



Software Design Specifications Document

VPN Spyglass: VPN Traffic Analyzer

By

Hassan Abdullah 337275

Sameen Mubashar 346848

Supervisor:

Dr. Mehdi Hussain

Co-Supervisor:

Dr. Arsalan Ahmad

Bachelor of Science in Computer Science (2020-2024)

Department of Computing

School of Electrical Engineering and Computer Science

National University of Sciences & Technology

Table of Contents

List of Tables	v
List of All Diagrams	v
List of Activity Diagrams	v
List of Data Flow Diagrams	v
List of User Interface Diagrams	v
1. Introduction.....	1
1.1 System Purpose	1
1.2 System Scope.....	1
1.3 Definitions, Acronyms, and Abbreviations	1
2. Design Methodology and Software Process Model	2
2.1 Design Methodology	2
2.1.1 Website based dashboard.	3
2.1.2 Integration of Switch/Firewall	3
2.1.3 User Authentication	3
2.2 Software Process Model	4
3. System Overview	4
3.1 System Operations and Requirements coverage.....	4
3.2 Architectural Design.....	4
3.2.1 Inter-Module Dependencies	5
3.2.1.1 User Interfaces	5
3.2.1.2 Controllers.....	5
3.2.1.3 Real-time Traffic Module	5
3.2.1.4 VPN Categorization Module.....	5
3.2.1.5 VPN Blocking Module	5
3.2.1.6 Backend Services	5
3.2.1.7 Database Management	5
3.2.2 Box and Line Architecture Diagram	5
3.2.3 Three-tier Client-Server Architecture	6
3.2.3.1 Project Modules in Three-Tier Model	6
4. Design Models.....	7
4.1 Activity Diagram	7
4.1.1 User Authentication Process	7
4.1.2 VPN Detection Process	8
4.1.3 VPN Blocking Process	8
4.2 Data Flow Diagram	9

4.2.1	Level 0 (Context Diagram)	10
	Description	10
4.2.2	Level 1 DFD.....	10
	Description	10
4.2.3	Level 2 DFD.....	11
	Packet Identification	11
	Description	11
	Diagram.....	11
	Explanation	12
	VPN Blocking.....	12
	Description	12
	Diagram.....	12
	Explanation	12
4.3	State Transition Diagram.....	13
4.4	Sequence Diagram.....	14
	Diagram.....	15
	Explanation	15
5.	Data Design.....	16
5.1	Data Structures and Storage	16
5.1.1	ERD Diagram.....	16
5.1.2	XML Schema	17
	Description of the Schema	18
	XML Schema	18
5.2	Data Dictionary	19
6.	User Interface Design	20
6.1	Functionality from User's Perspective	20
6.1.1	Login & Authentication Screen.....	20
6.1.2	Dashboard/Home Screen.....	20
6.1.3	VPN detection and Analysis Section	20
6.1.4	VPN Management Section	20
6.1.5	Settings & Configuration	20
6.1.6	Reports & Logs	20
6.2	Screen Images.....	21
6.2.1	Login & Authentication Screen.....	21
6.2.2	Dashboard/Home Screen.....	21
6.2.3	VPN detection and Analysis Section	22
6.2.4	VPN Management Section.....	22

6.2.5	Settings & Configuration:	23
6.2.6	Reports & Logs:	23
6.3	Screen Objects and Actions.....	24
6.3.1	Navigation Bar	24
6.3.2	Real-time Traffic Graph (Dashboard)	24
6.3.3	VPN Traffic List (Analysis Screen)	24
6.3.4	Alerts List.....	24
6.3.5	Historical Data Charts	24
6.3.6	Account Management Forms	24
6.3.7	Breakdown into smaller parts.....	24

List of Tables

Table 1: Definitions, Acronyms, and Abbreviations	2
---	---

List of All Diagrams

Figure 1: System Architecture	5
Figure 2: Client-Server Architecture	6
Figure 3: User Authentication Process	7
Figure 4: VPN Detection Process	8
Figure 5: VPN Blocking Process	9
Figure 6: Level 0 (Context Diagram).....	10
Figure 7: Level 1 DFD	11
Figure 8: Packet Identification Level 2.....	11
Figure 9: Packet Blocking Level 2.....	12
Figure 10: State Transition Diagram.....	14
Figure 11: Sequence Diagram.....	15
Figure 12: ERD Diagram	17
Figure 13: Login & Authentication Screen.....	21
Figure 14: Dashboard/Home Screen	21
Figure 15: VPN detection and Analysis Section.....	22
Figure 16: VPN Management Section	22
Figure 17: Settings & Configuration.....	23
Figure 18: Reports & Logs	23

List of Activity Diagrams

Figure 3: User Authentication Process	7
Figure 4: VPN Detection Process	8
Figure 5: VPN Blocking Process	9

List of Data Flow Diagrams

Figure 6: Level 0 (Context Diagram).....	10
Figure 7: Level 1 DFD	11
Figure 8: Packet Identification Level 2.....	11
Figure 9: Packet Blocking Level 2.....	12

List of User Interface Diagrams

Figure 13: Login & Authentication Screen.....	21
Figure 14: Dashboard/Home Screen	21
Figure 15: VPN detection and Analysis Section.....	22
Figure 16: VPN Management Section	22
Figure 17: Settings & Configuration.....	23
Figure 18: Reports & Logs	23

Revision History

Name	Date	Reason for changes	Version
Hassan	1 st October, 2023	Introduction & Design Methodology	1.1
Sameen	20 th October, 2023	Software Design & Architecture Design	1.2
Sameen	15 th November 2023	Client Server Model & Data Flow Diagram	1.3
Hassan	20 th November 2023	Box line & Data Design	1.4
Hassan	8 th December 2023	Activity Diagrams & Data Design	1.5
Sameen	20 th December 2023	State Transition & User Interface	1.6

Application Evaluation History

Comments (by committee) *include the ones given at scope time both in doc and presentation	Action Taken

Supervised by
Supervisor's Name

Signature_____

1. Introduction

1.1 System Purpose

This document is the next step work after the completion of Software Requirements Specification (SRS). The purpose of this document is to describe the detail architecture and design specifications for the project entitled “*VPN Spyglass*”. This project aims to deliver robust, user friendly open-source tool for detecting and analyzing VPN activity in a network. It empowers network managers to identify, categorize and gain meaningful insights into VPN usage. Using cutting-edge deep packet inspection, the program distinguishes between regular and VPN traffic based on protocols. The ultimate goal is to enhance network monitoring, detect unauthorized VPN connections, and mitigate security threats. The real-time insights are accessible through a dynamic online dashboard built with modern technologies like ReactJS, Django and Python. This project bridges the gap between powerful network analysis and an intuitive interface, offering a comprehensive solution to control VPN usage and bolster network security.

1.2 System Scope

The VPN Spyglass system's goal is to provide advanced capabilities to network administrators for detecting, categorizing, and regulating VPN traffic within their network. The solution attempts to improve network security and visibility while also providing a user-friendly monitoring and management interface. For the VPN Spyglass system, we have been able successfully implement three critical modules:

- 1. VPN Detection Module:** Through rigorous development and deployment, we have incorporated robust packet analysis techniques into the VPN Spyglass system. This module effectively identifies VPN traffic within your network, offering network administrators a comprehensive tool for detecting and monitoring VPN activities.
- 2. Categorization Module:** Our team has meticulously crafted algorithms within the system to categorize VPN traffic based on distinct VPN protocols or types. This strategic enhancement significantly augments network visibility, enabling administrators to precisely classify and understand the nature of VPN traffic flowing through the network.
- 3. Web Dashboard Development:** The implementation of a sophisticated web dashboard is a hallmark of our work. We have successfully designed and integrated a user-friendly interface that provides real-time traffic visualization and control options. This dashboard empowers network administrators with the tools needed to efficiently monitor and manage VPN traffic, contributing to an enhanced level of network security and visibility.

1.3 Definitions, Acronyms, and Abbreviations

Word	Explanation
SRS	A document that specifies the functional and non-functional requirements of a software project.
VPN	A secure network connection that enables users to communicate over a public network connection as if their computing devices are directly connected to a private network.

API	A set of rules that allows one software application to interact with another one.
OpenFlow	A communication protocol that enables control plane at network layer to direct traffic on a device connected to network
BMV2	A software switch used for testing a prototyping of OpenFlow-enabled devices
MVC	A software design pattern that separates an application into three connected components: model (for data and business logic), View (For user interactivity), Controller (for handling user input and updates in model)
Firewall	A network security system that is used to monitor and control incoming and outgoing network traffic based on predetermined security rules.
API calls	Requests made by one application to another through defined interfaces.
Dashboard	A visual representation of data, often providing real time insights and analytics.
Network Congestion	A condition in data networking where network is queued with so much data that it slows down or disrupts normal flow of traffic.

Table 1: Definitions, Acronyms, and Abbreviations

2. Design Methodology and Software Process Model

In the development of our project, a meticulous choice of design methodologies has been made for the creation of an efficient system. The project being divided into two parts, the front-end and backend required us to ponder on the most suitable approach for both of them while maintaining consistency in the project. The front-end is crafted using React, following procedural design methodology, which compliments Reacts component-based architecture and declarative approach. Simultaneously, the backend is implemented in Django using python, embracing procedural principles. This not only provides consistency throughout the project but also enables a systematic and structured development process providing clarity in both user interface design and data management.

In the subsequent sections, the choice of the procedural design methodology and process model to be used will be explored and justified accordingly and in greater detail.

2.1 Design Methodology

While *Procedural Approach* is not a common design methodology being used by the developers, the modern tool like React uses a declarative approach for the development of the client-side of the project. Procedural Approach is often a valid approach for the backend development using Django.

The rationale for employing the procedural design methodology on both the client-side and server-side modules is expounded upon in the subsequent explanation.

2.1.1 Website based dashboard.

The utilization of a procedural design methodology in developing a React-based website dashboard is justified as for the following reasons:

- **Modular Component Composition:** React uses Component-based Architectural approach so procedural design facilitates the decomposition of the dashboard into modular components, enhancing the maintainability and clarity in Structuring UI elements.
- **Effective Event Handling:** Procedural design effectively handles user interactions and events, enabling the definition of specific procedures for known tasks related to event-driven functionalities.
- **Structured Control Flow:** The procedural paradigm aids the component-based Architecture of React as it allows for a systematic control flow, aiding in step-by-step execution of data processing tasks, contributing to a well-organized codebase within the dashboard components.

2.1.2 Integration of Switch/Firewall

The adoption of a procedural design methodology for integrating a Switch/Firewall into React-based application within the Django Framework, rather than opting for an object-oriented approach is substantiated by:

- **Modularity and Code Separation:** Procedural design streamlines the process of integrating Switch/ Switch/Firewall by separating concerns into distinct procedures, enhancing maintainability. In contrast, an object-oriented approach might complicate modularization with a more interconnected class structure. The procedural approach's emphasis on discrete procedures aligns well with the segmented nature of Switch/Firewall operations, resulting in simpler and modular codebase.
- **Simplicity and Readability:** Procedural Design approach with its emphasis on functional programming approach, can often lead to more straightforward and readable code. Integration of Switch/Firewall allows to execute only specific operations and procedures so in this case procedural approach enhances code clarity while using OOP might end up into unnecessary creation of classes having complex relation with each other.
- **Practicality and Specific Use Cases:** For certain functional requirements such as configuring and managing Switch/Firewall rules through a request from client-side, a procedural approach may be more practical and directly aligned with the task's procedural nature. OOP, while powerful in other contexts, might introduce unnecessary complexity for this specific use case.

