SECURENET VPN APPLICATION

LIM XIAO ZE

UNIVERSITI TUNKU ABDUL RAHMAN

SECURENET VPN APPLICATION

LIM XIAO ZE

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Science (Honours) Software Engineering

Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

October 2024

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	[illegible]
Name	:	LIM XIAO ZE
ID No	:	1901159
Date	:	1/10/2024

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"SECURENET VPN APPLICATION"** was prepared by **LIM XIAO ZE** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Science Software Engineering with Honours at University Tunku Abdul Rahman.

Approved by,

Signature	:	[illegible]
Supervisor	:	Mohammad Babrdel Bonab
Date	:	1/10/2024
Signature	:	
Co-Supervisor	:	
Date	:	

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

ABSTRACT

In an era of growing concerns over online privacy and security, current VPN solutions often fail to meet user expectations due to unreliable protocols, weak authentication mechanisms, and overly complex interfaces. This project, SecureNet VPN, addresses these issues by focusing on three core objectives: the implementation of robust encryption standards and reliable open-source protocols like OpenVPN, the integration of two-factor authentication (2FA) to enhance access control, and the design of a user-friendly interface to improve usability.By adopting industry-standard encryption (AES-256) and a no-log policy, SecureNet VPN ensures data security and privacy. It mitigates the risks posed by inadequate protocols and weak authentication practices, while its intuitive interface simplifies the VPN experience for users, regardless of technical expertise. SecureNet VPN aims to deliver a dependable, secure, and easy-to-use virtual private network solution that effectively addresses the challenges facing current VPN services, thereby enhancing both user privacy and online security.

Chapter 1
Introduction

1.1 Introduction

In this age of increasing apprehension regarding online privacy and data security, SecureNet VPN emerges as a dependable virtual private network solution that emanates confidence. Given the presence of vulnerabilities and threats, our primary objective is to provide users with a seamless, user-friendly connectivity experience while simultaneously establishing an impregnable barrier against malicious intent and intrusive eyes.

SecureNet VPN signifies that user privacy is protected and digital defenses have been fortified; it is not merely an application. By integrating advanced authentication mechanisms, cutting-edge encryption protocols, and an intuitive user interface, our project endeavors to fundamentally transform VPN technology by strict adherence to rigorous standards.

With a triple objective, the initiative strives to achieve its ultimate purpose. Before anything else, we intend to integrate established VPN protocols, such as OpenVPN, so that data security can be ensured, and data transmission can be optimized when our product is in use. Subsequently, however, effort has been devoted to developing a user authentication system that is fortified through the implementation of robust authorization mechanisms and multiple authentication methods, thereby restricting access to our VPN service exclusively to authorized users. In conclusion, our objective is to streamline the server selection process by implementing an intuitive interface, enable effortless monitoring of connection status, and furnish users with customization options.

The objective of the project is to create a VPN application which emphasize on secure and user-friendly. SecureNet VPN endeavors to significantly contribute to the fulfilment of this expanding need by guaranteeing secure online communications and robust data protection via unwavering commitment and groundbreaking advancements.

1.2 Problem Background

In this age of advanced technology, internet usage had expanded across various sectors. As the internet develops, the issue of cybercrime and the importance of privacy is also gradually magnifying. The concern of this causes growing need for secure communication methods especially for business, organization and for those who had requirements for secure communications method to secure their data and privacy(NIKI, 2023). Virtual Private Networks(VPN) had become the best solution and crucial for safeguarding online activities and criminal activities. Virtual Private Networks core function is encrypt user data during transmission of data, which able to provide protection against malicious hackers, privacy invaders and cyber criminals.

Nowadays most of the VPN application has multiple issue which causing insecure communications, data leakage and interception. First issue comes from the unreliable protocol issue, many VPN application does not implement and integrated with reliable protocol and encryption standard(Williams, 2020).. Encryption of data as the core functionalities in VPN, selecting and implementation of encryption standard and reliable protocol significantly impact the effectiveness of VPN solution regarding the security issue. Reliable encryption algorithm and protocol is crucial for establishing a secure connection and maintaining communication stability.

Additionally, weak authentication mechanisms or lack of adequate access control measure will also affect the effectiveness of VPN solution which may cause unauthorized access to the services causing result such as data breaching and jeopardizing the integrity and confidentiality of transmitted data. Besides, complexity of user interface of VPN application is another issue led to ineffective of using VPN solution. Many applications had provided many features that need user to configure themselves, but most of the user did not have such technical skills to support them to configure the VPN application and causing frustration. As a result, user will select to using default setting ,which many no provide optimal security and undermine the effectiveness of VPN solution. This poses a significant risk as improperly configured VPN may leave users vulnerable to security breaches and privacy violations(Sablah, 2024).

In conclusion, implement and integrate reliable VPN protocol and encryption standards able to provide secure communication channels to user to secure their data and privacy, while access controls able to protect user while using the VPN services and high usability and intuitive user interface can help user to achieve their task effectively.

1.3 Problem Statement

Based on the problem background, we will continue discuss three major problem that has the most influence on VPN solutions.

1.3.1 Unreliable Protocol and Encryption Standards

Some VPN solution had issue regarding of lacking implementation of reliable protocol or using closed source protocol and encryption standards. This issue will compromise the security and reliability of the VPN service and causing user' data may be intercept and may exploit by malicious parties for malicious activities and this also undermines the fundamental purpose of VPN service to secure online activities. These issue are able to solve in a very effective way by implementing those reliable protocol such as OpenVPN, WireGuard, IPsec,L2TP which are consider as famous, reliable and transparent due to its open-source attribute.

1.3.2 Lack of Robust Authentication and Access Control Measures

Existing VPN apps tend to overlook the importance of a robust user authentication and access control of VPN services. This will expose services and user's privacy and information to security risks. For example, unauthorized users may exploit this flaw gain network access which will able to intercept and exploit user's data during transmission. Based on this, we can say that inadequate control increases the risk of data exposure and jeopardizing the integrity and confidentiality of transmitted data. Implementing strong authentication is essential for VPN services such as implementing multifactor authentication and effective access control measure to control user access and ensure only authenticated user to use the service prevent malicious event happening.

1.3.3 Complexity of User Interface

Many VPN applications had problem that had design overly complex user interfaces, this cause usability issue for the application which will causing user hard to understand how to properly use the VPN application and difficult for users to navigate and utilize all the features effectively. Besides that, complexity of VPN application interfaces can hinder user adoption rates which users may be discouraged from using the service due to its low usability and they may faced security and privacy issue during online activity. There is a necessary for VPN applications had a simplify user interfaces to improve the usability and accessibility of for all users regardless of their expertise, by emphasizing usability of user interface we able to leading higher user satisfaction and usage to secure user's online activities.

In conclusion, the problems associated with VPN services including lacking implementation of reliable protocol and encryption standards, absent of robust user authentication and access control measures and the complexity of user interface, these challenges had significantly affected the effectiveness of VPN services.

1.4 Objective

Based on the problem statement, the 3 objectives of this project will define and discussed.

1.4.1 Implement and Integration of Robust Encryption Standards and Reliable Protocols

In this project, we aim to implement a robust encryption standard as industry standard such as AES and RSA for data integrity and security and integrate open-source VPN protocol such as OpenVPN into SecureNet VPN to provide a reliable network establishment and connection to users. By implementing robust encryption and reliable protocol, we able to mitigate the concern of privacy issue and criminal activities and also safeguarding user's information during transmission.

1.4.2 Enhance User Authentication Mechanisms By Integrated With Two-Factor-Authentication.

In this project, we will implement user authentication mechanisms for access our VPN service. This action able to increase awareness of user about the importance of secure authentication practices. By employing stronger password policies and utilizing multi-factor authentication techniques, our application will integrated with an additional layers of security for user authentication and authorization. With an additional layer of security, SecureNet VPN application able to protecting sensitive user data from unauthorized access, data breaches, mitigating risk of account hijacking and identity theft which reduce the likelihood of unauthorized access attempts and improved protection of user credentials and personal information,.

1.4.3 Optimize User Interface Design By Providing Friendly User Interface

We aim to provide a user friendly interface for better usability. We plan to optimizing interface layout for simplicity and ease of use which able to enhancing user experience while using our application. For example, we will implementing responsive design, streamlining work flows and minimizing user friction. Additional feature for improving user usability included providing clear and concise instructions and prompts while using the application. By prioritizing and emphasizing in usability of user interface, the application can improved efficiency and effectiveness for user to complete their tasks and reduce the cost of support and training due to intuitive design, last but not least able to increase user base for secure and safeguarding their privacy and information for online activities.

With these goals, SecureNet VPN application strives to provide a secure, user-friendly and dependable virtual private networking experience that able to provide user necessary VPN functions while solving the addressing key problems currently facing the industry.

1.5 System overview

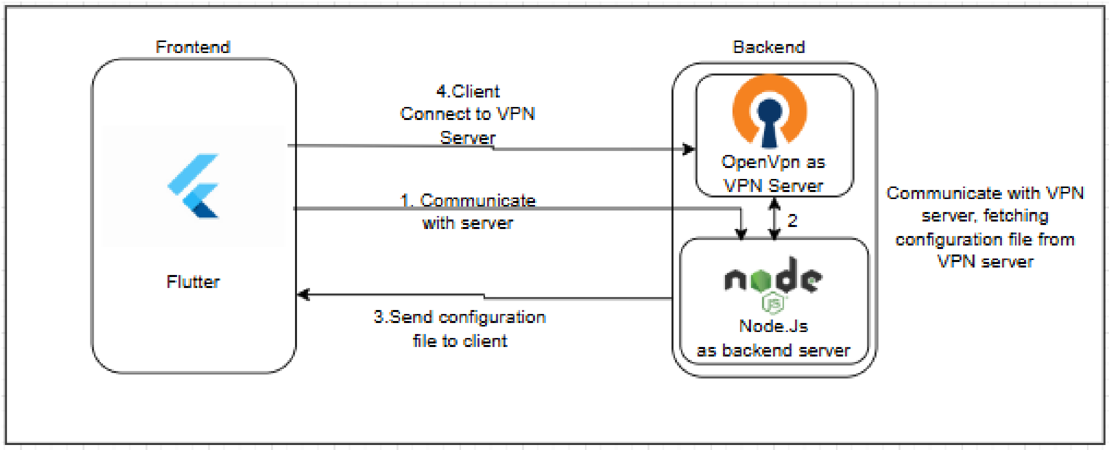

Frontend
Flutter
Backend
OpenVpn as VPN Server
Node.Js as backend server
4.Client Connect to VPN Server
1. Communicate with server
3.Send configuration file to client
2
Communicate with VPN server, fetching configuration file from VPN server

Figure 1.1 System Overview

SecureNet VPN aim to emphasize on 3 objectives. First is to implement and integrated reliable VPN protocol and encryption standard to the application. SecureNet VPN will integrate OpenVPN, an open-source VPN protocol which are famous for reliable and flexibility to provide the VPN services. Second, SecureNet VPN will implement user access-control measures such as multi-factor authentication (MAF) or two factor authentication(2FA) to control only authenticate and authorize user able to access to the services to increase the level of security of the application.

The last factor is provided a intuitive user interface which able to increase the efficient of the application for user. Besides, SecureNet VPN also will provide multiple features such as kill switch function which will disconnect the internet access if the VPN connection lost accidentally and the connection of the VPN will be setup to become one-to-one mapping. In SecureNet VPN, each user will have own unique configuration file use to establish connection to the server, one-to-one mapping will only allow one connection to the VPN server at a time in the case of many users holding the same configuration file due to leakage of configuration file. This able to ensure user does not use any file not belongs to them and increase the security level of the whole system.

SecureNet VPN application will have a frontend (android)which had a user-friendly user interface for letting user to initiate to access to our services. Before using our service, user need to login or register in our databased before proceed to use the services, this practice aim for emphasizing the importance of user authentication and access control. After login to our application, user able to initiate a request which request to connect our VPN service to our backend server.

Our backend server will be responsible for user authentication and access control to protect and allow only trusted user able to use the services. After all required validation done, backend server will communicate with our VPN server and fetching necessary file from it and send back to authenticated user. Lastly, user able to connect to our VPN service, which had implemented robust encryption algorithms and reliable protocol to safeguarding user's online activities by encrypt user's data during transmission across the network.

Our system strives to resolving current VPN solutions faced issues and provide better solution in VPN services by emphasizing, user-friendly, user authentication and access control practices and implementation of robust encryption standards and reliable protocols to users to safeguarding them.

1.6 Project Scope

The project scope encompasses the development of a VPN application with a focus on security, user authentication, and a user-friendly interface. The application will support standard VPN protocols and provide a reliable and efficient connection for users.

1.6.1 Tools

This project involved various tools such as framework, protocols, encryption algorithms and virtual machines or device for testing.

Development IDEs: Virtual Studio Code, Android Studio

Framework: Node.js for backend, React-Native for frontend

Database: MySQL

Programming language: JavaScript, Linux command

Protocols: OpenVPN

Operating system: Android, Linux

Version Control Tools: Git

Testing Device: Android, Window PC

Others: Virtual Machines

1.6.2 Platform Support

SecureNet VPN will support for Android platforms only by providing a consistent and intuitive user interface.

1.6.3 Feature Covered

This project deliverables included a fully functional SecureNet VPN application for Android platforms. The applications will integrated with secure VPN Protocols (OpenVPN) with optimization and implementation of robust user authentication and authorization system. Besides, it also provides a consistent and intuitive user interface and lastly documentation for this project will covering the development process, security measures and user guidelines.

Expected Deliverables:

1. Fully functional SecureNet VPN application for Android
2. Integration of secure VPN protocols (OpenVPN) with optimization.
3. Implementation of a robust user authentication and authorization system.
4. User friendly and consistent user interface.
5. Documentation covering development processes, security measures, and user guidelines.
6. Enhanced Security Features such as DNS leak protection, IPV6 support and split tunneling to prevent data leakage.
7. Selection of multiple country of the VPN services.

1.6.4 Features Not Covered

1. Advanced Network Configuration

As the application prioritize on user friendly, advanced customizing features such as divided tunnel, port forward could make the application become more complex and confusing. Therefore, we will not support for these advanced setting to keep applications simple and user-friendly.

2. Additional Authentication Methods

Additional authentication methods bring complexity when there are several authentication methods. In this project we will have simple management of this includes strong username/password authentication system with token-based management focusing on simplicity and security.

3. Advanced Protocols Options

In this project, we will not support selection of protocols which may cause more complexity and reduce usability of the application. This project focus more on simplicity, we will only support for one protocol for simplicity while using industry-standard protocols such as OpenVPN for a robust security measure without adding complexity

1.7 Defining "Reliable" in the Security Context

1.7.1 Introduction

In the implementation of SecureNet VPN, the term "reliable" needs to be clearly defined within the security framework to ensure that the encryption standards and VPN protocols used provide comprehensive protection against a range of security threats. Reliability, in this context, extends beyond basic functionality, focusing on the consistent and secure performance of the cryptographic systems and VPN protocols. This section seeks to narrow down the definition of "reliable" specifically from a security perspective, emphasizing data protection and resilience against cyber threats.

1.7.2 Definition of "Reliable" in Security

From a security standpoint, reliability refers to the system's ability to consistently maintain the confidentiality, integrity, and availability of data under various operational conditions. In this project, where AES (Advanced Encryption Standard) and RSA (Rivest-Shamir-Adleman) encryption are used, reliability focuses on the following aspects:

1. **Confidentiality**: The encryption algorithms employed (AES and RSA) must reliably prevent unauthorized access to sensitive information. A reliable encryption standard ensures that data is kept confidential during transmission, even if intercepted, by making it computationally infeasible for attackers to decrypt the data without the correct keys.
2. **Integrity**: Reliable encryption ensures that the data remains unchanged during transit. Any tampering or alteration during transmission should be detectable by the recipient through cryptographic hash functions or digital signatures. In this project, reliable encryption guarantees that the information received is exactly what was sent.
3. **Authentication**: RSA, being a public-key cryptosystem, provides a mechanism for verifying the authenticity of the sender. Reliability in this context ensures that parties communicating over SecureNet VPN can authenticate each other’s identities, minimizing the risk of impersonation attacks (e.g., man-in-the-middle attacks).
4. **Non-Repudiation**: A reliable system also provides non-repudiation, ensuring that once a transaction or communication is made, the sender cannot deny having sent the message. RSA-based digital signatures play a crucial role in this, ensuring that the identity of the sender is verifiable and binding.
5. **Resistance to Cryptographic Attacks**: Reliability includes the system's resilience against known cryptographic attacks, such as brute force, side-channel, and timing attacks. AES and RSA must be implemented securely,

using strong keys (e.g., 256-bit for AES and 2048-bit for RSA), to provide long-term security against evolving threats.

6. **Secure Key Exchange and Management**: A reliable encryption system ensures the safe generation, exchange, and storage of cryptographic keys. This is particularly crucial in the use of RSA, where secure key management ensures that private keys are kept confidential and are not compromised during transmission.

In summary, reliability in the context of security for this project encompasses the encryption system's ability to maintain confidentiality, integrity, and authentication consistently, alongside the VPN protocol's capacity to securely transmit data over the network without leaks, attacks, or interruptions. By narrowing the definition of "reliable" to focus specifically on security, we emphasize the importance of robust encryption standards (AES and RSA) and the secure, stable operation of the VPN protocol (OpenVPN) to protect user data against both passive and active threats throughout its transmission.

This definition helps guide the implementation of a system that not only functions correctly but also consistently meets high-security standards in a hostile and evolving cyber environment.

Chapter 2

Literature Review

2.1 Introduction to VPN

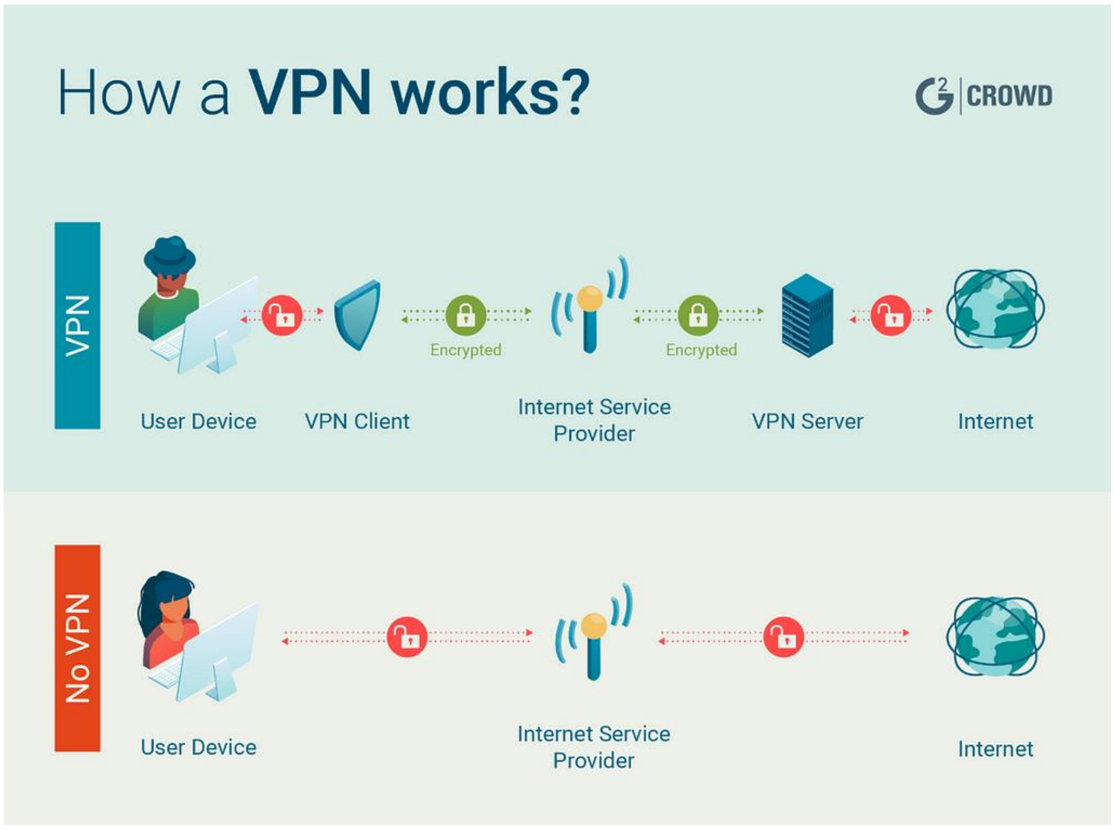


Figure 2.1 VPN Mechanism (Gershwin, 2019)

Virtual Private Network (VPN) is a technology that creates a secure and encrypted connection over a less secure network such as internet. VPN primarily used to provide secure access to resources and data on internet remotely and ensure confidentiality, data integrity and authentication (Cisco, 2024).

In common scenario of browsing or getting access to resources online, user request will be route to the specific server by its internet service provider. In this screnario,user's request such as resources, data as well as IP address are been capture by internet service provider which means user's activity are all under monitoring(Microsoft, 2024). Beside of service provider, during the data transmission, user’s data packet also able to be capture and exploit by malicious parties even though nowadays our internet traffic had implemented multiple security measure during data transmission such as HTTPs but it still far enough to have a completely secure communication channel.

Virtual Private Network solution able to provide user a more secure environment to access and communicate to online resources due to its mechanism. While user connecting to the VPN server, user’s request will be forward to the VPN server and the request will be made by the VPN server. After getting the result VPN server will send back to the user which will encrypt all the traffic end-to-end to user device. In this scenario, internet service providers only able to detect and monitor user’s keep associate with VPN server but did not know what is the request afterward since request will be done by VPN server, client just need to forward to VPN server(OPENVPN, 2024). Besides, due to its encryption of traffic, it is hard for malicious parties to decrypt the traffic even if they able to intercept it.

Virtual Private Network (VPN), on the other hand, ensures that data transmission over the internet is secure and encrypted to form a private network out of a public internet source. It provide function such as end-to-end encryption to all internet traffic, establish secure connections through robust authentication mechanisms and encryption protocol, which providing a more comprehensive layer of security, ensuring all the data between VPN Server remains protected from interception and tampering. Beside of secure user’s privacy and information, VPN able to spoof user’s geographic location which enhancing anonymity and privacy online by hiding user’s real IP address replace with the IP address of VPN server(Microsoft, 2024).

Therefore, VPNs are important in protecting privacy online, boosting security level and unblocking internet bans. Whether it is about keeping confidential data safe from intruders' eyes, getting into locally based information, or ensuring secure remote entry to corporate networks; VPN have become fundamental for individuals and companies grappling with the intricacies of contemporary digital era.

2.2 Type of VPN

Several important type of VPN knowledge need to explain.3 type of VPN will be discussed which is Remote Access VPN, Site to Site VPN and Client to Client VPN.

2.2.1 Remote Access VPN

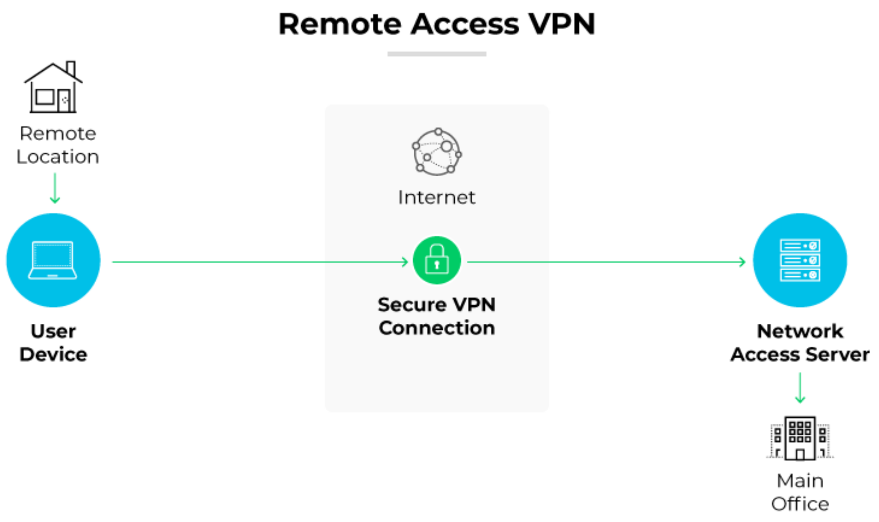


Figure 2.2 Remote Access VPN (Shweta, 2024)

Remote access VPN service allows individual users to securely connect to a private network securely over the network. It allows individuals user or devices that outside the corporate network to access to resources, applications, and data as they currently connected to the network locally. For remote access VPN, the establishment of connection typically initiated from software client on their device and users must typically authenticate themselves before gaining access to VPN which ensures that only authorized users can connect and access to corporate resources while maintaining confidentiality, integrity, and availability of data. Inside the private network, the data is decrypted and processed as usual which ensuring user can interact with network resources as if they were directly connected. This mechanism allows safe and remote access to resources while maintaining the security and integrity of data(FORTINET, 2024).

