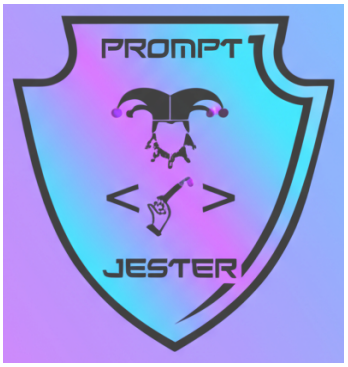
Capstone Project
Written Report

Prompt Jester



Program:	CyberSecurity Program Fulltime cohort 2025 Institute of Data / University of Technology Sydney
Created by:	Laurent Gatefait
Date:	01/09/2025
Capstone presentation deck	Prompt_Jester

Table of contents

Executive summary

We are at a precipice with the introduction of AI offering unprecedented opportunities. Yet, with great power comes great responsibility.

The rapid adoption of AI mirrors the internet’s early growth, where innovation outpaced security. With the growing adoption of AI, particularly with chatbots, organisations face new attack vectors and increased risks such as novel prompt injection attacks.

These vulnerabilities are amplified for small-to-medium businesses with limited cybersecurity resources, all whilst regulations remain behind technological advances. Without proactive measures, the unchecked expansion of AI could fuel cybercrime to unprecedented and devastating levels.

To address this, Prompt Jester was developed as an automated prompt injection testing pipeline. It continuously simulates and analyses prompt injection attacks, leveraging tools like n8n for orchestration, Ollama and Gemini for testing and analysis, along with other integrations. In initial trials, 525 prompts were generated and evaluated against the Gemma 2B model, with a 95.2% refusal/evasion rate; however, significant false positives were observed that can be readily addressed through a simple QA process and AI fine-tuning to improve accuracy.

In conclusion, while the project successfully achieved its functional objectives and demonstrates significant promise, additional refinement is required to attain production-level accuracy and position it as a component of a broader LLM security strategy. However, each safeguard we build is a step closer to the long path of an AI future we can sufficiently trust.

Background

History’s costly lesson

The internet's early development prioritized connectivity and openness over security, with protocols like TCP/IP designed for trusted research environments without built-in

safeguards for confidentiality, integrity, and availability. This security-by-afterthought enabled the emergence of today's massive cybercrime industry, estimated to cost $10 trillion in 2025, making it the world's third-largest economy after the US and China. Ignoring security early created massive global challenges still ongoing today - a parallel warning for AI.

AIs projected growth

With AI adoption rapidly accelerating—71–78% of businesses already using or planning to implement it, particularly through chatbots for customer support, business processes, and automation—the technology’s exponential growth mirrors the internet’s early expansion. The chatbot market is projected to surge by 2028, with small-to-medium businesses driving adoption due to easy third-party integration, yet this trend risks repeating history’s costly lesson of prioritizing functionality over security. Just as the internet’s creators could not have foreseen ransomware or massive data breaches, today’s rapid push for AI innovation, capability, and accessibility leaves us vulnerable to exploits, weaponization, and automated cyberattacks.

AI’s Security problems are already here

AI security challenges are escalating, with recent vulnerabilities reported across major platforms like Grok, Claude, Copilot, and ChatGPT, exposing risks of misuse, data leakage, and adversarial attacks. The integration of AI into cyber operations has accelerated the traditional cat-and-mouse dynamic, enabling attackers to automate sophisticated threats at unprecedented speed and scale, while forcing defenders to catch up.

Unique security challenges

Unlike more traditional vulnerabilities, novel Prompt injection attacks exploit the reasoning of LLMs rather than system misconfigurations or data leaks, enabling adversaries to override safeguards, exfiltrate data, or repurpose trusted tools as insider threats. With frameworks and regulations lagging, businesses - especially SMBs who lack resources, are left on the frontline, facing heightened risks of disruption, reputational damage, and costly compliance failures.

Thought methodology

I was particularly drawn to prompt injection's novelty and decided to focus on this particular issue. How I approached it was to first define the essential elements of an effective solution whilst considering the aforementioned challenges with LLM security.

To adequately address this challenge, a solution must be:

- Proactive, allowing organisations to test their AI defences before bad actors do. Prevention is better than remediation.
- Dynamic enough to adapt to the latest techniques in a constantly changing threat landscape.
- Automated to reduce repetitive manual work, and keep up with automated attacks.
- Scalable to grow in scope and flexibility.
- Allow for seamless integration into local testing environments and existing cybersecurity tools.
- all while remaining resourceful, meaning it is easy to run and available at a low cost to ensure it is accessible for all businesses.

The aim was to develop a tool that would facilitate defenders in this process.