2.1.3 User Authentication

The utilization of a procedural design methodology for the development of a database storing the user data for authentication purposes is justified as for the following reasons:

- **Modular Authorization Logic:** Chosen Design approach enhances code modularity in Django, allowing discrete steps for user authentication for maintainability and readability purposes. Using OOP may increase the complexity for straight forward authentication. It will add unnecessary abstraction layer to simple authentication logic.
- **Interoperability with Django's Class-Based Views:** Procedural design seamlessly integrates with Django's class-based views, aligning with the framework's conventions. OOP, while capable, introduces a steeper learning curve for developers unfamiliar with the framework's conventions.

2.2 Software Process Model

Given the ever-changing landscape of network dynamics and security concerns, the most suitable software process model is **`Agile (Iterative) Development`**. This software process model accommodates frequent modifications in project development seamlessly. Agile provides flexibility to discard or modify project functionality at any stage without disrupting other phases in Software Development Life Cycle (SDLC). More advantages associated with this process model is explained in greater details in subsequent section.

- **Iterative Development:** The dynamic nature of the network security aligns with the agile iterative approach, enabling frequent reassessment and adaption to evolving requirements. Its incremental feature delivery is well-suited for VPN Spyglass, facilitating the gradual development and delivery of functional requirements of VPN Spyglass based on their priority.
- **Collaboration and Adaptability:** The essential role of cross-functional teams in Agile is evident in VPN Spyglass, promoting collaboration between development of User Interface and its integration with database and Switch/Firewall. The iterative feedback loop in Agile ensures constant input, crucial for enhancing the services of our project.
- **Flexibility and Adaption:** It accommodates changes in the security landscape, enabling the team to address new threats and regulatory requirements in subsequent iterations. The model's responsiveness ensures prompt updates to security measures and integration of new features of modifying the dynamics of built features.

3. System Overview

VPN Spyglass is implemented using ReactJS, Django and Python. ReactJS is a powerful JavaScript framework which will be used in the frontend or user interface of our Project. Django provides the set of tools and conventional method making the data handling tasks very convenient, so it is being as backend language.

The main purpose of the system is to give meaningful analysis of the network traffic and clear categorization between normal traffic and VPN traffic.

3.1 System Operations and Requirements coverage

VPN Spyglass, our innovative project, provides an online portal for login/register. The authenticated users are provided with real-time visibility into network traffic dynamics. Through a web-based dashboard, users gain insights into the percentage of VPN traffic within a network, uncovering crucial information about state of network. The dashboard also highlights the top VPNs currently in use as well as ranks the most utilized VPNs over past week, empowering the network administrators to take timely decisions about network security and its optimization.

In this system, the database plays a crucial role in storing user information and related activities. Live network traffic is dynamically displayed through API calls. Users can proactively manage the network using the analytics provided by the dashboard, enabling actions such as blocking a specific VPN in response to network congestion.

3.2 Architectural Design

The system's modular structure is designed to achieve a comprehensive functionality through the collaboration of distinct modules (*as shown in Figure 1*). The high-level overview aims to provide a general understanding of the system's decomposition and the interplay between the individual modules.

3.2.1 Inter-Module Dependencies

3.2.1.1 User Interfaces

- User Interfaces represent the front-end layer.
- Communicates user inputs and requests to the corresponding module.

3.2.1.2 Controllers

- Serves as the intermediary between the UI components and the backend modules.
- Synchronize the flow of requests and corresponding actions to be taken.

3.2.1.3 Real-time Traffic Module

- Utilizes API calls for real-time traffic information.
- Collaborates with VPN categorization module to distinguish VPN traffic from regular network traffic.

3.2.1.4 VPN Categorization Module

- Collaborates with the Live Traffic module to categorize traffic and distinguish VPN connections.
- Feeds the distinguished data to the controller for further processing.

3.2.1.5 VPN Blocking Module

- Implements logic to monitor VPN traffic thresholds and dynamically block VPN connections.
- Also allows manual blocking of specific VPNs based on user or system decisions.

3.2.1.6 Backend Services

- Comprises multiple modules handling specific functionalities (e.g., data preprocessing, business logic or external integrations).
- Communicates with each other through well-defined interfaces.

3.2.1.7 Database Management

- Manages data storage and retrieval for the system.
- Connected to backend modular services for seamless data flow.

3.2.2 Box and Line Architecture Diagram

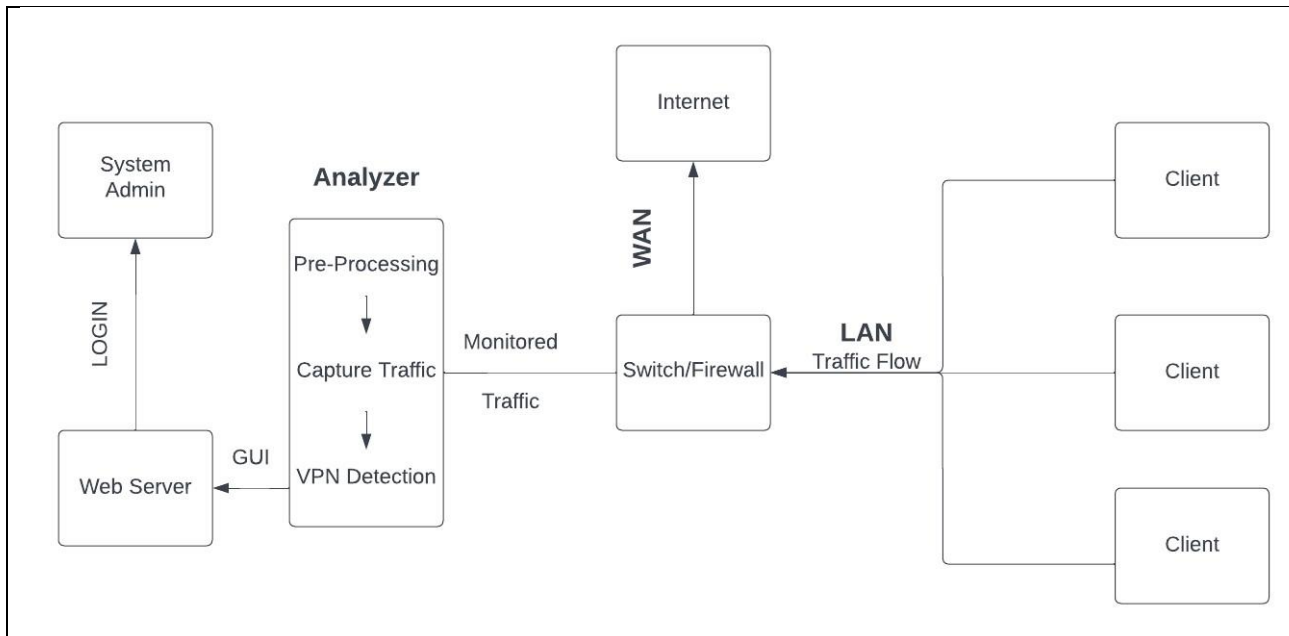


Figure 1: System Architecture

3.2.3 Three-tier Client-Server Architecture

The *three-tier client-server model* is a widely adopted architectural pattern that organizes software application into three distinct layers, each responsible for specific functions. The utilization of this model is justified by its capability to enhance modularity, scalability and maintainability by separating the user interface (presentation tier), application logic (application tier) and data management (data tier) (as shown in Figure 2). This division of responsibilities simplifies development, maintenance, and scalability of software systems.

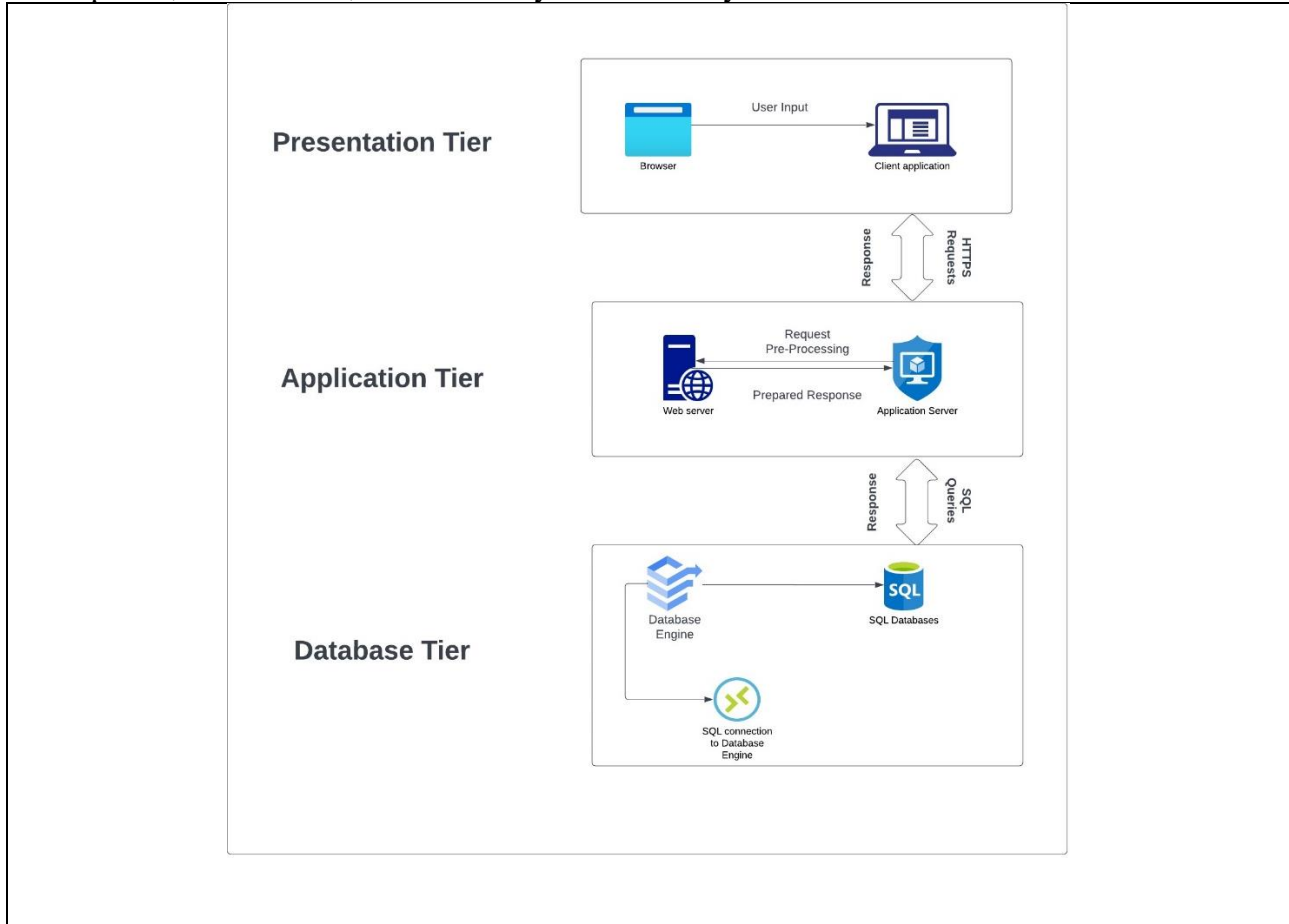


Figure 2: Client-Server Architecture

3.2.3.1 Project Modules in Three-Tier Model

Now let's map the modules/components of VPN Spyglass to three-tier client-server model.

Presentation Tier: The web dashboard developed with ReactJS, Django and python, resides in this tier. It allows to interact with the application layer of VPN detection system.

