National College of Ireland

Configuration Manual

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
Lecturer:	Prof. Michael Prior
Submission Due Date:	11/12/2025
Project Title:	Analyzing and Mitigating Prompt Injection Attacks in AI-driven Cloud Applications
Word Count:	991 **Page Count: 07**

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):	

Configuration Manual

Shreyas Kiran Badgujar
Student ID: 23417561

1 Introduction

This configuration manual gives a step by step instruction on how to install, operate, and implement the Prompt Injection Detection System that was created in this study. It consists of a TF-IDF + Logistic Regression classifier, a calibrated safety component, and a cloud-based prompt detection and mitigation interface. This manual deals with setting up of the project, tools necessary, initial environment setup, security validation and deployment instructions. It contains also necessary code bits and the important output numbers that required like the confusion matrix, per-class measures, top-token analysis, and word clouds aid in visualising model behaviour.

In addition to the detection pipeline, this project also includes a fully functional prototype chatbot interface built specifically to demonstrate and validate the Prompt Injection Detection System. For prototyping purposes, a lightweight Next.js chatbot UI was integrated with GPT-4 acting as the core LLM backend. On top of this interface, the complete security stack was implemented—zero-shot safety classification, the Structured Query Filter (SQF) mechanism, dynamic prompt sanitisation, and routing logic. This custom-built chatbot served as the controlled environment for testing benign prompts, semi-malicious attempts, and direct injection attacks, allowing real-time observation of how the detection layers respond. The prototype is not intended as a production chatbot but as an experimental test platform to evaluate system behaviour, mitigation performance, and attack handling under realistic user interaction conditions.

2 System Setup Overview

The system follows a modular pipeline:

1. Dataset Loading & Cleaning
2. TF-IDF Feature Extraction
3. Attack Prompt Visualization & Token Analytics
4. Logistic Regression Classification
5. Calibration & Threshold Selection
6. Frontend Integration (Home, Register, Login, Chat)
7. Backend integration (python FastAPI)

8. Structured Query Filtering
9. GPT-4 API Integration
10. Prompt Injection testing
11. Final Deployment

The common configuration includes the following steps: 1) The cleaned dataset should be put in the data/ folder of the project and run the preprocessing script.

```
import pandas as pd

df = pd.read_csv("shield_combined_cleaned.csv")
print("Total Rows:", len(df))
print(df.head())
```

3 Tools & Technologies Used

Your system requires the following components:
Software & Libraries

- Python 3.10
- scikit-learn
- matplotlib
- wordcloud
- nltk
- Flask / FastAPI (for backend)
- Next.js (or Frontend)
- GPT 4- API
- Google Cloud

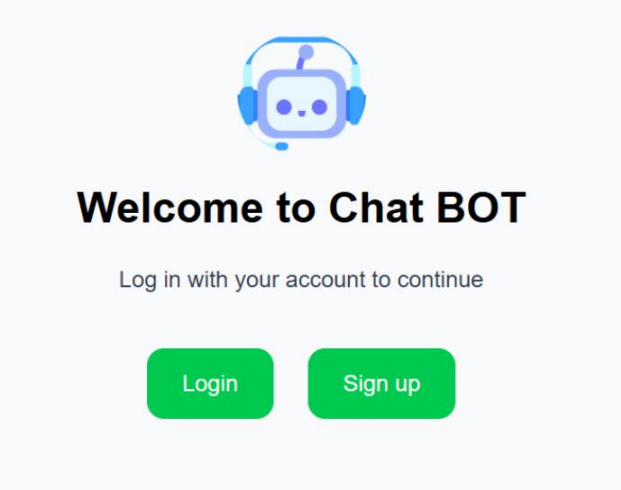

Figure 1(a)

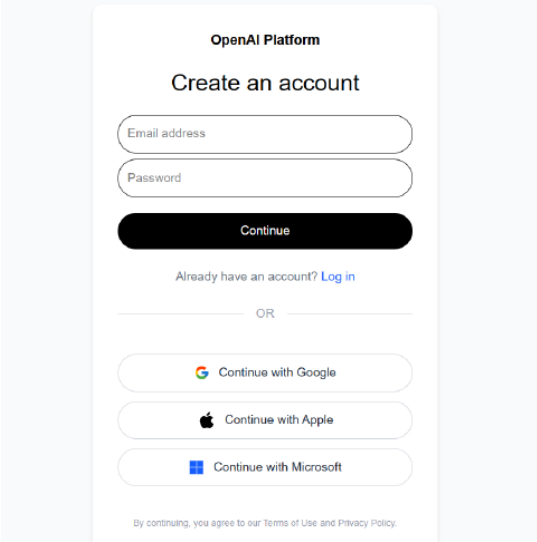

Figure 1(b)

