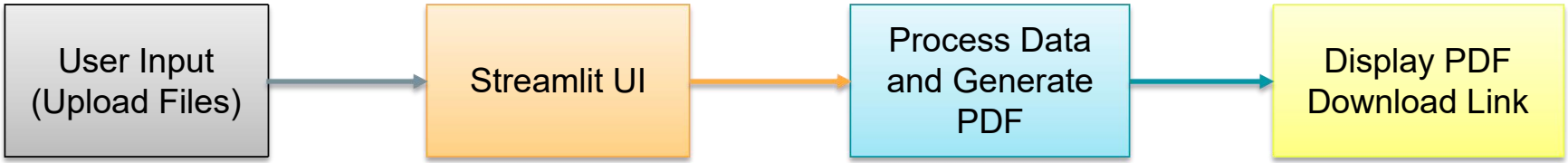
Compliance Checking:

The *check_compliance* method is responsible for determining if a given entity complies with a specific rule defined in the *rule_definitions*. It takes an entity and a rule as inputs and checks whether the entity's compliance matches the rule.

For instance, let's say there's a rule that states: "Sensitive data must not be shared." If the entity extracted from a log is "credit card number," the *check_compliance* method will verify if this entity complies with the rule mentioned.

These functionalities together enable the *ComplianceMonitor* class to effectively analyze log entries, extract meaningful entities, and check their compliance based on the defined rules and standards. This class serves as the backbone for the compliance monitoring process within the system.

c)Flow Chart:-



d)Code Submisssion:-

You can find code here: https://github.com/shubhman20/log-analysis

Glossary:-

- ***NLP – Natural Language Processing***
- ***LLM – Large Language Model***
- ***NER – Named Entity Recognition***
- ***PDF – Portable Document Format***

Use-cases:

1.IT Security Compliance:-

•***Use Case:*** Ensuring that an organization's IT systems adhere to industry-specific security standards and regulations. To find violations of compliance regulations, the solution can examine system logs, configurations, and access controls.

2.Data Privacy Compliance:-

•***Use Case:*** Ensuring compliance with data privacy regulations such as GDPR or CCPA. To find instances of unauthorized data access, potential data breaches, or violations of data handling standards, it can examine logs and user access to sensitive data.

3.Financial Regulations Compliance:-

•***Use Case:*** Ensuring financial institutions comply with regulations like SOX (Sarbanes-Oxley Act) or Basel III. To find anomalies, unlawful financial transactions, and suspected fraud attempts, the solution may examine system activity logs and financial transaction records.

4.Regulatory Audits and Reporting:-

•***Use Case:*** Simplifying the process of preparing for regulatory audits and generating compliance reports.
The system has the ability to automatically analyze logs, configurations, and access patterns to produce compliance reports that point out both legal and illegal activity.

Proposed approach:

Problem Overview: Ensuring Compliance in Complex Systems through Log Analysis:-In today's technology-driven landscape, big organizations are facing the daunting task of handling vast volumes of data and intricate systems. A paramount concern for such companies is the assurance of compliance with security policies, standards, and baselines. As organizations navigates the challenges posed by ever-increasing data and complex systems, the need to uphold stringent compliance measures becomes a critical imperative.

Challenges: -The rapid growth of data and complexity in the tech giants introduces a significant challenge in ensuring that security policies and standards are consistently adhered to. With an intricate web of interconnected components and processes, the manual review of logs, system configurations, access controls, and user privileges becomes arduous and error-prone.

Objective:- The primary objective of this endeavor is to devise a comprehensive solution that capitalizes on the capabilities of large language models (LLMs) to address the compliance conundrum. The solution should center around automated compliance monitoring and enforcement through meticulous log analysis, derived from pertinent sources.

Overall Solution:- The proposed solution entails the development of a sophisticated system capable of undertaking compliance monitoring and enforcement tasks by leveraging the power of ***LLMs***. The system employs a combination of ***NLP techniques***, ***rule-based compliance checks***, and ***PDF report generation*** to provide actionable insights for security and compliance improvements.

We can divide the problem into 6 sub-problems as discussed below:-

Sub-Problem 1: Processing Log Files -

Solution Overview: The first step is to read and process the log files provided as input. This involves reading each line from the log file and passing it through the ComplianceMonitor class(custom class) for analysis.

Detailed Solution Steps:

File Upload: In the proposed solution we utilizes ***Streamlit's file uploader*** to allow users to upload rule definitions, compliance standard and log file. Streamlit is an open-source Python library designed to create interactive web applications.