Application Tier (Server): The backend components involving Mininet, BMV2 switch, Python, P4 language, and the web server, are part of the application tier. It manages the business logic, communication with the data tier, and overall functionality of the system.

Data Tier (Database): The data tier is SQL database which will be used to store data for user authentication. Moreover, The BMV2 switch can also act as data storage and management components as they store and provide access to real-time data for web interface.

The interactions between the intermodular components and how they communicate with each other is shown below:

1. **Browser to Client-Side Application:** The user interacts with the client-side application in the presentation tier.
2. **Client-Side Application to Application Server:** Requests for data or services from the application tier.
3. **Application Server to Business Logic Layer:** Transfer requests from the presentation tier to the business logic layer for processing.
4. **Application Server to Data Tier:** Retrieves or updates data in the data tier based on requests from presentation tier.

4. Design Models

4.1 Activity Diagram

4.1.1 User Authentication Process

The User Authentication Process is a crucial component of the VPN Spyglass system. It ensures the security and integrity of the system by verifying the identity of users attempting to access it. This process is vital because it safeguards the system from unauthorized access, protecting sensitive data and resources, and ensuring that only authorized personnel can utilize the VPN Spyglass network monitoring and management capabilities.

The process of User Authentication can be expressed in following steps:

1. **Start:** The process begins.
2. **Enter Credentials:** User inputs username and password.
3. **Send Authentication Request:** Credentials are sent to the Authentication Server.
4. **Verify Credentials:** Server checks credentials against the database.
5. **Authentication Successful?**
 - a. Yes: Proceed to "Access Granted."
 - b. No: Go back to "Enter Credentials."
6. **Access Granted:** User gains access to the system.
7. **End:** The process concludes.

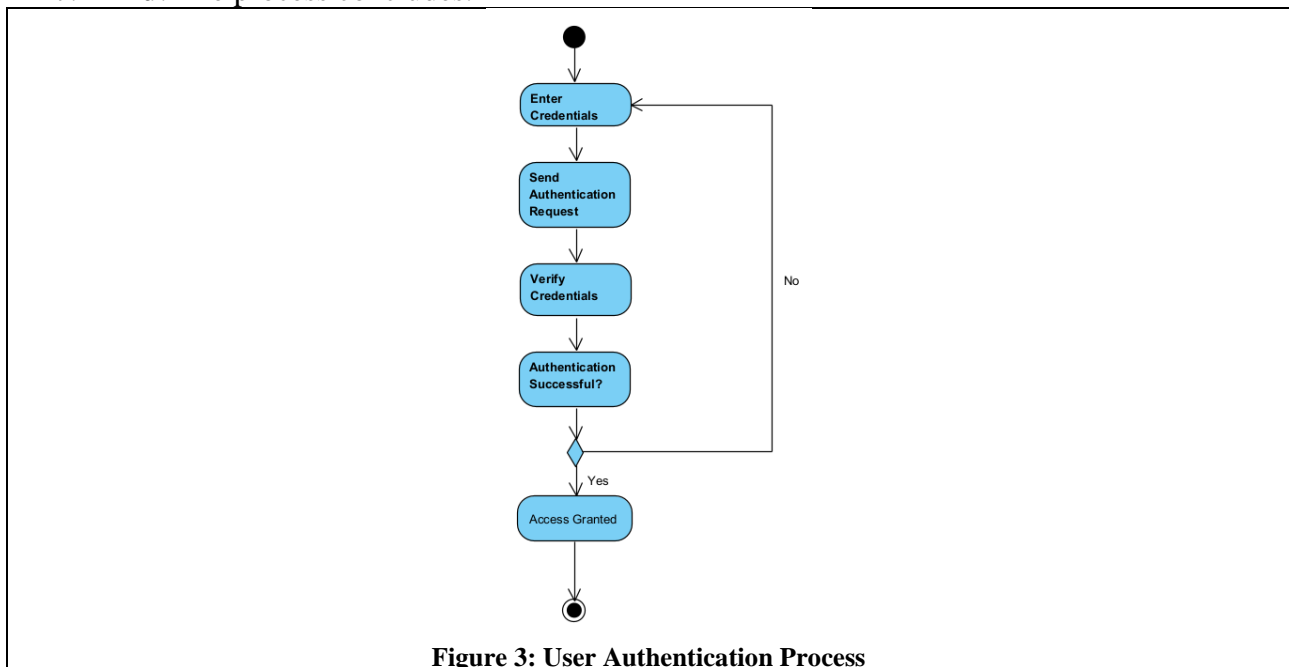


Figure 3: User Authentication Process

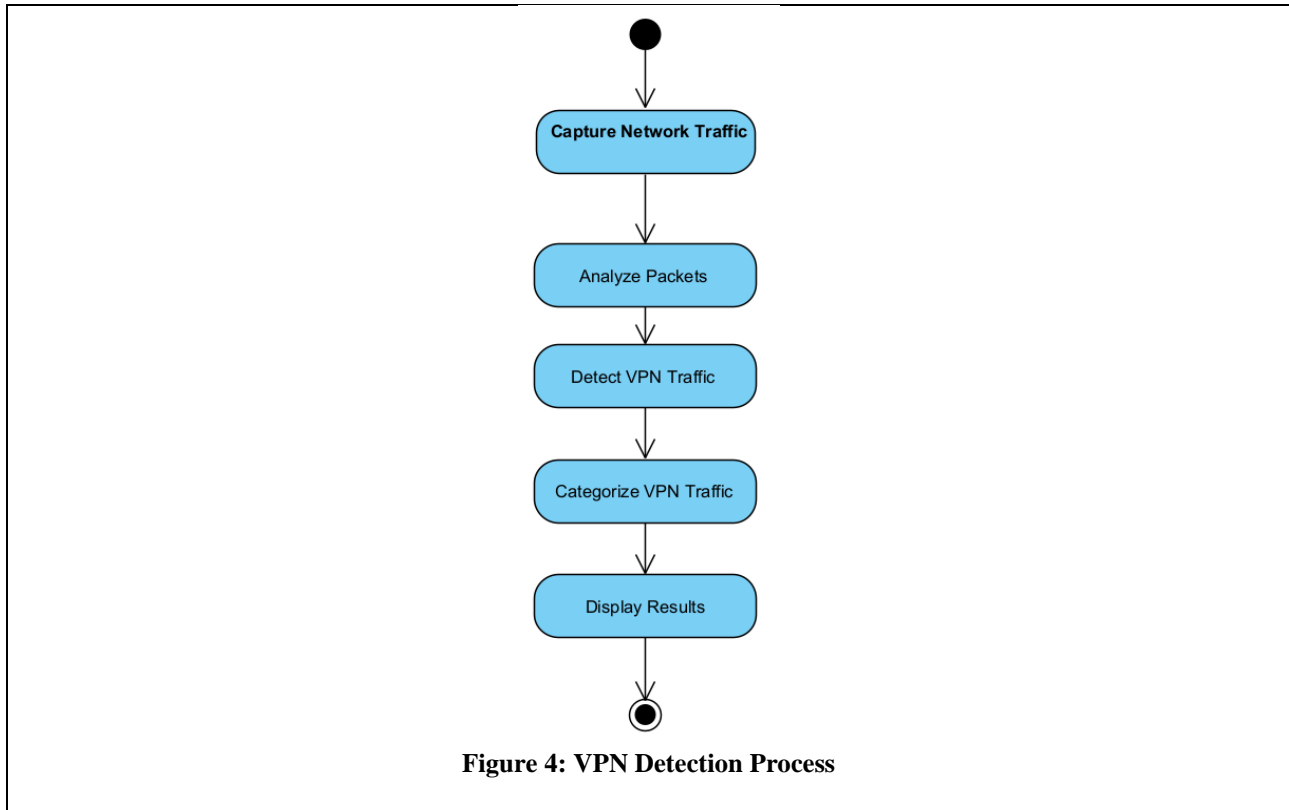
4.1.2 VPN Detection Process

The VPN Detection Process is a critical component of the VPN Spyglass system, playing a pivotal role in ensuring network security. This process involves capturing network traffic, analyzing packets using deep packet inspection, and identifying potential VPN traffic.

This process is of utmost importance as it helps administrators proactively detect unauthorized VPN usage, providing an essential layer of security to prevent potential data breaches and unauthorized access to the network. It empowers administrators with the means to enforce network policies effectively and maintain the integrity of their network infrastructure.

The process of VPN Detection can be expressed as follows:

1. **Start:** The process begins.
2. **Capture Network Traffic:** System captures packets from the network.
3. **Analyze Packets:** Deep Packet Inspection is performed.
4. **Detect VPN Traffic:** System identifies potential VPN traffic.
5. **Categorize VPN Traffic:** System classifies the VPN traffic by protocol or type.
6. **Display Results:** Results are shown on the dashboard.
7. **End:** The process concludes.



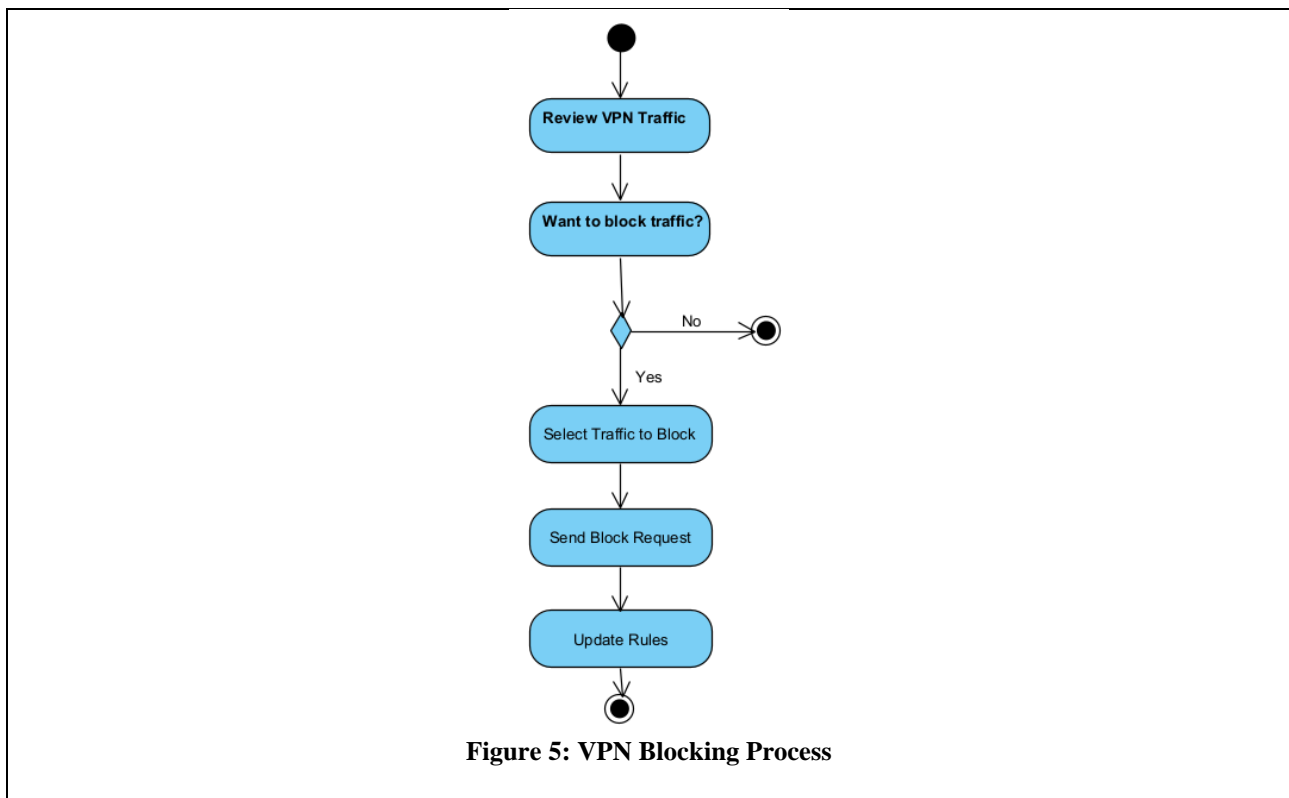
4.1.3 VPN Blocking Process

The VPN Blocking Process is a critical security measure in network administration that involves the identification and restriction of unauthorized VPN (Virtual Private Network) traffic within a network. This process is of paramount importance as it helps network administrators maintain control over their network resources, prevent potential security breaches, and enforce corporate

policies by selectively blocking VPN connections that may circumvent security protocols or access restrictions.