2.2.2 Site to Site VPN

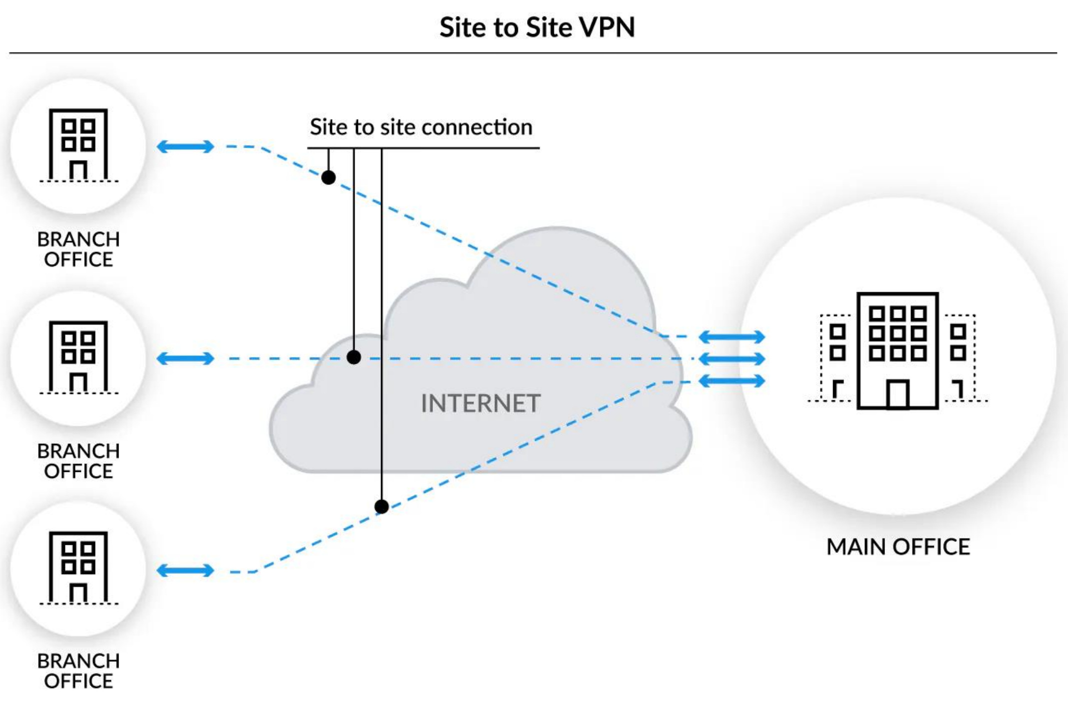


Figure 2.3 Site to Site VPN (Paloalto, 2024)

Site to site VPN also known as router-to-router VPN, is a type of VPN connection that able connects multiple networks or sites securely over network. Site to site VPN establish secure connection between entire network or subnets and frequently used by companies which have multiple offices in different geographical location that its employee needs to access to and use the corporate network. With site-to-site VPN, company can securely connect to its corporate network with remote offices to share resources and communicate with them as a single network. At each site, a VPN gateway or router is used to establish and manage the VPN connection which serves as the endpoint for VPN tunnel and responsible for authenticating each other for ensuring the legitimacy of endpoint and encrypting and decrypting data during transmission between the sites. By enforcing security policies such as traffic filtering and access control, the VPN gateway able to safeguard the network from unauthorized access or malicious activities. Additionally, its high availability and redundancy attribute ensure that uninterrupted connectivity which is vital for maintaining seamless operation across distributed networks(Paloato, 2024).

2.2.3 Client to Client VPN

Figure 2.4 Client to Client VPN(FORTINET, 2024)

Client to client VPN also known as peer-to-peer VPN, where individual devices or clients connect directly to each other over secure encrypted tunnel. In this scenario, each client act as a VPN client and VPN server for allowing direct communication between them without the need for a central VPN server infrastructure. Through mutual authentication and encryption mechanism, client device will create aa secure tunnel for communication which ensuring the confidentiality and integrity of data exchange between them. This configuration enables direct peer-to-peer communication and able to have feature such as peer to peer file sharing, collaborative work and remote desktop access. Client to client VPN offer flexibility and decentralized connectivity and integrated with robust security measures such as strong authentication and encryption mechanisms. Overall, client to client VPN able to provide versatile solution for establishing secure communications channel between individual user's device where centralized server is not practical or desire(FORTINET, 2024).

2.3 VPN Protocol

In this part, we will have a discussion on different types of VPN protocol by comparing each strength and weaknesses and from this we can saw the reason behind choosing OpenVPN protocol for this project due to its strengths and acceptable weakness

2.3.1 OpenVPN

Figure2.5 OpenVPN (OpenVPN, 2018)

OpenVPN Is an open-source virtual private network protocol that establish secure communication over the network. It employs SSL/TSL protocols for encryption, authentication and key exchange which ensuring the confidentiality and integrity of data transmitted between devices or networks. OpenVPN is popular choice for secure VPN communications due to its high security level features, flexibility and extensive community support(OpenVPN, 2022).

2.3.1.1 OpenVPN Strengths

- **High security level**

  OpenVPN offer robust security features such as encryption, authentication and key management which ensuring data privacy and protection against unauthorized access.

- **Flexibility**

OpenVPN support various authentication methods and encryption algorithms which allowing to customization based on specific security requirements.

- **Community Support**

OpenVPN as an open-source project, it benefits from dedicated community of developers and users which contributing to its ongoing development, support and continuous improvement.

2.3.1.2 OpenVPN Weaknesses

- **Potential Vulnerabilities**

OpenVPN may have security vulnerabilities if not properly maintained or updated like any other software

- **Scalability**

OpenVPN having challenges in scaling to support large numbers of simultaneous connections or complex network configurations.

2.3.2 Internet Protocol Security (IPsec)

Internet Protocol Security (IPsec) is a protocol used to secure internet protocol communications by authenticating and encrypting IP packet of a communication session(Cloudflare, 2024).

2.3.2.1 IPsec Strengths

- **Security**

IPsec provides strong security features such as encryption, authentication and integrity of data checking which ensures that data confidential and tamper-proof during transmission

- **Flexibility**

IPsec support various encryption algorithms, authentication method and key exchange protocol which allowing customization based on specific security requirements

- **Transparent Integration**

  IPsec operate at network layer of OSI model, it able to provide transparent security for all IP-based applications and protocols without requiring modifications to individual applications

2.3.2.2 IPsec Weaknesses (tailscale, 2024)

- **CPU Overhead**

  IPsec is well known for high CPU usage. It requires higher processing power to encrypt and decrypt all the data pass through the server. It may cause overhead when small data packet pass through server.

- **Higher Complexity**

  IPsec is more complex due to it included many features and options which will lead in probability of weakness in protocol

- **Limited NAT Traversal Support**

  IPsec may have difficulties when traversing network address translation (NAT) device which might require additional extra arrangement of mechanism to address the challenges.

2.3.3 L2TP/IPsec (Internet Key Exchange Version 2/IP Security)

L2TP/IPsec is combination of L2TP with IPsec which aim to provides secure communication over internet. L2TP is tunneling protocol used to create virtual private network over internet but it does not provide encryption function and relies on other protocol for data confidentiality and integrity. It creates tunnel between client and server, encapsulating data packets and allow packet to transmit over internet securely(Zola, 2021).

2.3.3.1 L2TP/IPsec Strengths

- **Widespread Support**

  L2TP/IPsec is widely supported by most operating systems, including Windows, macOS, Linux, iOS, and Android.

- **Strong Security**

  The combination of L2TP for tunneling and IPsec for encryption, authentication, and integrity protection, make it able to offers robust security features while IPsec provides strong encryption algorithms and authentication methods to ensure the confidentiality and integrity of data transmitted over the VPN connection.

- **Ease of Setup**

  L2TP/IPsec is easy to set up and configure compared to other VPN protocols like OpenVPN. Besides, many operating systems and VPN clients provide built-in support for L2TP/IPsec which simplifying the configuration process for users.

2.3.3.2 L2TP/IPsec Weaknesses (Dmitry, 2023)

- **Limited Anonymity**

  L2TP/IPsec does not provide strong anonymity features compared to some other VPN protocols. Due to it relies on fixed protocols and ports and it may be easier for network administrators or ISPs to identify and block L2TP/IPsec traffic.

2.3.4 PPTP (Point-to-Point Tunneling Protocol)

PPTP is one of the earliest virtual private network protocols developed and widely used to establish secure connections over internet. PPTP operates at the layer 2 of OSI model and provide encapsulation and encryption mechanisms to secure data transmission between endpoints.

2.3.4.1 PPTP Strengths

- **Ease of Setup**

  PPTP is easy to set up and configure which making it accessible to users with limited technical expertise and it is supported by most operating systems and devices making it widely available.

- **Compatibility**

  PPTP is supported by a wide range of devices and operating systems, including Windows, macOS, Linux, iOS, and Android.

- **Performance**

  PPTP is known for its relatively fast connection speeds and low latency compared to some other VPN protocols.

2.3.4.2 PPTP Weaknesses

- **Security Concerns**

  PPTP has known security vulnerabilities and weaknesses which making it less secure compared to more modern VPN protocols. Besides ,it uses relatively weak encryption algorithms and its security mechanisms have been compromised in the past.

- **Lack of Strong Authentication**

PPTP relies on MS-CHAP v2 (Microsoft Challenge Handshake Authentication Protocol version 2) for authentication which has been found to be susceptible to certain attacks, including brute force attacks

- **Limited Support for Modern Features**

  PPTP lacks support for advanced features such as strong encryption, multi-factor authentication and robust key exchange mechanisms. Its outdated design and limited security features may not meet current requirements

2.3.5 SSTP (Secure Socket Tunneling Protocol)

SSTP is a virtual private network developed by Microsoft. It designed to establish secure and encrypted connections between client and VPN server over internet. SSTP operate at later 4 of OSI model using SSL/TLS encryption to secure data during transmission. Point-to-Point Tunneling Protocol (PPTP) connections does not apply SSL/TLS due to this SSTP was introduced to improve the security of data transfers and to avoid limitations of PPTP(proofpoint, 2024).

2.3.5.1 SSTP Strengths

- **Strong Encryption**

  SSTP utilizes SSL/TLS encryption to secure data transmission between the client and the server which ensuring the confidentiality, integrity, and authenticity of the VPN connection.

- **Firewall-Friendly**

  SSTP operates on TCP port 443 which is the same port used for secure HTTPS traffic. This makes SSTP traffic indistinguishable from regular HTTPS traffic and allowing it to bypass firewalls and network restrictions that may block other VPN protocols.

2.3.5.2 SSTP Weaknesses (Perimeter 81, 2023)

- **Limited Platform Support**

  SSTP is primarily supported on Windows-based platforms such as Windows Server and desktop editions, native support may be limited or absent on non-Windows platforms.

2.3.6 WireGuard

Figure 2.6 WireGuard (BlessingGeek, 2019)

WireGuard is a modern lightweight VPN protocol designed aim to be easy to implement, highly efficient and secure. It improves upon existing VPN protocol by simplifying codebase, reducing attack surface, and providing better performance(Donenfeld, 2024).

2.3.6.1 WireGuard Strengths

- **Simplicity and Efficiency**

  WireGuard's minimalistic design with small codebase which making it easier to understand implement and audit. Its lightweight nature results in faster connection speeds and lower latency compared to other protocol.

- **Security**

  WireGuard employed modern cryptographic techniques such as Curve25519 for key exchange and ChaCha20Poly1305 for encryption and authentication which ensuring strong security.

- **Reliability**

  WireGuard has undergone extensive testing and development which include security audits by third-party researchers and has demonstrated stability and reliability in ral-work deployments.

2.3.6.2 WiredGuard Weaknesses(Jack, 2023)

- **Maturity**

WireGuard as a relatively new protocol there could be unforeseen challenges or issues that arise as it continues to mature.

2.3.7 IKEv2/IPsec (Internet Key Exchange version2/IP security)

IKEv2/IPsec is combination of two protocol that use to create secure virtual private network connections over the network. IKEv2 is a protocol use for establishing secure connection between 2 devices in VPN setup which primary function is to facilitate the negotiation of cryptographic keys and parameters that necessary to establish encrypted tunnel between both client and VPN server. IKEv2 are responsible for authentication, managing and generating key encryption key securely and managing parameters of security association such as encryption algorithms, integrity protection and lifetime. By combining both protocols, this solution able to provide strong security features, efficient connection management and support for mobility(Miklos, 2024).

2.3.7.1 IKEv2/IPsec Strengths

- **High Security**

  IKEv2/IPsec able to offer robust security features such as encryption, authentication, and key management which ensuring the confidentiality, integrity of data during transmission over VPN connection.

- **Resilience and Mobility**

IKEv2/IPsec able provides resilience to network changes and disruptions. IKEv2 able to quickly re-establish connections along with IPsec support for mobility making it suitable for uses who frequently switch between networks such as mobile devices.

- **Stability**

  IKEv2/IPsec can provide a stable connection and allow user to switch its internet connections without losing protection.

2.3.7.2 IKEv2/IPsec Weaknesses

- **Compatibility Issues**

  IKEv2/IPsec is supported on most modern platforms but have compatibility issues with older devices or operating systems which may leading to connectivity problems in certain scenarios.

- **Potential Performance Impact**

  The encryption and decryption processes involved in IPsec can introduce overhead and impact network performance which is critical on low-powered devices or high-bandwidth connections.

2.4 Encryption Algorithm

We will discuss variety of cryptographic techniques employed to secure data transmission over networks. By analyze and considering factors like security level, computational efficiency, and compatibility with different network architectures. Through this exploration, we aim to gain a comprehensive understanding of encryption strategies in VPN environments and their implications for ensuring data confidentiality, integrity, and authenticity.

2.4.1 AES (Advanced Encryption Standard)

AES is asymmetric encryption algorithm used to protect sensitive data which it using the same key for both encryption and decryption process. AES supports key sizes of 128 ,192 and 256 bits while longer key sized provide stronger encryption but may require more computational resources. AES able to performs multiple rounds of substitution, permutation and mixing operations on the data with the number of rounds determined by the key size. AES has undergone extensive scrutiny by cryptographers and is trusted by governments and organizations globally. Its computational efficiency ensures swift encryption and decryption processes which making it suitable for diverse applications such as secure communications, data storage and protecting sensitive information(Awati et al., 2024).

2.4.2 ChaCha20

ChaCha20 is a symmetric encryption algorithm operates as a stream cipher which it will generate a continuous stream of pseudorandom data and then is combined with a plaintext to produce ciphertext. ChaCha20 is optimized for performance across a wide range of platforms including both software and hardware implementations due to its ability of efficient and fast encryption and decryption operations making it suitable for use in resource-constrained environments such as mobile devices and embedded systems. Besides, it able to provide strong security guarantees and resistant to known cryptanalytic attacks and offers high level of security when used with appropriate key sizes and often used as encryption algorithm in various protocol and application including VPN(Nagaraj, 2023).

2.4.3 3DES (Triple Data Encryption Standard)

3DES is a symmetric encryption algorithm that applies the Data Encryption Standard (DES) cipher algorithm three times to each block of data which more secure compare to the original DES algorithm.3DES offers stronger security that DES but it suffers slower performance due to need of multiple encryption operation and may result not efficient as modern encryption algorithm such as AES or ChaCha20 and it had become vulnerable to brute-force attacks due to its relative short key length(Cobb, 2023).

2.5 Authentication Algorithm

We will look into various authentication algorithms in the context of VPN. Explore how these algorithms facilitate the exchange of credentials, certificates, or cryptographic keys between the VPN client and server, enabling mutual authentication and protecting user information safety and integrity against unauthorized access or data interception.

2.5.1 HMAC-SHA256

Hash-based Message Authentication Code with Secure Hash Algorithm 256 (HMAC-SHA256) is a cryptographic algorithm use for generating messages authentication codes to ensure the integrity and authenticity of data that transmitted over network. It combines the strength of SHA-256 hash function with security properties of HMAC (Keyed-Hash Message Authentication Code) to produce a fixed size hash value that can be used to verify, and integrity data transmitted(Okta, 2023).

2.5.2 HMAC-SHA512

Same with HMAC-SHA256, the difference is on the hash value which SHA512 can produces 512bits has value compared to SHA256 which is 256bits.SHA512 offers larger hash output size compared to SHA256 which providing higher security and resistance against brute-force attacks. This algorithm is commonly used when higher level of security is required.

2.5.3 EAP (Extensible Authentication Protocol)

Extensible Authentication Protocol (EAP) is an authentication framework used in network communication protocols. It provides a flexible and extensible mechanism for supporting various authentication methods authentication methods such as passwords, digital certificates, token-based authentication, one-time passwords, biometrics, and more. These methods can be integrated into the EAP framework to provide authentication flexibility based on the specific security requirements of the network(Webster & Contributor, 2021).

2.6 Key Exchange Protocols

We will look into various key exchange protocols which show the different mechanisms used to securely establish cryptographic key between communicating parties and examining their strength and weakness in different scenarios.

2.6.1 RSA(Rivest-Shamir-Adleman)

RSA is an asymmetric encryption algorithm which using two different keys, a public key and private key for encryption and decryption. The public key is sued for encryption and the private key is used for decryption. RSA facilitates the secure exchange of encryption keys between VPN endpoints which ensure that the sensitive data remain confidential during transmission. The key exchange process forms the bedrock of secure communication within VPN infrastructure and safeguarding users against unauthorized access and interception. Besides, RSA enable endpoints to verity each other’s identities by using digital certificates so that only authenticated and authorized parties gain access to the VPN network which can mitigating risk associated with unauthorized intrusion and malicious activities(Cobb, 2021).

Furthermore, RSA have the ability to ensure data integrity verification through the generation of digital signatures. These signatures serve as cryptographic assurances of the authenticity and integrity of transmitted data which aim to safeguarding data against tampering and unauthorized modifications. Due to its mechanism, VPN endpoints can reliably validate the trustworthiness of data exchanges and forming a secure and trustworthy communication environment.

2.6.2 DH (Diffie-Hellman)

DH key exchange protocol is a foundational cryptographic technique utilized within VPNs to establish secure communication channels between the VPN endpoints. DH enables two parties to jointly generate a shared secret key over an insecure communication channel without ever exchanging the key itself. Initially, both parties needed to agree upon certain public parameters such as a prime number and a generator and based on this, each party will independently generate its own secret value. After that through a series of mathematical computations, the secret values are combined with the public parameters to produce a shared secret key which remains unknown to eavesdroppers even if they had intercepted the exchanged messages(Gillis, 2022).

In VPN contexts, DH key exchange occurs during the initial phase of establishing a connection where both client and server negotiate cryptographic parameters and compute the shared secret key. The key subsequently serves as the foundation for encrypting and decrypting data exchanged between the endpoints throughout the VPN session.

2.7 Comparison of VPN Application

In this section, we will compare 3 famous VPN application in current market. Discussion will cover many factors such as features, performance, strength, and weakness. By analyze each VPN application, we able to have a better guidance and insight through identify common factor, good practice and security strategy, and from this making our application better.

2.7.1 Hola VPN

Figure 2.7 Hola VPN (Macarthur, 2021)

Hola VPN is a widely used Virtual Private Network service renowned for its user-friendly interface, extensive features and expansive global server network. Hola provided user effortless experience bypassing geo restriction which allow user to access blocked content and websites. Although Hola VPN is a popular VPN service provider, but it also known with many flaws especially in terms of security such as lack of a robust user authentication and authorization mechanisms which will raising security concern. Beside Hola VPN does not provide kill switch feature which may cause data leaks if VPN server down and its unstable performance which may cause usability issue(Williams, 2020).

2.7.1.1 Hola VPN Features

- **User Access Control (Authentication and Authorization)**

  Hola VPN operates on a unique peer-to-peer network model where users share their IP addresses with others to access content from different regions, but this model lacks traditional user authentication and authorization mechanisms which potentially raising security concerns.

- **Security Features**

Hola VPN offers limited security features compared to traditional VPN services due to its lack of robust encryption protocols and a kill switch, which may compromise user privacy and security.

- **Device Compatibility**

  Hola VPN is compatible with desktops, mobile devices and offers browser extensions for popular browsers which providing wide device support.

- **Performance**

  Hola VPN's performance can vary depending on the peer-to-peer network's availability and user bandwidth. Speeds may be inconsistent and users may experience latency issues when using the service.

- **Privacy and Security:**

  Hola VPN has faced criticism regarding its privacy practices and lack of a clear no-logs policy which potentially compromising user privacy informed consent.

- **User Interface and Experience:**

  Hola VPN offers a user-friendly interface with an easy setup process. It provides tutorials and support documentation for assistance which making it accessible to users with varying technical expertise levels.

2.7.2 Hotspot Sheild

Figure 2.8 Hotspot Sheild (i2Coalition, 2020)

Hotspot Sheild is a Virtual Private Network service offered by AnchorFree. It is one of the most popular VPN service providers known for its ease of use, fast connection speed and broad platform support. Hotspot Sheild designed to offer users enhanced privacy, security and access to geo-restricted content by encrypts internet traffic, routing through VPN server to safeguard data against interception. Hotspot Sheild encrypting data between users' devices and its servers to ensures confidentiality protecting against cyber threats, ISP monitoring, and government surveillance.

However, despite its popularity, Hotspot Sheild faced scrutiny regarding its security practices. One of the major concerns of this VPN is its logging policy which has raised question about the collection and potential sharing of user's connection log such as IP address and timestamps while the company claims not to log users' browsing activity. Moreover, some users have raised doubts about the robustness of its encryption compared to other VPN providers and this can potentially leave users vulnerable to surveillance or attack(Millares, 2024).

2.7.2.1 Hotspot Sheild Features:

- **User Access Control (Authentication and Authorization)**

Hotspot Shield provide user authentication through account creation and login credentials which ensuring secure access to the VPN service.

- **Security Features**

  Hotspot Shield utilizes strong encryption protocols, including the proprietary Catapult Hydra protocol which providing secure data transmission when using the service. It also includes a kill switch feature for added security. Besides, Hotspot Shield utilize Transport Layer Security (TLS) encryption which designed to provide secure communication over a network.

- **Device Compatibility**

  Hotspot Shield supports desktop and mobile devices including Windows, macOS, Android and iOS. It also provides browser extensions for Chrome and Firefox for ensuring broad device compatibility.

- **Performance**

  Hotspot Shield is known for its fast speeds due to its proprietary Catapult Hydra protocol which able providing users with a reliable and high-performance VPN experience.

- **Privacy and Security**

  Hotspot Shield has improved its privacy policy over time but there have been concerns about logging user data. However, it offers strong encryption and DNS leak protection, prioritizing user security.

- **User Interface and Experience**

  Hotspot Shield provides a user-friendly interface with an easy setup process. It offers tutorials and support documentation for assistance which ensuring a positive user experience.

2.7.3 Surfshark VPN

Surfshark®

Figure 2.9 Surfshark VPN (Max Eddy , 2023)

Surfshark VPN is widely regarded as one of the premier VPN service providers in the industry. Renowned had implemented robust security features, user-friendly interface extensive server network spanning numerous countries and has earned a reputation for reliability and performance. Surfshark operates under a strict no-logs policy, ensuring user data remains confidential and by implementing advanced encryption protocols and built-in ad blocker offer users peace of mind while browsing the internet (Attila, 2024).

We will discover multiple factors to analyzing Surfshark’s feature and identify areas for improvement in our own applications to ultimately striving to deliver a superior VPN experience to our users.

2.7.3.1 Surfshark VPN features

➢ **User Access Control (Authentication and Authorization)**

Surfshark provides multiple authentication methods including username/password, two-factor authentication (2FA) and in some cases it also supports for authentication via third-party identity providers like Google or Facebook which provide a solid basis for secure access to their service based on authentication and authorization.

- **Security Features**

  Surfshark VPN uses AES-256 encryption which is considered one of the most secure encryption standards in current industry and it also offers kill switch feature which will terminates internet traffic if the VPN connection drops unexpectedly to preventing data leaks. Besides, it also supports various VPN protocols such as OpenVPN and IKEv2/IPsec for providing flexibility and compatibility to users with different devices and networks.

- **Device Compatibility**

  Surfshark VPN is compatible with a wide range of devices and platforms such as on Windows, macOS, iOS, Android, Linux and routers. Besides, it also offers browser extensions for popular web browsers like Chrome and Firefox. Surfshark broad device compatibility attribute ensures that users can protect their online activities across multiple devices and operating systems which provide huge convenient for user's experience.

- **Performance**

  Surfshark VPN generally offers good performance with fast connection speeds and low latency for their VPN service. Actual performance may vary depending on factors such as server location, network congestion and the user's internet connection speed but overall Surfshark does provide good performance for their VPN service. Besides, Surfshark has maintains a large network of servers in numerous countries worldwide which allowing users to find a server with optimal performance for their needs.