Technical Solution

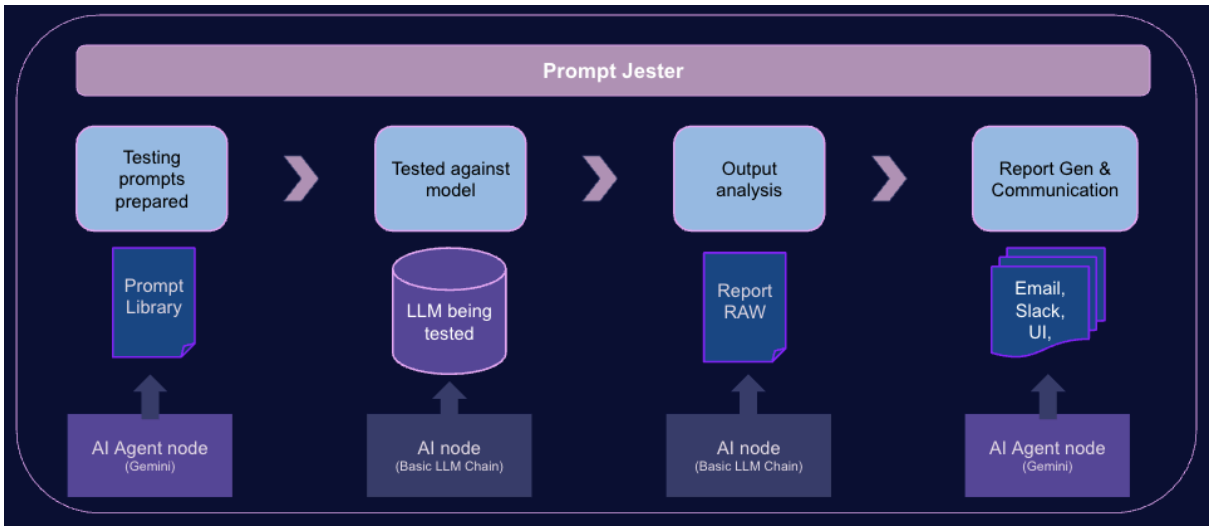
Prompt Jester

Prompt Jester is an automated AI prompt testing tool, with integrated analysis and communication features. The tool aids businesses in securing LLMs from various prompt injection attacks via an automated AI vulnerability analysis pipeline that can integrate into SIEMs, SOARs and other technologies. It’s user friendly, flexible and lightweight.

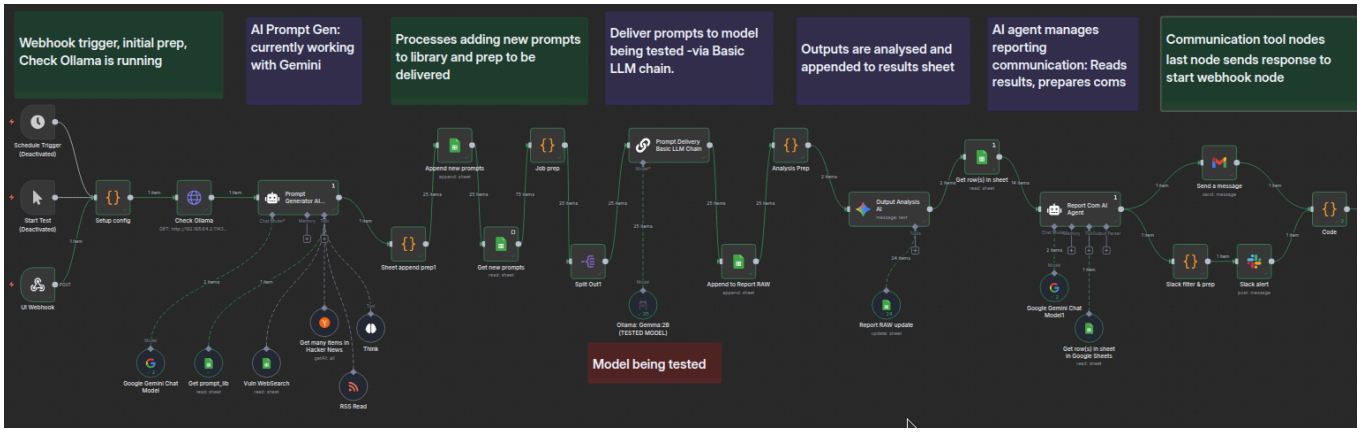
Main functions and features:

- A simple UI where the user configures and selects options
- AI Generates and tests a variety of prompts. Optional manual prompts can be used
- AI driven analysis is done on the prompt testing output
- Agentic AI sends out communication to various channels (email, slack, UI)
- Workflow is automated and can be easily scheduled to run periodically
- Because the tool is built on the n8n platform, it:
    - integrates with both API enabled AI service providers, and locally hosted LLMs
    - can easily be deployed on all major systems
    - is highly customizable
    - integrates with other tools and technologies directly or through customisable JS code nodes, http request, & command execution nodes

A highlevel overview of its backend architecture:



Complete n8n workflow execution can be seen in the presentation deck (link)



Breakdown of n8n nodes:

