Requirement Analysis

Group Name: *TeamOne*

ID

Team member Name

7741923

Abdulhayye Maricar

7811056

Abdul Rahim Kalsekar

8061634

Ali Sina Mohammad Arif

7869472

Mansoor Kalemzai

8044661

Varun Tulsiani





#### Proposal Cover Sheet

|  |  |  |
| --- | --- | --- |
| **Subject Code:** | CSIT321 |  |
| **Subject Name:** | Project |  |
| **Submission Type:** | Report |  |
| **Project Title:** | Requirement Analysis Report | |
| **Student/Team Name:** | Team Name *TeamOne*  **Team members**  Student Name Student ID Role Ali Sina Mohammad Arif 8061634 Leader  Mansoor Kalemzai 7869472 Presentation Coordinator  Varun Tulsiani 8044661 Management Tool Coordinator  Abdul Rahim Kalsekar 7811056 Submission Coordinator  Abdulhayye Maricar Asfaq Ahamed 7741923 Scribe | |
| **Student Phone/Mobile No.** | +971568648103, +971569879006, +971562859550, +971524628293,  +971507907565 | |
| **Student E-mail:** | [ama438@uowmail.edu.au](mailto:ama438@uowmail.edu.au), [mk085@uowmail.edu.au](mailto:mk085@uowmail.edu.au), [vmt979@uowmail.edu.au](mailto:vmt979@uowmail.edu.au), [arrk807@uowmail.edu.au](mailto:arrk807@uowmail.edu.au), [amaa959@uowmail.edu.au](mailto:amaa959@uowmail.edu.au) | |
| **Lecturer Name:** | Dr. May El Barachi | |
| **Due Date:** | 06 Nov, 2024 |  |
| **Date Submitted:** | 06 Nov, 2024 |  |
| **PLAGIARISM:**  The penalty for deliberate plagiarism is FAILURE in the subject. Plagiarism is cheating by using the written ideas or submitted work of someone else. UOWD has a strong policy against plagiarism.  The University of Wollongong in Dubai also endorses a policy of non-discriminatory language practice and presentation.  **PLEASE NOTE:** STUDENTS MUST RETAIN A COPY OF ANY WORK SUBMITTED | | **DECLARATION:**  I/We certify that this is entirely my/our own work, except where I/we have given fully-documented references to the work of others, and that the material contained in this document has not previously been submitted for assessment in any formal course of study. I/we understand the definition and consequences of plagiarism. We/I declare that the project proposal has not been used in any UOWD courses before and that this project idea is a total new idea of the project team  **Signature of Student:** Ali Sina |

**Optional Marks: Comments:**

https://my.uowdubai.ac.ae/images/scissors.gif

|  |  |
| --- | --- |
| **Lecturer Project Proposal Receipt** (To be filled in by student and retained by Lecturer upon return of assignment) | |
| **Subject:** | **Project Title:** |
| **Student / Team Name:** | **Student Number:** |
| **Due Date:** | **Date Submitted:** |
| **Signature of Student:** |  |

|  |  |
| --- | --- |
| **Student Project Proposal Receipt** (To be filled in and retained by Student upon submission of assignment) | |
| **Subject:** | **Project Title:** |
| **Student/Team Name:** | **Student Number:** |
| **Due Date:** | **Date Submitted:** |
| **Signature of Lecturer:** |  |

Table of Contents

[1. Introduction 4](#_Toc194181359)

[1.1 Purpose 4](#_Toc194181360)

[1.2 Document Conventions 4](#_Toc194181361)

[1.3 Intended Audience and Reading Suggestions 4](#_Toc194181362)

[1.4 Product Scope 4](#_Toc194181363)

[1.5 References 5](#_Toc194181364)

[2. Overall Description 6](#_Toc194181365)

[2.1 Product Perspective 6](#_Toc194181366)

[2.2 Product Functions 6](#_Toc194181367)

[2.3 User Classes and Characteristics 7](#_Toc194181368)

[2.4 Operating Environment 8](#_Toc194181369)

[2.5 Design and Implementation Constraints 8](#_Toc194181370)

[2.6 User Documentation 8](#_Toc194181371)

[2.7 Assumptions and Dependencies 9](#_Toc194181372)

[3. External Interface Requirements 9](#_Toc194181373)

