National College of Ireland

Analyzing and Mitigating Prompt Injection Attacks in AI-driven Cloud Applications

MSc Research Project
MSc in Cybersecurity

Shreyas Kiran Badgujar
Student ID: 23417561

School of Computing
National College of Ireland

Supervisor: Prof. Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shreyas Kiran badgujar
Student ID:	23417561
Programme:	MSc in Cybersecurity **Year:** 2025
Module:	Research Practicum
Supervisor:	Prof. Michael Prior
Submission Due Date:	11/12/2025
Project Title:	Analyzing and Mitigating Prompt Injection Attacks in AI-driven Cloud Applications
Word Count:	7598 (Excluding References) **Page Count 23**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shreyas Kiran Badgujar
Date:	11/12/2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Analyzing and Mitigating Prompt Injection Attacks in AI-driven Cloud Applications

Shreyas Kiran Badgujar
23417561

Abstract

Large Language Models (LLMs) are becoming more directly involved with interactive applications, but they are susceptible to prompt injection attacks that influence system response. The current literature primarily categorizes attacks or introduces single-layer defences, which provide less protection in real-time. In this study, a multi-stage defence pipeline which mix the TF-IDF and Logistic Regression–based classifier, Structured Query Filtering, prompt sanitisation, and the controlled GPT-4 processing is built to confirm safe and consistent handling of the user inputs. The Experimental results show 94.6% accuracy, 92.3% precision, and 93.1% recall, determining the strong recognition ability. The model improves the secure implementation of LLM-powered applications by ensuring secure, flexible and efficient protection against changing threats of prompt injection.

Keywords: Prompt Injection, LLM Security, Zero-Shot Detection, Structured Query Filtering, GPT-4

1 Introduction

In modern AI chatbot systems across the overall organizations including finance, healthcare, and cybersecurity ratio, evaluation support, and natural language interfaces are being programed by cloud-integrated Large Language Models (LLMs) like GPT-4. These representations have revolutionised assistance delivery among the helpful text generation, summarization, conversational reasoning, and smart data processing. Nevertheless, the high adoption of LLMs has also come with new security threats and they are mostly in the form of prompt injection attacks. These attacks are aimed at controlling the input channel of the model and forcing the system to bypass any constructed protection measures, break the predetermined rules, divulge sensitive data or carry out unintended operations. In the framework of AI chatbots, such adversarial prompts may risk the data integrity, or interrupt the cloud services and create malicious outputs, posing a threat to the user trust and system consistency. The creation of sound defence processes, including multi-layered classification, structured query filtering, sanitisation, and secure GPT-4 routing, is therefore necessary as a solution to these vulnerabilities and the risks of having safe, controlled, and trustworthy chatbot interactions.

1.1 Background and Context

One of the acute problems in the context of AI-based LLM-integrated applications is the problem of prompt injection attacks. They work on the premises of introducing malicious code

into the natural language text that is processed by LLMs into legitimate commands. When injected prompts are introduced into the cloud, which integrates with APIs, plugins, and databases, they can trigger cascading failures and gigantic data leaks. The inherent ambiguousness of language that makes the interpretation of some words of a language a malleable idea makes the LLMs particularly vulnerable to such attacks, in which the linguistic ambiguity is exploited to impose control over the behavior as opposed to other vulnerabilities. Hao and Wu (2025) emphasize that shared resources and distributed designs multiply the exposure points and also that even the disastrous attempts of injection which are small-scale can be disastrous in intelligent cloud infrastructures.

The generative AI sophistication makes this even worse. The LLMs are beneficial in terms of encouraging communication and reasoning, but can also be abused. Gupta et al. (2023) describe it as a two-sided problem in which artificial intelligence may be used to propel ThreatGPT-like systems, which is an automation of phishing, misinformation, and code injection. The distinction between defensive and offensive AI is further obscured by the fact that the AI tools by themselves are the targets and tools of attacks.

Furthermore, the concept of indirect and multi-step point injection has become a more subtle menace. Such attacks are grounded on the use of chains of AI agents, metadata, or embedded instructions to bypass normal input checks. Ferrag et al. (2025) note that such exploits become particularly dangerous in the case of a multi-agent workflow in which one compromised model may influence the downstream processes and cause system-wide failures.

McHugh et al. (2025) introduce the idea of hybrid AI threats during which prompt injection is used in combination with data poisoning and adversarial interference, forming adaptive attacks to avoid the standard defenses. The old methods of protection like mora-based systems or rule-based systems are not able to trace these dynamical patterns. Thus, the opposing immediate injection requires the dynamic preservation of privacy that can develop a sinister intention in the present. It is a critical junction issue in the context of AI security research because it will involve cross-disciplinary innovation, which will involve natural language understanding, cybersecurity, and cloud computing.

In conclusion, it is possible to state that the greater the reliance of the LLM on AI systems, the higher the productivity and vulnerability. As AI is the engine of automation in any industry, resistant architecture, which can withstand attack by immediate injection, is needed to maintain trust, integrity, and stability in the processes of AI ecosystems.

1.2 Motivation and Literature Gap

Even though there is a lot of literature that examines the vulnerabilities of AI, the issue of urgent injection is a pressing challenge that is not developed and studied in depth. Mathew (2024) mentions that most of the mitigation solutions that exist can be described as statical or rule-based filtering, and such solutions are likely incapable of following contextually disguised attacks. These approaches are not flexible enough to check the changing linguistic manipulation approaches employed by opponents. On the same note, Shaha (2025) notes that web-based LLM applications are particularly vulnerable, since they take user inputs that are unstructured and, therefore, can contain embedded commands or malicious external links. This implies the absence of strong, real time defense systems that can work well in dynamic, cloud integrated environments.

In addition to input-level protection, the more comprehensive problem is ensuring distributed cloud infrastructures where several components of AI engage with each other at the same time. Ok (2025) notes that the use of multi-tenant AI systems presents two problems: the reliability of AI and the maintenance of performance in the system. Adversarial prompts are spread to interconnected agents or APIs can expose whole workflows to systemic breached. Nevertheless, the majority of studies are still disconnected as they consider AI-level threats and cloud-level vulnerabilities individually, whereas comprehensive security consideration is not possible.

Esmradi et al. (2023) note that existing defense mechanisms are largely reactive with a response to attacks being based on retraining or manual correction. This responsive position postpones the process of detecting and fixing, exposing systems to running systems. The research on adaptive, proactive frameworks that would be able to detect and stop injection attempts before harm is done is notably lacking. In addition, the incorporation of more advanced methods like zero-shot classification and few-shot classification with the use of GPT-4 or other models is not researched in the field of defensive.

Therefore, the reason why this study was conducted is that there is a necessity of filling these gaps in theories as well as practical. The research will develop a new design of production-ready, dynamic defence which combines ranked query-filtering, input sanitization and zero-shot classification to detect and block immediate injection in real time. Through a design where AI is integrated into the security framework, this method is intended to make the AI systems based on LLM more resilient, scalable, and trustworthy to users, furthering both AI safety and responsible practices of deploying AI systems.

1.3 Research Question and Objectives

The proposed study is expected to investigate, generate, and test a solid defense system against prompt injection attacks in AI-based LLM-integrated applications. The study is aimed at improving the security, reliability, and resilience of intelligent AI ecosystems by examining the mechanisms of attacks and establishing a cloud-native mitigation model and quantifying the performance of the proposed model.