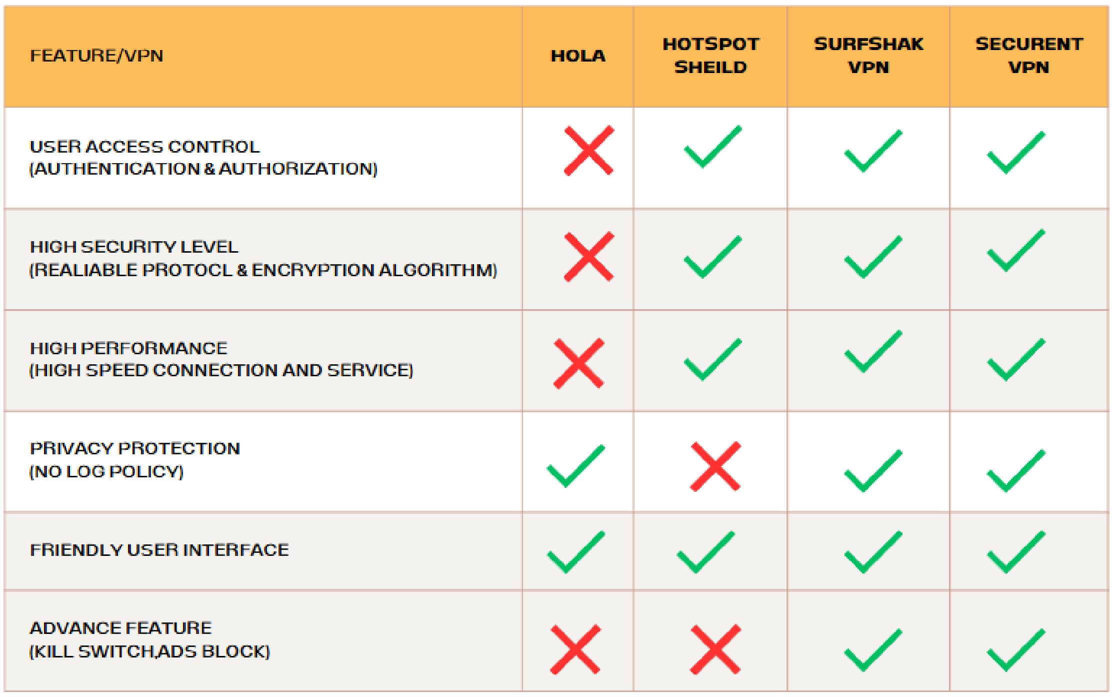
- **Privacy and Security**

  Surfshark VPN has strong privacy policy that states it does not log user's information such as their browsing activity, IP addresses or other personally identifiable information.

- **User Interface and Experience**

  Surfshark VPN offers user-friendly apps and interfaces across various devices and platforms. The application are well-designed, intuitive and easy to use which making it simple for users to carry out multiple operation such as connect to VPN servers, configure settings and access additional features. Surfshark also provides comprehensive support documentation and customer support channels to assist users with any questions or issues they may encounter.

2.7.4 Chart of Comparison



FEATURE/VPN	HOLA	HOTSPOT SHEILD	SURFSHAK VPN	SECURENT VPN
USER ACCESS CONTROL (AUTHENTICATION & AUTHORIZATION)	✗	✓	✓	✓
HIGH SECURITY LEVEL (REALIABLE PROTOCL & ENCRYPTION ALGORITHM)	✗	✓	✓	✓
HIGH PERFORMANCE (HIGH SPEED CONNECTION AND SERVICE)	✗	✓	✓	✓
PRIVACY PROTECTION (NO LOG POLICY)	✓	✗	✓	✓
FRIENDLY USER INTERFACE	✓	✓	✓	✓
ADVANCE FEATURE (KILL SWITCH,ADS BLOCK)	✗	✗	✓	✓

Figure 2.10 Chart of Comparison

2.8 Reason For Choosing OPENVPN

In the rapidly evolving landscape of network security, the choice of Virtual Private Network (VPN) solutions plays a crucial role in ensuring secure communication over the internet. For our VPN project, OpenVPN has been chosen due to its proven track record in reliability, robust security features, and outstanding performance. This section delves into the detailed reasons why OpenVPN is the preferred solution and supported by insights from scholarly articles and technical evaluations.

2.8.1 Reliability of OpenVPN

OpenVPN is widely recognized for its reliability in various networking environments. Its ability to maintain stable connections over long periods and across different network conditions is a testament to its robust architecture. OpenVPN use of User Datagram Protocol (UDP) as its default transport layer protocol contributes significantly to this reliability. UDP which being a connectionless protocol allows OpenVPN to handle network fluctuations without dropping connections and this attribute making it ideal for environments with variable network quality.

Moreover, OpenVPN’s ability to traverse Network Address Translation (NAT) and firewalls without compromising performance further underscores its reliability. The software’s design includes features like adaptive compression and automatic reconnection, which ensure that even in the face of packet loss or network interruptions, which make the VPN connection remains stable and effective (Muhammad Iqbal & Imam Riadi, 2019).

2.8.2 Security of OpenVPN

Security is one of the most compelling reasons for choosing OpenVPN. It uses a combination of cryptographic techniques to secure data transmission which including SSL/TLS for key exchange and AES-256 encryption for data protection. These technologies are industry standards for secure communications which providing a high level of assurance that data transmitted over OpenVPN is well-protected against eavesdropping and tampering.

OpenVPN also supports a variety of authentication methods, its supports including X.509 certificates which provide a secure and flexible mechanism for verifying the identities of parties involved in the communication. Besides, the use of Perfect Forward Secrecy (PFS) further enhances security by ensuring that even if a session key is compromised, it cannot be used to decrypt past sessions (Aleksandar Skendzic & Božidar Kovačić, 2017).

Additionally, OpenVPN's integration with OpenSSL provides robust support for cryptographic algorithms and ensures that security updates and patches are promptly applied, maintaining the software's resistance to emerging threats. This integration also means that OpenVPN benefits from the continuous improvements and audits conducted by the broader OpenSSL community which is critical for staying ahead of potential vulnerabilities (Muhammad Iqbal & Imam Riadi, 2019).

2.8.3 Performance of OpenVPN

Performance is another area where OpenVPN excels. The protocol is designed to be lightweight to provide capabilities allowing it to operate efficiently even on lower-end hardware without significant performance degradation. This efficiency is partly due to its use of the UDP protocol which is less resource-intensive compared to TCP protocol, particularly in environments with high latency or jitter (Aleksandar Skendzic & Božidar Kovačić, 2017).

OpenVPN also supports multi-threading, allowing it to take full advantage of multi-core processors to improve throughput and reduce latency. This capability is crucial for organizations that need to secure high volumes of traffic without introducing bottlenecks. Furthermore, OpenVPN’s ability to work seamlessly with different network infrastructures, including wireless networks, VPNs over satellite, and mobile networks, demonstrates its versatility in delivering high performance under various conditions (Muhammad Iqbal & Imam Riadi, 2019).

Performance evaluations have shown that OpenVPN can achieve high throughput rates while maintaining low latency, making it suitable for both high-bandwidth applications like video streaming and critical low-latency applications like VoIP. Its support for adaptive compression also optimizes the amount of data transmitted, further enhancing performance (Aleksandar Skendzic & Božidar Kovačić, 2017).

2.8.4 Proven Track Record

OpenVPN’s reliability, security, and performance are not just theoretical advantages instead it has been proven in real-world scenarios. Many enterprises, government agencies and non-profits rely on OpenVPN to protect their communications. Its open-source nature allows for continuous peer review and community-driven improvements, ensuring that the software remains up-to-date with the latest security practices and performance enhancements (Aleksandar Skendzic & Božidar Kovačić, 2017).

Moreover, OpenVPN has been subjected to rigorous testing and audits by independent security experts which further validating its effectiveness as a secure and reliable VPN solution. Its adoption across various sectors is a strong endorsement of its capability to meet the demanding requirements of modern secure communications (Muhammad Iqbal & Imam Riadi, 2019)..

2.8.5 Conclusion

In conclusion, OpenVPN stands out as a highly reliable, secure, and performant VPN solution, making it the ideal choice for our VPN project. Its proven ability to maintain stable connections, robust security protocols, and efficient performance across diverse network conditions provides the assurance needed for protecting our communications. The combination of these factors, supported by strong community and commercial backing, ensures that OpenVPN will continue to be a leading VPN solution for years to come.

2.9 Reason For Choosing AES-256-GCM

In this project AES-256-GCM encryption algorithm is chosen for data encryption for the data transmit via the VPN tunnel.

2.9.1 Introduction to AES-256-GCM

AES-256-GCM (Advanced Encryption Standard with a 256-bit key in Galois/Counter Mode) is a widely adopted encryption algorithm that provides a combination of strong data encryption and authentication. It is a specific instance of AES-GCM, where AES is employed with a 256-bit key size (McGrew, D & Viega, 2005).

AES is a symmetric key encryption standard established by the U.S. National Institute of Standards and Technology (NIST) in 2001. It supports three key sizes: 128 bits, 192 bits, and 256 bits. The “256” in AES-256 refers to the length of the encryption key in bits (McGrew, D & Viega, 2005).

GCM (Galois/Counter Mode) is an advanced mode of operation for cryptographic block ciphers. It provides both data confidentiality (encryption) and data integrity (authentication) in a highly efficient manner and particularly suitable for parallel processing and high-speed environments (McGrew, D & Viega, 2005).

Why AES-256-GCM is Used

2.9.2 Strong Security

The primary reason for choosing AES-256-GCM for OpenVPN is its high level of security. AES-256 uses a 256-bit key, which provides an extremely large key space. This means that it would require an infeasible amount of computational power to break the encryption through brute-force attacks. Given current computational capabilities, AES-256 is considered secure against all known forms of cryptographic attacks (McGrew, D & Viega, 2005).

2.9.3 Comprehensive Protection with GCM

GCM mode not only encrypts data but also provides authentication. This dual functionality ensures that data is both confidential and verified, protecting against tampering and unauthorized alterations. GCM’s authentication is based on Galois field multiplication, which is highly efficient and secure, making AES-256-GCM suitable for environments where both encryption and integrity are crucial (McGrew, D & Viega, 2005).

2.9.4 Efficiency in Performance

Despite the increased key size, AES-256-GCM remains efficient, especially on modern processors that support hardware acceleration for AES operations, such as Intel’s AES-NI (AES New Instructions (McGrew, D & Viega, 2005). GCM mode is designed to take advantage of parallel processing, which enhances its speed and makes it suitable for high-performance applications like VPNs (McGrew, D & Viega, 2005).

2.9.5 Widespread Adoption and Support

AES-256-GCM is widely supported across various platforms and devices, ensuring broad compatibility. This widespread support makes it an excellent choice for OpenVPN, which aims to provide secure and private communication for a wide range of users and devices globally (McGrew, D & Viega, 2005).

2.9.6 Regulatory Compliance

Many regulatory standards and security frameworks recommend or mandate the use of AES-256 for protecting sensitive data. By using AES-256-GCM, OpenVPN adheres to these standards, ensuring compliance with security regulations and guidelines (McGrew, D & Viega, 2005).

2.9.7 Variations of AES-256

AES comes in three key sizes: 128, 192, and 256 bits. The key size impacts both security and performance:

- **AES-128:** Faster than AES-256 due to the smaller key size, but still provides strong security. It is often used where performance is a higher priority than the extra security provided by AES-256 (McGrew, D & Viega, 2005).
- **AES-192:** Less commonly used, it offers a middle ground between AES-128 and AES-256 in terms of security and performance (McGrew, D & Viega, 2005).
- **AES-256:** Provides the highest level of security among the three, making it ideal for environments where maximum security is required, such as VPNs and government applications (McGrew, D & Viega, 2005).

AES-256, despite being slightly slower than AES-128, is chosen for its superior security, which is critical for safeguarding highly sensitive data).

2.9.8 Conclusion

AES-256-GCM is used in OpenVPN due to its robust security features, efficient performance, and widespread support. Its ability to provide both encryption and authentication makes it a versatile and powerful choice for securing communications. The use of a 256-bit key in AES-256-GCM ensures that even with the advancement of computational power, the encryption remains secure against brute-force and other forms of cryptographic attacks (McGrew, D & Viega, 2005).

Chapter3

METHODOLOGY AND WORK PLAN

3.1 Introduction

The appropriate SDLC methodology for this project will be selected and discussed. The details of implementation flow and method will be listed in the Work Breakdown Structure and the Gantt chart are provided to illustrate the start and end date of the project. Lastly the development tools for this project will also be covered in this chapter

3.2 System Development Methodology

Iterative methodology was selected for SecureNet VPN. Figure 3.1 shows the lifecycle for iterative methodology. The reason for selecting iterative methodology for this project due to many factors regarding security since in this project we focus on security and it involved in many area such as VPN protocol, encryption algorithm, multi factor authentication and authorization. The challenge in this project primarily related to the security issue, which is difficult to develop a fully secure system in once. Instead, in order to ensure our system is reliable and secure, the effort will need to focus on keep refinement for the system and keeping discover vulnerabilities and enhance the system. Besides, high usability as one our project main objectives, iterative methodology able to keep refinement on the user interface through user feedback which can ensure the high usability and user-friendly interface of our system. Therefore, iterative methodology is suitable for this project due to its nature of iterative refinement allow to continuous improvement of the system and allow quick respond and adapted to changes in new requirement and feedback received from user regarding the Virtual Private Network services.

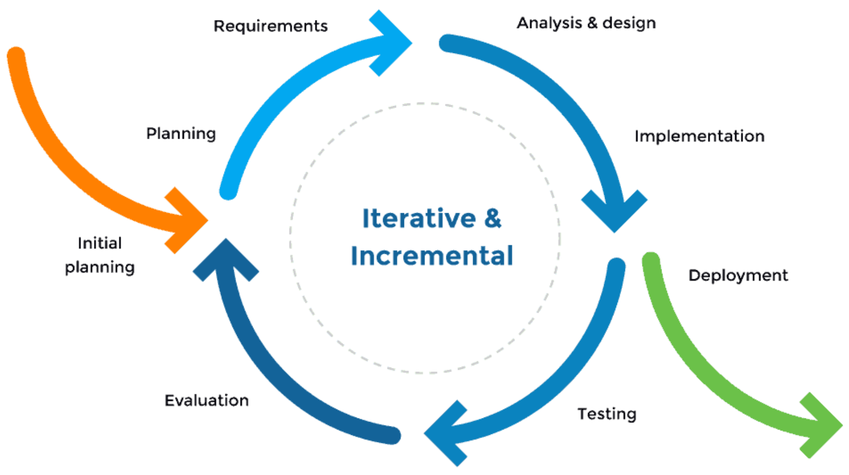


Figure 3.1 Iterative Methodology (Softwaretestingo Editorial Board, 2024)

After initial planning phase, the project will conduct multiple iteration to ensure iterative development and refinement to fulfil the requirements. Iteration will be performed 5 times and each iteration approximately around 25 to 30 days. Each iteration for this project has same process but each iteration will focus on different aspects which will be discussed.

In first iteration, main task will be focus on gathering requirement, performing analyze on requirement and design of the system. Documentation will also be performed such as use case diagram, user interface design and prototype will also be developed. Besides, the implementation of the system will also be conducted, this will cover such as setup project environment and tools and implement the basic VPN functionality.

In the second iteration, new requirement will be collected and analyzed and continue refinement for the existing system but will more focus on enhance the authentication functionality and with additional authentication methods such as multi-factor authentication and improve error handling of the system. Besides, we will also monitor the system performance and user feedback to discover any possible security vulnerabilities in the system and plan to improve it in the coming iteration.

For the third iteration, like iteration before new requirement will be collected and analyzed and continue refinement for the existing system, in this iteration, effort will keep focus on the security and Virtual Private Network part such as refine authentication and authorization mechanisms, complete integration for OpenVPN protocol and conduct extensive testing for security part and compatibility testing. Besides, documentation and user guides for the application will begin.

Forth iteration will focus on improvement of the user interface from the user feedback to ensure high usability and user-friendly interface. In the last iteration which is fifth iteration will focus on testing of the application to address any identified issues or bugs, functionalities, interoperability of the system to ensure that all component works well which related to security aspect, performance, and compatibility. After all testing completed, will prepared for deployment and official release of SecureNet VPN application.

3.3 Work Plan

In this section work breakdown structure will be planned to simplify the complex tasks into smaller tasks. Figure 3.2, figure 3.3 and figure 3.4 shows the work breakdown structure of the project. The specific work content and duration of work in each sprint are planned in Gantt Chart. Figure 3.5 and figure 3.6 display the Gantt Chart of the project.

3.3.1 Work Breakdown Structure

SecureNet VPN's Work Breakdown Structure:

1. Initial Planning

1.1 Project Planning

1.1.1 Introduction

1.1.2 Problem Background Research

1.1.3 Define Problem Statement

1.1.4 Define Project Objectives

1.1.5 Define Project Scope

1.1.6 Define Project Solution

1.2 Literature Review

1.2.1 Research of Virtual Private Network Application in Market

1.2.2 Comparison of Virtual Private Network Application

1,2,3 Research on Virtual Private Network Protocol

1.2.4 Research on Encryption Algorithms

1.2.5 Research on Authentication Algorithm

1.2.6 Research on Key Exchange Protocols

1.3 Methodology and Workplan

1.3.1 Select SDLC Methodology

1.3.2 Develop Project Workplan

1.3.3 Select Development Tools

2.Iterative Process

2.1 First Iteration

2.1.1 Requirement Gathering and Documentation

2.1.2 Analysis and Design

2.1.3 Implementation

2.1.4 Testing

2.1.5 Evaluation

2.2 Second Iteration

2.2.1 New Requirement Gathering and Documentation

2.2.2 Analysis and Design

2.2.3 Implementation

2.2.4 Testing

2.2.5 Evaluation

2.3 Second Iteration

2.2.1 New Requirement Gathering and Documentation

2.2.2 Analysis and Design

2.2.3 Implementation

2.2.4 Testing

2.2.5 Evaluation

2.3 Third Iteration

2.2.1 New Requirement Gathering and Documentation

2.2.2 Analysis and Design

2.2.3 Implementation

2.2.4 Testing

2.2.5 Evaluation

2.4 Forth Iteration

2.2.1 New Requirement Gathering and Documentation

2.2.2 Analysis and Design

2.2.3 Implementation

2.2.4 Testing

2.2.5 Evaluation

2.5 Fifth Iteration

2.2.1 New Requirement Gathering and Documentation

2.2.2 Analysis and Design

2.2.3 Implementation

2.2.4 Testing

2.2.5 Evaluation

3.Deployment Phase

3.1 System Deployment

4.Report Finalize

4.1 Complete Report Writing

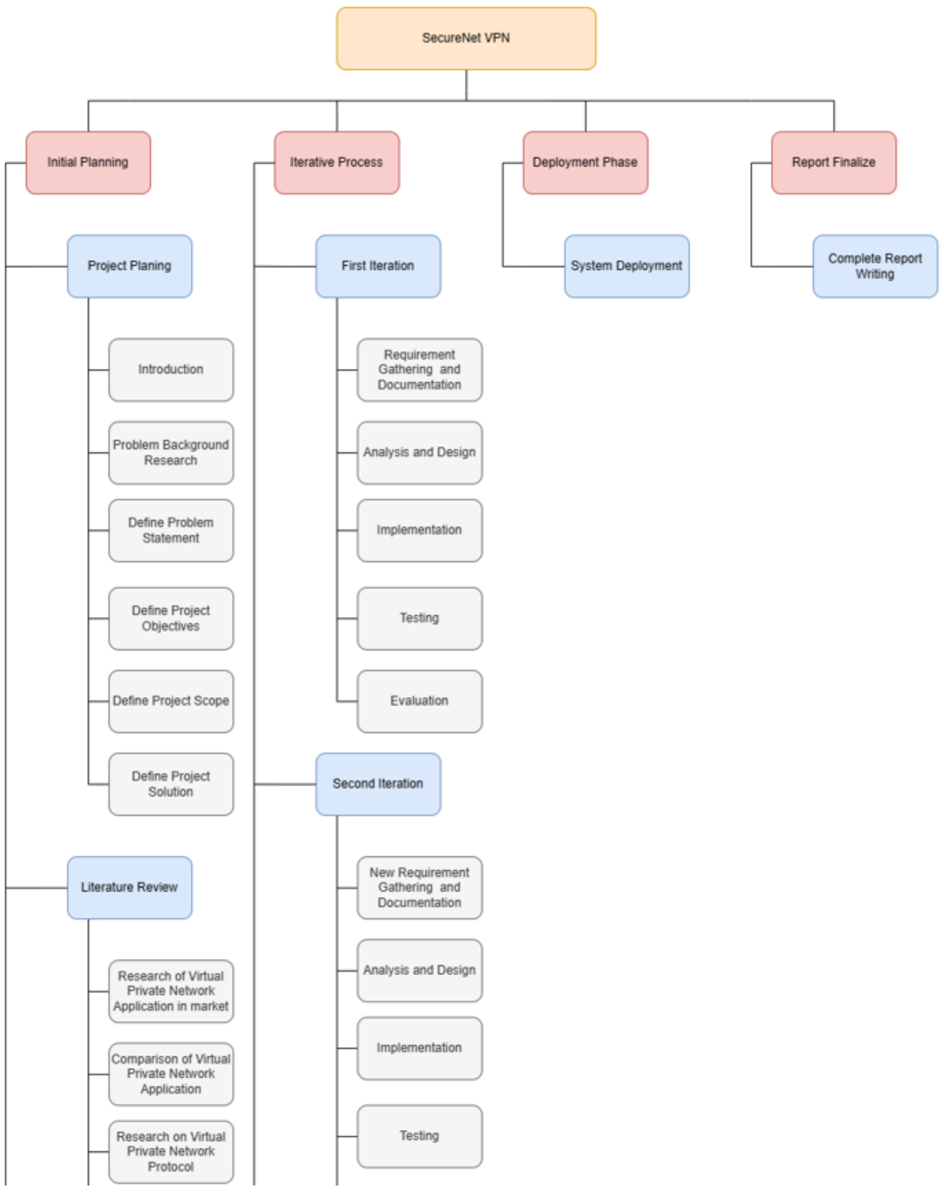


Figure 3.2 Work Breakdown Structure

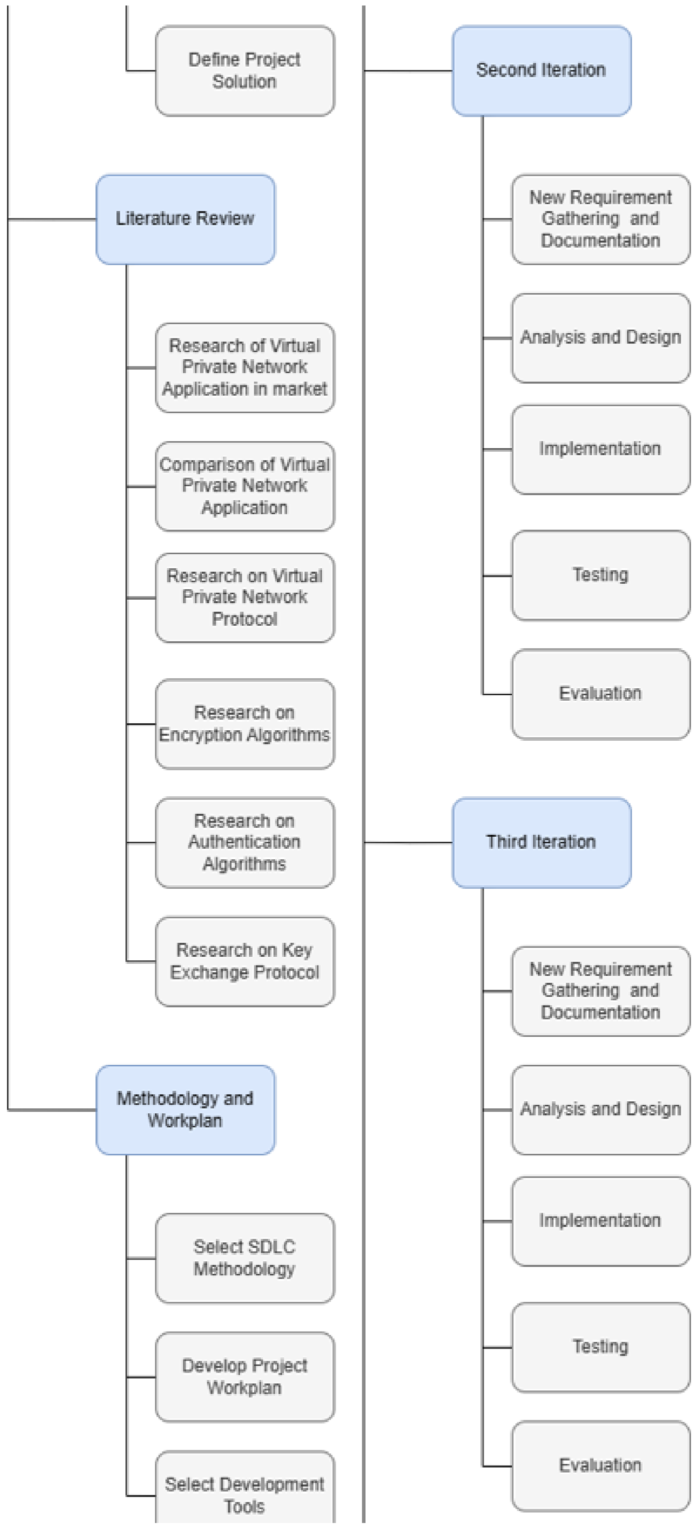

Define Project Solution
Literature Review
Research of Virtual Private Network Application in market
Comparison of Virtual Private Network Application
Research on Virtual Private Network Protocol
Research on Encryption Algorithms
Research on Authentication Algorithms
Research on Key Exchange Protocol
Methodology and Workplan
Select SDLC Methodology
Develop Project Workplan
Select Development Tools
Second Iteration
New Requirement Gathering and Documentation
Analysis and Design
Implementation
Testing
Evaluation
Third Iteration
New Requirement Gathering and Documentation
Analysis and Design
Implementation
Testing
Evaluation