[3.1 User Interfaces. 9](#_Toc194181374)

[3.2 Hardware Interfaces 10](#_Toc194181375)

[3.3 Software Interfaces 10](#_Toc194181376)

[3.4 Communications Interfaces 10](#_Toc194181377)

[4. Domain Model 10](#_Toc194181378)

[5. System Features (Use Cases) 11](#_Toc194181379)

[5.1 Use Case Overview 11](#_Toc194181380)

[5.2 Model Vulnerability Scanning 11](#_Toc194181381)

[5.3 Mitigation Recommendations 12](#_Toc194181382)

[5.4 User Registration and Login 12](#_Toc194181383)

[5.5 User Dashboard and Report Access 13](#_Toc194181384)

[6. Non-Functional Requirements 14](#_Toc194181385)

[6.1 Performance Requirements 14](#_Toc194181386)

[6.2 Safety Requirements 14](#_Toc194181387)

[6.3 Security Requirements 14](#_Toc194181388)

[6.4 Software Quality Attributes 14](#_Toc194181389)

[7. Other Requirements 14](#_Toc194181390)

[Appendix: Glossary 15](#_Toc194181391)

Table of Figures

[Figure 1 Use Case Overview 13](#_bookmark22)

[Figure 2 Model Vulnerability Scanning 14](#_bookmark24)

[Figure 3 Mitigation Recommendations 15](#_bookmark26)

[Figure 4 User Registration and Login 16](#_bookmark28)

[Figure 5 User Dashboard & Access 17](#_bookmark30)

# Introduction

## Purpose

The purpose of **Protego – AI Vulnerability Scanner for Neural Networks** is to detect and mitigate security vulnerabilities associated with Model Inversion attacks in AI systems. Created by TeamOne in collaboration with DTS Solutions, this scanner targets the critical Deployment stage of the AI lifecycle. Data‑sensitive industries—including healthcare, finance, and legal services—are particularly vulnerable to model inversion attacks, which enable adversaries to deduce private training data from model outputs. Organizations can evaluate and safeguard their models with Protego, thereby guaranteeing secure deployments and adherence to laws like GDPR and HIPAA.

## Document Conventions

This document complies with IEEE 830-98, the Standard for Software Requirements Specification. All acronyms and technical terminology are defined upon first use, and a glossary section provides a summary. To facilitate traceability, each requirement is assigned a unique number and category.

## Intended Audience and Reading Suggestions

This document is intended for Security Engineers, System Architects, Project Managers, Compliance Officers, and other stakeholders involved in AI deployment. The Introduction provides an overview, while technical teams should focus on Sections 2 and 3 for detailed requirements. Compliance Officers may find Section 2.5 on regulatory alignment particularly relevant.

## Product Scope

**Protego – AI Vulnerability Scanner** aims to secure Convolutional Neural Networks (CNNs) by identifying Model Inversion vulnerabilities on-demand. Unlike existing tools that primarily focus on adversarial robustness, Protego is specifically designed for post-training vulnerability scanning during the deployment stage. Key features include:

* Detection of Model Inversion risks via score-based and boundary-based attack simulations
* Actionable mitigation insights and vulnerability report.
* Secure, password-protected report generation

## References

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Author | Version | Date | Source Location |
| AI Vulnerability Scanner SRS | Team One | 1.0 | 28/09/2024 | Available upon request from Team One at: [ama438@uowmail.edu.a](mailto:ama438@uowmail.edu.au) [u](mailto:ama438@uowmail.edu.au) [mk085@uowmail.edu.au](mailto:mk085@uowmail.edu.au) [vmt979@uowmail.edu.au](mailto:ama438@uowmail.edu.au) [arrk807@uowmail.edu.au](mailto:arrk807@uowmail.edu.au) [amaa959@uowmail.edu.](mailto:ama438@uowmail.edu.au) [au](mailto:ama438@uowmail.edu.au) |
| Planning & Feasibility Report | Team One | 1.0 | 20/09/2024 | Available upon request from Team One at the above emails |

# Overall Description

## Product Perspective

**Protego** – AI Vulnerability Scanner for Neural Networks is a specialized tool developed by TeamOne in collaboration with DTS Solutions. It enhances the security of AI models by identifying and mitigating vulnerabilities associated with Model Inversion attacks. Designed for use during the Deployment and Monitoring stages of the AI lifecycle, Protego centers on protecting data privacy and ensuring regulatory compliance (e.g., GDPR, HIPAA) in sectors such as healthcare, finance, and legal services. While it functions independently, it can integrate with existing monitoring systems to provide ongoing post-training assessments.