Temporary Storage: Upon upload, it saves the uploaded log file temporarily in a named temporary file using the ***tempfile*** module. The ***tempfile*** module in Python provides functions for working with temporary files and directories. Temporary files are files that are automatically deleted when they are closed or when the program terminates.

Reading Log Lines: Open the temporary log file and read its contents line by line where each line represents a log entry.

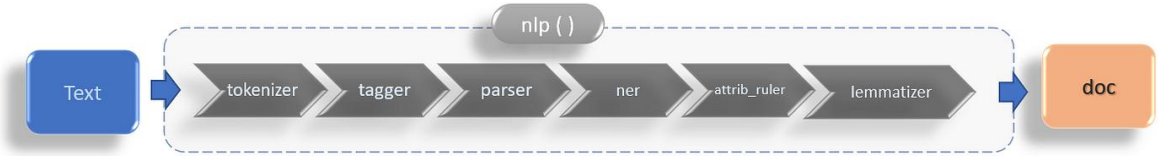


Fig. Working of NLP

Sub-Problem 2: Entity Extraction from Logs-

Solution Overview:To analyze compliance, relevant entities (information) must be extracted from each log entry. spaCy's Natural Language Processing (NLP) capabilities can be utilized for entity extraction.

Detailed Solution Steps:

Load Spacy Model: In this solution we utilizes *spaCy's* ***en_core_web_lg*** *model*, which is pre-trained for Large Language Model(English). spaCy is designed to provide fast and efficient tools for various tasks, including tokenization, part-of-speech tagging, named entity recognition, syntactic parsing, and more.

Processing Log Entries: For each log entry, we applied the ***spaCy's NLP model*** to tokenize and parse the text.
Entity List: Compile a list of extracted entities from each log entry.

Sub-Problem 3: Rule-Based Compliance Checks -

Solution Overview: Now the extracted entities need to be matched against predefined rules and compliance standards to determine if the log entry is compliant or not.

Detailed Solution Steps:

Rule Definitions and Compliance Standards: We load the rule definitions and compliance standards from uploaded files.

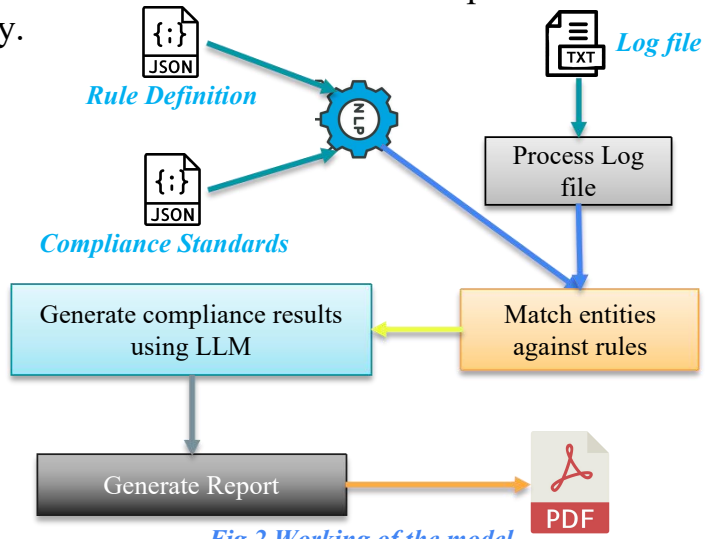


Fig.2 Working of the model

Rule Matching: For each extracted entity, we will check if there is a corresponding rule defined in the rule definitions or not. If a rule exists, mark the entity as relevant for compliance checking.

Compliance Checking: For each relevant entity, we have to check if it complies with the defined compliance standards. If the entity is compliant, mark it as such.

Sub-Problem 4: Generating Compliance Results-

Solution Overview:- Now we have to generate a summary of compliance results for each log entry, including compliant and non-compliant entities.

Detailed Solution Steps:

Results Structure: For storing compliance results we create a data structure. This could be a dictionary where log entries are keys, and corresponding compliance details are values.

Storing Results: Store compliance information for each log entry, including details about relevant entities, rules, and compliance status (compliant or non-compliant).

Sub-Problem 5: Report Generation -

Solution Overview:- For presenting the compliance analysis results in an understandable and informative manner we generate the PDF reports which consists of compliant and non-compliant logs.