The process of VPN blocking can be expressed as follows:

1. **Start:** The process begins.
2. **Review VPN Traffic:** Administrator reviews identified VPN traffic.
3. **Want to block the traffic?**
 - Yes: Continue to “Select Traffic to Block”
 - No: End (The process concludes)
4. **Select Traffic to Block:** Administrator selects unauthorized VPN traffic.
5. **Send Block Request:** Request is sent to VPN Blocking Module.
6. **Update Rules:** VPN Blocking Module activates block rules.
7. **End:** The process concludes



4.2 Data Flow Diagram

The VPN SpyGlass project's Data Flow Diagrams (DFDs) offer a thorough visual depiction of the flow and processing of data inside the system. A sophisticated program called VPN SpyGlass is made to watch over, examine, and control VPN activity inside a network architecture. These DFDs are meant to provide an organized and transparent picture of the system's operation by decomposing its intricate workings into simpler parts.

The DFDs are organized into layers that offer differing amounts of detail:

- **Level 0 DFD:** The context diagram, or level 0 of DFD, is the most abstract level. It shows how the VPN SpyGlass system interacts with outside parties such as network

administrators and the network infrastructure by presenting it as a single process. grasp the system's border and external interfaces requires a grasp of this diagram.

- **Level 1 DFD:** In this level, the primary process from the Context Diagram is dissected into its individual subprocesses. It explores the main features of VPN SpyGlass, such as data analysis, report creation, network traffic monitoring, and VPN traffic control. This level offers a more thorough examination of the system's network data processing and data storage interactions.
- **Level 2 DFD:** In this stage, every Level 1 subprocess is further broken down to disclose its inner workings. This level offers more detail on the particular functions of the system, including rule enforcement, packet capture, signature matching, and report visualization.

4.2.1 Level 0 (Context Diagram)

Description

- **System:** VPN SpyGlass.
- **External Entities:** Web Dashboard, and Switch/Firewall
- **Data Flows:** User commands, Traffic data, Update Firewall Rules, and Simulated Traffic.

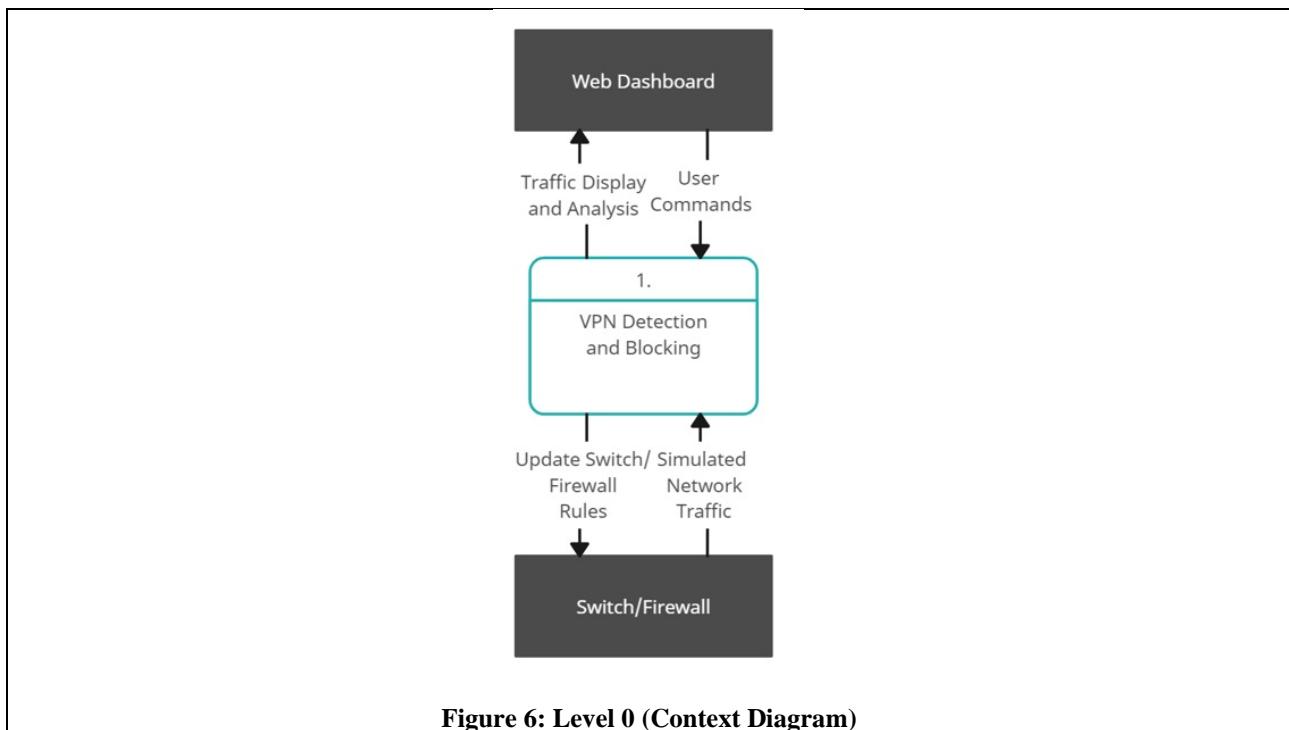


Figure 6: Level 0 (Context Diagram)

4.2.2 Level 1 DFD

Description

- **Processes:** User Interface, Traffic Monitoring, VPN Categorization, and VPN Blocking.
- **Data Stores:** Database.
- **Data Flows:** Enter Credentials, Verify Credentials, Access Granted, Block Traffic, Display Traffic, Categorized VPNs, Analysis, Simulated Traffic, and Command to Update Rules.

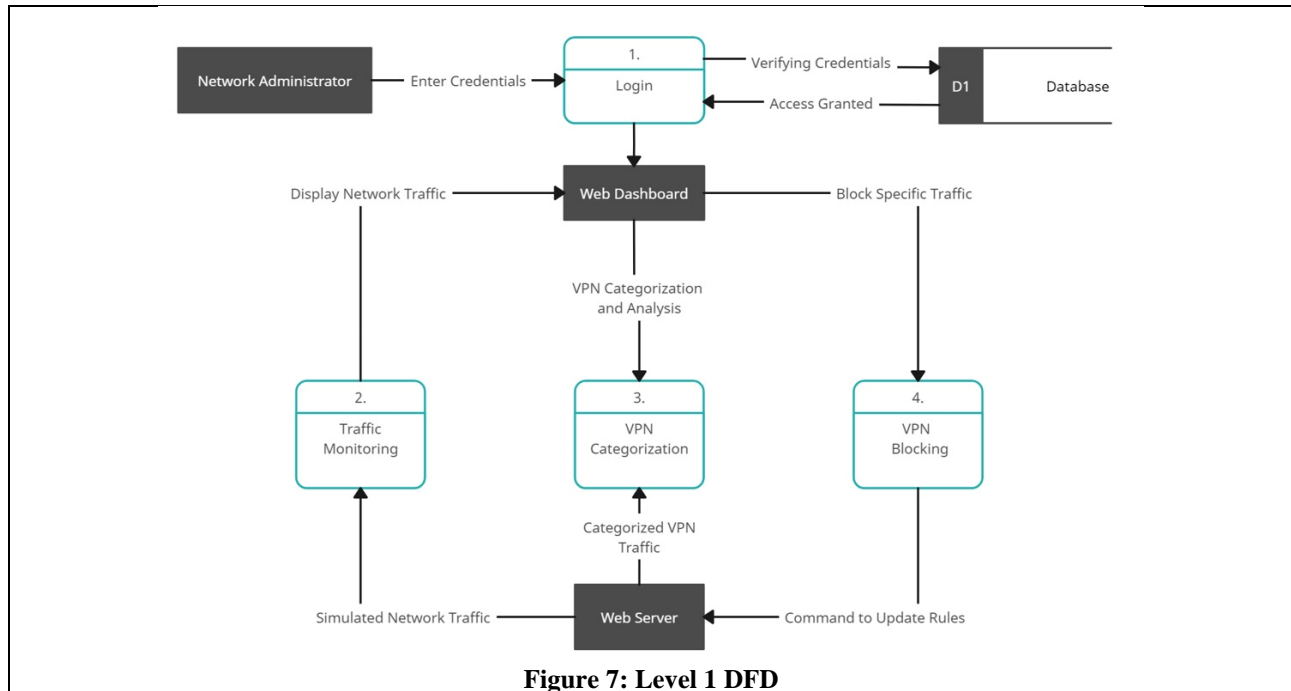


Figure 7: Level 1 DFD

4.2.3 Level 2 DFD

Packet Identification Description

- **Processes:** Receive Packet Data, Signature Pattern Matching, VPN Protocol Identification, Generate Match Report.
- **Data Stores:** Signature Patterns, Protocol Identification Rules, Match Reports.
- **Data Flows:** Packet data, signature match queries, protocol identification data, and report generation details.

Diagram

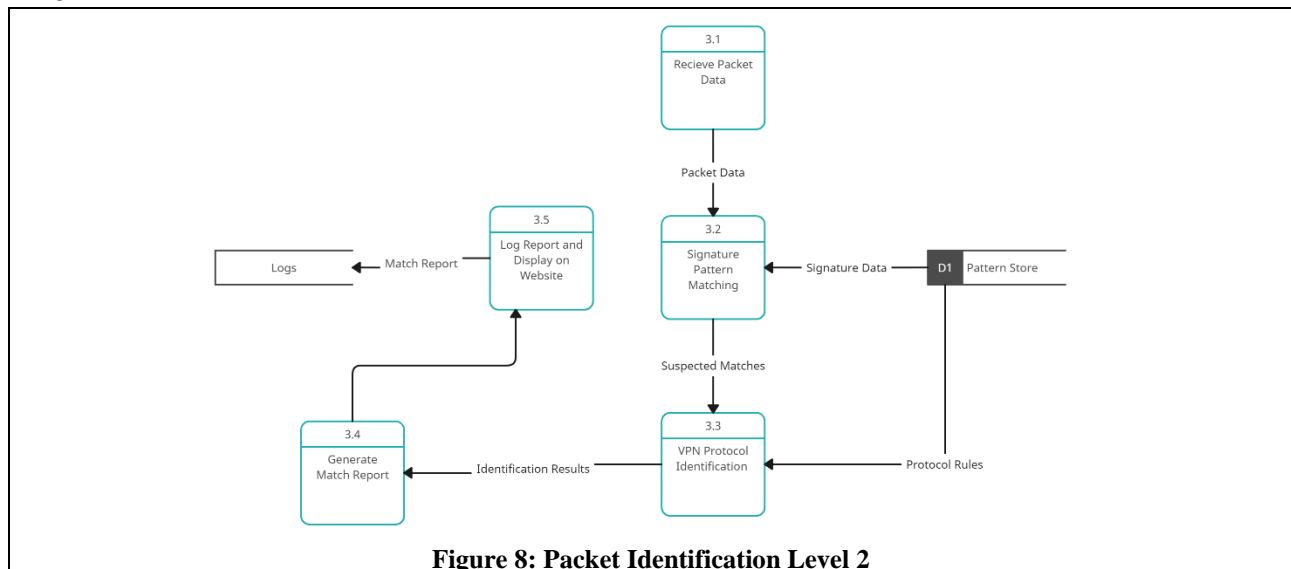


Figure 8: Packet Identification Level 2

Explanation

- **Receive Packet Data:** This process involves receiving the raw packet data from the previous stage.
- **Signature Pattern Matching:** Compares packet data against a database of signature patterns.
- **Patterns Store:** A data store containing various signature patterns used for matching against packet data.
- **VPN Protocol Identification:** Determines the specific VPN protocol used if a VPN signature is matched.
- **Protocol Rules:** A set of rules or criteria used to identify different VPN protocols.
- **Generate Match Report:** Creates a detailed report based on the signature matching and protocol identification results.
- **Match Reports:** A repository where these detailed match reports are stored.
- **Log Analysis:** The process where these reports are analyzed, and actions are determined.