## Product Functions

The main functions of Protego include:

* **Model Vulnerability Scanning:** Scans AI models—specifically pretrained CNNs (.h5 files)—to detect vulnerabilities to Model Inversion attacks, generating detailed reports with identified risks and recommended mitigation steps.
* **Mitigation Insights:** Provides actionable recommendations for securing models based on the scan results.
* **Data Privacy Compliance:** Ensures that AI model data handling aligns with data privacy regulations (e.g., GDPR, HIPAA) and generates compliance reports for audit purposes.
* **User Access Management:** Manages user roles and permissions to ensure secure access, supporting roles such as Security Engineers, Compliance Officers, and System Administrators.
* **Audit and Security Logs:** Tracks all system activities to support compliance checks and maintain security records.
* **User Dashboard and Report Access:** Provides a React‑based interface for uploading models, accessing scan history, and downloading comprehensive, password‑protected reports.

**System Configuration and Maintenance:** Enables administrators to configure scanning parameters, schedule maintenance, and initiate backups.

## User Classes and Characteristics

**Protego** is designed for multiple user roles, each with distinct access permissions and functional requirements:

* **Security Engineer:** Initiates vulnerability scans, configures scanning parameters, and analyzes scan results.
* **System Administrator:** Manages overall system settings (user permissions, audit logs, backups, maintenance schedules).
* **General User:** Has limited access, primarily to view vulnerability summaries and receive notifications on system updates.

Here is the **User/Function Cross-Reference Matrix** table for the AI Vulnerability Scanner, showing the functions available to each user type:

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Security Engineer | Admin | General User |
| Model Vulnerability Scanning | ✓ | ✓ | ✓ |
| Risk Mitigation | ✓ | ✓ | ✓ |
| User Access Management |  | ✓ |  |
| Audit and Security Logs | ✓ | ✓ |  |
| User Dashboard and Notifications | ✓ | ✓ | ✓ |
| System Configuration and Maintenance |  | ✓ |  |
| Data Management for Model Testing | ✓ | ✓ |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Reports and Notifications | ✓ | ✓ | ✓ |

✓ = Full access to function

**View Only** / **Summaries Only** / **Alerts Only** = Limited access to relevant information within the function

This role-based structure ensures that each user has access only to the necessary features aligned with their responsibilities, enhancing both security and efficiency.

## Operating Environment

**Protego** operates as a cloud‑based web platform. The system’s backend is implemented in Python using FastAPI and TensorFlow, while the front-end is developed with React, leveraging TypeScript and CSS for a responsive and user-friendly interface. For rapid prototyping and model development, Jupyter Notebook is employed.

The entire development lifecycle is managed via GitHub and Notion, ensuring robust version control and comprehensive documentation. All data transmissions are secured using SSL encryption, with access controlled by OAuth2 and JWT, following OWASP best practices. The client interface requires modern web browsers that support HTML5, CSS3, and JavaScript.

## Design and Implementation Constraints

The development and deployment of the AI Vulnerability Scanner adhere to several constraints:

* **Regulatory Compliance**: The scanner must align with industry regulations such as GDPR and HIPAA, ensuring that data privacy and protection requirements are strictly followed.
* **Data Encryption**: All communications and stored data require encryption to prevent unauthorized access, particularly when dealing with sensitive training datasets.
* **Role-Based Access Control (RBAC)**: Access to system features is restricted based on user roles, with permissions configured to limit sensitive operations to authorized personnel only.
* **Modular Updates**: The scanning algorithms and attack detection modules should be easily updatable to address emerging vulnerabilities without affecting overall system stability.

These constraints guide the design of the AI Vulnerability Scanner to ensure it remains secure, compliant, and adaptable to evolving security standards.

## User Documentation

A comprehensive user manual and system documentation will be provided upon deployment. Documentation will include installation, configuration, scanning procedures, notification settings, and user role management. Online help is integrated within the dashboard, and PDF manuals will be available for offline reference.

