A Software Requirements Specification on

VPN and Proxy Detector

Submitted in Partial Fulfillment for the Award of Degree of Bachelor of Technology in Computer Science and Engineering from Rajasthan Technical University, Kota

MENTOR:

Sneha Sharma

(Dept. of Computer Science & Engineering)

COORDINATOR:

Sumit Mathur

(Dept. of Computer Science & Engineering)

SUBMITTED BY:

Kartik Khorwal (22ESKCS112)

Jitender Kumar (22ESKCS108)

Hitesh Tank (22ESKCS105)

Khushang Ameta (22ESKCS115)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SWAMI KESHWANAND INSTITUTE OF TECHNOLOGY, MANAGEMENT & GRAMOTHAN

Ramnagaria (Jagatpura), Jaipur – 302017

SESSION 2025-26

Table of Contents

Chapter-1

Introduction

1. Introduction

This Software Requirements Specification (SRS) introduces the **VPN and Proxy Detector** system, a high-security, **self-hosted web application** designed to address a critical challenge faced by law enforcement and security analysts. The core purpose is to instantly and verifiably classify any submitted IPv4/IPv6 address or domain, determining if the endpoint is masked by an anonymizing service, such as a proxy or VPN. The necessity for this system stems from the escalating use of these technologies by cyber-offenders, which critically hinders official investigations by causing **delays and false leads**. The system employs a robust, **multi-tiered detection strategy** that includes **Passive CIDR checks** against known threat feeds, **Active TCP/UDP port probing** on proxy ports, and essential **WHOIS/RDAP data enrichment**. Crucially, its **self-hosted architecture** guarantees **data sovereignty** and includes a mandatory **Audit Log (FR-007)** for every lookup, providing the necessary institutional compliance and integrity that commercial APIs cannot offer. The system is developed using a **Node.js/React stack** and is engineered to meet aggressive performance targets, specifically a warm lookup latency of **$<$150 ms (NFR-001)**.

1.1 Purpose

The development of the **VPN and Proxy Detector** project strictly follows the **Agile Software Development Methodology**, specifically the **Scrum framework**, chosen for its ability to manage external dependencies and prioritize performance. This iterative approach ensures the system is built in incremental stages, allowing for continuous gathering of performance metrics and rapid incorporation of feedback, which is crucial for a threat intelligence-based application.

The initial phase involved a detailed **requirements analysis** to define the stringent security (NFR-002) and latency (NFR-001) needs of law enforcement analysts. Based on this, the system design was prepared, outlining the three-tier **Client-Server Architecture**, E-R data models, and the interface for single/bulk lookups. Implementation is executed through iterative sprints, with each sprint delivering a core, verifiable module, such as the **Redis caching layer** (for speed) or the **WHOIS/RDAP client** (for data enrichment).

For backend efficiency, the system utilizes **Node.js 18 + Express 5** to handle high-throughput API traffic, and MongoDB is used for persistent **Audit Logging (FR-007)**. The detection process is designed as a sequential, multi-layered service that handles **Passive CIDR checks** before escalating to **Active Port Probes**. On the frontend, a **React.js/MUI** interface ensures an intuitive, responsive experience across devices for submitting both single and bulk inputs.

Quality assurance is maintained through continuous **Jest and Supertest** integration testing for every core service, verifying that the bulk processing rate meets the **≥1,000 lookups per minute (NFR-003)** standard. Final deployment using **Docker Compose** ensures a reliable, easily managed production environment with high uptime (NFR-004) and robust technical documentation.

1.2 Need/Motivation

The motivation for this system is rooted in the critical and escalating challenge of **digital anonymity** utilized by malicious actors across various online sectors. Cyber-offenders deliberately employ Virtual Private Networks (VPNs), proxy servers, and the Tor network to **mask their true IP addresses and physical locations** during fraudulent transactions, unauthorized access attempts, and other criminal activities. This concealment severely impedes the efficiency and effectiveness of law enforcement, security monitoring, and forensic investigation teams, resulting directly in **investigation delays** and the pursuit of **false or misleading leads**.

Existing solutions often involve reliance on commercial, third-party API services, which, despite their accuracy, present significant risks and limitations. These platforms necessitate the external transfer of sensitive IP and network data, compromising **data sovereignty** and breaching institutional requirements for **chain-of-custody** and security compliance. Furthermore, reliance on external APIs introduces variable costs and strict **rate limits** (NFR-005), making large-scale **bulk processing** (FR-006) impractical or cost-prohibitive for government and enterprise security teams.

The primary motivation behind this project is to create a **customized, self-hosted, and high-security platform** that restores control and integrity to the investigative process. By implementing a **self-hosted architecture** with **zero runtime dependencies** outside the core stack, the system ensures that all sensitive data remains within the organizational network. A key feature is the mandated **Audit Log (FR-007)**, which records every single transaction, thereby establishing verifiable proof for compliance and legal processes. Ultimately, this project is motivated by the vision of providing analysts with a reliable, performance-driven tool capable of delivering immediate, auditable IP provenance data, strengthening organizational security posture and promoting digital justice.

Chapter-3

Market Survey

2. Market Survey

The demand for robust, real-time IP anonymity detection has grown significantly in recent years due to the increasing sophistication of cyber-offenses and the regulatory necessity for verifiable digital identity. Security analysts and law enforcement personnel today prefer transparent, auditable detection methods where they can directly verify IP provenance and receive instantaneous, contextual results. This shift away from proprietary, black-box third-party services toward **integrated, high-security, and compliance-driven solutions** has created a strong need for platforms that combine verifiable data, performance, and institutional control. At the same time, investigative bodies are searching for solutions that not only provide accurate detection but also allow seamless **bulk processing (FR-006)** and maintain a full **Audit Trail (FR-007)** for effective litigation support.

In the current technology landscape, several commercial and open-source platforms offer IP detection, such as MaxMind, IPinfo, and various threat intelligence feeds. These platforms provide large datasets and high-accuracy APIs, which are excellent for general web traffic filtering. However, they introduce significant **security and compliance risks**. Specifically, reliance on external services requires the transmission of sensitive IP data outside the secure institutional perimeter, violating **data sovereignty** and **chain-of-custody** requirements. Investigative agencies currently rely on complex, manual cross-referencing of blocklists or expensive, low-throughput commercial APIs, creating a gap in the market for a high-performance, **self-hosted platform** that balances security with operational functionality.

From an operational perspective, analysts require a platform that offers straightforward input via single and bulk forms, clear visualization of the **multi-tiered detection process**, and immediate feedback on the **WHOIS/RDAP enrichment**. They expect robust support for both IPv4 and IPv6 normalization (FR-009) and highly performant APIs that work with external scripts and internal tools. On the other hand, administrators require the ability to ensure the system's **reliability (NFR-004)**, secure API access via **JWT (NFR-002)**, and track all transaction history via automated reports and filters (FR-007). The combination of these stringent security and operational requirements validates the need for a detection platform that is not only high-speed but also **institution-oriented** and fully auditable.

Industry standards for security systems also highlight the importance of **scalability (NFR-003)**, **low latency (NFR-001)**, and **unbiased accuracy** in detection logic. The automated detection must be robust against evasion, reliable across both static CIDR lists and live probe checks, and highly performant to handle **≥1,000 lookups per minute**. Furthermore, the system must address the crucial security constraint of **WHOIS rate limiting (NFR-005)**. A platform that delivers these features in a secure, self-hosted package tailored for investigative workflow can decisively bridge the gap between existing public detection solutions and sensitive institutional requirements. The analysis of current trends, user needs, and external solution limitations makes it clear that there is strong potential for a **VPN and Proxy Detector** designed specifically for high-security, auditable forensic use.

Chapter-3

2.1 Objectives

The primary objective of this project is to design and deploy a high-security, **self-hosted detection platform** that enables authorized analysts to instantly verify the true provenance of any submitted IP address or domain. The system aims to provide a **multi-tiered analytical environment** where endpoints are classified through **Passive CIDR checks** and confirmed via **Active TCP/UDP Port Probing (FR-003)**, helping investigators rapidly identify concealed identities and overcome investigative roadblocks. By supporting robust validation and providing clear, composite verdicts, the platform ensures that users at different security clearance levels can benefit from consistent, high-confidence results.