Figure 3.3 Work Breakdown Structure (Continued)

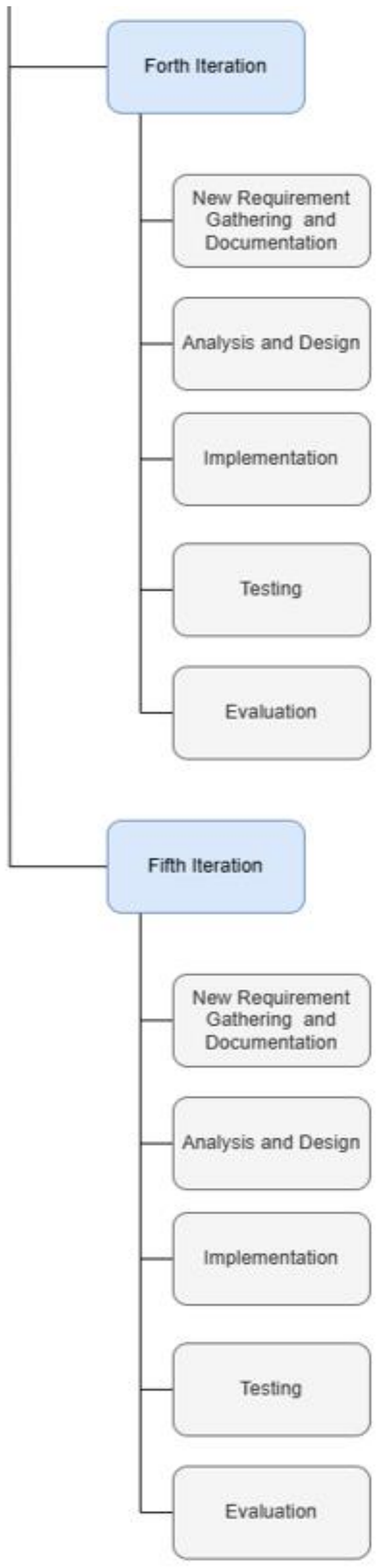
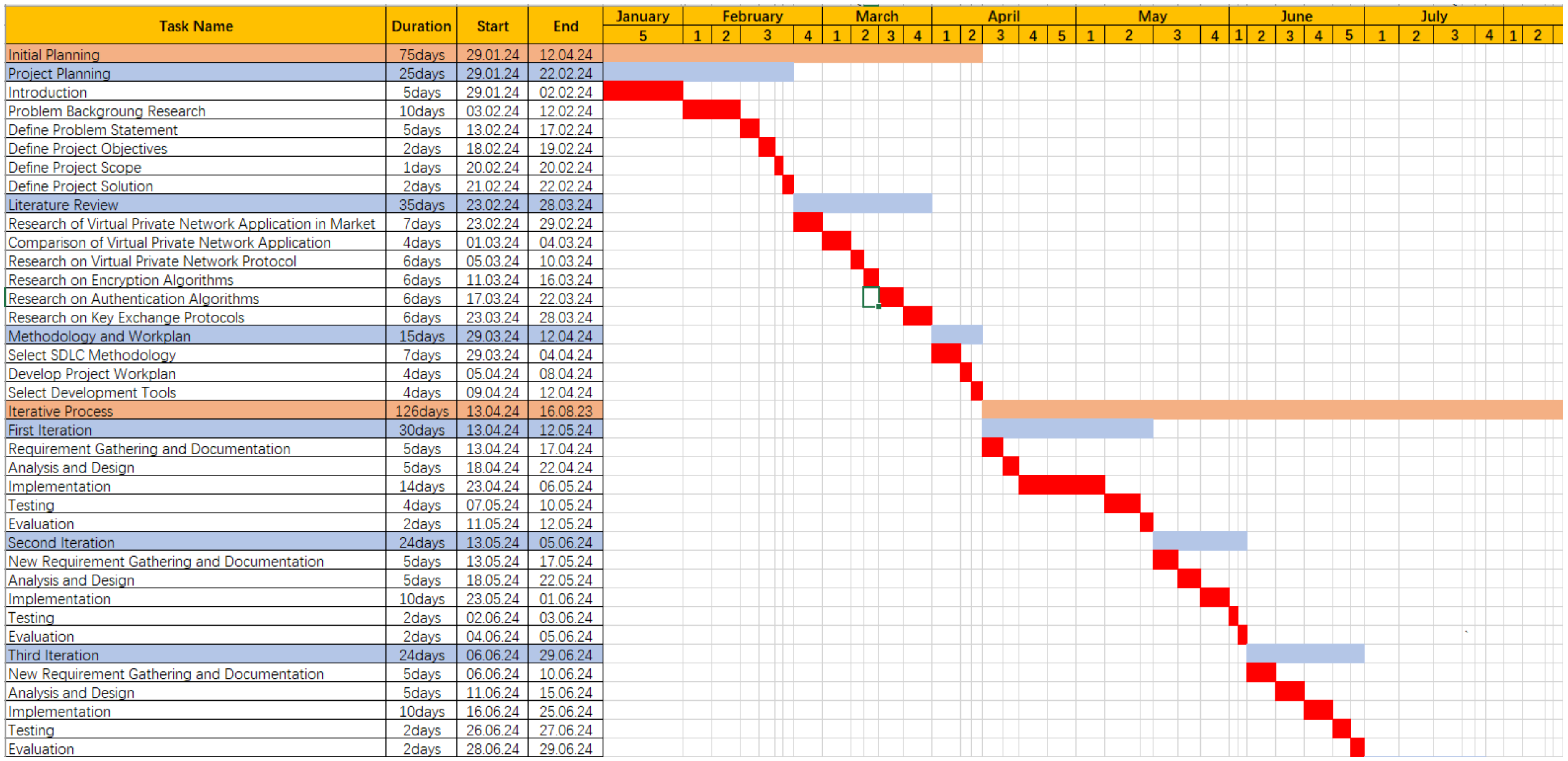


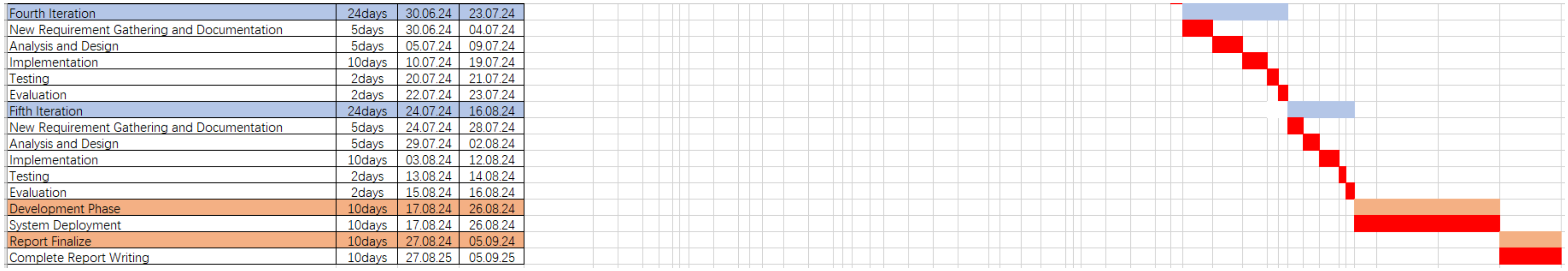
Figure 3.4 Work Breakdown Structure (Continued)

3.3.2 Gantt Chart



Task Name	Duration	Start	End
Initial Planning	75days	29.01.24	12.04.24
Project Planning	25days	29.01.24	22.02.24
Introduction	5days	29.01.24	02.02.24
Problem Backgroung Research	10days	03.02.24	12.02.24
Define Problem Statement	5days	13.02.24	17.02.24
Define Project Objectives	2days	18.02.24	19.02.24
Define Project Scope	1days	20.02.24	20.02.24
Define Project Solution	2days	21.02.24	22.02.24
Literature Review	35days	23.02.24	28.03.24
Research of Virtual Private Network Application in Market	7days	23.02.24	29.02.24
Comparison of Virtual Private Network Application	4days	01.03.24	04.03.24
Research on Virtual Private Network Protocol	6days	05.03.24	10.03.24
Research on Encryption Algorithms	6days	11.03.24	16.03.24
Research on Authentication Algorithms	6days	17.03.24	22.03.24
Research on Key Exchange Protocols	6days	23.03.24	28.03.24
Methodology and Workplan	15days	29.03.24	12.04.24
Select SDLC Methodology	7days	29.03.24	04.04.24
Develop Project Workplan	4days	05.04.24	08.04.24
Select Development Tools	4days	09.04.24	12.04.24
Iterative Process	126days	13.04.24	16.08.23
First Iteration	30days	13.04.24	12.05.24
Requirement Gathering and Documentation	5days	13.04.24	17.04.24
Analysis and Design	5days	18.04.24	22.04.24
Implementation	14days	23.04.24	06.05.24
Testing	4days	07.05.24	10.05.24
Evaluation	2days	11.05.24	12.05.24
Second Iteration	24days	13.05.24	05.06.24
New Requirement Gathering and Documentation	5days	13.05.24	17.05.24
Analysis and Design	5days	18.05.24	22.05.24
Implementation	10days	23.05.24	01.06.24
Testing	2days	02.06.24	03.06.24
Evaluation	2days	04.06.24	05.06.24
Third Iteration	24days	06.06.24	29.06.24
New Requirement Gathering and Documentation	5days	06.06.24	10.06.24
Analysis and Design	5days	11.06.24	15.06.24
Implementation	10days	16.06.24	25.06.24
Testing	2days	26.06.24	27.06.24
Evaluation	2days	28.06.24	29.06.24

Figure 3.5 Gantt Chart



Fourth Iteration	24days	30.06.24	23.07.24
New Requirement Gathering and Documentation	5days	30.06.24	04.07.24
Analysis and Design	5days	05.07.24	09.07.24
Implementation	10days	10.07.24	19.07.24
Testing	2days	20.07.24	21.07.24
Evaluation	2days	22.07.24	23.07.24
Fifth Iteration	24days	24.07.24	16.08.24
New Requirement Gathering and Documentation	5days	24.07.24	28.07.24
Analysis and Design	5days	29.07.24	02.08.24
Implementation	10days	03.08.24	12.08.24
Testing	2days	13.08.24	14.08.24
Evaluation	2days	15.08.24	16.08.24
Development Phase	10days	17.08.24	26.08.24
System Deployment	10days	17.08.24	26.08.24
Report Finalize	10days	27.08.24	05.09.24
Complete Report Writing	10days	27.08.25	05.09.25

Figure 3.6 Gantt Chart(Continued)

3.4 Development Tools

We will discuss the tools we will use and the reason for choosing the tools. We will discuss in 3 parts which is frontend, backend, and VPN server

3.4.1 Frontend Part

This project aims to provide high usability for user on Android platforms. The tools involved in the frontend part included Flutter Frameworks, Node Package Manager (NPM), Android Studio.

3.4.1.1 Flutter

Flutter is an open-source framework created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. The framework allows developers to build cross-platform mobile applications for Android and iOS using the Dart programming language. The reason for choosing Flutter for this project is due to its ability to deliver a highly customizable and visually appealing user interface. Flutter's widget-based architecture allows for rapid development and a consistent look and feel across platforms. Additionally, Flutter provides a rich set of pre-designed widgets that can be easily customized to match the project's design requirements. Flutter's strong community and ecosystem offer a vast range of packages and plugins, enabling developers to access device-specific features and functionalities with ease. This support network ensures that developers can find solutions to common challenges quickly, improving development efficiency. Moreover, Flutter's hot reload feature allows for real-time code changes, significantly speeding up the development process by allowing developers to see immediate results.

3.4.1.2 Android Studio

Android Studio is a development environment for Android applications development Its provides a comprehensive set of tools for designing, developing, testing, and debugging Android applications. In this project, we will use an emulator for Android device which can use for design and preview user interface for different screen size resolution. Besides, they had integrated with version control system which allowing developers to manage code repositories directly within the IDE and also provide tools for generate APK, configure app and monitor app performance directly from the IDE

3.4.2 Backend Part

In this project Amazon Web service will be use for deploying our VPN Server in different region and our backend for user management, authentication and authorization action. For the instance on Amazon Web service, we will be using Linux operating system, OpenVPN library for the VPN server and ??? as our backend interoperate with MySQL database for performing management of user profile

3.4.2.1 Amazon Web Service

Amazon web service is comprehensive cloud computing platform which provided wide range of services that enable businesses or anyone to used for build , deploy and manage their own applications and infrastructure in the cloud. This project offer VPN services which means we need to deploy our server among multiple country. AWS can provide basic computing node such as the elastic cloud computing (EC2) to fulfil our requirements for deploy our services on it, beside we can manage it effectively through amazon own AWS console and its charging is based on a pay-as-you-go model

3.4.2.2 OpenVPN

OpenVPN is an open-source virtual private network protocol which allowing for creating a secure point to point or site to site connection based on configuration. This protocol is widely use for secure remote access to networks and for creating secure connection between network over internet. OpenVPN protocol as the core component in this project, it able to provide high security level by employ encryption algorithms which ensure secure communication over internet, it support end-to end encryption, authentication and data integrity verification which making the communication is highly secure against malicious parties or activities like tampering, eavesdropping and other security threat

3.4.2.3 MongoDB

MongoDB is a popular open-source NoSQL database known for its flexibility, scalability, and performance. It stores data in a JSON-like BSON format, making it ideal for handling unstructured or semi-structured data. MongoDB's schema-less design allows for dynamic and evolving data models, which is particularly advantageous in agile development environments where requirements may change frequently. Its horizontal scalability, with built-in replication and sharding, ensures the database can handle large volumes of data and high traffic while maintaining high availability. MongoDB's rich query language supports complex queries and analytics, and its extensive ecosystem and active community provide valuable resources for developers, making it an excellent choice for projects requiring a flexible and robust data management solution.

3.4.2.4 Node.js

Node.js is an open-source, server-side JavaScript runtime environment which enable developers to create web servers and build scalable network applications. Node.js uses an event-driven and non-blocking I/O model, which makes it efficient and suitable for handling concurrent operations such as serving multiple requests simultaneously. It is popular for building real-time applications, RESTful APIs, and backend services due to its fast execution and vast ecosystem of libraries and frameworks.

3.4.3 Version Management Tools

3.4.3.1 **Git**

Git is a distributed version control system that used for tracking changes in our development environment. It allows to track all the changes to files and directories in the project overtime and allow to record changes, revert to previous version and provide collaborate flexibility with others on the same codebase which provided effect and efficient in managing project.

3.4.3.2 Node Package Manager (NPM)

Node Package Manager is the dependency or package manager for JavaScript runtime environment. It provides automation of update and managing dependency and installing all the necessary dependency from trusted source. Node Package Manager play crucial role in JavaScript ecosystem which enable developers to leverage vast of opensource library and tools to build applications for efficiently.

3.4.4 Other Tools

3.4.4.1 Virtual machine

Virtual machine will be used in this project for the purpose of configuring, testing, and remote access to the VPN server on the cloud services which can provide a sandboxed environment that is isolated from the host system can help prevent conflict between VPN server and other software service running on host system.

3.4.4.2 Visual Studio Code

A free open-source code editor which compatible for various programming languages and platforms. It is a lightweight and fast development tool with rich features integrated into it such as terminal integration, git integration and extensible through it extension market which provide high efficient during development.

3.5 Development Workflow

At the initial phase of this project which involves identify problem background and analyze user's need to discover the details requirements including functionality, security features, user interface and performance expectations. By analyze and prioritize the requirements based on their importance and feasibility, all the tasks will be break down into smaller task to for easing execution of the development.

After a set of analysis phase of requirement and design of the system, the next step is to embark on the development of prototype. The prototype will be develop using Figma which will serve as preliminary version of SecureNet Application which able to offer a tangible representation of the applications' functionalities and design. The major objective of prototyping is to provide stakeholders and end-users a concrete visualization of the proposed solution to enabling them to provide feedback and validate the proposed approach. Through early feedback from prototyping, the application can refine and iterate on design and functionality of the application which ensuring all requirement aligns with identified requirements and needs. After that, based on prototype, the production of frontend user interface will majorly using Flutter to build an interactive application which specifically for Android Platform.

Following the frontend development phase, the focus shifts to backend implementation of SecureNet VPN application. The tools and environment needed here including AWS EC2(Elastic Cloud Compute) instance, OpenVPN library, Certificate Authority (CA) and Easy-RSA. Once EC2 instance is provisioned, OpenVPN library packages and dependencies needed to download on the EC2 instance which includes installing OpenVPN software along with all the libraries and utilities required for its operation. Besides, setting up a Certificate Authority (CA) and generating RSA key pairs which able to facilitate the secure communication between the VPN server and client devices.

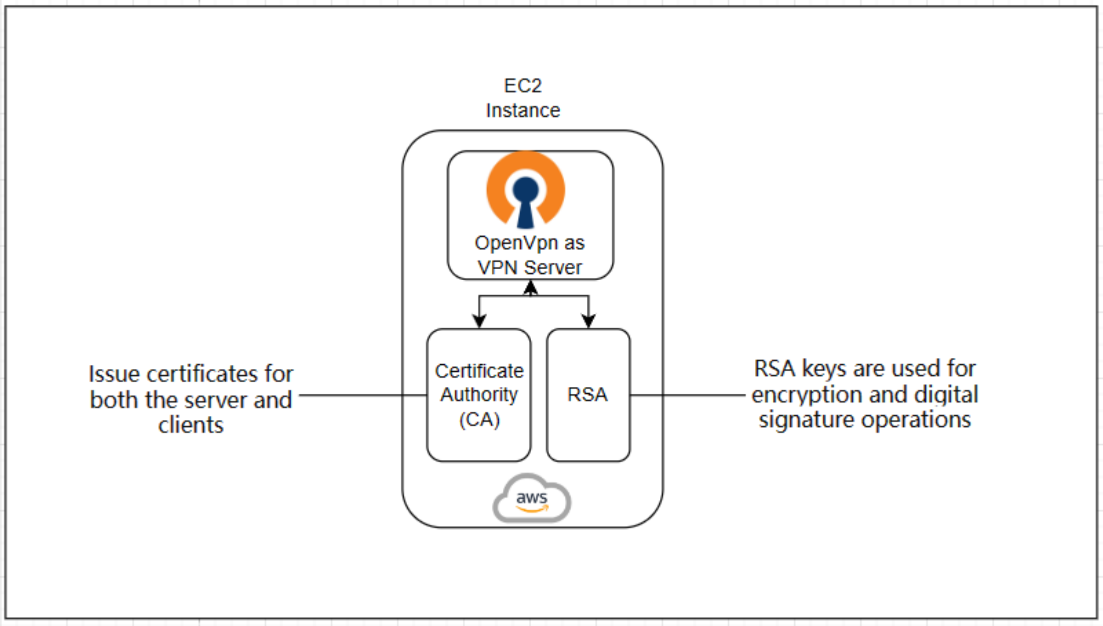

EC2
Instance
OpenVpn as VPN Server
Certificate Authority (CA)
RSA
aws
Issue certificates for both the server and clients
RSA keys are used for encryption and digital signature operations

Figure 3.7 Workflow of OpenVPN with CA and RSA

With OpenVPN installed and all necessary security tools has been setup, next phase entails with setting up and configuring the VPN server to establish secure connections with client devices. This including configures VPN network setting, like IP address, routing subnet configuration, encryption protocols, authentication algorithms, port number parameters to ensure secure communication channels. After that we will need to setup Certificate Authority (CA) to establishing trust and security within the VPN infrastructure.CA is responsible for issuing digital certificates that authenticate the identities of the VPN server and client devices which CA will generates a root certificate and private key to sign server and client certificates to ensure the integrity and authenticity of communication channels. RSA key pairs play crucial role in encrypting data and facilitating secure authentication within VPN ecosystem. RSA keys are generated for the VPN server and client devices which comprising a public key for encryption and a private key for decryption.

For connected to the VPN server, client configuration file is needed to for client to establish connections from their device. After client has register themselves in database or if their configuration file is expired, backend server will be invoked to request VPN server to generate new client configuration file which contain all necessary parameters such as IP address, port number, authentication, and encryption algorithms as well as along with the encrypted certificate and key pair to ensure integrity. If any of the encrypted key pair or certificate had been modify which mean the document is not more secure and the VPN server will reject to establish connection between it with client, this process also been known as Transport Layer Security (TLS) process.

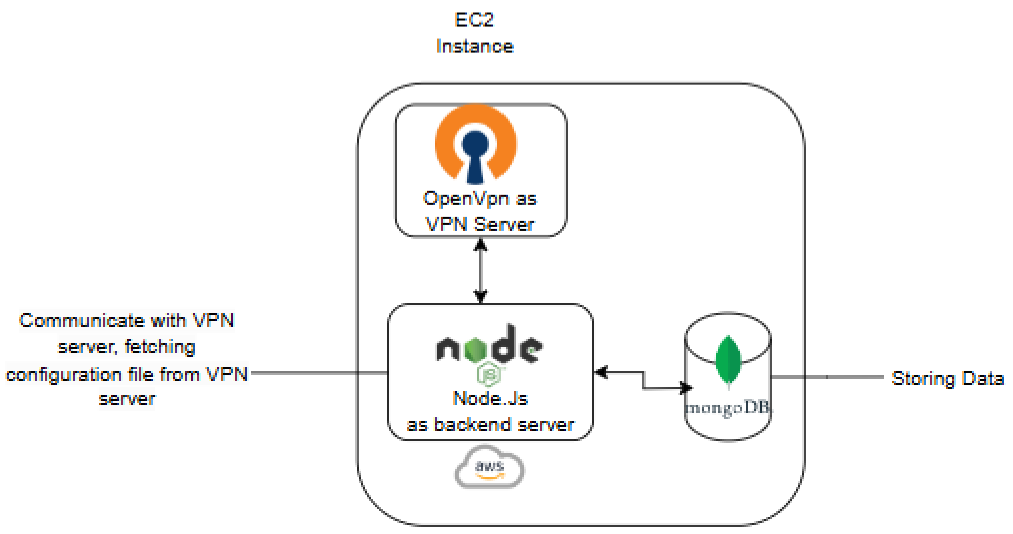


Figure3.8 Workflow of Backend Server

For another backend which using NodeJS major function is for authentication and authorize users, store user's profile and hosting RESTful API endpoints for responding request from client. The backend is hosted on the same EC2 instance with the VPN server which aiming to invoke the VPN server for several purposes such as generating new client configuration file for establish VPN connections and fetching the file from VPN server. MongoDB database as reliable data storage will work with NodeJS backend for storing and managing user's profile and necessary information.

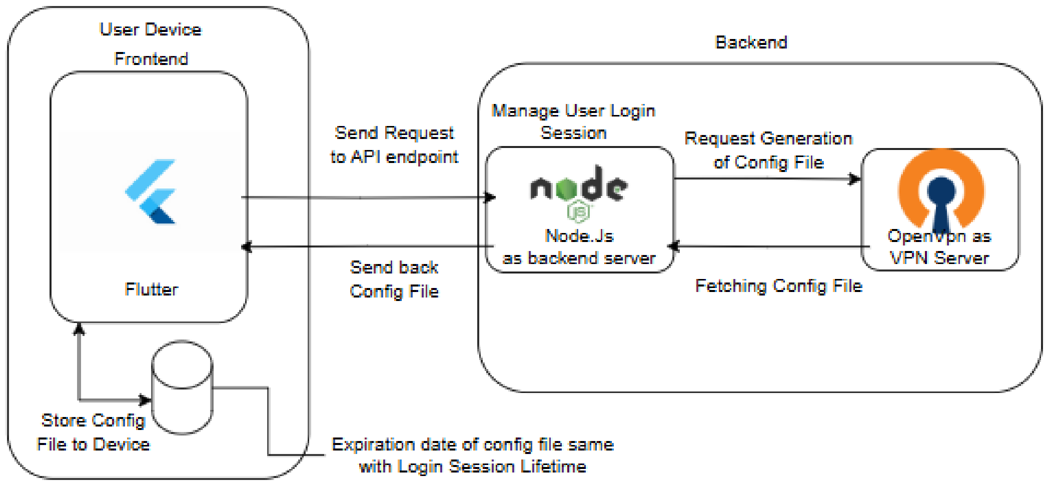


Figure 3.9 Frontend Workflow

For frontend part, frontend will integrate with OpenVPN library for establish connections to the server. On client device, the application will first fetch the configuration file first when they login or register themselves, the flow will be request to backend API endpoint and backend server will communicate with VPN server in order to generate or fetching generated file from database. The configuration file will be stored in user device and will set an expiration date along with the login session lifetime with using JWT(JSON WEB TOKEN). This ensures that user will need to re-login themselves after a period of time and ensure the renew of configuration file to prevent abuse such as distribution of configuration file.