## Assumptions and Dependencies

The following assumptions and dependencies are identified for the successful operation of the AI Vulnerability Scanner:

* **Availability of Pretrained Models:** Organizations have access to pretrained CNN models (.h5) for scanning.
* **Existing IT Infrastructure:** The system integrates with current network monitoring and management systems.
* **DTS Solutions Collaboration:** Ongoing technical support and updates are provided by DTS Solutions.

**Compliance with Security Standards:** Organizations adhere to internal security protocols aligned with industry best practices.

# External Interface Requirements

## User Interfaces.

The Graphical User Interface (GUI) is the primary portal for controlling Protego. After logging in, users view an overview of recent scan operations, vulnerabilities detected, and available mitigation options for each scanned model. The interface displays the logged-in user’s name and provides intuitive navigation through standard web components (buttons, hyperlinks, etc.).

## Hardware Interfaces

The AI Vulnerability Scanner is intended to be hardware-agnostic, which means it may function independently of specific physical hardware. However, the underlying platform must be capable of operating a web server with PHP support to provide backend functionality. For best results, a dedicated MySQL database for storing and managing vulnerability data might be hosted on the same server as the web server or on a different workstation. To guarantee that all interactive components work properly, end users who visit the scanner's interface must have JavaScript enabled in their web browsers.

## Software Interfaces

The GUI will communicate with both the Database and Scanning Engine Subsystems by executing parameterized commands formatted as follows:

* DB\_Subsys (<Command name>,<param 1>, <param 2>, ,...)
* Scanning Engine Subsystem: SE\_Subsys((<Command name>,<param 1>, <param 2>,…)

The GUI will receive XML replies from both subsystems, in line with the AJAX approach, which allows for real-time modifications to the interface without requiring page reloads.

## Communications Interfaces

To provide smooth interactions within the GUI Subsystem, the system will use Asynchronous JavaScript and XML (AJAX). The front end will be created using JavaScript, CSS, and HTML, with server-side scripts handling backend processing. This configuration provides a fluid, application-like user experience, with smooth navigation and faster reaction times than standard web applications, which rely on frequent page reloads.

# Domain Model

None

# System Features (Use Cases)

## Use Case Overview

A screenshot of a diagram

Description automatically generated

Figure 1 Use Case Overview

## Model Vulnerability Scanning

## A diagram of a company Description automatically generated

Figure 2 Model Vulnerability Scanning

* Name: Model Vulnerability Scanning
* Goal: Identify security vulnerabilities in a pretrained CNN model (.h5) using query‑based model inversion attack simulations (score‑based and boundary‑based).
* Input:
* A pretrained CNN model selected by the user.
* Configuration of scan parameters (e.g., selecting score‑based or boundary‑based attack type).
* Output:
* A detailed, password‑protected scan report indicating detected vulnerabilities and recommended mitigation steps.
* Main Scenario:

A Security Engineer or Admin logs into the system, selects a pretrained CNN model, configures the scan to simulate score‑based and boundary‑based attacks, and initiates the scan. The system processes the model, detects vulnerabilities, and generates a secure report summarizing the findings.

* Pre‑condition:

The user is logged into the system with the appropriate permissions and has selected a valid pretrained model.

* Steps:

1. The user logs into the Protego dashboard.
2. The user navigates to the scanning module.
3. The user selects a pretrained CNN model from the available list (e.g., ResNet, MobileNet, EfficientNet).
4. The user configures the scan parameters (score‑based and/or boundary‑based attacks).
5. The user initiates the scan.
6. The system processes the model and simulates the specified inversion attacks.
7. The system generates and displays a detailed, password‑protected scan report.

* Post‑condition:

The vulnerability scan is completed, and the scan report is stored in the system for later review and download.

* Exceptional Scenario 1:

If the user enters invalid scan parameters, the system prompts the user to correct them before the scan can be initiated.

* Example:

A Security Engineer logs in, selects a MobileNet model, sets the scan to simulate both score‑based and boundary‑based attacks, and initiates the scan. The system completes the scan and generates a report detailing the vulnerabilities found and suggesting actions like adversarial training and input filtering.