Another paramount objective is to empower institutional security teams with auditable data and superior operational performance. The system must meet the stringent performance metrics, achieving a warm lookup latency of **$<$150 ms (NFR-001)** and handling asynchronous **Bulk Processing (FR-006)** at a minimum rate of **≥1,000 IP lookups per minute (NFR-003)**. Administrators will gain the ability to ensure API security via **JWT authentication (NFR-002)** and maintain a complete record of every transaction through the mandatory **Audit Log (FR-007)**. This not only promotes operational efficiency but also guarantees **data sovereignty** and **chain-of-custody** compliance in investigative reporting.

Additionally, the project seeks to provide granular **WHOIS/RDAP enrichment** for every lookup, offering valuable network context that allows analysts to verify data center hosting or registrar flags. By integrating features such as real-time bulk job status via **Socket.io** and comprehensive history export, the system bridges the gap between static threat intelligence and practical, verifiable forensic application. Ultimately, the core objective is to ensure analysts are better equipped to counter digital anonymization and strengthen overall digital security posture.

3. Requirements

Every software system is built on a clear understanding of its anticipated performance and operational mandate, and the **VPN and Proxy Detector** system is no exception. The requirements for this high-security platform were rigorously identified through an analysis of **law enforcement investigative workflows**, institutional needs for **data sovereignty**, and the stringent metrics of external detection services. These requirements form the foundational commitment of the system, ensuring that the final product is not only functionally accurate but also reliably integrates into sensitive security infrastructures.

Chapter-3

The functional requirements (FRs) describe the specific operations that the system must perform, such as secure endpoint validation, multi-tiered detection logic, asynchronous bulk processing, and auditable logging. These directly define the verifiable capabilities of the system upon deployment. In addition to these, the non-functional requirements (NFRs) ensure that the system not only classifies correctly but also performs with exceptional efficiency. Critical aspects such as **sub-150ms latency (NFR-001)**, **security (NFR-002)**, **scalability (NFR-003)**, and **reliability (NFR-004)** have been meticulously defined to make the platform suitable for continuous, high-volume investigative use.

Furthermore, the requirements also extend to detailed hardware and software specifications, which are essential for supporting the extreme performance demanded by the classification model and the high-speed Redis caching layer. Since the platform involves near **real-time detection and verification**, it mandates a secure and robust environment that can handle concurrent lookups without compromising the stringent latency targets. The development process is guided by the **Agile Scrum methodology**, allowing the technical requirements and performance goals to be refined in iterative stages, adapted based on continuous performance testing and external threat intelligence updates. By clearly defining these requirements, the project ensures the final system meets the expectations of analysts and security administrators, while maintaining flexibility for future advanced detection enhancements.

3.1 Functional Requirements

The proposed system must provide all essential functions that allow analysts and administrators to interact with the platform effectively, prioritizing compliance and robust data handling. The first functional requirement (FR-001) is that the system must support secure input handling and acceptance for a single IPv4/IPv6 address or domain via a dedicated UI or API endpoint. This input handling ensures that only authenticated users (NFR-002) can submit data, with role-based access providing appropriate privileges to analysts versus administrators. Security and integrity at this level are crucial because the platform handles sensitive investigative data.

Another paramount functional requirement is the **multi-tiered detection methodology**. The system must first perform **Passive CIDR checks (FR-002)** against curated threat feeds, followed by mandatory **WHOIS/RDAP data retrieval (FR-004)** for enrichment. If initial passive checks are inconclusive, the system must escalate to **Active TCP/UDP Port Probing (FR-003)** on common proxy ports (80, 443, 1194) to confirm anonymity indicators. This layered verification guarantees a high confidence score for the final classification verdict.

The system must also allow analysts to submit large amounts of data via **Bulk CSV Upload (FR-006)**, which initiates an asynchronous processing task managed by a job queue. Once submitted, the system must execute lookups efficiently, satisfying the minimum **≥1,000 IP**

lookups per minute (NFR-003) rate. Analysts must be able to view results immediately in the UI and via API, including the composite verdict, breakdown of checks, and enriched WHOIS data (FR-005). This instant feedback is essential for maintaining investigative momentum.

The platform should further support the automatic, non-repudiable logging of all submitted transactions via the **Audit/History Module (FR-007)**. This functionality guarantees that solutions are judged fairly and consistently without manual intervention and ensures a complete **chain-of-custody** record. The results of the audit log must be filterable and exportable for legal reporting.

3.2 Non Functional Requirements

In addition to the core detection and auditing capabilities, the proposed system must fulfill several non-functional requirements to ensure it is secure, highly performant, and reliable for institutional use. One of the most important NFRs is **Security (NFR-002)**, mandating that the platform enforce **HTTPS/TLS encryption** across all API endpoints and use robust **JWT/API-Key authorization** to protect sensitive investigative data and prevent unauthorized access.

Another critical requirement is **Performance (NFR-001)**, specifically the low latency target. Since the platform is intended for real-time analysis, the system must return a verdict in **$<$150 ms** from the warm cache and **$<$500 ms** from the cold cache. Any delay in reflecting results or processing bulk jobs can severely compromise the investigative value of the data. Therefore, performance optimization, driven by the **Redis Caching Layer**, plays a key role in ensuring smooth and high-throughput interactions.

The platform must also maintain a dashboard that is both responsive and user-friendly **(NFR-005 - Usability)**. Analysts and administrators should find it easy to navigate through the system, submit requests, view the check breakdown, and manage the history log.

Finally, the system must ensure **Reliability (NFR-004)**, aiming for **≥99.5% uptime** and managing external constraints. This includes implementing the constraint that the tool must enforce a request throttle on outgoing WHOIS requests to respect external server rate limits **(NFR-005 - Constraint Handling)**. By meeting these non-functional requirements, the platform will deliver not only correctness in detection but also trustworthiness, speed, and long-term operational viability.

3.3 Hardware Requirements

The proposed **VPN and Proxy Detector** platform is a high-performance, web-based system that necessitates a reliable and resource-optimized hardware environment, particularly on the server side, to meet its strict NFRs. The server must be fundamentally capable of handling multiple concurrent API requests (FR-001), performing the computationally intensive, asynchronous detection logic (FR-

003), managing high-speed data I/O for the **Redis cache (NFR-001)**, and logging all transactions to the database (FR-007). Therefore, a system with a powerful multi-core processor, substantial in-memory capacity, and fast storage is required to ensure smooth, low-latency execution of all critical services.

On the server side, a machine or VM with a **multi-core CPU (e.g., Intel i7 or AMD Ryzen 7 equivalent)** is recommended to handle the concurrent demands of the Node.js event loop. A minimum of **16 GB RAM is required, but 32 GB or more is highly recommended**, specifically to ensure the **Redis V7+ cache** can hold large portions of the threat intelligence and WHOIS data in memory. This memory capacity is directly responsible for achieving the crucial **$<$150 ms warm lookup latency (NFR-001)**. Storage must be provided by a **Solid-State Drive (SSD)** to facilitate the rapid loading of the CIDR lists and fast MongoDB write operations for audit logging. For high-volume institutional deployment, the use of cloud-based servers with horizontal scaling capabilities is preferred to ensure the system handles the **≥1,000 lookups per minute (NFR-003)** standard.

On the client side, analysts and administrators can access the platform through standard desktops or modern tablets. A system with a minimum of 4 GB RAM and a modern web browser is sufficient to run the **React.js/MUI interface**. Since the system is web-based, all heavy processing is offloaded to the server, requiring only a stable, secure internet connection on the client side. These detailed hardware specifications collectively ensure that the platform maintains its efficiency, supports concurrent users securely, and delivers a seamlessly auditable and high-speed detection experience.

3.4 Software Requirements

The proposed **VPN and Proxy Detector** platform is architected as a full-stack, web-based system, relying on a set of robust, version-specific software tools and frameworks to guarantee functionality and scalability. Both the server and client environments require carefully selected components to support development, containerized deployment, and final usage.