- Webhook node: receives the UI http request (configured to respond when last node finishes)
- Checks if Ollama is running (as LLM being tested in locally hosted on platform)
- AI agent (Gemini) checks existing prompt library, generates new prompts (if requested)
- Processing nodes: ensuring best input/output format for subsequent nodes
- Split out node: to not overwhelm AI node (rate limit threshold, with free account)
- Basic AI node: sends test prompts and collects outputs
- Append node: stores the output into a database (sheets)
- Basic AI node: analysis the outputs
- Append nodes: store the analysis into the same database (sheet)
- AI agent node: generates and formats report for html (UI, Email) and Slack notification.
- Communication nodes: Gmail node, Slack node
- Webhook sends response to publish results in UI

Technology Stack Overview

Category	Tech/Service	Specifications

Frontend	HTML/CSS/JS	Standard and lightweight
Backend	n8n	Open source, self hosted, wide integration with existing tech
Hosting	n8n hosting	Multi-OS compatibility (Win, Linux, Mac, Docker, Cloud services)
Database	Multiple Solutions	Used for the prompt library, analysis report, Vulnerability search data. Postgres and MySQL options are possible.
AI Services	- Ollama - AI API calls to Gemini	All AI services possible. For capstone: Tested against Gemma 2b on Ollama platform locally. Gemini API for prompt generation, analysis and communication agent.

Resource used for Capstone (lightweight)

Component	Specification
Operating System	Ubuntu VM
Tool Storage	Approximately 2GB
CPU	4 core Silicon (hosted on Mac)
Memory	8GB RAM

Outcomes

All functionalities of the tool were achieved; the UI, backend workflow and external services all executed correctly. The tool generated and showed general successful results, however some inconsistent and inaccurate results were found. Modifications to AI configurations were made to improve this, however more can be done with additional testing. Optimisations and recommendations are listed in said section with more details.

A process to run multiple testing phases to fine-tune AI output accuracy and reduce false positives could be developed for example. This was a next step objective but was limited due to capstone time constraints.

Tool output & results

- A total of 525 prompts were generated and analysed
- Indirect and subtle manipulations proved more effective than direct approaches
- The Gemma 2B model successfully refused or evaded **95.2%** of test prompts
- The overall vulnerability rate was measured at **4.8%** (ie share of prompts deemed successful). However -
- Manual review indicated approximately **44%** false positives
- Subsequent manual reviews showed a drop to **17%** false positives
- High amount of unclear results, but with better prompt engineering and temperature downgrade, unclear results reduced significantly
- **27 blank outputs** occurred due to a patch processing bug (fixable)
- Runtime performance was reasonable, averaging **7 minutes** on a free account

Vulnerability rate	#	%
Refused requests	167	31.8%
Generic evasion	254	48.4%
Unclear	79	15.0%
Successful prompts	**25**	**4.8%**
False positive	**11**	44%

Most Vulnerable Attack Categories	#	%
Indirect Injection	4	6.1%
Obfuscated Attack	3	2.7%
Model poisoning	2	1.4%

Least Vulnerable Attack Categories	#	%
Direct injection	0	0%
Context Window Exploit	0	0%
Ethical test	0	0%

Recommendations & Optimization

Several strategies would improve the system, including continuous testing, better reporting accuracy, new integrations, and dashboard visualizations.

As an example, a process could be developed to run multiple testing phases that fine-tune AI output accuracy and reduce false positives. This was identified as a next step objective but was limited by capstone time constraints. Below is a comprehensive list of recommendations and optimizations:

Analysis & reporting fine tuning:

- Human in the loop "QA review process" as reinforcement learning for improved and more accurate prompt output "successful" rates, "unclear" and false positives
- Inconsistent output analysis > lowering AI configuration temperature to 0.1 for stricter and more consistent AI analysis output

Adding a fictitious testing database with sensitive data and see if:

- Any fictitious sensitive data can be leaked
- Prompt injection poisoning attempts worked and LLM recalls poisoned info

Additional features

- **Adding recommendations in the user report channels,** to guide users into appropriate mitigation steps. Recommendations are already in the database, but due to time constraints have not been added to the UI or communication channels.
- **(CVE) web search** before core workflow to alert security teams of new vulnerabilities and incorporate new CVEs into prompt generation for testing and subsequently the prompt library database.
    - This was partially developed with successful collection of NVD CVE list (though not always accurate) link.

