Faculty of Computers and Data Science

Alexandria National University

Integration of Penetration Testing and Threat Modeling for Continuous Monitoring

TIBSA Platform

A Graduation Project Submitted to
The Department of Cybersecurity

In Partial Fulfillment of the Requirements for
Bachelor Degree in Cybersecurity

Prepared by:

Nadine Rasmy Soliman	2205203
Alaa Hassan Melook	2205214
Yumna Medhat Anter	2205231
Rana Ashraf Abdelaziz	2205019
Mahmoud Amr Zaghloula	2205055
Abdelrahman Hisham Elmoghazy	2205032

Supervised by:

Dr. Essam Shabaan

Academic Year: 2025-2026

Abstract

Cybersecurity assessments in modern organizations often suffer from fragmented practices, where threat modeling, penetration testing, and continuous monitoring operate in isolation. This disconnection leads to inefficient risk management, redundant vulnerabilities, and delayed responses to evolving cyber threats.

This project proposes an intelligence-driven framework that integrates **Threat Modeling**, **Penetration Testing**, and **Threat Intelligence** to create a unified, risk-based security assessment process. By combining the predictive capabilities of threat modeling with the practical validation of penetration testing, the framework ensures that testing efforts are aligned with real business risks. The incorporation of threat intelligence enables adaptive and context-aware updates, allowing security assessments to evolve with the current threat landscape.

The proposed solution includes automated threat-to-test mapping, dynamic scope definition, centralized documentation, and continuous feedback loops for ongoing improvement. Expected outcomes include enhanced testing efficiency, better risk prioritization, and improved collaboration among security teams. Ultimately, this project aims to bridge the gap between design-time threat analysis and real-world defense validation, providing a modern, intelligence-driven approach to proactive cybersecurity management.

Acknowledgments

We would like to express our sincere gratitude to Dr. Essam Shabaan for his invaluable guidance and support throughout this project. His expertise and encouragement have been instrumental in shaping our research direction and achieving our objectives.

We would also like to extend our heartfelt thanks to Prof. Dr. Tamer Helmy, Dean of the Faculty, for his continuous support and for providing a motivating academic environment.

Our sincere appreciation goes to our academic advisor, Dr. Reem Essameldin Ebrahim, for her guidance, valuable feedback, and encouragement throughout the research process.

We are also grateful to the Faculty of Computers and Data Science at Alexandria National University for providing the necessary resources and environment for conducting this research.

Finally, we thank our families and colleagues for their unwavering support and motivation during the completion of this project.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
ATT&CK	Adversarial Tactics, Techniques & Common Knowledge
CAPEC	Common Attack Pattern Enumeration and Classification
CDN	Content Delivery Network
CI/CD	Continuous Integration/Continuous Deployment
CTI	Cyber Threat Intelligence
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DFD	Data Flow Diagram
DREAD	Damage, Reproducibility, Exploitability, Affected users, Discoverability
MISP	Malware Information Sharing Platform
ML	Machine Learning
OWASP	Open Web Application Security Project
SIEM	Security Information and Event Management
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
TIBSA	Threat-Intelligence Based Security Assessment
WAF	Web Application Firewall

Chapter 1

Introduction

Figure 1.1: Figure from document

1.1 Project proposal

1.1.1 Integration of penetration testing and threat modelling for continuous monitoring

1.2 Supervised by:

1.2.1 Dr. Essam Shabaan

Submitted by: Nadine Rasmy Soliman - 2205203
Alaa Hassan Melook - 2205214
Yumna Medhat Anter -2205231
Rana Ashraf Abdelaziz- 2205019
Mahmoud Amr Zaghloula - 2205055
Abdelrahman Hisham Elmoghazy - 2205032

1.3 Department:

1.3.1 Cybersecurity Department

1.4 Academic Year:

1.4.1 2025 / 2026

1.5 TABLE OF CONTENT

1.6 Abstract

Cybersecurity assessments in modern organizations often suffer from fragmented practices, where threat modeling, penetration testing, and continuous monitoring operate in isolation. This disconnection leads to inefficient risk management, redundant vulnerabilities, and delayed responses to evolving cyber threats.

This project proposes an intelligence-driven framework that integrates Threat Modeling, Penetration Testing, and Threat Intelligence to create a unified, risk-based security assessment process. By combining the predictive capabilities of threat modeling with the practical validation of penetration testing, the framework ensures that testing efforts are aligned with real business risks. The incorporation of threat intelligence enables adaptive and context-aware updates, allowing security assessments to evolve with the current threat landscape.

The proposed solution includes automated threat-to-test mapping, dynamic scope definition, centralized documentation, and continuous feedback loops for ongoing improvement. Expected outcomes include enhanced testing efficiency, better risk prioritization, and improved collaboration among security teams. Ultimately, this project aims to bridge the gap between design-time threat analysis and real-world defense validation, providing a modern, intelligence-driven approach to proactive cybersecurity management.

1.7 Introduction

This project focuses on integrating Threat Modeling with Penetration Testing to develop a more risk-driven and intelligence-oriented security assessment framework. By combining the predictive power of threat modeling with the practical validation of penetration testing, the proposed integration aims to help organizations prioritize real business risks, improve testing efficiency, and enhance their overall security posture.

Furthermore, the project incorporates Threat Intelligence into this integration to make the process more context-aware and adaptive. Threat intelligence provides insights into real-world attack patterns, emerging threats, and adversary tactics—allowing the threat model to evolve dynamically in line with the current threat landscape.

This unified approach ensures that both threat modeling and penetration testing are driven by up-to-date, intelligence-backed data, leading to more relevant, actionable, and continuous security assessments.

1.8 Problem statement

In many organizations, Threat Modeling, Penetration Testing, and Continuous Monitoring are performed as isolated processes, resulting in fragmented and inefficient security management.

Threat modeling identifies potential attack vectors but lacks real-world validation.

Penetration testing confirms vulnerabilities but does not leverage predictive threat insights.

Security assessments that exclude threat intelligence often fail to capture emerging attack trends and adversary behaviors.

This disconnection leads to delayed detection, misprioritized vulnerabilities, and repeated exposures—leaving systems vulnerable to preventable attacks.

Therefore, a unified framework is needed to integrate Threat Modeling, Penetration Testing, Threat Intelligence, and Continuous Monitoring, creating an adaptive and intelligence-driven security process.

1.9 Motivation

As cyberattacks become increasingly sophisticated and dynamic, traditional one-time assessments are insufficient to ensure ongoing protection.

Integrating threat intelligence with threat modeling and penetration testing improves both accuracy and timeliness in threat detection and validation.

1.9.1 This integration allows security teams to:

Focus on high-impact business risks.

Simulate real-world adversary tactics.

Build smarter and adaptive defense strategies.

For cybersecurity students and professionals, this project bridges the gap between predictive analysis and practical validation, providing valuable hands-on experience aligned with modern organizational security practices.

1.10 Project Objectives

Integrate Threat Modeling with Penetration Testing to create a risk-driven, efficient, and intelligence-informed security assessment process.

Focus testing on high-impact assets and critical attack vectors based on real business risks.

Translate threat scenarios into actionable penetration test cases for targeted validation.

Establish a continuous feedback loop where pentest results refine the threat model.

Strengthen the organization's security posture through collaboration and risk-based decision-making.

1.11 Expected Solutions

1- Threat-Driven Testing Framework

Penetration testing activities will be derived directly from the threat model, ensuring focused testing on high-risk and business-critical areas.

2- Automated Threat Mapping

Each identified threat will be automatically mapped to corresponding test cases, attack vectors, and mitigation strategies, allowing faster and more structured analysis.

3- Dynamic Scope Definition

The testing scope will dynamically adjust based on system updates and the evolving threat model—preventing scope creep while maintaining complete coverage of critical assets.

4- Centralized Documentation

All threat models, pentest results, vulnerabilities, and remediation actions will be stored in a unified repository to ensure traceability, auditability, and collaboration.

5- Risk Prioritization and Visualization

A visual risk scoring tool (e.g., heatmaps, risk matrices) will be incorporated to help stakeholders prioritize vulnerabilities based on impact, likelihood, and exploitability.

6- Continuous Feedback and Improvement Loop

Findings from penetration testing will feed back into the threat model, maintaining a continuous improvement cycle that adapts to new threats.

7- Collaboration and Role Integration

Encourages communication between developers, architects, and security testers through shared threat models, diagrams, and testing objectives—bridging the gap between design and defense.

8- Comprehensive Reporting

Generates integrated reports that link each vulnerability to:

Its originating threat scenario

The associated risk level

The recommended mitigation strategy

1.12 System Scope

The project aims to deliver a framework and process model (not a commercial platform) that demonstrates how Threat Modeling, Penetration Testing, and Threat Intelligence can be seamlessly integrated.

It focuses on:

Designing the integration workflow between these components.

Implementing sample use cases to validate the concept.

Providing risk visualization and documentation tools to support security decision-making.

1.13 Out of Scope

The following features and tools are not part of the main project scope, but may serve as potential extensions or future enhancements:

Development of a website or web-based service that provides analytical tools such as:

1.13.1 Malware Analyzer

1.13.2 URL Analysis

1.13.3 IP Analysis

1.13.4 Port Scanner

1.13.5 Hash Analyzer

1.13.6 Password Strength Checker

Dashboards that visualize:

Attack trends over time (e.g., via graphs)

Statistical data such as:

Most common malware types

Ratio of malicious vs. clean files

Total number of files analyzed

These functionalities represent a separate web application layer, which is outside the project's core research and implementation scope.

1.14 Expected Outcomes

A proof-of-concept framework that demonstrates how integrating threat modeling, penetration testing, and threat intelligence enhances organizational security.

Documentation and reports showing the relationship between modeled threats and verified vulnerabilities.

A risk-driven testing process adaptable to evolving cyber threats.

A collaborative workflow improving communication among security and development teams.

1.15 Conclusion

This project proposes a unified, intelligence-driven security assessment framework that bridges predictive analysis and practical validation. By integrating threat modeling, penetration testing, and threat intelligence, the approach empowers organizations to prioritize real risks, respond faster, and continuously strengthen their defenses in an ever-evolving threat landscape.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews existing research and methodologies related to threat modeling, penetration testing, and predictive security approaches. The literature is organized into three key areas: (1) traditional threat modeling frameworks, (2) AI and automation-based methodologies, and (3) integration of threat modeling with penetration testing.

2.2 Foundational Threat Modeling Approaches

Xu et al. introduced an automated security test generation approach based on formal threat models using Predicate/Transition (PrT) nets. The framework achieved about 95% executable attack paths and a 90% mutant kill rate on Magento and FileZilla Server, demonstrating high test effectiveness [?].

Marback et al. proposed integrating Data Flow Diagrams (DFDs) with STRIDE methodology to identify threats and transform them into executable test cases. The approach improved test coverage and detected multi-step attacks on real web applications [?].

Palanivel and Selvadurai presented a risk-driven testing approach integrating Extended Finite State Machines (EFSMs) with STRIDE-based threat modeling. The methodology reduced test cases by around 20% without compromising coverage [?].

Rahim et al. applied STRIDE-DREAD-based threat modeling to Water Grid Systems (WGS) using the Microsoft Threat Modeling Tool. They identified 154 potential threats with highest risks in Tampering (score 14), DoS (score 13), and Repudiation (score 12) [?].

Alozie examined threat modeling frameworks (PASTA, STRIDE, Attack Trees) for healthcare cybersecurity, recommending STRIDE-based models with automation and ML support for improved healthcare system resilience [?].

2.3 Automated and AI-Enhanced Threat Modeling

Granata and Rak conducted a systematic analysis of open-source automated threat modeling tools. Microsoft's tool demonstrated the highest automation (88 threats), OWASP Dragon identified 31 threats, and PyTM detected 91 detailed threats using CAPEC and MITRE data [?].

Shin et al. proposed the Actionable Intelligence-Oriented Cyber Threat Modeling Framework, implementing the TIME prototype to automate correlation of assets, vulnerabilities, and external CTI data using NIST SCAP standards [?].

Dekker and Alevizos developed TIBSA (Threat-Intelligence Based Security Assessment), a six-step methodology integrating CTI and causal graph modeling to assess cyber risks while minimizing uncertainty and bias [?].

Smith et al. proposed Predictive Behavioral Mapping (PBM) framework for ransomware detection. PBM achieved 98.6% detection accuracy and 1.8% false positive rate, significantly outperforming conventional approaches [?].

Bin Sarhan and Altwaijry presented a machine learning-based insider threat detection framework using Deep Feature Synthesis (DFS), PCA, and SMOTE on the CERT r4.2 dataset. The SVM classification model achieved 100% accuracy [?].

2.4 Integration of Threat Modeling with Penetration Testing

Alharbi et al. evaluated the security of a Siemens SICAM CMIC Remote Terminal Unit (RTU) by integrating formal threat modeling with practical black-box penetration testing using Nmap, OWASP ZAP, Nikto, Nexpose, and SQLMap [?].

Chu discussed automation of penetration testing using Belief-Desire-Intention (BDI) architecture with ontology-based reasoning. The automated approach reduced execution time from 179 seconds to 52 seconds on Metasploitable2 [?].

Huang and Zhu proposed PenHeal, a two-stage LLM framework for automated penetration testing. PenHeal improved vulnerability detection coverage by 31%, remediation effectiveness by 32%, and reduced remediation costs by 46% [?].

Chauhan analyzed penetration testing's impact on mitigating insider threats through Systematic Literature Review (SLR) methodology across IEEE Xplore, ScienceDirect, SpringerLink, ACM Digital Library, and Google Scholar [?].

Sanagana integrated threat modeling with Next-Generation Firewall (NGFW) architectures, proposing a proactive framework combining DPI, IPS, and Application Awareness with threat intelligence feeds [?].

Maniraj et al. showcased OWASP ZAP for web application security testing. Experimental results showed 88-94% coverage, identifying 120-150 vulnerabilities including 20-35 critical ones in 25-35 minutes [?].

2.5 Research Gap

The reviewed literature demonstrates significant advancements in integrating threat modeling with automated security testing, penetration testing, and AI-driven defense mechanisms. While approaches such as PrT-net-based automation, STRIDE-DFD threat modeling, and risk-driven EFSM testing improved accuracy and efficiency, they often required high modeling expertise and faced scalability challenges. AI-enhanced and CTI-integrated frameworks like TIBSA, TIME, and PenHeal achieved automation and predictive intelligence but remained limited by data quality, computational overhead, and domain specificity. Tools like OWASP ZAP and NGFW integrations showed strong practical applicability but suffered from false positives and limited coverage of complex attack vectors. These findings reveal the need for a unified, intelligent framework that bridges automated threat modeling, AI-based prediction, and dynamic penetration testing.

2.6 Available Solutions Analysis

2.6.1 Commercial Solutions

Microsoft Threat Modeling Tool: Automates threat identification using DFDs and STRIDE framework with intuitive graphical interface but operates primarily as standalone design-time tool with limited integration for continuous monitoring.

Cigent Platform: Combines automated threat discovery with risk quantification, integrating with existing security tools but lacking native penetration testing capabilities.

ThreatModeler: Automates threat modeling across SDLC supporting STRIDE, PASTA, and OCTAVE methodologies with CI/CD integration but limited penetration testing beyond basic vulnerability scanning.

IriusRisk: Provides automated threat modeling with DevSecOps focus, extensive threat pattern library, and API-based integrations but no native penetration test execution.

2.6.2 Open-Source Solutions

OWASP Threat Dragon: Free tool supporting STRIDE-based threat identification through visual DFD modeling but lacking automated testing capabilities.

PyTM: Code-based framework allowing programmatic architecture definition with CAPEC and MITRE ATT&CK databases but remaining primarily modeling tool.

Threagile: Risk-centric tool using YAML-based architecture definitions with quantitative risk scoring but no active testing capabilities.

OWASP ZAP: Extensively used for web application security testing with automated vulnerability scanning but lacking threat modeling capabilities.

Metasploit Framework: Comprehensive penetration testing framework with vast exploit collection but no threat modeling or risk prioritization mechanisms.

2.6.3 Gap Analysis

Critical gaps identified include: Integration Gap (disconnect between threat identification and validation), Automation Gap (lack of end-to-end continuous validation), Intelligence Gap (limited AI/ML for predictive analysis), Continuous Monitoring Gap (point-in-time assessments), Unified Risk Context Gap (fragmented results), and Feedback Loop Gap (limited capability for results to update models).

2.7 Tools Background

2.7.1 Threat Modeling Frameworks

STRIDE: Microsoft's mnemonic-based classification system covering Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege.

DREAD: Risk assessment model scoring threats on Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability.

PASTA: Seven-stage risk-centric methodology integrating business objectives with technical security analysis.

MITRE ATT&CK: Globally-accessible knowledge base of adversary behaviors organizing attack techniques into tactical categories.

CAPEC: Comprehensive dictionary of known attack patterns maintained by MITRE.

2.7.2 Penetration Testing Tools

Metasploit: World's most widely used framework with over 2,000 exploits and 500 payloads.

Nmap: Industry-standard tool for network discovery supporting various scanning techniques.

OWASP ZAP: Integrated tool acting as man-in-the-middle proxy for web application testing.

Burp Suite: Comprehensive platform combining manual and automated testing capabilities.

Nikto: Open-source web server scanner checking over 6,700 potentially dangerous files.

SQLMap: Automated SQL injection detection tool supporting wide range of database systems.

Hydra: Fast login cracker supporting numerous protocols for authentication testing.

2.7.3 Vulnerability Assessment Tools

Nessus: Widely deployed scanner with extensive database (over 165,000 plugins) covering CVEs, configuration issues, and compliance violations.

OpenVAS: Open-source framework with continuously updated feed of Network Vulnerability Tests.

Nexpose: Enterprise-grade solution combining comprehensive scanning with risk-based prioritization.

2.7.4 Integration Frameworks

PTES: Comprehensive framework defining standard methodology for penetration testing engagements with seven phases.

OWASP Testing Guide: Resource for web application security testing maintained by OWASP community.

Jenkins: Open-source automation server for CI/CD pipelines enabling automated security testing.

2.7.5 AI and ML Tools

TensorFlow and PyTorch: Leading frameworks for developing ML models for anomaly detection and threat prediction.

Scikit-learn: Python library for data mining and machine learning including classification algorithms.

NLP Tools: Tools like spaCy and transformers for automated analysis of security documentation.

2.7.6 Threat Intelligence Platforms

MISP: Open-source platform for sharing, storing, and correlating Indicators of Compromise.

STIX/TAXII: OASIS standards for representing and exchanging cyber threat intelligence.

AlienVault OTX: Community-driven platform providing access to millions of threat indicators.

2.7.7 Virtualization Technologies

Docker: Containerization platform for creating consistent, reproducible testing environments.

Kubernetes: Container orchestration platform for managing large-scale testing operations.

VirtualBox and VMware: Virtualization platforms for isolated testing environments.

2.8 Summary

The tools and frameworks reviewed represent current state-of-the-art in threat modeling, penetration testing, and security automation. Each tool addresses specific aspects yet typically operates independently. Understanding their capabilities and limitations provides foundation for designing integrated framework leveraging their strengths while addressing identified gaps.

Chapter 3

Requirements Analysis

3.1 Chapter Overview

Project Title: Integration of Penetration Testing and Threat Modeling for Continuous Monitoring

This chapter presents a comprehensive requirements analysis for the proposed system, integrating and refining the business, user, and system requirements in alignment with the project proposal and the research gaps identified in Chapter 2. The chapter follows an IEEE-style structure and adopts an Agile'€"Scrum development methodology to support continuous adaptation to evolving cybersecurity threats.

3.1 Business Requirements Identification

The business requirements for this graduation project were identified through an iterative and collaborative process involving academic supervisors, cybersecurity domain knowledge, and continuous team discussions. These requirements are directly derived from the problem statement, project objectives, expected solutions, and the research gaps highlighted in Chapter 2. Current cybersecurity practices often suffer from fragmentation, where threat modeling remains largely theoretical, penetration testing is reactive and unguided by predictive risk analysis, and continuous monitoring lacks adaptability to emerging threats. This project addresses these limitations by proposing a unified, intelligence-driven framework that integrates threat modeling, penetration testing, threat intelligence, and continuous monitoring.

The key business requirements are summarized as follows:

Seamless Integration of Security Processes: The system shall unify threat modeling (e.g., STRIDE and DFD), penetration testing (e.g., Nmap, OWASP ZAP, Metasploit), threat intelligence (e.g., AlienVault OTX, MISP), and continuous monitoring within a single cohesive workflow. This requirement directly addresses the integration gap identified in Chapter 2, where existing tools lack end-to-end validation capabilities.

Risk-Driven Prioritization: The system shall prioritize testing activities based on quantified business risk using scoring models such as DREAD and CVSS. This ensures focus on high-impact assets and attack vectors, improving efficiency without sacrificing coverage, as supported by risk-driven approaches discussed in the literature.

Enhanced Collaboration and Traceability: The system shall support collaboration among developers, security analysts, penetration testers, and architects through centralized repositories, shared visualizations, and traceable artifacts. This requirement mitigates the unified risk context gap and supports compliance with standards such as ISO 27005 and NIST CSF.

Continuous Adaptation and Improvement: The system shall incorporate automated feedback loops where penetration testing results and threat intelligence updates dynamically refine threat models. This addresses the continuous monitoring and feedback loop gaps identified in prior research.

Automation and Scalability: The system shall automate threat-to-test mapping, dynamic scope definition, and report generation to reduce manual effort and support scalability in complex environments, such as large-scale networks or cyber-physical systems.

Comprehensive Reporting and Compliance: The system shall generate actionable reports linking vulnerabilities to threat scenarios, risk levels, exploitability, and mitigation strategies, supporting regulatory compliance (e.g., GDPR, HIPAA) and informed decision-making.

These business requirements ensure that the proposed framework bridges the gap between predictive security analysis and practical validation while remaining feasible within the constraints of an academic graduation project.

3.1.1 Software Development Methodology The Agile software development methodology, implemented through the Scrum framework, was adopted for this project due to the dynamic and exploratory nature of cybersecurity research. Agile emphasizes flexibility, iterative development, and continuous feedback, making it particularly suitable for systems that must evolve alongside emerging threats and changing requirements.

3.1.2 Justification for Scrum Selection Scrum was chosen for this project because it effectively addresses the dynamic and uncertain nature of cybersecurity development. Cyber threats evolve rapidly, and system requirements cannot be fully defined upfront; Scrum'€™s short, time-boxed Sprints enable rapid adaptation to new threat intelligence or testing outcomes. The methodology supports incremental value delivery, allowing the project to produce functional components gradually, which facilitates early validation and refinement. Scrum also enhances collaboration and visibility through ceremonies such as Daily Stand-ups, Sprint Reviews, and Retrospectives, ensuring continuous alignment among team members. Furthermore, continuous backlog prioritization and retrospective analysis help identify and mitigate technical and design risks early. Finally, Scrum'€™s lightweight structure aligns well with academic constraints, supporting effective progress tracking and delivery within limited time and resources.