VPN Blocking

Description

- **Processes:** Receive Block Request, Validate VPN, Update Blocking Rules, and Apply Blocking.
- **Data Stores:** VPN Identity Database
- **Data Flows:** Block requests, rule updates, and applied block.

Diagram

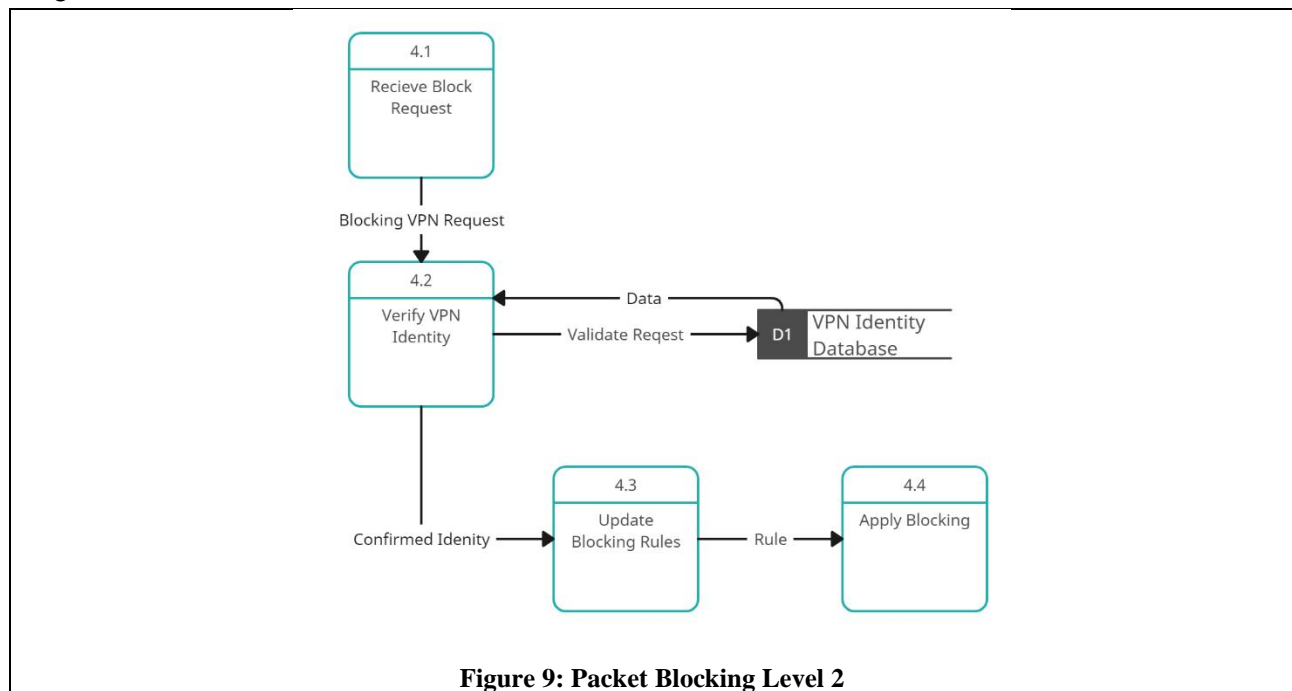


Figure 9: Packet Blocking Level 2

Explanation

- **Receive Block Request:** This process starts with receiving a request to block a specific VPN connection.
- **Validate VPN Identity:** Validates the identity of the VPN connection against a database to ensure accuracy.
- **VPN Identity Database:** A data store containing details about known VPN connections and identifiers.

- **Update Blocking Rules:** Updates the system's blocking rules with the new VPN to be blocked.
- **Blocking Rules:** A database of rules used by the system to block VPN traffic.
- **Apply Blocking:** The process where the updated rules are applied to the network, effectively blocking the VPN traffic.
- **VPN Management:** The higher-level process that oversees and manages the VPN blocking actions.

4.3 State Transition Diagram

For the purpose of the VPN SpyGlass project, a Behavioral State Machine Diagram must be created. This entails outlining all the states and transitions that are pertinent to the system's functioning, with special attention paid to how the system responds to user interactions and network activity.

State Working

What each state does is as follows:

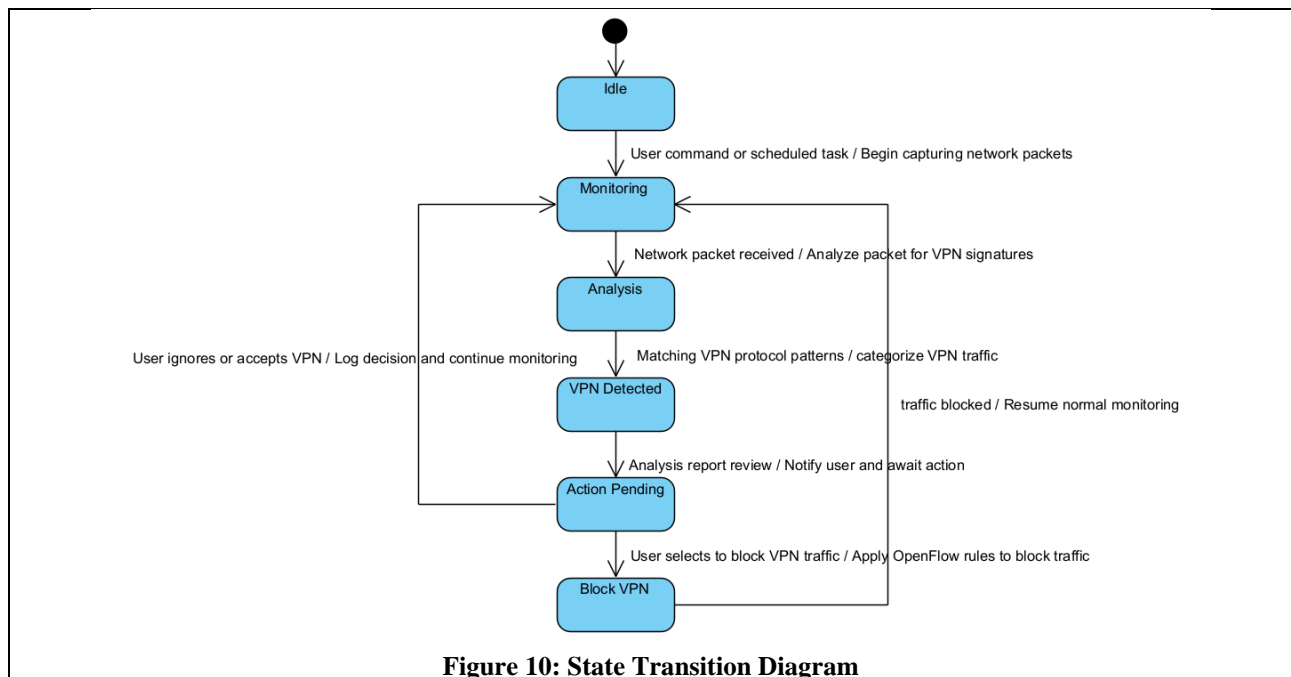
1. **Idle State:** Waiting for a scheduling trigger or user action to initiate monitoring.
2. **Monitoring State:** obtaining network packets and examining them to look for VPN signatures.
3. **Analysis State:** VPN traffic is identified using deep packet inspection.
4. **VPN Detected State:** VPN communication is recognized and recorded.
5. **Action Pending State:** User-selected handling of observed VPN traffic is still pending.
6. **Block VPN State:** Uses user-defined criteria to prevent certain VPN traffic from passing through.
- 7.

Transitions

The state transition can be given as:

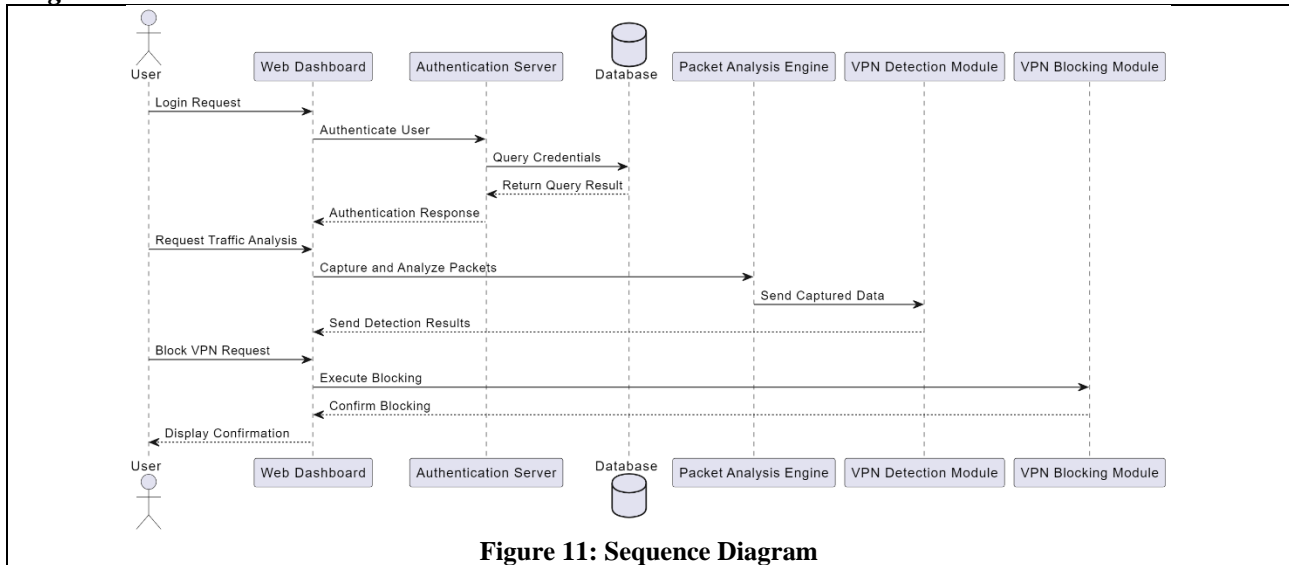
1. **Idle State:** The system is running but not actively monitoring network traffic.
 - **Transition:** Start monitoring.
 - **Event:** User command or scheduled task.
 - **Action:** Begin capturing network packets.
2. **Monitoring State:** The system is actively monitoring network traffic.
 - **Transition to Analysis:** Packet capture.
 - **Event:** Network packet received.
 - **Action:** Analyze packet for VPN signatures.
3. **Analysis State:** The system is analyzing a network packet.
 - **Transition to VPN Detected:** VPN signature found.
 - **Event:** Matching VPN protocol patterns.
 - **Action:** Log and categorize VPN traffic.
4. **Transition to Monitoring:** Packet analyzed.
 - **Event:** Analysis complete.
 - **Action:** Resume monitoring.
5. **VPN Detected State:** A VPN packet has been identified.

