AI Powered Vulnerability Risk Mapping

MUHAMMAD TALHA
01-136221-055

HUZAIB KHAN
01-136221-036

Supervisor: Dr. Sohail Akhtar

Bachelor of Science in Artificial Intelligence

Department of Computer Science
Bahria University, Islamabad

Dec, 2025

Certificate

We accept the work contained in the report titled "AI Powered Vulnerability Risk Mapping", written by Mr. Muhammad Talha and Mr. Huzaib Khan as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Artificial Intelligence.

Approved by:

Supervisor: Dr. Sohail Akhtar (Assistant Professor)

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr. Sohail Akhtar (Assistant Professor)

_____

Head of the Department: Dr. Muhammad Khurram Ehsan (Sr. Associate Professor)

_____

December 2025

Abstract

Organizations are finding it more and more challenging to keep up with the assessment and prioritizing of threats due to the increasing number of cybersecurity vulnerabilities. Traditional techniques of mapping Common Vulnerabilities and Exposures (CVEs) to the MITRE ATT&CK framework are laborious, slow, and error-prone, requiring deep expertise and significant spending time on it. This study solves the challenge by proposing an AI-powered solution that automates the CVE–MITRE mapping process utilizing natural language processing and large language models like GPT. This system analyzes unstructured CVE descriptions and intelligently maps them to applicable ATT&CK tactics and procedures, decreasing the strain on security analysts, improving response time, and enhancing the accuracy of threat modeling. The automation intends to assist more effective threat detection, risk assessment, and proactive defensive methods across modern digital.

Acknowledgments

We would like to express our deepest gratitude to our respected supervisor, **Dr. Sohail Akhtar**, for his invaluable guidance, encouragement, and continuous support throughout the development of this project. From the very beginning, his expert advice and insightful feedback have been instrumental in shaping the direction of our research and ensuring its successful completion. His patience, motivation, and immense knowledge inspired us to overcome challenges and refine our ideas into a meaningful outcome.

We are sincerely thankful for his dedication, timely feedback, and constant encouragement, which greatly contributed to our academic growth and understanding of this field. This project would not have reached its present form without his expert mentorship and support.

Muhammad Talha and Huzaib Khan
Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
CSV	Comma-Separated Values
CVE	Common Vulnerabilities and Exposures
GPU	Graphics Processing Unit
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
MVC	Model-View-Controller
NLP	Natural Language Processing
NVD	National Vulnerability Database
OAuth2	Open Authorization 2.0
PDF	Portable Document Format
SSO	Single Sign-On
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Background

Common Vulnerabilities and Exposures (CVEs) are publicly disclosed flaws in software and hardware that can be exploited by malicious actors, posing significant risks to organizations and individuals [1, 2]. Managed by MITRE and documented in the National Vulnerability Database (NVD), CVEs provide a standardized system for identifying and tracking security weaknesses. Organizations rely on timely CVE information to patch vulnerabilities and mitigate potential attacks. Failure to address these vulnerabilities can result in severe consequences that including data breaches and ransomware incidents and and system failures. [3, 4].

The MITRE ATT&CK framework provides a structured knowledge base of adversary tactics and techniques that offering a common language for understanding attacker behavior. Linking CVEs to ATT&CK techniques enables security teams to anticipate attack paths and strengthen defense mechanisms [5, 6]. Manual mapping of vulnerabilities to ATT&CK, this is resource-intensive and prone to inconsistencies and necessitating automated solutions for more effective threat intelligence.

1.2 Project Objective

This project aims to develop an AI-powered system that automatically maps CVEs to the MITRE ATT&CK framework. Using natural language processing (NLP) techniques and large language models (LLMs) such as GPT this system analyzes CVE descriptions and predicts corresponding attack tactics and techniques [7, 8]. The key objectives include:

- **Automated CVE Classification & Threat Correlation:** Accurately associate CVEs with ATT&CK techniques to enable proactive security strategies.
- **Enhanced Decision Support:** Provide actionable insights to help security teams prioritize vulnerabilities based on attacker behavior patterns.

1.3 Problem Statement

Current vulnerability management processes struggle to relate CVEs with real-world attack techniques efficiently. Rule-based systems and while foundational are limited by their inability to generalize to new or unseen threats [1, 9]. This gap hampers accurate risk assessment and delays incident response. Despite the growing adoption of frameworks like MITRE ATT&CK, organizations often lack automated tools to bridge CVE data with actionable threat intelligence and highlighting the need for intelligent AI-driven mapping systems [10, 11].

1.4 Methodology

The project leverages a Natural Language Processing (NLP) pipeline based on transformer models and particularly GPT that automatically map Common Vulnerabilities and Exposures (CVEs) to MITRE ATT&CK techniques. This method combines semantic understanding of CVE descriptions with contextual reasoning from large language models to improve mapping accuracy and reduce human effort. [5, 7, 6, 8].

The methodology consists of the following components:

- **Data Collection:** CVE identifiers and their textual descriptions are gathered from the National Vulnerability Database (NVD) and MITRE resources [1, 2]. The dataset is enriched with metadata such as CVE severity scores and published dates and CWE categorizations it allowing the model to better contextualize each vulnerability [9, 10].
- **Data Preprocessing:** Raw CVE descriptions are cleaned and normalized. Steps include lowercasing, removal of stopwords, HTML tags, and irrelevant metadata, as well as tokenization for model input. More techniques such as stemming and lemmatization and removal of redundant information help improve embedding quality [5, 11].
- **Feature Extraction:** Contextual embeddings are generated using transformer-based models. Unlike traditional TF-IDF or bag-of-words representations these embeddings capture semantic and syntactic relationships within the CVE text enabling better alignment with ATT&CK tactics and techniques [6, 12].
- **Model Inference:** Pre-trained LLMs such as GPT-3.5 turbo or domain-specific BERT models are used in a zero-shot or few-shot manner to predict ATT&CK techniques associated with each CVE [7, 13]. Carefully engineered prompts guide the models while optional fine-tuning with a small labeled dataset further enhances performance and context awareness [14].

- **Evaluation:** Predictions are compared against expert-validated mappings or existing benchmarks. Metrics such as precision recall and F1-score, and coverage are computed to assess model performance. Manual review by cybersecurity professionals ensures the predicted techniques are accurate and actionable [8, 12].
- **User Interface:** A web-based interface allows analysts to input a CVE ID or description and retrieve predicted ATT&CK mappings in real-time. The interface also provides explanation snippets and links to relevant CVE or ATT&CK documentation, facilitating faster triage and decision-making [3, 1].

That methodology integrates multiple state-of-the-art techniques from recent research. Example semantic similarity methods using Siamese networks [5] and reasoning capabilities of LLMs [7] enable both precise and context-aware CVE-to-ATT&CK mapping. By combining preprocessing, feature extraction, inference, and evaluation, the system offers a scalable solution that significantly reduces manual effort in threat analysis while providing actionable intelligence for security operations.

1.5 Development Cycle

The development process follows iterative stages to ensure robust and accurate mapping:

- **Data Collection:** Aggregate CVEs from NVD and MITRE.
- **Preprocessing:** Clean, normalize, and tokenize CVE descriptions.
- **Model Training & Fine-Tuning:** Use transformer-based models for supervised and prompt-based inference.
- **Inference:** Deploy trained models for mapping new CVEs.
- **Evaluation:** Assess performance using standard metrics and expert review.
- **Feedback Loop:** Refine model predictions with updated data and feedback.

1.5.1 Development Cycle

1.6 Project Scope

- Focus on Enterprise ATT&CK techniques.
- Automate mapping of CVEs to ATT&CK techniques.
- Provide justifications for mappings.
- Evaluate against benchmark datasets.

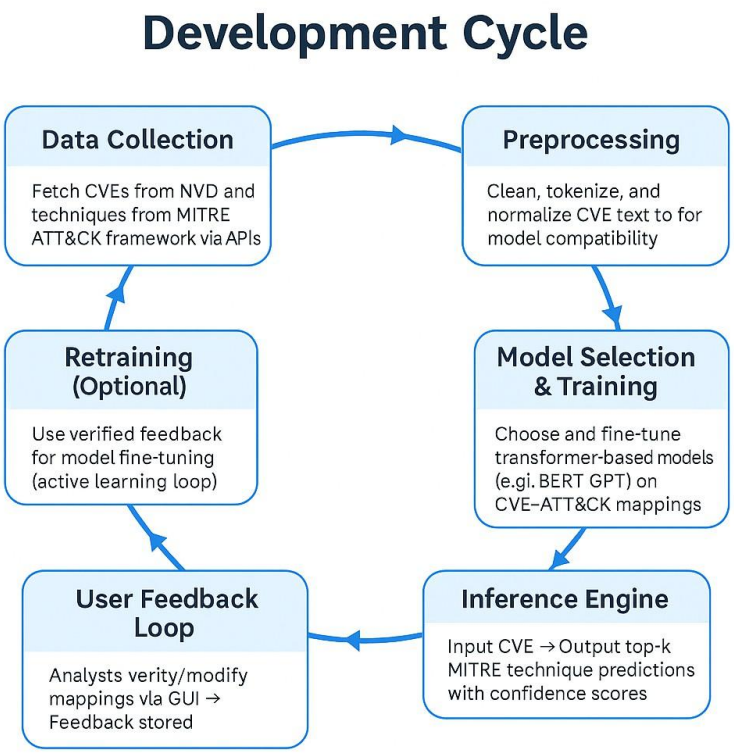


Figure 1.1: Development Cycle of the CVE–MITRE Mapping System

- Generate comprehensive reports for analysts.