3.1.3 Scrum-Based Development Plan The project was implemented over multiple Sprints spanning approximately 15 weeks, as summarized below:

This iterative approach ensured continuous validation of assumptions, alignment with project objectives, and adaptability to emerging cybersecurity threats.

3.2 User Functional Requirements

The system targets multiple types of users, each with distinct responsibilities within the integrated threat modeling and penetration testing framework. Defining these roles ensures that functional requirements are aligned with user needs and access privileges.

3.2.1 User Roles

3.2.2 Functional Requirements User functional requirements (UFRs) define the interactions between end users and the system. Each requirement is aligned with one or more of the roles defined above. These requirements were specified following IEEE Std 830 principles and refined through Scrum backlog iterations.

This structured approach ensures clarity, traceability, and alignment of responsibilities with system features, improving usability, collaboration, and adaptability for all stakeholders.

3.3 System Functional and Non-Functional Requirements

This section defines the functional and non-functional requirements of the system, ensuring alignment with business and user needs. Functional requirements describe the system behavior and services, while non-functional requirements specify constraints and quality attributes.

3.3.1 System Functional Requirements

3.3.2 Non-Functional Requirements Non-functional requirements specify quality attributes and operational constraints of the system:

These non-functional requirements ensure the system'€™s robustness, adaptability, and user-centered design, complementing the functional capabilities described above.

3.4 System Evaluation and Validation Criteria

This section defines the evaluation and validation criteria derived directly from the identified business, user, and system requirements. These criteria establish a structured basis for assessing whether the proposed framework meets its intended objectives. In accordance with IEEE practices, the criteria are specified at a high level and are used later to guide system validation and performance evaluation, without presenting implementation results at this stage.

The evaluation criteria are grouped into functional effectiveness, performance efficiency, risk and security impact, and usability and collaboration.

3.4.1 Functional Validation Criteria

These criteria evaluate the system'€™s ability to fulfill its intended functional requirements:

Threat-to-Test Mapping Accuracy: The system shall correctly map identified threat scenarios to relevant penetration testing techniques and attack vectors.

Threat Coverage: The system shall ensure that all high-risk threats identified during threat modeling are addressed through corresponding penetration tests.

Traceability: The system shall maintain clear traceability between assets, threats, vulnerabilities, test cases, and mitigation strategies.

Automation Effectiveness: The system shall reduce reliance on manual configuration during threat analysis and penetration testing activities.

3.4.2 Performance and Efficiency Criteria

These criteria assess the operational efficiency of the proposed framework:

Time Efficiency: The system shall reduce the time required to plan and execute security assessments compared to traditional isolated approaches.

Responsiveness: The system shall update threat models and risk scores promptly following penetration testing activities or threat intelligence updates.

Scalability Support: The system shall handle increasing numbers of assets, threats, and test cases without significant degradation in performance.

3.4.3 Risk and Security Effectiveness Criteria

These criteria measure the system'€™s effectiveness in supporting risk-driven security decision-making:

Risk Prioritization Accuracy: The system shall correctly prioritize threats and vulnerabilities based on impact, likelihood, and exploitability.

Adaptive Risk Management: The system shall demonstrate the ability to adapt to emerging threats through continuous feedback and threat intelligence integration.

Improved Risk Visibility: The system shall provide clear and actionable visual representations of risk levels to support informed security decisions.

3.4.4 Usability and Collaboration Criteria

These criteria evaluate how effectively the system supports user interaction and teamwork:

Ease of Use: The system shall provide intuitive interfaces that allow users to perform core security tasks with minimal training.

Collaboration Support: The system shall facilitate effective collaboration through shared documentation, notifications, and role-based access control mechanisms.

Report Clarity: The system shall generate clear, structured, and understandable reports suitable for both technical and non-technical stakeholders.

3.5 Chapter Summary

This chapter presented an integrated requirements analysis aligned with the project proposal and research gaps. It justified the adoption of the Agile'€“Scrum methodology and defined the business, user, and system requirements that guide the system design and implementation discussed in the subsequent chapters.

Sprint	Duration	Key Activities / Deliverables
Sprint 1	Weeks 1’€“3	Requirements elicitation, initial system architecture design, STRIDE-based threat modeling, backlog prioritization
Sprint 2	Weeks 4’€“6	Core threat modeling and automated threat mapping to attack patterns (MITRE ATT&CK, CAPEC)
Sprint 3	Weeks 7’€“9	Penetration testing integration (OWASP ZAP, Nmap) and initial automation workflows
Sprint 4	Weeks 10’€“12	Continuous feedback loops implementation to update threat models and risk scores
Sprint 5	Weeks 13’€“15	Risk visualization dashboards, comprehensive reporting mechanisms, final system refinement

Table 3.1: Table from document

Role	Description / Responsibilities
Security Analyst	Creates and maintains threat models, analyzes risk scores, and monitors continuous threat intelligence updates.
Penetration Tester	Executes penetration testing activities, validates vulnerabilities, and provides feedback to update threat models.
Developer	Reviews identified vulnerabilities, implements fixes, and integrates mitigations into the system design.
System Architect	Ensures that system design aligns with security requirements, evaluates architectural risks, and reviews threat model scenarios.
Compliance Stakeholder	Reviews security assessment reports to ensure regulatory and organizational compliance, and monitors adherence to security policies.

Table 3.2: Table from document

UFR ID	Requirement	Primary User Role(s)
UFR-1	The system shall allow users to create, edit, and version-control threat models using STRIDE and DFD techniques.	Security Analyst, System Architect
UFR-2	The system shall automatically map identified threats to corresponding penetration testing techniques and tools.	Security Analyst, Penetration Tester
UFR-3	The system shall dynamically adjust testing scope based on system changes and emerging threats.	Penetration Tester, Security Analyst
UFR-4	The system shall provide a centralized repository for storing threat models, testing results, and mitigation actions.	All Roles
UFR-5	The system shall present interactive risk visualizations (e.g., heatmaps, matrices) to support prioritization.	Security Analyst, System Architect, Compliance Stakeholder
UFR-6	The system shall automatically update threat models based on penetration testing outcomes.	Security Analyst, Penetration Tester
UFR-7	The system shall support collaborative features such as shared access, notifications, and role-based permissions.	All Roles
UFR-8	The system shall generate comprehensive security assessment reports for technical and non-technical stakeholders.	Compliance Stakeholder, Security Analyst
UFR-9	The system shall	Security Analyst

SFR ID	Requirement	Primary User Role
SFR-1	The system shall provide a threat modeling engine supporting STRIDE-based classification and asset mapping.	Security Analyst, System Architect
SFR-2	The system shall automate penetration testing execution using integrated tools (e.g., OWASP ZAP, Nmap, Metasploit).	Penetration Tester
SFR-3	The system shall integrate external threat intelligence and correlate it with internal assets.	Security Analyst, System Architect
SFR-4	The system shall automate threat-to-test mapping and dynamic scoping.	Security Analyst, Penetration Tester
SFR-5	The system shall implement a continuous feedback loop to refine threat models and risk scores.	Security Analyst, Penetration Tester
SFR-6	The system shall generate visual dashboards and structured security reports for decision-making.	Security Analyst, System Architect, Compliance Stakeholder
SFR-7	The system shall maintain secure, traceable storage of all security artifacts.	All Roles
SFR-8	The system shall support virtualized or containerized testing environments.	Penetration Tester, Security Analyst

Table 3.4: Table from document

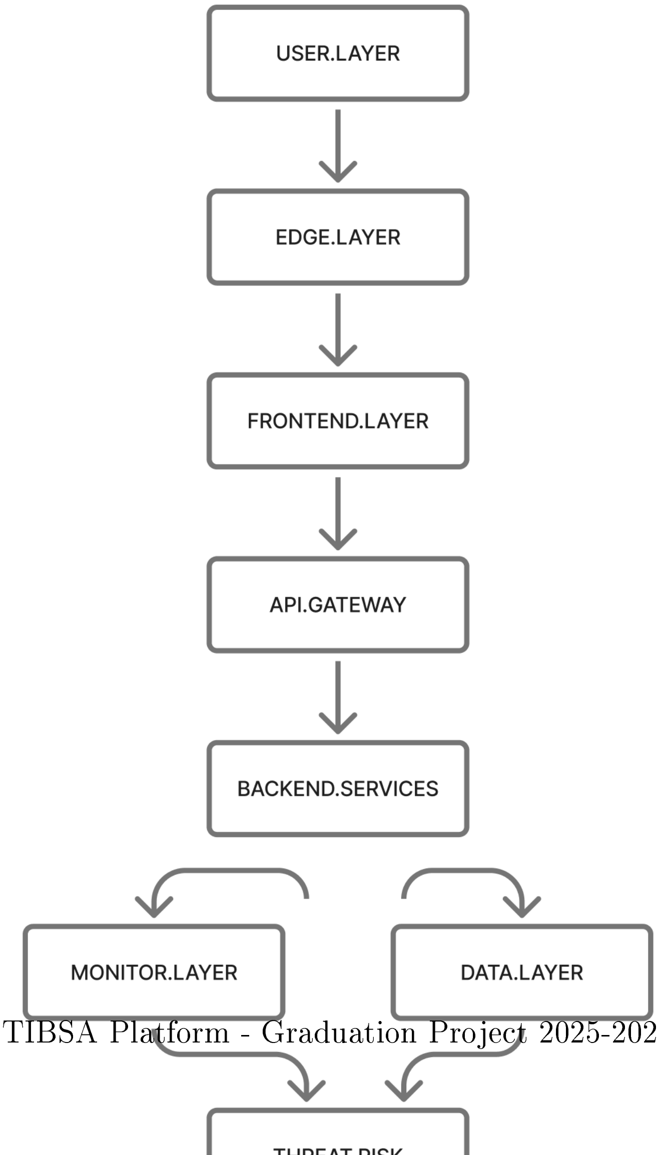
Attribute	Requirement
Performance	The system shall process threat analysis and testing results within acceptable response tim
Scalability	The system shall support growth in assets, threats, and testing tools without requiring ma
Reliability	The system shall ensure consistent operation, accurate data correlation, and availability of
Security	The system shall protect data confidentiality, integrity, and availability using encryption, a
Usability	The system shall provide intuitive interfaces, clear visualizations, and user-friendly navigat
Maintainability	The system shall support modular updates, integration of new intelligence sources, and effi
Portability	The system shall be deployable across multiple platforms and environments, including loca

Table 3.5: Table from document

Chapter 4

System Design

4.1 TIBSA Platform - Complete Architecture with Tools Integration



4.1.1 Table of Contents

4.1.2 LAYER 1 - User Layer

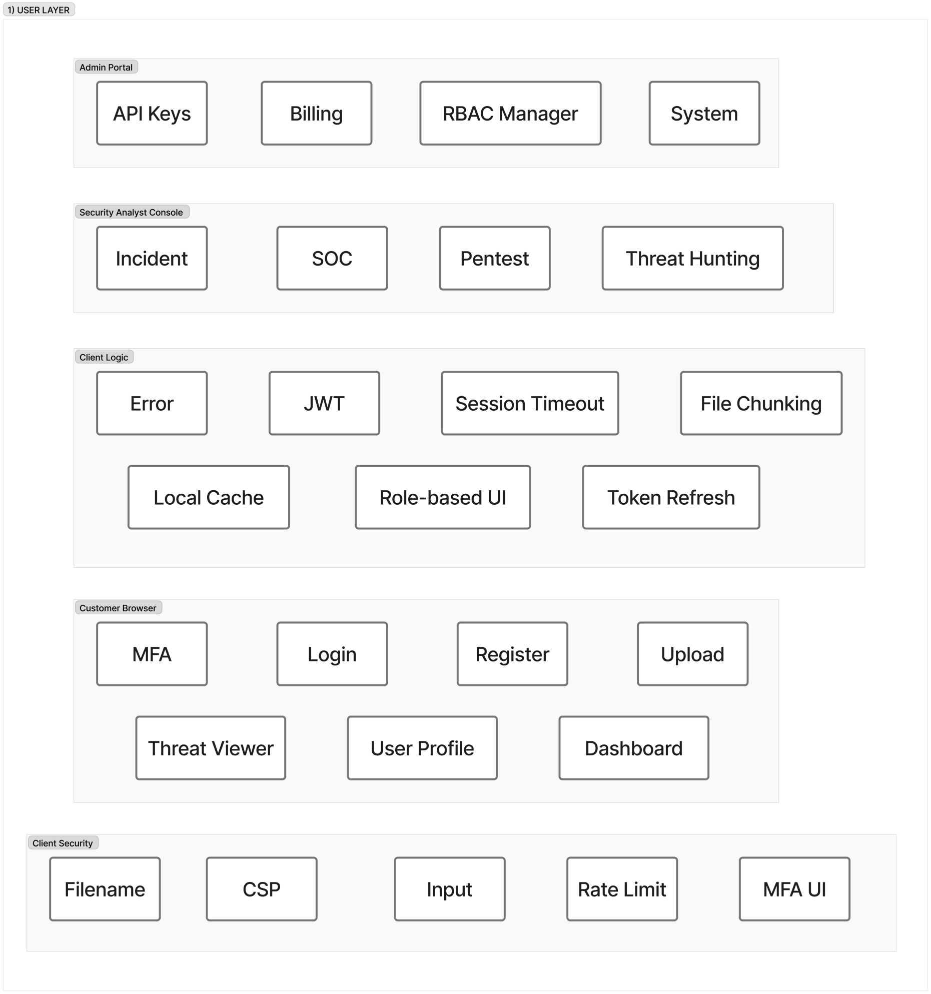


Figure 4.2: Figure from document

1.1 Security Analyst Console

The Security Analyst Console is the primary interface used by SOC analysts, threat hunters, and penetration testers. It provides access to dashboards, incident views, analytics tools, and pentest management interfaces. This layer uses no external tools, as it is a pure UI layer that serves as the entry point for security professionals.

The console provides several key services including a threat hunting interface, alert investigation dashboard, pentest workflow triggering capabilities, and a results review console. Analysts interact with the system through various actions such as queries, pentest runs, and incident views, all of which are authenticated using JWT tokens and tracked through UI interactions.

The data flow begins when the console receives analyst actions and JWT tokens as input, which are then processed into authenticated API requests, analysis queries, and pentest triggers as output. These requests are first sent to Frontend Security Controls in Layer 3 for input sanitization, then forwarded to the API Gateway in Layer 4 for authentication and routing. The console receives data back from Backend Services in Layer 5 via the API Gateway for display to the analyst.

1.2 Client Logic Module

The Client Logic Module serves as the browser-side brain of the platform, managing the complete lifecycle of user sessions and authentication. It operates entirely as client-side JavaScript logic without relying on external tools, handling critical functions that ensure smooth and secure user interactions.

This module provides comprehensive services including JWT token management, token refresh automation, session timeout handling, UI state management based on RBAC (Role-Based Access Control), API error handling, and file chunking preparation. It continuously monitors token expiration and automatically refreshes them to maintain uninterrupted user sessions.

The module accepts user interactions, JWT tokens, UI events, form data, and files as input, transforming these into structured requests, refreshed tokens, chunked uploads, and properly handled errors as output. It maintains session state and authentication tokens while communicating with the Auth Interceptor in Layer 3 to attach tokens to outgoing requests. All backend communication flows through the API Gateway in Layer 4, and the module receives new tokens from the Authentication Service in Layer 5 during refresh cycles.

1.3 Customer Browser Interfaces

The Customer Browser Interfaces provide all customer-facing web pages, offering a complete set of UI components for user interaction without requiring external tools. These interfaces encompass every aspect of the customer experience from initial registration through daily platform usage.

The services provided include login and registration pages, MFA (Multi-Factor Authentication) enrollment and verification UI, comprehensive dashboard views, an intuitive file upload interface, and scanning tool initiation forms. Each interface is designed with user experience and security in mind, ensuring that all customer actions are properly validated before reaching backend systems.

Customer actions such as login attempts, registration submissions, file uploads, and scan initiations serve as input to these interfaces, along with MFA inputs when required. The interfaces generate clean UI-validated requests, handle session initialization, and manage navigation events as output. All requests are first sent to Client-Side Security Controls in Layer 3 for validation, then forwarded to the API Gateway in Layer 4 via authenticated requests. Dynamic content is received from Backend Services in Layer 5 to populate dashboards and display results.

1.4 Client-Side Security Controls

The Client-Side Security Controls act as the first major security checkpoint in the system, implementing multiple layers of protection using browser-native security features and custom validators. These controls serve as the gatekeeper between user input and the backend infrastructure, ensuring that only clean, validated data proceeds downstream.

The security services provided include XSS (Cross-Site Scripting) prevention through comprehensive input sanitization, Content Security Policy (CSP) enforcement, filename validation to prevent directory traversal attacks, soft rate limiting to prevent abuse, and MFA flow protection to secure authentication processes. Each control operates transparently to users while maintaining strict security standards.

The controls receive outgoing requests, form fields, filenames, and script execution attempts from all User Layer components (Security Analyst Console, Client Logic Module, and Customer Browser Interfaces) as input. They produce sanitized inputs, enforce CSP rules, validate safe file uploads, and apply rate-limited requests as output. After validation, all data is sent to the Frontend Layer (Layer 3), while malicious payloads are blocked before they can reach the backend systems.

4.1.3 LAYER 2 - Edge Layer

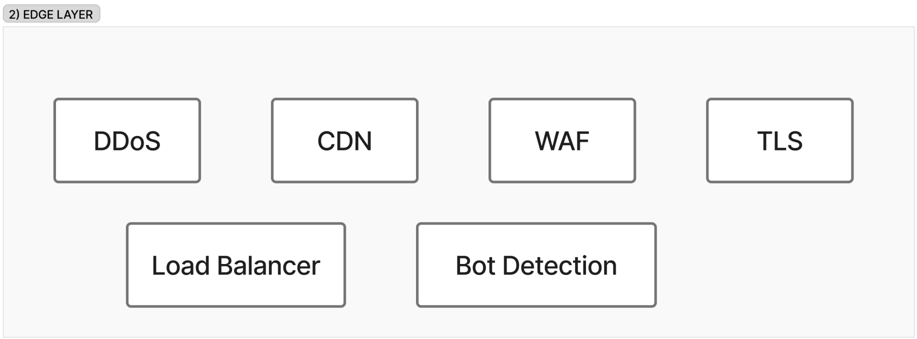


Figure 4.3: Figure from document

2.1 CDN (Content Delivery Network)

The CDN serves as the global distributed entry point for all traffic entering the platform, utilizing Cloudflare's Free Plan as Service #1. It functions as the first point of contact between users worldwide and the TIBSA platform, providing critical performance and availability improvements through intelligent caching and distribution.

The CDN provides several essential services including global content caching, static asset delivery for JavaScript, CSS, and images, nearest-node routing for low latency, basic request validation, and traffic absorption with load distribution. By caching content at edge locations around the world, the CDN dramatically reduces latency for users regardless of their geographic location.

Raw HTTP(S) requests from user browsers worldwide arrive at the CDN as input. When the CDN has cached content (a cache hit), it returns static responses directly to users, bypassing downstream systems entirely. For cache misses, the CDN forwards clean requests to the WAF in Layer 2.2 for application-layer inspection, then caches the response for future requests.

2.2 WAF (Web Application Firewall)

The WAF performs deep inspection of incoming requests by analyzing patterns associated with common web attacks, operating as part of Cloudflare's Free Plan (Service #1). It examines every request that passes through the CDN, looking for malicious patterns and known attack signatures before allowing traffic to proceed further into the infrastructure.

The WAF provides comprehensive protection including SQL Injection detection and blocking, Cross-Site Scripting (XSS) prevention, CSRF token validation, path traversal detection, malicious payload filtering, and CAPTCHA challenges for suspicious requests. Each of these protections operates in real-time, examining request headers, bodies, and parameters for signs of malicious intent.

After receiving CDN-filtered requests as input, the WAF produces three possible outputs: allowed requests are sent to DDoS Protection in Layer 2.3, blocked requests receive an error page, and suspicious requests are challenged with CAPTCHA or JavaScript verification. All blocked attempts are logged and sent to the SIEM analysis system in Layer 7 for security monitoring and threat intelligence purposes.

2.3 DDoS Protection

The DDoS Protection system identifies and mitigates abnormal traffic spikes and distributed denial-of-service attacks using Cloudflare's Free Plan (Service #1). It continuously monitors traffic patterns in real-time, detecting anomalies that could indicate an ongoing attack and taking immediate action to protect the platform's availability.

The system provides multi-layered protection including Layer 3, 4, and 7 DDoS attack mitigation, continuous traffic rate monitoring, anomaly detection using behavioral analysis, packet dropping for malicious traffic, and traffic normalization to ensure clean requests reach application servers. It can distinguish between legitimate traffic spikes (such as during peak usage) and malicious attack patterns.

WAF-validated requests arrive as input, and the system produces two types of output: normalized safe traffic is forwarded to TLS Termination in Layer 2.4, while abnormal or malicious packets are dropped immediately. The system protects all downstream services from volumetric attacks and continuously monitors traffic patterns, sending detailed metrics to the Monitoring Layer in Layer 7 for analysis and alerting.

2.4 TLS Termination

TLS Termination is the point where encrypted HTTPS traffic is securely decrypted, utilizing Cloudflare's TLS/SSL capabilities (Service #1). This critical security function validates certificates, ensures proper use of cryptographic standards, and offloads the computationally expensive encryption and decryption tasks from backend servers, allowing them to focus on application logic.

The system provides essential services including HTTPS traffic decryption, certificate validation, TLS 1.3 support for modern security standards, SSL/TLS offloading to improve backend performance, and secure session handling. It ensures that all traffic uses strong encryption in transit while making content accessible to downstream security inspection tools.

Encrypted HTTPS requests from DDoS Protection arrive as input, and the system validates SSL certificates and encryption standards before producing decrypted HTTP requests as output. These decrypted requests are then sent to the Load Balancer in Layer 2.5, enabling downstream systems to inspect and process the actual request content while maintaining end-to-end security.

2.5 Load Balancer

The Load Balancer distributes incoming requests across backend servers based on intelligent routing policies, using Cloudflare's Load Balancing feature (Service #1). It ensures optimal resource utilization and high availability by directing traffic to healthy servers and avoiding overloaded or failed instances.

The balancer provides comprehensive distribution services including request distribution across multiple servers, health check monitoring to detect server issues, round-robin routing for equal distribution, least connections algorithm to favor less-busy servers, weighted load distribution for capacity-based routing, and automatic failover handling when servers become unavailable. Each routing decision is made in real-time based on current system conditions.

Decrypted HTTP requests from TLS Termination serve as input, and the balancer monitors server health continuously to adjust routing accordingly. Output consists of properly routed requests sent to correct backend service instances based on health checks and load conditions. Before reaching backend services, requests pass through the Bot Detection Engine in Layer 2.6 for final edge validation.