After setting up all frontend, VPN server and backend server, testing will be covered to ensure interoperability of frontend, VPN server and backend server works well. Activities included unit testing, end-to-end testing, and user acceptance testing.

Final step is deployment of the SecureNet VPN Application online by hosting VPN server and backend server on Amazon Web Service and distribute the frontend application of SecureNet VPN for users so they can access to the server anytime. Besides, the system architecture, database schemas, description of server, technical details, used dependencies, API endpoints will be documented and accessible for users for continuous update, bug fixes and refinement of the application.

Chapter 4
Project Specification

4.1 Introduction

This chapter will cover the detailed requirements specification of this project, functional and non-functional requirements of the application will be defined. Use case, activity diagram and data flow diagram will outline interaction and processes of SecureNet VPN application. These information will be serves as foundation of the design and development of SecureNet VPN.

4.2 Requirement Specification

Requirement specification like a blueprint to defines what the application should do and behave, and for stakeholders to know what they can expect from the final product. The requirement is extract through the review of similar applications.

4.2.1 Functional Requirements

4.2.1.1 Frontend Functional Requirements

1. User Registration and Authentication
   - Allow users to register new accounts securely
   - Provide options for users to log in and log out of their accounts.

2. Country Selection
   - Present users with a list of available countries to select for VPN connection.
   - Display relevant information about each country, such as server location and connection speed.

3. Feedback Integration
   - Allow stakeholders and end-users to provide feedback on the application's functionalities and design

4.2.1.2 Backend Functional Requirements

1. User Authentication and Authorization
    - Implement secure authentication mechanisms to verify user identities.
    - Authorize users to access specific features based on their roles and permissions.

2. Profile Management
    - Store and manage user profiles securely in a MySQL database.
    - Allow users to update their profiles, including personal information and preferences

3. Client Configuration Handling
    - Handle requests from users to generate new client configuration files.
    - Ensure secure generation and delivery of configuration files to users 4.Integration with VPN Server
    - Facilitate communication between the frontend and VPN server for VPN connection setup

4.2.1.3 VPN Server Functional Requirements

1. Client Configuration Handling
    - Establish a Certificate Authority (CA) for issuing digital certificates
    - Generate RSA key pairs for encryption and authentication purposes

2. Client Configuration Handling
    - Configure VPN network settings, including IP addresses, routing subnet configurations, and encryption protocols.
    - Manage server parameters such as port numbers and authentication algorithms for secure communication channels.

3. Client Connection Handling
    - Receive and process requests from clients for establishing VPN connections.
    - Verify client certificates and key pairs for authentication and encryption.

4.2.2 Non-Functional Requirements

- The application should provide fast and responsive performance, with minimal latency during VPN connections
- The application's backend infrastructure should be optimized for scalability to handle increased user loads
- All data transmitted between the client and server should be encrypted using strong encryption algorithms. The application should implement multi-factor authentication for user logins to enhance security.
- User credentials and sensitive information should be securely stored and hashed to prevent unauthorized access
- The VPN connections should be stable and reliable, with automatic reconnection in case of network disruptions
- The user interface should be intuitive and easy to navigate, catering to users of varying technical expertise
- The application should provide clear instructions and error messages to guide users through the setup and configuration process
- It should offer multi-platform support and compatibility with a wide range of devices to maximize usability
- The VPN server should be accessible 24hours

4.3 Use Case Diagram

A use case diagram is provided to illustrate the interactions between users and the system by highlighting the various functionalities that the system supports from a user's perspective.

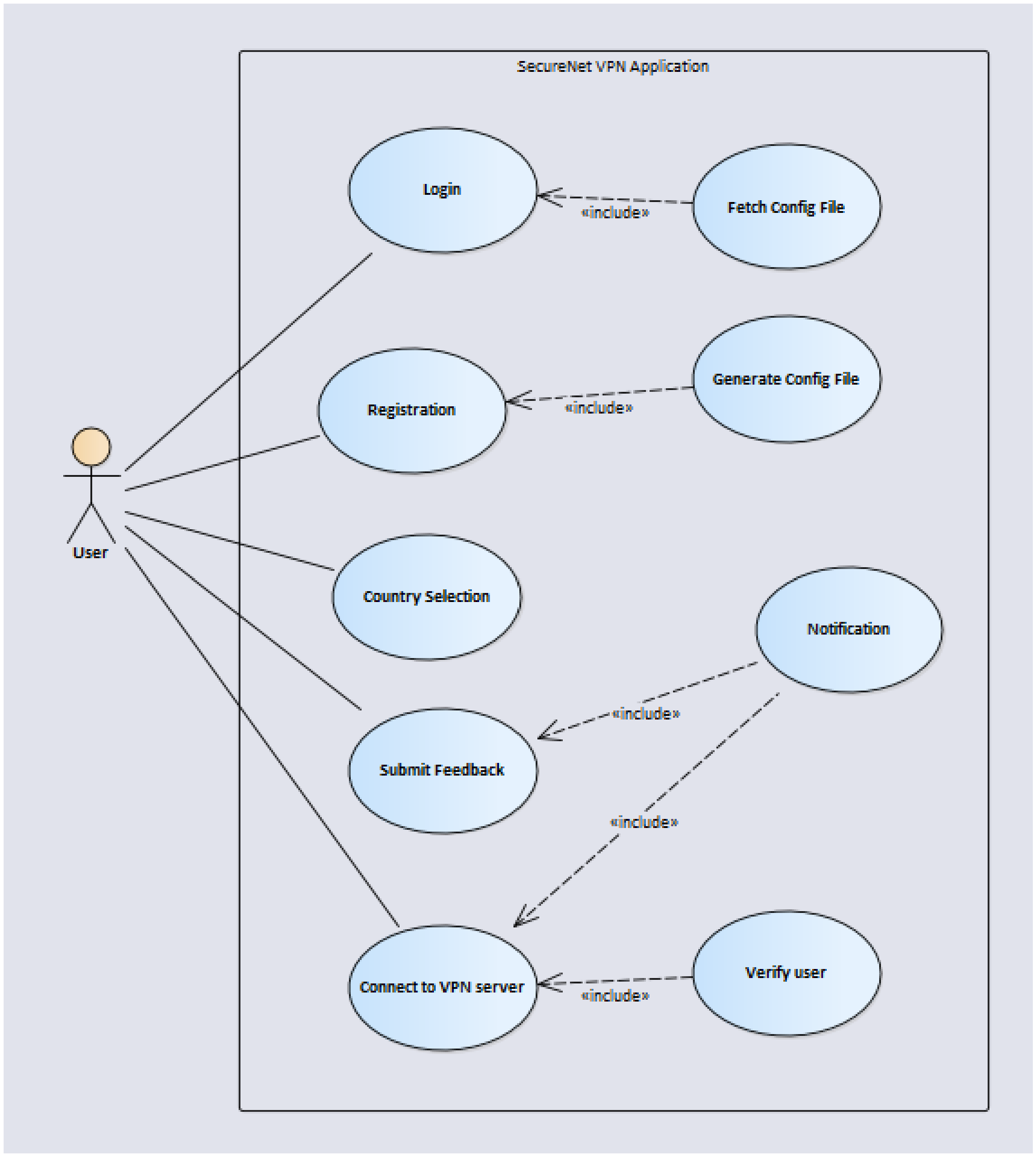


Figure 4.1 Use Case Diagram of SecureNet VPN Application

4.4 Use Case Description

Each use case description will describe in detail and show it relationship between use case and flow of event

4.4.1 Login

Use Case Name: Login	**ID:** UC01	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to login to the application		
Brief Description: This use case describes that user login to the application.		
Trigger: User wants to login to the application		
Relationships: Association :User Include :Fetch Config File Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. User access to the application 2. User select Login option 3. System will redirect user to login page. If user login before, perform exceptional flow 1.1 4. System will prompt user to input their information such as password and id 5. System will check user credential based on user input 6. If user data is not found, perform exceptional flow 1.2 7. If data is found and match ,system will redirect to main page of the application		
Sub-flows:N/A		
Alternate/Exceptional Flows: **1.1** Session found in application 1. System will redirect to main page without need user input their credential 1.2Data not found in database 1. System will prompt user to reinput their credential		

4.4.2 Registration

Use Case Name: Registration	**ID:** UC02	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to register to the application		
Brief Description: This use case describes that user register to the application.		
Trigger: User wants to register to the application		
Relationships: Association :User Include :Generate Config File Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. User access to application 2. User select register option 3. System navigate user to register page 4. System will prompt user to input email and password for registration 5. User input the id and password to the system 6. System validates both input is validated. If not perform exceptional flow 2.1. 7. System will show successful of the registration 8. System will navigate user to main page of the application		
Sub-flows:N/A		
Alternate/Exceptional Flows: **2.1 Error input for registration** 1. System will prompt user regarding the error of input 2. User need to reinput their information. 3. Continue to flow 4.		

4.4.3 Country Selection

Use Case Name: Country Selection	**ID:** UC03	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to select country of VPN connection		
Brief Description: This use case describes that user wants to select country of VPN connection.		
Trigger: User wants to select country of VPN connection		
Relationships: Association :User Include :N/A Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. User select the country they wish to connect to. 2. System fetch new config file for establish connection 3. System will prompt the change was successful 4. The country logo will be change to the selected country		
Sub-flows:N/A		
Alternate/Exceptional Flows:N/A		

4.4.4 Submit Feedback

Use Case Name: Submit Feedback	**ID:** UC04	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to submit feedback		
Brief Description: This use case describes that user submit feedback		
Trigger: User wants to submit feedback		
Relationships: Association :User Include : Notification Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. User select the feedback button. 2. System will show feedback form to user 3. User will input their feedback into the form 4. User submit the form 5. System will validate the input to check if any empty input.If contain empty input perform exceptional flow 3.1 6. System show that feedback submitted successfully		
Sub-flows:N/A		
Alternate/Exceptional Flows: **3.1 Error input for feedback** 1. System will prompt user regarding the error of input 2. User need to reinput their information. 3. Continue to flow 6.		

4.4.5 Connect to VPN Server

Use Case Name: Connect to VPN server	**ID:** UC05	**Importance Level:** High
Primary Actor: User	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to establish connection to VPN		
Brief Description: This use case describes that user wants to establish connection to VPN.		
Trigger: User wants to establish connection to VPN		
Relationships: Association :User Include :Verify User ,Notification Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. User click the button of connect to VPN to establish connection to VPN 2. System will prompt user to validate their identity before successfully connect to VPN 3. If user fail to validate themself, Perform exceptional flow 4.1 4. If user success to validate themselves, the connection will establish. If connection not successful, Perform exceptional flow 4.2 5. System will show connection successful.		
Sub-flows:N/A		
Alternate/Exceptional Flows: **4.1 Error in validate identity** 1. System will prompt that user fail to validate their identity 2.User need to validate their identity again 2.If fail many times, system will cancel the request to establish connection 4.2 Error in establishing connection 1. System will prompt connection error 2.User have to reclick connect button to establish connection again		

4.4.6 Fetch Config File

Use Case Name: Fetch Config File	**ID:** UC06	**Importance Level:** High
Primary Actor: N/A	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to login to the application		
Brief Description: This use case describes that user config file will be fetch after successfully login		
Trigger: User successfully login to the application		
Relationships: Association :N/A Include :N/A Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. After user successful login to the application, the client application will fetch the user config file from database 2. The config file will store in user device		
Sub-flows:N/A		
Alternate/Exceptional Flows:N/A		

4.4.7 Generate Config File

Use Case Name: Generate Config File	**ID:** UC07	**Importance Level:** High
Primary Actor: N/A	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – wants to register to the application		
Brief Description: This use case describes that user config file will be generate after successfully register		
Trigger: User successfully register to the application		
Relationships: Association :N/A Include :N/A Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. After user register login to the application, the client application will request to backend server 2. Backend server will request to VPN server to generate new config file 3. Backend server will store the config file in database		
Sub-flows:N/A		
Alternate/Exceptional Flows:N/A		

4.4.8 Notification

Use Case Name: Notification	**ID:** UC08	**Importance Level:** High
Primary Actor: N/A	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – establish connection to VPN, successfully submit feedback		
Brief Description: This use case describes that user will receive notification when establish connection to VPN and submit feedback		
Trigger: User establish connection to the application and submit feedback		
Relationships: Association :N/A Include :N/A Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. Establish to connection to VPN 1.1 User successfully establish connection to VPN 1.2 The application will pop up a notification showing current connection status 2. Submit feedback successfully 2.1 User submitted their feedback to application 2.2 The application will pop up a notification showing user successfully submit their feedback		
Sub-flows:N/A		
Alternate/Exceptional Flows:N/A		

4.4.9 Verify User

Use Case Name: Verify User	**ID:** UC09	**Importance Level:** High
Primary Actor: N/A	**Use Case Type:** Detail, Essential	
Stakeholders and Interests: User – establish connection to VPN		
Brief Description: This use case describes that user will need to verify their identify before establish connection to VPN		
Trigger: User initiate the connection to VPN		
Relationships: Association :N/A Include :N/A Extend :N/A Generalization:N/A		
Normal Flow of Events: 1. The application will pop up various authentication method such as fingerprint to check user identity. 2. System will query and validate the information from database 3. If validate success, system will proceed the next action after this function 4. If validate fail.Perform exceptional flows 5.1		
Sub-flows:N/A		
Alternate/Exceptional Flows 5.1Fail in identify verify 1. The application will prompt user fail to verify themselves 2. The application will show error		

4.5 Activity Diagram

Activity diagrams are provided to visually represent the flow of activities and process within SecureNet VPN system which can helping stakeholders understand the sequential order, decision point and interactions involved in achieving specific task.

4.5.1 Login

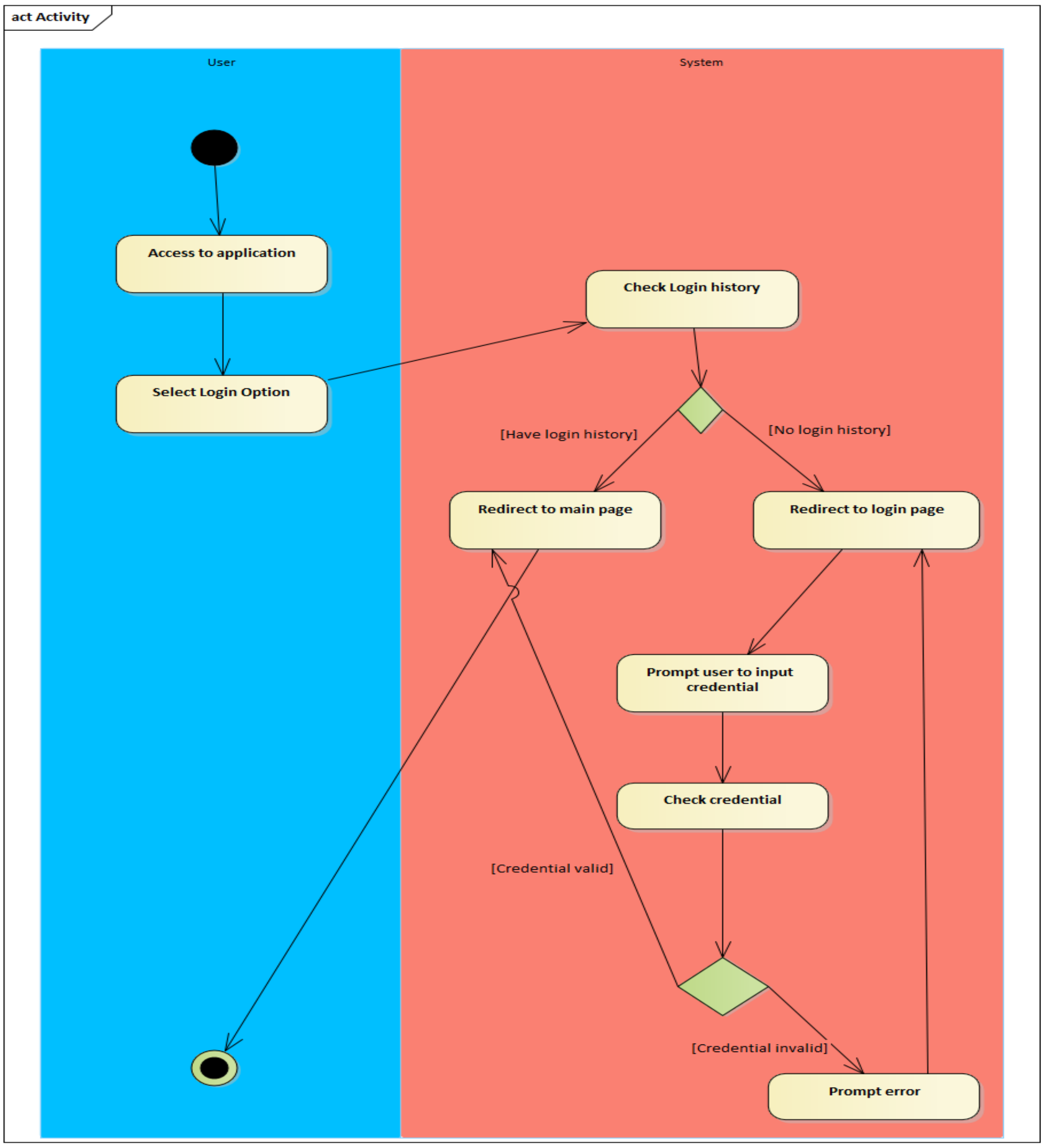


Figure 4.2 Activity diagram of Login

4.5.2 Registration

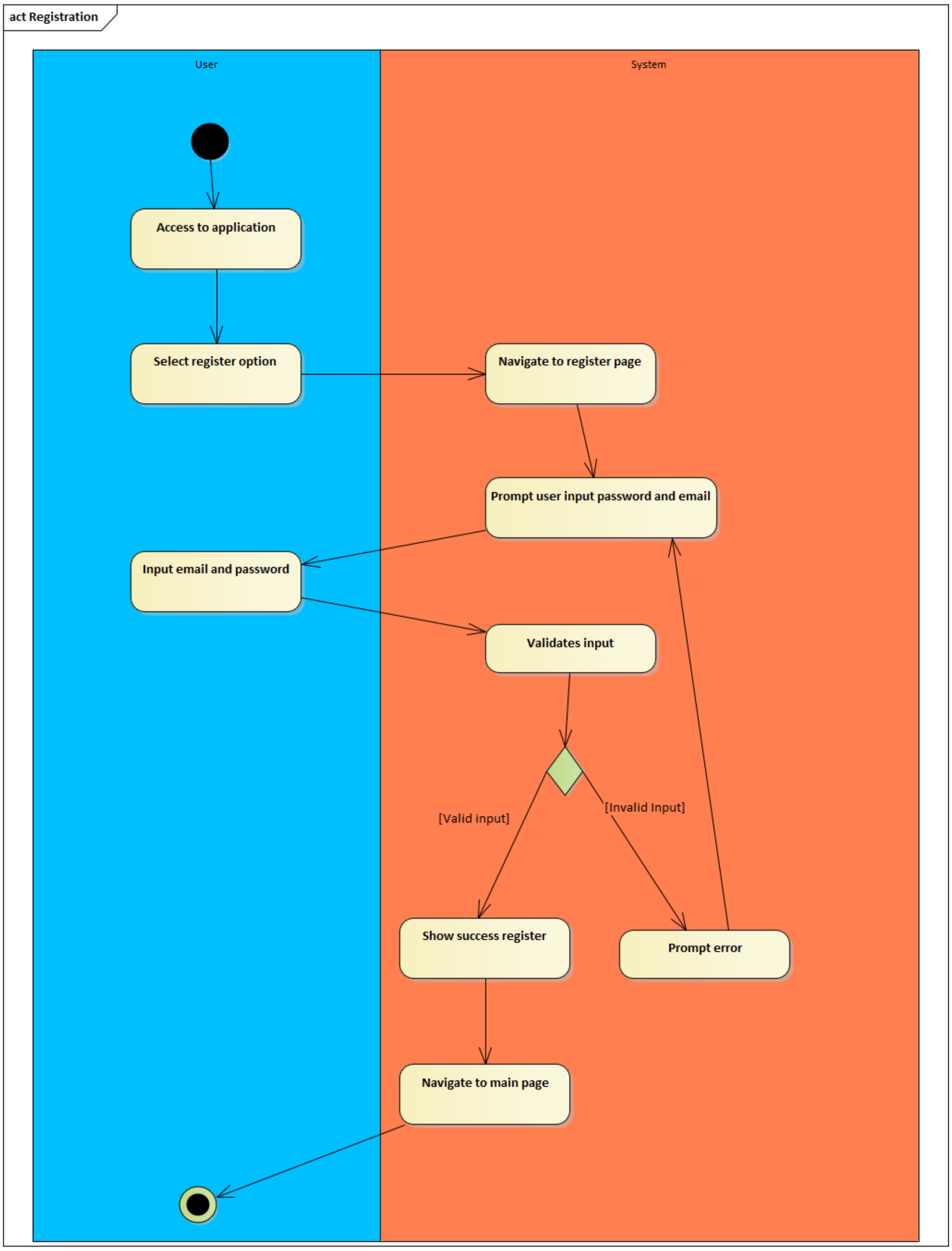


Figure 4.3 Activity diagram of Registration

4.5.3 Country Selection

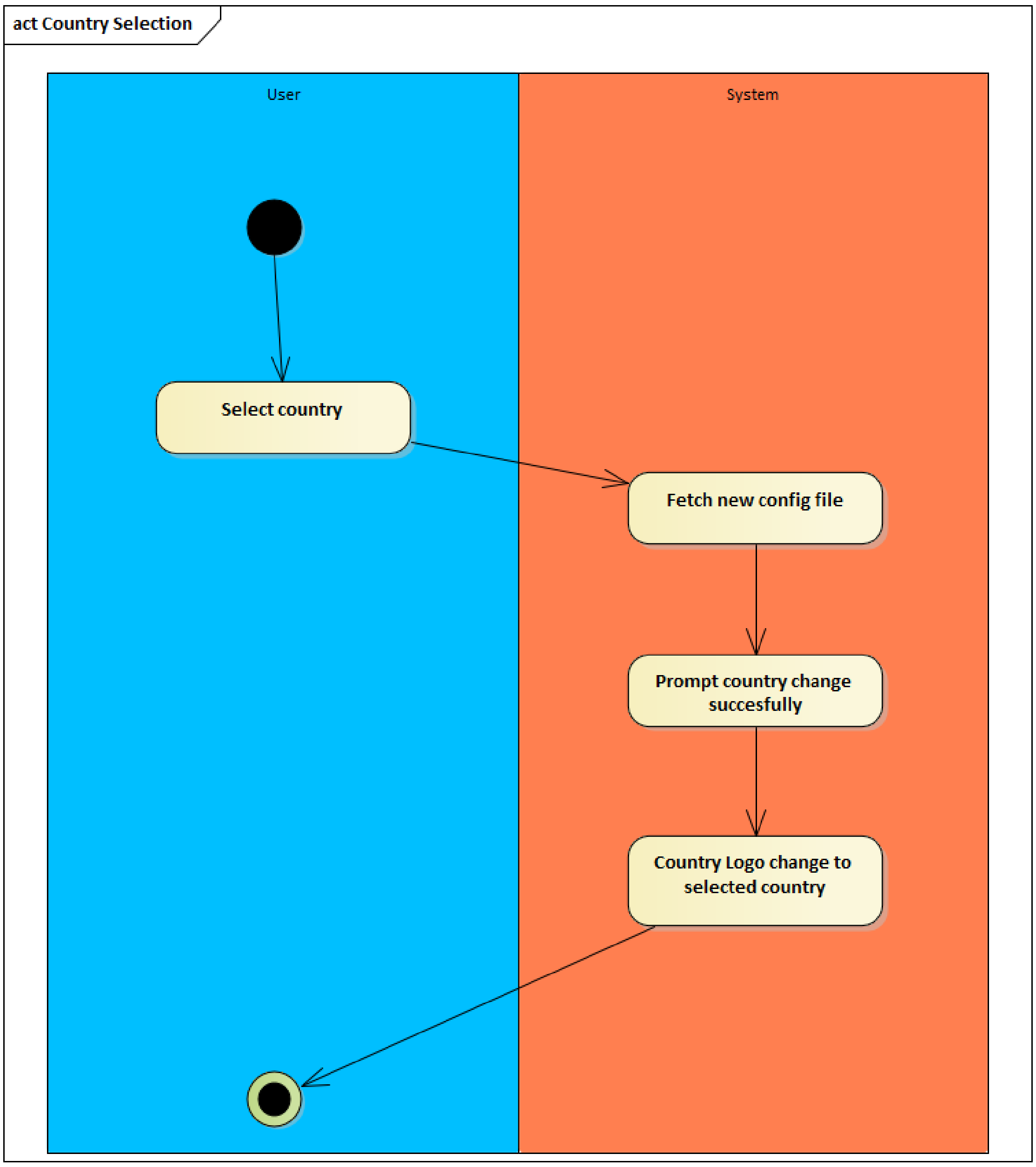

act Country Selection
User
System
Select country
Fetch new config file
Prompt country change succesfully
Country Logo change to selected country