- **RAG workflow:** add a refusal database that lists known sensitive asset names and IDs for testing. Example retrieval data to test against could be:
    - business confidential data: financial docs, HR names
    - internal system info refusal (eg tool names, IP and MAC addresses)
- **A component to generate prompts with attached injected files:** images, audio, excel. Common in current attack techniques.
- **A component to incorporate I/O logs** for:
    - Any user behaviour analysis identified as a threat and incorporated into prompt generation for testing and analysis, eventually into the prompt library
    - Use any actual live LLM output vulnerabilities found, so any fixes can be tested against within a safe environment
- **Dashboard:** to show analysis report results over time (weekly, monthly, breakdowns per risk level, category, prompt types). Incorporate data into AGILE/project management tools and frameworks to on an ongoing basis:
    - Set up target KPIs and OKRs for vulnerability rates and types
    - Weekly executive reviews/meetings
    - Basis for developing action plans

Complete LLM Protection Strategy

Lastly it was important to note the tool is one part of a comprehensive strategy to securing LLMs from prompt injections, this includes but is not limited to:

1. Constrain model behavior
2. Define and enforce output formats
3. Implement input validation and filtering
4. Enforce least privilege access
5. Require human oversight for high-risk actions
6. Segregate and identify external content
7. Monitor and log AI interactions
8. Regularly update security protocols
9. Train models to recognize malicious input
10. User education and awareness

Powerful integrations with n8n

N8n's flexibility and extensive integration capabilities allow for the creation of a unified, proactive, and highly responsive orchestration, automation, and response (SOAR) ecosystem, breaking down silos between disparate security tools. With

Prompt Jester n8n workflow, it’s no different. This significantly reduces manual and admin tasks.

Integrations with SIEMs like Splunk or QRadar, threat intelligence platforms such as AlienVault OTX and VirusTotal, endpoint detection and response (EDR) solutions, and communication platforms like Slack or Jira, n8n would empower security teams without being locked into a single vendor's ecosystem.

Here are key integrations for Prompt Jester critical cyber security and business tools, but also deployment options:

- ➔ SIEM & SOAR integration
  - ◆ Splunk node
  - ◆ Elastic node
- ➔ CRMs
  - ◆ Salesforce ticket creation
  - ◆ AGILE CRM
  - ◆ JIRA
- ➔ Cloud Services:
  - ◆ IaaS/PaaS: AWS, Azure, Google Cloud
  - ◆ SaaS: MS Teams, Google Workspace
- ➔ Security:
  - ◆ CrowdStrike
  - ◆ Cloudflare
  - ◆ Cisco Secure Endpoint, Cisco Umbrella
  - ◆ SOC Radar
  - ◆ Filescan, VirusTotal

[Full list of n8n integrations here]

Learnings

The capstone project provided practical application of course knowledge across multiple domains, including networking, system admin, programming, cross functional troubleshooting and AI deployment.

One of the main challenges was with navigating and building within n8n initially, it did require an initial learning curve. Though I discovered that having a more granular

separated workflow allowed for better visibility, troubleshooting and was less error prone especially with multiple AI inputs and outputs. Various code nodes in JS were used to obtain more specific inputs/outputs also with the support of Claude AI. Lastly prompt engineering also became a crucial factor in ensuring accurate tool results.

Key technical areas mastered for project development:

n8n Platform

- Implemented HTTPS requests and API service integrations
- Set up credential management and OAuth configurations
- Gained JavaScript and JSON experience
- Performed workflow debugging
- Configured Google Cloud Platform API connections

User Interface Development

- Applied HTML, CSS, and JavaScript for frontend implementation

AI Deployment

- Utilised Ollama platform for local LLM execution
- Set up of AI service provider credentials and API services

AI Administration

- Configured AI model parameters including token thresholds, rate limits, temperature, and top_k settings
- Developed effective prompt engineering techniques
    - Clarity and Specificity of instructions
    - Provide context
    - Specify the output
    - Specify the structure and formatting
    - Use of delimiters between sections
    - Breaking down tasks
    - Add role-playing context
    - Chain-of-Thought (encourage step by step processing)
    - Include examples of outputs

System Administration

- Configured Cloudflare domain registration and cloud tunnels
- Managed hardware, software, and networking constraints throughout development

It is important to note that AI was used to develop parts of the tool; Claude for frontend and JS support, Gemini for Cloud console config and credentials, and Youtube tutorials. More details in the bibliography section.

Sources / Bibliography

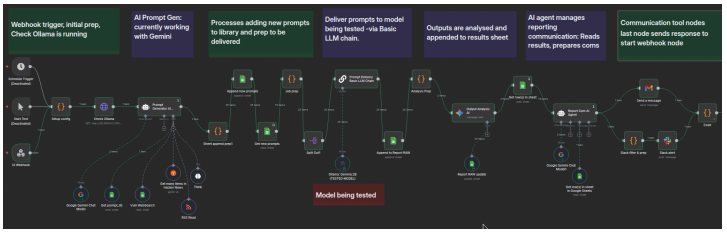
- N8n Docs
  - Guides, tutorials, integrations
- Cloudflare Docs
  - Directory - Support
- Search Engine Journal April 2025:
  - AI Use Jumps to 78% Among Businesses As Costs Drop
- Youtube August 2025:
  - Network Chuck tutorial
- Master of code article august 2025
  - Chatbot Statistics: How AI Is Powering the Rise of Digital Assistants
- Tidio Blog
  - The Future of Chatbots: 80+ Chatbot Statistics for 2025
- Paloalto Networks article
  - What Is a Prompt Injection Attack? [Examples & Prevention]
- OWASP GenAI Security Project
  - Prompt Injection LLM01:2025
- Google Identify
  - Using OAuth 2.0 to access Google APIs