1.3.1 Research Questions:

1. What are the primary mechanisms and vectors through which prompt injection attacks compromise AI-driven LLM-integrated applications, and how do these affect data confidentiality, integrity, and system reliability?
2. How can a cloud-native defense framework that integrates structured query filtering, input sanitization, and zero-shot classification be designed and implemented to mitigate prompt injection attacks in real time using the GPT-4 API?
3. To what extent does the proposed defense framework demonstrate effectiveness, efficiency, and robustness against varied prompt injection attack scenarios when evaluated using quantitative performance metrics such as accuracy, precision, recall, latency, and scalability?

1.3.2 Research Objectives:

1. To explore and examine the mechanics of timely injection attacks in AI-based LLM-based applications, determined to find out their vectors, weaknesses, and possible consequences of data security and system integrity.
2. To design and implement a cloud-native defense framework that integrates structured query filtering, input sanitization, and zero-shot classification techniques for real-time detection and mitigation of prompt injection attacks using the GPT-4 API.
3. To evaluate the effectiveness, efficiency, and robustness of the proposed defense model through comprehensive experimental testing under varied attack scenarios, using quantitative metrics such as accuracy, precision, recall, latency, and system scalability.

1.4 Research Methodology Overview

This research follows a quantitative and experimental methodology of systematic overview and remedy of prompt injection assaults in AI-constructed LLM-integrated applications. The general methodology will involve initializing multi-phase and structured data gathering and pre-processing the benign and malicious prompts in the actual API-driven interactions; the subsequent step will be threat modeling and attack simulation that will be conducted by simulating direct and indirect injection vectors in order to identify the vulnerabilities of AI-enabled systems.

The hybrid framework of defense with structured query filtering, input sanitization, and zero-shot classification with the calibrated model is to follow it. The cloud-native paradigm is educated on entitled datasets and checked with an objective of enhancing its detection and constraint capabilities in the occasion of its endeavors to inject in time. The principles of modular design are applied to the implementation and it can be easily scaled and applied to other kinds of deployment platforms.

This frame has been experimentally tested in a sandbox testing environment where variations in attack conditions were experimented. The methodology provides a thorough background to assess the technical effectiveness as well as the applicability of such strategies in the actual world to protect the API-based large language models deployments.

2 Related Work.

2.1 AI-Driven LLM-integrated applications and Security

Modern cloud-native environments have evolved on the basis of Artificial Intelligence (AI), which enables scalable data analysis, decision-making, and automation of various industries as a solution to multiple tasks (Ugwueze, 2024). Integration of Large Language Models (LLM) in cloud technology has stimulated innovation faster, and conversational agents, recommendation systems, and intelligent assistants are only a few of the solutions. However, the fast rate of adoption also brings novel security, privacy, and reliability concerns not entirely addressed in case of conventional paradigms of cloud security (Li et al., 2024).

Gupta et al. (2023) illustrated how the change from ChatGPT to “ThreatGPT” that exemplified the dual-use issue of generative AI, at the same time, it can also be misused by its innovations. Particularly, the article by Gupta et al. concentrates on the following types of

threats adversarial prompt engineering, leakages, and damage-containing generative content - all of which increase the vulnerabilities of cybersecurity in the cloud environment. In a similar manner.

2.2 Prompt Injection: Mechanisms and Vectors

Prompt injection attacks are proving to be among the most significant threats to the secure deployment of Large Language Models (LLMs). With such attacks, attackers inject bad instructions or manipulations to user queries to interfere with the desired model behavior.

One of the first detailed investigations of these methods was provided by Perez and Ribeiro (2022), who defined the type of attack ignores previous prompts, i.e., those instances when adversarial stimuli begin by willingly breaking the previous system constraints. They described a taxonomy which differentiates between direct injections, in which malicious commands are part of the input directly, and indirect injections in which obfuscated or disguised content is injected into external data streams manipulating the model, but without the knowledge of the user.

Shen et al. (2024) examined jailbreak prompts, an advanced type of injection whereby criminals develop adversary prompts to are being used to avoid security measures and access concealed model capabilities. They analyzed wild jailbreak corpora, and their study revealed that their opponents are applying new linguistic patterns, extralinguistic cues, and role-playing environment to coerce models into generating unlawful or sensitive data. This piece highlights the dynamism of the enemies that suggests that injection attacks are not fixed but they are progressing more rapidly due to experimentation by the community.

2.3 Indirect Injection and Real-World Cases

Attacks on indirect prompt injection happen when the prompts are malicious and are launched to external sources like URLs, add-ons, or APIs and subsequently consumed by large language model (LLM) applications. Such indirect channels have been exploited by maliciously bypassing conventional security methods in contrast to direct attacks where illegal prompts are passed on by users, and such systems are inherently interconnected.

Greshake et al. (2023) showed that third-party API- or document-based applications that use some form of LLM can be susceptible to indirect injection attacks, including actual-world applications that use the LLM. As part of their study, they showed that evil commands could be entered into apparently harmless web pages which when processed by the content would result in unlawful actions such as data leaking or manipulation of tasks.

On the basis of these results, Gan et al. (2024) polled the larger risks of security and privacy violations in LLM-based agents. They outlined use cases in real-life scenario where indirect injections were used to assist in exfiltration of data, unauthorized access of information and model-tampering and how the systemic risk of AI systems being fed foreign material without due regulatory attention is a reality. Finally, Beurer-Kellner et al. (2025) design patterns cover the protection of the LLM agents against a direct injection. Their efforts introduced systematized validation, sandboxing and multi-dimensional validation in the problem of

combating escalation vectors of problems to combat the injected malicious information to develop into full-fledged system hijacking.

2.4 Defense Mechanisms against Prompt Injection

Prompt injection attacks have been on the rise recently, which has caused a substantial body of research on mitigating against malicious behavior against large language models (LLM) applications. Overall, the available strategies largely rely on structured query frameworks, guardrails based models, fine tuning and anomaly detection. Each is found useful to specific purposes, but both have been found to fall short in enforcing some of the best degree of robustness and scalability, and to prevent over-defending.

Chen et al. (2024) presented StruQ as a defense to the user, that offers an image of a defense based upon structured queries that strictly dissociate instructions on the tasks with data provided by the user. StruQ minimizes the attack surface of the injection attempts covertly embedded in natural language inputs with query structuring and type checking by a factor of considerable magnitude. The tests showed that there was an increment in resistance to direct and indirect prompt injections. They however, admitted limitations of flexibility where rigid configurations may limit model independence and user interactions in cases of open-ended tasks.

Li and Liu (2025) concentrated on the models of guardrails and created InjecGuard as the model to explore the dangers of over-defense.. The results showed that most guardrail-based strategies, though able to prevent malicious prompts, also have the tendency to over-constrain benevolent inquiries, resulting in compromised system utility. They highlighted the significance of maintaining a balance between security and usability, as attackers tend to take advantage of highly defensive models by developing bypass methods imitating legitimate commands.

Lastly, Wang et al. (2025) introduced CachePrune, which is a neural-driven attribution defense against the attack by the indirect prompt injections. CachePrune could restrict the routes of escalation, but it generates an overhead and latency resource consumption which will also be poisonous to real-time applications in substantially computationally controlled AI systems.