2.6 Bot Detection Engine

The Bot Detection Engine analyzes traffic behavior, device fingerprints, user interaction patterns, request frequency, and known malicious IP ranges to distinguish human users from automated bots, utilizing Cloudflare's Bot Management feature (Service #1). This sophisticated analysis protects the platform from automated attacks, scraping, and abuse while allowing legitimate automation when appropriate.

The engine provides multiple security services including bot versus human traffic distinction, device fingerprinting to identify returning visitors and potential threats, behavioral analysis to detect automation patterns, request pattern detection to identify suspicious activity, malicious IP blocking, automated scraping prevention, and credential stuffing protection. These capabilities work together to create a comprehensive defense against bot-based threats.

Routed requests from the Load Balancer arrive as input along with metadata including IP addresses, user agents, and behavioral patterns. The engine produces two types of output: human-verified requests are forwarded to the Frontend Layer in Layer 3, while blocked or challenged bots receive either a challenge page requiring human verification or are blocked outright. As the final security check at the edge before traffic reaches the application layer, this engine provides critical protection against automated attacks and scraping attempts.

4.1.4 LAYER 3 - Frontend Layer

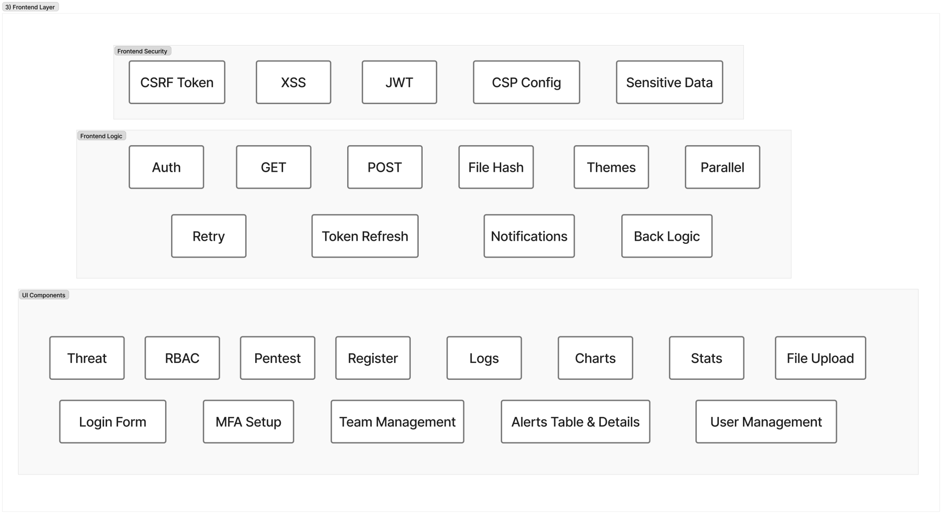


Figure 4.4: Figure from document

3.1 Frontend Framework & UI Components

The Frontend Framework provides all user interfaces, dashboards, and interactive components for the entire platform using Next.js 15, Tailwind CSS, and shadcn/ui as Service #2. This modern technology stack enables server-side rendering, static site generation, and dynamic client interactions while maintaining excellent performance and developer experience.

The framework provides comprehensive services including server-side rendering (SSR) for improved initial load times, static site generation (SSG) for cacheable pages, API routes for backend communication, complete dashboard interfaces for all user types, an intuitive file upload UI, threat modeling interfaces, authentication pages covering login, registration, and MFA, alerts and threat visualization tools, user management interfaces, a guided pentest wizard, and a comprehensive reports viewer.

User interactions, API responses, and routing requests serve as input to the framework, which produces rendered HTML and React components along with API calls to backend services as output. The framework consists of several specialized components: Login Form, Register Component, and MFA Setup UI handle user authentication by accepting email, password, and MFA codes as input and producing JWT access tokens and MFA confirmation as output, sending requests through the Auth Interceptor to the API Gateway and ultimately to Authentik in Layer 4.

Stats Widgets provide KPI visualization, scan statistics, and vulnerability metrics by receiving analytics API responses from backend services and displaying them as charts, counters, and time-series graphs. The Threat Map component offers geographic and topological threat visualization by connecting alert metadata and geo-location data to create an interactive threat map, integrating with the Alerts Table and MISP threat intelligence from Layer 5.

The Alerts Table and Alert Details Modal provide structured alert listing and detailed incident views, receiving data from the SIEM in Layer 7 and Notification Service in Layer 5 to display paginated alerts and drill-down details with evidence. The File Upload Component manages file submission with validation, using the File Hash Calculator and Chunk Uploader to initiate chunked uploads with progress tracking, sending data to the File Upload Handler in Layer 3.2.

The Pentest Wizard guides users through penetration test configuration, accepting scope, targets, scan depth, and schedule as input and producing validated job configurations that are sent to the Pentest Orchestrator in Layer 5. The Logs Console enables real-time log viewing and searching by receiving streamed logs from Loki in Layer 7 and rendering them with search capabilities and downloadable exports.

User Management UI, Team Management UI, and RBAC Matrix components handle user creation, role assignment, and permission management, accepting user data and role definitions as input and sending updates to the User Service in Layer 5 via the API Gateway. The framework receives all traffic from the Edge Layer after bot detection, communicates with the API Gateway in Layer 4 for all backend operations, renders dynamic content based on user roles and permissions, and sends large file operations to the File Upload Handler.

3.2 Frontend Security Controls

Frontend Security Controls implement browser-side security mechanisms that protect UI and client-side data using built-in browser security features combined with custom validators. These controls operate transparently to provide multiple layers of protection without impacting user experience.

The XSS Sanitizer provides input sanitization and script tag removal, accepting raw user text and HTML input from forms and producing sanitized, safe HTML strings that are sent to UI Components and the GET/POST Helpers. The JWT Validator performs client-side token validation by accepting JWTs from storage and producing validation status along with extracted claims including user ID, role, and expiry information, which are then used by RBAC Logic and the Auth Interceptor.

The CSRF Token Handler protects against cross-site request forgery by accepting CSRF tokens from the backend and attaching them as headers to all outgoing POST, PUT, and DELETE requests before sending them through the Auth Interceptor to the API Gateway. The CSP Config Loader enforces Content Security Policy by accepting CSP configuration from the server and applying CSP rules that are enforced by the browser, protecting against XSS and malicious script injection.

Sensitive Data Masking provides PII and sensitive data redaction by accepting raw scan results, alert details, and sensitive fields and producing masked values (such as "******1234") that are safely displayed in UI Components. These security controls protect all outgoing requests from the Frontend Layer, work closely with the Frontend Logic Layer to ensure secure communication, and send validated data to the API Gateway in Layer 4.

3.3 Frontend Logic Layer

The Frontend Logic Layer handles all client-side business logic, request preparation, and state management using custom JavaScript and TypeScript logic. This layer orchestrates the complex interactions between user interface components, security controls, and backend services.

The GET/POST Helpers provide standardized HTTP request handling, accepting API endpoints, URL parameters, body payloads, and headers as input and producing parsed JSON responses or standardized errors as output, which are then sent to UI Components and the Notifications Queue. The Auth Interceptor performs automatic authentication header attachment by accepting requests from GET/POST helpers along with access tokens and CSRF tokens, producing fully authenticated requests with Authorization and CSRF headers that are sent to the API Gateway in Layer 4, while also triggering token refresh on 401 or 403 errors.

Token Refresh Logic handles automatic token renewal by accepting expiring or expired tokens along with refresh tokens as input and producing new access tokens that are stored in browser storage as output. This component communicates with Authentik in Layer 4 for token refresh and updates the Auth Interceptor with new tokens to maintain uninterrupted sessions.

Retry Logic provides network failure recovery with exponential backoff, accepting failed requests and error details such as timeouts or 5xx errors as input and producing either successful responses or final errors that are sent to the Notifications Queue. This is particularly critical for large file uploads and long-running pentest calls where transient network issues should not cause complete failure.

RBAC Logic implements client-side role-based access control by accepting JWT claims and the RBAC matrix as input and producing visibility and permission flags for UI elements as output, controlling which UI components are shown or hidden per user role. It's important to note that while this provides immediate UI feedback, server-side validation is still enforced in backend services for security.

The File Hash Calculator computes SHA-256 hashes for integrity by accepting binary or text files as input and producing unique file hashes as output, which are sent to the backend as metadata and to the Chunk Uploader for integrity checks. This prevents duplicate scans and ensures forensic consistency across the platform.

Parallel Upload and Chunk Uploader manage large file chunking and parallel upload by accepting file objects, chunk size, upload URL, and session ID as input and producing chunk upload requests, progress updates, and final file IDs as output. This component works with Retry Logic for failed chunks and sends data to the Tusd Server in Layer 3.4, which stores files in Cloudflare R2 in Layer 6.

The Notifications Queue manages real-time notification handling by accepting backend events through polling or WebSockets along with frontend actions as input and producing UI notifications, alerts to the Alerts Table, and log events to the Logs Console as output. It displays scan completions, vulnerabilities, report readiness, and system warnings to keep users informed of important events.

Theme Manager controls visual theme preferences (light and dark mode) by accepting user preferences, system settings, and CSS definitions as input and producing applied CSS variables and dynamic styling as output, improving readability for dashboards and threat visualizations.

The Frontend Logic Layer receives data from Frontend Security Controls in Layer 3.2 after validation, sends requests to the API Gateway in Layer 4 via the Auth Interceptor, and manages all state and logic before backend communication occurs.

3.4 File Upload Handler

The File Upload Handler manages resumable, chunked uploads of large files up to 10 GB with integrity checking and resume support, utilizing Uppy and Tusd (open source tus.io server) as Service #8. This sophisticated upload system ensures that even very large files can be uploaded reliably over unreliable networks.

The handler provides essential services including resumable file uploads that can be paused and continued, chunked upload that splits large files into manageable pieces, upload progress tracking for user feedback, automatic retry on failure without losing progress, session persistence to resume interrupted uploads across browser sessions, support for files up to 10 GB in size, and integrity verification to ensure files are not corrupted during transmission.

File objects from the File Upload Component in Layer 3.1.5 serve as input along with chunk size, upload URL, and session ID. The handler produces several outputs including chunked file segments that are uploaded to the backend, progress updates to the UI to keep users informed, a final file ID upon completion, and upload confirmation.

The handler receives file hashes from the File Upload Component, uses the Chunk Uploader in Layer 3.3.7 for parallel upload management, and sends data to the Tusd Server backend component which stores files in Cloudflare R2 in Layer 6. Once upload is complete, the file is passed to the Scan Service in Layer 5 for processing, and the

Notifications Queue in Layer 3.3.8 is notified of completion or failure.

Integration points include working with the File Hash Calculator in Layer 3.3.6 for deduplication to avoid re-scanning identical files, coordinating with Retry Logic in Layer 3.3.4 for failed chunk recovery, storing final files in Cloudflare R2 in Layer 6 via the Tusd server, and triggering the backend processing pipeline that flows through Scan Service, Sandbox, and ML Engine.

4.1.5 LAYER 4 - API Engine

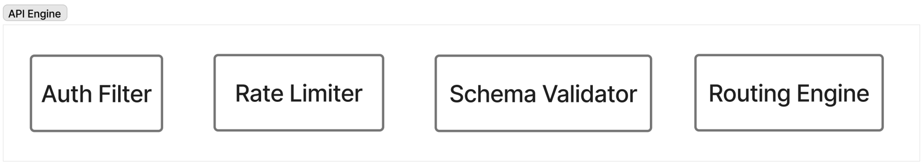


Figure 4.5: Figure from document

4.1 API Gateway

The API Gateway serves as the central entry point for all API requests using KrakenD Community Edition as Service #4, providing request validation, authentication, rate limiting, and intelligent routing to backend microservices. This critical component ensures that only valid, authenticated, and properly formatted requests reach backend services.

The gateway provides comprehensive services including serving as the central API entry point for all client requests, request routing to appropriate microservices, rate limiting enforcement to prevent abuse, request and response transformation to adapt between frontend and backend formats, API composition for aggregating multiple backend calls, circuit breaker pattern implementation to prevent cascade failures, and thorough request validation.

The gateway consists of several specialized components: The Auth Filter performs JWT token validation and permission checking by accepting HTTP requests with Authorization headers as input and producing authenticated request context or 401/403 errors as output. It validates tokens with Authentik in Layer 4.2 and sends valid requests to the Rate Limiter component.

The Rate Limiter controls request frequency and prevents DoS attacks by accepting request metadata including IP addresses, user IDs, and endpoint information as input. It uses Upstash Redis in Layer 6 for rate limit storage and produces either allowed requests that proceed to the Schema Validator or 429 Too Many Requests responses when limits are exceeded.

The Schema Validator ensures request body validity against JSON schemas by accepting JSON and FormData payloads as input and producing validated, sanitized data or 400 Bad Request errors as output. This ensures that backend services receive only correct, safe input and sends validated requests to the Routing Engine.

The Routing Engine performs intelligent request routing to appropriate microservices by accepting validated, authenticated requests with URL and HTTP method information as input and producing routed requests to target backend services as output. It can route to Authentication Service in Layer 4.2, User Service, Scan Service, Threat Intelligence Service, Billing Service, Notification Service, ML Engine, Sandbox, and Threat Modeling Engine, all in Layer 5.

The gateway receives requests from the Frontend Layer in Layer 3.3 via the Auth Interceptor, first checks the Auth Filter for token validation, then checks the Rate Limiter using Redis cache, validates request schema and payload, routes to appropriate backend services in Layer 5, and returns aggregated responses to the frontend.

4.2 Authentication & Authorization Service

The Authentication & Authorization Service manages user authentication flows including login, registration, token refresh, and multi-factor verification using Authentik, an open-source identity platform, as Service #3. This service is the cornerstone of platform security, ensuring that only authorized users can access protected resources.

The service provides extensive capabilities including user login and registration, Multi-Factor Authentication with TOTP and WebAuthn, comprehensive Role-Based Access Control (RBAC), user directory management, Single Sign-On (SSO) support, OAuth2 and OIDC provider functionality, LDAP and Active Directory integration, session management, token issuance and validation, and password reset flows.

The login function accepts email and password as input and produces JWT access tokens and refresh tokens as output. The process validates credentials, checks MFA status, and issues tokens that are sent to the frontend via the API Gateway. The register function accepts email, password, and user details as input and produces user account creation confirmation as output, validating input, creating users in the directory, and triggering verification emails via the Notification Service in Layer 5.

The refreshToken function accepts valid refresh tokens as input and produces new access tokens as output, validating refresh tokens and issuing new access tokens. This function is called by Token Refresh Logic in Layer 3.3.3 when access tokens expire. The verifyMFA function accepts user sessions and MFA codes (TOTP or WebAuthn) as input and produces MFA verification results and session upgrades as output, validating the second factor and upgrading session privileges for administrative actions and sensitive operations.

The service integrates with the API Gateway Auth Filter in Layer 4.1.1 for token validation, stores user credentials and sessions in Neon Postgres in Layer 6, uses Redis in Layer 6 for session caching, sends notifications via the Notification Service in Layer 5, enforces RBAC policies for all backend services, and provides tokens to all authenticated frontend requests.

RBAC integration includes defining roles such as Admin, Analyst, User, and Guest, managing permissions for scanning, viewing reports, managing users, and configuring the system, enforcing permission checks in backend services, and integrating with the User Management UI in Layer 3.1.8 for role assignment.

4.1.6 LAYER 5 - Backend Services

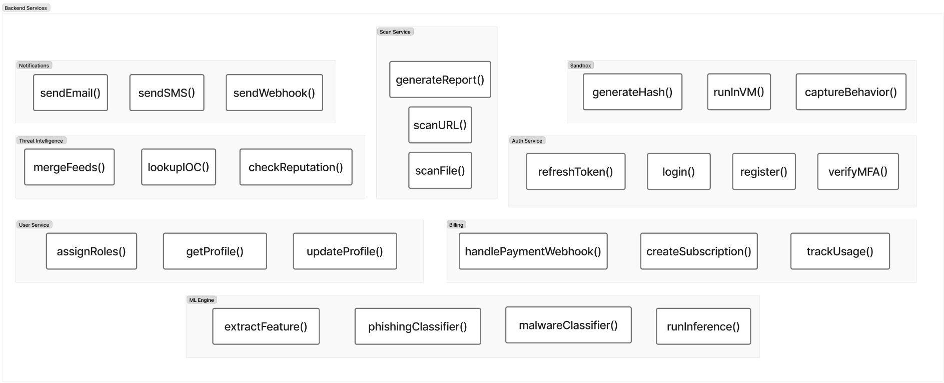


Figure 4.6: Figure from document

5.1 User Service

The User Service manages account-level functionality, user profiles, and role assignments using a custom Node.js or Python microservice. This service is responsible for all user-related operations beyond authentication, providing the foundation for personalized user experiences and access control.

The service provides comprehensive functionality including user profile management, account settings updates, role assignment and updates, user directory operations, and permission mapping. Each of these capabilities ensures that users can manage their accounts while administrators maintain proper access control.

The getProfile function accepts user ID from JWT as input and produces user profile data including name, email, role, and preferences as output, reading this information from Neon Postgres in Layer 6. The updateProfile function accepts user ID and updated profile data as input and produces update confirmation as output, writing changes to Neon Postgres and triggering audit log entries.

The assignRoles function accepts user ID and role list as input and produces role assignment confirmation as output. This function updates Authentik RBAC in Layer 4 and Postgres in Layer 6, but requires admin privileges to execute, ensuring that only authorized personnel can modify user permissions.

The service receives requests from the API Gateway in Layer 4, is authenticated by Authentik in Layer 4, stores data in Neon Postgres in Layer 6, logs all actions in Audit Logs Storage in Layer 6 for compliance and security monitoring, and returns data to the User Management UI in Layer 3.

5.2 Scan Service

The Scan Service coordinates security scanning operations for URLs and files using a custom orchestration service, managing the complex workflow of analyzing potentially malicious content through multiple analysis engines. This service is central to the platform's security analysis capabilities.

The service provides critical functionality including URL scanning for phishing and malware, file malware detection, report generation, and scan result aggregation. It orchestrates multiple specialized tools and services to provide comprehensive security analysis.

The scanURL function accepts URLs to scan as input and produces scan results including malicious indicators, reputation scores, and threat levels as output. The process checks URLs against Threat Intelligence in Layer 5.3, queries external feeds such as VirusTotal and URLhaus, runs ML classification through the ML Engine in Layer 5.6, and aggregates all results before storing them in the Scan Results Database in Layer 6.

The scanFile function accepts file IDs from R2 storage as input and produces comprehensive file analysis reports as output. The process retrieves files from Cloudflare R2 in Layer 6, sends them to CAPE Sandbox in Layer 5.7 for dynamic analysis, extracts features for the ML Engine in Layer 5.6, queries Threat Intelligence in Layer 5.3, aggregates all results, and stores them in the Scan Results Database in Layer 6.

The generateReport function accepts scan IDs as input and produces formatted security reports in JSON or PDF format as output. The process aggregates all scan data, formats the report according to templates, stores the report in R2, and returns the report URL to users.

The service receives scan requests from the API Gateway in Layer 4, retrieves files from Cloudflare R2 in Layer 6, sends files to CAPE Sandbox in Layer 5.7 for analysis, sends URLs and features to the ML Engine in Layer 5.6 for classification, queries the Threat Intelligence Service in Layer 5.3 for IOC lookups, stores results in Postgres Scan Results DB in Layer 6, stores reports in Cloudflare R2 in Layer 6, queues long-running jobs in BullMQ in Layer 5.9, and notifies users via the Notification Service in Layer 5.5.

5.3 Threat Intelligence Service

The Threat Intelligence Service provides threat enrichment, IOC lookups, and reputation intelligence from multiple sources using MISP (Malware Information Sharing Platform) as Service #11 and External Threat Feeds including VirusTotal, HybridAnalysis, AbuseIPDB, and URLhaus as Service #12. This service transforms raw indicators into actionable intelligence.

The service provides essential capabilities including IOC (Indicators of Compromise) lookup, domain, IP, and URL reputation checking, threat feed ingestion and correlation, threat intelligence sharing, and historical threat data storage. These capabilities enable the platform to leverage global threat intelligence for improved detection.

The lookupIOC function accepts IOCs such as IP addresses, domains, file hashes, and URLs as input and produces threat intelligence reports including associated campaigns, malware families, and threat actors as output. The function queries the MISP internal database, checks external APIs, and aggregates results from multiple sources to provide comprehensive intelligence.

The checkReputation function accepts IP addresses, domains, or URLs as input and produces reputation scores (clean, suspicious, or malicious) along with detailed context as output. It queries multiple sources including AbuseIPDB, URLhaus, VirusTotal, and HybridAnalysis in parallel, then aggregates scores to produce an overall assessment.

The mergeFeeds function accepts multiple threat feed sources as input and produces enriched, deduplicated threat intelligence as output. The process ingests feeds from various sources, deduplicates indicators, correlates related information, and stores the results in MISP. This function runs automatically via scheduled jobs managed by BullMQ.

The MISP configuration includes an internal MISP instance for threat storage and correlation, automated feed ingestion from public and private sources, event tagging with MITRE ATT&CK TTPs for standardized threat categorization, sharing groups for collaborative threat intelligence, and API integration with external services.

External threat feeds provide specialized intelligence: VirusTotal API offers file and URL scanning with hash lookup capabilities, HybridAnalysis provides automated malware analysis, AbuseIPDB delivers IP reputation and abuse reports, URLhaus maintains a malicious URL database, and AlienVault OTX contributes open threat intelligence.

The service receives IOC lookup requests from the Scan Service in Layer 5.2, queries the MISP local database and external APIs, stores intelligence in the MISP database backed by Postgres in Layer 6, updates continuously via automated feed ingestion, returns results to the Scan Service in Layer 5.2 for report inclusion, enriches alert data for the SIEM in Layer 7, and feeds into the Threat Map visualization in Layer 3.

5.4 Billing Service

The Billing Service manages subscriptions, usage tracking, and payment processing using Lemon Squeezy as Service #14. This service ensures proper monetization of the platform while providing transparent usage tracking and billing for customers.

The service provides comprehensive billing functionality including subscription plan management for Freemium, Pro, and Enterprise tiers, payment processing, invoice generation, usage tracking and metering, webhook handling for payment events, and subscription lifecycle management.

The createSubscription function accepts user ID and plan details as input and produces subscription ID and payment URL as output. The process creates a subscription in Lemon Squeezy, returns a checkout URL to the user, and stores subscription details in Postgres Billing Records in Layer 6.

The trackUsage function accepts user ID and resource usage including scans, storage, and API calls as input and produces updated usage metrics as output. The process increments usage counters and checks against plan limits, storing all data in Postgres in Layer 6 to ensure accurate billing and limit enforcement.

The handlePaymentWebhook function accepts webhook events from Lemon Squeezy as input and produces updated subscription status as output. The process validates webhooks, updates subscription information, and sends notifications to users. Events handled include payment success, payment failure, subscription renewal, and subscription cancellation.