## Mitigation Recommendations

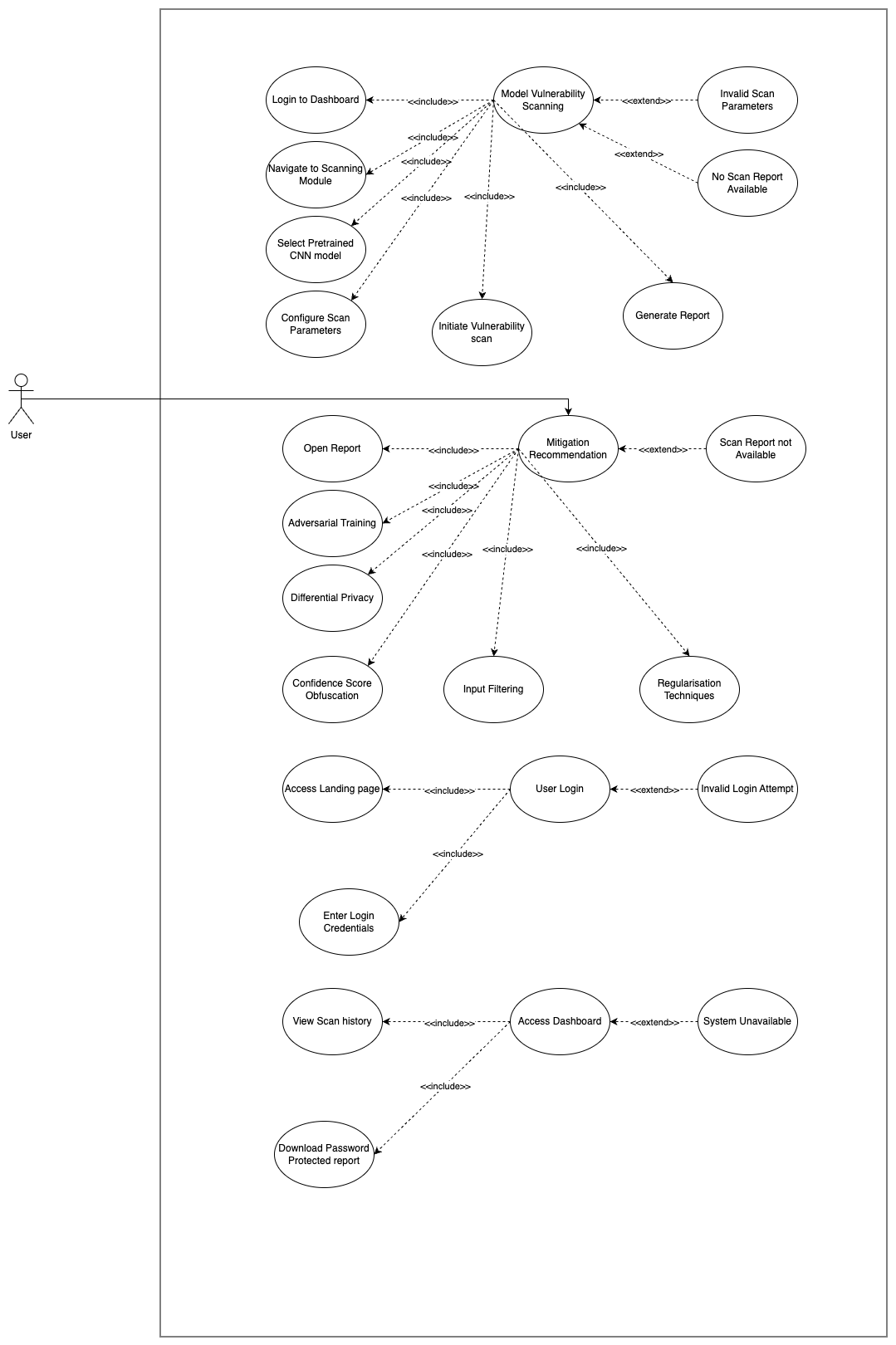
****

Figure 3 Mitigation Recommendations

* Name: Mitigation Recommendations
* Goal: Provide actionable recommendations to remediate vulnerabilities detected in the scan.
* Input:
* A completed scan report from the vulnerability scanning use case.
* Output:
* A list of recommended mitigation strategies tailored to the identified vulnerabilities.
* Main Scenario:

After a scan report is generated, a Security Engineer or Compliance Officer reviews the report to identify recommended mitigation steps.

