EthiCheck.AI : Keeping Conversational AI Chats Ethical

Team : WatchDogs

Debasmit Roy Anannyo Dey Kushal Das

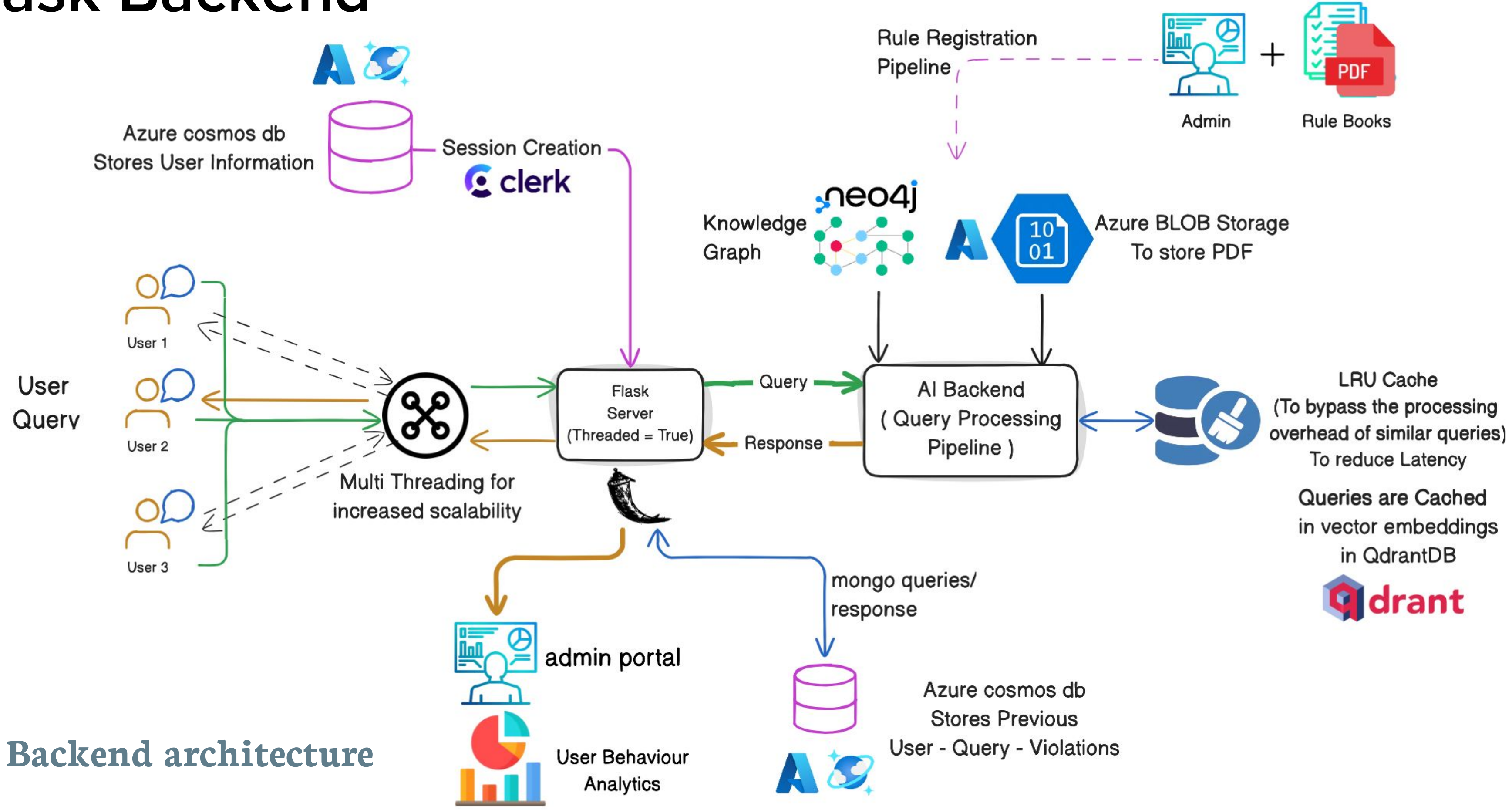
Approach

- Usage of **MS Azure CosmosDB** for storing user info and violations
- Implementation of **multithreading** in Flask for increased scalability
- Integration of **LRU cache** to bypass the processing overhead for similar queries, thereby reducing latency
- Development of LLM-powered Rule keyword extractor for creating **Rule Knowledge graph** (KG) from PDFs uploaded by admin for **complex RAG.** It can be seen as an **alternative of expensive fine-tuning.**
- **Azure Blob storage** to store Admin's Rule PDFs and **neo4j** to store the **KG**.
- **Semantic router** to retrieve relevant **subgraphs** based on user queries/GPT responses, followed by a **query pipeline** to identify violations (if any).
- User authentication using **Clerk** integrated with **Azure CosmosDB.**
- Deployment of entire solution on **Vercel (frontend) and Pythonanywhere (backend) and MS Azure (database and storage)**

Features

- Periodically updated Rule **Knowledge graph**, whenever admin uploads new PDFs