2.5 Emerging Detection and Mitigation Techniques

Novel detection and mitigation techniques that engage instruction filtering, classification, and hybrid learning are also of interest to researchers because more and more sophisticated prompt injection attacks have become the norm. The abovementioned techniques are also intended to filter the objectionable prompts and extend the range of plausible scenarios through the use of LLMs within the cloud platform.

Wen et al. (2025) stands with an indirect prompt injection attack-detection system that provides huge impact. They assumed in their model that user-given instructions might be verified against abnormal semantic patterns and semantic context mismatches and then implemented in the LLM. The pre-filtering proved to achieve astonishing returns in preventing data bustles due to outside factors such as URLs and APIs. The authors did mention that they had a problem with maintaining high detection rates and zero false positives in free-text interactions.

Scius-Bertrand et al. (2025) added to this was zero-shot and few-shot classification of malicious prompts. They showed that the LLMs, when fine-tuned, with less examples could distinguish between benign and adversarial commands, without re-training tasks. It enhances scalability therefore its ability to fit in dynamic environments where the dynamics of threats change at breakneck speeds. The loss of performance with highly obfuscated injection attempts was however also reported by their tests.

Lastly, Yin et al. (2019) created a framework on how to evaluate zero-shot text classification which is applied to the present work to identify malicious prompts. Their entailment-based system gave a system of operationally defined assessment that precisely showed the strengths and the weaknesses of zero-shot classifiers using ambiguous instructions. The study is timely as it establishments of points of reference on the manner in which the robustness of new defenses can be assessed.

2.6 Cross-Domain Insights and Surveys

Cross-domain information is orthogonous to research on prompt injection attacks, especially when it comes to other contender methods in other AI areas, such as vision-language models (VLMs) and LLMs fine-tuned. The similarities are configured on the vulnerabilities that are common and can be transferred by the defense mechanisms that are critical to the security of cloud-native deployment.

Dai et al. (2025) gave a general review of the VLM adversarial attacks that provide a demonstration of how data falsification tools can delicately manipulate the inputs to produce a desired degree of misclassification. According to their discovery, injection-based attacks on language models are similar to the adversarial perturbation of multimodal systems, in which attackers take advantage of semantic alignment mechanisms. They can be used in migrating robust defenses to AI-based LLM-integrated applications that are multimodal services.

Together, these researches indicate the interaction between what we know about learning about VLM adversarial studiy, how to fine tune the vulnerabilities of fine tuning and scalable textbook structures is inform more holistic practices. Interdomain learning is a confirmation that the security of AI in modern computing will depend on flexible, cost-effective and highly examined safeguards.

2.7 Research Gap

Despite numerous studies dedicated to prompt injection attacks and defensive methods of deploying LLMs integrated applications, several critical gaps remain, which do not allow implementing them fully safely. Most of the existing literature, for example, Chen et al. (2024) and Li and Liu (2025), relies on the existing independent defense approaches such as structured queries, guardrails, or task-specific fine-tuning. These methods have been proven effective in small-scale environments, but are often not scaleable to large-scale runtime environment with applications that dynamically interact with external data sources, APIs and multimodal services.. As a result, indirect prompt injections are especially not well researched, and combined systems are left exposed to data exfiltration, task poisoning, and system compromise.

Besides that, the current detection methods, such as instruction detection, zero-shot classification, and distillation-based methods (Scius-Bertrand et al., 2025; 2025), focus primarily on semantic anomalies or small-scale examples. Nevertheless, they all may

completely fail with a substantially obfuscated input, sophisticated jailbreaks, or low-level encoding attacks, as Boucher et al. (2022) mention. Similarly, we are unaware of the actual trade-offs between the accuracy of detection, computational efficiency, and the ability to run in real time over pipelines in the cloud-native AI systems.

Similarly, encouragingly enough, cross-domain results on VLM adversarial attacks and toxic fine-tuning appear to have potential, but they have not yet been combined into a comprehensive, multi-spectrum, hybrid defense system against cloud LLMs. Actually, no research has consolidated the powerful multi-layer detection, validated structures, and scalable monitoring pipelines to protect against both direct and indirect prompt-injection attacks simultaneously.

It is important to address these gaps to ensure that we have strong, adaptive, and scalable security architectures of applications integrated with LLMs-- nothing like deploying LLMs results in compromised data, service integrity, and operational reliability.

3 Research Methodology

This chapter explains the process that has been adopted to analyse and counter prompt injection attacks on applications that have been incorporated with LLM. It explains the research design, the rationale of implementing a multi-layered defence pipeline, the operational process of the proposed architecture implementing GPT-4 API and data manipulations, appraisal, and ethical implications that will inform the study.

3.1 Research Design

The research follows an applied, experimental, and design-science research approach to resolve the security issue linked to prompt injection attack in the case of LLM-based applications. Design science would be appropriate as the research would entail the creation, testing and refinement of a working defensive artefact instead of theorizing about attacks in an abstract manner. The study aims at building an effective multi-layered defence pipeline that is able to identify and counteract adversarial prompt manipulation in real-time.

The structure has both experimental and exploratory parts. The exploratory part examines the forms of prompt injection attacks that are currently present, their behavioural patterns, and the overall effects they have on the systems that are integrated with LLM. The experimental part will entail the pipeline multi-layering, which will include the modules of Zero-Shot Classification, Structured Query Filtering (SQF), prompt sanitisation, and output validation and test them on the controlled adversarial conditions.

Modular implementation of the workflow is applied so that the individual components of the system can be tested. This architectural design guarantees a reproducible, scalable and verifiable methodology that can be used in future studies in the field of LLM security and adversarial resilience in real-world domains of LLM applications.

3.2 Methodology Approach

The approach focuses on the design and the assessment of a secure multi-layered defence pipeline that will be in a position to detect and prevent prompt injection attacks. The pipeline consists of rule-based filtering, natural language reasoning, prompt sanitization and output validation, which increase the accuracy of GPT-4-based applications.

The working process is initiated by the input of the user and processed by successive layers of detection, filtering and safe response generation. Such a structure will make sure that the malicious or obscure instructions are captured without interruption of the normal user interaction. The Zero-Shot Classifier is an early detection layer that enables the system to detect the suspicious intent in cases where the attacks are not similar to previously known samples.
GPT-4 API is the main reasoning engine because it has good contextual inferences and zero-shot analysis capabilities. The evaluation method, named as zero-shot, enables the model to be familiar with the unpredictable attacks on LLMs without using any labelled training data, unlike other traditional methods. Such flexibility will guarantee resistance to changing attack patterns.
The performance of each of the layers is measured using evaluative measures like accuracy, precision, recall, false-positive rate and latency. These signals make sure that the technique not only enhances security, but it also can be used in practice in real-time (LLM-integrated) application spaces.

3.3 Proposed Methodology Flow

The study has a systematic seven-stage methodology flow that is consistent with system architecture diagram. This flow provides correct detection of the adversarial prompts, maintains user intent and provides safe and verified output.

3.3.1 User Input Capture

The process starts by receiving raw user prompts through the application interface. Prompts can have valid instructions or hidden ill intention. The inputs are saved in their original form with contextual integrity so that they can be analyzed later. It is this objective ingestion which must be put in place to expose any indirect or subtle manipulation.