On the server side, the backend is developed using **Node.js v18+ with Express 5** to host the core IP analysis logic, handle **JWT authentication (NFR-002)**, and manage **Socket.io** integration for real-time bulk job status updates (FR-006). The primary logic, including CIDR checks and port probing, is encapsulated within dedicated TypeScript services. The system database is managed using **MongoDB V8.0** for persistent, auditable storage of all lookup history and job metadata (FR-007). A critical component is the **Redis V7+** service, which acts as the high-speed cache, directly ensuring the low latency required by the system (NFR-001). For deployment, the system must be containerized using **Docker Compose**, ensuring portability, service isolation, and simplified management on Linux (Ubuntu preferred) or Windows servers.

On the client side, the platform is designed using **React.js v18+** and **Material-UI (MUI)** to create a responsive, user-friendly interface optimized for both desktop and tablet views (NFR-005). The frontend communicates with the secure backend services exclusively through encrypted **REST APIs (HTTPS/TLS)**, providing smooth navigation and dynamic updates for single lookups and bulk job results. A standard modern web browser (Google Chrome or Firefox) is required for accessing the system.

In addition, software tools like **Git/GitHub** are mandatory for rigorous version control and collaborative development. Quality assurance mandates the use of **Jest and Supertest** for automated unit and integration testing, verifying core logic performance against the NFR targets. By combining these robust software components, the platform guarantees modularity, security, and ease of deployment, making it suitable for rigorous institutional use.

3.5 Agile Model

The development of this project strictly follows the **Agile Software Development Model**, specifically the **Scrum framework**, which is exceptionally suited for high-risk projects dependent on external data and stringent performance goals. Unlike traditional sequential models, Scrum allows the system to be developed in small, highly focused iterations known as **sprints** (typically 2-4 weeks). Each sprint delivers a functional, testable increment of the system, such as the initial **WHOIS Caching Layer** or the **Bulk Job Queue Handler** (FR-006).

At the initiation of the project, all major requirements—from the ≤150 ms latency (NFR-001) to the ≥1,000 lookups/minute rate (NFR-003)—were prioritized and divided into manageable stories. Each sprint is planned to produce a functional and testable output that directly addresses a key requirement.

Agile fundamentally emphasizes continuous testing and feedback. After each sprint, the developed module is rigorously tested for performance, security (JWT/TLS), and accuracy. Performance testing is critical in the early stages to ensure the detection and caching logic can meet the speed targets. Feedback from test users and the project mentor is incorporated into the next sprint cycle, ensuring the platform evolves according to operational needs and mitigating risks related to technical feasibility. By adopting Scrum, the project ensures rapid delivery of critical features, adaptability to changes in external threat data, and greater collaboration, making the platform reliable and aligned with high-security industry requirements.

Chapter-4

System Architecture

4. System Architecture

The proposed **VPN and Proxy Detector** system is designed using a modular, highly secure, and three-tiered architecture to ensure low latency, scalability, and robust security (NFR-002). Since the system will be accessed by authorized analysts and administrators via secure web browsers and integrated scripts, the architecture strictly adheres to a **Client-Server-Data model**. The **Client Layer** initiates requests, the **Server Layer** implements the complex detection logic via secure APIs, and the **Data Layer** handles both high-speed caching and persistent auditing.

At the **Client Layer** (Presentation Layer), the frontend is built using **React.js v18+ with Material-UI (MUI)** to provide a responsive and intuitive interface. This layer handles user interactions, including single IP/domain submission, **Bulk CSV Upload (FR-006)**, and viewing the **History/Audit Log (FR-007)**.

The **Server Layer** (Application Layer) is developed using **Node.js 18 + Express 5**. It is responsible for implementing the entire business logic, managing **JWT/API-Key authentication (NFR-002)**, orchestrating the multi-tiered detection workflow, and controlling the job queue for bulk processing (FR-006). This layer acts as the secure API gateway.

The core detection service is a specialized module, logically separated within the server layer, where the sequential **Passive CIDR checks** and **Active Port Probes (FR-003)** occur. This module also interfaces with the **WHOIS/RDAP client**.

At the **Data Layer**, **MongoDB V8.0** is used as the primary database for storing all auditable records, user details, and bulk job metadata (FR-007). **Redis V7+** is implemented as the high-speed caching mechanism, essential for guaranteeing the **$<$150 ms warm lookup latency (NFR-001)**. The system architecture incorporates security modules, with all API traffic secured by **HTTPS/TLS** and continuous logging maintained for auditing purposes. Future features like JA3 Fingerprinting can be added as separate microservices without disturbing the existing core, thanks to its modular design. Thus, the system architecture ensures seamless communication and allows the platform to deliver auditable, high-speed results to investigators.

4.1 Client-Server Architecture

The platform operates on a rigid **Client-Server Architecture**, which fundamentally separates the user interface from the core processing and security logic to ensure high efficiency, modularity, and scalability (NFR-004). In this model, the **Client Layer** represents the **React.js/MUI** web application accessed securely by analysts, while the **Server Layer** manages all demanding backend processes, database operations, caching, and the multi-tiered detection system.

On the client side, the **React.js frontend** provides an intuitive interface where users can log in securely, utilize the single lookup form, upload **CSV files for bulk processing (FR-006)**, and view results visualized with status indicators and WHOIS details. The client performs minimal processing; instead, it communicates with the server by sending authenticated requests and receiving auditable JSON responses through secure **REST APIs**.

On the server side, the **Node.js/Express backend** handles all core business logic, including user authentication (JWT), managing the integrity of the threat intelligence feeds, and updating the **Redis cache** (NFR-001). It acts as the secure intermediary between the client and the dedicated analytical services. When an analyst submits data, the server orchestrates the sequence: it forwards the input to the **Detection Engine**, collects the verdict and WHOIS data, updates the **MongoDB Audit Log (FR-007)**, and then securely transmits the outcome back to the client.

The core **Detection Engine**, while logically part of the server, is managed as a self-contained service, ensuring the security and isolation of the crucial **Active Port Probing (FR-003)** process. This separation prevents any security issues from affecting the core application and guarantees fairness by evaluating every lookup under consistent conditions. The **Data Layer (MongoDB/Redis)** is also managed on the server side, ensuring quick, reliable, and auditable data retrieval, directly supporting the ambitious performance and compliance requirements. This client–server separation ensures the system remains responsive while securely distributing demanding processing tasks.

5. Design and Implementation

The design and implementation of the proposed **VPN and Proxy Detector** system have been meticulously carried out with a stringent focus on **modularity, high performance (NFR-001), and forensic audibility (FR-007)**. The system is logically partitioned into distinct modules—Authentication, Detection Engine, WHOIS Client, Bulk Processor, and Audit Log—each responsible for a specific, non-overlapping functionality. This modular structure, underpinned by the **Node.js/Express** microservice approach, ensures that the platform can be easily maintained, performance-tuned (e.g., the Redis cache), and extended with future enhancements like JA3 Fingerprinting.

In the **design phase**, essential system models such as the **Data Flow Diagram (DFD), Entity–Relationship Diagram (E-R Diagram), Use Case Diagram, Class Diagram, and Sequence Diagram** were prepared. These comprehensive diagrams serve as the critical blueprint, enabling both developers and stakeholders to fully visualize the precise data flow, structural components, and high-speed behavior of the multi-tiered detection process prior to coding.

The implementation phase strictly follows the **Agile methodology**, with each module developed and tested within dedicated iterative sprints. For instance, initial sprints focused on implementing the **Redis caching layer** and the **Passive CIDR check logic**, followed by the complex **Active Port Probing (FR-003)** and the **MongoDB Audit Log (FR-007)**. After each sprint, rigorous **Jest and Supertest** validation is performed to ensure the features meet critical performance metrics, especially the **$<$150 ms warm latency (NFR-001)**. Continuous feedback ensures the detection logic is iteratively refined.

Technologically, the frontend utilizes **React.js v18+ with MUI** to deliver a responsive interface for single and bulk lookups. The backend is implemented using **Node.js 18 + Express 5**, which handles all secure business and orchestration logic. **Redis V7+** acts as the high-speed cache, while **MongoDB V8.0** serves as the primary database for storing all auditable lookup records. The **Bulk Processor** module is designed to execute asynchronously, ensuring the system supports horizontal scaling to handle the ≥1,000 lookups per minute (NFR-003).