- Display of **rule violations and reasons** levels on admin dashboard
- Assignment of **Risk score based on number of violations and its risk le**vels
- User behaviour pattern analysis : **Category wise Rule Violation Count per day/week**
- **MS Azure** integration performed utilizing **Cosmos DB** containers and **Blob storage**

Flask Backend



Azure cosmos db
Stores User Information

Session Creation
clerk

Rule Registration
Pipeline

Admin

Rule Books

Knowledge
Graph

neo4j

Azure BLOB Storage
To store PDF

User
Query

User 1

User 2

User 3

Multi Threading for
increased scalability

Flask
Server
(Threaded = True)

Query

Response

AI Backend
(Query Processing
Pipeline)

LRU Cache
(To bypass the processing
overhead of similar queries)
To reduce Latency

Queries are Cached
in vector embeddings
in QdrantDB

qdrant

mongo queries/
response

admin portal

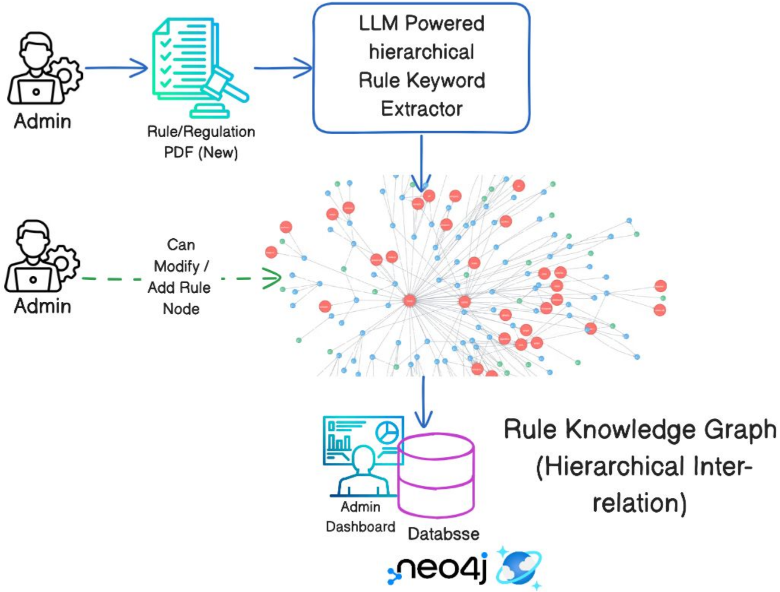
User Behaviour
Analytics

Azure cosmos db
Stores Previous
User - Query - Violations

Backend architecture

High Level Design AI Backend

Rule Registration Pipeline



Admin

Rule/Regulation PDF (New)