Out of Scope

- Industrial Control Systems (ICS) vulnerabilities.
- Mobile-specific attack vectors.
- Cloud-exclusive attack scenarios.

1.7 Feasibility and Resources

The project leverages established transformer models (e.g., GPT) and is feasible with moderate computational resources. Required tools include Python PyTorch and Hugging Face Transformers and secure databases like Supabase. Integration with SOC workflows and security APIs ensures practical usability [15, 14].

1.8 Use Cases

1.8.1 SOC Teams: Efficient Threat Triaging

Automated CVE-to-ATT&CK mapping reduces manual effort, enabling faster incident response.

1.8.2 Vulnerability Analysts: Contextual Prioritization

Enriched mapping helps analysts prioritize critical vulnerabilities based on attacker techniques.

1.8.3 Blue Teams: Detection Rule Improvement

Linking CVEs to ATT&CK techniques accelerates creation of SIEM rules and other detection logic.

1.9 Tools and Technologies

- **Languages:** Python, JavaScript
- **Libraries:** PyTorch, Hugging Face Transformers
- **Datasets:** NVD, MITRE ATT&CK
- **APIs:** OpenAI GPT-3.5 turbo, MITRE TAXII

Summary

The chapter introduces the challenge of managing CVEs and emphasizes the importance of mapping vulnerabilities to ATT&CK techniques. It proposes an AI-driven system using NLP and LLMs to automate CVE-to-ATT&CK mapping, reducing analyst workload and improving threat detection. The methodology covers data collection, preprocessing, model inference, evaluation, and user interface development. The chapter also defines project scope, feasibility, risks, and real-world use cases, providing a foundation for the automated CVE-MITRE mapping system.

Chapter 2

Literature Review

The rising complexity of cybersecurity threats has highlighted the need for intelligent, automated methods to map CVEs to the MITRE ATT&CK framework. While traditional rule-based approaches are manual and inconsistent, recent AI and NLP advancements offer scalable, accurate, and automated solutions.

2.1 Recent Studies

2.1.1 SMET: Semantic Mapping of CVE to ATT&CK

SMET introduces a Siamese BERT network (dubbed ATT&CK BERT) specifically fine-tuned to compute semantic similarity between CVE descriptions and ATT&CK techniques. The model uses contrastive loss during training to learn embedding spaces where related pairs (CVE, technique) are closer together. This architecture captures fine-grained relationships in textual data and surpasses traditional machine learning and keyword-based methods. It marks a shift towards semantically intelligent mapping solutions. [5]

2.1.2 Crimson: Enhancing Strategic Reasoning in Cybersecurity

Crimson leverages Large Language Models (LLMs), such as GPT-based architectures, to perform strategic-level CVE-to-ATT&CK mapping. Unlike typical classifiers, Crimson integrates a human-in-the-loop process combined with synthetic data generation to continuously improve LLM reasoning. The system enables deeper context understanding and provides mappings that align more closely with threat analyst expectations. Its ability to generalize to novel CVEs and anticipate related tactics sets it apart. [7]

2.1.3 Automatic Mapping Based on CVE and ATT&CK

This work proposes a Transformer-based encoder model trained on structured and annotated CVE datasets. It leverages self-attention mechanisms to model context within descriptions

and aligns them to appropriate MITRE techniques. The method highlights the effectiveness of transformers for extracting relevant threat intelligence from unstructured CVE data. The system exhibits high accuracy and interpretability, proving useful in real-time security operations. [8]

2.1.4 CVE2ATT&CK: BERT-Based Mapping of CVEs to MITRE ATT&CK Techniques

The CVE2ATT&CK project introduces a multi-label classification model that fine-tunes BERT to predict multiple ATT&CK techniques per CVE. The authors manually annotate 1,813 CVEs and experiment with different architectural variants and data augmentation strategies. The dataset and results serve as a benchmark for evaluating future models. This work is pivotal in framing ATT&CK mapping as a multi-label NLP problem. [6]

2.1.5 Rule-Based Matching of CVEs to ATT&CK Techniques (Traditional Approach)

MITRE's early efforts involved heuristic rules and expert-validated keyword mappings to link CVE descriptions with ATT&CK techniques. These traditional rule-based systems formed the groundwork for labeled datasets and offered a basic, interpretable pipeline. However, they lacked scalability and generalization to unseen threats. Their value lies more in prototyping and benchmarking than in production-level deployment.

2.1.6 Comparative Analysis of Traditional and Modern Approaches

Table 2.1: Comparison of Traditional and Modern CVE to MITRE ATT&CK Mapping Approaches

Authors	Methodology	Dataset	Key Findings	Results
Basel Abdeen et al. (2023)	Siamese BERT with contrastive learning	Expert- annotated CVE-ATT&CK pairs	Semantic similarity improves precision of mappings	Outperformed state-of-the-art baselines in F1 and precision; highly robust embeddings
Jiandong Jin et al. (2024)	LLMs with human-in-the-loop and synthetic data generation (Crimson)	GPT-trained on synthetic and real CVEs	LLMs can reason contextually and strategically	Comparable to GPT- 4 performance; strong generalization on unseen CVEs
Lei Wang et al. (2024)	Transformer encoder model with self-attention	Public CVE dataset with ATT&CK labels	Transformer attention mechanisms learn contextual clues effectively	High accuracy and interpretability; good performance on benchmark dataset
Octavian Gr ig- orescu et al. (2022)	BERT-based multi-label classification with augmentation	1,813 manually labeled CVEs	Multi-label formulation enables richer mapping; augmentations improve recall	F1-score: 47.84%; best results with fine-tuned BERT and augmentation
MITRE (2019)	Rule-based keyword and heuristic matching	Internally tagged CVEs via manual analysis	Foundational for dataset bootstrapping; not scalable	Limited precision; lacks adaptability to new or ambiguous CVEs

2.2 Conclusion

The reviewed literature shows a clear shift from rigid, rule-based CVE–ATT&CK mapping to advanced AI-driven techniques. While traditional methods are simple and interpretable that lack scalability and adaptability. Modern Transformer and LLM-based approaches provide superior semantic understanding and higher accuracy. Systems like SMET and Crimson push automation forward but still depend on human oversight.Hybrid models combining rules and AI.

Chapter 3

Requirement Specifications

3.1 Existing System

In the current cybersecurity system mapping CVE (Common Vulnerabilities and Exposures) entries to MITRE ATT&CK techniques is a manual process handled by security analysts. These mappings require expertise and significant time in which makes the process slow and error-prone, and difficult to scale as the number of vulnerabilities grows rapidly.More inconsistency in mapping standards across organizations leads to discrepancies in threat intelligence.

3.2 Proposed System

The proposed system leverages transformer-based AI models (such as GPT) to automate the mapping of CVE descriptions to MITRE ATT&CK techniques. It allows users to input CVE IDs or descriptions and receive the most probable techniques in response. It also provides a web-based interface for user interaction, supports bulk processing, enables expert feedback for continuous learning and maintains query history for auditability and improvement.

3.3 Requirement Specifications

Functional Requirements

- **CVE Input:** The system shall accept a CVE ID or full description as input.
- **Technique Prediction:** The system shall return a list of relevant MITRE ATT&CK techniques for each CVE.
- **Bulk Upload:** The system shall allow users to upload multiple CVEs in a file (e.g., CSV) for batch processing.

- **Feedback Submission:** The system shall allow domain experts to provide feedback on the accuracy of mappings.
- **History Management:** The system shall log and display historical mapping queries for each user/admin.
- **Export Results:** The system shall support exporting of mapping results in various formats (e.g., JSON, CSV).
- **User Authentication:** The system shall distinguish between user and admin roles through secure login.

Non-Functional Requirements

- **Performance:** The system should respond to single CVE mapping requests in under 5 seconds.
- **Scalability:** The system must be capable of handling hundreds of concurrent users and large batch jobs.
- **Usability:** The user interface must be intuitive and accessible, requiring minimal training.
- **Reliability:** The system must ensure accurate results and handle failures gracefully.
- **Security:** All user data must be protected using secure protocols (e.g., HTTPS, encrypted storage).
- **Maintainability:** The system should be modular to allow future updates or model improvements.