The implementation places paramount emphasis on **security and performance**. Authentication uses **JWT/TLS encryption (NFR-002)**, and all transactions are immutably logged for legal monitoring (FR-007). The architecture's inherent modularity ensures that the system is not only reliable and high-speed but also highly adaptable to future threat intelligence enhancements.

5.1 Product Features

The **VPN and Proxy Detector** platform has been designed with a comprehensive set of operational features that make it essential for both security analysts and system administrators. At its core, the system provides **secure authentication and user management (NFR-002)**. Analysts can securely log in with role-based access to perform lookups, while administrators are granted necessary privileges to manage threat feeds, monitor performance (NFR-001), and manage the system's compliance logs. This setup ensures analysts can focus on investigations while administrative oversight is maintained.

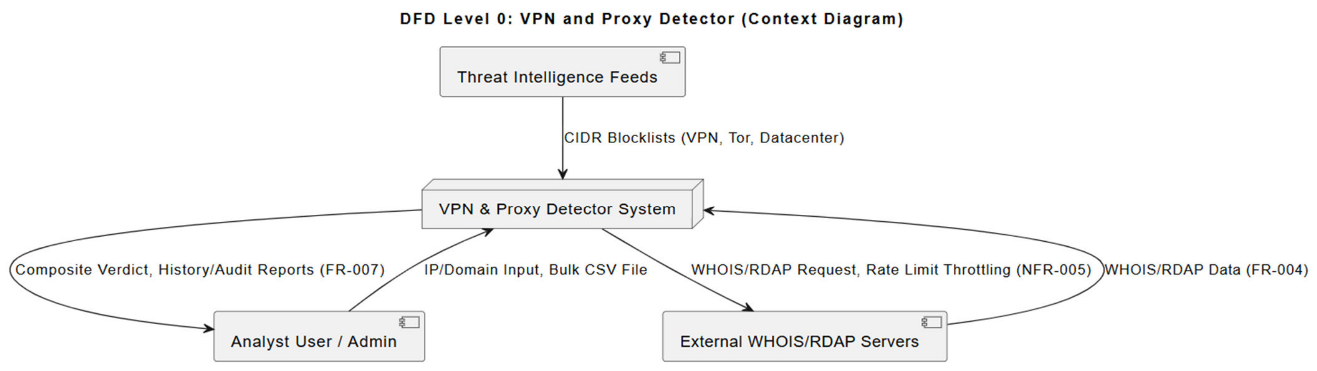
The **Single Lookup and Bulk Processing module (FR-001, FR-006)** is one of the most critical features of the platform. Analysts can input a single IP/domain or upload a **CSV file for Bulk Processing**. The system automatically processes the submissions, initiating the **multi-tiered detection engine (FR-002, FR-003)** within a low-latency environment. For bulk jobs, the system provides **real-time status polling via Socket.io** and guarantees a high-speed execution rate of **≥1,000 lookups per minute (NFR-003)**. Instant feedback is provided to the user in the form of a clear verdict, a composite score, and a breakdown of the checks performed.

The **WHOIS/RDAP Enrichment module (FR-004)** seamlessly integrates into the detection workflow, providing essential network context for every validated IP. This module retrieves registrar details, creation/expiry dates, and confirms if the IP belongs to a Data Center Host, caching the results in **Redis** to uphold the stringent **$<$150 ms warm latency (NFR-001)**. This enrichment is critical for separating residential IPs from commercial anonymizers.

The platform also provides a comprehensive **Audit Log and Reporting module (FR-007)**. This module records rankings and stores **every lookup transaction immutably** in **MongoDB**. It allows administrators to filter history by date, user, and verdict, and **export the data** for official **chain-of-custody** reporting. Finally, the system provides a **Performance Dashboard** that helps administrators monitor real-time latency (NFR-001) and bulk queue status. Together, these features make the platform a complete, auditable ecosystem for high-security IP provenance verification.

5.2 Data flow diagram

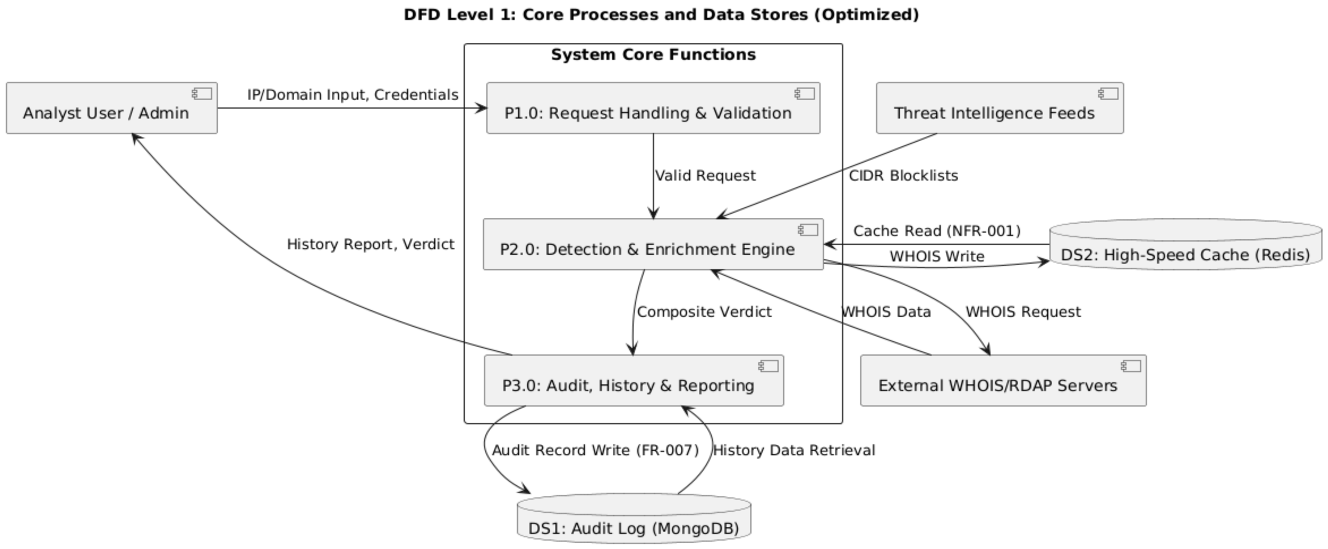
WriteA Data Flow Diagram (DFD) provides a graphical representation of how data moves through the system, showing the processes, external entities, and data stores involved. It helps in understanding the overall flow of information within the platform and ensures that all interactions between users and the system are clearly defined.


DFD Level 0: VPN and Proxy Detector (Context Diagram)
Threat Intelligence Feeds
CIDR Blocklists (VPN, Tor, Datacenter)
VPN & Proxy Detector System
Composite Verdict, History/Audit Reports (FR-007)
IP/Domain Input, Bulk CSV File
WHOIS/RDAP Request, Rate Limit Throttling (NFR-005)
WHOIS/RDAP Data (FR-004)
Analyst User / Admin
External WHOIS/RDAP Servers

Chapter-5

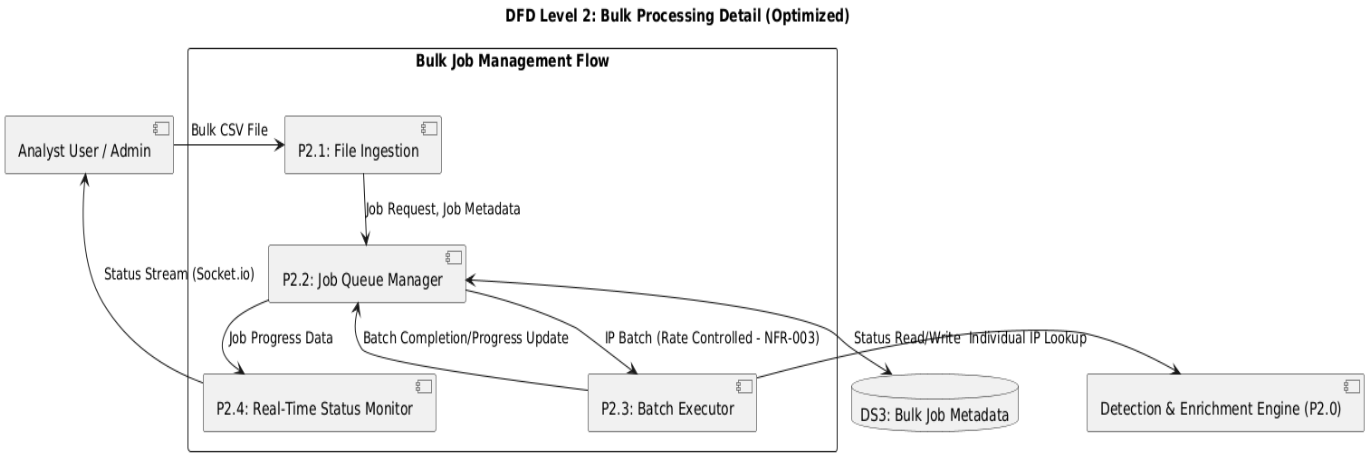
DFD-0

The Level 0 Context Diagram establishes the **system boundary**, showing the **VPN and Proxy Detector** as a single process. Its primary purpose is to define the crucial interactions with external entities: the **Analyst User**, **External WHOIS/RDAP Servers**, and **Threat Intelligence Feeds**. It clearly illustrates the high-level flow of **IP input, data enrichment, and auditable verdict output (FR-007)**..