LLM Powered hierarchical Rule Keyword Extractor

Admin

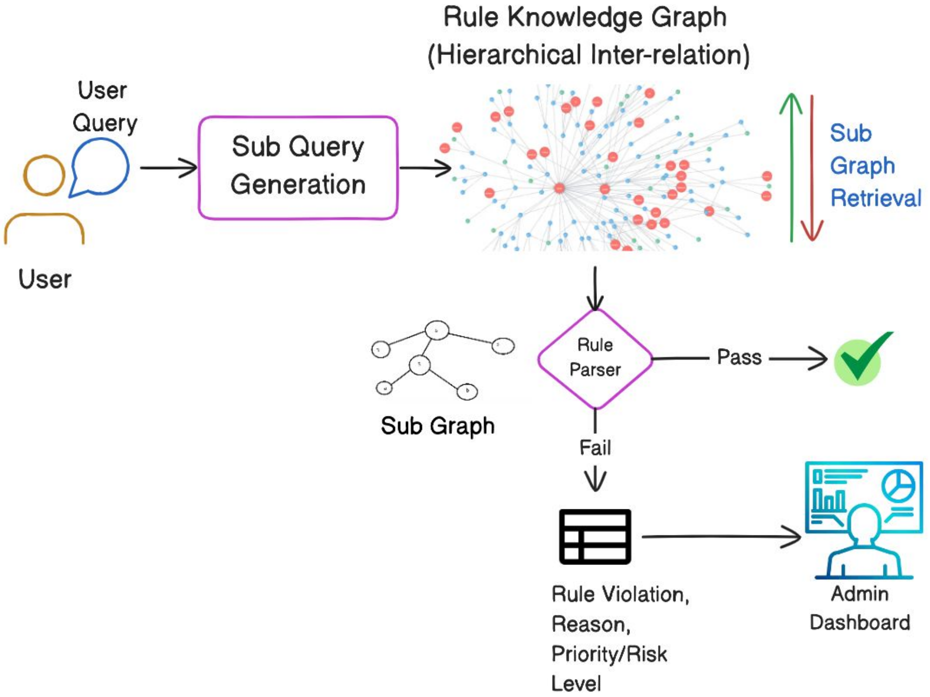
Can Modify / Add Rule Node

Admin Dashboard

Databsse

Rule Knowledge Graph (Hierarchical Inter-relation)

Query Processing Pipeline



User Query

User

Sub Query Generation

Rule Knowledge Graph (Hierarchical Inter-relation)

Sub Graph Retrieval

Sub Graph

Rule Parser

Pass

Fail

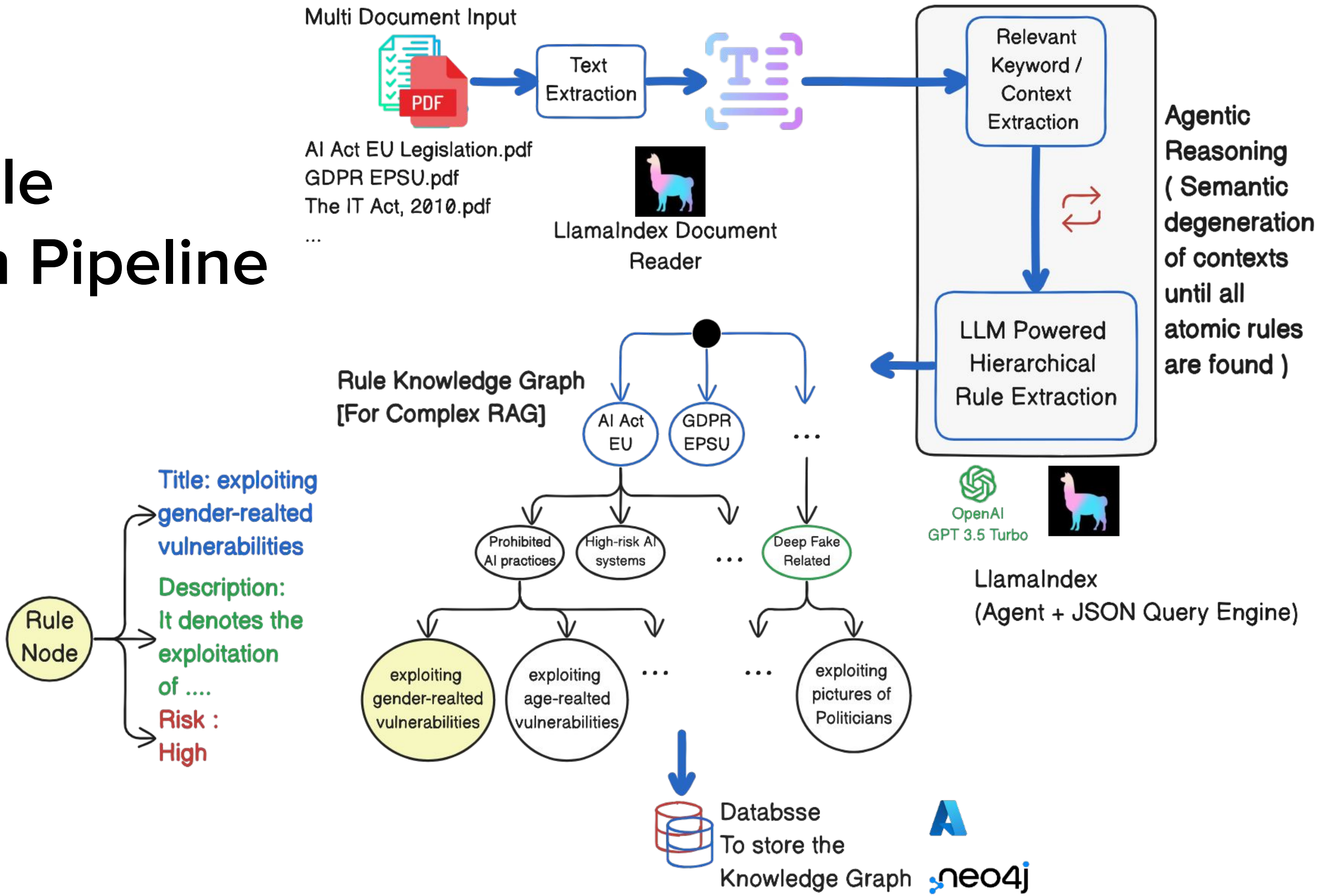
Rule Violation, Reason, Priority/Risk Level