Detailed Solution Steps:-

PDF Generation: Here we use ***ReportLab*** library to generate PDF reports summarizing compliance analysis results. The ReportLab library is a powerful Python library used for generating PDF documents. It provides a comprehensive set of tools for creating complex PDF documents from scratch.

Table Creation: Now we create tables in the PDF document to display compliant and non-compliant entities along with their associated information.

Styling: Apply styling to the tables to distinguish between compliant and non-compliant entries. For example we can change the background color for compliant table to green and non-compliant table to red and many more.

Sub-Problem 6: User Interface-

Solution Overview:- Designed a user interface to facilitate the interaction with the system, allowing users to upload rule definitions, compliance standards, and log files.

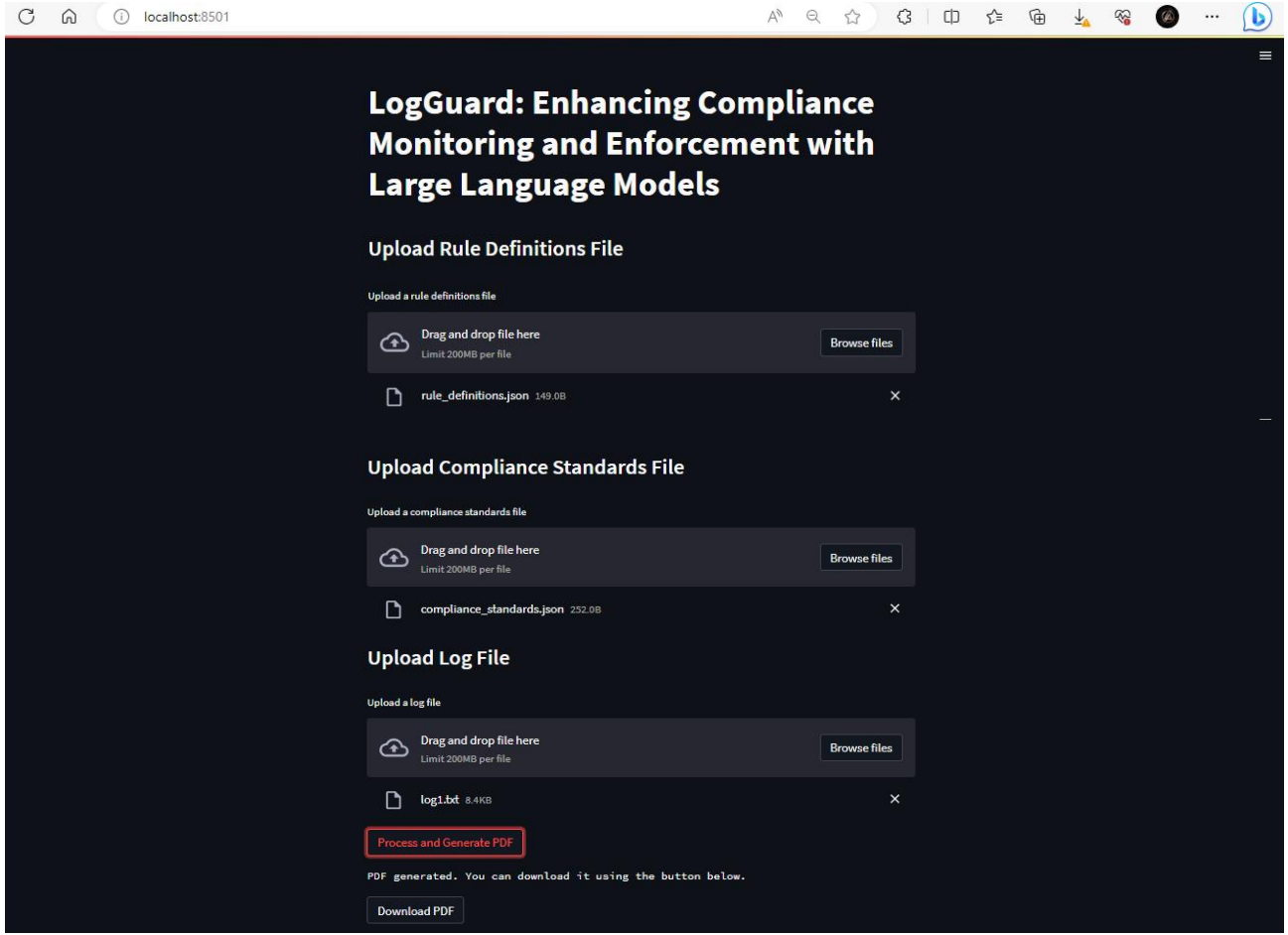
Detailed Solution Steps:-

Streamlit UI: Developed a user friendly interface using the Streamlit framework which includes file upload widgets for rule definitions, compliance standards, and log files.