DFD Level 1: Core Processes and Data Stores (Optimized)
System Core Functions
Analyst User / Admin
IP/Domain Input, Credentials
P1.0: Request Handling & Validation
Threat Intelligence Feeds
Valid Request
CIDR Blocklists
History Report, Verdict
P2.0: Detection & Enrichment Engine
Cache Read (NFR-001)
WHOIS Write
DS2: High-Speed Cache (Redis)
Composite Verdict
WHOIS Data
WHOIS Request
P3.0: Audit, History & Reporting
External WHOIS/RDAP Servers
Audit Record Write (FR-007)
History Data Retrieval
DS1: Audit Log (MongoDB)

DFD-1

This diagram decomposes the system into its major logical processes: **Request Handling (P1.0)**, the **Detection & Enrichment Engine (P2.0)**, and **Audit, History & Reporting (P3.0)**. It visualizes the flow of data between these processes and the essential **Data Stores (MongoDB Audit Log and Redis Cache)**. This view is vital for understanding the sequential nature of the classification and logging commitment (FR-007).

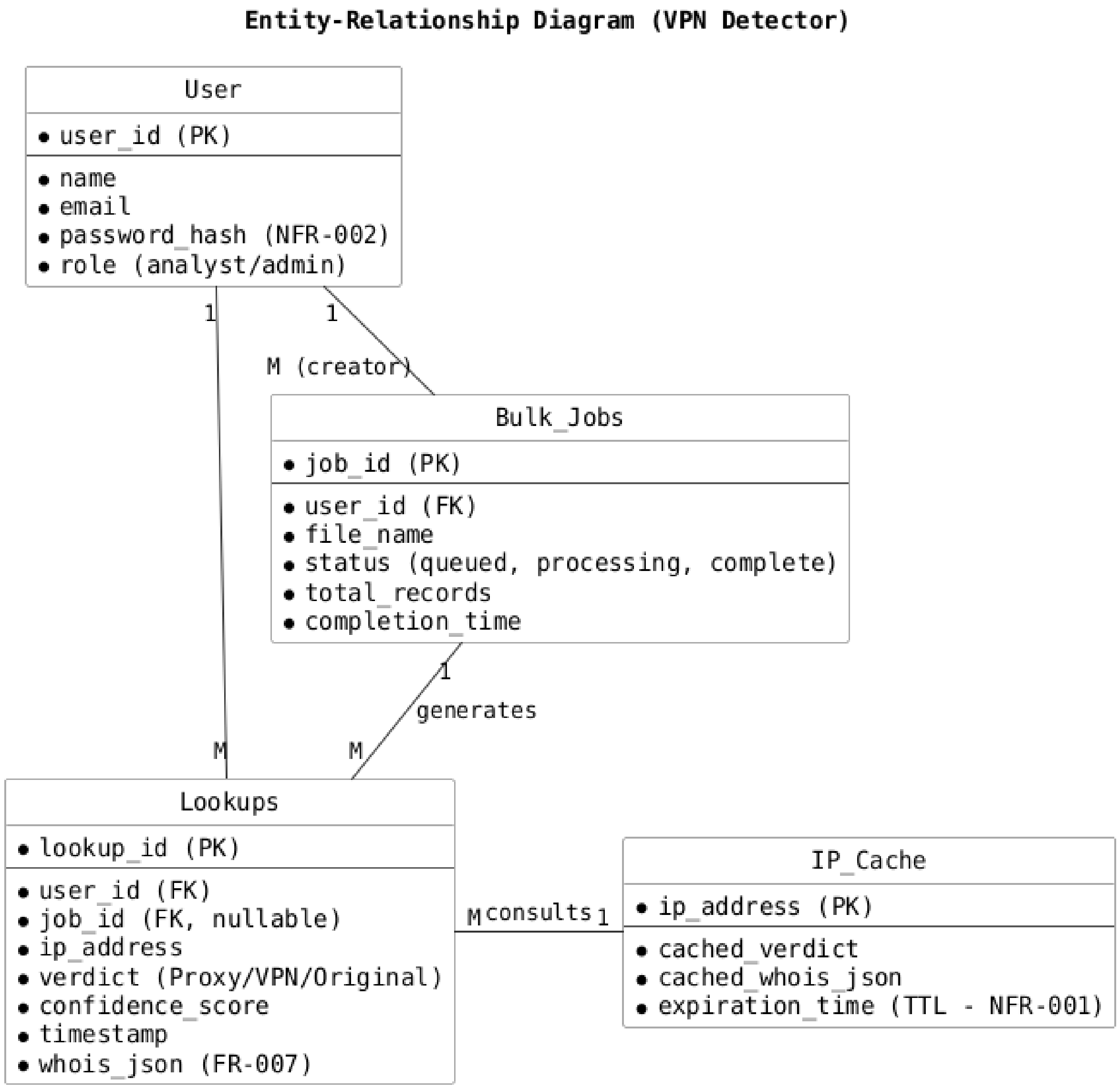

DFD Level 2: Bulk Processing Detail (Optimized)
Bulk Job Management Flow
Analyst User / Admin
Bulk CSV File
P2.1: File Ingestion
Job Request, Job Metadata
Status Stream (Socket.io)
P2.2: Job Queue Manager
Job Progress Data
Batch Completion/Progress Update
IP Batch (Rate Controlled - NFR-003)
Status Read/Write
Individual IP Lookup
P2.4: Real-Time Status Monitor
P2.3: Batch Executor
DS3: Bulk Job Metadata
Detection & Enrichment Engine (P2.0)

DFD-2

The Level 2 DFD provides granular detail for the highly critical, asynchronous **Bulk Processing (FR-006)** workflow. It illustrates how the system ingests the CSV, manages the **Job Queue Manager (P2.2)**, and controls the rate of execution for the **Batch Executor (P2.3)** to meet the stringent **≥1,000 lookups/minute (NFR-003)** requirement. It also highlights the **Socket.io** channel used for real-time status monitoring (P2.4).

The complete set of DFDs (DFD-0, DFD-1, and DFD-2) provides a clear picture of how data is processed and managed in the platform. They not only illustrate system behavior but also help in identifying possible improvements and ensuring that all requirements are met.

5.3 E-R diagram



Chapter-5

E-R Diagram

The **Entity–Relationship (E–R) Diagram** of the detection platform represents the logical structure of the database and explicitly shows how different entities in the system are interconnected. This model is a crucial part of the design process because it ensures that all necessary data for auditing (FR-007), job management (FR-006), and user security is accurately captured and efficiently organized. The design prioritizes compliance by making the audit log (Lookups) the central entity.

The main entities of the system include **Users**, **Lookups (Audit History)**, **Bulk_Jobs**, and **IP_Cache**. Each entity is defined by specific attributes that determine its role within the system. The **Users** entity stores details such as user_id (PK), name, email, and password_hash for secure access (NFR-002). The **Lookups** entity contains the definitive, auditable record: lookup_id (PK), user_id (FK), ip_address, timestamp, the classification verdict, and the full **whois_json** (FR-007).