* Pre‑condition:

A successful scan has been completed, and the report is available.

* Steps:

1. The user opens the completed scan report from the dashboard.
2. The user reviews the mitigation recommendations provided in the report.
3. The recommendations include measures such as adversarial training, differential privacy, confidence score obfuscation, input filtering, and regularization techniques.

* Post‑condition:

The user has a clear understanding of the steps required to mitigate the detected vulnerabilities.

* Exceptional Scenario 1:

If the scan report is not available, the system notifies the user that a vulnerability scan must be completed first.

* Example:

A Compliance Officer accesses a scan report for a ResNet model and finds a list of recommended actions to reduce the risk of data reconstruction, which they then forward to the security team for implementation.

## User Registration and Login

## 

Figure 4 User Registration and Login

* Name: User Registration and Login
* Goal: Allow new users to create accounts and existing users to securely log into the Protego system.
* Input:
* Registration details (e.g., username, password, email).
* Login credentials (e.g., username and password).
* Output:
* A new user account upon successful registration.
* A successful login session for existing users.
* Main Scenario:

Users access the system’s main landing page, where they can either register a new account or log in with existing credentials.

* Pre‑condition:

The system is online, and the user has access to the landing page.

* Steps:

1. The user navigates to the Protego landing page.
2. For registration, the user selects “Register” and provides the required details (username, password, email).
3. The system creates a new account and confirms successful registration.
4. For login, the user enters valid credentials in the login form.
5. The system authenticates the user and grants access to the dashboard.

* Post‑condition:

The user is logged into the system and can access scanning features or account settings.

* Exceptional Scenario 1:

If the user provides invalid credentials, the system displays an error message and denies access.

* Example:

A new user visits the Protego landing page, registers an account, and then logs in to begin scanning their pretrained CNN models for vulnerabilities.

## User Dashboard and Report Access

## A diagram of a company Description automatically generated

Figure 5 User Dashboard & Access

* Name: User Dashboard and Report Access
* Goal: Provide a centralized interface for users to manage scans and download generated reports (without real‑time scanning feedback).
* Input:
* User logs in and accesses the dashboard.
* Output:
* The dashboard displays the scan history and offers options to download password‑protected reports.
* Main Scenario:

The user logs in, uploads a pretrained CNN model for scanning, and after the scan is completed, accesses the dashboard to download the generated report.

* Pre‑condition:

The user is logged in with appropriate access rights.

* Steps:

1. The user logs into the Protego dashboard.
2. Uploads a model and initiates a scan.
3. Navigates to the dashboard after scan completion.
4. Reviews scan history and selects a report to download.

* Post‑condition:

The report remains available for download until the user retrieves it.

* Exceptional Scenario 1:

If the system is unavailable, an error message is displayed prompting the user to try again later.

* Example:

A Security Engineer downloads the report for a MobileNet scan from the dashboard and reviews the mitigation recommendations.

# Non-Functional Requirements

## Performance Requirements

The scanner should be capable of finding vulnerabilities in models with a response time of few seconds, ensuring that real-time feedback can be provided during deployment stage. The platform also needs to be scalable, supporting large neural network models, including CNNs, and adaptable to various deployment environments, from cloud-based systems to other infrastructures. Minimizing latency in data processing is crucial, as reduced latency will prevent delays in threat detection, which is essential for live monitoring scenarios.

## Safety Requirements

Ensuring data integrity is necessary, such that no part of the input data should be altered or modified during scanning or while simulating Model Inversion attacks. Moreover, the system must operate in a fail-safe manner; for example, if a failure occurs, it should default to a safe state, log detailed information about the failure, notify administrators, and refrain from interfering with live AI deployments. Effective error handling is also essential, allowing the scanner to manage errors gracefully and provide diagnostic information when detection errors occur, particularly critical in high-stakes applications such as healthcare and finance.

## Security Requirements

The scanner must ensure confidentiality by employing encryption to protect data transactions, whether in whole or in part, to prevent sensitive information from falling into unauthorized hands during scanning, transmission, and storage. Access to the system should be controlled through role-based access, allowing initiation of scans, viewing of reports, and modification of configurations to only authorized personnels. Compliance with data protection regulations such as GDPR and HIPAA is essential, requiring the solution to minimize exposure of sensitive information and to provide data anonymization wherever feasible. Additionally, comprehensive logging should record all scanner activities, and anomaly detection mechanisms should be in place to alert administrators of suspicious actions, thereby supporting system integrity and facilitating security audits.