Figure 4.4 Activity diagram of Country Selection

4.5.4 Submit Feedback

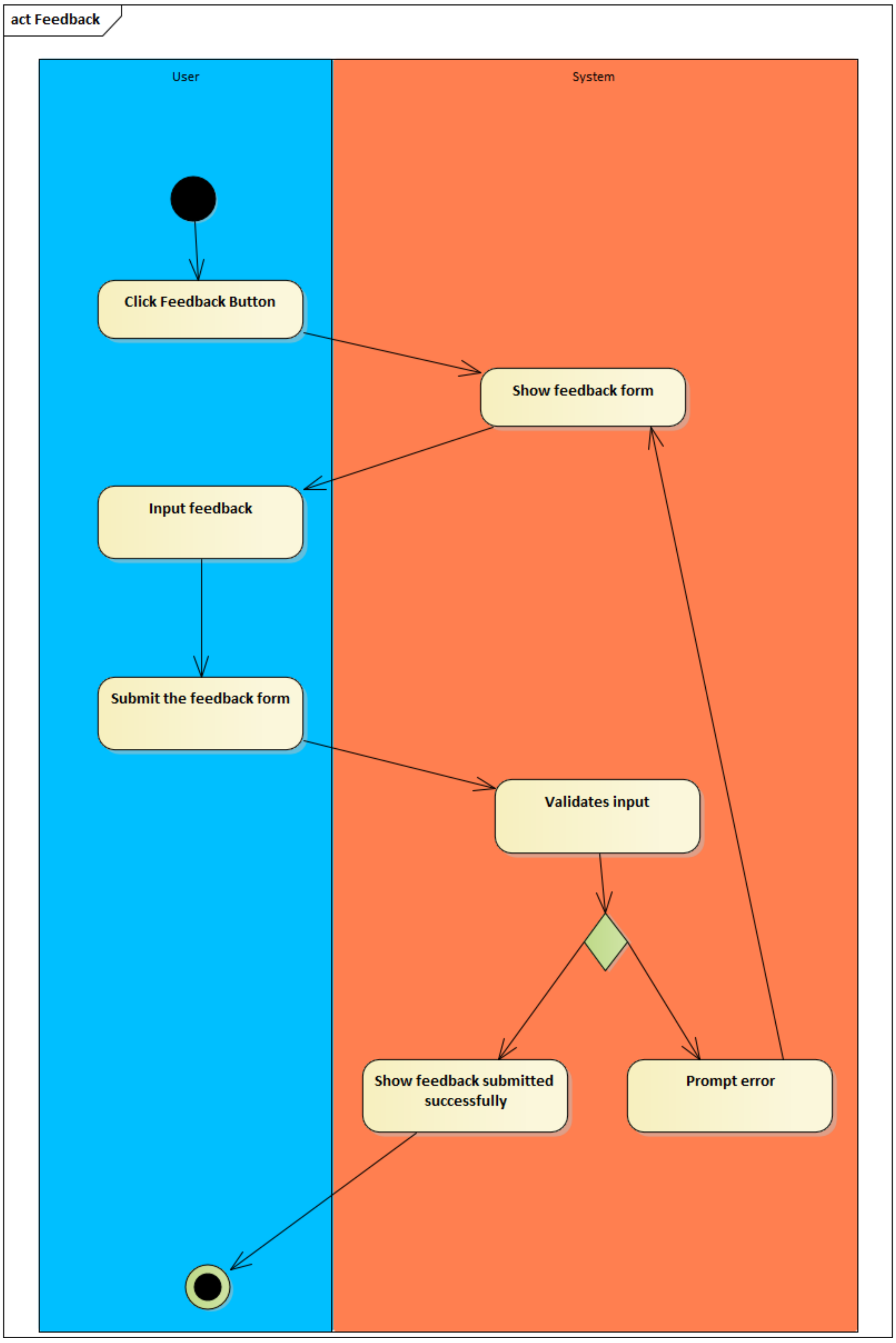


Figure 4.5 Activity diagram of Submit Feedback

4.5.5 Connect to VPN Server

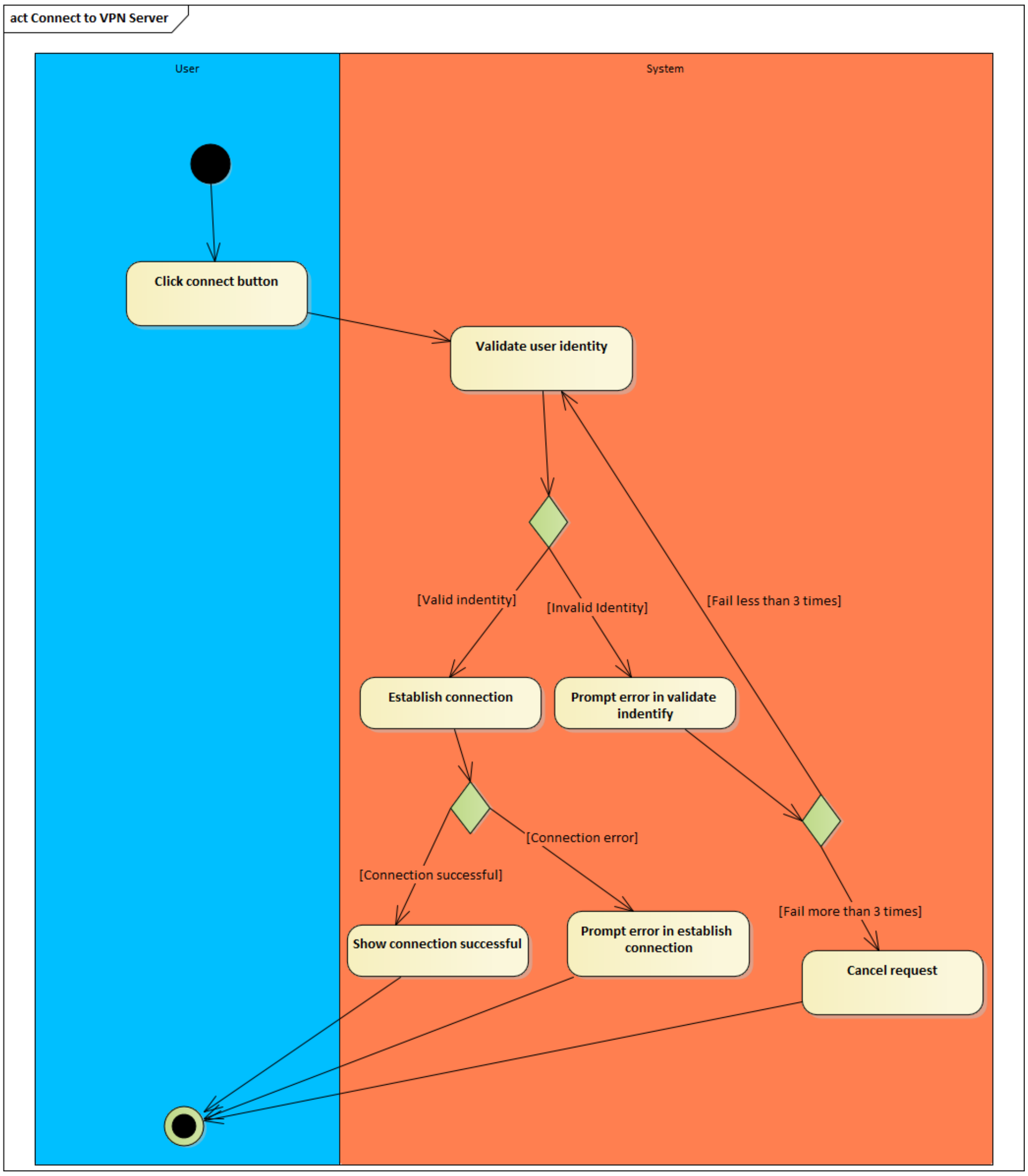


Figure 4.6 Activity diagram of Connect to VPN Server

4.5.6 Fetch Config file

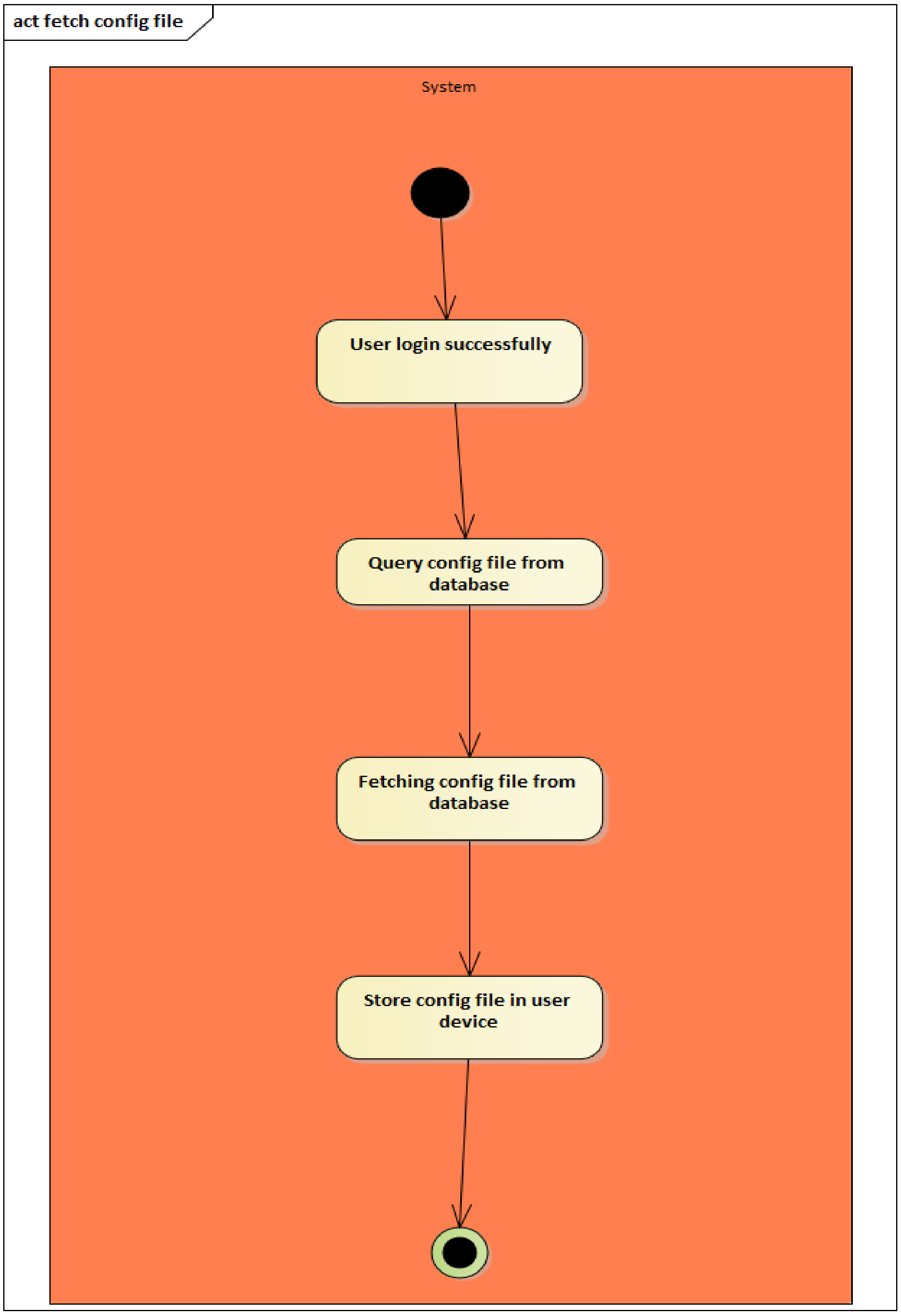

act fetch config file
System
User login successfully
Query config file from database
Fetching config file from database
Store config file in user device

Figure 4.7 Activity diagram of Fetch Config File

4.5.7 Generate Config File

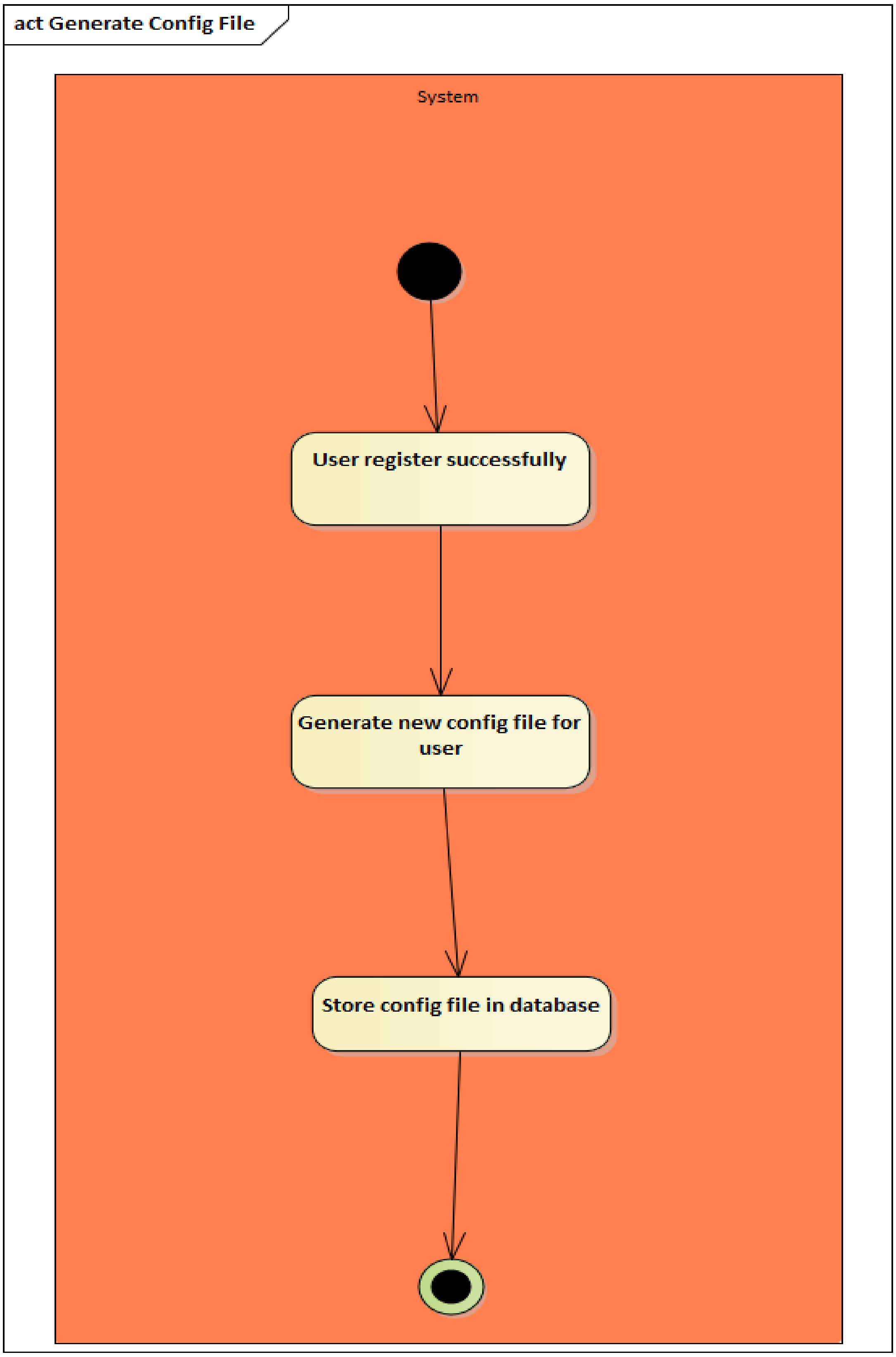

act Generate Config File
System
User register successfully
Generate new config file for user
Store config file in database

Figure 4.8 Activity diagram of Generate Config File

4.5.8 Notification

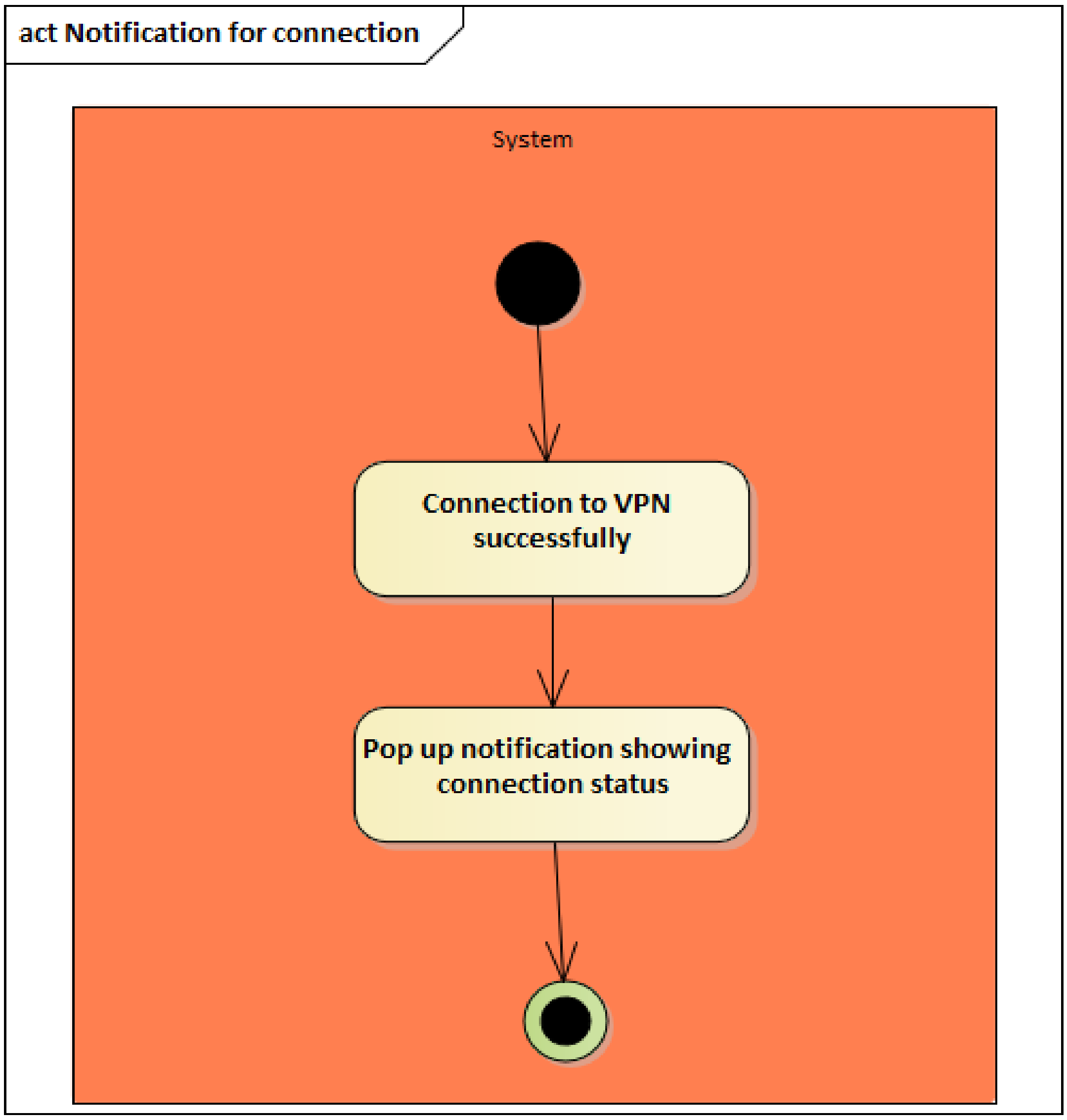

act Notification for connection
System
Connection to VPN successfully
Pop up notification showing connection status

Figure 4.9 Activity diagram of Notification for Connection

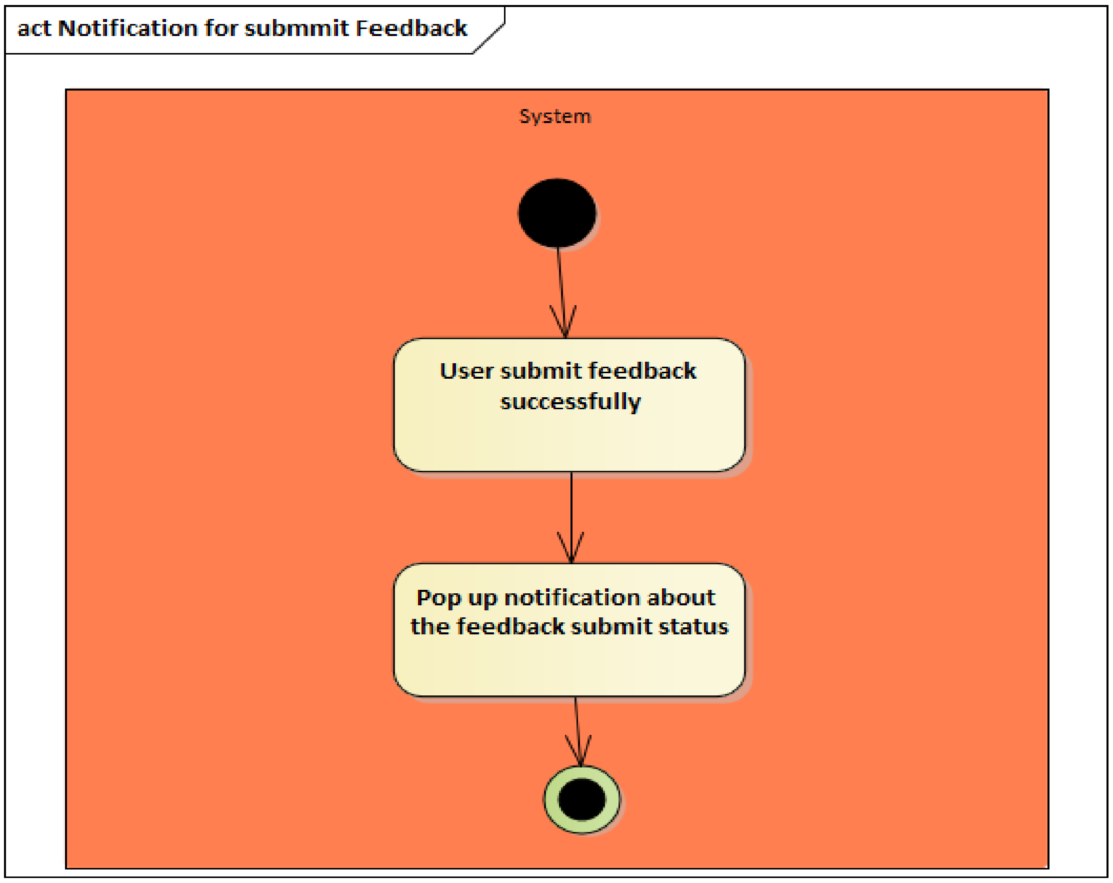

act Notification for submmit Feedback
System
User submit feedback successfully
Pop up notification about the feedback submit status

Figure 4.10 Activity diagram of Notification for Submit Feedback

4.5.9 Verify User

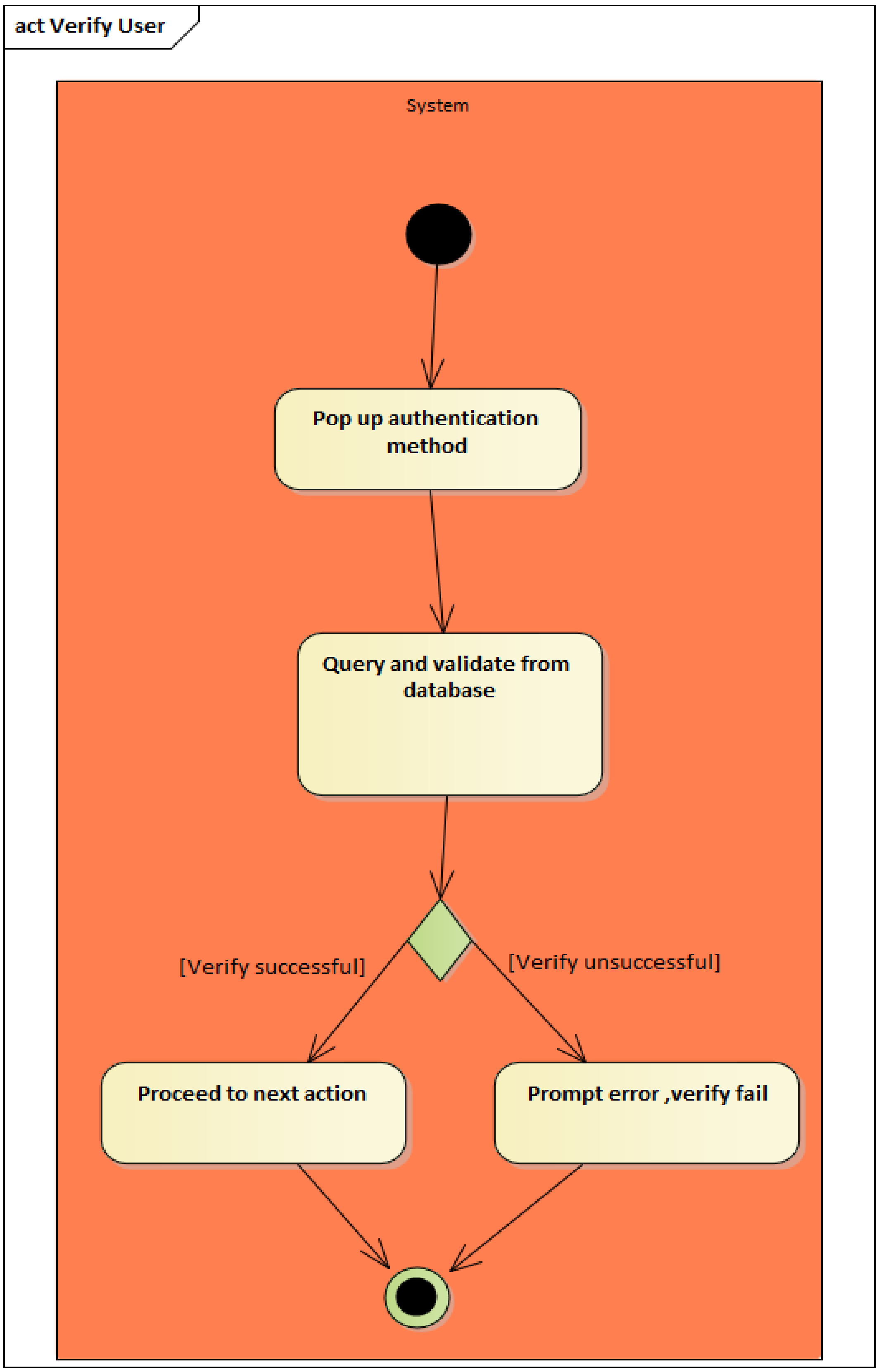

act Verify User
System
Pop up authentication method
Query and validate from database
[Verify successful]
[Verify unsuccessful]
Proceed to next action
Prompt error ,verify fail

Figure 4.11 Activity diagram of Verify User

4.6 Data Flow Diagram

Data flow diagram provided an overview of the data flow within SecureNet VPN application. It shows flow of the data between user and component of the system, such as user information, feedback, configuration file and the selection of country to establishing connection with VPN.