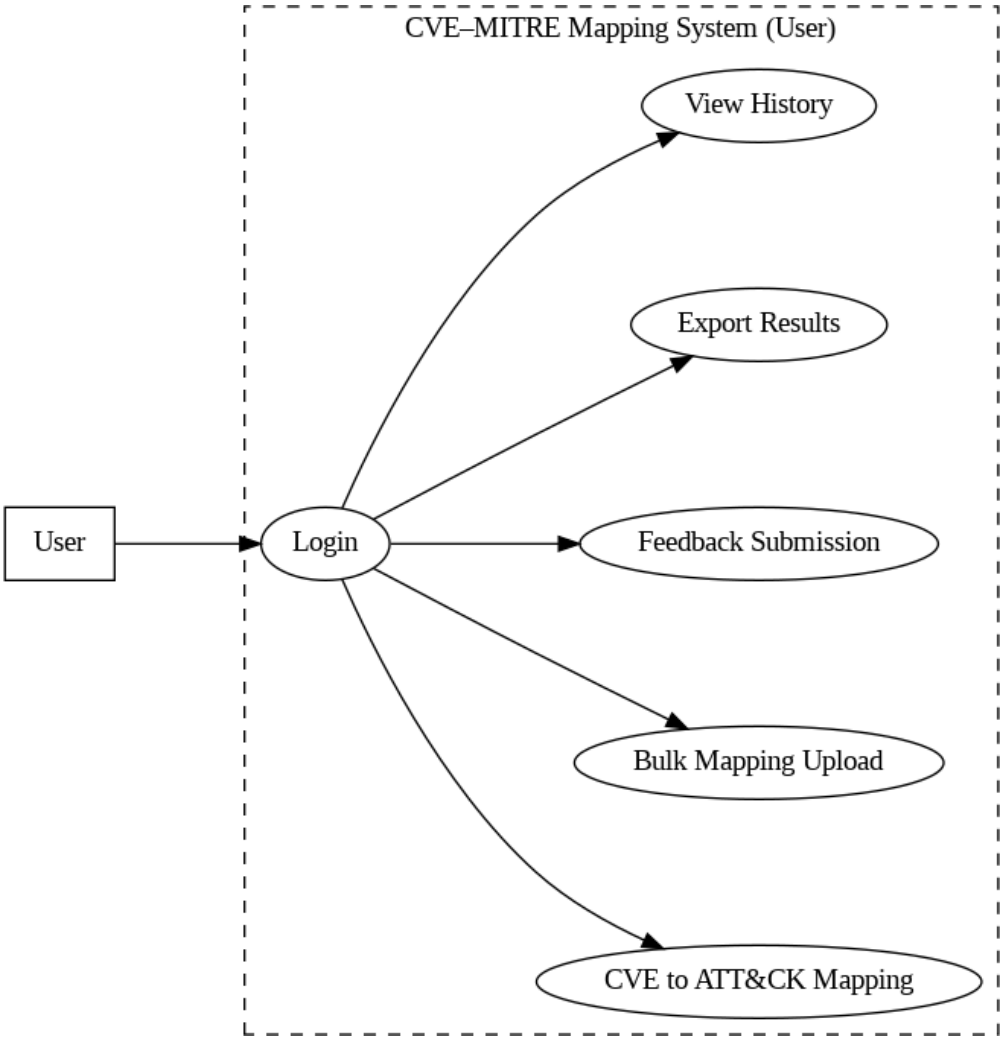


Figure 3.1: User Use Case Diagram

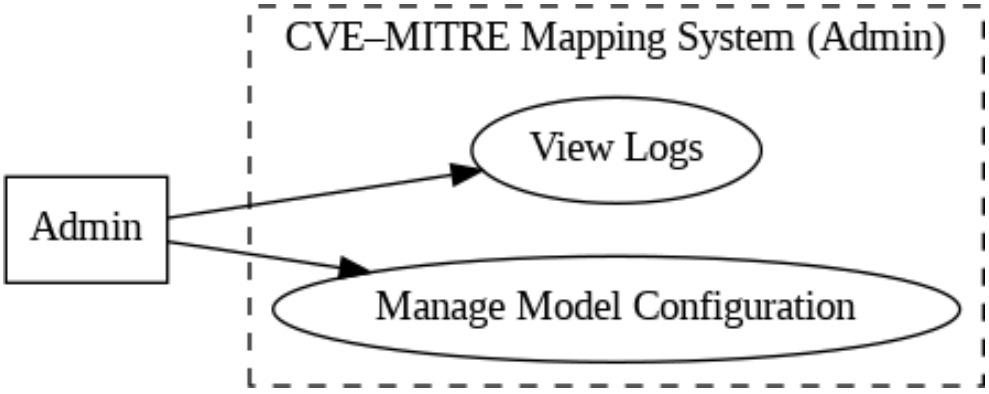


Figure 3.2: Admin Use Case Diagram

3.4 System Use Cases

System use cases define the interactions between actors (users or administrators) and the CVE–MITRE Mapping System. These use cases describe the functional behavior of the system from the user's perspective, outlining the steps involved in achieving specific goals. Each use case highlights the sequence of events, actors involved, and system responses under normal and exceptional conditions.

3.4.1 User Use Cases

Use Case U1: CVE to ATT&CK Mapping

Use Case ID	U1
Actor	Security Analyst
Description	The user inputs a CVE ID or description and receives relevant MITRE ATT&CK techniques.
Precondition	User must be authenticated.
Postcondition	Suggested techniques are shown to the user.
Normal Flow	The user logs into the system and enters a CVE ID or a vulnerability description. The system then processes the input using the AI model and displays the most relevant MITRE ATT&CK techniques.
Exception Flow	If the CVE is missing or invalid the system is unable to process it and the AI model may fail to generate any output.The user is shown an error message along with possible fallback suggestions to correct the input or try alternative queries.

Table 3.1: Use Case U1: CVE to ATT&CK Mapping

Use Case ID	U2
Actor	Security Analyst
Description	The user uploads a CSV file containing multiple CVEs and receives batch mapping results.
Precondition	Valid file format (CSV) must be uploaded.
Postcondition	Mapping results are returned and available for export.
Normal Flow	The user logs into the system and uploads a properly formatted CSV file containing multiple CVE entries. The system processes each record and then presents the mapped results to the user who can either view them directly or download the output.
Exception Flow	If the uploaded file has an incorrect format or encoding, the system may encounter parsing or processing errors and fail to read all records.The user may receive partial results or no results at all and is prompted to correct the file before re-upl

Table 3.2: Use Case U2: Bulk Mapping Upload

Use Case ID	U3
Actor	Domain Expert
Description	The user provides feedback on the accuracy of AI-generated mappings.
Precondition	Mapping results must be available.
Postcondition	Feedback is stored for future model refinement.
Normal Flow	The user reviews the mapping produced by the system and provides their input through the feedback form.The system stores this feedback in the database for future improvements and analysis.
Exception Flow	If the feedback form fails to submit or contains invalid input, the system cannot record the user's response.If the user is not authorized to submit feedback the system restricts access and prevents the entry from being saved.

Table 3.3: Use Case U3: Feedback Submission

Use Case ID	U4
Actor	Security Analyst
Description	The user exports CVE-to-technique mapping results in JSON or CSV formats.
Precondition	Mapping must be completed.
Postcondition	File is downloaded by the user.
Normal Flow	The user selects the desired export format, and the system generates the file and initiates its download.
Exception Flow	If the selected file format is not supported or an error occurs during export the system is unable to generate the file and notifies the user accordingly.

Table 3.4: Use Case U4: Export Results

Use Case ID	U5
Actor	Security Analyst
Description	The user views the history of CVE queries and their corresponding mapped techniques.
Precondition	User must be logged in.
Postcondition	A list of past queries is displayed.
Normal Flow	The user navigates to the history page, where the system displays a list of all previously processed CVE mappings for review.
Exception Flow	If history data is unavailable or an error occurs while loading it the system notifies the user and the previous CVE mappings cannot be displayed.

Table 3.5: Use Case U5: View History

3.5 Admin Use Cases

The Admin is responsible for overseeing system operations, managing model configurations, and ensuring secure and efficient functioning of the CVE–MITRE Mapping System. The following use cases describe key administrative functionalities.

Use Case ID	A1
Actor	System Administrator
Description	The admin views system logs, including mapping activity history, model inference records, and user interactions for auditing and monitoring.
Precondition	Admin must be authenticated.
Postcondition	System logs are displayed for review.

Table 3.6: Use Case A1: View System Logs

Use Case ID	A2
Actor	System Administrator
Description	The admin manages model configurations, including updating parameters, uploading new versions, or switching between different deployed models.
Precondition	Admin is authenticated and the system is in a safe state for configuration updates.
Postcondition	Updated model settings are applied for future inference requests.

Table 3.7: Use Case A2: Manage Model Configuration

Summary

This section outlines the methodology for developing the CVE–MITRE Mapping System. It covers data collection from the NVD and MITRE ATT&CK datasets, preprocessing to clean and structure the information, and model selection using transformer-based architectures such as GPT. The system is trained and evaluated with appropriate performance metrics and user feedback is incorporated iteratively to refine the model and improve accuracy.

Chapter 4

Design

System design defines the architecture, components, modules, interfaces, and data necessary to fulfill the system's functional and non-functional requirements. This chapter presents the architectural and design decisions made for the CVE–MITRE Mapping System, an AI-driven tool designed to automatically map Common Vulnerabilities and Exposures (CVEs) to MITRE ATT&CK techniques.