The service receives subscription requests from the API Gateway in Layer 4, integrates with the Lemon Squeezy external API for payment processing, receives webhooks from Lemon Squeezy for payment events, stores data in Postgres Billing Records in Layer 6, checks usage against plan limits before allowing scans, notifies users via the Notification Service in Layer 5.5, and blocks actions if usage exceeds plan limits.

5.5 Notification Service

The Notification Service handles all outbound messaging and alert delivery across multiple channels using Resend for email and built-in Webhook Support as Service #13. This service ensures that users and administrators are promptly informed of important events and security findings.

The service provides extensive notification capabilities including email notifications for scan results, alerts, password resets, webhook delivery to external systems such as Slack, Teams, and SIEM, optional SMS notifications, real-time alert distribution, and notification templates with customization options.

The sendEmail function accepts recipient, subject, body, and template information as input and produces email delivery confirmation as output. Use cases include scan completion notifications, vulnerability alerts, password reset instructions, and registration confirmations. The function integrates with the Resend API for reliable email delivery.

The sendWebhook function accepts webhook URL and event payload as input and produces webhook delivery status as output. Use cases include integration with Slack, Teams, Jira, ServiceNow, and external SIEM systems. The format uses JSON payloads with structured event data for easy integration with third-party systems.

The sendSMS function (optional feature) accepts phone number and message as input and produces SMS delivery confirmation as output. Use cases include critical security alerts and MFA codes for enhanced security.

The service is triggered by all backend services when scan completion, alerts, or auth events occur. It sends notifications via the Resend API for emails, delivers to user inboxes, webhook endpoints, and external systems, logs delivery in Audit Logs in Layer 6, queues messages in BullMQ in Layer 5.9 for reliable delivery, and integrates with Notification Dispatcher in Layer 7 for SIEM alerts.

5.6 ML Engine

The ML Engine powers AI-driven threat detection using machine learning models for phishing and malware classification, utilizing Ollama with Llama 3.2 3B/8B or Mistral-Nemo as Service #10. This service provides intelligent threat detection that adapts and improves over time.

The engine provides sophisticated services including phishing URL detection and classification, malware file classification, feature extraction from files and URLs, real-time inference for threat scoring, and model training and updates.

The phishingClassifier function accepts URLs, page content, and domain features as input and produces phishing probability scores from 0 to 1 along with classification (phishing or legitimate) as output. The process extracts features, runs inference using the model, and returns predictions. The model is a fine-tuned Llama 3.2 trained on comprehensive phishing datasets.

The malwareClassifier function accepts file features from static analysis including PE headers, imports, and strings as input and produces malware probability scores and malware family classification as output. The process performs feature extraction, runs model inference, and produces classification results. The model uses Llama 3.2 8B or Mistral-Nemo trained on extensive malware samples.

The extractFeatures function accepts raw files or URLs as input and produces feature vectors for ML input as output. For files, features extracted include PE structure, imports, entropy, strings, and API calls. For URLs, features extracted include domain age, lexical features, WHOIS data, and SSL information.

The runInference function accepts feature vectors as input and produces model predictions with confidence scores as output. The process loads the appropriate model, runs inference on the features, and returns predictions with confidence levels.

The Ollama configuration includes local model hosting for privacy and performance, GPU acceleration for faster inference, model versioning and A/B testing capabilities, and continuous learning from new threat samples to improve accuracy over time.

The engine receives files and URLs for classification from the Scan Service in Layer 5.2, receives features from CAPE Sandbox in Layer 5.7 based on dynamic analysis data, stores features in the ML Features Database in Layer 6, stores predictions in the Scan Results Database in Layer 6, queues inference jobs in BullMQ in Layer 5.9 for async processing, returns results to the Scan Service in Layer 5.2 for report inclusion, and improves from labeled threat data provided by Threat Intelligence in Layer 5.3.

5.7 Sandbox (Dynamic Malware Analysis)

The Sandbox executes suspicious files in isolated virtual machines to observe runtime behavior and extract malicious activity using CAPE Sandbox as Service #9. This critical component provides deep insight into malware behavior that static analysis cannot reveal.

The sandbox provides comprehensive analysis services including safe execution of suspicious files in isolated VMs, behavioral analysis of network connections, file modifications, and registry changes, API call tracing, memory dumping and analysis, screenshot capture during execution, network traffic capture in PCAP format, dropper and payload extraction, and process tree visualization.

The runInVM function accepts files from Cloudflare R2 and execution parameters as input and produces execution session IDs as output. The process spins up clean VM snapshots, loads files, executes them in isolation, and monitors behavior. VM types available include Windows 7, 10, and 11, Linux, and Android, all configurable based on analysis needs.

The captureBehavior function accepts execution session IDs as input and produces comprehensive behavior reports as output. The system captures network connections including IPs and domains contacted, file system changes including files created, modified, or deleted, registry modifications in Windows systems, process creation trees, API calls and system calls, memory dumps, and screenshots of the execution environment.

The generateHash function accepts analyzed files and extracted artifacts as input and produces file hashes including MD5, SHA1, SHA256, and SSDEEP fuzzy hashes as output. The purpose is IOC generation for threat intelligence sharing and future correlation.

The CAPE configuration includes multiple VM snapshots for different OS versions, network simulation for realistic malware execution, a comprehensive signature database for automated detection, Yara rules for pattern matching, and integration with VirusTotal for hash checking.

The sandbox receives files to analyze from the Scan Service in Layer 5.2, retrieves files from Cloudflare R2 in Layer 6, executes files in isolated VM environment on separate infrastructure, generates behavior reports, IOCs, and artifacts, stores results in Postgres Scan Results DB in Layer 6, stores artifacts including PCAP files, memory dumps, and screenshots in Cloudflare R2 in Layer 6, sends features to the ML Engine in Layer 5.6 for classification, sends IOCs to Threat Intelligence in Layer 5.3 for correlation, is queued via BullMQ in Layer 5.9 since sandbox jobs can take 5 to 15 minutes, and notifies the Notification Service in Layer 5.5 upon completion.

5.8 Threat Modeling Engine (TIBSA Core)

The Threat Modeling Engine automates threat modeling using the TIBSA methodology, generating DFDs, STRIDE analysis, MITRE ATT&CK mapping, and risk reports using Threagile as Service #17. This engine represents the core intellectual property of the platform, automating complex security analysis that traditionally requires expert manual effort.

The engine provides comprehensive services including automated Data Flow Diagram (DFD) generation, STRIDE threat identification, MITRE ATT&CK TTP mapping, risk scoring and prioritization, security control effectiveness evaluation, and PDF/Word report generation with visualizations.

The generateDFD function accepts system architecture descriptions in JSON or YAML format as input and produces auto-generated DFDs in PNG, SVG, or JSON format as output. The process parses architecture definitions, identifies processes, data stores, and flows, and generates professional diagrams.

The performSTRIDE function accepts DFDs and system components as input and produces STRIDE threat lists covering Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege as output. The process analyzes each component and maps applicable STRIDE threats systematically.

The mapMITRE function accepts identified threats as input and produces MITRE ATT&CK TTPs mapped to threats as output. The process correlates threats with the ATT&CK framework and links specific techniques and tactics, providing standardized threat intelligence nomenclature.

The scoreRisk function accepts threats, likelihood, impact, and existing controls as input and produces risk scores similar to CVSS along with prioritized threat lists as output. The process calculates likelihood multiplied by impact, adjusts for existing controls, and ranks threats by severity.

The generateThreatReport function accepts all threat modeling data as input and produces comprehensive PDF or Word reports with diagrams, threat analysis, and recommendations as output. The process compiles all data, formats reports according to professional templates, and generates visualizations for easy consumption.

The Threagile configuration includes YAML-based architecture definitions, an extensible threat catalog, custom risk formulas inspired by CVSS plus probability calculations, PDF report templates for technical and executive audiences, and integration with the

MITRE ATT&CK database.

The engine receives threat modeling requests from the API Gateway in Layer 4, accepts system architecture from users uploaded as JSON or YAML, generates DFDs, STRIDE analysis, MITRE mapping, and risk scores, stores models in Postgres in Layer 6 for threat models and risk data, stores reports in Cloudflare R2 in Layer 6 as PDF or Word documents, stores diagrams in Cloudflare R2 as PNG or SVG files, queues jobs in BullMQ in Layer 5.9 since threat modeling can be time-intensive, notifies the Notification Service in Layer 5.5 when reports are ready, and returns results to the Frontend in Layer 3 for display in dashboards.

5.9 Background Jobs & Workflow Queue

The Background Jobs & Workflow Queue manages long-running asynchronous tasks without blocking the UI or API using BullMQ with Upstash Redis as Service #18. This system ensures reliable job execution with retry logic, enabling the platform to handle time-consuming operations gracefully.

The queue provides essential services including job queue management for async tasks, reliable job execution with retry and backoff strategies, job prioritization and scheduling, worker process management, job status tracking and monitoring, and dead letter queue for failed jobs.

Sandbox Analysis Jobs are queued by the Scan Service in Layer 5.2 and execute CAPE Sandbox file analysis in Layer 5.7. These jobs typically take 5 to 15 minutes per file and are assigned high priority for user-submitted files to ensure timely results.

ML Inference Jobs are queued by the Scan Service in Layer 5.2 and execute ML Engine classification in Layer 5.6. These jobs typically take 10 to 60 seconds per inference and are assigned medium priority as they are faster than sandbox analysis.

Threat Modeling Jobs are queued by the Threat Modeling Engine in Layer 5.8 and execute full TIBSA analysis including DFD, STRIDE, MITRE, and Report generation. These jobs typically take 2 to 10 minutes per model and are assigned low priority as batch processing is acceptable for these comprehensive analyses.

Threat Feed Ingestion Jobs are queued by Threat Intelligence in Layer 5.3 and execute automated feed ingestion from external sources. These jobs typically take 5 to 30 minutes and are scheduled hourly or daily with low priority as they are background maintenance tasks.

Report Generation Jobs are queued by the Scan Service in Layer 5.2 and Threat Modeling in Layer 5.8, executing PDF or Word report compilation and rendering. These jobs typically take 30 to 120 seconds and are assigned medium priority for timely delivery to users.

The BullMQ configuration includes Redis-backed queue persistence using Upstash

Redis in Layer 6, multiple worker pools for different job types, exponential backoff for failed jobs, concurrency limits to prevent resource exhaustion, job monitoring via Grafana in Layer 7, and rate limiting for external API calls particularly for Threat Intel feeds.

The queue receives jobs from all backend services in Layer 5.2, 5.3, 5.6, 5.7, and 5.8, stores queue in Upstash Redis in Layer 6, executes via worker processes on backend infrastructure, monitors via Prometheus metrics that feed into Grafana dashboards in Layer 7, notifies on completion via the Notification Service in Layer 5.5, updates job status in Postgres in Layer 6 for user visibility, and logs failures in Audit Logs in Layer 6.

4.1.7 LAYER 6 - Data Storage Layer

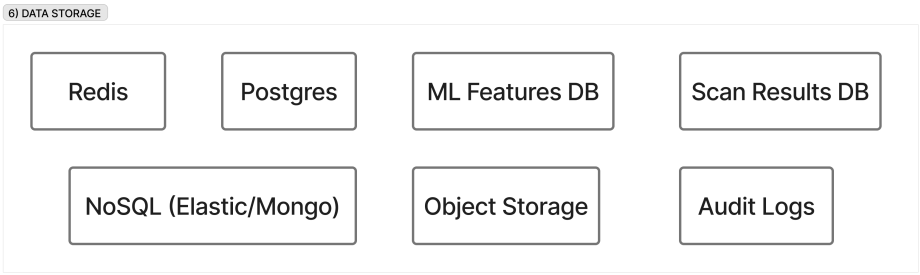


Figure 4.7: Figure from document

6.1 Relational Database (Postgres)

The Relational Database serves as the primary storage for all structured data requiring ACID transactions and complex queries using Neon Serverless Postgres as Service #5. This database is the foundation of data persistence for the entire platform, ensuring data integrity and consistency.

The database stores multiple categories of critical data: Users information includes credentials managed via Authentik, profiles, preferences, and roles. Scans data encompasses scan metadata, request parameters, and scan status. Reports information includes report metadata, generation timestamps, and access logs. Threat Models data covers architecture definitions, DFDs, STRIDE analysis, and risk scores. Audit Logs contain all user actions, system events, and administrative changes. Billing Records include subscriptions, invoices, usage metrics, and payment history. System Configuration stores platform settings, feature flags, and RBAC policies.

The schema design includes a users table with id, email, role, created_at, and last_login fields. The scans table contains id, user_id, scan_type, target, status, created_at, and results_json fields. The threat_models table includes id, user_id, architecture_json, dfd_url, stride_results, and risk_score fields. The audit_logs table stores id, user_id, action, resource, timestamp, and ip_address information. The billing table maintains id, user_id, plan, usage, and subscription_status data.

The database is written by all backend services in Layer 5.1 through 5.9 via ORM, read by all backend services for queries, backed up through automated daily backups provided by Neon, replicated across multiple regions for high availability, and indexed with optimized queries for user_id, scan_id, and timestamps. It connects to Authentik in Layer 4 for user authentication data and is queried by Frontend dashboards in Layer 3 via the API Gateway in Layer 4.

6.2 Cache & Queue (Redis)

The Cache & Queue system provides in-memory caching for performance optimization and queue management for background jobs using Upstash Serverless Redis as Service #6. This high-speed data store dramatically improves platform performance by reducing database load and enabling real-time features.

The system stores multiple types of data: Session Cache maintains active user sessions and JWT refresh tokens. Rate Limiting stores request counters per user and IP for the API Gateway in Layer 4. Query Cache holds frequently accessed data such as user profiles and scan results. Job Queue manages BullMQ job queue in Layer 5.9 for background tasks. Temporary Results stores ML inference results before they are written to Postgres. Real-time Metrics tracks live scan counts, active users, and system load.

The Redis data structures used include Strings for session tokens and cached JSON responses, Hashes for user session data and cached objects, Lists for job queues used by BullMQ, Sets for active users and IP blacklists, and Sorted Sets for rate limit counters with TTL.

The cache is read from and written to by the API Gateway in Layer 4 for rate limiting, used by BullMQ in Layer 5.9 for job queue persistence, utilized by all backend services in Layer 5.1 through 5.8 for performance, serves as session storage for Authentik in Layer 4, manages TTL with auto-expiry for stale cache entries, and falls back to Postgres in Layer 6.1 on cache miss. The system is monitored by Prometheus in Layer 7 for hit and miss rates.

6.3 Object Storage (S3-compatible)

The Object Storage system provides scalable storage for large binary objects, files, and generated artifacts using Cloudflare R2 as Service #7. This cost-effective storage solution handles all large files without egress fees, making it ideal for security artifacts and reports.

The storage holds multiple types of data: Uploaded Files include user-submitted files for scanning such as binaries, documents, and PCAPs. PDF Reports contain generated security reports from the Scan Service in Layer 5.2 and Threat Modeling in Layer 5.8.

Threat Model Diagrams include DFD images in PNG and SVG formats from Threagile in Layer 5.8. Sandbox Artifacts encompass CAPE screenshots, memory dumps, PCAP files, and extracted payloads. Generated Documents include Word reports and executive summaries. User Uploads store profile pictures and team logos if applicable.

The bucket structure organizes files into uploaded-files/ for user-submitted files pending scanning, scan-reports/ for PDF and JSON scan reports, threat-models/ for DFD diagrams and model documents, sandbox-artifacts/ for CAPE analysis outputs including memory dumps, PCAPs, and screenshots, and user-assets/ for profile pictures and logos.

The storage is written to by the File Upload Handler in Layer 3.4 via Tusd, the Scan Service in Layer 5.2 for reports, the Threat Modeling Engine in Layer 5.8 for diagrams and reports, and CAPE Sandbox in Layer 5.7 for artifacts. It is read by the Scan Service in Layer 5.2 to retrieve files for analysis, the Frontend in Layer 3 for report downloads, and CAPE Sandbox in Layer 5.7 to fetch files for execution. Metadata is stored in Postgres in Layer 6.1 including file IDs, URLs, sizes, and timestamps. Access control uses pre-signed URLs for secure, time-limited access, lifecycle policies auto-delete old files after retention periods, and CDN integration with Cloudflare CDN in Layer 2 caches frequently accessed reports.

6.4 Audit Logs Storage

The Audit Logs Storage provides immutable storage of all user actions, system events, and administrative changes for compliance and forensics using a dedicated Postgres table in Neon as Service #5 or a separate append-only log store. This system ensures complete accountability and traceability for all platform activities.

The storage logs multiple categories of events: User Actions include login and logout, file uploads, scan initiations, and report downloads. Administrative Changes cover user creation, role assignments, and configuration updates. System Events track service starts and stops, errors, and security incidents. API Requests log all API calls with timestamps, user IDs, IP addresses, and payloads. Authentication Events record MFA enrollments, password changes, and failed login attempts. Data Access tracks who accessed which reports, threat models, and scan results.

The log structure includes Timestamp for precise event time in ISO 8601 format, User ID for the actor whether user or system, Action for the verb such as created, updated, deleted, or accessed, Resource for the target such as scan ID, user ID, or report URL, IP Address for the source IP, User Agent for browser or client information, Result indicating success or failure, and Metadata providing additional context in JSON format.

The logs are written by all backend services in Layer 5.1 through 5.9 via logging middleware, are immutable as append-only with no updates or deletes, are queried by admin dashboards for compliance reporting, are indexed by timestamp, user_id, and action for fast searches, are exported to Grafana Loki in Layer 7 for log aggregation, meet compliance requirements for GDPR, SOC 2, and ISO 27001 audit trails, and maintain configurable retention periods such as 7 years for compliance.

6.5 ML Features Database

The ML Features Database provides specialized storage for machine learning training data, feature vectors, and model metadata using dedicated Postgres tables or NoSQL such as MongoDB or Elastic as an extension of Service #5. This specialized storage supports the continuous improvement of the platform's AI capabilities.

The database stores multiple types of ML-specific data: Feature Vectors contain extracted features from files and URLs for ML training. Training Datasets include labeled samples categorized as phishing or benign, malware or clean. Model Metadata tracks model versions, training dates, hyperparameters, and accuracy metrics. Inference Results store ML predictions with confidence scores. Evaluation Metrics maintain precision, recall, F1 scores, and confusion matrices.

The schema design includes a features table with id, file_hash or url, feature_vector in JSON or JSONB, label, and timestamp fields. The models table contains id, model_name, version, trained_at, accuracy, and model_file_url pointing to R2. The predictions table includes id, scan_id, model_id, prediction, confidence, and timestamp fields.

The database is written by the ML Engine in Layer 5.6 for feature extraction and predictions, and by CAPE Sandbox in Layer 5.7 for behavioral features. It is read by the ML Engine in Layer 5.6 for training and inference. The training pipeline performs periodic retraining with new samples, model versioning tracks model improvements over time, A/B testing compares multiple models in production, and the database integrates with Threat Intelligence in Layer 5.3 for labeled threat data.

6.6 Scan Results Database

The Scan Results Database stores comprehensive results from all security scans including URL, file, sandbox, and threat intel using Postgres tables in Neon as Service #5 or NoSQL for flexibility. This database aggregates findings from multiple analysis engines into cohesive scan results.

The database stores multiple categories of scan data: Scan Metadata includes scan ID, user ID, scan type, target, and timestamp. Detection Results contain malicious indicators and threat classifications. Threat Intelligence includes IOC lookups and reputation scores. Sandbox Results contain behavioral analysis summaries. ML Predictions store classification scores from the ML Engine. External Feed Results include VirusTotal and HybridAnalysis responses. Historical Data maintains previous scan results for trending and comparison.

The schema design includes a scans table with id, user_id, scan_type, target, status, and created_at fields. The scan_results table contains id, scan_id, source indicating sandbox, ml, or threat_intel, result_json, and severity fields. The ioc_matches table includes id, scan_id, ioc_type, ioc_value, and threat_intel_source fields.

The database is written by the Scan Service in Layer 5.2 for aggregated scan results, CAPE Sandbox in Layer 5.7 for behavioral analysis, the ML Engine in Layer 5.6 for classification predictions, and Threat Intelligence in Layer 5.3 for IOC matches. It is read by the Scan Service in Layer 5.2 for report generation, Frontend dashboards in Layer 3 for visualization, and the Alerts Engine in Layer 7 for alert triggering. The database is indexed by scan_id, user_id, timestamp, and severity, maintains configurable retention such as keeping the last 90 days and archiving older data, and is exported to the SIEM in Layer 7 for security monitoring.

4.1.8 LAYER 7 - Monitoring & Threat Analysis

Monitoring & Threat Analysis
SIEM
Alert
Threat
Notification

Figure 4.8: Figure from document

7.1 SIEM (Security Information and Event Management)

The SIEM collects, aggregates, and analyzes security events from all platform components to detect attacks on the infrastructure itself using Wazuh Open Source as Service #15. This system provides the security monitoring backbone for the entire platform, protecting the protector.

The SIEM provides comprehensive services including log collection from all services including API Gateway, backend services, and databases, security event correlation, intrusion detection with IDS and IPS capabilities, file integrity monitoring (FIM), vulnerability detection, compliance monitoring for PCI DSS, GDPR, and HIPAA, real-time alerting, and a threat hunting interface.

Data sources include API Gateway logs covering request patterns, authentication failures, and rate limit violations, Backend service logs capturing application errors and suspicious activity, System logs recording OS-level events, process executions, and network connections, Database audit logs tracking unauthorized access attempts and schema changes, and Edge layer logs containing WAF blocks and DDoS events from Cloudflare.

Detection capabilities include identifying brute-force attacks through multiple failed login attempts, detecting SQL injection attempts through malicious query patterns in logs, monitoring file tampering for unauthorized file modifications, detecting privilege escalation through unexpected role changes, identifying data exfiltration through unusual data transfer patterns, and detecting malware on infrastructure through suspicious process execution.

The SIEM collects from all layers including Edge, Frontend, API, Backend, and Storage, receives logs via Syslog, file monitoring, and API integrations, stores logs in Wazuh Elasticsearch backend, correlates events using a real-time analysis engine, sends alerts to the Alert Engine in Layer 7.2 for prioritization, integrates with Grafana Loki in Layer 7.4 for unified log view, feeds the Threat Dashboard in Layer 7.3 with security metrics, and triggers the Notification Dispatcher in Layer 7.4 for critical alerts.

7.2 Alert Engine

The Alert Engine processes, filters, and prioritizes alerts from SIEM, reducing noise and triggering appropriate responses using custom alert processing logic as part of Wazuh or as a standalone component. This intelligent filtering ensures that security teams focus on genuine threats rather than false positives.

The engine provides critical services including alert deduplication to suppress repeated alerts, severity-based prioritization, context enrichment adding user, asset, and threat intel context, alert correlation to link related events, false positive suppression, automated response triggering, and escalation to human analysts.