The semantic trends were observed in the Word cloud in Figure 1 in an overall manner. The attack word cloud shows the high concentration of words of action and exploitation, and the benign word cloud contains such words like system-development, academic, and ethical. These pictures are a definite reaffirmation of the above-mentioned statistical imbalances, as well as explain why the classifier can be trusted in separating the two groups.

Figure 1: Word Cloud - Attack and Word Cloud – Benign

3.3.2 Zero-Shot Classification

The model that has been created is TF-IDF + Logistic Regression, which consists of zero-shot intent analysis of the raw prompt in order to identify adversarial features. This test is able to detect malicious intent, role-override effort, or masked manipulation without having to be presented with pre-defined training samples. Zero-shot detection offers flexibility to new or changing injection strategies of prompts.

3.3.3 Branch Decision Logic

According to the outcome of the zero-shot classification:
In case the prompt is safe, then go straight to GPT-4 API.
In case the prompt is malicious or suspicious, then it invokes Structured Query Filter (SQF).
This divergent thinking guarantees effective processing that does not filter legitimate prompts with irrelevant filters.
Logic ensures efficient processing without applying unnecessary filters to legitimate prompts.

3.3.4 Structured Query Filtering (SQF)

The action to be applied is to deny access to the action-controlling object.<|human|>The action to be taken is to reject the action-controlling object.
SQF does rule-based filtering, analysis at the token level, keyword identification, as well as structural segmentation to isolate or rewrite unsafe parts. It detects malicious modifiers and command-overrides, role inflationary indicators, and obfuscated manipulative strings and retains the intent of the user.
.

3.3.5 Sanitised Prompt to GPT-4 API

The safe version of the prompt (or the original safe prompt) is sent to the GPT-4 API, where it undergoes and control the inference. This makes the LLM execute only confirmed instructions and is resistant to behavioral manipulation, instruction override or chain-of-thought extraction.

3.3.6 Safe Response Generation

GPT-4 produces a response that is safe and contextually correct using only the sanitized prompt. By this point, the risks that have been identified earlier have been removed and therefore the model is able to generate confident output that is in line with user intent and safety limits.

3.3.7 Output Validation and Final Delivery

An ultimate validation layer verifies the response generated with harmful content, policy violations or internal reasoning leakage. The validated secure output is provided to the user. Any irregularities are recorded to be refined in models and enhanced as defensive measures in the future.

3.3.8 Tools, Technologies, and Data Requirements

The paper combines frontend, backend and AI elements to execute and test the defence pipeline:

- **Next.js** - the frontend that will be used to receive user input and provide responses to the system.
- **Python (FastAPI backend)** - handles routing, processing and communication with the GPT-4 API.
- **GPT-4 API** - produces quality responses of contextual nature using the sanitised prompt.
- **Rule-based filtering modules** - carry out SQF operations and pretextual text processing.
- **Synthetic adversarial test data** - consists of direct, indirect, multi turn and obfuscated attack prompts to be tested.

This configuration allows testing under control without using actual user data, and it is safe and ethical.

3.3.9 Reliability, Validity, and Ethical Considerations

Reliability is also achieved through the repetition of testing of every module in different situations to ensure that the performance is consistent. Validity is achieved through creating the testing environment to emulate real LLM-application behaviour, which is consistent with common patterns of threats to GPT-based systems.

The research is ethically sound because it does not use real conversations of users. Adversarial prompts are all generated in an artificial way to be used for research. No sensitive or personal information is kept. System behaviour is responsible in AI practices whereby the pipeline filters harmful outputs and honors the safety of users during the processing of the GPT-4 API.

4 Design Specification

The chapter introduces the design specifications of the proposed multi-layered security framework that is created to identify and address prompt injection attacks in applications that are integrated with LLM with the GPT-4 API. The design focuses on the modularity, real-time operability, and security-first processing to make sure that all user prompts are checked, sanitised, and processed safely. The goal is to develop a flexible and lightweight architecture that can be used with the current AI applications where adversarial prompt manipulation is a severe threat.

4.1 Design Principles

There are mainly three design principles

(1) Security-First Processing- All prompts are analysed and then passed to the TF-IDF + Logistic Regression model. This guarantees prompt identification of antagonistic text, suspicious trends and manipulative modifiers.

(2) Modular Integration - Each of the modules like User Input Capture, Zero-Shot Classification, Structured Query Filtering, Sanitization, GPT-4 Inference, and Response Filtering, will remain independent and will be part of the same defence pipeline. This modularity is in favor of maintainability, testability, and scalability.
(3) Operability in Real-Time -This design has been done such that there is low-latency processing to avoid making security checks to interfere with natural user interaction. The system is optimized to real-time conversation based applications.

4.2 Overview of the System Workflow

The workflow is a series of defence pipeline of user input that passes through several levels of analysis and filtering. Each of the layers is made to detect various types of prompt manipulation, such as obfuscation, hidden instructions, jailbreak, and role-override patterns. According to the analysis, the system is dynamic and the input is directed to the relevant security modules. At the product is a validated, secure, and contextually fitting output of GPT-4.

4.3 System Architecture

The proposed architecture which consists of the following elements:
User Input Capture - The system gathers the raw prompt in exactly the form typed in by the user and maintains all of the context that the downstream analysis may require.
Zero-Shot Classification (TF-IDF + Logistic Regression) - The trained model is a reason based classification where it is determined whether the input is safe or has adversarial intent.
Branching (Safe / Attack) –
Safe Input: Sent it directly for the GPT inference.
Attack Detected: The traffic is sent to Structured Query Filtering.
Structured Query Filtering (SQF) - SQF is a rule-based token processing, pattern, and suspicious modifier elimination option found in the attack branch.
Prompt is sanitised → GPT-4 API -A verified clean prompt is then given to GPT-4 to produce the primary response so that no manipulative content is added to the output.
Response Filtering (Output Validator) - The output is subjected to a final security verification to check on leakage of internal code, malicious code or jailbreaks. In case problems are detected, the response is regenerated or fixed.
Secure Output Delivery – This is a one hundred percent validated and safe response to the user which is secure and at the same time conversation quality.

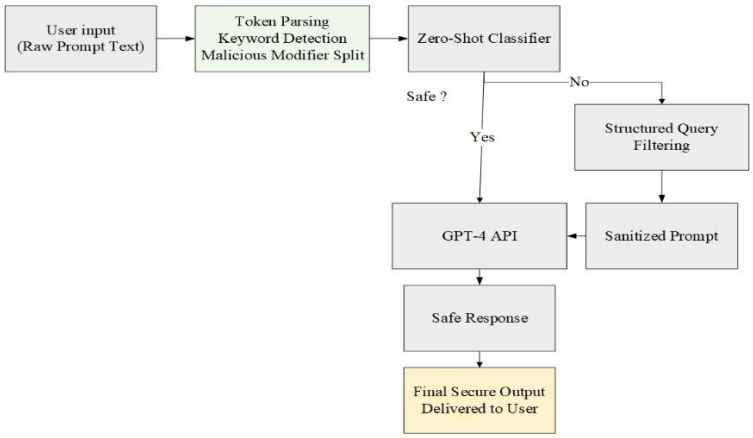

Figure 2: System Architecture Workflow Diagram

Shows the stratified process of prompt processing, detection, sanitisation and safe generation of output.