- **Transition to Action Pending:** User or system decision.
 - **Event:** Analysis report review.
 - **Action:** Notify user and await action.
6. **Action Pending State:** Awaiting user decision on detected VPN traffic.
- **Transition to Block VPN:** User command.
 - **Event:** User selects to block VPN traffic.
 - **Action:** Apply OpenFlow rules to block traffic.
 - **Transition to Monitoring:** User ignores or accepts VPN.
 - **Event:** User decision.
 - **Action:** Log decision and continue monitoring.
7. **Block VPN State:** The system blocks identified VPN traffic.
- **Transition to Monitoring:** Blocking rules applied.
 - **Event:** VPN traffic blocked.
 - **Action:** Resume normal monitoring.



4.4 Sequence Diagram

The sequence diagram given here is intended to provide a clear and systematic visual representation of a user's interactions with the VPN SpyGlass system. The graphic depicts the communication flows that occur when a user interacts with the system's online dashboard and associated backend services for authentication, traffic analysis, and VPN blocking.

Diagram**Figure 11: Sequence Diagram****Explanation****1. Login Procedure for Users:**

- A Login Request to the online dashboard is initiated by the user.
- This request is forwarded to the Authentication Server via the web dashboard.
- With the user's credentials, the Authentication Server searches the Database.
- The query result is returned to the Authentication Server by the Database.
- The Authentication Server returns to the web dashboard an Authentication Response, which presumably indicates whether the login attempt was successful.
- The conclusion is displayed to the user via the online dashboard.

2. Request for Traffic Analysis

- After logging in successfully, the user requests Traffic Analysis using the online dashboard.
- The online dashboard starts the Capture and Analyze Packets process using the Packet Analysis Engine.
- The Packet Analysis Engine returns the Detection Results to the online dashboard after the analysis is complete.
- The online dashboard shows the user the results of the analysis.

3. VPN Blocking Request

- If a user detects unwanted VPN traffic, they may utilize the online dashboard to submit a VPN Blocking Request.
- The VPN Blocking Module is instructed to execute blocking by the online dashboard.
- The VPN Blocking Module then validates the activity by returning to the web dashboard with a Confirm Blocking message.
- Finally, the web dashboard displays a Display Confirmation message to the user, indicating that the VPN has been successfully blocked.

5. Data Design

The VPN SpyGlass project's Data Design section is an essential part that painstakingly describes how the system's informational domain is transformed into organized and effective data structures. To ensure the efficient handling of network traffic data, this section explores the nuances of the system's data collection, processing, and organization, with a special emphasis on the identification and management of VPN traffic. Important components include thorough explanations of the links and interactions between the data structures that are used to store user credentials, VPN signatures, packet metadata, and system settings. The VPN SpyGlass system places a strong emphasis on the architecture of databases and data storage techniques, making use of tables and schemas like Entity-Relationship Diagrams (ERDs) to clearly explain how data is kept and retrieved. This method guarantees a strong and dependable tool for network administrators by supporting the system's capability in network monitoring and security management in addition to facilitating effective data processing and retrieval.

The information domain is organized into certain data structures as part of the VPN SpyGlass project's data architecture to enable effective processing, storage, and retrieval. The main data types handled by the system are VPN classification rules, user management data, and network traffic data.

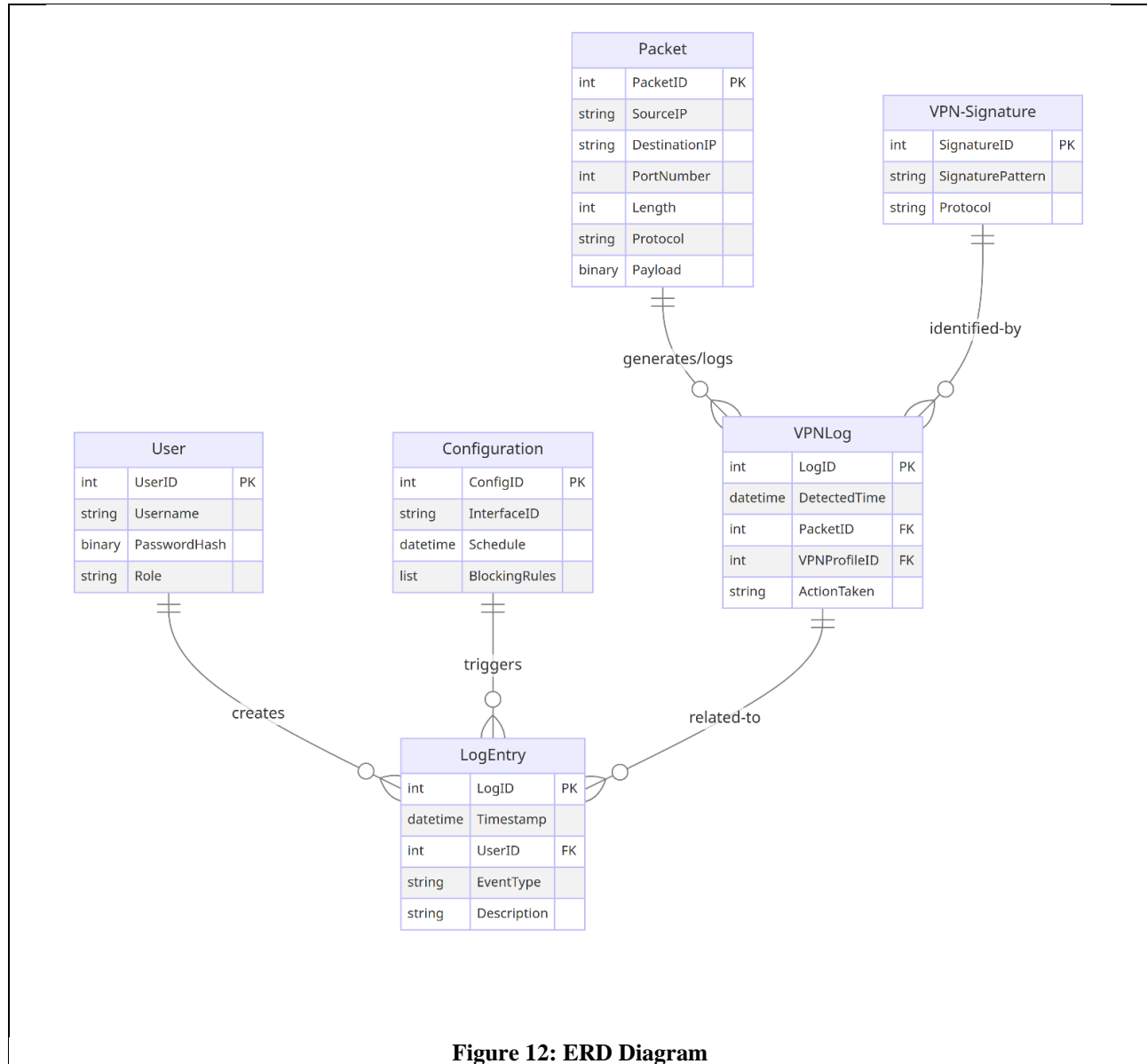
5.1 Data Structures and Storage

- **Network Traffic Data:** This covers network packets that have been intercepted. A data structure including the source and destination IP addresses, port numbers, timestamps, packet lengths, and protocol types is used to represent each packet. After being evaluated in real time for VPN detection, this data is archived for further review.
- **User Management Data:** Includes settings and login credentials for administrators to access the online dashboard. contains user-specific dashboard settings, role information, encrypted password, and username.
- **VPN Signature Database:** A list of well-known VPN patterns and signatures that are utilized for deep packet inspection. Types of VPN protocols, signature patterns, and related metadata are all included in this database.
- **VPN Traffic Logs:** Records of VPN traffic that is identified and stored, including information on the kind of VPN, the source and destination IPs, the detection time, and the action taken (blocked, authorized, etc.).
- **System Configuration Data:** Configuration parameters for the system, such as thresholds for alarms, network interfaces to watch, and other operational details.

5.1.1 ERD Diagram

The VPN SpyGlass project's Entity-Relationship Diagram (ERD) offers a thorough visual depiction of the data linkages and system structures. Understanding how various elements, including packets, VPN signatures, user logs, system configurations, and VPN logs, interact and link inside a network traffic analysis tool is made easier with the help of this graphic. The ERD provides a concise and thorough summary of the system's data architecture by outlining each entity's characteristics and the ways in which they are related to one another. It is a crucial tool for the project's development and maintenance stages, guaranteeing that all data management facets are logically and effectively

integrated to support the main objective of the VPN SpyGlass system, which is to efficiently monitor, identify, and control VPN traffic inside a network environment.



5.1.2 XML Schema

Within network security and monitoring, the VPN SpyGlass project is a state-of-the-art instrument for managing and analyzing VPN traffic. We have implemented an XML schema to arrange the important data elements in a hierarchical and structured fashion in order to manage the complex and varied data that is involved in this process. The configuration settings, user profiles, and log entries, among other elements of the VPN SpyGlass system, must all be represented by this schema. The XML schema guarantees consistency, integrity, and simplicity of data modification and retrieval by giving our data a precise and well-defined structure. It functions as the foundation for information processing and storage, making accurate and efficient VPN traffic analysis possible.

The XML schema used in the VPN SpyGlass project is described in the section that follows. It describes the format and structure of the main data pieces that power our system's operation.

Description of the Schema

- **Configurations:** Shows the settings for the system configuration. The network interface ID, the monitoring schedule, and a set of traffic analysis and blocking rules are all included.
- **User:** Specifies the structure of the user profile. This contains the user's identity, username, password hash (which is saved as a base64 encoded binary for security purposes), and role (administrator, analyst, etc.).
- **LogEntry:** Stores system-generated log entries. Every entry contains an ID, a timestamp, the kind of event, the user ID that started it, and a description.
- **VPNLog:** Dedicated to recording VPN detection occurrences. It contains the VPN profile ID that matched, the log ID, the time of detection, the related packet ID, and the reaction (block, allow, alert) to the detection.