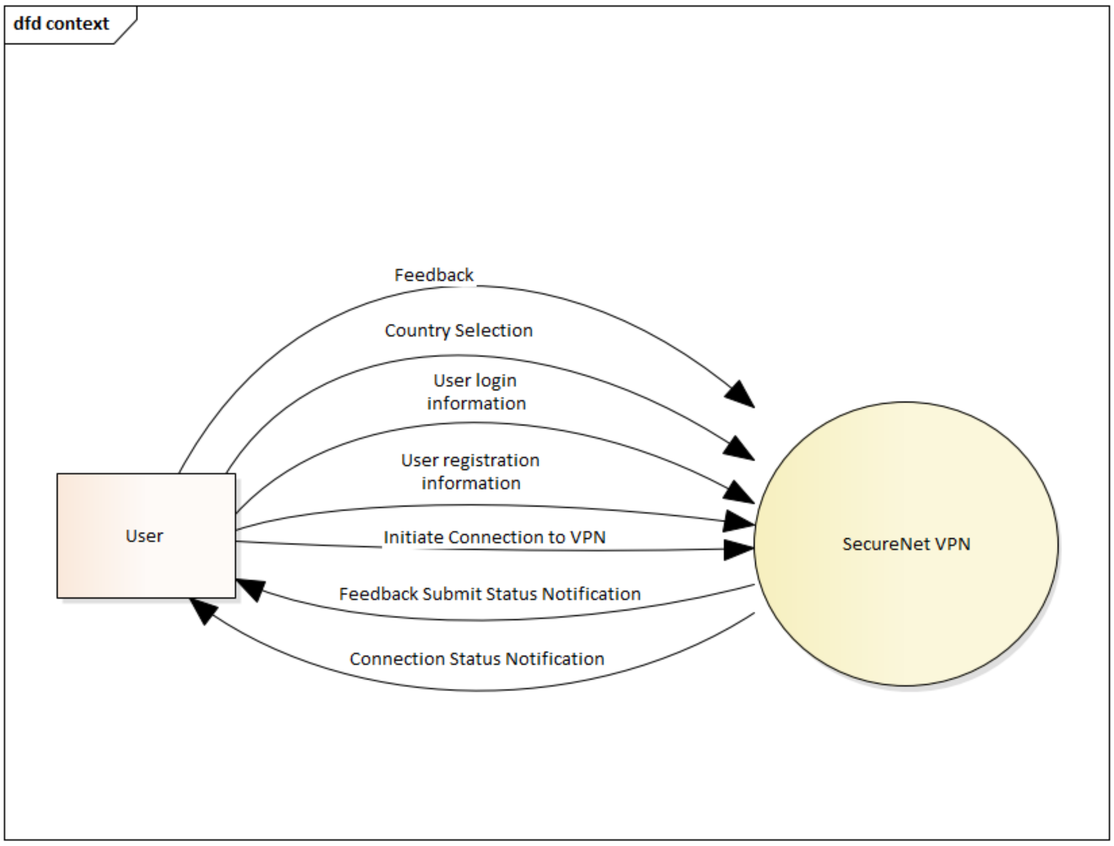

dfd context
User
SecureNet VPN
Feedback
Country Selection
User login information
User registration information
Initiate Connection to VPN
Feedback Submit Status Notification
Connection Status Notification

Figure 4.12 Context diagram of SecureNet VPN

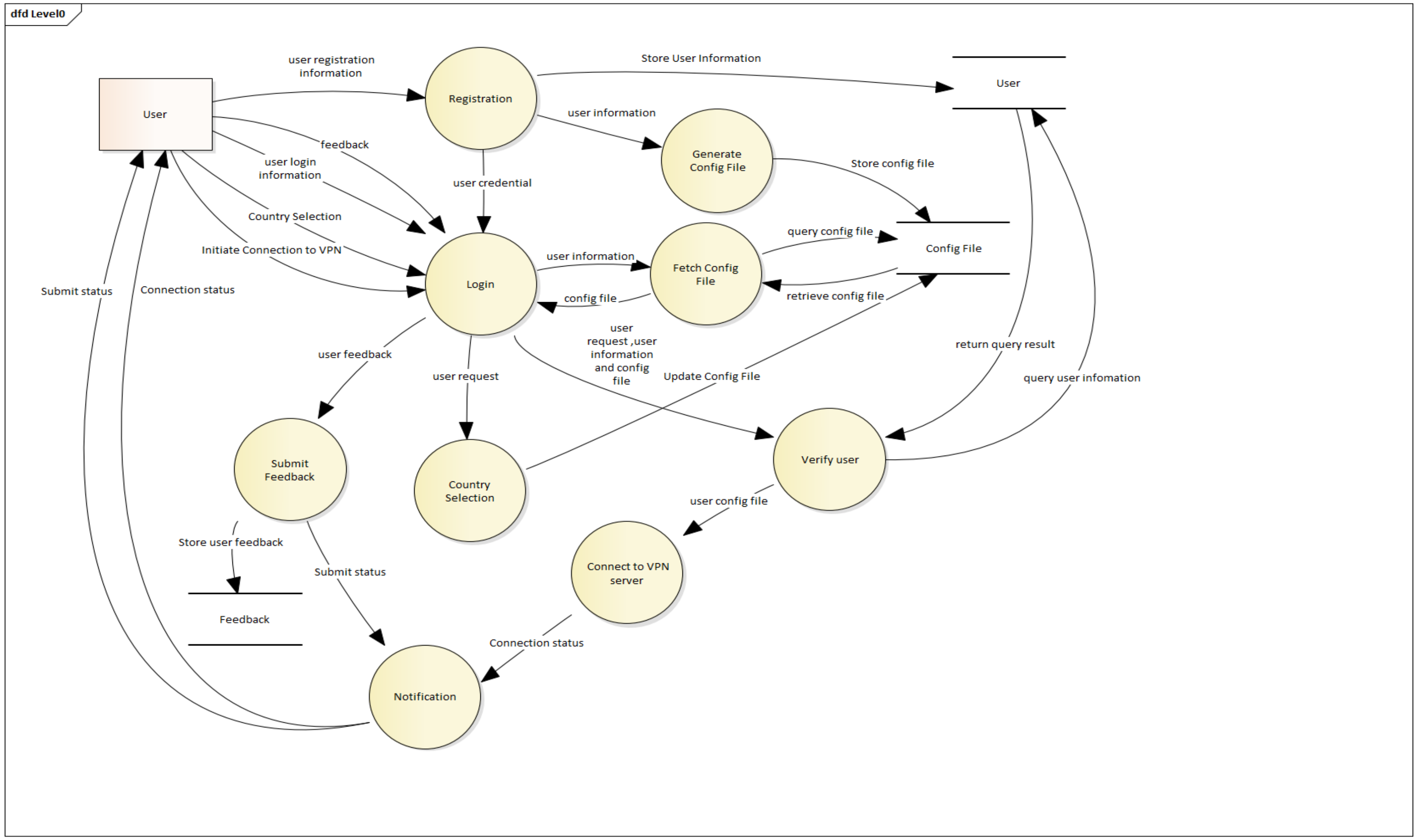


Figure 4.13 Level 0 of SecureNet VPN

Chapter 5

System Design

5.1 Introduction

In this chapter, the system architecture will be discussed, focusing on a high-level overview of the system. Each component of the system will be discussed which include Client Frontend, Centralized backend, VPN server and Security Concern. In this chapter we will able to understand how each component working with each other and showing the behavior and data flow among each component.

5.2 System Architecture Design

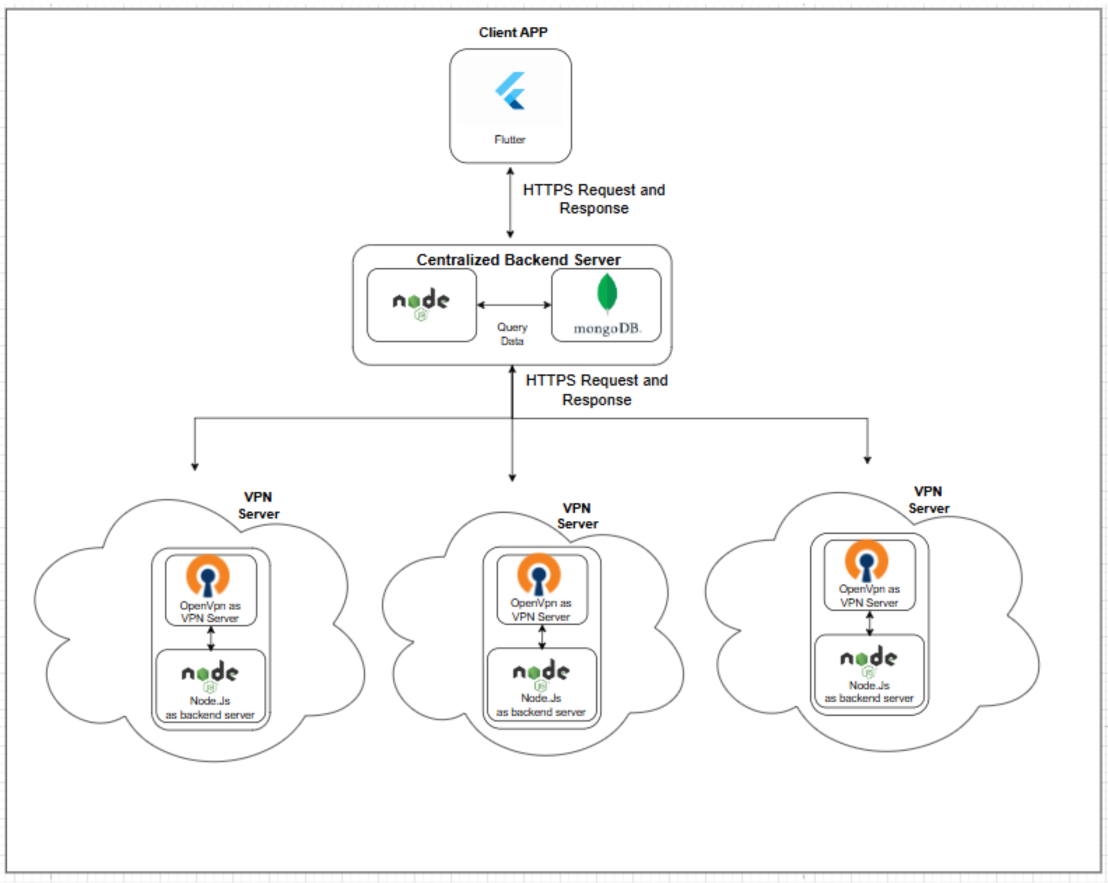

Client APP
Flutter
HTTPS Request and Response
Centralized Backend Server
node
Query Data
mongoDB.
HTTPS Request and Response
VPN Server
OpenVpn as VPN Server
node
Node.Js as backend server
VPN Server
OpenVpn as VPN Server
node
Node.Js as backend server
VPN Server
OpenVpn as VPN Server
node
Node.Js as backend server

Figure 5.1 System Architecture

The system architecture diagram shows a multi-tiered application. Client frontend, a centralized backend and multiple remote servers for VPN services which manage by centralized backend. Here’s the breakdown of each component and their interaction and responsibilities.

5.3 Client Frontend (VPN Client App)

This component represents the frontend of the application which is built by using Flutter framework. It allows users to interact with the centralized backend for certain request. This client app is designed with a user-centric approach. Key features such as clean, easy navigate interface, optimized for various screen sized and performance and the allow user to establish connection to VPN easily.

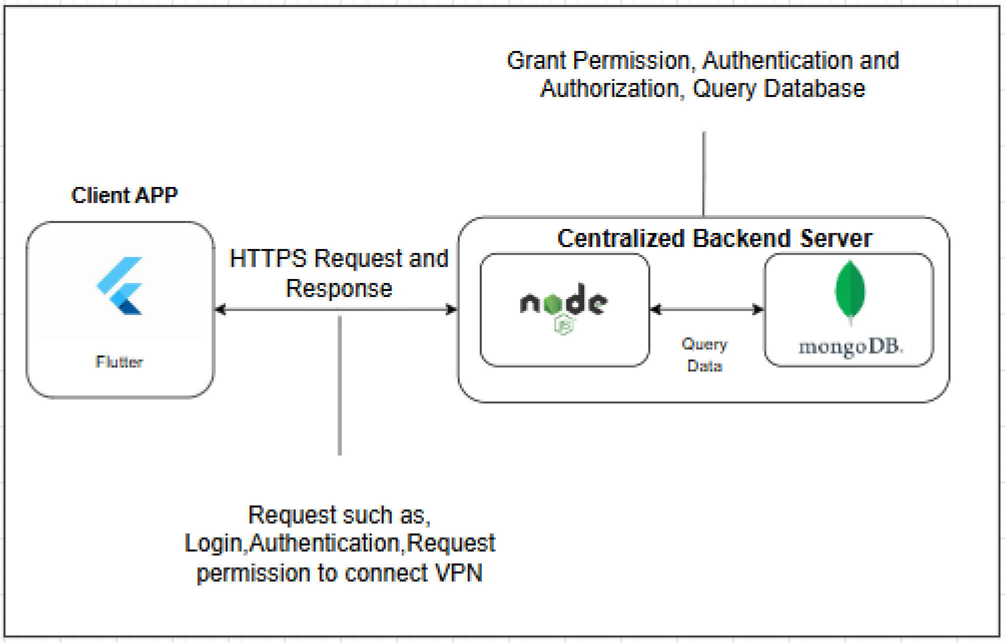

Grant Permission, Authentication and Authorization, Query Database
Client APP
Flutter
HTTPS Request and Response
Centralized Backend Server
node
Query Data
mongoDB.
Request such as, Login,Authentication,Request permission to connect VPN

Figure 5.2 Client Frontend

Client app will interact with centralized backend with HTTPS request such as user management and authentication, discover available VPN services and fetching configuration for establish connection to VPN service. While the centralized backend will authenticate and authorized frontend to validate user credentials and send back appropriate response to frontend such as VPN configuration, available country list and user information.

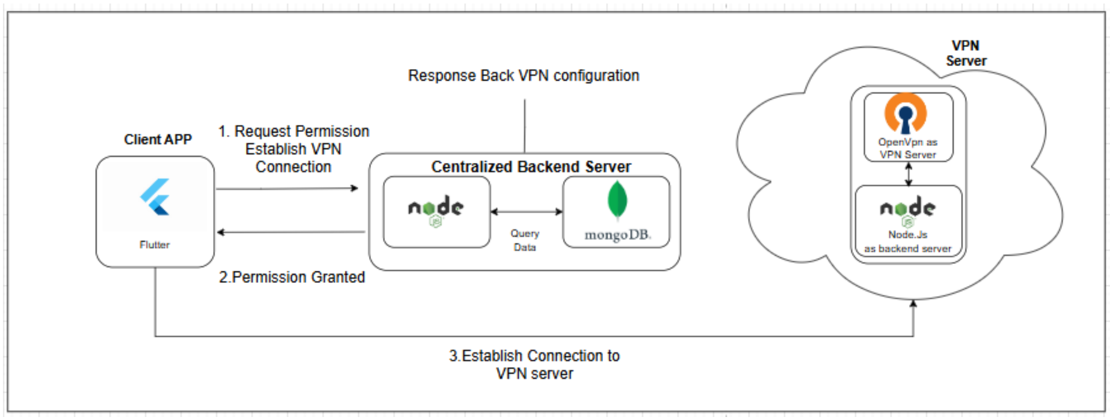

Response Back VPN configuration
Client APP
1. Request Permission Establish VPN Connection
Centralized Backend Server
Flutter
Query Data
mongoDB.
2.Permission Granted
VPN Server
OpenVpn as VPN Server
Node.Js as backend server
3.Establish Connection to VPN server

Figure 5.3 Client Frontend 2

In order to establish connections to VPN server, necessary flow for authentication and authorization been design as image above. Client first needs to request permission from centralized backend and get back response from backend such as “Granted” or “Denied” based on user’s allow usage of the services. If user is denied for connection the client app will show pop up to inform user that they are not allowed to establish connections for certain reason such as credential issue and usage issue. If user is granted to establish connection, the VPN library module which integrated with the client app will establish connections to the VPN server based on the configuration such as IP address, port of the service, protocol, authentication and encryption algorithm and other necessary network setting.

5.4 Centralized Backend

The centralized backend is built by using Nodejs with express as backend and working with MongoDB, the NoSQL for handing request from frontend as we discussed in client frontend such as user login, register, authentication, authorization, response back available VPN server list and VPN configuration for client. All the APIs of the backend are designed as Restful API ensuing efficient and organized communication between client frontend and backend. The centralized backend also has the capability to granted client app to establish connection to the VPN services. The centralized backend manages users group using stateless authentication which implement by using JWT (JSON Web Tokens).

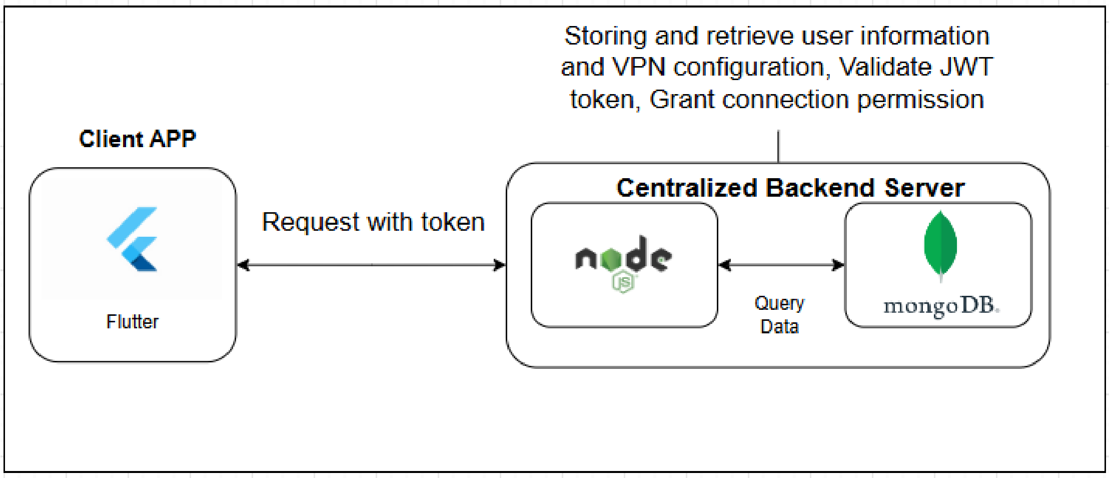

Storing and retrieve user information and VPN configuration, Validate JWT token, Grant connection permission
Client APP
Flutter
Request with token
Centralized Backend Server
Query Data
mongoDB.

Figure 5.4 Centralized Backend 1

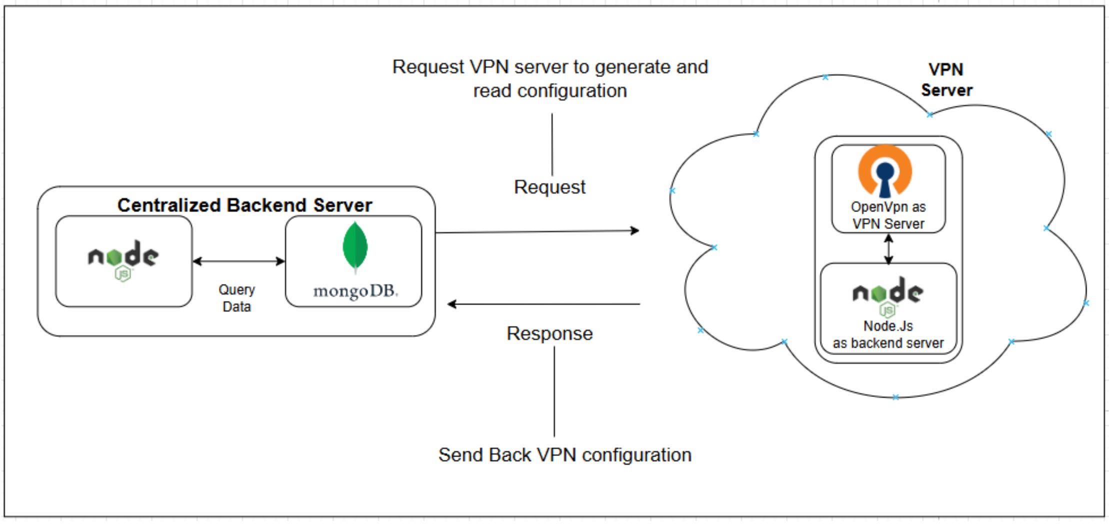

Request VPN server to generate and read configuration
VPN Server
Request
Centralized Backend Server
Query Data
mongoDB.
OpenVpn as VPN Server
Node.Js as backend server
Response
Send Back VPN configuration

Figure 5.5 Centralized Backend 2

Beside handling frontend request, the centralized backend will also manage the remote VPN server such as request VPN server to generate or reading configuration file based on request from client app and then storing configuration from the VPN server into the database for better performance when handling with user request in future. Additionally, the centralized backend implements robust error handling to maintain system stability. By efficiently managing these tasks, the centralized backend able to ensure the system operated smoothly and enhancing overall reliability and performance when handling request.

5.5 VPN Server

The remote VPN server are instance on Amazon Web Service. Each of the instance have 2 important module, OpenVPN library for providing VPN services and a microservices construct with NodeJS and Express framework for listening request from the centralized backend. The responsibilities for instance include providing VPN services connection for client app and listen request from the centralized backend for some certain tasks such as generate new configuration for new client, read and return configuration to the centralized backend.