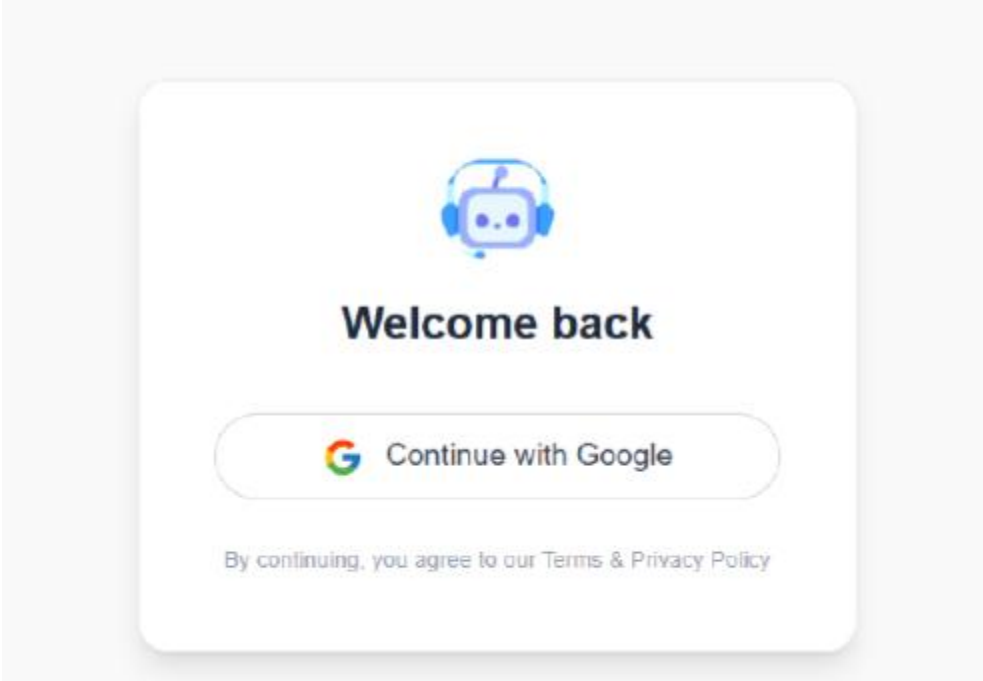

Figure 1(c)

Figure 1(d)

Figure1(a) – 1(d): Application Interface (Home, Register, Login, Chat Pages)

.

4 Initial Environment Configuration

Create a new virtual environment and install all dependencies:

```
pip install -r requirements.txt
```

Run preprocessing and observe class distribution:

```
print(df['label'].value_counts())
```

Semantic validation was done through both attack and benign word clouds visualisation. The instructional and exploitation driven terms were marked in the attack word cloud and the generic-purpose and informational terms were present in the benign word cloud.

Figure 2: Word Cloud - Attack and Word Cloud – Benign

The visual checks ensured that the classifier made the right distinction of the contextual tone of prompts.