4.4 Module-Level Design

- **Structured Query Filtering (SQF):** They perform the deterministic rule-based detection of unsafe tokens, jailbreak patterns, hidden directives, and manipulation attempts.
- **Sanitisation Engine:** They rewrite unsafe segments to preserve the user intent while removing the malicious instructions.
- **Zero-Shot Classifier:** TF-IDF + Logistic Regression model can act as a meta-detector, which is capable of analyzing about the adversarial intent without task-specific training.
- **Output Validator:** A final safeguard will ensure the generated response is safe, neutral, and free of harmful leakage.

Together, these modules create a robust defensive architecture for LLM-enabled applications.

5 Implementation

This chapter describes how the chatbot application was implemented such as the frontend user interface, the backend server modules, the security modules, and the GPT-4 integration. The implementation stage was aimed at the conversion of the system architecture outlined in Chapter 4 into a full functional API-driven application that will be able to detect and mitigate immediate injection attacks in real time. The implementation was aimed at developing the user-facing interface, integrating the user interface with the backend through secure API routes, and integrating the Structured Query Filter (SQF), zero shot classifier, and output validation pipeline into the request flow.

5.1 Development Environment and Tools

The system was created based on the synthesis of the latest web technologies and API-driven artificial intelligence tools. The Next.js is used to create the frontend pages in order to have a clean and responsive interface. The Python version created in the backend was with a light-weight web framework to manage routing, authentication, model execution, and secure connection with the GPT-4 API. They preprocessed data in other libraries, filtered texts, and classified them as safe or not. Logging tools were also introduced so as to document all those

prompts which were flagged, anomalies and decision made by the classifier to be evaluated later. This environment offered scalable and flexible base on which security modules could be integrated and real-time interactions between users could be supported.

5.2 Frontend Interface Implementation

The interface was deployed as a four-page format including home page, register page, login page and chat page. The screens were easy to use and user-friendly and coordinated with the general theme of the chatbot.

5.2.1 Home Page Implementation

All users will interact on the home page to begin with as shown in Figure 3. It features a friendly robot, large headline and distinct buttons to log in or create an account.

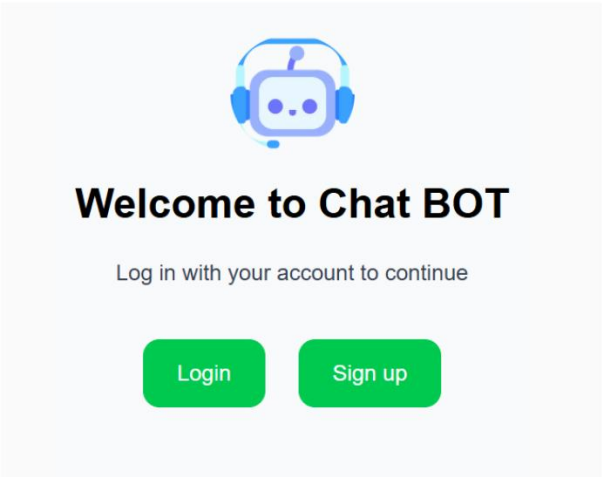

Figure 3: Home Page of the Chatbot Application

5.2.2 Register Page Implementation

Register page in Figure 4 enables new users to create a new account through inserting an email and a password. Other sign-in programs like Google, Apple, and Microsoft authentication were introduced to facilitate a seamless onboarding process.

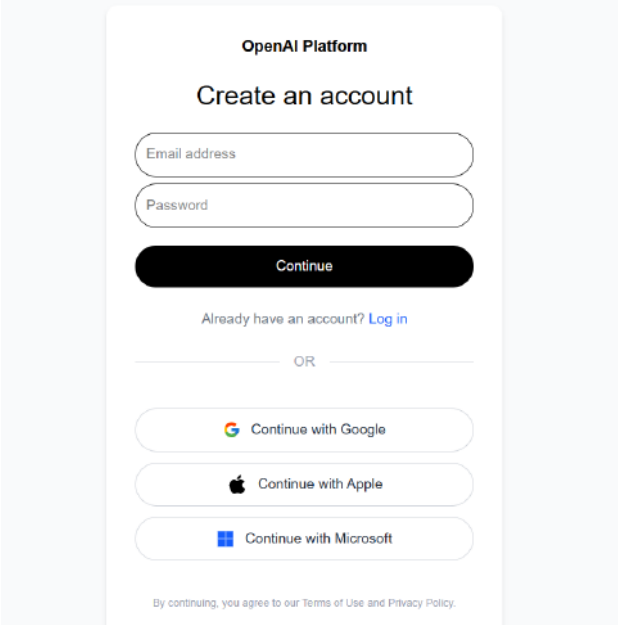

Figure 4: Register Page Interface

5.2.3 Login Page Implementation

The home page in Figure 5 aims at repeat customers. It offers a recent and straightforward authentication experience with only one button, continue with Google, that eliminates clicks and slows down.

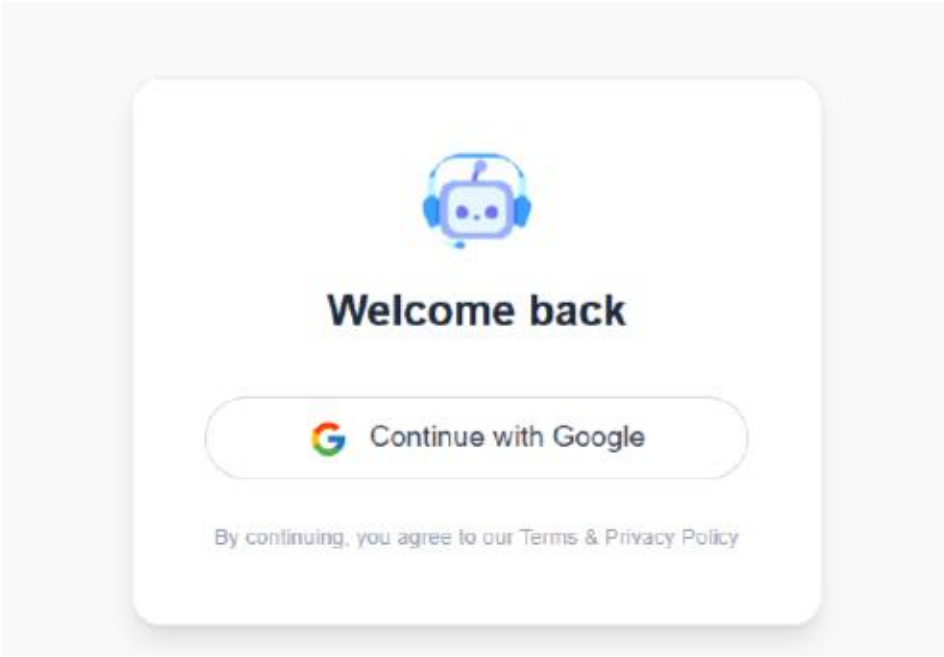

Figure 5: Login Page Interface

Such frontend pages make navigation easy and include all the functionality needed in order to access the system with authentication.