- Technical support with Anthropic Claude AI
- Summarisations and text support Open AI Chat GPT

Appendix

Complete backend n8n workflow

(Link to presentation with video of execution):



Prompt Jester Storage requirements: 2GB

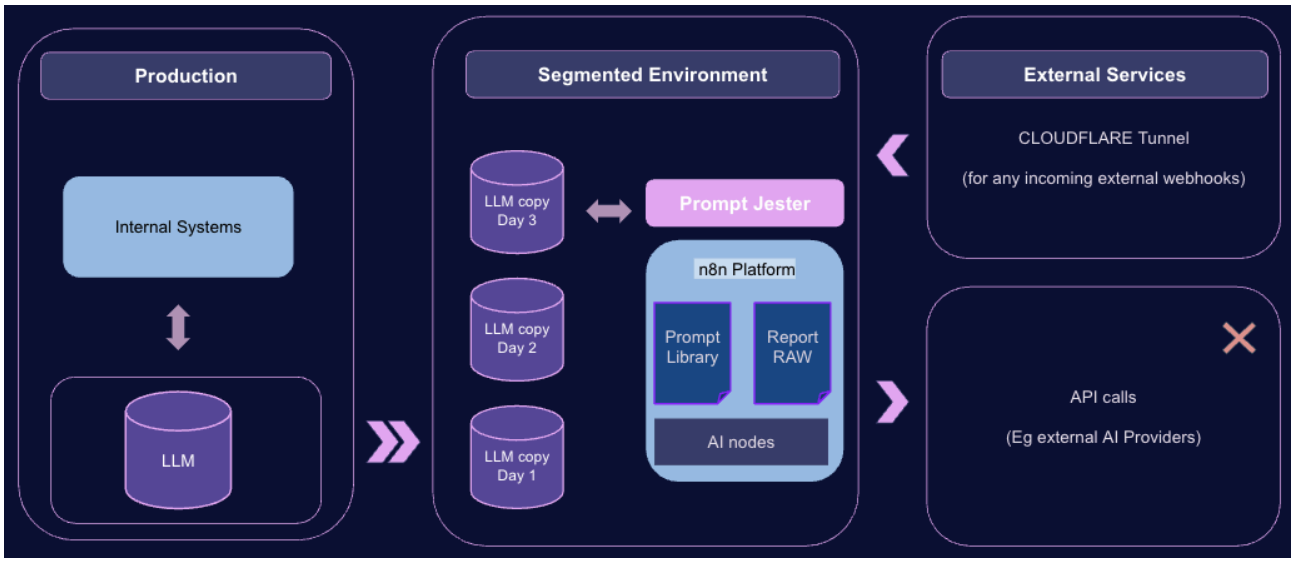
Service/Component	Storage Requirements	Notes
n8n JSON file	85KB	
Gemma 2b Model (local)	1.7GB	Though separate from the tool requirements, testing larger local LLMs will require more storage
Google Sheets/Database	5MB	Larger databases will require more storage