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- Configuration Settings -->
  <xs:element name="Configurations">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="InterfaceID" type="xs:string"/>
        <xs:element name="Schedule" type="xs:dateTime"/>
        <xs:element name="Rules" type="xs:string"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- User Profile -->
  <xs:element name="User">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserID" type="xs:string"/>
        <xs:element name="Username" type="xs:string"/>
        <xs:element name="PasswordHash" type="xs:base64Binary"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Log Entry -->
  <xs:element name="LogEntry">
    <xs:complexType>
```

```

        <xs:sequence>
            <xs:element name="EntryID" type="xs:int"/>
            <xs:element name="Timestamp" type="xs:dateTime"/>
            <xs:element name="UserID" type="xs:string"/>
            <xs:element name="EventType" type="xs:string"/>
            <xs:element name="Description" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- VPN Log -->
<xs:element name="VPNLog">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="LogID" type="xs:int"/>
            <xs:element name="DetectedTime" type="xs:dateTime"/>
            <xs:element name="PacketID" type="xs:int"/>
            <xs:element name="VPNProfileID" type="xs:int"/>
            <xs:element name="ActionTaken" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>

```

5.2 Data Dictionary

1. Network Packet

- **Attributes:** Source IP (String), Destination IP (String), Port Number (Integer), Timestamp (DateTime), Packet Length (Integer), Protocol Type (String).
- **Description:** Represents a network packet captured for analysis.

2. User

- **Attributes:** Username (String), Password (Encrypted String), Role (String), Settings (JSON/Object).
- **Methods:** Authenticate(), UpdateSettings().
- **Description:** User entity for access control to the dashboard.

3. VPN Signature

- **Attributes:** Signature ID (Integer), Protocol Type (String), Pattern (String), Metadata (JSON/Object).
- **Description:** Represents a VPN traffic pattern for detection purposes.

4. VPN Log

- **Attributes:** Log ID (Integer), Detection Time (DateTime), VPN Type (String), Source IP (String), Destination IP (String), Action (String).
- **Description:** Log entry for each VPN traffic detection.

5. System Configuration

- **Attributes:** Config ID (Integer), Network Interfaces (Array of Strings), Alert Thresholds (JSON/Object), Operational Parameters (JSON/Object).
- **Description:** Stores system operational settings.

6. User Interface Design

Now let's design the VPN SpyGlass user interface, emphasizing usability, social media engagement, and system input. The design would aim to provide a smooth and user-friendly interface for network traffic management and VPN traffic monitoring.

6.1 Functionality from User's Perspective

Network administrators, in particular, may easily monitor, analyze, and control VPN traffic on their network thanks to the user-friendly interface of VPN SpyGlass. Important features from the viewpoint of the user consist of:

6.1.1 Login & Authentication Screen

Functionality: To gain secure access, users must first log in using their credentials.

Feedback: Users receive authentication success or failure messages.

6.1.2 Dashboard/Home Screen

Functionality: The main interface displays real-time network traffic as well as VPN operation indicators. Users can start and stop monitoring as well as view active connections.

Feedback: Live network traffic updates, alarms on detected VPN traffic, and system status

6.1.3 VPN detection and Analysis Section

Functionality: Provide comprehensive details, such as IP addresses, protocols, and actions done, regarding VPN traffic that has been discovered.

Feedback: Historical statistics, classification information, and a list of VPN connections that have been found.

6.1.4 VPN Management Section

Functionality: Choices to permit or prohibit particular VPN traffic. Rules and exceptions can be managed by users.