## Software Quality Attributes

The scanner will have to be highly reliable, maintaining at least 99% uptime to ensure continuous real-time scanning for vulnerabilities. Usability is also essential; the system should feature an intuitive, user-friendly interface that enables users with limited cybersecurity or AI expertise to easily set up and deploy it. For maintainability, the scanner should support modular updates, particularly in attack detection algorithms, to adapt to emerging types of model inversion attacks and vulnerability patterns.

# Other Requirements

None

# Appendix: Glossary

AI Lifecycle: The complete cycle of developing, deploying, and maintaining an AI model, including data collection, training, evaluation, deployment, and monitoring.

AJAX (Asynchronous JavaScript and XML): A technology that enables asynchronous data retrieval without requiring a full page reload, enhancing user interaction and responsiveness in web applications.

Anomaly Detection: The identification of unusual patterns within data that do not conform to expected behavior, often used for identifying potential security risks.

Audit Logs: Records of system activity, used to track user actions, system changes, and support compliance and security monitoring.

Compliance Officer: A designated role responsible for ensuring that the system adheres to legal and regulatory standards, such as GDPR and HIPAA.

Convolutional Neural Networks (CNNs): A type of neural network particularly effective for image recognition tasks, commonly targeted in model inversion attacks.

Data Encryption: The process of converting information or data into a code to prevent unauthorized access, particularly essential in sensitive data transmission and storage.

Data Privacy Compliance: Ensuring data handling and storage practices align with privacy regulations, such as GDPR and HIPAA.

Data Anonymization: The process of transforming data so that individuals or entities cannot be identified, often used to meet privacy and regulatory requirements.

Deployment Stage: The phase where an AI model is integrated and activated in a production environment for real-world use.

DTS Solutions: The partner organization collaborating on this project, providing technical expertise in developing and supporting the AI Vulnerability Scanner.

Fail-Safe Mechanism: A design feature that enables a system to default to a safe condition in the event of a failure, ensuring that it does not impact other processes.

GDPR (General Data Protection Regulation): A regulation in EU law that focuses on data protection and privacy, setting standards for handling personal data.

GUI (Graphical User Interface): A user interface allowing users to interact with the application through graphical elements like buttons, icons, and windows.

HIPAA (Health Insurance Portability and Accountability Act): A U.S. law governing data privacy and security standards for sensitive health information.

JavaScript: A client-side scripting language used in web development to create dynamic and interactive user experiences.

Latency: The delay between a request and the corresponding response, particularly important in real-time systems where low latency is crucial for effective performance.

Model Inversion Attack: A security threat in which attackers deduce private information from a model's output, potentially exposing sensitive training data.

Modular Updates: The ability to update parts of the system independently without impacting the overall functionality, useful for addressing emerging threats.

MySQL: An open-source relational database management system used for storing and retrieving data.

Performance Requirement: Specifications that define the expected response time, processing speed, and scalability of a system.

PHP: A server-side scripting language widely used for web development, providing the backend functionality for many applications.

RBAC (Role-Based Access Control): A security mechanism that restricts system access based on user roles, allowing only authorized personnel to perform sensitive actions.

Resilience Rating: A measure of an AI model's strength and robustness against specific vulnerabilities, providing organizations with a benchmark for security.

Scheduled Reports: Predefined, automated reports that inform stakeholders about system activity, status, or compliance metrics.

Security Engineer: A professional responsible for managing the security of the AI system, including initiating scans, analyzing results, and implementing security measures.

System Administrator: A role responsible for configuring, maintaining, and managing user permissions within the system, including handling audit logs and backups.

System Reliability: The ability of a system to operate continuously without failure, often quantified as uptime percentage.

User Dashboard: A centralized interface displaying key information, alerts, and controls for users, allowing for easy interaction with the system.

Vulnerability Database Subsystem: A component of the scanner that stores and manages known vulnerabilities, facilitating real-time scanning and alerting.

XML (eXtensible Markup Language): A standard data format used for structuring data, enabling information sharing between the GUI and backend components.

**Document Update Notice**  
This Requirement Analysis document has been updated as of **25 March 2025**.