FortiLM

Project Team

Muhammad Ahmed	22i-7441
Irsam Maan	22i-7435
Saifullah	22i-1312

Session 2022-2026

Supervised by

Ms. Hina bint e haq

Co-Supervised by

Ms. Saadia Saad

Department of Computing

National University of Computer and Emerging Sciences
Islamabad, Pakistan

June, 2026

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This proposal presents FortiLM, a security layer for Large Language Models (LLMs) that enhances safety, privacy, and trustworthiness in AI systems. The purpose of this document is to outline the problem, motivation, proposed solution, and stakeholders, while highlighting the growing need for reliable AI safety frameworks.

FortiLM integrates hybrid jailbreak detection, privacy preservation, explainability, and adaptive learning into a unified middleware. Unlike existing tools that address only specific concerns (e.g., harmful output detection or privacy filters), FortiLM provides a comprehensive defense against jailbreak attacks, prompt injections, unsafe outputs, and data privacy risks.

Furthermore, recent studies have shown that most major LLMs (such as ChatGPT, Gemini, and Claude) are easily manipulated to produce unsafe or misleading responses (1). High-profile incidents like Samsung's accidental leakage of proprietary code through ChatGPT illustrate the urgent need for protective measures. FortiLM aims to fill this gap by acting as a plugin-style middleware that can be integrated into enterprise and academic systems to safeguard LLM usage at scale.

1.1 Problem Statement

The rapid rise of large language models (LLMs) has introduced a range of security and privacy challenges that traditional moderation tools are unable to address effectively. Jailbreak attacks, such as the widely known DAN GPT, exploit weaknesses in safety protocols and have demonstrated success rates between 52% and 74% in single-turn tests across major AI systems (2).

More advanced, multi-turn exploit strategies like *Crescendo* and *Bad Likert Judge* maintain high success rates of 43–46%, even when defensive measures are active (2). These

attacks reveal the limitations of current safeguards and emphasize the need for more resilient defensive architectures.

In addition to jailbreak techniques, significant data privacy risks exist. Studies show that up to 19% of sensitive data can leak during fine-tuning in the absence of differential privacy safeguards (3). Beyond data leakage, LLMs are also prone to generating unsafe outputs, which may be toxic, biased, or hallucinated, thereby eroding user trust.

In high-stakes sectors such as healthcare, finance, and education, these vulnerabilities could result in financial losses, misinformation, or even risks to human lives. Existing solutions like OpenAI's Moderation API, Guardrails AI, and LlamaGuard provide partial protection but remain limited, as they often lack jailbreak resistance, fail to protect personally identifiable information (PII), or do not provide sufficient explainability.

1.2 Motivation

Ensuring AI safety is both a technical and ethical imperative, as well as a financial necessity. IBM reports that 20% of jailbreak attempts succeed in less than a minute (4), demonstrating how quickly these vulnerabilities can be exploited.

The financial implications of such breaches are staggering. The global average cost of a data breach reached $4.88 million in 2024, with U.S. organizations averaging as high as $10.22 million in 2025 (5). Breaches involving "shadow AI," where unauthorized AI tools are used without oversight, add an additional financial burden of approximately $670,000 per incident (6).

Privacy risks are also magnified by the ability of GPT-4 to infer personal traits with up to 95% accuracy (7), raising concerns about discrimination and profiling. (1) further confirmed that current AI chatbots are easily manipulated to produce unsafe outputs, exposing systemic weaknesses in existing defenses.

Figures 1.2 and 1.3 highlight both the rising costs of breaches over time and the increased likelihood of harmful responses when prompt complexity scales. Together, these findings reveal that without robust safeguards, LLM deployment will continue to pose significant risks.

FortiLM is motivated by this urgent need to build trustworthy AI systems that safeguard organizations, users, and society at large.

1.3 Problem Solution

FortiLM directly addresses these challenges by acting as a comprehensive security middleware for AI systems. Its hybrid jailbreak detection mechanism combines traditional regex filters with machine learning models to identify and block adversarial prompts effectively.