Admin Dashboard

Rule Registration Pipeline

1. The admin sends PDFs containing Rule to the system, which is stored on **Azure CosmosDB** then passed to the **Llama Index document reader** to extract the text.
2. The entire PDF text is decomposed into multiple contexts, which are then further broken down into more atomic components. Utilizing the Llama Index powered agentic reasoning pipeline, we identify and decompose relevant contexts until no additional rules can be extracted.
3. Similar rule phrases are grouped to form a **knowledge graph**, delineating the **hierarchical relations** among several rules. This will be further used to parse **complex RAG queries**. Risk levels are assigned for each terminal nodes as mentioned in the pdf document. **Knowledge graph** is stored in **neo4j graph-db.**
4. **External knowledge source** also integrated to check rules related to recent topics.
5. The admin can also **modify** the AI generated tree structure from Admin dashboard if necessary, by **adding, deleting or editing items**

Detailed Rule Registration Pipeline



Multi Document Input

PDF

AI Act EU Legislation.pdf
GDPR EPSU.pdf
The IT Act, 2010.pdf
...

Text Extraction

LlamaIndex Document Reader

Relevant Keyword / Context Extraction

LLM Powered Hierarchical Rule Extraction

Agentic Reasoning (Semantic degeneration of contexts until all atomic rules are found)

OpenAI
GPT 3.5 Turbo

LlamaIndex
(Agent + JSON Query Engine)

Rule Knowledge Graph
[For Complex RAG]

AI Act EU
GDPR EPSU
. . .

Prohibited AI practices
High-risk AI systems
. . .
Deep Fake Related

exploiting gender-realted vulnerabilities
exploiting age-realted vulnerabilities
. . .
. . .
exploiting pictures of Politicians

Rule Node

Title: exploiting gender-realted vulnerabilities

Description: It denotes the exploitation of ....