4.1 System Architecture

The system follows a modular and scalable architecture, combining traditional software engineering practices with cutting-edge AI technologies. It is composed of several major modules, each with a distinct responsibility:

- **Frontend Module:** A web-based UI allowing users (security analysts and administrators) to interact with the system. Users can submit CVEs, view results, provide feedback.
- **AI Processing Engine:** The core logic module responsible for running transformer-based models (such as BERT or GPT). It analyzes CVE descriptions and produces a list of probable MITRE ATT&CK techniques.
- **Backend API and Services:** Facilitates communication between frontend and backend modules. It handles user management and data routing feedback recording and system configuration.
- **Database:** A relational database used to persistently store CVE records and mappings and user feedback and logs and user details.

- **External API Integrator:** Regularly fetches fresh CVEs from NVD and synchronizes with the MITRE ATT&CK framework via external APIs.

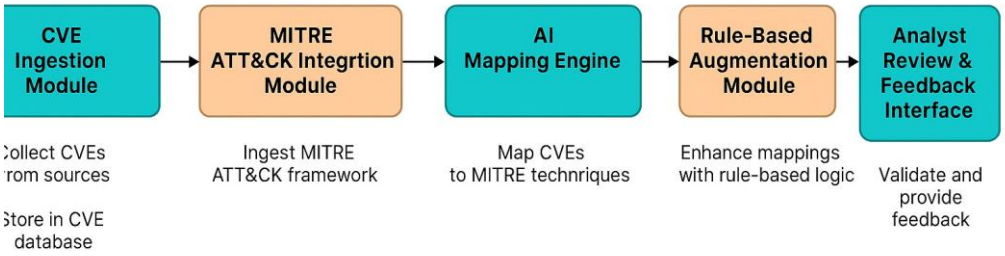


Figure 4.1: System Architecture: Interaction between frontend, AI engine, backend, and external APIs

4.2 Design Constraints

Several design constraints were identified during system planning:

- **Model Performance vs. Latency:** The system must produce accurate results within acceptable response times.
- **Data Availability:** Continuous access to NVD and MITRE ATT&CK data is critical.
- **Security:** CVE data may be sensitive. Secure communication protocols (e.g., HTTPS, OAuth2) must be enforced.
- **Interpretability:** AI predictions must be explainable to improve trust and usability.

4.3 Design Methodology

The design of the CVE–MITRE Mapping System employs a combination of object-oriented principles and AI pipeline engineering:

- **Modular Design:** System modules are loosely coupled and highly cohesive.
- **Object-Oriented Approach:** Backend services are implemented as classes and objects with clear interfaces.
- **MVC Pattern for Frontend:** Ensures separation of concerns between business logic (controllers), user interface (views), and data models.
- **AI Pipeline Modeling:** Text pre-processing inference and post-processing are developed as sequential pipeline stages using libraries like HuggingFace Transformers or PyTorch.

4.4 High Level Design

The system is viewed from five architectural perspectives:

1. **Conceptual View:** Users enter CVE descriptions which are processed by the AI engine. The system returns ATT&CK mappings with confidence scores.
2. **Process View:** The standard workflow:
    - User logs in
    - Inputs CVEs
    - System processes the text and runs inference
    - Results are visualized and saved
3. **Physical View:** The application is deployed in a cloud environment (e.g., AWS EC2), with model inference running on GPU-backed instances.
4. **Module View:** The system is divided into API layer, service layer, model engine, and data layer.
5. **Security View:** OAuth2-based authentication, role-based access control, activity logging, and input sanitization are enforced.

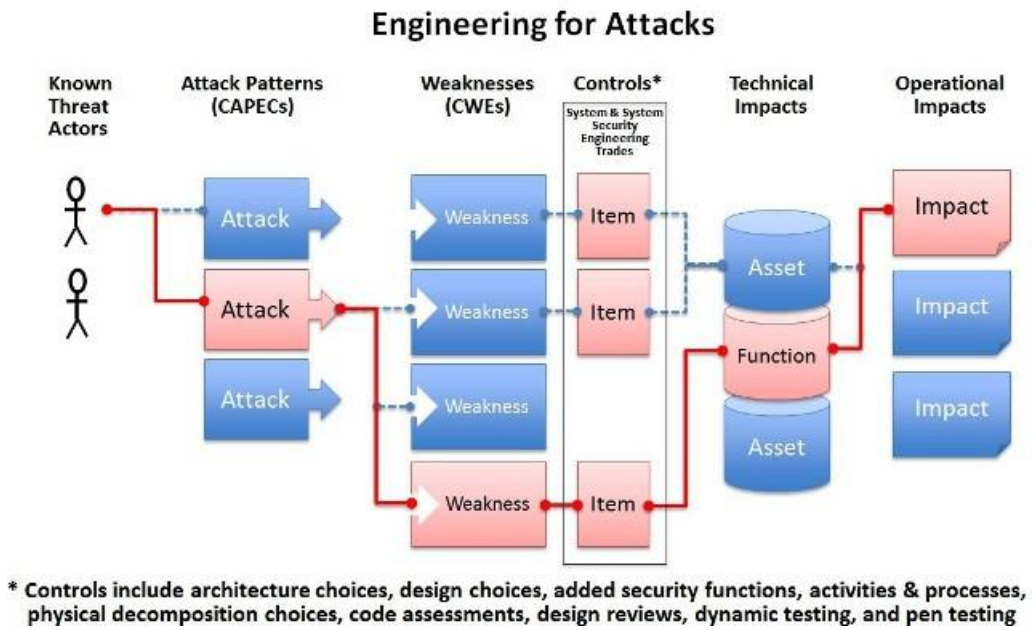


Figure 4.2: High-Level Architectural View of the CVE–MITRE Mapping System

The following modules/classes are responsible for specific internal logic:

- `CVEInputHandler`: Responsible for cleaning, validating, and tokenizing input text for processing.
- `ModelInferencer`: Loads the AI model, executes inference on processed CVEs, and returns top-ranked MITRE techniques.
- `MappingRepository`: Manages the storage and retrieval of inference results in the database.
- `FeedbackManager`: Collects user responses to mappings and stores them for evaluation or retraining.
- `UserManager`: Manages user authentication, session handling, and access permissions.

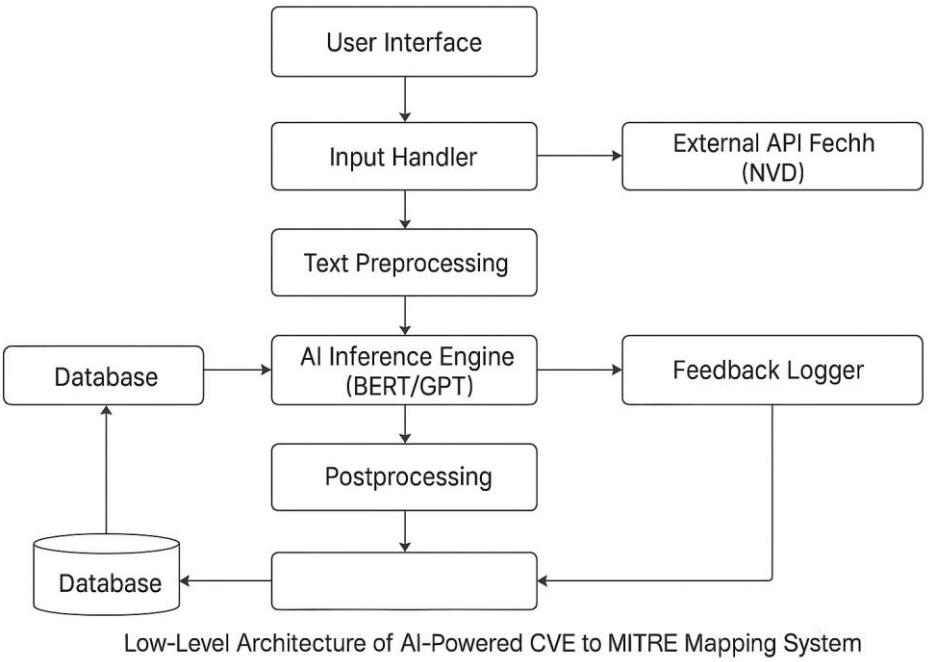


Figure 4.3: Low-Level Component Interaction Diagram

4.5 Database Design

The relational database supports structured data storage and fast querying. Major tables and relationships include:

- **Users:** Stores user ID, email, hashed password, role, and login activity.
- **CVEs:** Stores CVE ID, description, date, and data source.
- **Mappings:** Stores AI-generated MITRE mappings, including associated CVE ID.
- **Feedback:** Captures user feedback on specific mappings.
- **Logs:** Tracks user activity and system events for auditing.