The **Bulk_Jobs** entity tracks the lifecycle of asynchronous processing tasks created by CSV uploads (FR-006). It includes job_id (PK), user_id (FK), file_name, and the job status (queued, processing, complete). The **IP_Cache** entity, managed primarily by Redis, tracks IPs and their cached_verdict and expiration_time to support the **$<$150 ms warm latency (NFR-001)**.

The relationships between these entities are as follows:

- A **User** can perform multiple **Lookups** (1-to-Many).
- A **User** can create multiple **Bulk_Jobs** (1-to-Many).
- **Lookups** are derived from **Bulk_Jobs** (1-to-Many, one job generates many lookup records).
- The **IP_Cache** is consulted by every **Lookup** for efficiency.

This E–R diagram provides the structural backbone of the MongoDB database, ensuring that data is stored efficiently, relationships are maintained for traceability, and all functional and compliance requirements (especially FR-007) are supported.

5.4 Class diagram

Class Diagram

The **Class Diagram** provides a detailed, static view of the object-oriented structure of the **VPN and Proxy Detector** system, essential for mapping real-world requirements into implementable software components. This diagram represents the distinct classes in the Node.js/TypeScript backend, their defining attributes (data), critical methods (behavior), and the relationships that govern their interactions. This structural blueprint is crucial during the implementation phase as it directly informs the system's modularity and maintainability.

The main classes of the platform are **ExpressServer**, **AuthMiddleware**, **DetectorService**, **WhoisClient**, **BulkProcessor**, and **AuditLogManager**.

- The **ExpressServer** class acts as the API gateway and orchestrator. Its methods include run(), postDetectEndpoint(), and postBulkEndpoint().

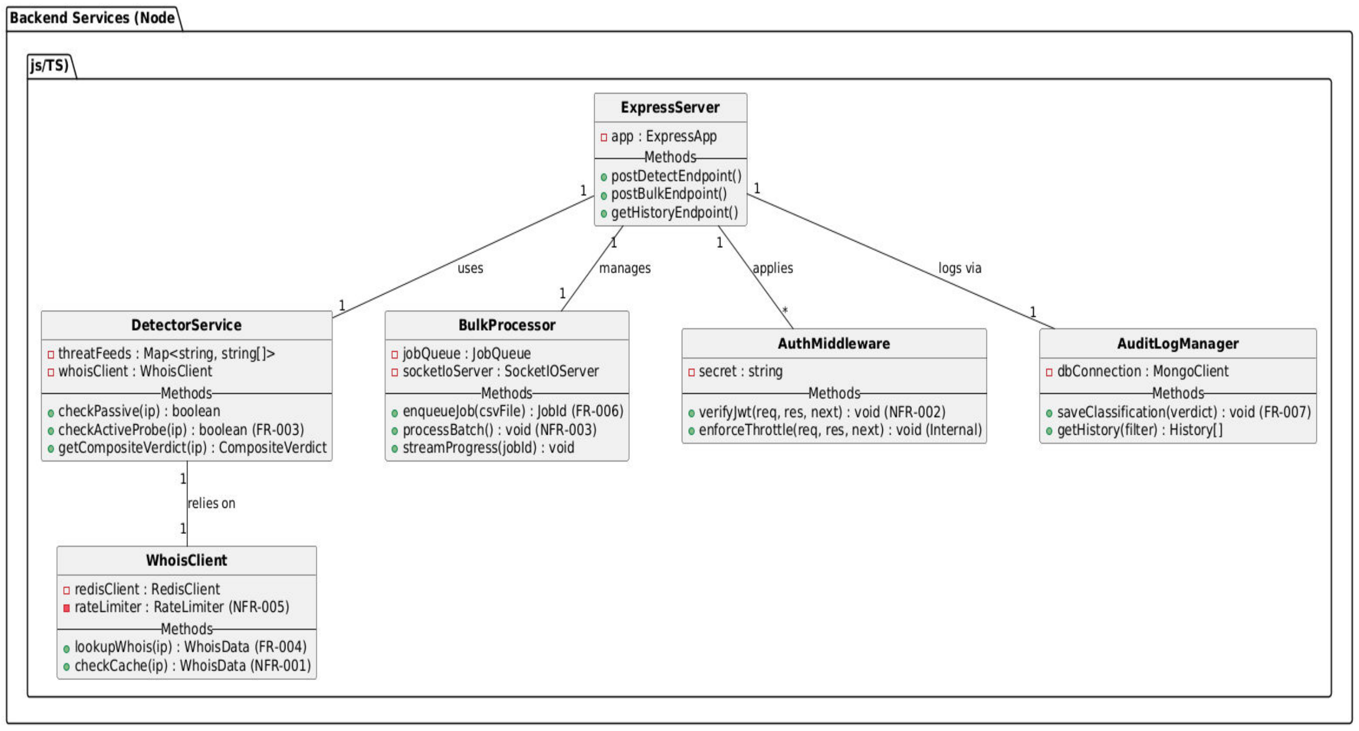
- The **DetectorService** class contains attributes for threat feeds and methods like checkPassive() (FR-002) and checkActiveProbe() (FR-003).
- The **WhoisClient** class manages external data fetching and performance. It includes the redisClient attribute and methods like lookupWhois() and checkCache() (NFR-001).
- The **BulkProcessor** class manages asynchronous job execution (FR-006). Its methods include enqueueJob() and streamProgress() (using Socket.io).
- The **AuditLogManager** class is crucial for compliance (FR-007), with methods like saveClassification() and getHistory().
- The **AuthMiddleware** class handles security checks, with the verifyJwt() and enforceThrottle() methods (NFR-002, NFR-005).

The key relationships among classes are defined by the workflow:

- **ExpressServer** uses **AuthMiddleware** and relies on **DetectorService** and **BulkProcessor** to execute core logic.
- **DetectorService** strictly depends on **WhoisClient** for enrichment and caching.
- All core processes rely on the **AuditLogManager** to record transactions (FR-007).

This Class Diagram forms the structural backbone of the object-oriented design, ensuring the system is modular, high-performing, and easy to maintain while meeting all security and audit requirements.