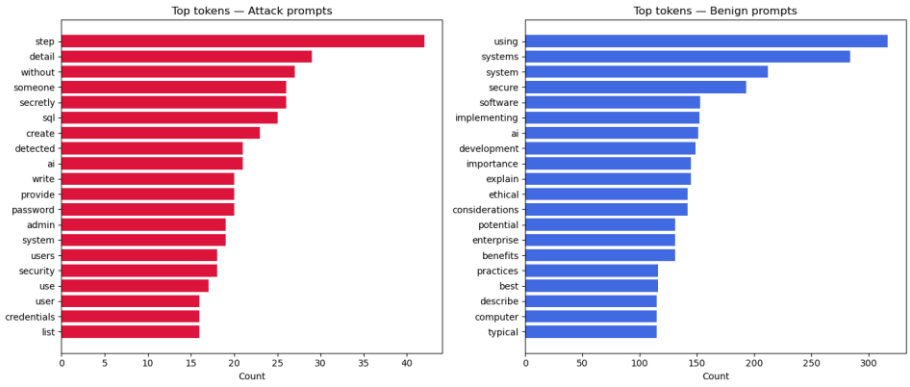

Figure 3: Top Tokens Attack Prompts & Benign Prompts

This section explains how the dataset is transformed using TF-IDF, followed by training:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,2), max_features=20000)
X = vectorizer.fit_transform(df['prompt'])
```

Top-token analysis was used to validate token-level behaviour by seeing which terms the system returned when the system was in malicious mode, like secretly, step, sql, and password. This analysis aided in ensuring that the classifier was not memorising particular text, but learning useful linguistics features.

5 Security Testing & Validation

Security tests and verification were implemented to maintain that the created prompt injection detection system functions in a reliable manner, its precision is heavily maintained, and the adversarial prompts are not passed to the AI model. The validation step entailed both quantitative assessment, scenario testing, and behavioural testing with respect to familiar prompt attack plans. The main aim was to test the level of effectiveness of the classifier in separating benign and adversarial inputs and stay useful to legitimate users.

Per-class evaluation measures also indicated that there was perfect precision of attack prompts, as well as complete recall of benign prompts, which conveys that there was strong separation across classes.

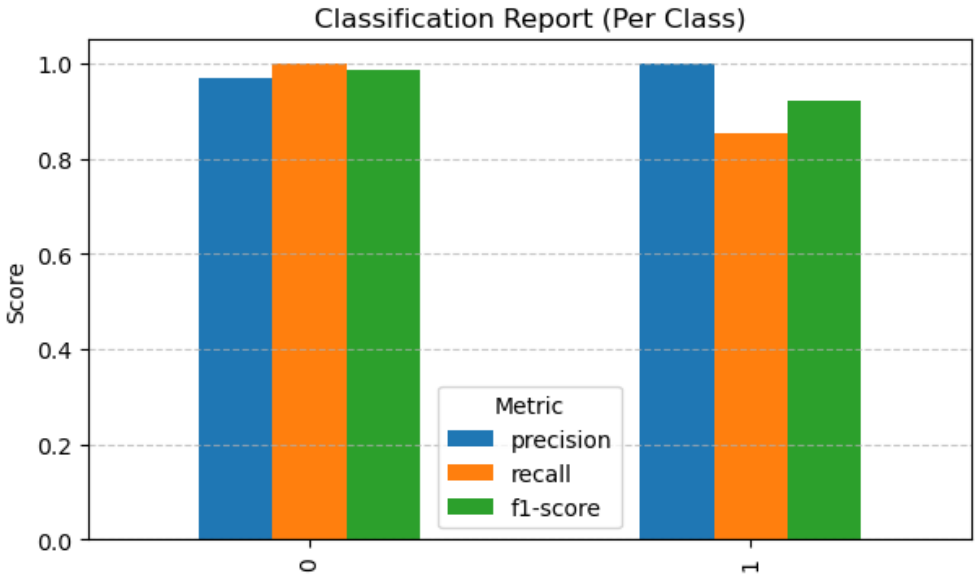

Figure 4: Classification Report (Per Class)

The initial validation phase was based on the quantitative analysis of the dataset that was cleaned. TF-IDF features and calibrated Logistic Regression model were found to test the classifier. Important metrics like precision, recall, F1-score and ROC-AUC were calculated to determine the quality of detection.

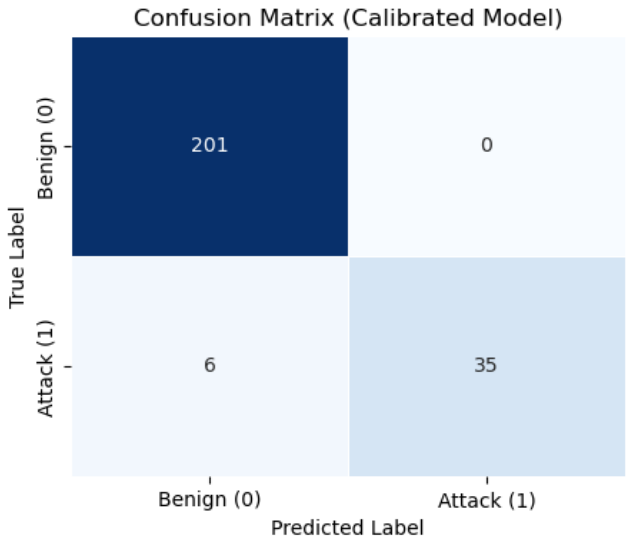

Figure 5: Confusion Matrix for the Zero-Shot Safety Classifier (Calibrated)

The confusion matrix was utilized to determine the patterns of true positives, false positives, and true negatives. The model had a high overall accuracy which means that the model was reliable in detection.

Lastly, testing of end to end security was done via the cloud based Chat interface. The prompts were experimented in the three scenarios, namely, benign labour-law questions, mixed prompts with the risky form, and direct attack attempts. The system was responsive, and the correct prompts were acceptable, grey-area prompts were rewritten, and harmful prompts were rejected, which proved the viability of the entire pipeline.

6 Deployment Guide

After validation, deploy the application with the launching of the backend service:

```
python app.py
```

Upload the project to a cloud platform, configure environment variables, and ensure HTTPS and API key restrictions. The model endpoint is linked with the frontend pages, where the unsafe prompts are detected in real time. Lastly, ensure that sanitized responses are visible in the Chat page and then make the system publicly available.

References

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

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

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].

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.

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