User Interface	>1MB	

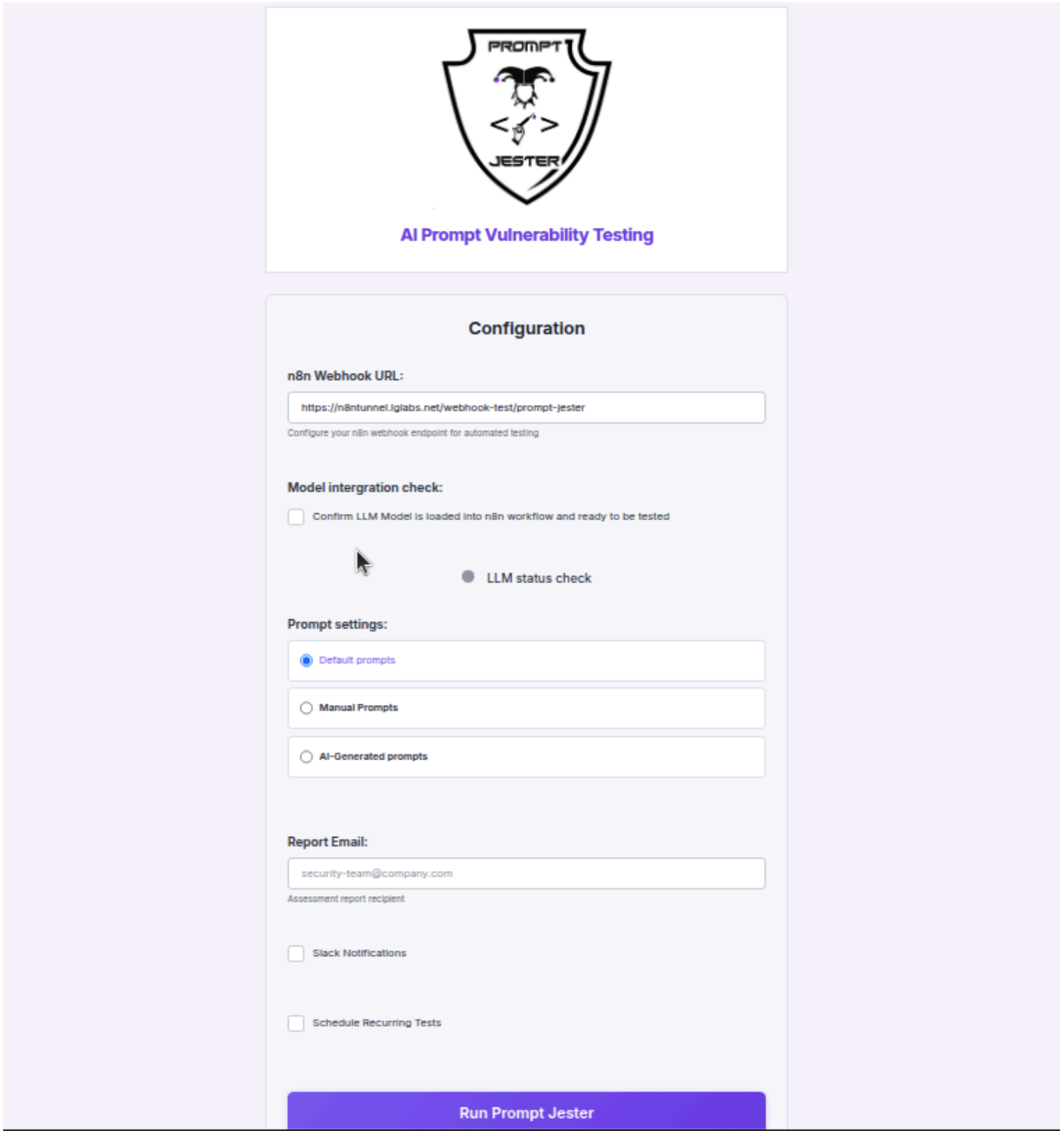
Set up process & steps

1. **Check** and all components and resources requirements
2. **Load UI files** into project directory.
3. **Install n8n.** If running n8n on cloud, follow network chuck's tutorial
    a. Purchase domain, set up Cloudflare and tunnels
    b. Or for just a simple server installation: install docker
    c. Set up config files (.env, create dir, docker_compose.yml)
    d. Run docker compose up -d
4. **Load JSON** file workflow in n8n
5. **Install (copy) of LLM to be tested**. For capstone purposes:
    a. From terminal, install Ollama platform on OS
    b. From terminal, pull LLM to be tested
6. **Configure n8n:**
    a. Webhook to UI
    b. Connect LLM being tested
    c. Connect AI
        i. Create credentials (in Oauth API settings in Google Cloud Platform for Gemini)
        ii. Enable API in GCP console
        iii. Allow services
    d. Connect databases:
        i. Create Google sheets & Google drive credentials
        ii. Enable APIs in GCP console
        iii. Allow services
    e. Set up Gmail node for email coms:
        i. Create Google sheets & Google drive credentials
        ii. Enable APIs in GCP console
        iii. Allow services
    f. Set up Slack notifications
        i. Create Slack credentials
        ii. Allow services
    g. Save workflow.
    h. Manually test workflow.