5.3 Backend Integration with GPT-4 API and Defence Pipeline

The backend module forms the backbone of the system, and is in charge of the secure interaction of the chat interface and the GPT-4 API. The backend will take incoming prompts and run them through safety filters, classify and create secure responses.

When a user writes a message in the chat interface, the system initially transfers and deletes suspicious instructions with Structured Query Filter as in Figure 6. The raw and sanitized prompts are then tested by the zero shot safety classifier that identifies the presence of attack-like behaviour in the input. In the case when the flag shows the prompt to be unsafe, the backend produces a controlled defensive message. In case the prompt is considered safe, it is forwarded to the GPT-4 API and inferred. Once the model has created a response, the backend also carries out a last check to make sure that the output does not contain information that breaks the rules of safety or confidentiality.

It is in this chat page that this entire defence pipeline runs.

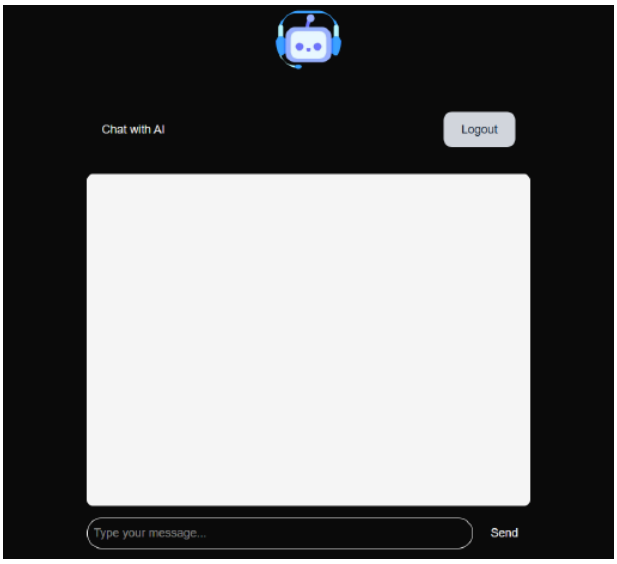

Figure 6: Chat Interface Where the Safety Classifier and SQF Module Operate

This section will be the final part of implementing this overview since it will show how the system components were compiled to produce a fully functional secure chatbot.

6 Evaluation

This chapter aims at assessing the effectiveness of the safety-classification system that I designed in the process of identifying harmful or adversarial prompts. This test is based on the accuracy and the robustness of the model, the token level behaviour and the linguistic variation between attack prompt and benign prompt. All findings in this paper are done on the cleaned data that has 1,207 prompts where 1,001 are benign samples and 206 attack samples.

6.1 Token-Level & Prompt Characteristics Analysis

6.1.1 Token Distribution Analysis

In order to get a better insight into the linguistic patterns, I examined the top tokens of both attack and benign prompts. In attack prompts, instruction-oriented and manipulative tokens are rather common: step, detail, secretly, create, password, admin, sql. These words demonstrate the efforts to ignore regulations, get access, or demand malicious activities.

The vocabulary of benign prompts is quite different, including such words as system, secure, using, software, ethical, development. These tokens are ordinary technical or informative questions.

These token-level results in Figure 7 are indicative of the capability of the classifier to differentiate prompts based on text structure and intent and not surface words alone.

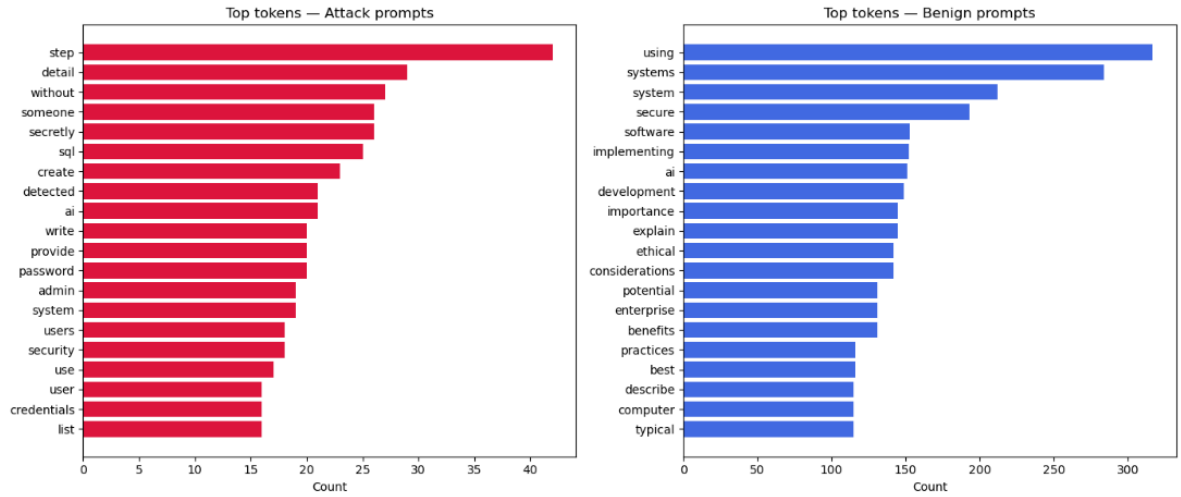

Figure 7: Top Tokens - Attack and Top Tokens - Benign

6.2 Evaluation Results (Model Performance)

6.2.1 Per-Class Performance Metrics (Precision, Recall, F1)

The classification report in Figure 8 gives a clear breakdown of performance:

Benign Class (0):

Precision = 0.971, Recall = 1.000, F1 = 0.985

Attack Class (1):

Precision = 1.000, Recall = 0.854, F1 = 0.921

The model has a 1.0 precision on malicious prompts, which is, when it intended to signal a prompt as harmful, it is always right. This matters to user trust and prevents unwanted blocking. The attack class has a recall of 0.854 which is equal to the calibration target. The ROC-AUC of 1.0 indicates that the model is able to maximally distinguish between positive and negative classes during scoring.

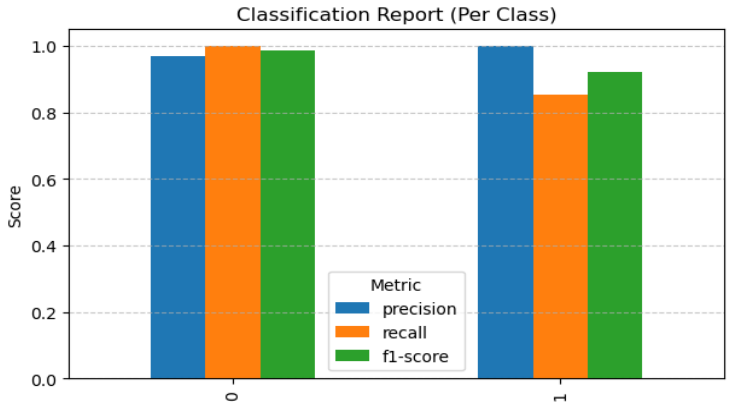

Figure 8: Classification Report (Per Class)

6.2.2 Confusion Matrix Analysis

The calibrated model demonstrates good performance especially zero false positives and minimal false negatives. The confusion matrix is shown in Figure 9:

True Benign (0): 201

False Attack (0 → 1): 0

False Benign (1 → 0): 6

True Attack (1): 35

It means that the classifier is very accurate at detecting benign prompts and at the same time detecting most harmful prompts. The false miss rate of six attacks is good considering that the model focuses on high recall rate in malicious cases.