The alert processing pipeline follows a structured workflow: First, it receives raw alerts from the SIEM in Layer 7.1. Second, it deduplicates to suppress duplicate alerts within time windows. Third, it enriches alerts by adding context from Postgres for user info and Threat Intel in Layer 5.3. Fourth, it correlates by linking related alerts such as multiple failed logins indicating brute force. Fifth, it prioritizes by scoring based on severity, asset criticality, and threat intel. Sixth, it routes with critical alerts receiving immediate escalation and low priority alerts queued for review. Finally, it responds by triggering automated actions such as blocking IPs, disabling users, or isolating hosts.

The engine receives raw security events from the SIEM in Layer 7.1, enriches with Threat Intelligence from Layer 5.3 for IOC context and Postgres in Layer 6.1 for user and asset data, sends processed alerts to the Threat Dashboard in Layer 7.3 for visualization and the Notification Dispatcher in Layer 7.4 for stakeholder alerts, triggers automated responses such as API calls to block IPs in Cloudflare, and logs actions in Audit Logs in Layer 6.4.

7.3 Threat Dashboard

The Threat Dashboard provides visual interface for security analysts to monitor threats, investigate incidents, and track security posture using Grafana dashboards as part of Service #16 combined with custom UI in Next.js from Layer 3. This dashboard transforms raw security data into actionable intelligence.

The dashboard provides multiple visualizations: Active Alerts shows a real-time alert feed with severity indicators. Threat Heatmap displays geographic visualization of attack origins. Attack Timeline provides chronological view of security events. Top Threats highlights most frequent attack types and sources. Asset Risk Scores displays criticality-based asset view. Incident Status tracks open, in-progress, and closed incidents. Alert Trends shows historical alert volume over time. Attack Graphs provides visual representation of multi-step attacks.

Features include drill-down capability to click alerts and view full details and evidence, filtering by severity, time, asset, and alert type, search functionality with full-text search across all security events, export capability for reports for stakeholders and compliance audits, and collaboration features to assign alerts to analysts and add notes.

The dashboard receives data from the Alert Engine in Layer 7.2 for processed, prioritized alerts, the SIEM in Layer 7.1 for raw event data, and Threat Intelligence in Layer 5.3 for IOC context. It displays on the Frontend in Layer 3 specifically in the Security Analyst Console in Layer 1.1, integrates with the Threat Map in Layer 3 for geo-visualization and the Alerts Table in Layer 3 for tabular view, and queries Postgres in Layer 6.1 for historical data.

7.4 Observability Stack (Monitoring, Metrics, Logs)

The Observability Stack provides comprehensive observability for system performance, errors, and operational metrics using Grafana, Prometheus, and Loki as Service #16. This stack ensures that operations teams have complete visibility into platform health and performance.

Prometheus handles metrics collection as a time-series database, providing services including time-series metrics collection, service health monitoring, resource usage tracking for CPU, memory, disk, and network, API response times and error rates, database query performance, queue lengths for BullMQ jobs, and cache hit and miss rates for Redis.

Metrics collected include API Gateway metrics covering requests per second, latency, error rates, and rate limit hits, Backend Services metrics tracking processing times, job queue depths, and error counts, Database metrics monitoring query times, connection pools, and disk usage, ML Engine metrics recording inference times and model accuracy, and Sandbox metrics tracking VM utilization and analysis completion times.

Loki handles log aggregation, providing services including centralized log collection from all services, log querying with LogQL, log correlation with traces, and long-term log storage. Logs aggregated include application logs covering errors, warnings, and info, access logs for API requests, audit logs for user actions, system logs for OS events, and security logs for authentication and authorization.

Grafana provides visualization and dashboards, offering services including unified dashboard for metrics and logs, custom dashboards per service, alerting based on thresholds, anomaly detection, and performance trending.

Dashboards include Platform Overview showing overall system health, active users, and scan volume, API Gateway displaying request rates, error rates, and latency percentiles, Backend Services showing service-specific metrics and job processing times, Database Performance tracking query times, connection pools, and slow queries, Security Monitoring displaying failed logins, rate limit violations, and WAF blocks, and Cost Monitoring showing resource usage for billing optimization.

The stack scrapes metrics from all backend services in Layer 5.1 through 5.9, the API Gateway in Layer 4.1, and Databases in Layer 6.1 through 6.6. It collects logs from all services via Loki agents, visualizes in Grafana dashboards, sends alerts to the Notification Service in Layer 5.5 on threshold violations, integrates with the SIEM in Layer 7.1 for security event correlation, and is accessed by DevOps team, SREs, and Security analysts.

7.5 Notification Dispatcher

The Notification Dispatcher routes alerts and notifications to appropriate stakeholders via multiple communication channels using Resend as Service #13 for emails and webhook integrations for Slack, Teams, and Jira. This system ensures that critical information reaches the right people through their preferred channels.

The dispatcher provides essential services including email alert delivery for critical security events and system outages, Slack and Teams integration for real-time team notifications, ticketing system integration with Jira and ServiceNow for incident tracking, optional SMS alerts for critical events via Twilio integration, and escalation management to notify senior analysts if alerts go unaddressed.

Notification types include Critical Security Alerts for brute-force attacks, data breaches, and infrastructure compromise, System Health notifications for service outages, database failures, and high error rates, User Notifications for scan completions and report readiness from Layer 5, and Operational Alerts for queue backlogs, resource exhaustion, and failed jobs.

The dispatcher receives from the Alert Engine in Layer 7.2 for security alerts, Prometheus in Layer 7.4 for system health alerts, and the SIEM in Layer 7.1 for critical security events. It delivers via Resend for email, Webhooks for Slack, Teams, and Jira, and optionally SMS. The dispatcher logs delivery in Audit Logs in Layer 6.4 and escalates unacknowledged critical alerts to senior staff.

4.1.9 LAYER 8 - TIBSA Suite (Core)

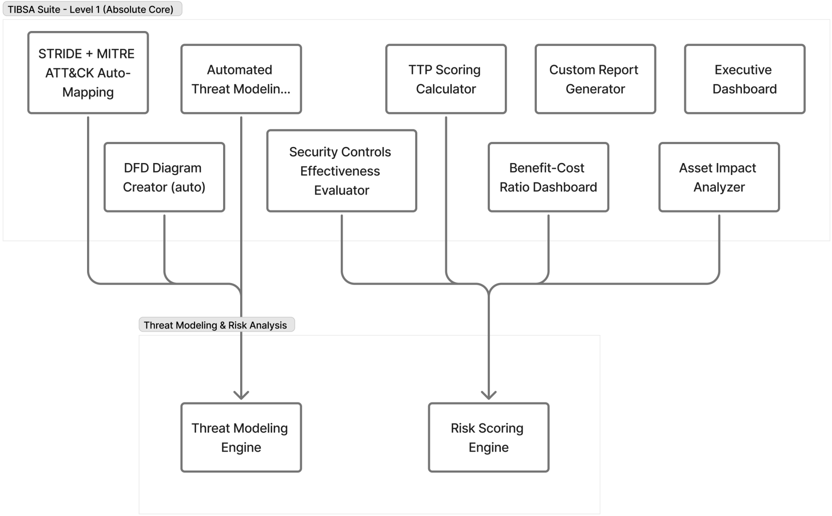

TIBSA Suite - Level 1 (Absolute Core)
STRIDE + MITRE ATT&CK Auto-Mapping
Automated Threat Modelin...
TTP Scoring Calculator
Custom Report Generator
Executive Dashboard
DFD Diagram Creator (auto)
Security Controls Effectiveness Evaluator
Benefit-Cost Ratio Dashboard
Asset Impact Analyzer
Threat Modeling & Risk Analysis
Threat Modeling Engine
Risk Scoring Engine

Figure 4.9: Figure from document

8.1 TIBSA Suite - Level 1 (Absolute Core)

The TIBSA Suite represents the foundational automation engines that execute the complete threat-modeling and risk-analysis workflow, transforming raw system information into structured models, identified threats, calculated risks, and actionable reports using Threagile as Service #17 as the core automation engine.

The Automated Threat Modeling Engine applies the complete TIBSA methodology automatically, accepting system architecture in JSON or YAML format, asset inventory, and data flows as input and producing complete threat models with identified threats as output. The process parses architecture, identifies assets, maps data flows, detects threat patterns, and produces comprehensive models.

The DFD Diagram Creator generates Data Flow Diagrams automatically, accepting parsed system descriptions as input and producing structured DFDs in PNG, SVG, or JSON format as output showing processes, data stores, external entities, and data flows. The process identifies components, maps relationships, generates visual diagrams, and stores them in Cloudflare R2 in Layer 6.

The STRIDE and MITRE ATT&CK Auto-Mapping component provides standardized threat categorization and TTP mapping, accepting system components, DFDs, and identified weak points as input and producing STRIDE threat categories plus correlated MITRE ATT&CK techniques as output. The process maps components to STRIDE categories covering Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege, correlates threats with MITRE ATT&CK TTPs, and links to real-world attack patterns ensuring industry-aligned, standardized threat identification.

The TTP Scoring Calculator provides risk scoring for each threat and TTP, accepting identified threats, likelihood factors, impact assessment, exploitation difficulty, and environmental context as input and producing risk scores on a 0 to 10 scale along with likelihood and impact ratings as output. The scoring model considers likelihood based on threat actor capability, attack complexity, and existing controls, impact based on asset criticality, data sensitivity, and business disruption, and combines these into a combined score calculated as likelihood multiplied by impact with contextual adjustments using a formula similar to CVSS but enhanced with probability estimation.

The Security Controls Effectiveness Evaluator assesses existing security controls against identified threats, accepting organization's security controls inventory and identified threats as input and producing control effectiveness ratings, coverage gaps, redundancies, and priority controls needed as output. The evaluation determines which threats are adequately mitigated, which threats lack sufficient controls, which controls are redundant or ineffective, and the cost-benefit of proposed controls.

The Benefit-Cost Ratio Dashboard provides cost-effectiveness analysis for control recommendations, accepting recommended controls, implementation costs, and expected risk reduction as input and producing B/C ratios and prioritized control recommendations ranked by cost-effectiveness as output. This helps decision-makers allocate security budget optimally.

The Asset Impact Analyzer provides business and technical impact assessment for critical assets, accepting asset inventory, business criticality, and dependencies as input and producing asset criticality scores, impact categories covering financial, operational, reputational, and compliance aspects, and priority rankings as output. This supports risk-based decision-making and resource allocation.

The Custom Report Generator automates comprehensive report generation, accepting all threat modeling data including DFD, STRIDE, MITRE, risks, and controls as input and producing PDF or Word reports as output. The reports include executive summary, technical threat analysis, risk scores and rankings, recommended controls with priorities, diagrams and visualizations, and compliance mappings. Templates are customizable for different audiences including technical, executive, and compliance stakeholders.

The Executive Dashboard provides high-level leadership dashboard, accepting aggregated threat modeling results as input and producing single-page summary as output. The summary includes top threats and risk levels, critical assets at risk, recommended priority actions, overall security posture score, and trend indicators showing whether security is improving or worsening. The purpose is to enable rapid executive decision-making.

The suite receives threat modeling requests from the API Gateway in Layer 4, accepts input data from users via the Frontend in Layer 3 in the form of architecture descriptions, executes via the Threagile automation engine, stores models in Postgres in Layer 6 for threat data and risk scores, stores artifacts in Cloudflare R2 in Layer 6 for DFD diagrams and PDF or Word reports, queues jobs in BullMQ in Layer 5.9 for async processing taking 2 to 10 minutes, integrates with the MITRE ATT&CK database for TTP mapping, notifies via the Notification Service in Layer 5.5 when reports are ready, displays in the Frontend Dashboard in Layer 3 specifically in the Executive Dashboard component, and exports to the SIEM in Layer 7 for security monitoring integration.

4.1.10 Complete Data Flow Map

Flow 1: User Authentication & Session Management

The authentication flow begins when a user accesses the platform through Cloudflare Edge which provides CDN, WAF, DDoS protection, TLS termination, and bot detection. The request proceeds to the Frontend using Next.js where the Login Form is presented. User credentials are processed through Client-Side Security including XSS Sanitizer and CSRF Handler before being passed to the Auth Interceptor. The Auth Interceptor forwards the authenticated request to the API Gateway (KrakenD) which applies the Auth Filter and Rate Limiter. The request is then sent to Authentik which handles JWT issuance and MFA verification. The session is stored in Redis in Layer 6 while user information is stored in Postgres in Layer 6. The JWT is returned to the Frontend and stored in the browser. All subsequent requests include the JWT via the Auth Interceptor.

Flow 2: File Upload & Malware Analysis

The file upload and analysis flow starts when a user selects a file in the File Upload Component in Frontend Layer 3. The File Hash Calculator computes a SHA-256 hash of the file for deduplication. The Chunk Uploader splits the file into manageable chunks for reliable upload. Uppy and Tusd handle the resumable upload process, sending chunks to Cloudflare R2 in Layer 6 where the complete file is stored. The API Gateway routes the scan request to the Scan Service in Layer 5 which initiates comprehensive analysis. BullMQ queues a sandbox job in Layer 5.9 for asynchronous processing. CAPE Sandbox in Layer 5.7 retrieves the file from R2 and executes it in an isolated VM. The sandbox captures behavior including network connections, file modifications, and API calls, generating IOCs in the process. Artifacts including screenshots, memory dumps, and PCAP files are stored in R2 while results are stored in Postgres. The ML Engine in Layer 5.6 classifies the file based on behavioral features and stores predictions in the ML Features DB. Threat Intelligence in Layer 5.3 checks IOCs via MISP and external APIs. The Scan Service aggregates all results from sandbox, ML, and threat intel sources. A comprehensive PDF report is generated and stored in R2. The Notification Service sends a completion email to the user. Finally, the user views the detailed report in the Frontend Dashboard.

Flow 3: URL Phishing Detection

The URL scanning flow begins when a user submits a URL to the Scan Service in Layer 5. The ML Engine in Layer 5.6 extracts URL features including domain age, lexical analysis, WHOIS data, and SSL information. The Phishing Classifier runs inference using Llama 3.2 to produce a probability score. Threat Intelligence checks URL reputation by querying VirusTotal, URLhaus, and other external feeds. Results from ML classification and threat intelligence are aggregated by the Scan Service. The complete analysis is stored in the Scan Results DB in Layer 6. A report is generated and stored in R2 for user download. A notification is sent to the user via the Notification Service. Results are displayed in the Alerts Table in Frontend Layer 3 for immediate review.

Flow 4: Threat Modeling Workflow

The threat modeling workflow starts when a user uploads system architecture in JSON or YAML format through the Frontend Pentest Wizard in Layer 3. The request passes through the API Gateway to the Threat Modeling Engine (Threagile) in Layer 5. BullMQ queues the threat modeling job for asynchronous processing. Threagile parses the architecture definition and identifies system components. A DFD is automatically generated and stored in R2. STRIDE analysis is performed to identify threats across all categories. MITRE ATT&CK TTPs are mapped to identified threats for standardization. Risk scores are calculated based on likelihood, impact, and existing controls. Existing controls are evaluated for effectiveness and coverage gaps. Benefit-Cost ratios are computed for recommended controls. A comprehensive PDF or Word report is produced including all analysis, diagrams, and recommendations. The complete threat model data is stored in Postgres in Layer 6 while artifacts are stored in R2. A notification is sent to the user via the Notification Service. The user views the executive summary and full report in the Executive Dashboard in Layer 3.

Flow 5: Background Job Processing

The background job processing flow begins when any service including Scan, ML, Threat Modeling, or Threat Intel needs to perform a long-running task. The service queues a job in BullMQ in Layer 5.9 with appropriate priority and parameters. The job is stored in Upstash Redis in Layer 6 for persistence. A worker process picks up the job from the queue based on priority and availability. The worker executes the long-running task which may include Sandbox analysis, ML inference, or report generation. Job status is continuously updated in Redis to track progress. Results are stored in Postgres for structured data or R2 for large files and reports in Layer 6. Upon completion, the Notification Service is triggered to inform the user. The Frontend polls for status updates or receives WebSocket notifications to update the UI in real-time.

Flow 6: Security Monitoring & Alerting

The security monitoring flow operates continuously as all services in Layers 2 through 5 generate logs during normal operation. Logs are sent to Wazuh SIEM in Layer 7 via various methods including Syslog, file monitoring, and API integrations. The SIEM correlates events across different services to detect complex attack patterns. When threats are detected, security alerts are generated with detailed context. The Alert Engine in Layer 7 processes these alerts by deduplicating repeated events, prioritizing based on severity, and enriching with context from Threat Intelligence in Layer 5. Enriched and prioritized alerts are sent to the Threat Dashboard in Layer 7 for visualization. Critical alerts are sent to the Notification Dispatcher which delivers notifications via email, Slack, Teams, or other channels. Alerts are displayed in the Security Analyst Console in Layer 1 for investigation and response.

Flow 7: Metrics & Performance Monitoring

The metrics and performance monitoring flow operates as all services expose metrics endpoints in Prometheus format. Prometheus in Layer 7 scrapes metrics from all services every 15 seconds. Metrics are stored in Prometheus time-series database for historical analysis. Grafana in Layer 7 queries Prometheus to populate real-time dashboards. Dashboards display critical metrics including request rates, error rates, latency, resource utilization, and business metrics. When metrics exceed defined thresholds, alerts are triggered automatically. Notifications are sent via the Notification Service to inform the DevOps team. The DevOps team responds to alerts by investigating issues and taking corrective action to maintain platform health.

Flow 8: Threat Intelligence Enrichment

The threat intelligence enrichment flow begins when the Scan Service identifies an IOC such as an IP address, domain, file hash, or URL during analysis. The Threat Intelligence Service in Layer 5 is queried with the IOC. The service first queries the MISP local database for known threats. External APIs including VirusTotal, AbuseIPDB, URLhaus, and HybridAnalysis are queried in parallel. Reputation scores and threat context are aggregated from all sources. Enriched context including associated campaigns, malware families, and threat actors is returned to the Scan Service. Results are included in the comprehensive scan report. IOCs are stored in MISP for future correlation and threat intelligence sharing. Automated feed ingestion via BullMQ scheduled jobs continuously updates the threat intelligence database with the latest indicators.

Flow 9: Billing & Subscription Management

The billing flow starts when a user initiates a subscription through the Frontend. The Billing Service in Layer 5 receives the subscription request. A subscription is created in Lemon Squeezy via API call. A checkout URL is returned to the user for payment. The user completes payment in the Lemon Squeezy hosted checkout. Lemon Squeezy sends a webhook to the Billing Service confirming payment. The Billing Service validates the webhook signature for security. Subscription status is updated in Postgres in Layer 6. A welcome notification is sent to the user via the Notification Service. During ongoing usage, every scan or API call increments the usage counter. Usage is checked against plan limits before allowing resource-intensive operations. When usage approaches limits, overage alerts are sent to users via the Notification Service. Users exceeding plan limits are blocked from additional scans until they upgrade or usage resets.

Flow 10: Audit & Compliance Logging

The audit and compliance logging flow operates continuously as every user action including login, scan initiation, report download, and configuration change is captured. Backend services log actions immediately as they occur. Audit log entries are written to Postgres in Layer 6 with complete details. Each log entry includes timestamp, user ID, action type, resource affected, IP address, and result status. Logs are exported to Loki in Layer 7 for aggregation with other log types. Administrators can query audit logs via the Frontend UI with powerful search and filtering. Compliance reports are generated from audit logs on demand or on schedule. The immutable audit trail supports forensic investigations and compliance audits for GDPR, SOC 2, ISO 27001, and other frameworks.

4.1.11 Summary of Tool Distribution by Layer

Layer 1 - User Layer

This layer uses no external tools, consisting entirely of pure UI and client logic components.

Layer 2 - Edge Layer

This layer utilizes Cloudflare Free Plan as Service #1, which provides comprehensive edge services including CDN, WAF, DDoS Protection, TLS Termination, Load Balancing, and Bot Detection.

Layer 3 - Frontend Layer

This layer employs Next.js 15 with Tailwind CSS and shadcn/ui as Service #2 for all UI components, dashboards, and forms. Additionally, it uses Uppy with Tusd as Service #8 for resumable file upload and chunking capabilities.

Layer 4 - API Engine

This layer leverages KrakenD Community Edition as Service #4 for the API Gateway providing routing, rate limiting, and validation. It also uses Authentik Open Source as Service #3 for authentication, MFA, RBAC, and user management.

Layer 5 - Backend Services

This layer integrates multiple specialized services: CAPE Sandbox as Service #9 for dynamic malware analysis and VM execution, Ollama with Llama 3.2 or Mistral-Nemo as Service #10 for ML-powered phishing and malware classification, MISP as Service #11 for threat intelligence platform and IOC management, External Threat Feeds as Service #12 including VirusTotal, HybridAnalysis, AbuseIPDB, and URLhaus, Resend with Webhooks as Service #13 for email and webhook notifications, Lemon Squeezy as Service #14 for billing and subscription management, Threagile as Service #17 for automated threat modeling representing the TIBSA core, and BullMQ with Upstash Redis as Service #18 for background job queue management.

Layer 6 - Data Storage

This layer utilizes Neon Serverless Postgres as Service #5 as the primary relational database, Upstash Serverless Redis as Service #6 for caching, session storage, rate limiting, and job queue, and Cloudflare R2 as Service #7 for object storage of files, reports, and artifacts.

Layer 7 - Monitoring & Security

This layer employs Wazuh Open Source as Service #15 for SIEM, security monitoring, and intrusion detection. It also uses Grafana with Prometheus and Loki as Service #16 for metrics collection, log aggregation, and dashboards.