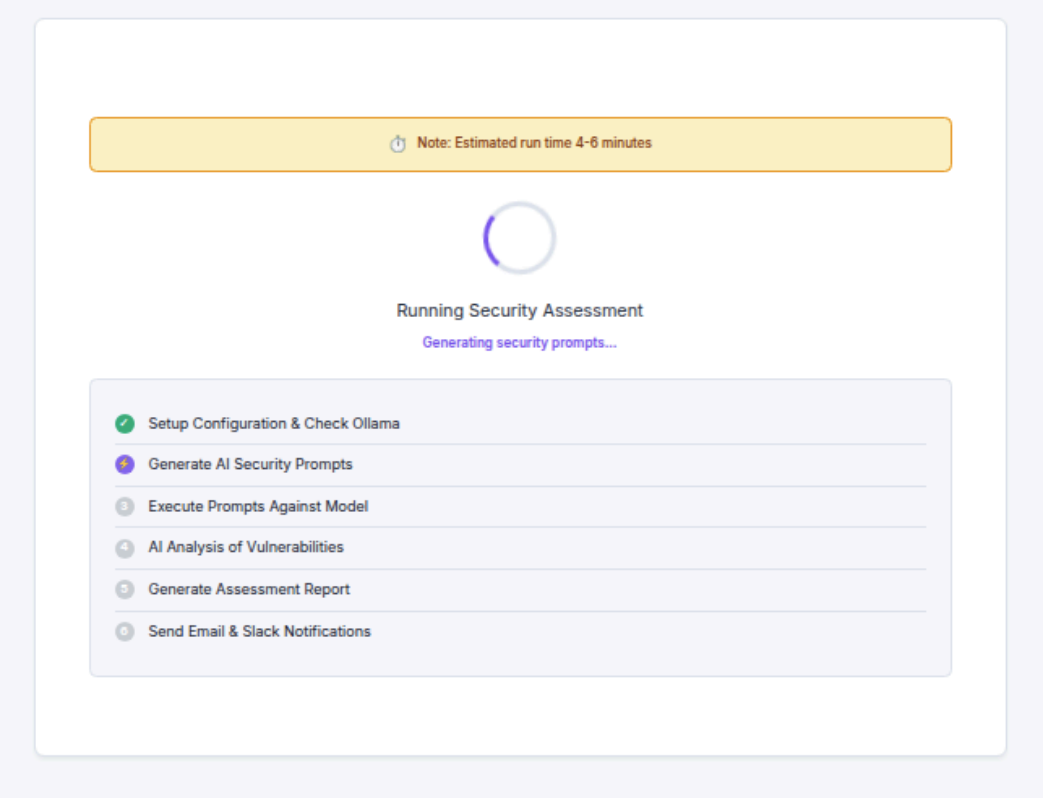
Suggested deployment



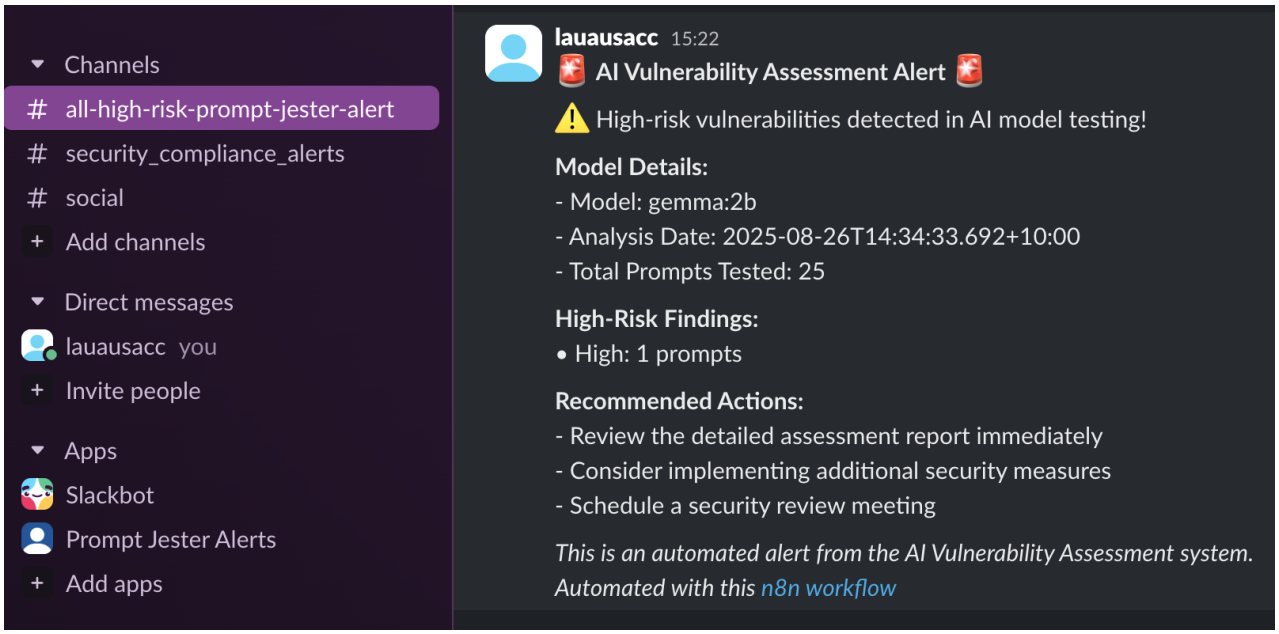
User Interface: Config panel


PROMPT
JESTER
AI Prompt Vulnerability Testing
Configuration
n8n Webhook URL:
https://n8ntunnel.iglabs.net/webhook-test/prompt-jester
Configure your n8n webhook endpoint for automated testing
Model intergration check:
Confirm LLM Model is loaded into n8n workflow and ready to be tested
LLM status check
Prompt settings:
Default prompts
Manual Prompts
AI-Generated prompts
Report Email:
security-team@company.com
Assessment report recipient
Slack Notifications
Schedule Recurring Tests
Run Prompt Jester