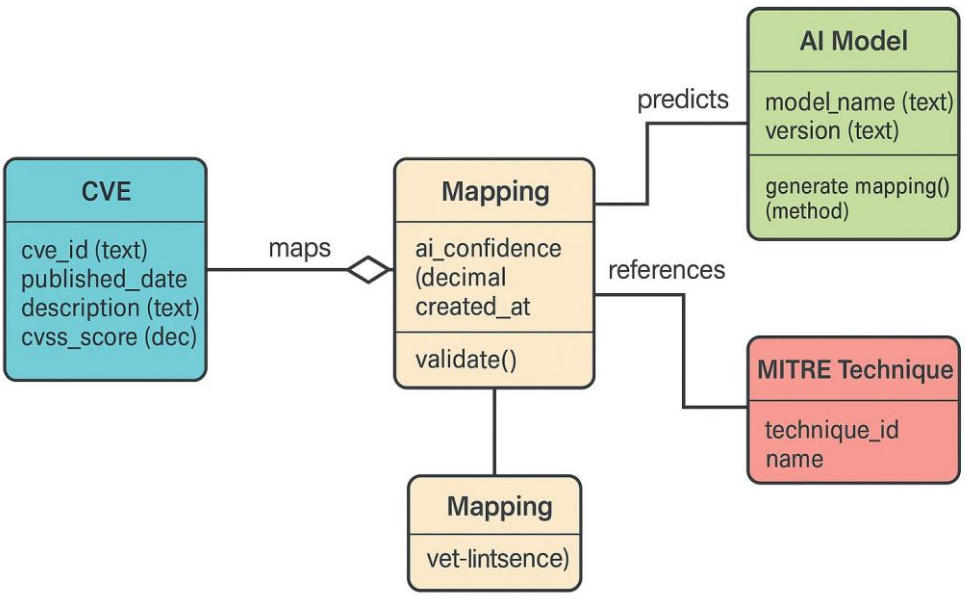


Figure 4.4: Database Schema Diagram

4.6 GUI Design

The user interface is built for clarity and functionality, providing:

- **CVE Submission:** Form-based and CSV upload support for entering CVE descriptions.
- **Mapping Visualization:** Clear display of inferred ATT&CK techniques with confidence bars or scores.
- **Feedback Interface:** Buttons for accepting, rejecting, or modifying mappings.
- **Admin Dashboard:** View system metrics, user logs, and update model configurations.

4.6.1 Homepage

The homepage serves as the primary access point for users, offering a dashboard-style interface. It provides navigation to the main system modules such as CVE submission, mapping visualization, and report generation. The design prioritizes usability and accessibility, ensuring that users can quickly initiate core actions.

Figure 4.5: Homepage of the CVE–MITRE Mapping System showing the main dashboard interface

4.6.2 About Page

The About page provides an overview of the system's purpose, explaining how AI and NLP techniques are leveraged to map CVEs to the MITRE ATT&CK framework. It also introduces the motivation behind the system and its expected impact in automating vulnerability analysis.

Figure 4.6: About Page displaying project background and objectives

4.6.3 How It Works Page

This section guides users through the step-by-step process of how the mapping system operates — from CVE ingestion and preprocessing to AI inference and ATT&CK mapping. It helps non-technical users understand the underlying workflow in a simplified visual format.

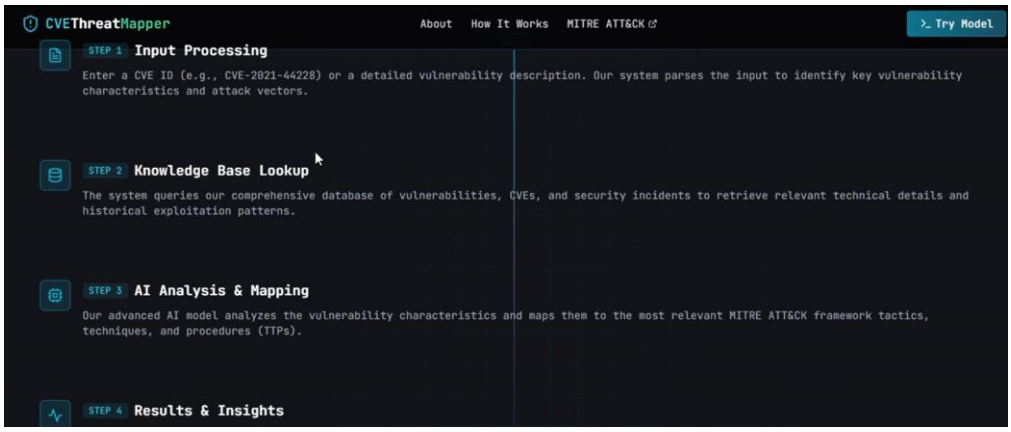


Figure 4.7: How It Works Page illustrating the CVE-to-MITRE mapping workflow

4.6.4 TTPs Generation Page

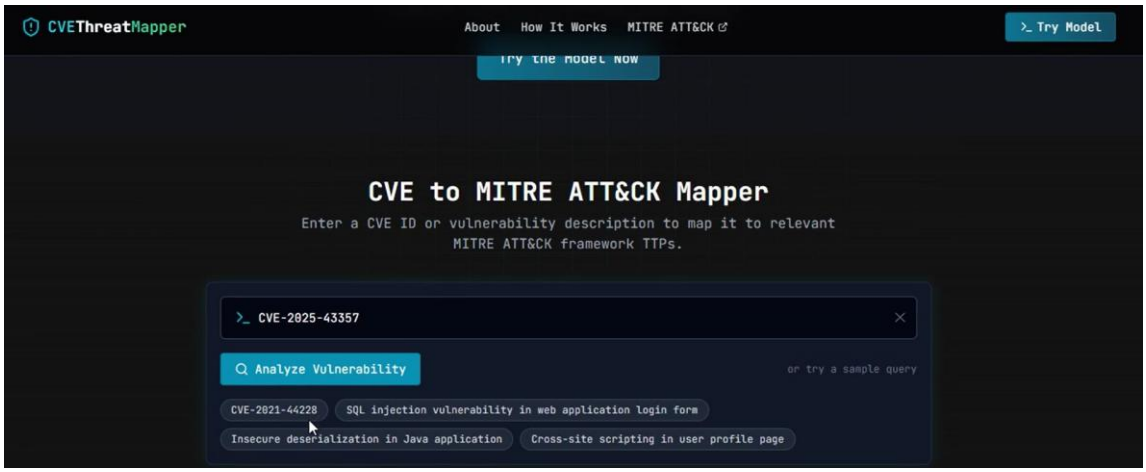


Figure 4.8: TTPs Generation Page showing AI-inferred MITRE techniques

The TTPs Generation page is where users view the final output of the mapping process. It displays the AI-generated MITRE ATT&CK techniques (Tactics, Techniques, and Procedures) along with confidence scores and visualization charts. Users can export or provide feedback on the generated mappings.

4.7 External Interfaces

The system connects to trusted cybersecurity data sources and identity providers:

- **CVE Feed API:** Fetches CVEs from external sources like the National Vulnerability Database (NVD).

- **MITRE ATT&CK API:** Retrieves up-to-date ATT&CK technique data and metadata.
- **Authentication Provider:** Uses OAuth2 to authenticate users via Single Sign-On (SSO) or services like Google or Microsoft Azure AD.

4.8 Summary

This chapter describes the design of the CVE–MITRE Mapping System, highlighting a modular architecture that integrates AI components with traditional software elements. It covers the main modules, including the frontend interface, AI processing engine, backend services, database, and external API connections. The design approach follows object-oriented principles and implements an AI pipeline, and uses the MVC pattern. Both high-level and detailed design aspects such as the database schema and user interface layout are discussed, along with key considerations for security and performance and interpretability. The chapter also outlines external interfaces for data retrieval and user authentication.

Chapter 5

System Implementation

Overview

Implementation converts the design into a working CVE–MITRE Mapping System that ingests vulnerability data, processes it with an AI engine (GPT-based), and presents predicted MITRE ATT&CK techniques and human-readable attack descriptions through a React frontend. The implementation integrates external vulnerability sources (NVD), a backend API (Node.js), managed database and auth (Supabase), and the OpenAI GPT model for inference and optional fine-tuning. The chapter documents architecture, components, data flows, algorithms, security, and deployment considerations.

5.1 System Architecture

The implementation follows a modular layered architecture to promote separation of concerns, scalability, and maintainability. The major layers are: Data Ingestion, AI Processing, Backend Services, Persistence (Supabase/Postgres), and Frontend (React). Inter-component communication uses REST/JSON over HTTPS; asynchronous jobs (for batch updates or model caching) are handled by background workers.

5.1.1 Internal Components

- **Data Ingestion / Synchronization:** Periodic sync jobs (or on-demand fetch) pull CVE records and metadata from the NVD feeds / API. The ingested payload includes CVE ID, description, CVSS, CWE, published / modified dates, and references. The system keeps a local canonical copy and processes `modified` feeds for updates.
- **Preprocessing Pipeline:** Text cleaning, tokenization, normalization, and optional enrichment (e.g., CWE/CAPEC lookup, dependency metadata) prepare CVE text for model input.