Class Diagram: VPN and Proxy Detector

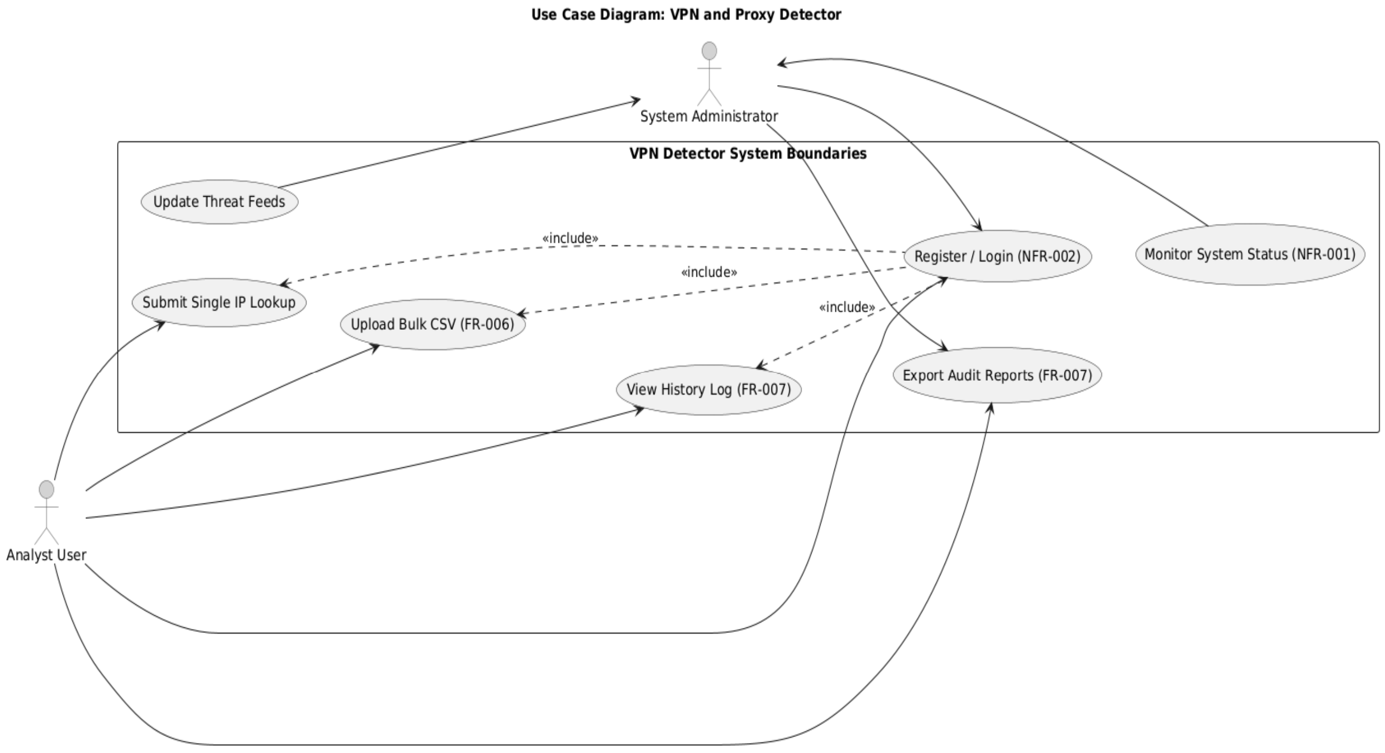

Backend Services (Node
js/TS)
ExpressServer
app : ExpressApp
Methods
postDetectEndpoint()
postBulkEndpoint()
getHistoryEndpoint()
uses
manages
applies
logs via
1
1
1
1
1
1
*
1
DetectorService
threatFeeds : Map<string, string[]>
whoisClient : WhoisClient
Methods
checkPassive(ip) : boolean
checkActiveProbe(ip) : boolean (FR-003)
getCompositeVerdict(ip) : CompositeVerdict
BulkProcessor
jobQueue : JobQueue
socketIoServer : SocketIOServer
Methods
enqueueJob(csvFile) : JobId (FR-006)
processBatch() : void (NFR-003)
streamProgress(jobId) : void
AuthMiddleware
secret : string
Methods
verifyJwt(req, res, next) : void (NFR-002)
enforceThrottle(req, res, next) : void (Internal)
AuditLogManager
dbConnection : MongoClient
Methods
saveClassification(verdict) : void (FR-007)
getHistory(filter) : History[]
1
relies on
1
WhoisClient
redisClient : RedisClient
rateLimiter : RateLimiter (NFR-005)
Methods
lookupWhois(ip) : WhoisData (FR-004)
checkCache(ip) : WhoisData (NFR-001)

Chapter-5

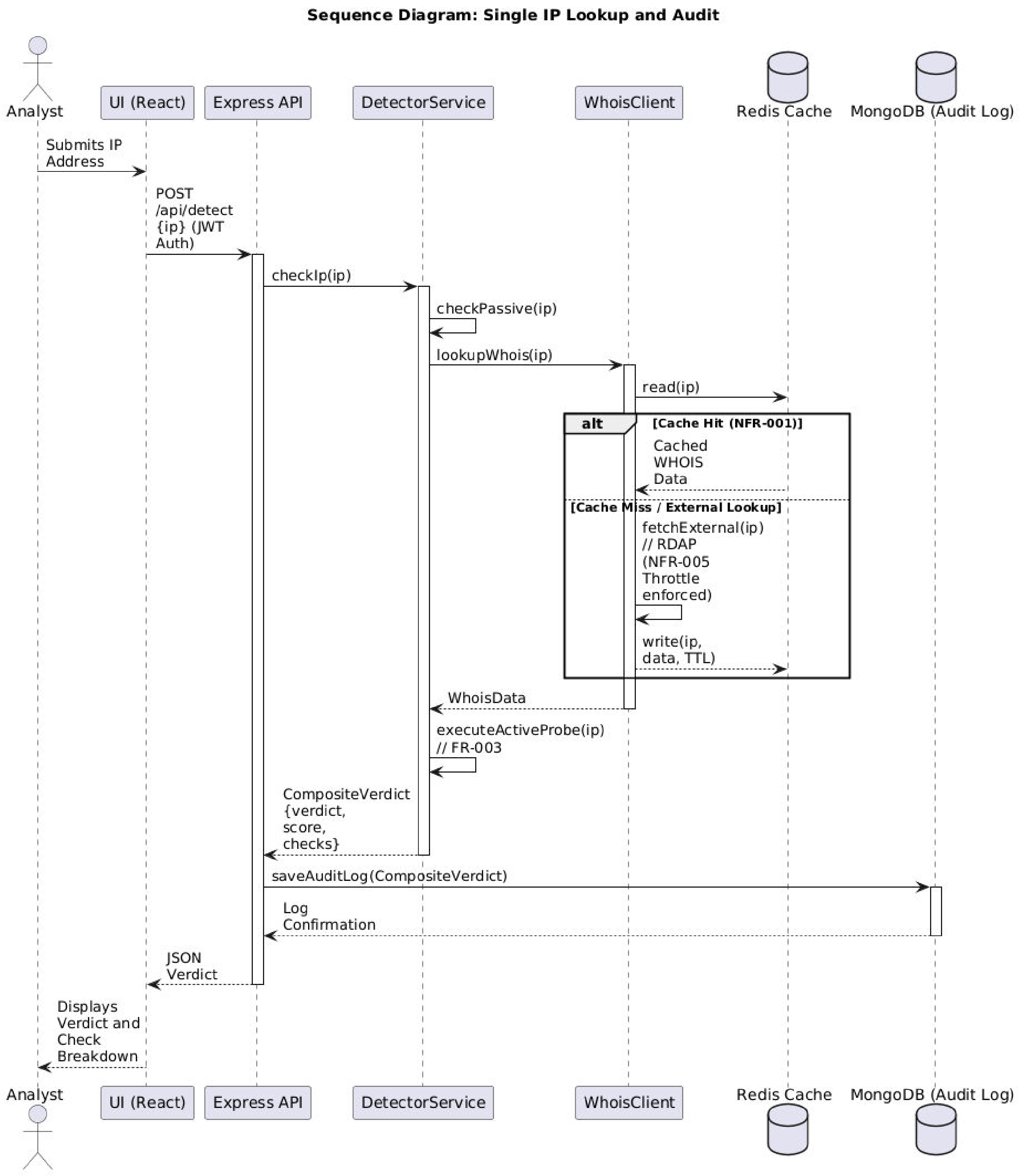
5.5 Use Case diagram

The Use Case Diagram illustrates the interactions between the primary actors of the system and the major services provided by the platform, clearly defining the system's behavioral scope. The main actors in the VPN and Proxy Detector system are the Analyst User and the System Administrator/Developer. Analyst Users utilize the system for core investigative tasks: they must Register/Login (NFR-002), Submit Single IP/Domain Lookup (FR-001), Upload Bulk CSV (FR-006), and View Lookups History (FR-007). They also receive the Composite Verdict and Bulk Job Status (Socket.io).

System Administrators oversee maintenance and compliance: they can Register/Login, Generate Audit Reports (FR-007), and Monitor System Status (checking latency NFR-001 and queue status NFR-003). The Administrator is also responsible for updating the external Threat Intelligence Feeds (CIDR lists). The diagram visually groups these actions to demonstrate how each role engages with the system's specific functionality, providing a quick reference for understanding security and operational boundaries. All actors share the prerequisite use case of Register/Login to ensure secure, authenticated access (NFR-002).


Use Case Diagram: VPN and Proxy Detector
System Administrator
VPN Detector System Boundaries
Update Threat Feeds
«include»
Register / Login (NFR-002)
Monitor System Status (NFR-001)
«include»
Submit Single IP Lookup
Upload Bulk CSV (FR-006)
«include»
Export Audit Reports (FR-007)
View History Log (FR-007)
Analyst User

5.6 Sequence diagram


Sequence Diagram: Single IP Lookup and Audit
Analyst
UI (React)
Express API
DetectorService
WhoisClient
Redis Cache
MongoDB (Audit Log)
Submits IP Address
POST /api/detect {ip} (JWT Auth)
checkIp(ip)
checkPassive(ip)
lookupWhois(ip)
read(ip)
alt
[Cache Hit (NFR-001)]
Cached WHOIS Data
[Cache Miss / External Lookup]
fetchExternal(ip) // RDAP (NFR-005 Throttle enforced)
write(ip, data, TTL)
WhoisData
executeActiveProbe(ip) // FR-003
CompositeVerdict {verdict, score, checks}
saveAuditLog(CompositeVerdict)
Log Confirmation
JSON Verdict
Displays Verdict and Check Breakdown

Chapter-5

Sequence Diagram

The **Sequence Diagram** illustrates the runtime interaction and message flow between the primary system components involved in a typical **Single IP Lookup with Auditing** cycle. The flow begins when an Analyst User submits an IP address via the **React-based User Interface**. The **Frontend** sends a secure request to the **Express API endpoint (POST /api/detect)**, including the IP address and **JWT authentication token (NFR-002)**. Upon receiving the request, the **Backend API** enforces security checks and immediately passes the IP to the **DetectorService**.