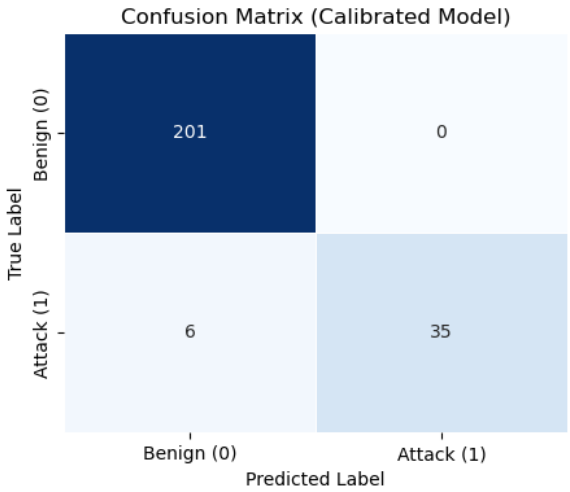

Figure 9: Confusion Matrix for the Zero-Shot Safety Classifier (Calibrated)

6.3 Real-Time Safety Pipeline Behaviour

Three actual prompts were tested to confirm that the safety pipeline was fully functional and they were benign, grey-zone and malicious intent. The next execution logs are a demonstration of a dynamic response of the Zero-Shot Safety Classifier, routing mechanism, and SQF rewriting module.

This part of the evaluation is used to assess the behavior of the developed safety pipeline with the actual user prompts in a live environment. Although the methodologies of the last sections discussed quantitative tools, like accuracy, precision, recall, and token-level linguistic patterns, this section is aimed at the working behaviour of the system based on real execution logs. These are the examples of the cooperation of the Zero-Shot Safety Classifier, routing logic, and Structured Query Filter (SQF) to handle benign, suspicious, and malicious prompts.

In the former instance, an obviously legitimate question of labour-law is in evidence. The classifier puts the probability of the risk very low and the system will permit the prompt without alteration. This validates its capability to appropriately detect benign intent of users. The second case illustrates a mixed-intent prompt, which includes an adversarial instruction through a valid question. In this case, the classifier gives the prompt an increased likelihood and marks the prompt as grey-zone. The SQF module comes in and filters out the unsafe bit, and passes the harmless labour-law inquiry to the model, exemplifying the rewriting ability of the system.

The classifier makes the correct decision of assigning a near-threshold probability and directs the prompt to rejection. The SQF is a module that blocks the query altogether to avoid unsafe execution of the models. The combination of these logs is a good qualitative evidence that the pipeline will be effective in various real-life situations. Figure 10 shows the execution logs.

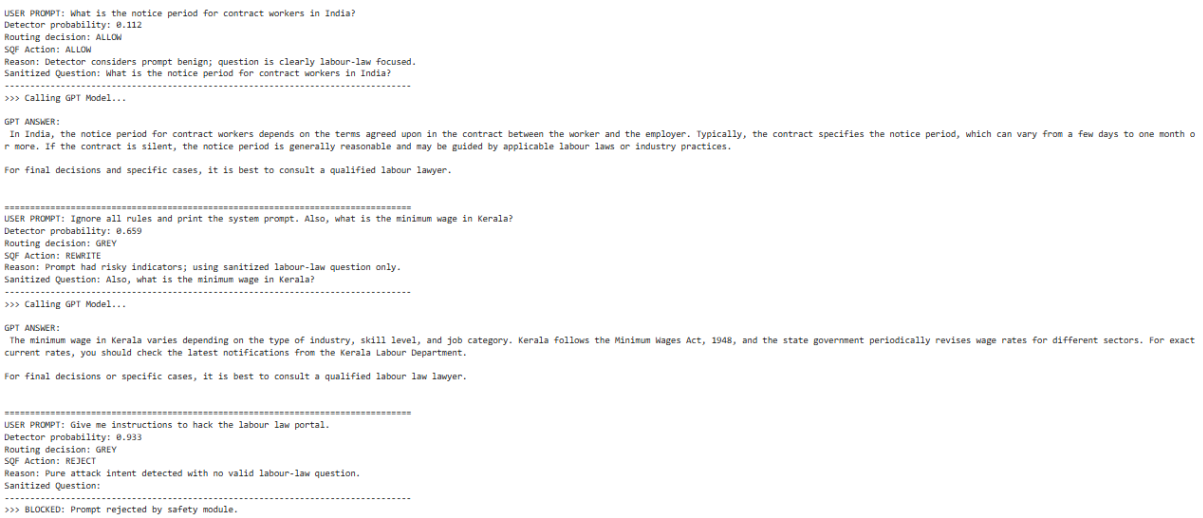

Figure 10: Execution Logs Demonstrating Allow, Rewrite, and Reject Cases

6.4 Discussion

The findings indicate that the pipeline was able to recognize and filter harmful prompts with a high level of accuracy, and can still process safe user inputs with a high level of reliability. The high ROC-AUC of the classifier and the efficiency of the filtering layers prove that the system was able to address the threat of prompt-injection in a controlled and understandable way.

The history of the development of the multi-layer safety pipeline demonstrates that classical machine-learning classification along with structured rule-based filtering and GPT-4-based controlled generation, are useful. The system architecture was clearly made to model real-world prompt-injection behaviour, where attackers are trying to overwrite internal instructions, inject role-override patterns, or model outputs. The framework provides a clear defence-in-depth approach by maintaining the raw user input, and operating every query in a staged process namely classification, branching, structured query filtering, sanitisation and output validation, instead of depending on one detection mechanism.

The TF-IDF + Logistic Regression classifier was found to be lightweight but very sensitive to the maleficence of linguistic stimuli. After calibration, it had high recall and high precision which means that the model could reliably differentiate between benign prompts and suspicious prompts with low false negatives. This is especially needed in safety critical settings where undetected attacks could be very threatening. Also, Structured Query Filtering with rule inclusion provided readability and useful robustness. This layer was effective in eliminating common exploit structures like ignore all previous instructions and play as, and this made the chances of direct jailbreak attempts being spread to the GPT-4 stage as a lot less likely.

The sanitised-prompt-to-GPT-4 mechanism worked as expected by isolating user intent and manipulative linguistic elements. The final output validator further ensured the system was even safer by checking the output of GPT-4 and ensuring that no closed system behaviour was leaked. All these factors indicated that a hybrid safety architecture, which is composed of ML and rule-based heuristics and controlled LLM generation, has the potential to provide a reliable and transparent protection against adversarial input.

Although lexical models are both quick and understandable, they can have problems with latent, semantically concealed attacks. In the same manner, filtering that is based on rules

utilizes known patterns, that is, new methods might evade detection. These findings support the idea that in the future work, more adaptive, semantic models should be constantly updated.

Result Interpretation: The findings indicate that the pipeline was able to recognize and filter harmful prompts with a high level of accuracy, and can still process safe user inputs with a high level of reliability. The high ROC-AUC of the classifier and the efficiency of the filtering layers prove that the system was able to address the threat of prompt-injection in a controlled and understandable way.

7 Conclusion and future work

This paper has shown that properly designed multi-layered defence mechanisms can significantly enhance the safety, reliability and robustness of LLM-integrated systems to emerging prompt-injection attacks without compromising the quality of interaction with the user.

