

Software Requirements Specification (SRS)

Project Title: LLMScan – LLM Misbehaviour Detection System

1. Introduction

1.1 Purpose

This document specifies the requirements for **LLMScan**, a system designed to automatically detect misbehaviour in Large Language Model (LLM) outputs such as hallucinations, toxic content, jailbreak attempts, and unsafe responses.

It serves as a reference for:

- Developers
 - Mentors
 - Reviewers
 - Future contributors
-

1.2 Scope

LLMScan will:

- Accept prompts as input
- Generate LLM responses
- Analyze outputs
- Detect unsafe/misleading behaviour
- Provide risk scores and reports

The system focuses on **detection**, not correction.

1.3 Definitions

- LLM: Large Language Model
 - Misbehaviour: Unsafe, toxic, hallucinated, or policy-violating output
 - Scanner: Module that evaluates model responses
-

2. Overall Description

2.1 Product Perspective

LLMScan is a standalone Python-based ML pipeline that can:

- run locally
 - integrate with APIs
 - be extended for research
-

2.2 Product Functions

The system will:

- Load datasets
 - Generate responses from LLM
 - Extract features
 - Apply rule-based checks
 - Train ML classifier
 - Evaluate performance
 - Display results via dashboard/CLI
-

2.3 User Classes

User	Description
Developer	builds and improves system
Researcher	evaluates misbehaviour
Tester	validates results

2.4 Operating Environment

- Python 3.10+
- Windows/Linux/macOS

- 8GB+ RAM recommended
-

2.5 Constraints

- Depends on LLM API availability
 - Requires labeled datasets
 - Limited by local compute
-

3. Functional Requirements

Input Handling

FR1: System shall accept text prompts as input

FR2: System shall batch process multiple prompts

Response Generation

FR3: System shall generate LLM outputs using API/local model

FR4: System shall store prompt–response pairs

Detection

FR5: System shall detect toxicity using rules

FR6: System shall detect hallucinations

FR7: System shall detect jailbreak/prompt injection

FR8: System shall compute risk score

Machine Learning

FR9: System shall extract textual features

FR10: System shall train classifier

FR11: System shall predict misbehaviour class

Evaluation

FR12: System shall compute accuracy, precision, recall, F1

FR13: System shall generate confusion matrix

Reporting

FR14: System shall display results in dashboard/CLI

FR15: System shall export reports (CSV/JSON)

4. Non-Functional Requirements

Performance

- Response time < 3 seconds per prompt
- Dataset processing within reasonable time

Reliability

- System shall not crash on invalid inputs

Maintainability

- Modular code structure
- Separate files for each component

Usability

- Simple commands to run
- Clear README instructions

Scalability

- Support larger datasets
- Allow adding new detection modules

5. System Architecture

High-Level Flow

Prompt Input

↓

LLM Generator

↓

Scanner (Rules + ML)

↓

Classifier

↓

Report / Dashboard

Modules

1. Data Loader

Loads and preprocesses datasets

2. Generator

Calls LLM and stores outputs

3. Rule Scanner

Keyword/regex/toxicity checks

4. ML Scanner

Feature extraction + classification

5. Evaluation

Metrics computation

6. Interface

CLI or dashboard

6. Data Requirements

Input Format

CSV/JSON:

prompt, response, label

Labels

- safe
 - toxic
 - hallucination
 - jailbreak
-

7. External Interfaces

Software

- Python libraries (scikit-learn, transformers)
- LLM API

Hardware

- Standard laptop/PC

8. Tools & Technologies

- Python
- scikit-learn
- pandas
- transformers
- Git
- GitHub
- Streamlit/Flask (optional)

9. Project Timeline (High-Level)

Phase	Work
Week 1	Setup + SRS + dataset
Week 2	Data preprocessing
Week 3	LLM response generator
Week 4	Rule-based detection
Week 5	ML classifier
Week 6	Evaluation
Week 7	Dashboard
Week 8	Testing + polish

10. Future Enhancements

- Real-time monitoring
- Multi-model support
- Advanced transformer classifiers
- Web deployment
- Auto retraining

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins or other markings on the paper.

1. Acceptance Criteria

Project is complete when:{Goal}

- Detection accuracy $\geq 80\%$
- Working demo available
- Clean GitHub repo
- Documentation complete