The **DetectorService** initiates the multi-tiered classification: it performs the **Passive CIDR check (FR-002)** and then requests **WHOIS/RDAP enrichment** from the **WhoisClient**. The **WhoisClient** first queries the **Redis Cache (NFR-001)**; if a miss occurs, it performs an external lookup, managing the necessary throttling (NFR-005). Once the enrichment data returns, the service executes the **Active Port Probe (FR-003)** and compiles the final **Composite Verdict**.

Critically, before sending any result back to the user, the **Backend API** sends the complete verdict and WHOIS data to the **MongoDB Audit Log Manager**. This step records the transaction immutably (FR-007) and receives a log confirmation. Finally, the **API** returns the JSON verdict to the **Frontend**, which displays the classification and check breakdown to the Analyst User. This sequence ensures secure execution, low latency (NFR-001), and a fully auditable record of every detection request (FR-007).

6. Conclusion & Future Scope

6.1 Conclusion

The development of the **VPN and Proxy Detector** project has been a highly complex and crucial initiative, successfully addressing critical security and compliance challenges faced by investigative institutions. Throughout the design and implementation stages, we have rigorously adhered to the principle that digital security requires not just speed, but verifiable data integrity. High-level anonymity tactics, such as those employing VPNs and proxies, often limit the growth of investigations by obscuring verifiable IP provenance. Our project directly resolves this issue by providing an all-in-one, **self-hosted platform** where analysts can obtain instant, multi-tiered IP verification in a systematic, auditable manner.

From a technical perspective, the project successfully demonstrates the power of a finely tuned **Node.js/React stack** integrated with specialized components. By integrating frontend services with the **Express backend**, asynchronous Bulk Processing services, and a performance-critical **Redis caching layer (NFR-001)**, the system showcases how modern technologies can be combined to create scalable and **high-security platforms**. The modular architecture ensures that each component—whether it is **JWT authentication (NFR-002)**, the **Detection Engine (FR-003)**, or the **MongoDB Audit Log (FR-007)**—works independently yet integrates smoothly to uphold the collective performance commitment. This modularity makes the platform not only reliable but also easy to maintain and extend with future detection logic.

The platform also successfully bridges the gap between investigative needs and institutional compliance. In today's regulatory environment, all forensic data must be auditable and secured within the organizational perimeter. By implementing a self-hosted architecture, our system directly counters the security risks associated with third-party APIs. By simulating an industry-standard, high-speed environment capable of **≥1,000 lookups per minute (NFR-003)**, our system prepares security teams to face high-volume, real-time intelligence gathering tasks with confidence. Moreover, the mandatory **Audit Log (FR-007)** gives institutions a powerful tool to verify data, essential for legal reporting. This dual benefit—speed for analysts and compliance for institutions—makes the system highly valuable for high-security deployment.

Another important outcome of this project is the **integrity and transparency** it guarantees. By providing a college-managed, fully auditable platform, we ensure that every lookup, regardless of its source, is recorded and verifiable. In this way, the platform supports the idea of strengthening **digital justice (SDG 16)** by providing equal opportunities for thorough investigation to all authorized analysts.

Chapter-6

6.2 Future Scope and Enhancements

While the current version of the project already provides core features—secure authentication, multi-tiered detection, asynchronous bulk processing, and auditable reporting—the future scope of the system is vast and focuses on resisting technological obsolescence. Some possible extensions include:

- **JA3 TLS Fingerprinting:** The system will integrate techniques to analyze the TLS handshake characteristics. This allows the platform to identify specific VPN client software signatures, providing an additional layer of detection that is independent of the IP address itself.
- **Machine Learning / Flow Analytics:** The system will integrate artificial intelligence to analyze network traffic and behavioral patterns. For example, if an IP connection exhibits traffic flow characteristics inconsistent with residential use, the system can recommend a higher suspicion score, improving accuracy beyond static lists.
- **Proactive Evasion Detection:** One major challenge is obfuscation. Future versions will include advanced algorithms that compare port probe responses against known evasion patterns to identify stealth proxy techniques.
- **Integration with SIEM/SOAR Systems:** The platform can be extended to connect with existing enterprise Security Information and Event Management (SIEM) platforms or Security Orchestration, Automation, and Response (SOAR) systems. This integration would allow real-time classification to trigger automated response actions.
- **Advanced Geolocation Dashboards:** With the rise of data science, the platform can be equipped with powerful dashboards that provide deep, visual insights into IP geolocation trends. Administrators can use these to identify sources of abuse and adapt network filtering policies accordingly.
- **Support for Large-Scale Threat Feeds:** Currently designed for college-level deployment, the system can be scaled to support inter-agency or corporate use, handling integration with massive commercial threat intelligence feeds. This would increase the platform's data versatility.
- **Mobile Command Support:** To improve accessibility for field analysts, the platform can be extended to mobile devices with a dedicated app interface. This will help analysts perform lookups anytime and anywhere.

6.3 Final Thoughts

In conclusion, the **VPN and Proxy Detector** project provides a strong foundation for an institutional tool that not only benefits security analysts but also aligns with the global vision of **digital justice and strong institutions (SDG 16)**. The project demonstrates that with the right use of technology, critical security functions can be made auditable, high-performance, and secure. The successful implementation of this system proves that even complex real-world problems, such as secure IP provenance verification, can be solved effectively through teamwork and modern development practices. The future of this project is full of possibilities. With continuous improvements, community feedback, and institutional support, the platform has the potential to evolve into a complete ecosystem for digital threat intelligence.

Chapter-7

UN sustainable development goals

7. Mapping with UN sustainable development goals

7.1 Mapping with UN Sustainable Development Goals

The United Nations has identified 17 Sustainable Development Goals (SDGs) as part of the 2030 Agenda for Sustainable Development. Our project, the **VPN and Proxy Detector**, aligns particularly with goals related to **peace, justice, strong institutions, and technological innovation**. The system's social mission directly supports the reduction of digital violence and the establishment of auditable justice mechanisms. The following table highlights the correlation of our project objectives with the most relevant SDGs:

SDG Title :			
S.No	**Product Objective**	**Proposed Solution**	**Mapping 3/2/1**
1	To significantly reduce digital crime by mitigating anonymity.	The system provides **true IP provenance** for law enforcement, directly aiding the tracking and prosecution of cybercriminals who hide their identities.	**3 (Substantial) – Supports Target 16.1 (Reduce violence).**
2	To ensure accountability and equal access to digital justice for investigations.	The mandatory **Audit Log (FR-007)** ensures all lookups are consistent, transparent, and legally verifiable, contributing to accountable digital practices.	**3 (Substantial) – Supports Target 16.3 & 16.6 (Promote rule of law, effective institutions).**
3	To provide a robust, resilient, and self-hosted digital infrastructure tool.	The use of advanced technologies (**Node.js/React, Redis Cache, Multi-tiered Detection**) fosters innovation and a scalable infrastructure for institutional security.	**2 (Moderate) – Supports Target 9.A (Facilitate sustainable infrastructure development).**
4	To enhance security against financial fraud and cyber-exploitation of vulnerable users.	By enabling effective investigation and countering financial cybercrime, the platform helps protect against digital exploitation, contributing to reduction of inequality in safety.	**2 (Moderate) – Supports T inclusion). Export to Sheets**

Chapter-7

The matrix given represents the correlation of Product objectives with the SDGs. This correlation is defined based on the following levels:

- **1: Slight (Low)**
- **2: Moderate (Medium)**
- **3: Substantial (High)**