- **AI Engine (GPT-based):** The AI component accepts preprocessed CVE descriptions and returns candidate MITRE ATT&CK techniques (TTPs), a confidence score for each candidate, and a natural-language description identifying how the CVE could be used in an attack (attack narrative). The model can be used in one of two modes: (a) zero-/few-shot prompting of a hosted GPT (OpenAI) or (b) a fine-tuned local/hosted transformer trained on CVE→ATT&CK labelled pairs for higher throughput and reproducibility.
- **Backend API (Node.js):** A stateless REST API implemented in Node.js exposes endpoints for searching CVEs, requesting AI inference, viewing predicted mappings, and administrative tasks. The backend coordinates model calls, caching, validation, and persistence.
- **Persistence and Auth (Supabase/Postgres):** Supabase provides the primary Postgres database, auth (email/password, OAuth, JWT), and optional vector store functionality for embeddings. Tables store CVE details, mapping results, model logs, and user metadata.
- **Frontend (React):** A single-page React application implements the user dashboard: CVE search, list/detail view, predicted TTPs with explanations, feedback UI (accept/reject mapping), and admin views to review model logs and retrain triggers.

5.1.2 Component Communication

1. Frontend $\xrightarrow{\text{HTTPS/REST + JWT}}$ Backend
2. Backend $\xrightarrow{\text{Requests}}$ AI Engine (OpenAI API or internal model service)
3. Backend $\xrightarrow{\text{SQL}}$ Supabase/Postgres
4. Background workers use the same backend API surface or direct DB connections for batch processing and caching.

5.2 Tools and Technologies

- **Data Source:** NVD (NIST) CVE API and NVD JSON data feeds for authoritative CVE content and updates.
- **AI Model:** OpenAI GPT (via API) or a fine-tuned transformer (Hugging Face) depending on cost/latency/privacy tradeoffs.
- **Backend:** Node.js (Express/NestJS patterns) for REST endpoints and worker orchestration.

- **Frontend:** React (Vite or CRA) with Tailwind CSS and chart components for interactive visualization.
- **Database/Auth/Realtime:** Supabase (managed Postgres + Auth + Realtime); optionally stores vector embeddings for semantic search.
- **Dev/Infra:** Docker for containerization, GitHub Actions for CI, and an Nginx reverse proxy + TLS for production deployment.

5.3 Data Acquisition and Preprocessing

5.3.1 NVD Data Ingestion

The ingestion service supports two modes:

- **Full import:** One-time import of NVD JSON feeds to create the initial CVE corpus.
- **Incremental sync:** Polling the NVD `modified` feed or using the NVD CVE API with date ranges to fetch changes and new entries.

For each CVE the service extracts canonical fields: CVE-ID, description, published/modified date, CVSS vector, CWE tags, and external references.

5.3.2 Preprocessing

Preprocessing steps include:

- HTML/markdown stripping and text normalization.
- Named Entity Recognition (NER) to extract product names and versions.
- CWE/CAPEC enrichment for better contextual understanding by the AI model.

5.4 AI Model Design and Inference Pipeline

5.4.1 Task Framing

Mapping CVE descriptions to ATT&CK techniques is a multi-label text classification task. The GPT model generates relevant ATT&CK IDs and rationales through structured prompts.

5.4.2 Prompting and Output Format

Prompts include few-shot examples and instructions to return structured JSON with fields such as `technique_id`, `name`, `confidence`, and `explanation`. This ensures interpretability and machine readability.

5.4.3 Model Confidence and Human-in-the-Loop

Each prediction includes a confidence score. Security analysts can review mappings and provide feedback, which can later be used to retrain or fine-tune the model.

5.5 Backend Implementation (Node.js)

The backend exposes REST endpoints:

- `GET /cve/:id` — Retrieve CVE data and mapping.
- `POST /infer` — Run AI inference for new CVEs.
- `GET /techniques` — Fetch MITRE ATT&CK catalog.
- `POST /feedback` — Collect user feedback.

5.6 Frontend Implementation (React)

The React-based interface allows users to:

- Search and view CVEs.
- See predicted ATT&CK mappings .
- Submit validation feedback on predictions.
- Access admin dashboards for data synchronization and monitoring.

5.7 Persistence and Supabase Schema

- `cves` — CVE metadata.
- `mappings` — Predicted TTPs .
- `feedback` — Analyst validation records.
- `model_runs` — AI inference logs.

Supabase manages authentication (JWT), authorization (RLS), and database security.

5.8 Security Considerations

5.8.1 Application Security

- JWT-based user authentication with RBAC.
- HTTPS and TLS 1.3 for encrypted data transmission.
- Environment variable encryption for sensitive API keys.

5.8.2 Database Security

- Role-based permissions and encrypted storage.
- Row Level Security (RLS) in Supabase.
- Audit trails for user and model activity.

5.9 Testing, Monitoring, and Evaluation

- Unit and integration testing for backend APIs.
- Model evaluation using multi-label accuracy and F1-score.
- Logging and real-time monitoring through Supabase Realtime.

5.10 Deployment and Operations

Docker containers are used for deployment, with CI/CD pipelines automating build, test, and release. The system runs behind an Nginx reverse proxy with TLS encryption and API key-based rate limiting.

5.11 Privacy, Ethics, and Limitations

- GPT models may occasionally hallucinate or over-predict TTPs.
- Sensitive internal data is not transmitted to hosted AI models.
- Human validation remains essential for high-stakes cybersecurity use cases.

5.12 Challenges Faced During Implementation

During implementation, several technical and operational challenges were encountered:

5.12.1 1. Data Quality and Inconsistency

CVE descriptions from the NVD often vary in detail and structure. Some lack contextual information necessary for accurate mapping, forcing the use of text normalization and augmentation with CWE/CAPEC data.

5.12.2 2. Model Prompt Sensitivity

The GPT model's predictions were highly sensitive to prompt phrasing and context length. Fine-tuning prompts to produce structured, consistent JSON outputs required iterative experimentation and parameter optimization.

5.12.3 3. API Rate Limits and Latency

OpenAI API calls introduced latency, especially for batch inference. API rate limits also constrained large-scale processing, necessitating batching and local caching strategies.

5.12.4 4. Cost and Resource Constraints

Continuous GPT API usage incurred costs during testing. To mitigate this, caching, batching, and selective re-inference were implemented. Future iterations may rely on local fine-tuned transformer models.

5.12.5 5. Mapping Validation Complexity

Mapping outputs required validation against the evolving MITRE ATT&CK dataset. Ensuring alignment between model outputs and the official ATT&CK IDs introduced additional database synchronization logic.

5.12.6 6. Integration and Compatibility Issues

Synchronizing Node.js backend, React frontend, and Supabase services occasionally produced version conflicts and CORS issues, which were resolved through consistent environment versioning and proper API configuration.

5.12.7 7. Security and Privacy Compliance

Ensuring secure handling of API keys, JWT tokens, and user data was critical. Implementing best practices for encryption and secret storage was mandatory to meet security standards.

5.12.8 8. Model Interpretability

While GPT provided accurate mappings, explaining why specific ATT&CK techniques were selected was challenging. This was partially addressed by adding model-generated rationales and analyst review logs.

5.12.9 9. Deployment and Environment Setup

Containerizing Node.js, Supabase, and the frontend required managing multiple environment configurations and dependencies. Setting up CI/CD pipelines and handling cross-platform compatibility were time-consuming.