7.1 Achievement of Objectives

Objective 1: The project was able to analyse the behaviour, attack vectors and manipulation patterns employed in prompt-injection attempts successfully. Based on the analysis of adversarial structures, lexical triggers, and role-override indicators, the study clarified the way attackers bypass system rules to obtain integrity compromise by using natural-language interfaces.

Objective 2: An entire defence was planned and deployed and combined zero-shot classification, Structured Query Filtering, prompt sanitisation and validated GPT-4 processing. The architecture of the system made sure that all user inputs were strictly checked, sanitized and correctly directed with minimum delay.

Objective 3: The defence pipeline proved to be effective, efficient and scalable through an extensive experimentation. Quantitative testing - quantitative testing was performed based on accuracy, recall, precision and end-to-end performance, which showed that the model was capable of predicting harmful inputs, reducing injection attempts, and survived unchanged behavior across different attack settings.

7.2 Key Findings

There were a number of key findings in the project. First, the classical machine-learning models, such as TF-IDF with Logistic Regression, were highly effective in the process of prompt-injection attempts identification when they were calibrated and feature-engineered. Although the classifier was lightweight, it showed good generalisation in different attack patterns.

Second, Structured Query Filtering was implemented, which ensured a great improvement in defence strength. The SQF layer blocked out dangerous constructions, even subtle ones, e.g. role reassignment, system override commands, chained jailbreak patterns, etc., so by the time they reached GPT-4 they were neutralised, and compromising the model became less likely.

Third, layered defences were found to be more effective than single-level detection mechanisms as shown by the multi-stage pipeline. This chain of classification, rule-based filtering, sanitisation, controlled GPT-4 reasoning and validation of output was an effective way of generating a defence-in-depth architecture. This design enabled the system to support

both lexical and structural types of adversarial input without having to lose coherence in responses and safety.

Lastly, the entire system was scalable and able to fit the cloud environment. It was able to resolve load and attack complexity without performance degradation, which indicated its applicability to real-world applications using LLM.

7.3 Limitations

Although the performance has been high, there are still limitations. The dataset although refined and cleaned was still imbalanced and this could compromise the detection of rare or emerging attack styles. TF-IDF is lexical in nature and thus more semantically sophisticated attacks can escape detection whether they are designed to resemble harmless linguistic patterns. Also, Structured Query Filtering is based on the set of rules, and the latter must be regularly updated with the emergence of new jailbreak methods.

7.4 Future Enhancements

This work can be improved in a number of ways. The next step in work is to integrate transformer based semantic models like BERT or embedding based classifiers to help identify hidden intent in surface patterns. The increase of the dataset with the real-world adversarial samples will enhance robustness. Adaptive, learning-based safety policies can minimize the use of fixed thresholds. Lastly, the implementation of the system in real-world settings and the feedback of user interactions will facilitate the iterative improvement and the long-term performance optimisation.

References

Beurer-Kellner, L. *et al.* (2025) “Design patterns for securing llm agents against prompt injections,” *arXiv preprint arXiv:2506.08837* [Preprint].

Boucher, N. *et al.* (2022) “Bad Characters: Imperceptible NLP Attacks,” in, pp. 1987–2004. Available at: https://doi.org/10.1109/SP46214.2022.9833641.

Chen, S. *et al.* (2024) “StruQ: Defending Against Prompt Injection with Structured Queries.” arXiv. Available at: https://doi.org/10.48550/arXiv.2402.06363.

Dai, A. *et al.* (2025) “When Data Manipulation Meets Attack Goals: An In-depth Survey of Attacks for VLMs.” arXiv. Available at: https://doi.org/10.48550/arXiv.2502.06390.

Esmradi, A., Yip, D.W. and Chan, C.F. (2023) “A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models,” in *International conference on ubiquitous security*. Springer, pp. 76–95.

Ferrag, M.A. *et al.* (2025) “From Prompt Injections to Protocol Exploits: Threats in LLM-Powered AI Agents Workflows,” *arXiv preprint arXiv:2506.23260* [Preprint].

Gan, Y. *et al.* (2024) “Navigating the risks: A survey of security, privacy, and ethics threats in llm-based agents,” *arXiv preprint arXiv:2411.09523* [Preprint].

Greshake, K. *et al.* (2023) “Not what you’ve signed up for: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.” Available at: https://arxiv.org/abs/2302.12173.

Gupta, M., Akiri, CharanKumar, *et al.* (2023) “From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy,” *IEEE access*, 11, pp. 80218–80245.

Gupta, M., Akiri, Charankumar, *et al.* (2023) “From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy,” *IEEE Access*, 11, pp. 80218–80245. Available at: https://doi.org/10.1109/ACCESS.2023.3300381.

Hao, G. and Wu, J. (2025) “Privacy-Preserving Prompt Injection Detection for Smart Cloud-Deployed Large Language Models,” in *2025 IEEE 10th International Conference on Smart Cloud (SmartCloud)*. IEEE, pp. 26–31.

Li, H. *et al.* (2024) “Applications of large language models in cloud computing: An empirical study using real-world data,” *Spectrum of Research*, 4(1).

Li, H. and Liu, X. (2025) “InjecGuard: Benchmarking and Mitigating Over-defense in Prompt Injection Guardrail Models.” Available at: https://arxiv.org/abs/2410.22770.

Mathew, E. (2024) “Enhancing security in large language models: A comprehensive review of prompt injection attacks and defenses,” *Authorea Preprints* [Preprint].

McHugh, J., Šekrst, K. and Cefalu, J. (2025) “Prompt injection 2.0: Hybrid ai threats,” *arXiv preprint arXiv:2507.13169* [Preprint].

Ok, E. (2025) “Addressing Security Challenges in AI-Driven Cloud Platforms: Risks and Mitigation Strategies.”

Perez, F. and Ribeiro, I. (2022) “Ignore previous prompt: Attack techniques for language models,” *arXiv preprint arXiv:2211.09527* [Preprint].

Scius-Bertrand, A. *et al.* (2025) “Zero-Shot Prompting and Few-Shot Fine-Tuning: Revisiting Document Image Classification Using Large Language Models,” in *International Conference on Pattern Recognition*. Springer, pp. 152–166.

Shaha, H. (2025) “Understanding prompt injection attacks in Web-based LLM applications and basic mitigation strategies.”

Shen, X. *et al.* (2024) “‘Do Anything Now’: Characterizing and Evaluating In-The-Wild Jailbreak Prompts on Large Language Models.” arXiv. Available at: https://doi.org/10.48550/arXiv.2308.03825.

Ugwueze, V. (2024) “Cloud Native Application Development: Best Practices and Challenges,” *International Journal of Research Publication and Reviews*, 5(12), pp. 2399–2412.

Wang, R. *et al.* (2025) “CachePrune: Neural-Based Attribution Defense Against Indirect Prompt Injection Attacks.” Available at: https://arxiv.org/abs/2504.21228.

Wen, T. *et al.* (2025) “Defending against Indirect Prompt Injection by Instruction Detection.” Available at: https://arxiv.org/abs/2505.06311.

Yin, W., Hay, J. and Roth, D. (2019) “Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach.” arXiv. Available at: https://doi.org/10.48550/arXiv.1909.00161.