Risk : High

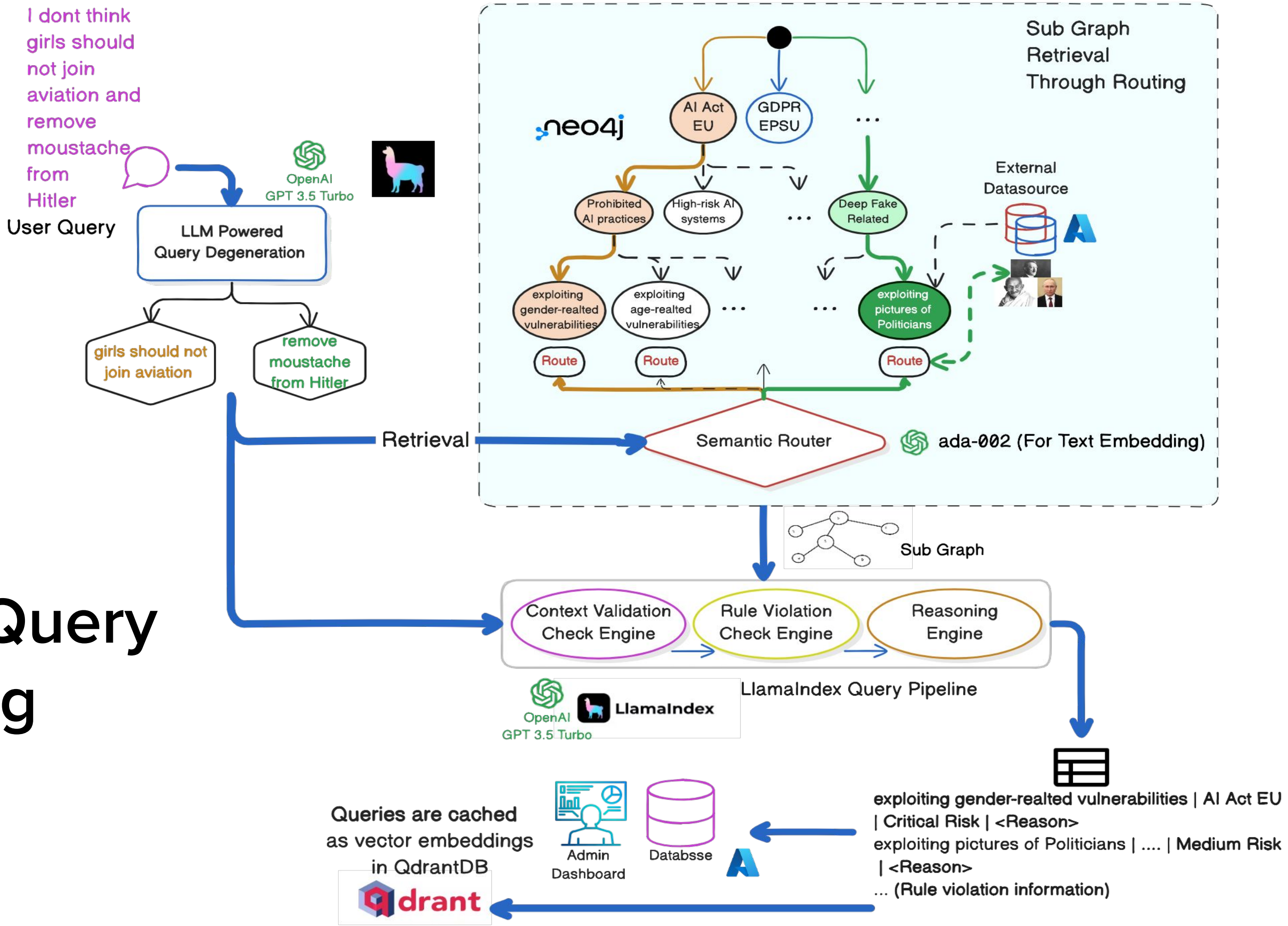
Databsse To store the Knowledge Graph

neo4j

Query Processing Pipeline

1. Firstly, user query is degenerated into subqueries using **Llama Index JSON Query Engine**.
2. These subqueries are then routed to the existing knowledge graph through a **semantic router**, using **text-embedding-ada-002** for semantic similarity calculation. Subsequently, all relevant paths in the form of **subgraphs** are retrieved.
3. The **subqueries** along with the relevant **subgraph** are ingested into a **LlamIndex Query Pipeline** where **three sequential LLM engines** operates:
    i. **Context Validation Check Engine** validates whether **subqueries** and **subgraph** belong to a same contexts
    ii. **Rule Violation Check Engine** then check whether the **subqueries** violate any rule from subgraph
    iii. **Reasoning Engine** simultaneously states the **reason** for violation if any for each subquery.
4. All possible violations are listed along with calculated risk count. These are stored on **MS Azure CosmosDB containers.**
5. Queries are **cached** as vector embeddings in **Qdrant Vector DB** along with the response to reduce the processing overhead of very similar queries.