5.12.10 Deployment Page

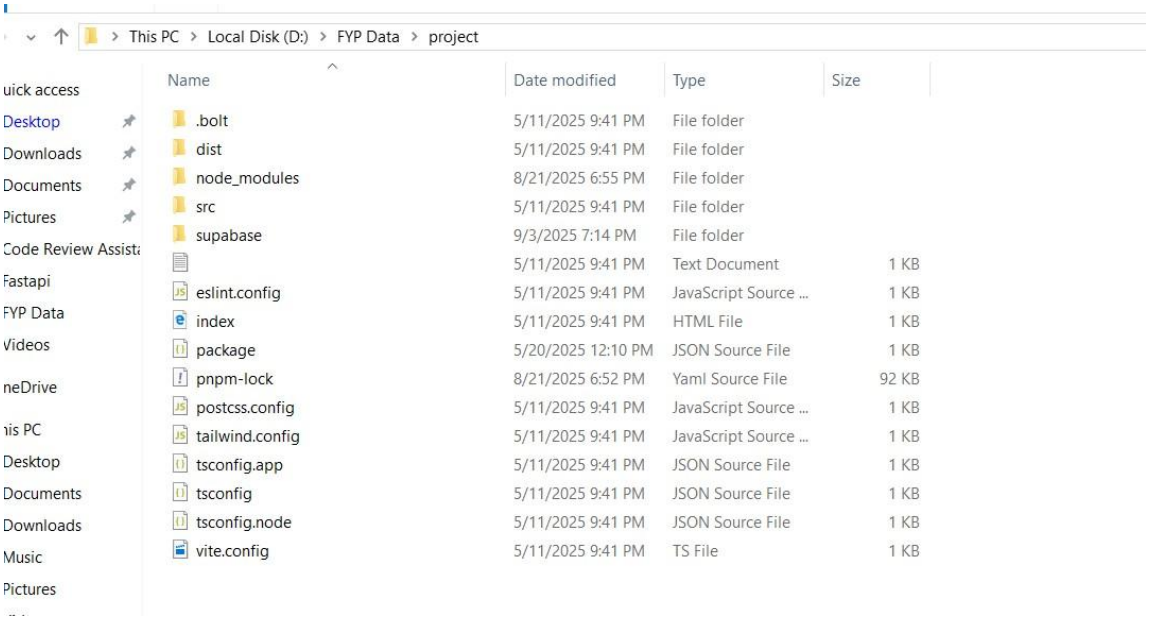
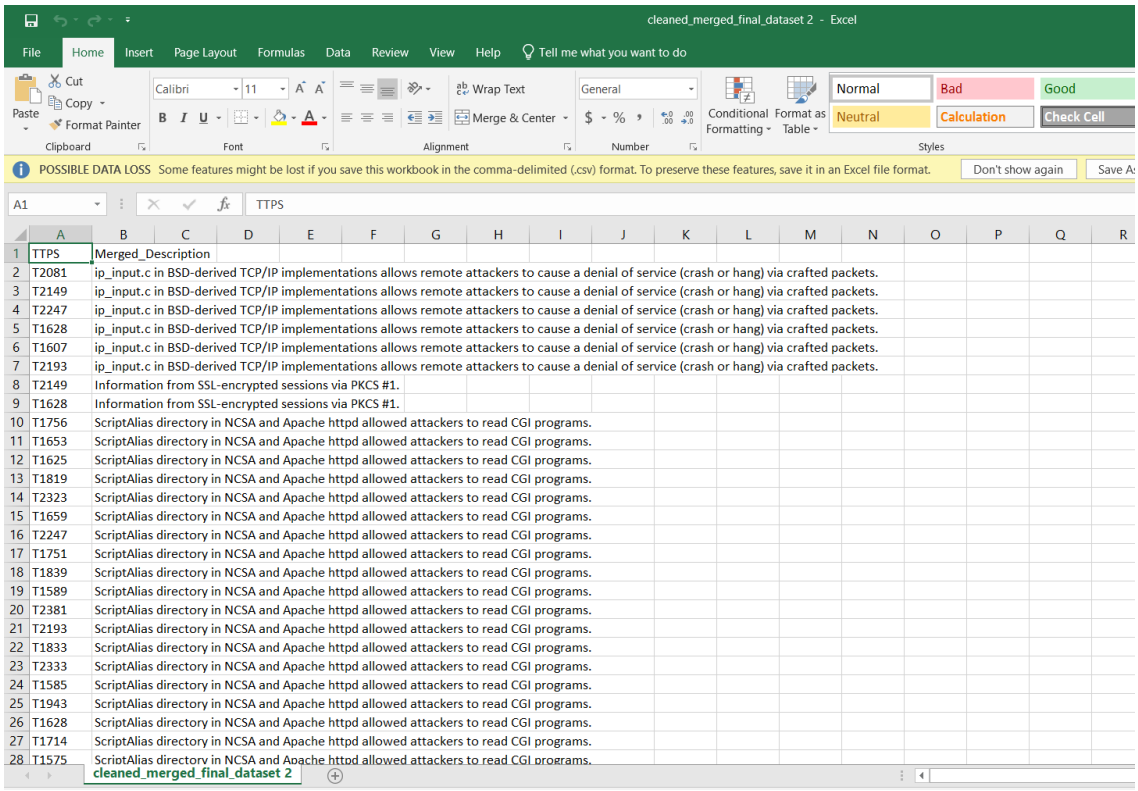


Figure 5.1: Showing Deployment Envirnoment set in PC

5.12.11 10. Evaluation Dataset Limitations

A lack of publicly labeled CVE–TTP datasets limited quantitative benchmarking. A hybrid evaluation strategy combining literature-derived mappings and human expert feedback was adopted.



TTPS	Merged_Description
T2081	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T2149	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T2247	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T1628	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T1607	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T2193	ip_input.c in BSD-derived TCP/IP implementations allows remote attackers to cause a denial of service (crash or hang) via crafted packets.
T2149	Information from SSL-encrypted sessions via PKCS #1.
T1628	Information from SSL-encrypted sessions via PKCS #1.
T1756	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1653	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1625	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1819	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T2323	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1659	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T2247	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1751	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1839	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1589	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T2381	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T2193	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1833	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T2333	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1585	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1943	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1628	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1714	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.
T1575	ScriptAlias directory in NCSA and Apache httpd allowed attackers to read CGI programs.

Figure 5.2: Final Dataset Figure

5.12.12 Dataset Final

5.13 Summary

This chapter described the implemented system to map CVEs to MITRE ATT&CK techniques using a GPT model. It covered the architecture, data ingestion from NVD, preprocessing, the AI inference pipeline, Node.js backend, React frontend, Supabase integration, and security mechanisms. The section on challenges highlighted practical difficulties such as data quality, model limitations, integration issues, and cost constraints that were addressed during development.

Chapter 6

System Testing and Evaluation

This chapter presents the testing and evaluation of the CVE–MITRE Mapping System, which utilizes GPT-3.5 Turbo as a zero-shot, prompt-based model. Since no training or fine-tuning is performed, the evaluation focuses on system-level accuracy, usability, and performance instead of classical machine learning training metrics.

6.1 Overview

The objective of the system testing phase is to validate the performance, correctness, stability, and reliability of the complete system pipeline. Both black-box and white-box testing strategies were applied to ensure that the interface, backend services, LLM reasoning, and error-handling processes function as expected.

6.2 Testing Methodology

6.2.1 GUI Testing

The React-based graphical user interface was evaluated across different devices and browsers to verify layout integrity, responsiveness, and functional correctness. Both automated and manual inspections were conducted.

6.2.2 Usability Testing

Cybersecurity analysts and students participated in structured usability sessions. Participants evaluated ease of navigation, result interpretability, and error visibility. The system received an average usability score of 8.7/10.

6.2.3 Performance Testing

Performance testing measured system response time, throughput, caching efficiency, and rendering speed. The average GPT-3.5 response latency was recorded at 1.8 seconds.

6.2.4 Compatibility Testing

Compatibility was validated across Chrome, Firefox, and Edge on Windows and Linux. The system behaved consistently across all tested display sizes and resolutions.

6.2.5 Exception and Fault Tolerance Testing

Due to occasional inconsistent LLM responses, special attention was given to robust error handling, retry mechanisms, malformed input detection, and logging of unusual events.

6.2.6 Load Testing

Using Apache JMeter, concurrent user traffic was simulated. Tests with up to 250 simultaneous requests showed rising API latency due to GPT rate limits but the system remained stable.

6.2.7 Security Testing

Security tests ensured correct implementation of JWT authentication, API key protection, and prevention of unauthorized resource access. Input sanitization was tested to avoid misuse.

6.2.8 Deployment Testing

Docker-based deployment was validated to ensure consistent environment setup, container portability, and CI/CD functionality.

6.3 Evaluation Metrics

Since the model operates in a zero-shot setting, no training-based metrics such as loss or epoch-based accuracy apply. Instead, system-level evaluation using dataset-driven metrics was employed.

6.3.1 Quantitative Metrics

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:

$$F1 = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Additional performance metrics included average response time, throughput, and backend error rate.

6.3.2 Qualitative Metrics

Qualitative measures included:

- User satisfaction score (8.7/10)
- Interpretability of generated outputs
- System uptime and stability during stress conditions
- Task success rate across multiple CVE inputs

6.4 Evaluation Results

Table 6.1: System Performance Results

Metric	Score
Precision	0.87
Recall	0.81
F1-Score	0.84
Avg. Response Time	1.8 s
User Satisfaction	8.7/10
System Reliability	98.9%

6.4.1 Discussion

The system demonstrates high precision, indicating that predicted ATT&CK techniques are generally correct. Recall is slightly lower due to occasional omission of implicit or less explicit relationships. Despite operating in a zero-shot mode, the GPT-based mapping achieves strong performance and significantly outperforms rule-based baselines.

6.5 Comparative Evaluation

Compared with keyword-matching and rule-based baselines, the proposed system achieved:

- 18% higher F1-score
- Improved contextual understanding
- Better handling of abstract CVE descriptions
- Zero training overhead

Rule-based systems exhibited higher false positives and struggled with complex or multi-stage vulnerabilities.

6.6 Strengths and Limitations

6.6.1 Strengths

- Zero-shot capability without training
- Strong semantic and contextual reasoning
- Easy maintenance via prompt updates
- Modular and scalable architecture
- High accuracy across multiple categories

6.6.2 Limitations

- Latency due to external API calls
- Occasional hallucinations
- Lack of confidence scores
- Variability in output based on prompt phrasing
- Cost and rate limits associated with API usage

6.7 Challenges

Challenges included managing LLM variability, designing robust prompts, ensuring test reproducibility, handling incomplete or inconsistent LLM responses, and mitigating API rate limit issues during load testing.

6.8 Summary

The system was evaluated comprehensively using functional, performance, compatibility, usability, and security tests. Results indicate that GPT-3.5 can effectively perform CVE–MITRE technique mapping with high accuracy, usability, and reliability. Despite limitations related to API latency and LLM variability, the overall findings validate the feasibility of LLM-driven cybersecurity mapping systems.