Feedback: Verification of the rules used, connections successfully blocked, and any mistakes.

6.1.5 Settings & Configuration

Functionality: Manage user accounts, set up notification preferences, and alter the tool's settings.

Feedback: System warnings, configuration status, and confirmation of changes.

6.1.6 Reports & Logs

Functionality: Access and produce reports on system performance, user activity, and VPN traffic.

Feedback: Analytic charts, historical logs, and generated reports.

6.2 Screen Images

6.2.1 Login & Authentication Screen

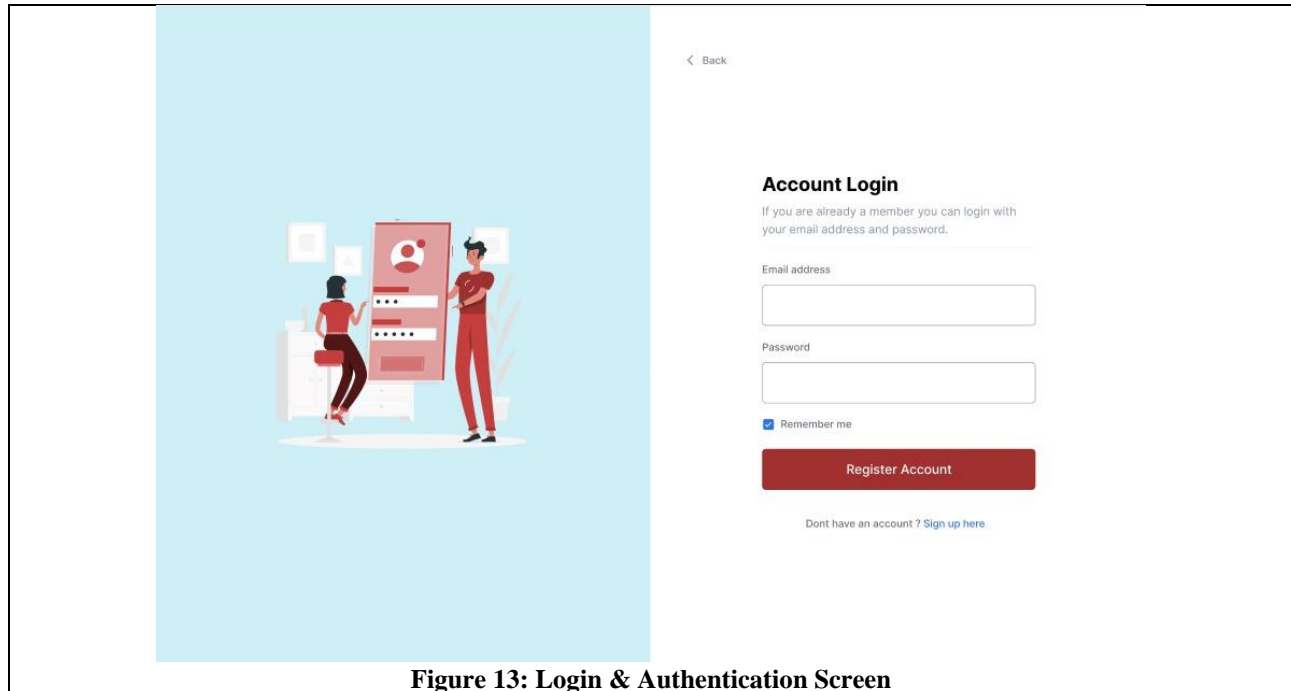


Figure 13: Login & Authentication Screen

6.2.2 Dashboard/Home Screen

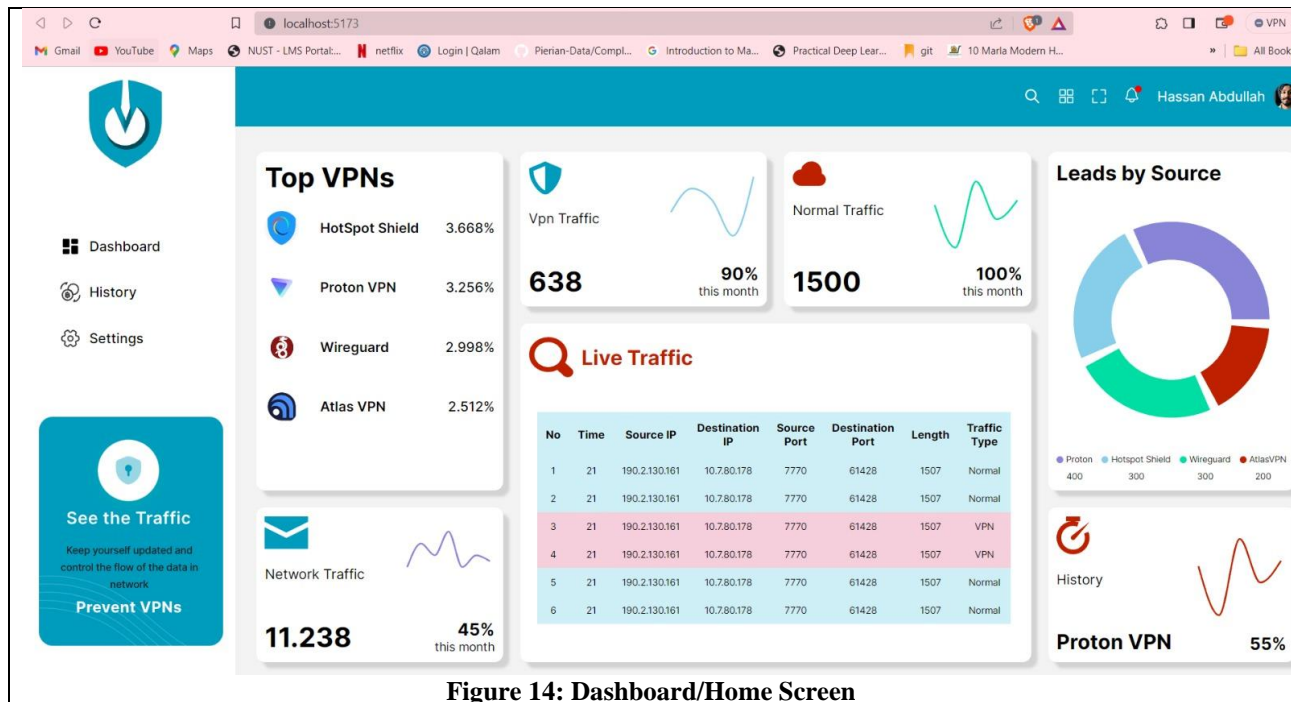
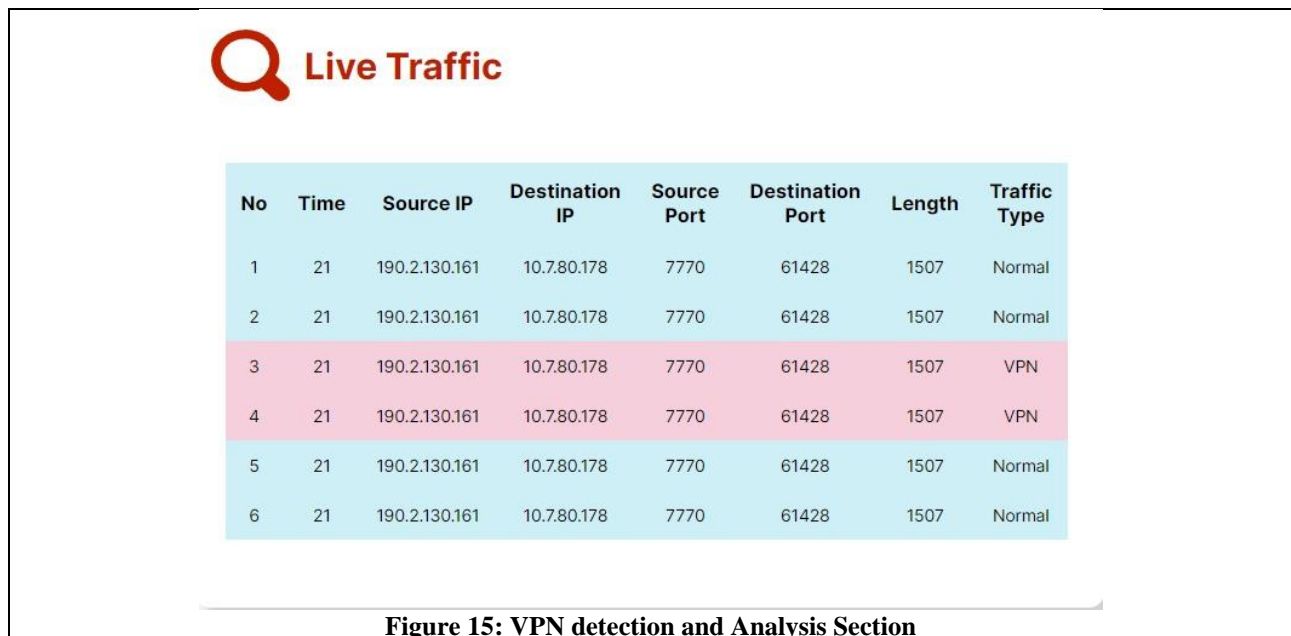
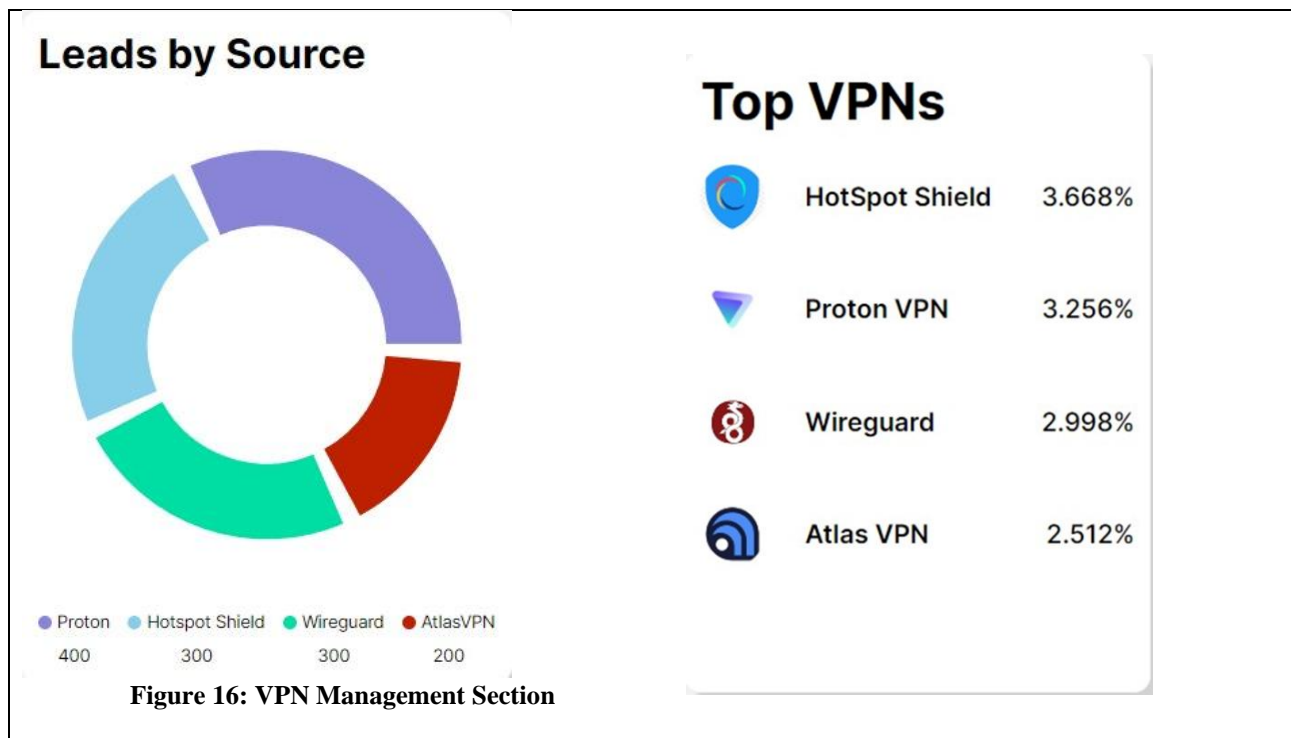


Figure 14: Dashboard/Home Screen

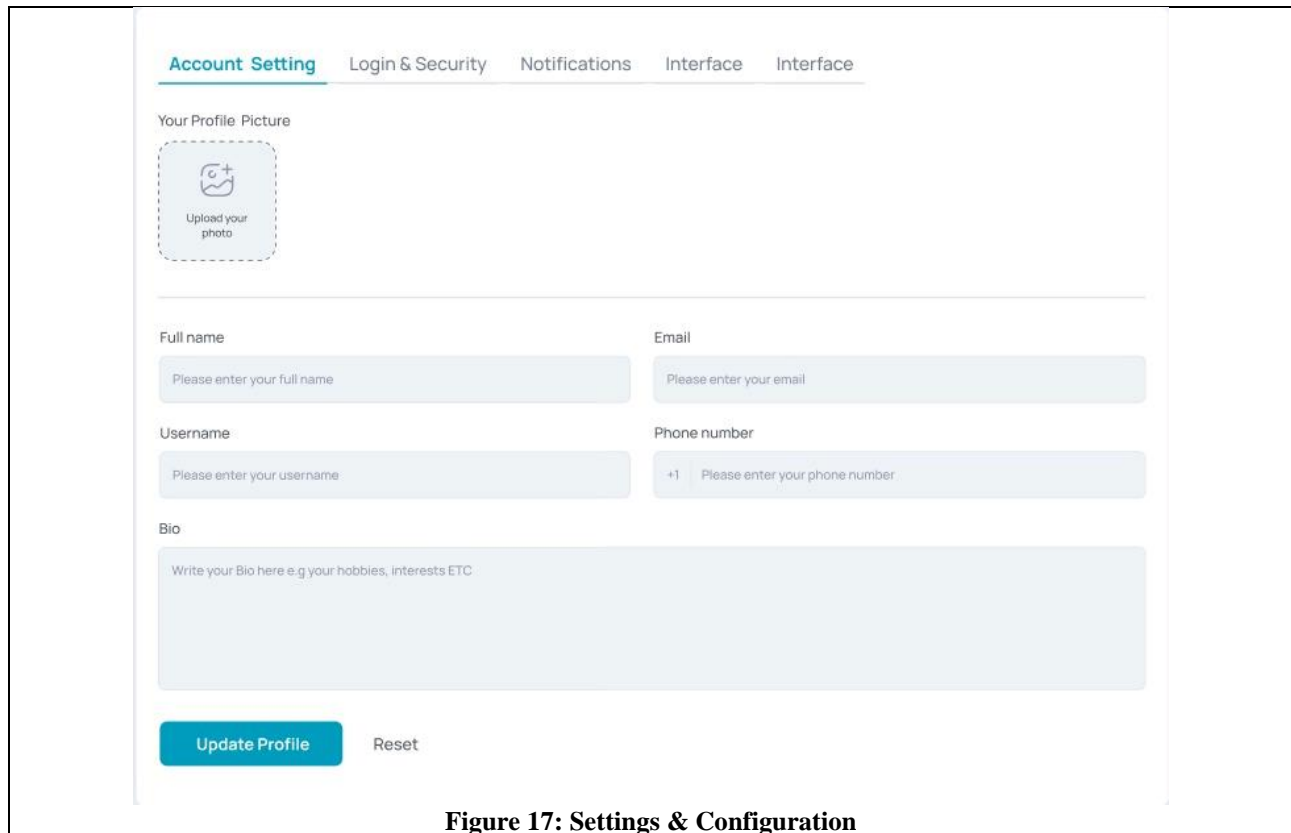
6.2.3 VPN detection and Analysis Section



6.2.4 VPN Management Section



6.2.5 Settings & Configuration:



The screenshot displays the 'Account Setting' page, which is part of a settings menu. The menu includes 'Account Setting', 'Login & Security', 'Notifications', and two 'Interface' entries. The 'Account Setting' section contains a profile picture upload area with a dashed border and a camera icon, labeled 'Your Profile Picture' and 'Upload your photo'. Below this are four input fields: 'Full name' (placeholder: 'Please enter your full name'), 'Email' (placeholder: 'Please enter your email'), 'Username' (placeholder: 'Please enter your username'), and 'Phone number' (placeholder: '+1 Please enter your phone number'). A large text area for 'Bio' is also present, with a placeholder 'Write your Bio here e.g your hobbies, interests ETC'. At the bottom, there are two buttons: 'Update Profile' (in teal) and 'Reset'.

Figure 17: Settings & Configuration

6.2.6 Reports & Logs:

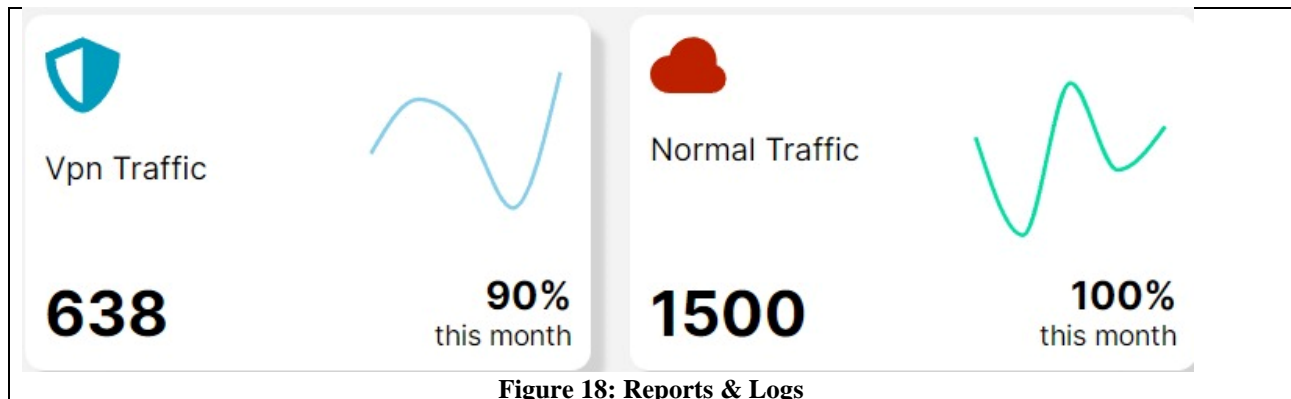


Figure 18: Reports & Logs

6.3 Screen Objects and Actions

6.3.1 Navigation Bar

Objects: Account Settings, Historical Data, Alerts, Traffic Analysis, and Dashboard menu items.

Actions: The user can access the corresponding sections by clicking on these items.

6.3.2 Real-time Traffic Graph (Dashboard)

Objects: VPN traffic highlight, line chart or heat map.

Actions: Specific traffic information is displayed when you hover over an element. The VPN Analysis screen is displayed to the user when clicking on a VPN highlight.

6.3.3 VPN Traffic List (Analysis Screen)

Objects: List of VPN connections, 'Details' and 'Block' buttons.

Actions: Selecting 'Details' displays a modal window containing further details. Verifying VPN blocking is prompted when you click "Block."

6.3.4 Alerts List

Objects: A list of alerts with a synopsis for each.

Actions: Tapping an alert enlarges it to provide further information or directs the user to the relevant traffic analysis.

6.3.5 Historical Data Charts

Objects: A range of filters and charts.

Actions: By using filters, the charts are updated to show the necessary data.

6.3.6 Account Management Forms

Objects: Notification settings, password, and user information input boxes.

Actions: The user's account settings are updated upon completing and submitting the form.

6.3.7 Breakdown into smaller parts

- **Buttons:** Save settings, create reports, block/unblock VPN, start/stop monitoring. These can be clicked to carry out the corresponding activities.
- **Fields:** Input fields for settings parameters, list filters, and login. Users have the option to choose or enter the required data.
- **Tabs:** Navigate through the various areas, such as Reports, Dashboard, VPN Management, and Settings.
- **Alerts and Notifications:** Real-time feedback on system status, VPN detections, or faults through pop-ups or dedicated alert sections.
- **Graphs and Charts:** Data is visually represented to make traffic patterns and VPN activity easier to comprehend.
- **Tables/List Views:** Provide choices for sorting and other actions, as well as full information on identified VPNs, logs, and user activity.