Detailed Query Processing Pipeline



I dont think girls should not join aviation and remove moustache from Hitler

User Query

OpenAI GPT 3.5 Turbo

LLM Powered Query Degeneration

girls should not join aviation

remove moustache from Hitler

Retrieval

Sub Graph Retrieval Through Routing

Semantic Router

ada-002 (For Text Embedding)

Sub Graph

Context Validation Check Engine

Rule Violation Check Engine

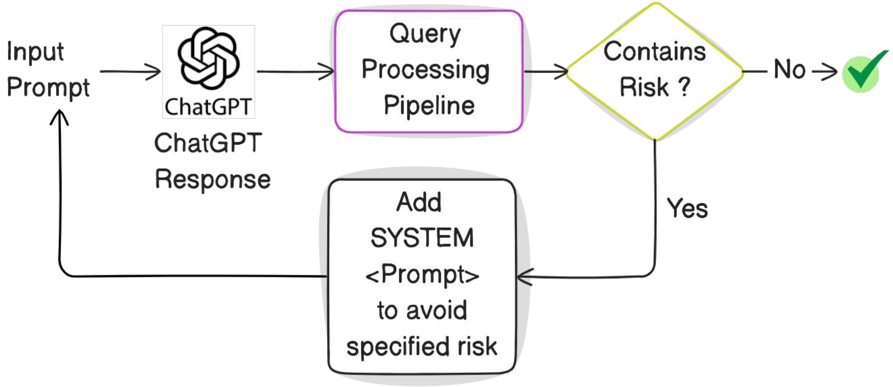
Reasoning Engine

LlamaIndex Query Pipeline

exploiting gender-realted vulnerabilities | AI Act EU | Critical Risk | <Reason>
exploiting pictures of Politicians | .... | Medium Risk | <Reason>
... (Rule violation information)