To preserve user privacy, the Privacy Preserver module automatically detects and redacts personally identifiable information before an output is generated. The Output Filtering module ensures that toxic, biased, or unsafe responses are blocked, while an integrated Explainability Module provides interpretable reasoning—using methods such as LIME—for why a particular response was filtered.

FortiLM also incorporates an Adaptive Learning Loop, which continuously retrains on newly observed adversarial data, ensuring resilience against evolving attack strategies. For enterprise integration, an Admin Dashboard supports monitoring, reporting, and feedback mechanisms.

Designed as a plugin-style middleware, FortiLM can be seamlessly integrated into existing systems through APIs, making it flexible for enterprises, universities, and developers alike, while remaining extensible and open-source friendly. Its primary objectives are to automate the detection of jailbreaks and prompt injections, safeguard sensitive data through PII redaction, provide safe and transparent outputs through explainability, deliver enterprise-ready middleware with plugin capabilities, and support adaptive retraining to stay ahead of emerging threats.

1.4 Stake Holders

Stakeholders are the individuals, groups, or organizations that have an interest in or are affected by the deployment of FortiLM. They can directly influence how the system is developed, integrated, and regulated. The table below summarizes the key stakeholders and their respective roles and responsibilities.

Stakeholder	**Role / Responsibility**
Enterprises (Finance, Healthcare, Education)	Deploy FortiLM as a plugin to secure sensitive LLM applications.
AI Developers & Researchers	Extend FortiLM modules, contribute to open-source improvements, and test adversarial robustness.
End Users	Benefit from safer, more trustworthy AI interactions without exposure to harmful or private information.
System Administrators	Manage FortiLM integration, monitor dashboards, and enforce security configurations.
Regulators & Policy Makers	Use FortiLM as a benchmark framework for compliance in AI safety and data protection.

Table 1.1: Stakeholders and Their Roles/Responsibilities in FortiLM Deployment

Chapter 2

Project Description

2.1 Scope

FortiLM is a middleware platform that enhances the security, privacy, and reliability of interactions with large language models (LLM) and small language models (SLM). The system will act as an intelligent layer between the user and the connected LLM, ensuring that all user inputs and AI outputs are processed safely before delivery. FortiLM will implement a hybrid jailbreak detection module, combining rule-based methods with AI-driven classifiers to detect and block adversarial or malicious prompts. To further strengthen data security, the system will include a privacy preserver module, which automatically detects sensitive information such as personally identifiable information (PII), credentials, and API keys, and applies masking or redaction before they are sent to the model. The solution will incorporate an output filter that inspects generated responses for harmful, biased, or toxic content, ensuring that end-users only receive safe and reliable information.

FortiLM will also feature an explainability module which will enable transparency in decision making by displaying why certain prompts or outputs were flagged. The system will feature an admin dashboard for centralized control, monitoring, and log access, allowing stakeholders to visualize threats, analyze system performance, and adjust configurations. All processed interactions will be stored in a database-backed logging system to support auditing, future improvements, and our adaptive learning module. The solution will be developed with a modular architecture, making it flexible enough to be integrable. The scope of FortiLM does not include building a new LLM from scratch but rather focuses on delivering security, monitoring, both plugin- and web- based deployment, and privacy-preserving middleware functionalities that enhance the safety of existing AI applications

2.2 Modules

Write down the modules of the proposed project. Each module should highlight features, using bulleted/numbered notation. When developing both a mobile app and a web app, group the modules according to the system types, such as, Client Web App, Client Mobile App, Admin Web App etc.

2.2.1 Module 1 : Hybrid Jailbreak Detection

The module ensures prompts entered are analyzed for adversarial or jailbreak attempts before reaching the LLM. It uses both rule-based detection for known attack patterns and an AI-based classifier for semantic analysis, providing a hybrid defense.

1. Rule-based detection
2. AI-based classifier
3. Hybrid scoring system combining both methods.

2.2.2 Module 2 : Privacy Preserver

This module protects sensitive data by detecting and masking Personally Identifiable Information (PII) and credentials in prompts and outputs. It ensures that private data never leaves the middleware unprotected.

