LLMScan — LLM Misbehaviour Detection System

A research-oriented system that automatically detects **unsafe or unreliable outputs** generated by Large Language Models (LLMs).

LLMScan analyzes responses and flags:

- Hallucinations
- Toxic content
- Jailbreak / prompt injection attempts
- Policy violations

Built as a **modular ML + rule-based scanning pipeline** using Python and version controlled with **Git**, hosted on **GitHub**.

Problem Statement

LLMs sometimes generate:

- factually incorrect information
- harmful or toxic content
- unsafe instructions
- adversarial responses

Manual monitoring is slow and unreliable.

Goal:
Automatically detect and classify misbehaviour in LLM outputs using rule-based checks and machine learning.

Features

- Prompt → Response generation
- Rule-based safety scanner
- ML-based misbehaviour classifier
- Risk scoring
- Evaluation metrics (Accuracy, Precision, Recall, F1)
- Dashboard / CLI reporting

System Architecture

Prompt

↓

LLM Generator

↓

Rule Scanner

↓

ML Classifier

↓

Risk Score + Report

Tech Stack

- Python 3.10+
- pandas / numpy
- scikit-learn
- transformers (optional)
- Streamlit/Flask (UI)
- Git + GitHub

Project Structure

LLMScan/

```
│
├── src/            # core modules
│   ├── data_loader.py
│   ├── generator.py
│   ├── scanner_rules.py
│   ├── scanner_ml.py
│   ├── evaluate.py
│
├── data/
│   ├── raw/
│   └── processed/
│
├── notebooks/      # experiments + EDA
├── tests/
├── docs/
│   └── SRS.md
│
├── requirements.txt
└── README.md
```

Installation

1. Clone repo

git clone <repo-url>

cd LLMScan

2. Create environment

python -m venv venv

source venv/bin/activate # windows: venv\Scripts\activate

3. Install dependencies

pip install -r requirements.txt

Usage

Run data preprocessing

python src/data_loader.py

Generate responses

python src/generator.py

Run scanner

python src/scanner_ml.py

Evaluate model

python src/evaluate.py

(Optional)

streamlit run app.py

Dataset Format

prompt, response, label

Labels:

- safe
- toxic
- hallucination
- jailbreak

Evaluation Metrics

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Results (to be updated)

Model	Accuracy	F1
Rule-based	—	—
ML Classifier	—	—

Weekly Development Plan

- Week 1 → Setup + SRS + dataset

- Week 2 → Preprocessing
- Week 3 → Response generator
- Week 4 → Rule scanner
- Week 5 → ML classifier
- Week 6 → Evaluation
- Week 7 → Dashboard
- Week 8 → Testing & polish

Future Improvements

- Real-time monitoring
- Multi-model support
- Transformer-based classifiers
- Web deployment
- Auto retraining

Contributing

1. Create branch
2. Make changes
3. Commit clearly
4. Open PR

Example:

git checkout -b feature/ml-model

git commit -m "Added TF-IDF classifier"

git push