Layer 8 - TIBSA Core

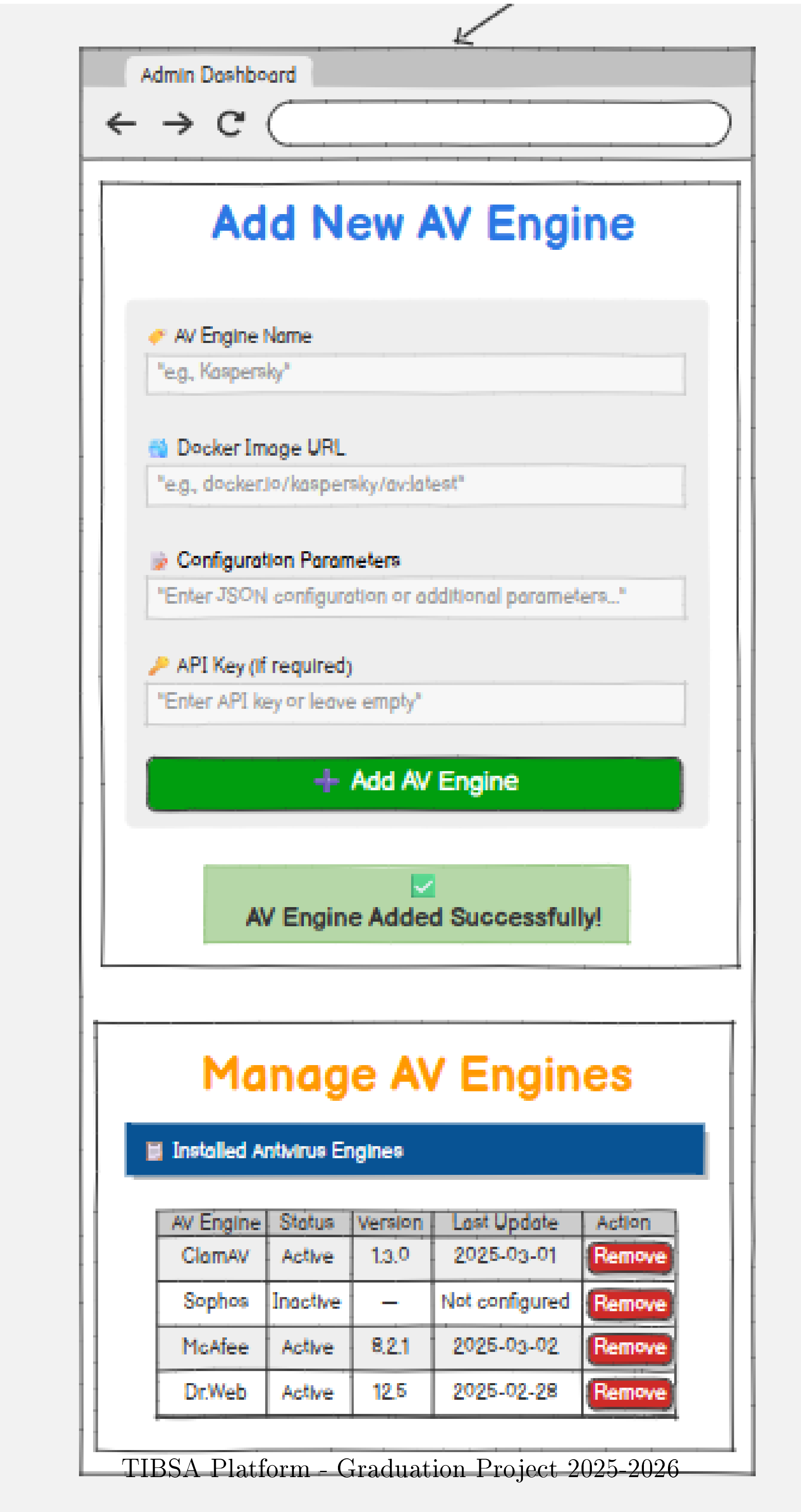
This layer is powered by Threagile as Service #17 for all core TIBSA automation including DFD generation, STRIDE analysis, MITRE mapping, risk scoring, and report generation.

4.1.12 Key Integration Points Summary

The authentication flow moves from Cloudflare through Next.js to KrakenD to Authentik and finally to Backend Services. The file analysis pipeline flows from Uppy/Tusd to R2 to CAPE Sandbox to ML Engine to MISP and culminates in comprehensive reports. The threat modeling workflow begins with user input, processes through Threagile to produce DFD, STRIDE, and MITRE analysis, and stores reports in R2. Background processing flows from all services through BullMQ to Redis Queue where workers process jobs and store results in appropriate storage. The monitoring flow captures data from all services, sends it to Prometheus and Loki, displays it in Grafana Dashboards, and generates alerts as needed. Security monitoring aggregates all logs in Wazuh SIEM, processes them through the Alert Engine, and distributes notifications via the Notification Dispatcher. Data persistence is handled with all services storing structured data in Postgres, utilizing Redis for cache, and storing large objects in R2.

4.1.13 Scenarios

4.1.14 Login Screen



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

Installed Antivirus Engines

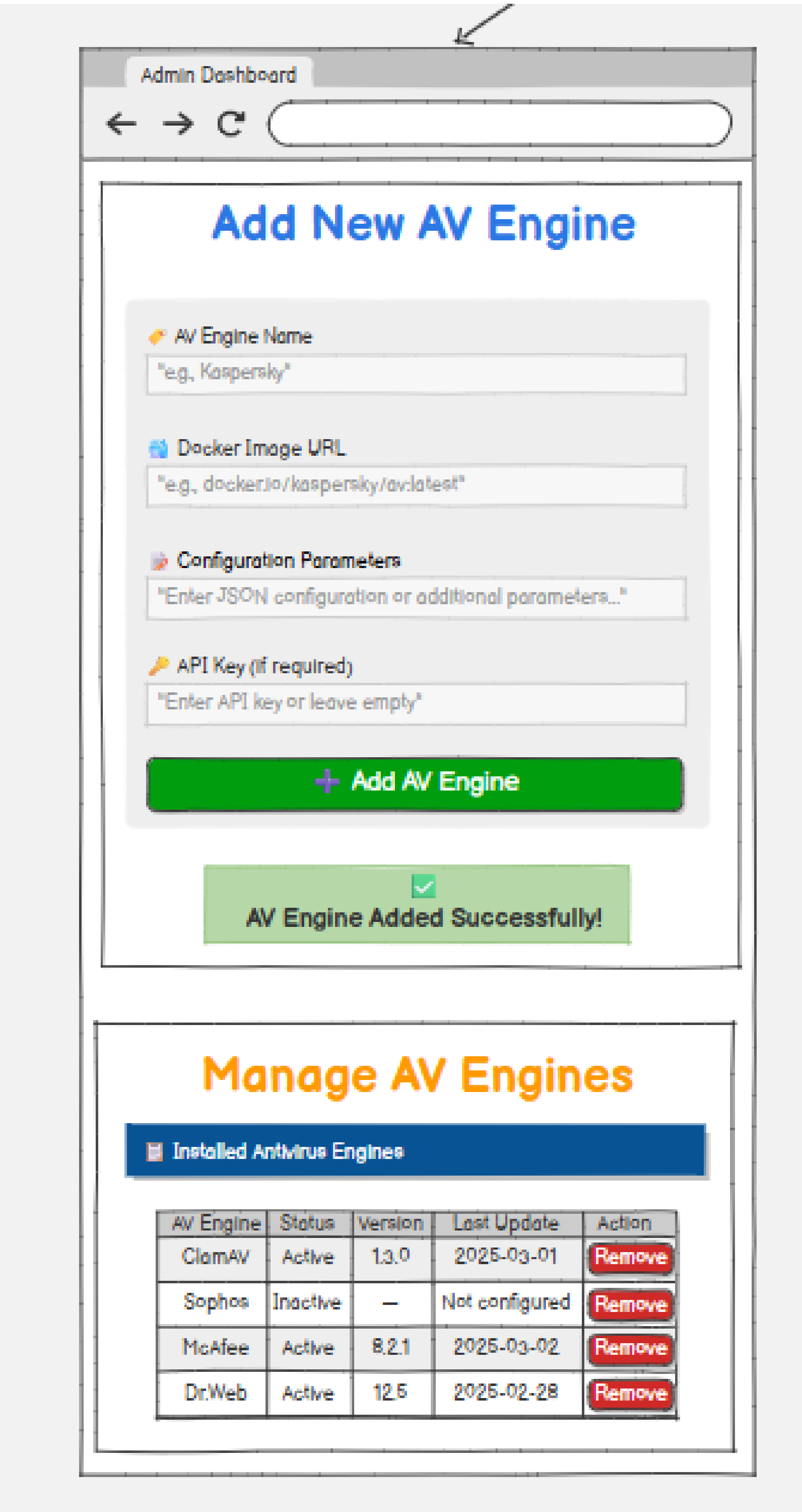
AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	—	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

TIBSA Platform - Graduation Project 2025-2026

Figure 4.10: Figure from document

The Login screen is the main entry point to the system. It allows registered users to authenticate using their email and password. The screen contains clearly labeled input fields for credentials and a primary login button to initiate authentication. Additional options such as '€œRemember Me'€ and '€œForgot Password'€ are included to enhance usability and account recovery. This wireframe focuses on simplicity and security by limiting distractions and guiding the user directly toward authentication.

Registration Screen



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

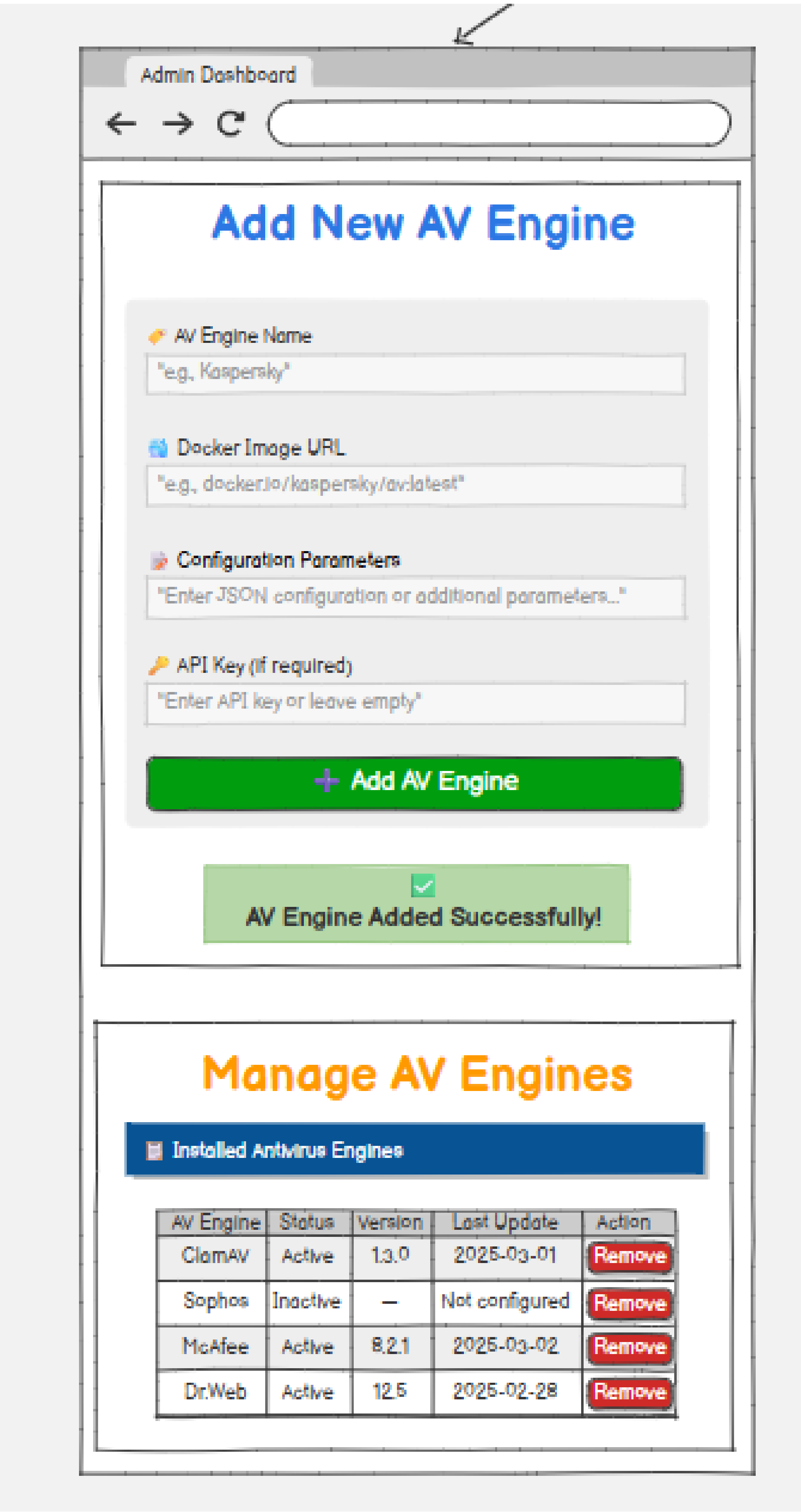
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.11: Figure from document

The Registration screen enables new users to create an account on the platform. It includes a structured form collecting essential information such as username, email address, and password. Validation indicators are used to guide the user during data entry. Upon successful registration, the user is redirected to the login process. This screen ensures a smooth onboarding experience while maintaining data consistency and security requirements.

Main Dashboard



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓
AV Engine Added Successfully!

Manage AV Engines

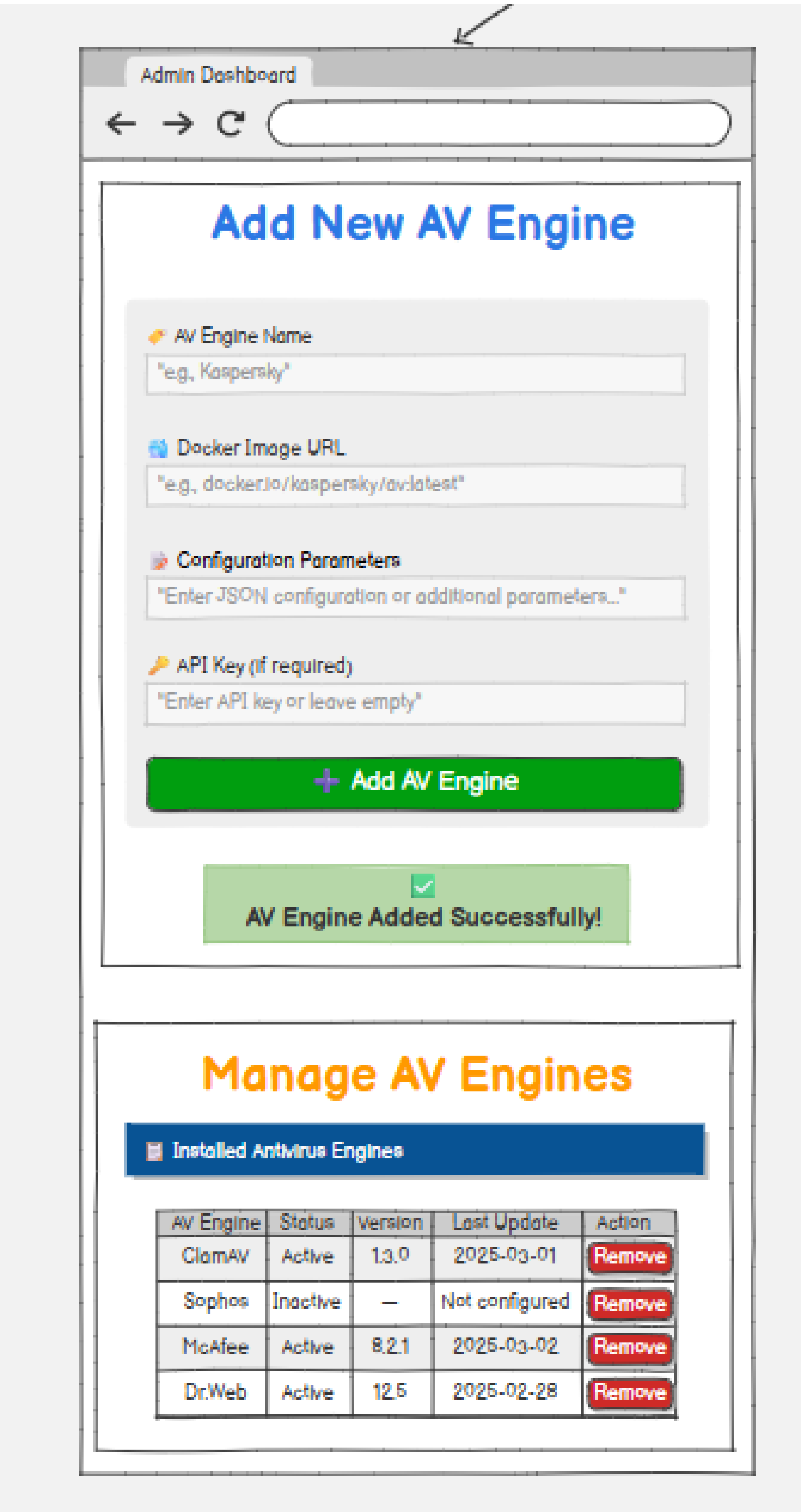
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	—	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.12: Figure from document

The Dashboard represents the central control panel of the system after successful login. It provides an overview of system status, recent scan activities, and engine statistics. Key information is displayed using summary cards, charts, and status indicators to allow users to quickly understand the current security state. Navigation elements are placed at the top or side to provide quick access to core system features such as scan management, engine configuration, and reports.

Available Antivirus Engines



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

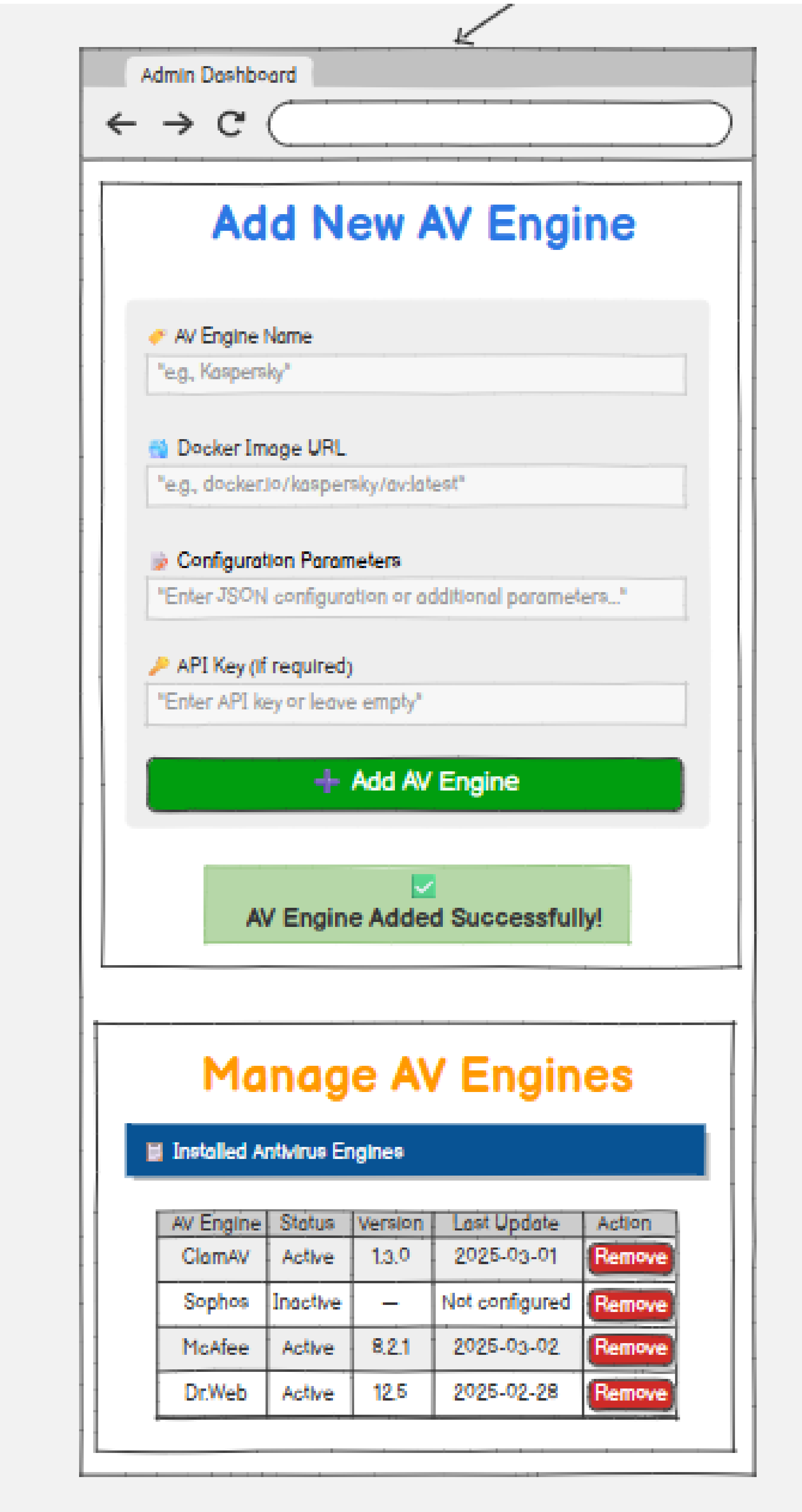
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.13: Figure from document

This screen displays a list of all available antivirus engines supported by the system. Each engine is presented with its current status, version, and last update time. Action buttons allow the user to enable, disable, or configure individual engines. The wireframe highlights modularity by allowing multiple engines to operate independently within the same system.

Update Antivirus Database



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

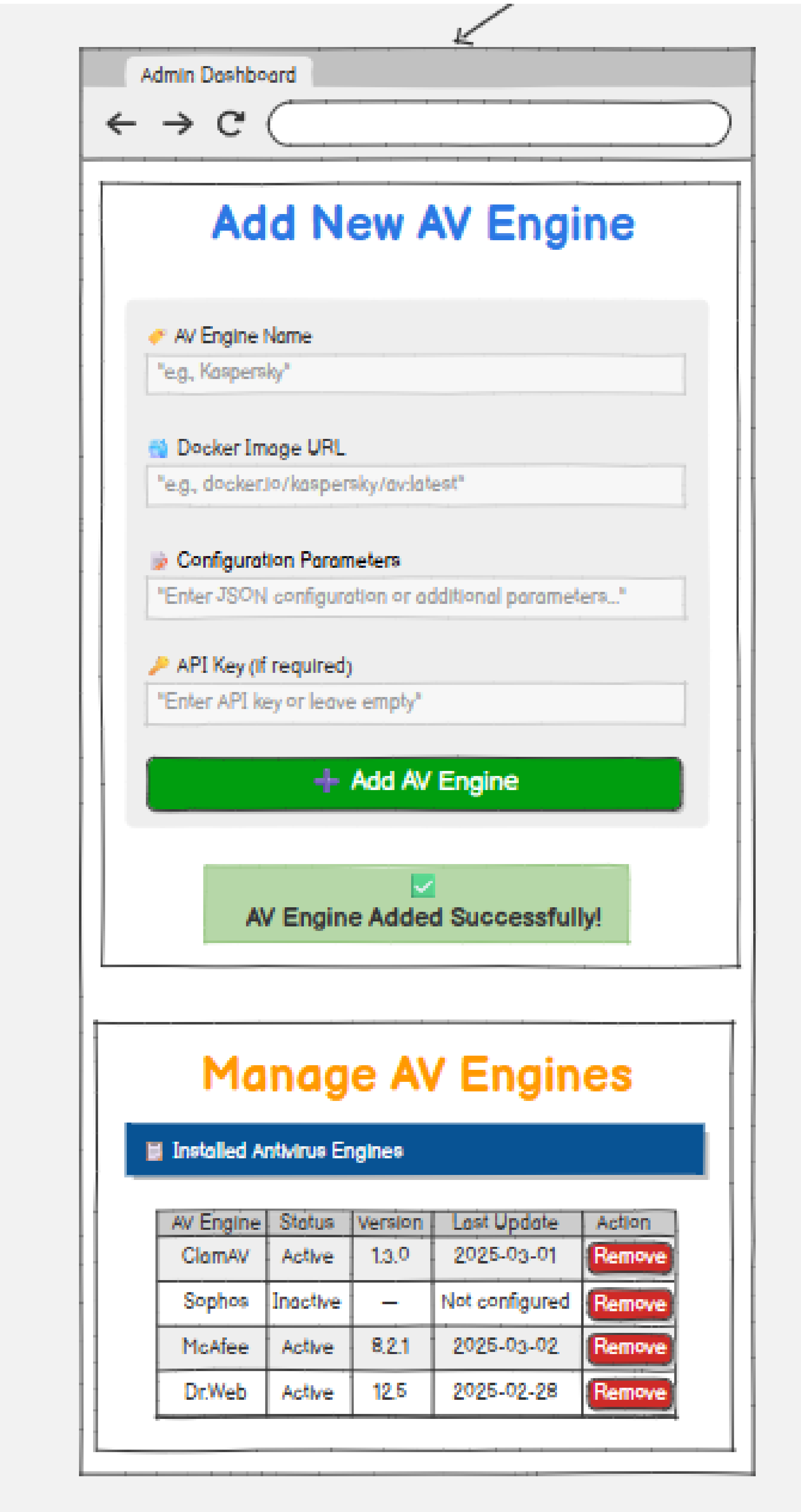
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.14: Figure from document

This screen provides functionality for updating antivirus signature databases. It displays update progress, current database version, and update history. A progress bar visually communicates update status, ensuring transparency during long-running operations. This screen is essential for maintaining detection accuracy and system reliability.