1. PII detection using custom NER models and regex-based rules
2. Automatic masking/replacement of sensitive values with tokens
3. Context preservation of prompt with placeholder substitution and entity-type replacement

2.2.3 Module 3 : Output Filter

This module reviews the LLM-generated responses before presenting them to users, ensuring outputs are safe, unbiased, and useful. It implements a quality-control end to end secure system.

1. Harmful/unsafe content detection
2. Toxicity and offensive language filtering

3. Output side PII masking in generated outputs
4. Smart actions: allow, mask, warn, or block.

2.2.4 Module 4 : Explainability

This module provides interpretability for system decisions by explaining why a prompt or response was blocked, flagged, or modified.

1. Shows which rules or classifiers triggered a decision.
2. Provides confidence scores for model judgments.
3. Generates human-readable explanations.
4. Supports appeals and administrator review.

2.2.5 Module 5 : Adaptive Learning

This module enables continuous improvement of the system by learning from flagged cases, admin feedback, and evolving attack patterns.

1. Collects feedback from users and admins.
2. Updates detection rules dynamically.
3. Classifiers retraining with new examples

2.2.6 Module 6 : Admin Dashboard

This module provides administrators with complete visibility and control over the middleware. It allows monitoring, feedback collection, and system management.

1. Real-time monitoring of flagged prompts and responses.
2. Visualization of system logs and analytics.
3. Admin tools for feedback, overrides, and fine-tuning.
4. User and role management for secure access control.

2.2.7 Module 7 : Web UI

This module provides a simple and interactive interface for end-users to input prompts and receive safe responses. It acts as the main interaction point with the middleware.

1. Clean and user-friendly chat interface.
2. Real-time display of safe/filtered outputs.
3. Visual alerts when prompts or responses are flagged.

2.2.8 Module 8 : Plugin

This module allows the middleware to be integrated natively into existing or custom chatbot systems or LLMs, so users don't need a separate application.

1. Easy integration into UIs as a middleware plugin.
2. API-based communication for flexibility.

2.3 Tools and Technologies

For successful development and deployment of our project, we have carefully selected tools and techniques that match both the functional requirements and the research scope of FortiLM. These technologies will help us build a secure, scalable, and user-friendly system by covering frontend design, backend communication, AI model integration, databases, explainability, and deployment. The chosen stack ensures that each module of our middleware can be developed, tested, and deployed efficiently while keeping the system flexible for future improvements.

2.3.0.1 React.js

We will use React.js to build both the chat interface and admin dashboard. It will handle user input, show safe responses in real-time, and connect to our backend through APIs.

2.3.0.2 TailwindCSS

TailwindCSS will help us quickly style the web app and dashboard. Its utility classes make the interface responsive, clean, and modern without writing too much custom CSS.

2.3.0.3 Recharts / Chart.js

For the admin dashboard, we plan to use Recharts or Chart.js to plot logs and analytics. This will let us visualize flagged cases, attack trends, and system performance in graphs.

2.3.0.4 FastAPI

FastAPI will serve as our backend framework. It will route the prompt through privacy, detection, and filtering modules, and then return the safe output to the frontend via APIs.

2.3.0.5 PostgreSQL

We will use PostgreSQL to store structured data like user prompts, system logs, policies, and admin feedback. Its relational structure ensures secure and reliable data management.

2.3.0.6 MongoDB

MongoDB will store unstructured logs such as flagged examples or attack patterns. These records will later support the adaptive learning module for model improvement.

2.3.0.7 Hugging Face + PyTorch

For the AI part, we will rely on Hugging Face models with PyTorch. These will power classifiers like BERT for jailbreak detection and toxicity filtering, both training and inference.

2.3.0.8 SpaCy + Regex

Our Privacy Preserver will combine SpaCy and regex. SpaCy's NER can detect entities like names, while regex will catch structured data like phone numbers and emails for masking.

2.3.0.9 Detoxify + Perspective API

We will use Detoxify to locally classify text for toxicity, and optionally Perspective API for harmful content scoring. Together, they will help filter unsafe or biased outputs.