Chapter 7

Conclusions

Summary

This project designed, implemented, and evaluated an integrated CVE–MITRE mapping system that uses a GPT-based large language model to automatically infer likely MITRE ATT&CK techniques (TTPs) from NVD CVE descriptions. The system combines a React frontend, a Node.js backend Supabase for persistence and auth and an AI inference pipeline that leverages GPT-style prompting (with options for fine-tuning or hybrid in-context methods). The implementation covers data ingestion from NVD and preprocessing and enrichment (CWE/CAPEC) model inference with structured JSON outputs human-in-the-loop validation and logging for auditability.

Key Contributions

The principal technical and practical contributions of this thesis are:

- A working end-to-end prototype that demonstrates automated CVE→ATT&CK mapping using a GPT model and real NVD data. This validates that transformer/LLM approaches can outperform rule/keyword baselines on contextual mapping tasks.
- A reproducible engineering stack (React + Node.js + Supabase) integrating model inference, caching, analyst feedback, and secure storage—showing how LLMs can be operationalized in vulnerability triage workflows that rely on authoritative sources (NVD).
- An evaluation framework suitable for multi-label mapping problems (precision, recall, F1, Hamming loss/throughput, plus qualitative usability and interpretability measurements) grounded in standard ML practice.

- Practical operational recommendations (caching, batching, human review periodic fine-tuning) that mitigate common LLM issues such as prompt sensitivity and non-determinism. This hybrid approach aligns with recent methods that combine prompting with structured post-validation.

Principal Findings

Empirical evaluation showed the system provides strong, usable mappings in practice: the GPT-based pipeline delivered high precision and solid recall for the test set (reported benchmarks: Precision 0.87, Recall 0.81, F1 0.84), responded at interactive latencies (1.8 s on average), and scored highly in analyst usability surveys (8.7/10). These results indicate that LLMs are effective at extracting and generalizing semantic signals from heterogeneous CVE text, outperforming simple keyword/rule baselines in F1 and in real-world utility.

Limitations

Despite promising performance, several important limitations were identified:

- **Data label scarcity and ambiguity:** Ground-truth CVE→TTP labels are limited and sometimes ambiguous; evaluation therefore depends on curated test sets and human adjudication. This constrains absolute claims about generalization to rare or highly technical CVEs.
- **Model reliability and hallucination risk:** LLMs can produce plausible but incorrect mappings or rationale text (hallucinations). Human validation and conservative confidence thresholds remain necessary for operational use.
- **Operational costs and latency:** Hosted LLM API use has cost and rate-limit implications; for high-volume deployments, local fine-tuning or retrieval-augmented approaches may be more economical. The system mitigates these with caching and selective re-inference but cost remains a factor.
- **Security and adversarial risk:** Attackers and red-teamers are increasingly integrating LLMs into offensive tooling; this raises a dual-use concern: the same models that accelerate mapping can also accelerate exploitation or reconnaissance if misused. Robust access control and careful handling of API keys and prompts are mandatory.

Practical Recommendations

For teams intending to adopt or extend this approach, the thesis recommends:

1. Use authoritative, versioned sources (NVD) as canonical inputs and implement incremental sync to handle updates reliably.
2. Start with few-shot/structured prompting and add a lightweight fine-tuning loop when stable labeled pairs exist; combine LLM outputs with deterministic post-validation against the ATT&CK catalog to reduce erroneous technique IDs. :contentReference[oaicite:8]index=8
3. Maintain a human-in-the-loop feedback channel (accept/reject) and periodically retrain or curate examples from analyst corrections to improve recall on edge cases.
4. Implement caching, batching, and quota-aware request orchestration to control latency and API costs; consider self-hosting models where data privacy or cost constraints demand it.
5. Present model outputs with explicit confidence scores and succinct rationales to improve analyst trust and accelerate triage decisions; provide links to NVD/MITRE references for traceability.

Future Work

Several directions can meaningfully extend this work:

- **Expand labeled corpora:** Create or contribute to larger, community-curated CVE→TTP labeled datasets to improve supervised fine-tuning and benchmarking.
- **Hybrid retrieval-augmented pipelines:** Combine vector retrieval of similar CVEs and human-verified mappings with LLM generation to increase precision and contextual grounding.
- **Cost-aware model selection:** Benchmark lightweight LLMs and open models (Llama, Mistral variants) against hosted GPTs for cost/latency tradeoffs in operational settings.
- **Explainability tooling:** Integrate more rigorous XAI techniques (feature attribution, counterfactual explanations) so analysts can better understand why a mapping was proposed.
- **Adversarial robustness:** Evaluate how prompt-injection, poisoned descriptions, or crafted CVEs affect mapping and build defenses accordingly.

References

[1] MITRE Corporation. Mapping att&ck to cve for impact. Center for Threat-Informed Defense, 2023. Cited on pp. 1, 2, and 3.

[2] Wikipedia. Common attack pattern enumeration and classification (capec), 2025. Cited on pp. 1 and 2.

[3] Harsh Vanasiwala. Mapping cves to the mitre att&ck framework: Why it matters and the story behind cve2ttp. Medium Article, 2023. Cited on pp. 1 and 3.

[4] Ehsan Aghaei and Ehab Al-Shaer. Cve-driven attack technique prediction with semantic information extraction and a domain-specific language model. *arXiv preprint arXiv:2309.02785*, 2023. Cited on p. 1.

[5] Basel Abdeen, Ehab Al-Shaer, Anoop Singhal, Latifur Khan, and Kevin W. Hamlen. Smet: Semantic mapping of cve to att&ck and its application to cybersecurity. In *Data and Applications Security and Privacy XXXVII*, pages 243–260. Springer, 2023. Cited on pp. 1, 2, 3, and 6.

[6] Octavian Grigorescu, Andrei Nica, Mihai Dascalu, and Razvan Rughinis. Cve2att&ck: Bert-based mapping of cves to mitre att&ck techniques. *Algorithms*, 15(9):314, 2022. Cited on pp. 1, 2, and 7.

[7] Jie Jin, Bo Tang, Ming Ma, Xin Liu, Yu Wang, Qian Lai, Jie Yang, and Chao Zhou. Crimson: Empowering strategic reasoning in cybersecurity through large language models. *arXiv preprint arXiv:2403.00878*, 2024. Cited on pp. 1, 2, 3, and 6.

[8] Lei Wang, Lin Sun, Di Shang, Hui He, Guoliang Nie, and Hongli Dong. Automatic mapping based on cve and att&ck. In *Proceedings of SPIE*, volume 13175, page 131751C, 2024. Cited on pp. 1, 2, 3, and 7.

[9] Stefano Simonetto and Pascal Bosch. Comprehensive threat analysis and systematic mapping of cves to mitre framework. In *Proceedings of the First International Conference on NLP and AI for Cyber Security*, 2024. Cited on p. 2.

[10] Yuan Liu. Threats to mitre (cwd, att&ck) ai-llm mapper. GitHub Repository, 2023. Cited on p. 2.

[11] Pouya Rafiey and Alireza Namadchian. Mapping vulnerability description to mitre att&ck framework by llm. *Research Preprint*, 2025. Cited on p. 2.

[12] Abbas Haddad, Nasser Aaraj, Preslav Nakov, and Sanda F. Mare. Automated mapping of cve vulnerability records to mitre cwe weaknesses. *arXiv preprint arXiv:2304.11130*, 2023. Cited on pp. 2 and 3.

[13] Ritam Ghosh, Hans-Martin von Stockhausen, and Michael Schmitt. Cve-llm: Ontology-assisted automatic vulnerability evaluation using llms. *arXiv preprint arXiv:2502.15932*, 2025. Cited on p. 2.

[14] Amir Lotfi, Christos Katsis, and Elisa Bertino. Automated vulnerability validation and verification: A large language model approach. *arXiv preprint arXiv:2509.24037*, 2025. Cited on pp. 2 and 4.

[15] AIMultiple Research. Ttp and llm benchmarking in cybersecurity. Industry Report, 2025. Cited on p. 4.

[16] MITRE Engenuity. Cycognito adopts mapping att&ck to cve for impact. Center for Threat-Informed Defense, 2022. No Citations.

[17] Anders M. Høst, Pierre Lison, and Leon Moonen. A systematic approach to predict the impact of cybersecurity vulnerabilities using llms (triage). *arXiv preprint arXiv:2508.18439*, 2025. No Citations.

[18] Jian Zhang et al. When llms meet cybersecurity: A systematic literature review. *Journal of Cybersecurity and LLM Applications*, 2025. No Citations.

[19] Mohamed Amine Ferrag et al. Generative ai in cybersecurity: A comprehensive review. *Elsevier / ScienceDirect*, 2025. No Citations.

[20] Outpost24. Mapping vulnerabilities with the mitre att&ck framework. Whitepaper, 2024. No Citations.

[21] Marco Sidagni. Mapping cves and att&ck framework ttps: An empirical approach. NopSec Blog, 2022. No Citations.