Add & Manage AV Engines



Add New AV Engine

AV Engine Name: "e.g., Kaspersky"

Docker Image URL: "e.g., docker.io/kaspersky/av:latest"

Configuration Parameters: "Enter JSON configuration or additional parameters..."

API Key (if required): "Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

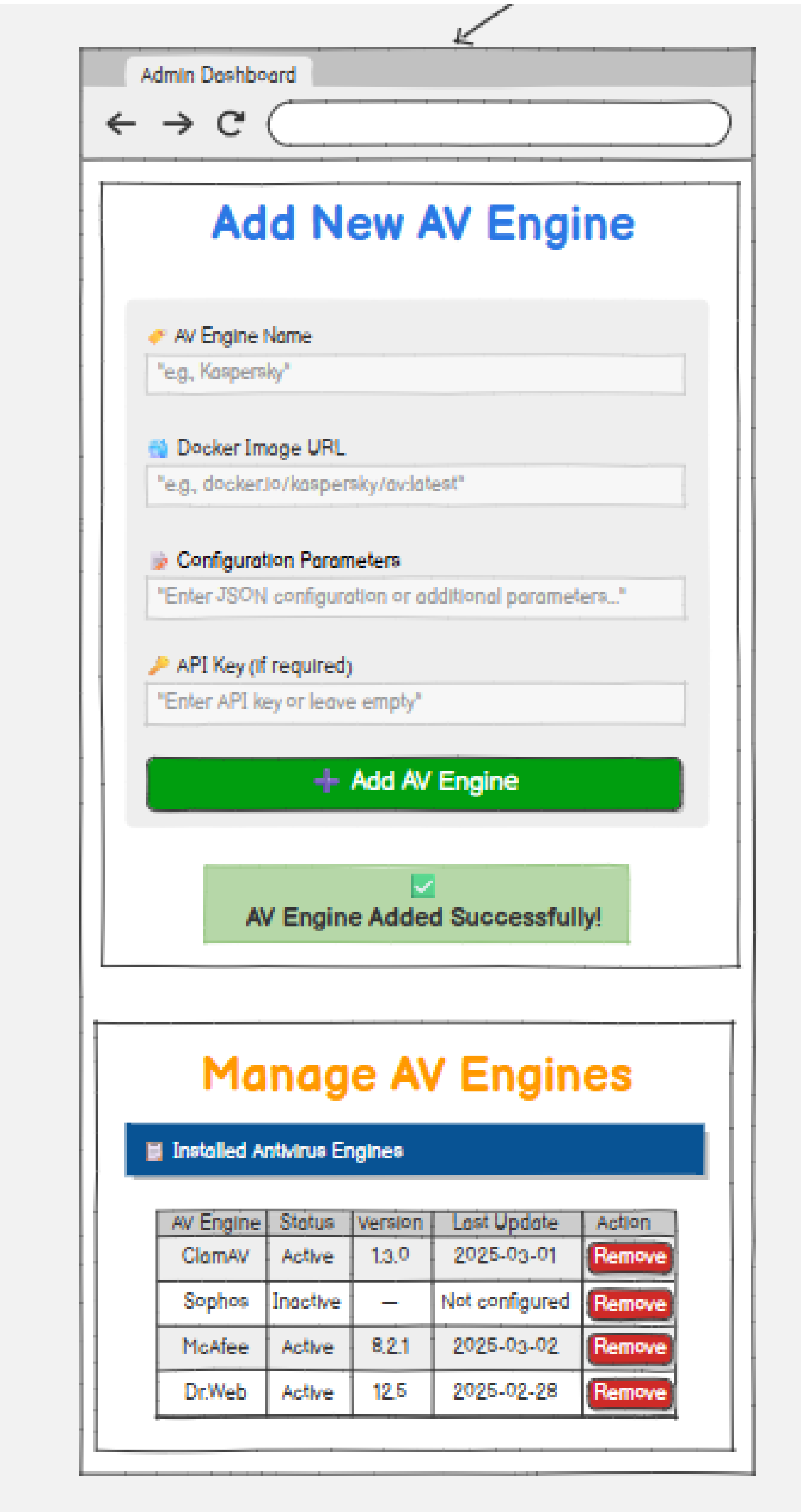
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	—	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.15: Figure from document

The Admin Dashboard page allows administrators to add and manage antivirus engines. The Add New AV Engine section includes input fields for the engine name, Docker image URL, configuration parameters, and an optional API key, with a prominent ’€œAdd AV Engine’€ button to submit the information. A confirmation message appears upon successful addition. The Manage AV Engines section displays a table of installed antivirus engines with columns for engine name, status, version, last update, and an action button to remove any engine. This interface provides a clear and centralized way to configure and maintain multiple antivirus engines.

File Upload & Scan Request



Add New AV Engine

AV Engine Name

"e.g., Kaspersky"

Docker Image URL

"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters

"Enter JSON configuration or additional parameters..."

API Key (if required)

"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

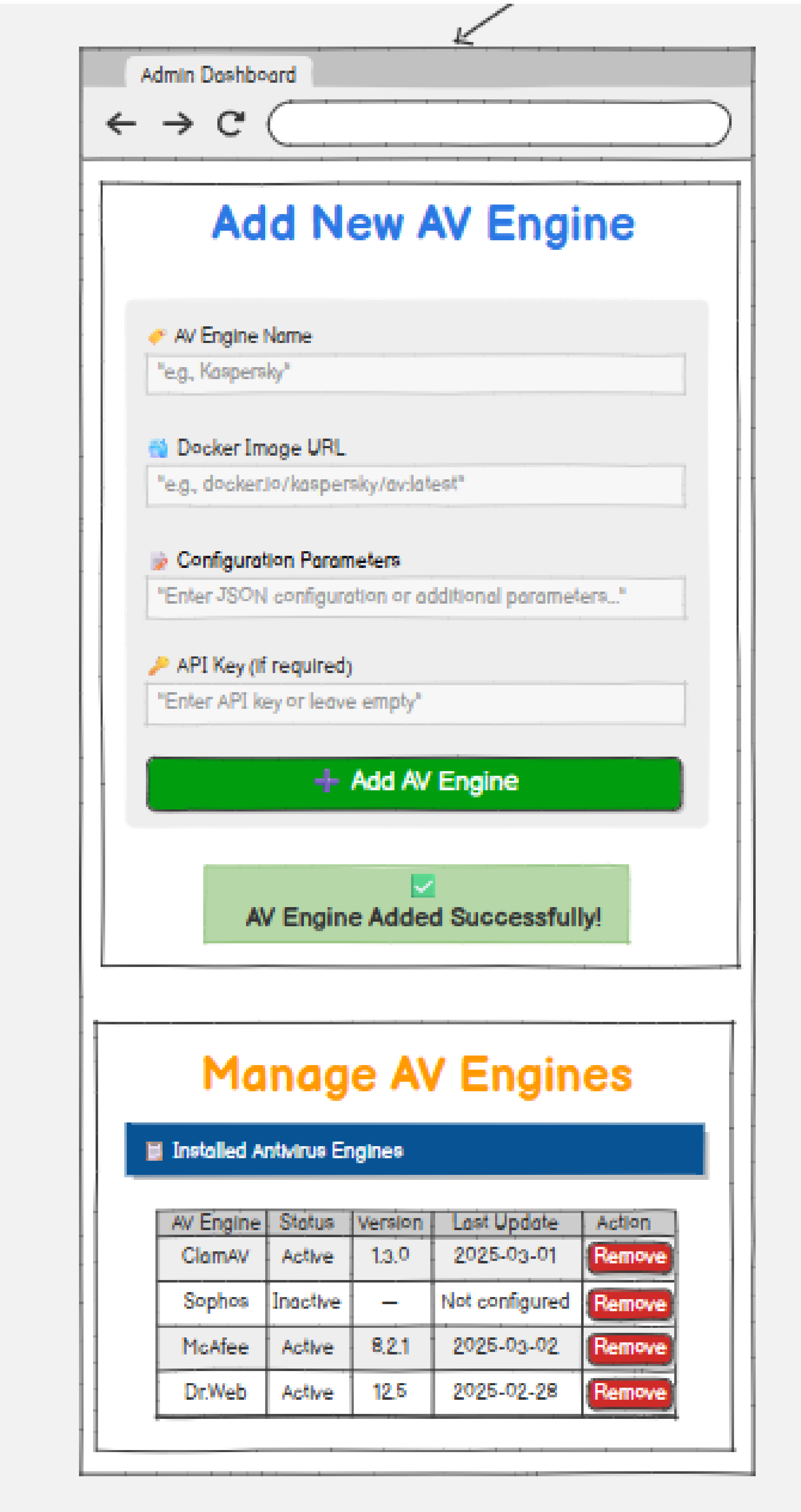
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.16: Figure from document

This screen is the primary interface for file security analysis. It includes a drag-and-drop zone for file uploads, with options to browse files, enter a URL, or submit a file hash. Users can upload multiple files for batch processing. Key buttons include "Analyze" to start scanning, "Get Info" for metadata, and "Check File Details." The top navigation bar provides links to Home, About, Docs, Services, and Profile, along with Login and notifications. The design focuses on flexibility and a clean, user-friendly experience.

4.1.15 Multiple File Scan

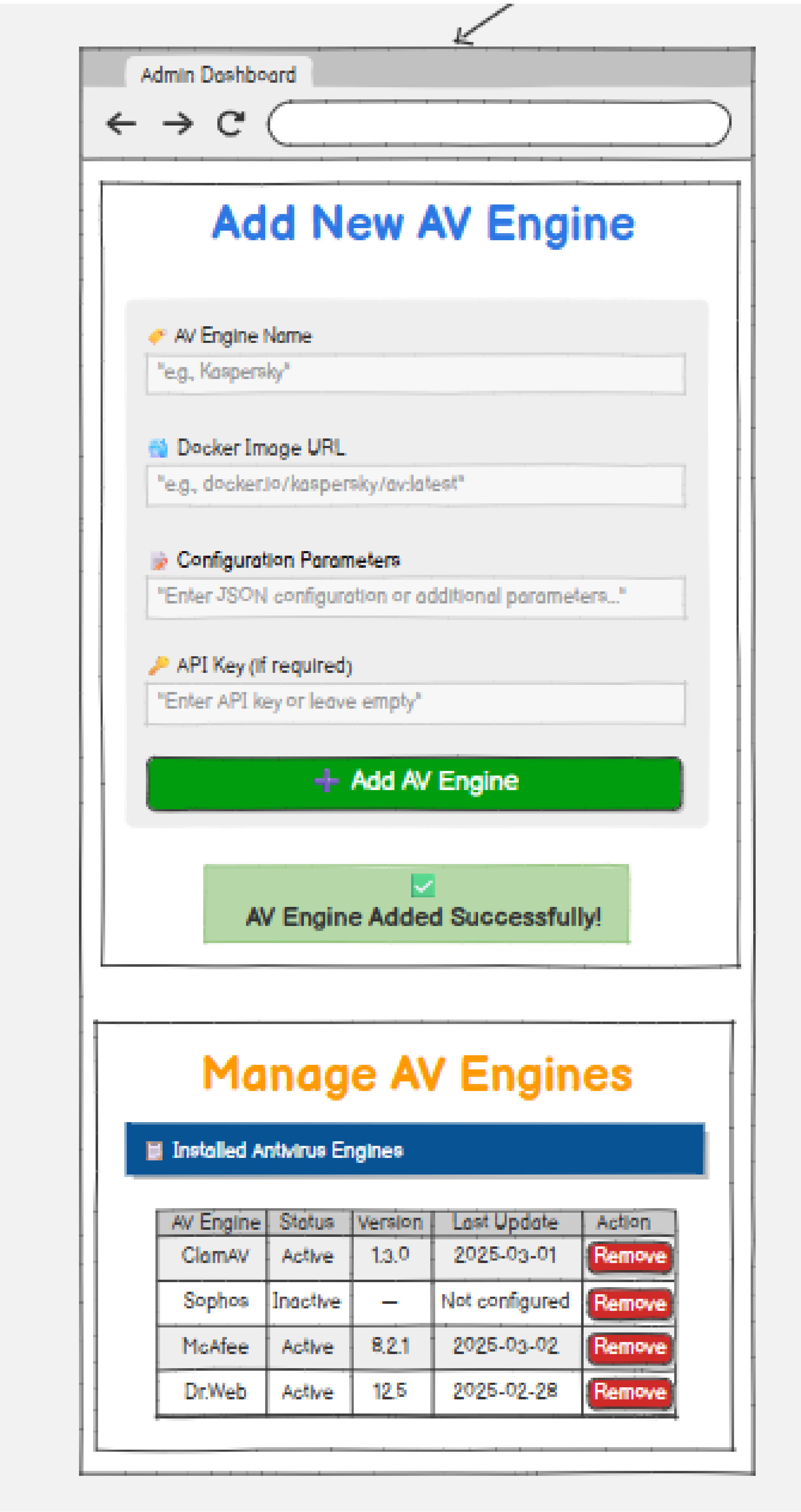

Admin Dashboard
Add New AV Engine
AV Engine Name
"e.g., Kaspersky"
Docker Image URL
"e.g., docker.io/kaspersky/av:latest"
Configuration Parameters
"Enter JSON configuration or additional parameters..."
API Key (if required)
"Enter API key or leave empty"
+ Add AV Engine
✓
AV Engine Added Successfully!
Manage AV Engines
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.17: Figure from document

This screen displays the results of scanning multiple uploaded files at once. The table lists each file by name, overall status (Malicious or Clean), detection ratio (number of engines that flagged it as a threat), and a "View Details" button for further information. A summary at the bottom indicates the total number of malicious files detected. Buttons at the top allow uploading additional files and starting the scan process. The layout provides a concise tabular overview to facilitate quick assessment of batch scan outcomes.

Scan Results Overview



Add New AV Engine

AV Engine Name

"e.g., Kaspersky"

Docker Image URL

"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters

"Enter JSON configuration or additional parameters..."

API Key (if required)

"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

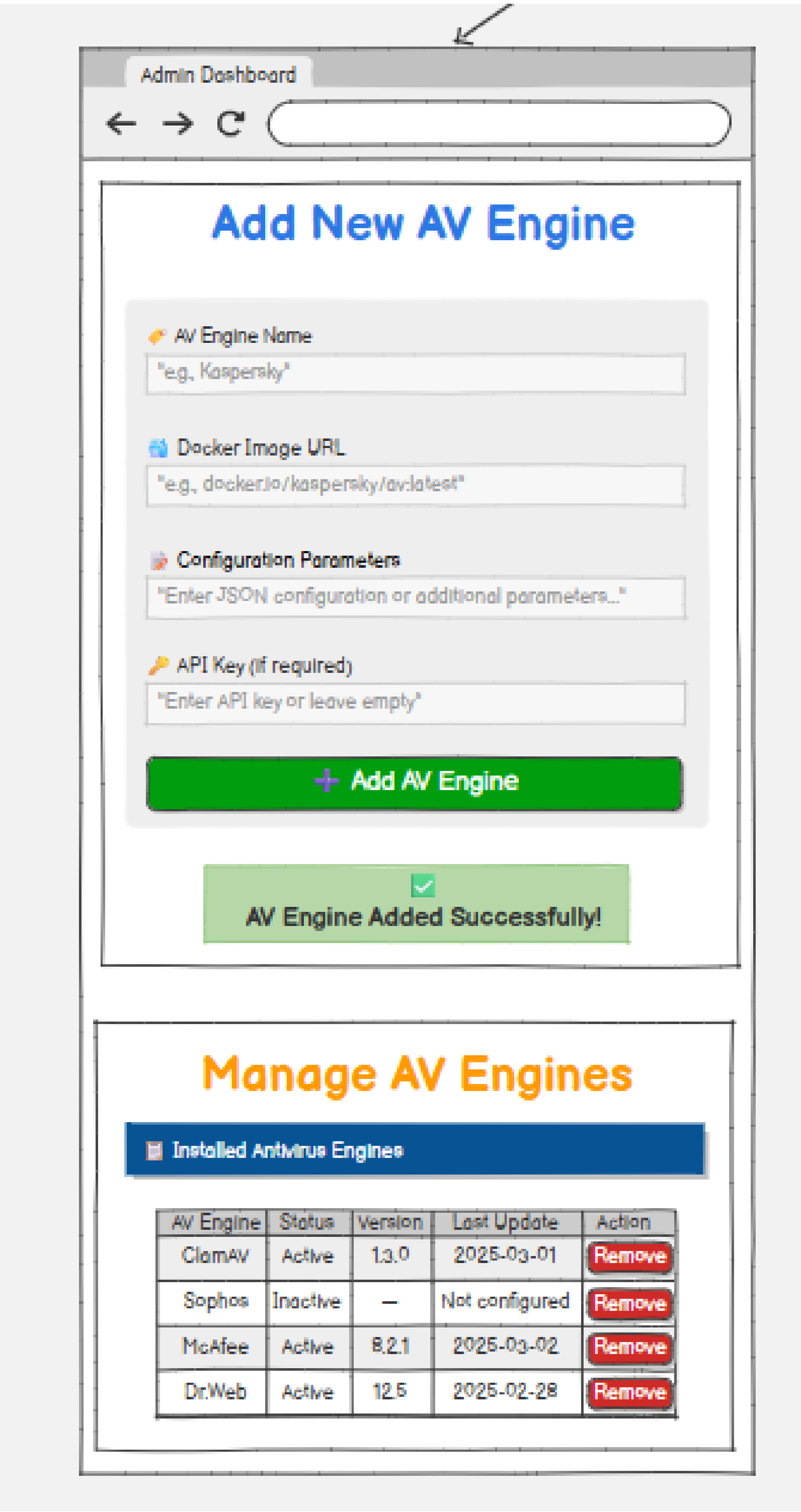
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	—	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.18: Figure from document

This screen shows a comprehensive scan report for the analyzed file. It displays file details including name, malicious status, size, and hash value. A table lists results from each antivirus engine with specific detection outcomes. Summary sections highlight the number of engines detecting the file as malicious and clean, along with the total engines used. Export buttons allow downloading the report as PDF or JSON. A timestamp at the bottom records the report generation date and time. The design offers a structured and clear view for thorough threat evaluation.

4.1.16 Detailed File Information



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

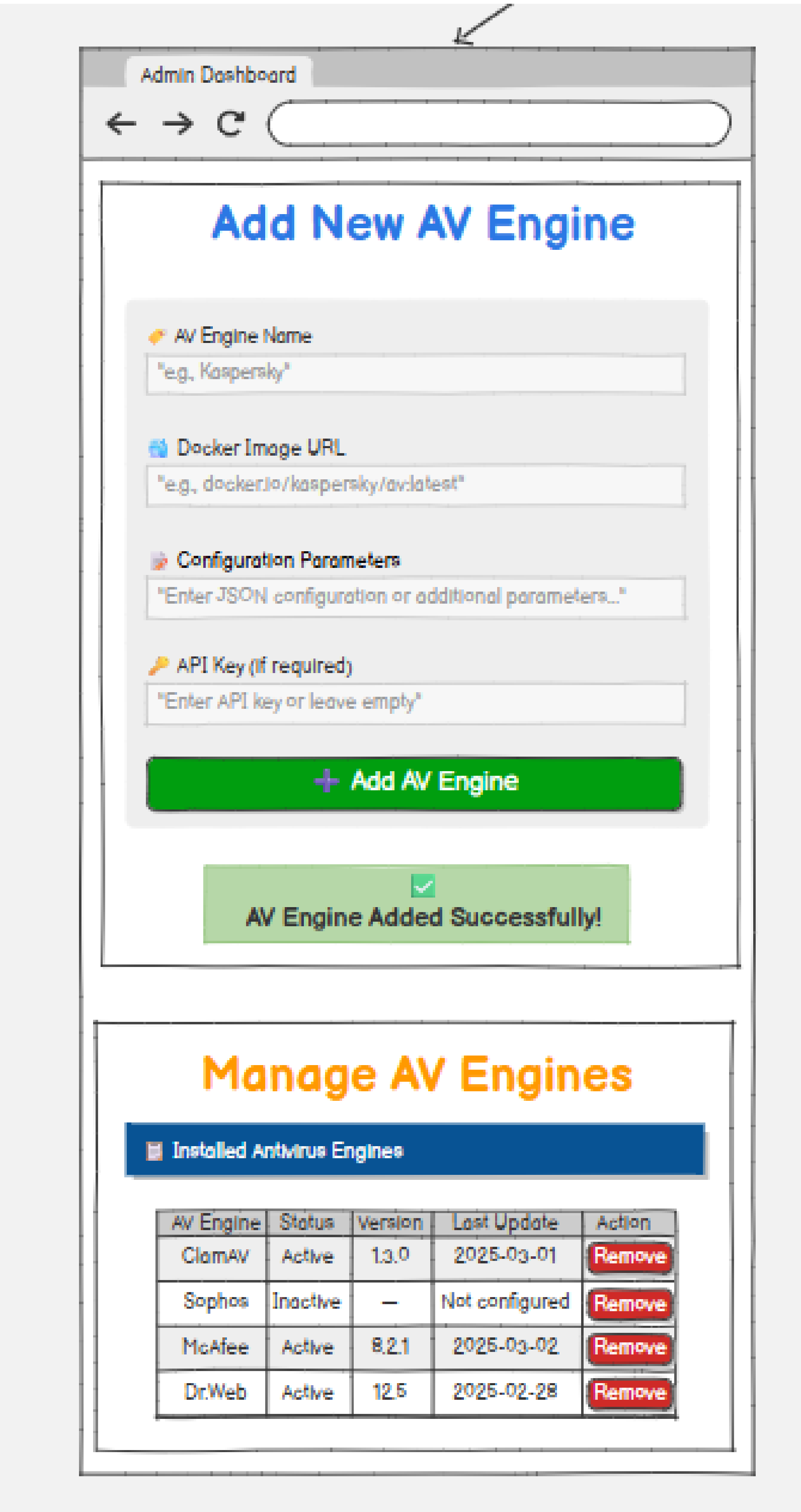
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.19: Figure from document

This screen shows detailed information for the scanned file, including name, size, MD5, SHA-1, SHA-256 hashes, extension, MIME type, and file type. Below, it lists detection results from various antivirus engines with color-coded indicators for threats like trojans, malware, and generic detections, or clean verdicts. Buttons at the bottom provide options for scan report export and sharing results. The layout offers clear metadata and per-engine insights for effective threat analysis.