2.3.0.10 SHAP/LIME

For explainability, we plan to integrate SHAP/LIME as feasible. These tools will show why a decision was made, which tokens or rules triggered it, and give confidence scores for review.

2.3.0.11 MLFlow

MLFlow will manage our adaptive learning loop by tracking model versions and retraining runs. This ensures our classifiers improve with new flagged cases over time.

2.3.0.12 Docker

To keep deployment consistent, we will containerize our backend, models, and databases using Docker. This way, the system runs smoothly on both local setups and cloud servers.

2.3.0.13 AWS/GCP/Azure

Finally, we will deploy on a cloud platform like AWS, GCP, or Azure. These will host our Dockerized system, provide scalability, and allow secure enterprise-level integration

2.4 Work Division

The work is divided among team members based on their expertise and the modules of the proposed project. Each member is assigned modules and features that align with their strengths, ensuring balanced contribution and effective collaboration.

2.5 Timeline

The project is divided into four iterations, where each iteration corresponds to specific tasks and modules. The timeline with respective months is shown in Table 2.

Name	Tasks
Muhammad Ahmed	• Module 1: Hybrid Jailbreak Detection (Lead) • Module 2: Privacy Preserver (Support) • Module 2: Output Filter (Support) • Module 4: Explainability (Lead) • Module 5: Adaptive Learning (Support) • Module 6: Admin Dashboard (Lead) • Module 8: Plugin Integration (Lead) • Data Engineering & preprocessing
Irsam Maan	• Module 1: Hybrid Jailbreak Detection (Support) • Module 2: Privacy Preserver (Lead) • Module 3: Output Filter (Support) • Module 4: Explainability (Support) • Module 5: Adaptive Learning (Lead) • Module 8: Plugin Integration (Support)
Saifullah	• Module 2: Privacy Preserver (support) • Module 3: Output Filter (Lead) • Module 6: Admin Dashboard (Support) • Module 7: Web UI (Lead) • Module 8: Plugin Integration (Lead) • Data Engineering & preprocessing

Table 2.1: Work Division According to Modules

Iteration #	Time Frame	Tasks / Modules
01	Sept–Oct	• Dataset Collection • Build Web UI and Backend • Connect LLM with UI • Setup Database, FastAPI • Admin Dashboard (Basic)
02	Nov–Dec	• Implement Privacy Preserver • Implement Output Filter • Integrate into Dashboard
03	Jan–Feb	• Hybrid Threat Detection • Explainability Module • Pipeline Integration • Admin Dashboard (Advanced)
04	Mar–Apr	• Plugin development • Adaptive Learning Loop • Optimisation • Deployment • Final Testing

Table 2.2: Project Iterations and Timeline

Bibliography

[1] The Guardian, “Most major llms (chatgpt, gemini, claude) are easily tricked into unsafe outputs,” Guardian Article, 2025.

[2] Palo Alto Networks Unit42, “Single-turn jailbreak asr between 52–74%, multi-turn 43–46%,” Unit42 Report, 2024.

[3] Secludy Research, “19% of sensitive data leaks during fine-tuning without differential privacy,” Medium Article, 2024.

[4] IBM Security, “20% of jailbreak attempts succeed, often within 42 seconds,” *IBM Think Insights*, 2024.

[5] IBM, “Cost of a data breach report 2023–2025,” IBM Security Report, 2025, global average cost $4.45M (2023), $4.88M (2024), U.S. $10.22M (2025).

[6] Cybersecurity Dive, “Shadow ai breaches increase cost by $670,000 on average,” Cybersecurity Dive Report, 2025.

[7] Wired, “Gpt-4 can infer personal traits with 85–95% accuracy (race, location, occupation),” Wired Article, 2024.

[8] X. Zeng *et al.*, “Over 90% success in function-call jailbreak attacks,” *arXiv preprint arXiv:2407.17915*, 2024.

[9] Anthropic, “Many-shot jailbreaking increases success rates with prompt scaling,” Anthropic Research, 2023.