Queries are cached as vector embeddings in QdrantDB

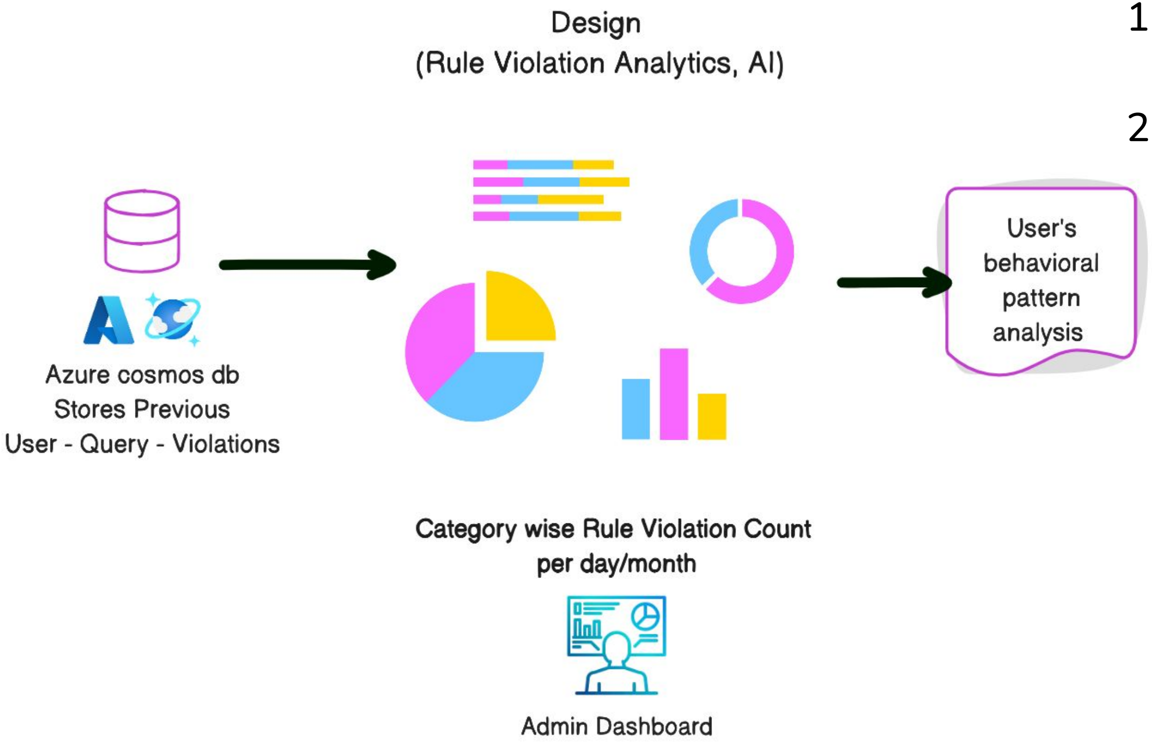
Mitigating risky ChatGPT response


Input Prompt
ChatGPT
ChatGPT Response
Query Processing Pipeline
Contains Risk ?
No
Yes
Add SYSTEM <Prompt> to avoid specified risk

Risk Scoring Algorithm

We assign risk scores to every violation in the form of user query and GPT response. This is based on risk level categorization of the **EU AI act** - **Critical**, **High**, **Medium** & **Minimal**, where each level has a **defined score** and based on the number and nature of violations in the query/response we calculate the **Risk score**. We show risk levels and their categories in the admin analytics panel. **Risk thresholding** is performed on the scores calculated.

Analysis and Visualization



Design
(Rule Violation Analytics, AI)

Azure cosmos db
Stores Previous
User - Query - Violations

User's behavioral pattern analysis

Category wise Rule Violation Count
per day/month

Admin Dashboard

1. All violations for each user query are stored in **Azure Cosmos DB**
2. These data are displayed on **Admin Dashboard**, in form of charts-
   a. Line and Bar charts: **No. of violations** within **custom time-frame for each risk level**
   b. Stacked Chart: **Compare no. of Rule violations** per **high-level category** between 2 dates
   c. Pie chart: **Compares %** of **violations** and **safe queries**
   d. User-wise violation analytics: **Top5 violations per user**

Challenges and Solution

1. LLM Hallucination: We mitigate this by using **low temperature LLMs** and in-context RAG based search in our Rule parse tree (Knowledge Graph, KG). As we fragment the query into smaller phrases and then pass it through the router, we can avoid dependency on high-temperature language model (LLM).

2. Improving explainability: Parsing user queries and GPT responses using Rule parse tree KG and semantic router makes it **explainable both to the admin and end user**, as to **exactly which violations have occurred and their severity levels.**

3. Improving scalability: We support multiple users and GPT sessions simultaneously using **multithreading**. Admin has the privilege to **add more rule nodes** to the KG making it more scalable.

4. Reducing latency: We have taken several measure like using **LRU cache, mitigating the usage of high temperature LLM** etc. to reduce the response time and make the entire process real-time, even while handling multiple clients.

Tech Stack

- Python
- ReactJS
- Chart.js
- Flask
- Clerk
- Redux

- OpenAI
- LangChain
- Qdrant Vector Store
- neo4j
- LlamaIndex

- Azure
- Cosmos DB
- Blob Storage

- Vercel
- pythonanywhere by ANACONDA

Models

1. LLM Model: gpt-3.5-turbo-0613 (Temperature 0.0)
2. Dense Embedding Model : openai/text-embedding-ada-002

Future Scopes:

1. Integration of GPT4 models which can analyze multimodal queries for potential risks.
2. To safeguard responses based on publicly available code of conduct beyond the provided rulebooks.

Thank You

Link : https://ethicheck-ai.vercel.app/
Demo video: https://www.youtube.com/watch?v=EKxfSHc3kj4
Code: https://github.com/orgs/knacktohack314/repositories