4.1.17 Scan Report Sharing



Admin Dashboard

Add New AV Engine

AV Engine Name

"e.g., Kaspersky"

Docker Image URL

"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters

"Enter JSON configuration or additional parameters..."

API Key (if required)

"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

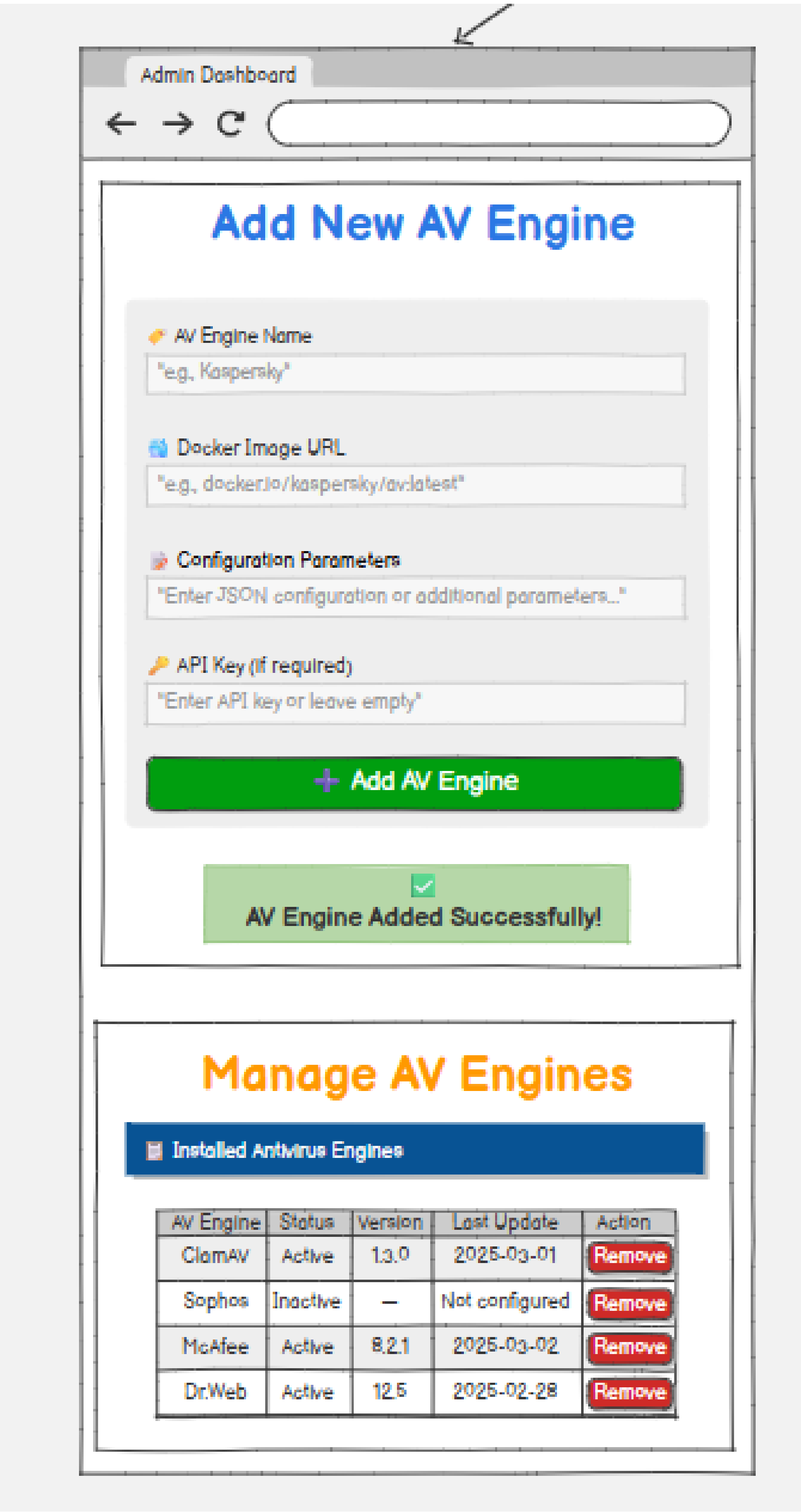
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.20: Figure from document

This screen allows users to share the scan results of a malicious file. It displays file details including name, status, and detection ratio. Options include generating a shareable link with a pre-filled URL and copy button, or sending results directly via email with fields for recipient addresses and an optional message. The layout provides secure and convenient sharing methods for collaboration or reporting.

4.1.18 Scan URL



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

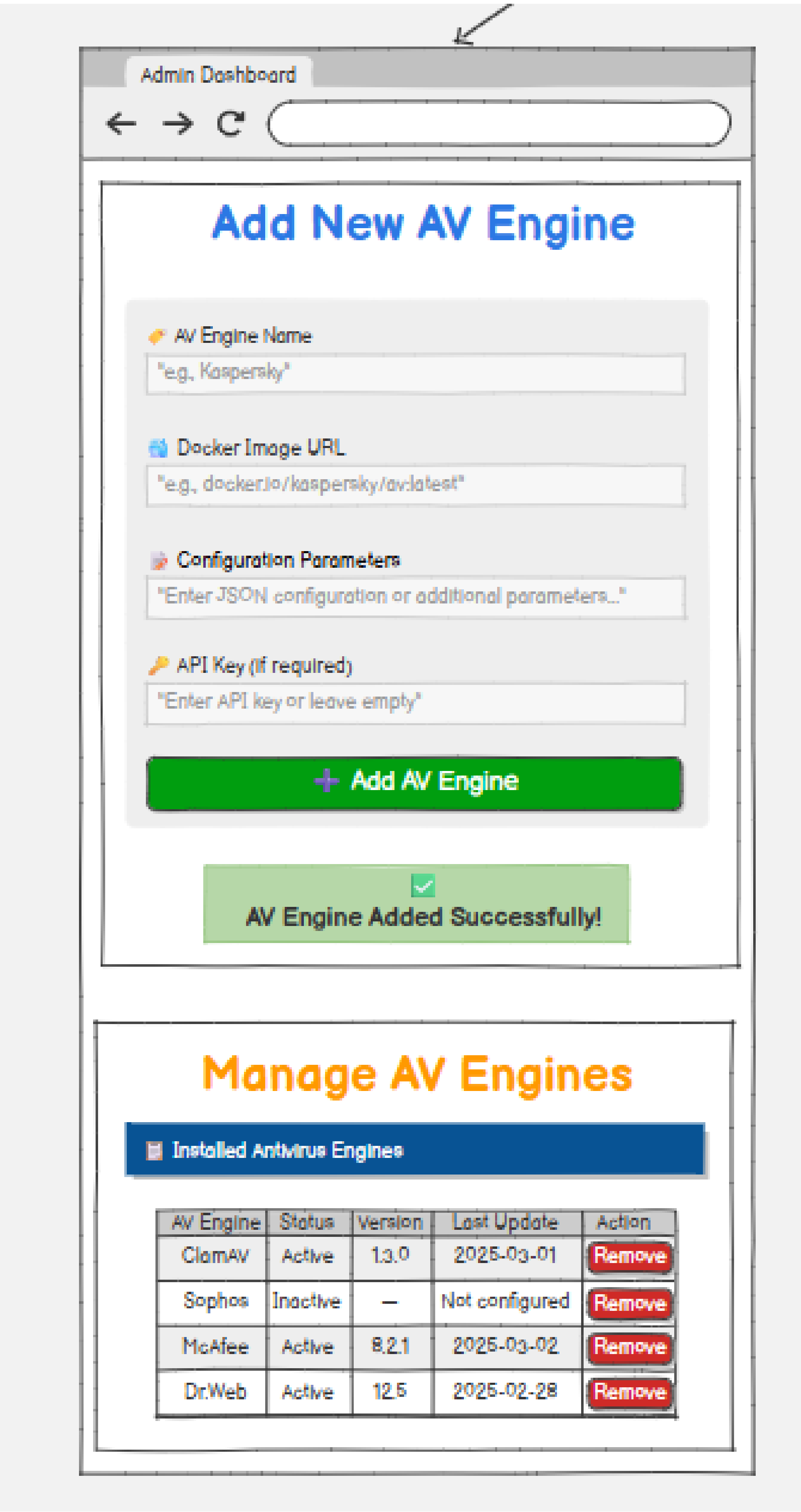
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.21: Figure from document

The URL Scan Interface allows users to submit web addresses for security analysis. It features a simple input field for entering the URL and a primary scan button to initiate analysis. A warning message informs users that files will be temporarily downloaded with a maximum size of 100 MB. Example URLs demonstrate proper format, and supported protocols (HTTPS, HTTP, FTP) are clearly displayed. This screen emphasizes simplicity and ease of use for quick URL verification.

4.1.19 User Profile



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

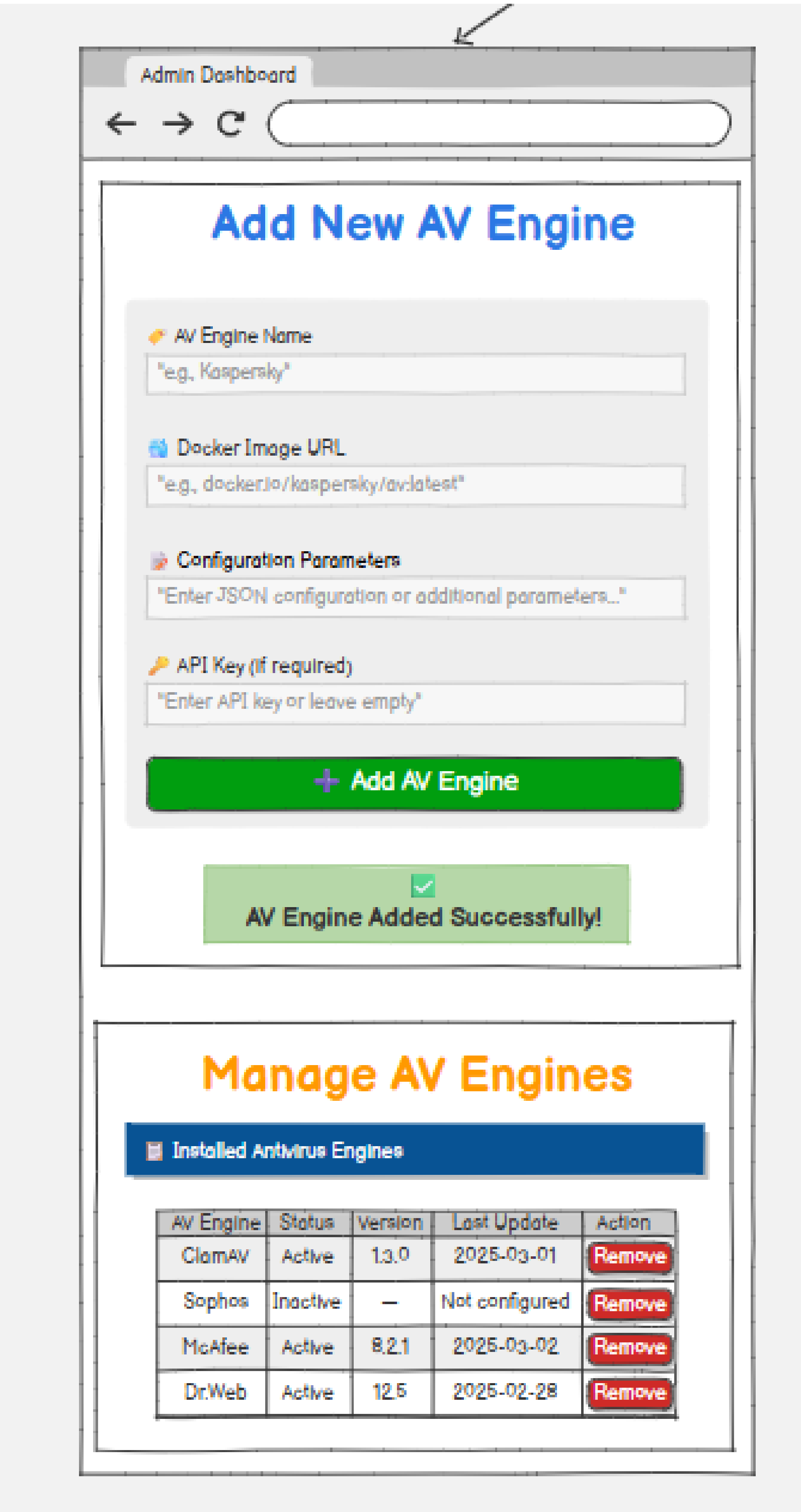
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.22: Figure from document

This screen displays the user's profile information on the left sidebar, including name, email address, and key statistics such as total scans performed, threats found, and clean files processed. Navigation menu options include Account Settings, View Dashboard, Scan History, Notifications, and a prominent Logout button. The layout provides a quick overview of user activity and easy access to account-related features.

4.1.20 System Settings



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

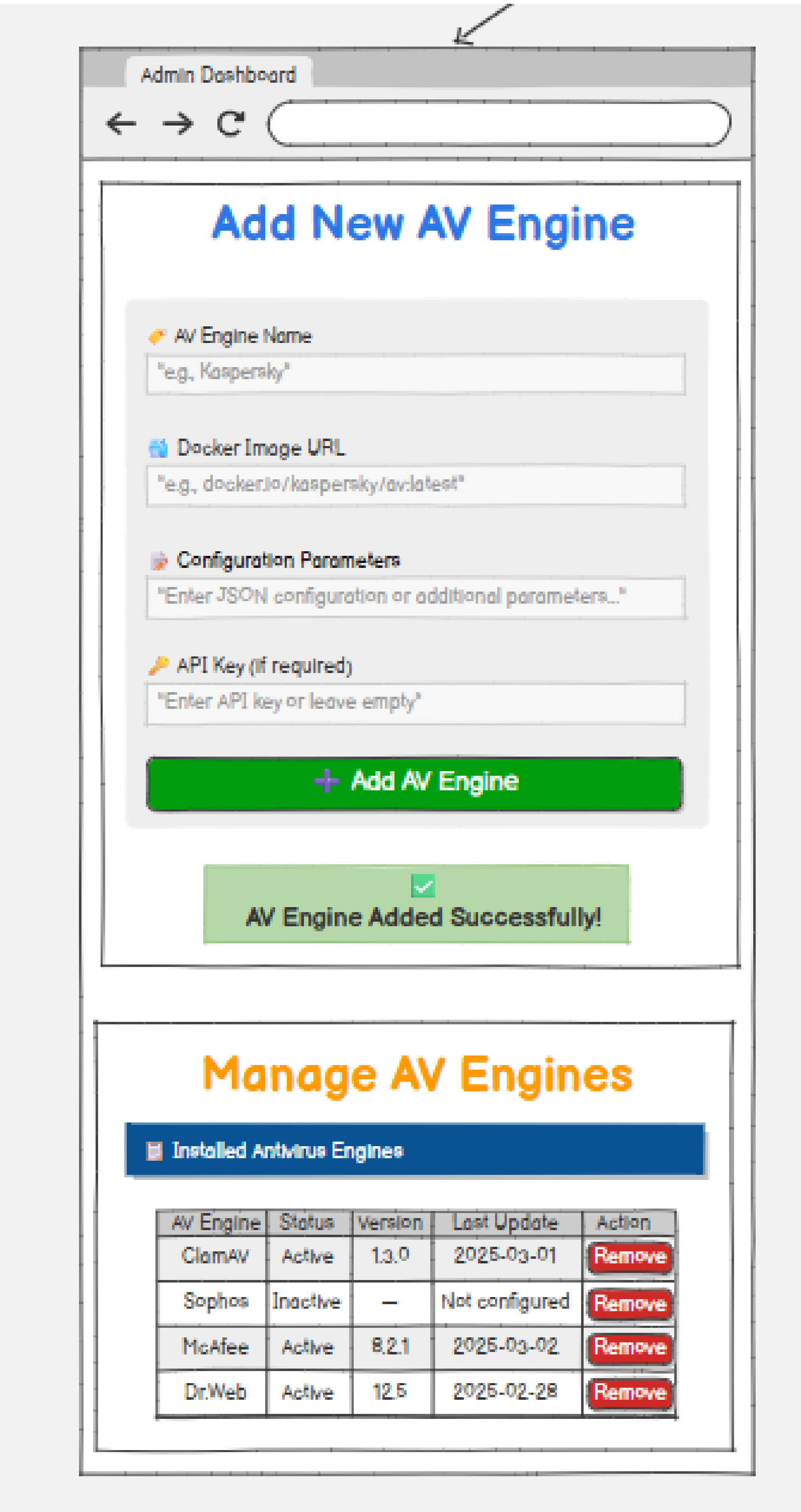
Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.23: Figure from document

This screen allows customization of user settings. It includes sections for selecting default antivirus engines with checkboxes, choosing appearance mode (light or dark), language selection, and notification toggles for email notifications, browser alerts, and update reminders. A "Save Preferences" button applies the changes. The design focuses on personalization to enhance the user experience.

4.1.21 Help & Support Screen



Add New AV Engine

AV Engine Name
"e.g., Kaspersky"

Docker Image URL
"e.g., docker.io/kaspersky/av:latest"

Configuration Parameters
"Enter JSON configuration or additional parameters..."

API Key (if required)
"Enter API key or leave empty"

✓ AV Engine Added Successfully!

Manage AV Engines

Installed Antivirus Engines

AV Engine	Status	Version	Last Update	Action
ClamAV	Active	1.3.0	2025-03-01	Remove
Sophos	Inactive	–	Not configured	Remove
McAfee	Active	8.2.1	2025-03-02	Remove
Dr.Web	Active	12.5	2025-02-28	Remove

Figure 4.24: Figure from document

This screen provides comprehensive user assistance. It features quick access tiles for Documentation, Video Tutorials, and Live Chat. A Frequently Asked Questions section lists common queries with expandable answers. The Contact Support form allows submission of name, email, subject, and message. Additional Resources include links to User Guide, API Documentation, Report a Bug, Feature Requests, Quick Start, News, Community, and Contact Us. The layout centralizes support options for efficient issue resolution and self-help.

4.1.22 Conclusion

This architecture provides a comprehensive, scalable, and secure platform for the TIBSA threat intelligence and modeling system. Each tool is strategically placed within its appropriate layer with clear data flows and integrations ensuring seamless operation. The architecture ensures security through multi-layered protection from Edge to Data layers, scalability through serverless and distributed components including Neon, Upstash, R2, and Cloudflare, reliability through background job processing, retry logic, and comprehensive monitoring, performance through caching, CDN, load balancing, and async processing, observability through comprehensive logging, metrics, and alerting, and compliance through audit trails, RBAC, and secure authentication. The clear separation of concerns and well-defined interfaces between layers enable maintainability, testability, and future extensibility of the platform.

Chapter 5

Conclusion and Future Work

5.1 Summary

This project proposes a **unified, intelligence-driven security assessment framework** that bridges predictive analysis and practical validation.

5.2 Future Work

Future enhancements will be documented here.

Bibliography

[1] D. S. Xu, B. S. Chaparro, X. Ou, and P. Rao, "Automated Security Test Generation with Formal Threat Models," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 232–244, 2013.

[2] B. Marback, C. Doerr, P. Rao, B. S. Chaparro, and X. Ou, "A Threat Model-Based Approach to Security Testing," *Software: Practice and Experience*, vol. 43, no. 2, pp. 241–258, 2013.

[3] R. Palanivel and A. P. Selvadurai, "Risk-Driven Security Testing Using Risk Analysis with Threat Modeling Approach," *SpringerPlus*, vol. 3, no. 754, pp. 1–11, 2014.

[4] F. A. Rahim, N. Jamil, Z. C. Cob, L. M. Sidek, and N. I. I. Sharizan, "Risk Analysis of Water Grid Systems Using Threat Modeling," *Journal of Physics: Conference Series*, vol. 2319, no. 1, 2022.

[5] C. E. Alozie, "Threat Modeling for Healthcare Cybersecurity: A Comparative Analysis of STRIDE, PASTA, and Attack Trees," *Journal of Cybersecurity Research*, vol. 8, no. 2, pp. 45–62, 2024.

[6] D. Granata and M. Rak, "A Comparative Study of Open-Source Automated Threat Modeling Tools," in *Proc. IEEE International Conference on Cloud Computing Technology and Science*, pp. 156–163, 2020.

[7] B. Shin, D. Elkins, E. Larson, R. Perez, and G. Cameron, "Actionable Intelligence-Oriented Cyber Threat Modeling Framework," in *Proc. 55th Hawaii International Conference on System Sciences (HICSS)*, pp. 6782–6791, 2022.

[8] M. Dekker and L. Alevizos, "A Threat-Intelligence Driven Methodology to Incorporate Uncertainty in Cyber Risk Analysis and Enhance Decision Making," *Computers & Security*, vol. 127, 2023.

[9] J. Smith, R. Johnson, and L. Williams, "Predictive Behavioral Mapping for Ransomware Detection Using Machine Learning," *Journal of Information Security and Applications*, vol. 74, 2023.

[10] B. Bin Sarhan and N. Altwaijry, “Insider Threat Detection Using Machine Learning Approach,” *Applied Sciences*, vol. 13, no. 259, pp. 1–19, 2023.

[11] M. Alharbi, E. Al-Shaer, and N. Almakhdhub, “Ethical Hacking: Threat Modeling and Penetration Testing a Remote Terminal Unit,” KTH Royal Institute of Technology, Stockholm, Sweden, 2020.

[12] G. Chu, “Automation of Penetration Testing,” Ph.D. dissertation, University of Liverpool, Liverpool, U.K., Sept. 2021.

[13] J. Huang and Q. Zhu, “PenHeal: A Two-Stage LLM Framework for Automated Pentesting and Optimal Remediation,” *arXiv preprint*, vol. abs/2407.17788, 2024.

[14] K. Chauhan, “Insider Threats Mitigation: Role of Penetration Testing,” Department of Computer Science, University of Guelph, Guelph, Canada, 2024.

[15] D. P. R. Sanagana, “Mitigating Network Threats: Integrating Threat Modeling in Next-Generation Firewall Architecture,” *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 8, 2023.

[16] S. P. Maniraj, C. S. Ranganathan, and S. Sekar, “Securing Web Applications with OWASP ZAP for Comprehensive Security Testing,” *International Journal of Advanced Signal and Image Sciences*, vol. 10, no. 2, pp. 12–23, 2024.