Processing Trigger: Implement a button that triggers the processing of uploaded files and initiates compliance analysis if all the three documents are successfully uploaded(rule definitions, compliance standard and log files).

Download PDF: Created a download button that allow users to download the generated PDF report.

App Demo:-


localhost:8501
LogGuard: Enhancing Compliance Monitoring and Enforcement with Large Language Models
Upload Rule Definitions File
Upload a rule definitions file
Drag and drop file here
Limit 200MB per file
Browse files
rule_definitions.json 149.0B
Upload Compliance Standards File
Upload a compliance standards file
Drag and drop file here
Limit 200MB per file
Browse files
compliance_standards.json 252.0B
Upload Log File
Upload a log file
Drag and drop file here
Limit 200MB per file
Browse files
log1.txt 8.4KB
Process and Generate PDF
PDF generated. You can download it using the button below.
Download PDF

Generated Compliance PDF:-

Compliant Logs
Log 1: UserAuthentication: Compliant (All users must have strong passwords)
Log 3: UserAuthentication: Compliant (All users must have strong passwords)
Log 4: DataAccess: Compliant (Access to sensitive data must be logged and authorized)
Log 5: UserAuthentication: Compliant (All users must have strong passwords)
Log 7: UserAuthentication: Compliant (All users must have strong passwords)
Log 8: DataAccess: Compliant (Access to sensitive data must be logged and authorized)
Log 10: DataAccess: Compliant (Access to sensitive data must be logged and authorized)
Log 14: DataAccess: Compliant (Access to sensitive data must be logged and authorized)
Log 18: DataAccess: Compliant (Access to sensitive data must be logged and authorized)
Log 19: UserAuthentication: Compliant (All users must have strong passwords)
Log 22: DataAccess: Compliant (Access to sensitive data must be logged and authorized)

Non-compliant Logs
Log 2: Non-compliant (All users must have strong passwords)
Log 6: Non-compliant (All users must have strong passwords)
Log 9: Non-compliant (Access to sensitive data must be logged and authorized)
Log 11: Non-compliant (Access to sensitive data must be logged and authorized)
Log 12: Non-compliant (Access to sensitive data must be logged and authorized)
Log 13: Non-compliant (Access to sensitive data must be logged and authorized)
Log 15: Non-compliant (Access to sensitive data must be logged and authorized)
Log 16: Non-compliant (Access to sensitive data must be logged and authorized)
Log 17: Non-compliant (Access to sensitive data must be logged and authorized)
Log 20: Non-compliant (All users must have strong passwords)
Log 21: Non-compliant (All users must have strong passwords)
Log 23: Non-compliant (Access to sensitive data must be logged and authorized)

Limitations:

Dependency on External Services:- The application depends on external services like Streamlit and spaCy. If these services have downtime or compatibility issues, the application's functionality could be affected. To overcome dependency on Streamlit we can make our custom website using HTML,CSS & JavaScript which can be more reliable than Streamlit.

Limited Reporting and Visualization:- The generated PDF report lacks some comprehensive visualization of compliance results.

No real-time analysis or ongoing monitoring:- Real-time log analysis or continuous monitoring are not features of the current implementation. Real-time skills are crucial for proactive compliance enforcement.

Future Scope:

Potential future scope for Compliance Monitoring and Enforcement Log Analysis using Large Language Models (LLMs):-

Enhanced NLP Models:- Future iterations of LLMs might provide even greater contextual comprehension and analysis of logs as natural language processing (NLP) models advance, allowing for more precise compliance checks and insights.

Continuous Learning and Improvement:- By continuously learning from the data they examine, LLMs can get better. Self-improving algorithms may be used by future systems to improve their accuracy over time.

AI-Driven Threat Hunting:- By examining logs for complex attacks and advanced persistent threats that may be difficult to identify using conventional techniques, businesses could undertake proactive threat hunting using AI-powered LLMs.

Industry-Specific Solutions:- Every industry has different compliance standards. In order to meet certain compliance requirements, future systems might provide industry-specific configurations and rule sets.