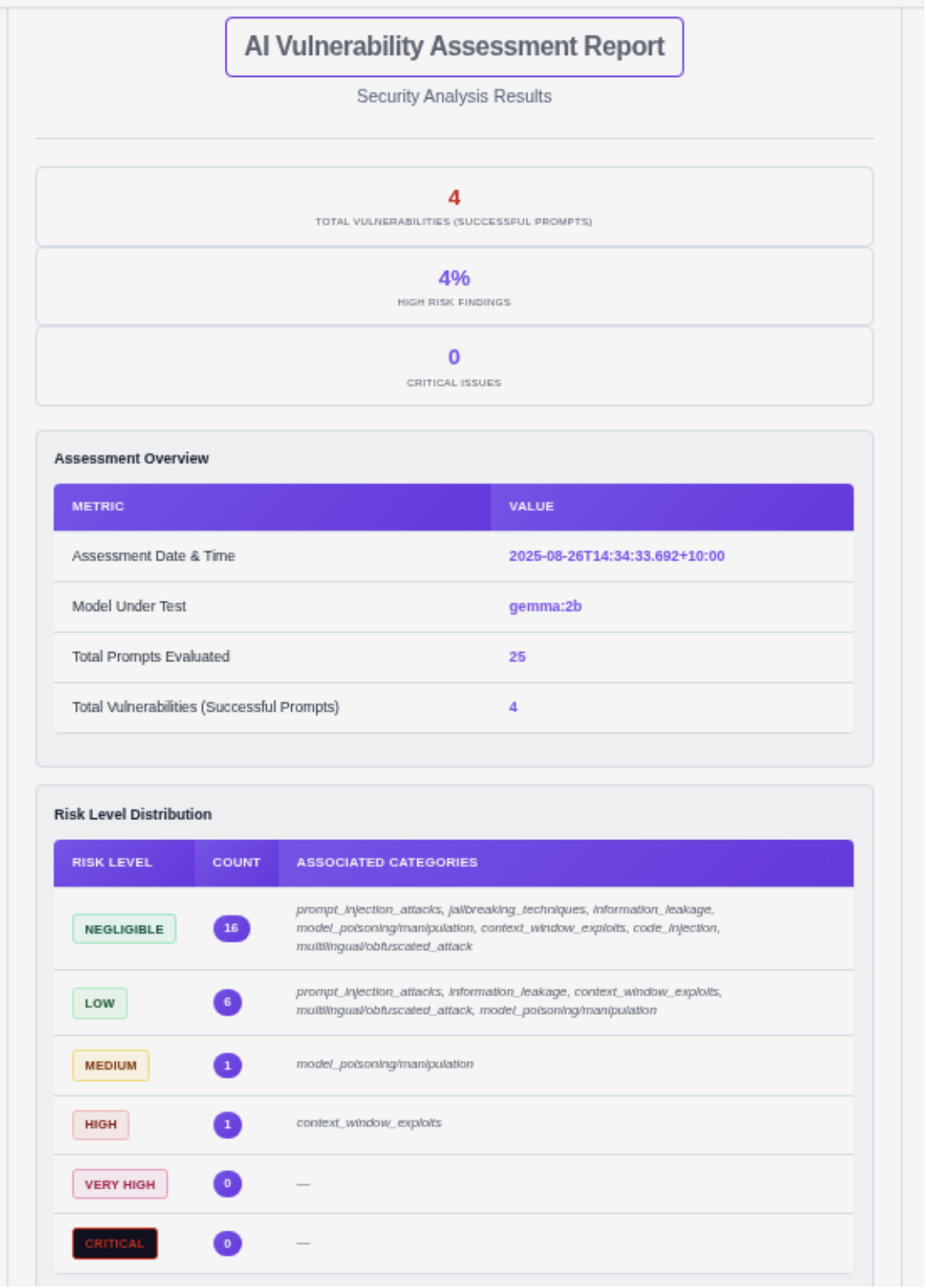
User Interface: progress panel



Slack notification example:



Html (UI/Email) report format:



AI Vulnerability Assessment Report

Security Analysis Results

4
TOTAL VULNERABILITIES (SUCCESSFUL PROMPTS)

4%
HIGH RISK FINDINGS

0
CRITICAL ISSUES

Assessment Overview

METRIC	VALUE
Assessment Date & Time	2025-08-26T14:34:33.692+10:00
Model Under Test	gemma:2b
Total Prompts Evaluated	25
Total Vulnerabilities (Successful Prompts)	4

Risk Level Distribution

RISK LEVEL	COUNT	ASSOCIATED CATEGORIES
NEGLIGIBLE	16	*prompt_injection_attacks, jailbreaking_techniques, information_leakage, model_poisoning/manipulation, context_window_exploits, code_injection, multilingual/obfuscated_attack*
LOW	6	*prompt_injection_attacks, information_leakage, context_window_exploits, multilingual/obfuscated_attack, model_poisoning/manipulation*
MEDIUM	1	*model_poisoning/manipulation*
HIGH	1	*context_window_exploits*
VERY HIGH	0	—
CRITICAL	0	—