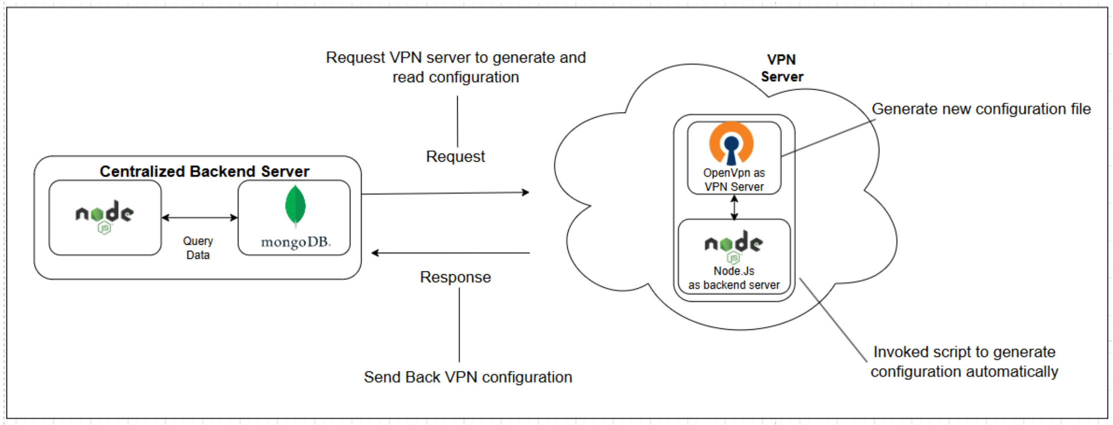

Request VPN server to generate and read configuration
Request
Centralized Backend Server
Query Data
mongoDB.
Response
Send Back VPN configuration
VPN Server
Generate new configuration file
OpenVpn as VPN Server
Node.Js as backend server
Invoked script to generate configuration automatically

Figure 5.6 VPN Server

The microservices within the instance of VPN server, will keep listening request from centralized backend. When centralized backend sends a request to it, it will execute some script to run some system command in order to invoke OpenVPN library to generate new configuration file for client. After VPN library generated the new configuration file. The content of the file is read and response back to centralized backend for further action such as send to client or storing within database. After client able to fetch the configuration file, they will able to establish connection to the VPN server.

5.6 Security Considerations

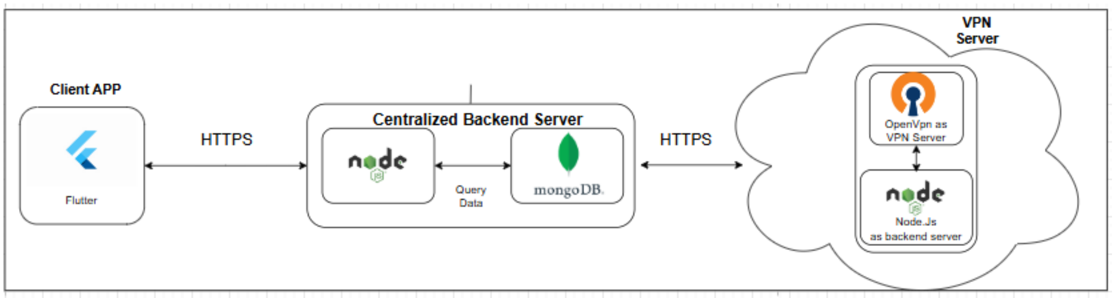


Figure 5.7 Https

All the communication between client frontend, centralized backend, and VPN server (microservices) are based on HTTPS protocol which ensure secure communication between each component which able protect data from being intercepted or tampered with during transit between components.

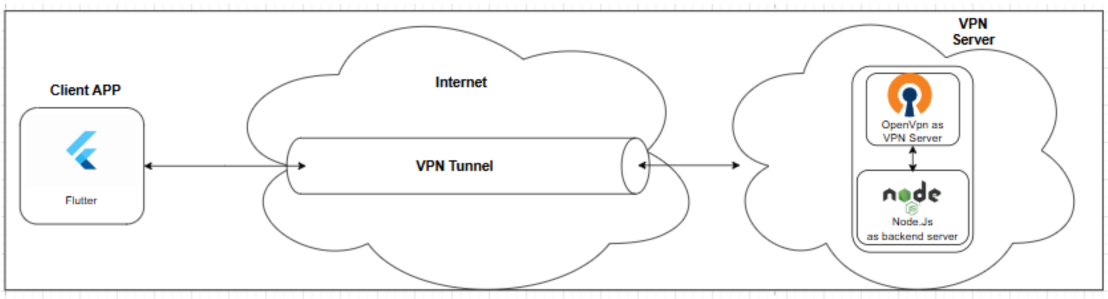


Figure 5.8 Communicate via VPN tunnel

While for the connection between client frontend and the VPN server, all the traffic will be encrypted through the tunnels establish between clients and VPN server. The tunnels create a secure, encrypted path for all traffic between client and the VPN server which ensures any data transmitted through the VPN tunnel. This encryption safeguards sensitive information preventing eavesdropping and man-in-the-middle attacks. This encryption safeguards all communications, enhancing security and privacy by masking users' IP addresses and activities. The direct, secure connection via encrypted VPN tunnels ensures robust and protected communication exclusively between the client frontend and the VPN server.

5.7 Conclusion

In this chapter, we had provided an in-depth overview of our system architecture including its structure and how each component within it interacts with the others. These components included a Client Frontend, a Centralized Backend, and multiple remote VPN servers which collectively ensure that the entire system runs smoothly.

The client frontend built using flutter, offers a user-friendly interface that allows users to easily navigate and interact with the VPN services and it also handles user interaction and communicates with the Centralized Backend via HTTPS for tasks such as authentication, and fetching VPN configurations.

The **Centralized Backend**, developed using Node.js and Express, manages user authentication and authorization, processes requests from the Client Frontend, and coordinates with the VPN servers, and its use MongoDB for efficient data handling and employs JWT for secure stateless authentication. Additionally, it implements robust error handling to ensure system stability and performance.

The **VPN Servers** which hosted on Amazon Web Services, run instances with OpenVPN library and a Node.js-based microservice architecture. It provide VPN connectivity and respond to configuration requests from the Centralized Backend. The VPN servers generate and manage VPN configurations, which are then communicated back to the backend and subsequently to the client.

By integrating these components, the system ensures a secure, reliable, and efficient VPN service that meets user needs while maintaining high standards of security and performance. The clear separation of concerns and the use of secure communication protocols enhance the overall robustness of the system, providing a seamless and protected user experience.

Chapter 6

IMPLEMENTATION

6.1 Introduction

This chapter outlines implementation of SecureNet VPN application, which involved, frontend, centralized backend, VPN Server, microservices on VPN server and security configuration. Through this chapter all the implementation of important module and consideration will be discussed and clarified.

6.2 Frontend (VPN Client Application)

Our client frontend is built by Flutter framework. In this section, we will discuss the important component which contribute heavily to this VPN application.Before development of each component, we need to undergo a series of initialization setting of the flutter project, which include

1. Initialization of flutter project
2. Design of user interface
3. Installation of required fonts
4. Preparation of necessary asset such as icons or images for UI design
5. Creation of independent component which avoid code redundancy and improve maintainability
6. Define application navigation and routes to different section of the application

Next, we will discuss about the important module or library that contribute heavily to the frontend application.

6.2.1 Input Validation

Input Validation ensures that the data entered by the user meets certain criteria before we processed the data. This is crucial for security and data integrity, and it also reduces the stress of backend when processing the data. We utilize the “Form” widget in flutter along with the “TextFormField” and the validation functions which natively provided by the “Form” widget to achieve validation functions for common requirements like email and password validation.

6.2.2 JSON Web Token

The main purpose for the JWT is for stateless authentication and information exchange. JWT allows the application authenticate users without maintain session data on server. In our application, user will receive a JWT after login which able use to prove their identity on subsequent request, which achieving a stateless architecture.

JWT also able to securely transmit information between frontend and backend. Backend able to inject payload in the token to helping frontend for displaying important information to the users and allow frontend to execute certain logic based on the information in the payload.

6.2.3 Secure Storage

For sensitive information such as the JSON Web Token, OpenVPN configuration and user information, we will be storing it using secure storage for secure and persistent storage. Flutter Secure Storage is a storage solution specifically designed to handle sensitive data securely, it able encrypt the data storing within it with using encryption protocol such as AES encryption which providing an additional layer of security.

By using secure Storage, we can safely store critical information like JWT which used for authenticating and authorizing user session, as well as

OpenVPN configurations that is crucial for establish VPN connections. For user information, which include personal and sensitive information will also be protected using secure storage.

6.2.4 Background service

In order to enhance functionality and user experience of the application, we will integrate background services for handling specific task. The background services will either run in background or foreground based on different condition and usage of the application. The use case of background services in our application is for monitor user’s connections time to the VPN services. It able to control the connections of the VPN services based on the usage quota allows for users, if typical users had exceeded their allows quota, for example exceed predefined allow connection time. The background services will turn off the connections automatically. The background services will also run-in foreground in form of notification banner when user is connected to the VPN services to display necessary information and notify user that VPN services is currently connected on the mobile.

6.2.5 OpenVPN Library

The OpenVPN module is the most important component in the frontend application. It provides ability to establish connection to VPN services which based on OpenVPN protocol. This library is built on top of 2 popular open-source project on GitHub, “ICS-OpenVPN” and “OpenVPN-Flutter”. The ICS-OpenVPN is the native module allowing android establish connection without root access to OpenVPN services. This repository had contributed heavily to many OpenVPN projects, while the OpenVPN-flutter is another repository build on top of the ICS-OpenVPN core module which bridging the native code of ICS-OpenVPN to compatible with Flutter environment.

The library able to handles various aspect of VPN connectivity which including supports multiple methods such as username/password and certificate-based authentication. It also utilized strong encryption protocols to ensure secure data transmission and manages the VPN connection lifecycle including establishment, maintenance, and termination of the connections.

Although these 2 libraries are considered popular library, but for OpenVPN-flutter, since this library is still new and undergo maintenance, it still has many issues and problems need pay attention to. In our application, we had made some modifications and customization to suite the library to meet our own requirements. The original capability of the OpenVPN-Flutter library was to read configuration files. We have enhanced it to read configuration data directly from variables, providing greater flexibility and integration with dynamic configuration sources.

By utilize these 2 libraries we able to ensures robust, secure, and user-friendly VPN connectivity, enhancing the privacy and security of user data.

6.2.6 Error Handling

Effective error handling is essential for ensuring smooth and reliable user experience in our application, especially when our application needs dealing with various backend services such as login, registration, fetching available country list and VPN connections.

6.2.7 VPN Connection Errors

In our application, handling VPN-specific errors involves managing various connection states and failures

1. **Connection Status**

   The application will display the connections status during or after establishing connection to VPN services

2. **Connection Error**

   The application will display error message if any occurs error and will prompt user to retry again or try again later

3. **Connection Timeout**

   The application will display error message indicate there is connection timeout issue and will prompt user to retry again or try again later

4. **Connection Drops**

   The application will display error message indicate connection drops and will automatically retry to reconnect to the VPN services and update connections status.

6.2.8 HTTP Request Errors

Handling HTTP request errors involves ensuring that users are promptly informed of any issues with their requests. When an error occurs, the frontend will display clear and user-friendly messages, such as notifying users of network connectivity issues, server errors, or request timeouts. This approach helps users understand the nature of the problem and provides guidance on what actions they might need to take, such as checking their internet connection or retrying the request.

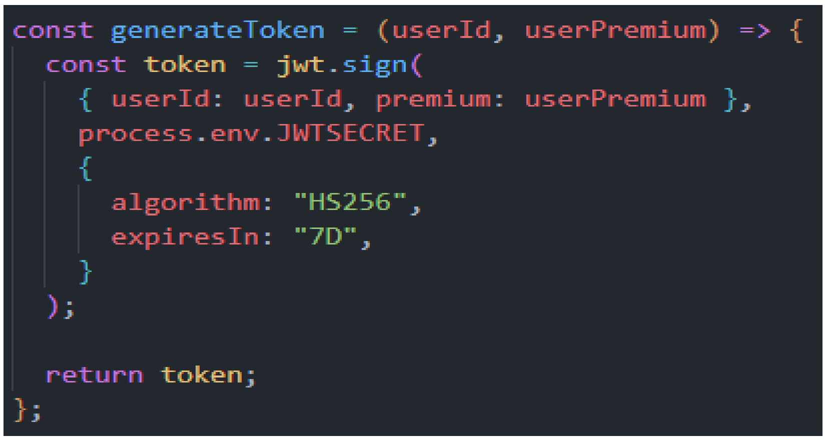
6.3 Centralized Backend

The centralized backend is built by using Node.js and express.js framework to implement backend functionality because of their efficiency, scalability, and ease of use in handling asynchronous operations, making them ideal for building fast and responsive web applications.

6.3.1 Authentication

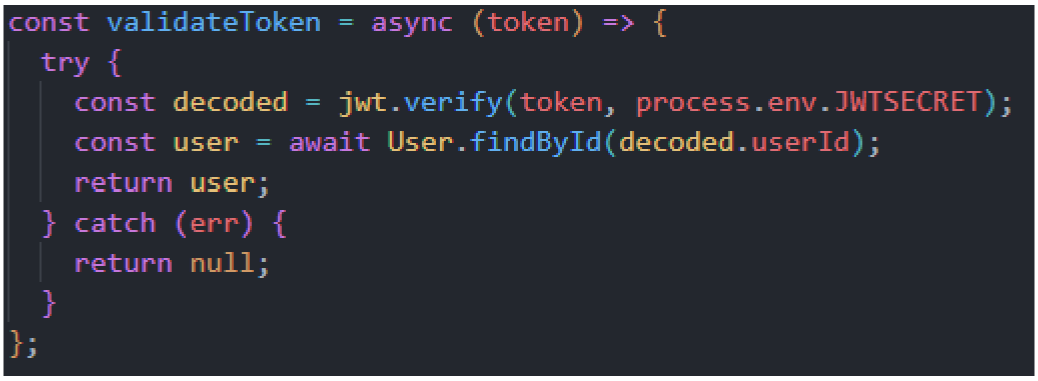
Authentication is a critical component in securing the centralized backend. It ensures that only authorized users can access the system and resources. The major usage of centralized backend is handled user management such as registration, login, and access control.

In our application, when a user success login themselves, the backend will send back a JSON Web Token to frontend, with information such as user information in payload by using sign() method in jsonwebtoken library. This token is set to expired in 7 days, during the period, user able to send request that required authentication to backend to identify themselves and get authorized from backend for example action such as reset password, request permission for VPN connection and fetch VPN configuration data. The backend will use verify() method in jsonwebtoken library to verify the validity of the token with the secret key which use to sign the token. For each API endpoint which required validation of token will pass through a middleware which responsible for validating the token.



```
const generateToken = (userId, userPremium) => {
  const token = jwt.sign(
    { userId: userId, premium: userPremium },
    process.env.JWTSECRET,
    {
      algorithm: "HS256",
      expiresIn: "7D",
    }
  );

  return token;
};
```

Figure 6.1 JSON Web Token

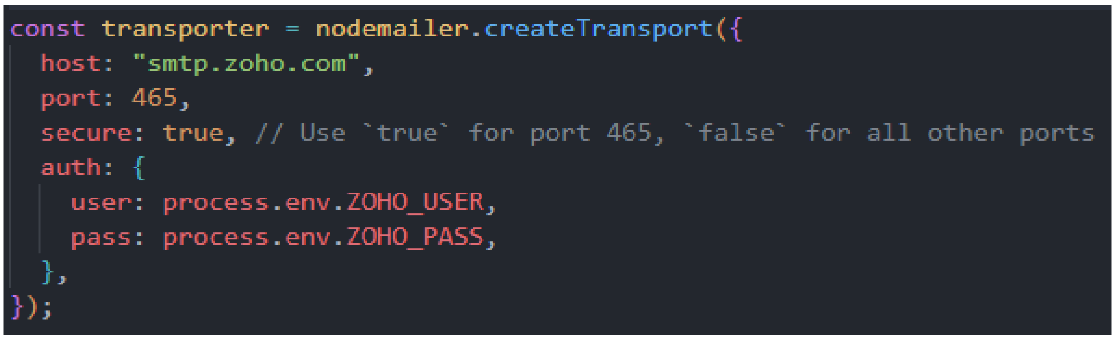


```
const validateToken = async (token) => {
  try {
    const decoded = jwt.verify(token, process.env.JWTSECRET);
    const user = await User.findById(decoded.userId);
    return user;
  } catch (err) {
    return null;
  }
};
```

Figure 6.2 Token Validation

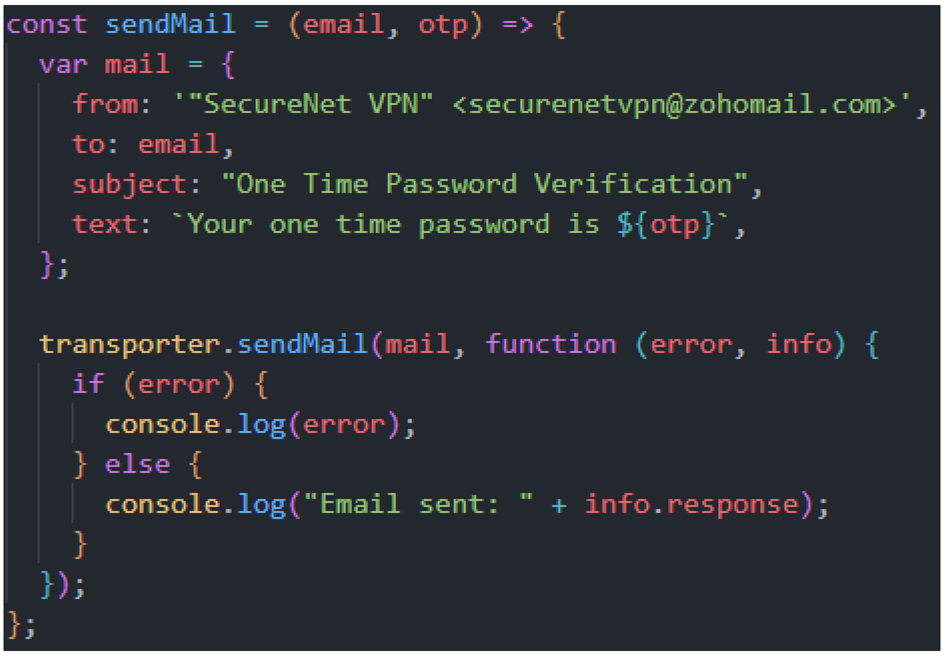
For certain request such as registration and reset password, the application will have a 2FA authentication before executing the request. The first authentication is through the email and password provided by user while second factor will through email One Time Password (OTP) to authenticate themselves. The one time password will send to user provided email, and with the help of nodemailer ,our backend can send email and the one time password to user’s email and user need to input the one time password at the frontend application.

In our application we use email provided Zoho with nodemailer for sending email. The reason of using Zoho is due to Zoho had provided offer for creating email with their domain name and its easy configuration which provided clear documentation and support for SMTP and IMAP configuration making it easy to integrate with nodemailer for sending email programmatically beside that nodemailer documentation also recommend users to use other mail services rather than Gmail. While Gmail might face some restriction due to its own policy issue.



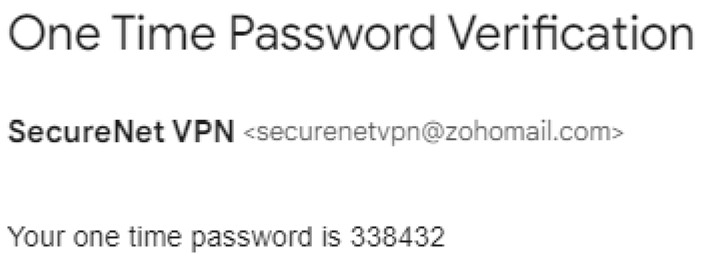
```
const transporter = nodemailer.createTransport({
  host: "smtp.zoho.com",
  port: 465,
  secure: true, // Use `true` for port 465, `false` for all other ports
  auth: {
    user: process.env.ZOHO_USER,
    pass: process.env.ZOHO_PASS,
  },
});
```

Figure 6.3 Mail transporter



```
const sendMail = (email, otp) => {
  var mail = {
    from: '"SecureNet VPN" <securenetvpn@zohomail.com>',
    to: email,
    subject: "One Time Password Verification",
    text: `Your one time password is ${otp}`,
  };

  transporter.sendMail(mail, function (error, info) {
    if (error) {
      console.log(error);
    } else {
      console.log("Email sent: " + info.response);
    }
  });
};
```

Figure 6.4 Send Main Function



One Time Password Verification

SecureNet VPN <securenetvpn@zohomail.com>

Your one time password is 338432

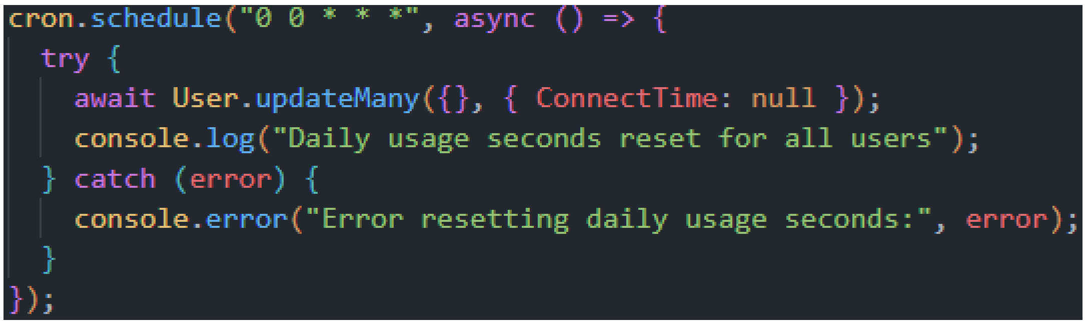
Figure 6.5 One Time Password Email

6.3.2 Usage Tracking

The is an endpoint in our backend which will trigger whenever user try to establish connection to VPN services, this endpoint helps track how long users are using the service each day. When a user connects, the frontend application will provide a time indicating when they are disconnected. The system uses this information to calculate how much time they spent online. If it's the user's first connection of the day, the system starts tracking from that point. As the user continues to connect and disconnect, the system keeps adding up their usage time. Non-premium users have a daily limit on how long they can use the service, while premium users can use it without any limits. The response from the backend will also tell the user if they can continue using the service and how much time they have left for the day.

6.3.3 Schedule

The backend has a daily schedule for reset daily usage for free tier user which they only allow to establish connection to VPN services one hour every day. **Node Cron** module is a tiny task scheduler in pure JavaScript for Node.js. This module allows us to schedule task in Node.js environment. In our application Cron are used for reset daily usage so free tier usage can enjoy the services every day.



```
cron.schedule("0 0 * * *", async () => {
  try {
    await User.updateMany({}, { ConnectTime: null });
    console.log("Daily usage seconds reset for all users");
  } catch (error) {
    console.error("Error resetting daily usage seconds:", error);
  }
});
```

Figure 6.6 Reset Task

6.3.4 Database

Our backend system works with MongoDB, which is a powerful NoSQL database that can handle wide range of data structure and scales well for many applications. In our application we had define multiple Schema of the collections in database with the help of using Mongoose. Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, providing a straightforward, schema-based solution to model our application data.

1. OTP Collection

   This Collection storing the One Time Password which send to user's email for two factor authentication. When user try to validate the OTP, the application will retrieve the OTP from this collection. This collection had been set an expired time of 5 minutes which the record will be remove from the collection after expired.

Table 6.1 OTP Collection

Column	**Type**	**Required**	**Default**	**Expired**

_id	String	True	Null	N/A
otp	String	True	Null	N/A
createdAt	Date	False	DateNow()	5 minutes

2. Temporary User Collection

This Collection will be storing user's credential such as email and password temporary. The purpose for this collection is working with the OTP Collection, for storing user information temporary before they successfully validate the One Time Password. In Our application, for action such as registration, reset password, user need input email and password and send to backend, after backend received it will generate an One Time Password, then will store user credential in this collection and the One Time Password in OTP Collection. Temporarily storing user credentials and OTPs in separate collections able ensures security and data integrity by validating user actions like registration and password resets before permanently saving their information.

Table 6.2 Temporary User Collection

Column	**Type**	**Required**	**Default**
_id	String	True	Null
pass	String	True	Null
createdAt	Date	False	DateNow()

3. User Collection

This collection will store user information permanently after user successfully pass the verification of the One Time Password. This collection had the record of user type which either premium or non-premium, and the record of their connection time and usage. These record majorly use for tracing user's VPN connections usages and allow or restrict them based on the record.

Table 6.3 User Collection

Column	**Type**	**Required**	**Default**
_id	String	True	Null
pass	String	True	Null
premium	Boolean	False	False
dailyUsage	Number	False	0
ConnectTime	Date	False	Null
DisconnectTime	Date	False	Null
createdAt	Date	False	DateNow()

4. VPN Configuration Collection

This collection stores the VPN configuration of each user for each country. When typical user send request to get a configuration for the VPN. The backend will first check this collection, if not exist in this collection, backend will send request to microservice on VPN server to generate new configuration for the user and then store in this collection. If configuration exist, backend will direct retrieve the record and send to user for better performance instead fetch from VPN server again.

Table 6.4 VPN Configuration Collection

Column	**Type**	**Required**	**Default**
user	String	True	Null
country	String	True	Null
config	String	True	Null
createdAt	Date	False	DateNow()

6.3.5 API List

Table 6.5 shows the breakpoints, HTTP methods, parameters and description of the different routes of backend.

Table 6.5 API List

Endpoint	**HTTP method**	**Parameter**	**Description**
/register	POST	{email,password}	Store user credential and One Time Password temporary and send One Time Password to user's email.
/validate	POST	{email,otp}	Validate the One Time Password, and store user credential as permanent if success.
/login	POST	{email,password}	Endpoint for user login, checking credential and compare from database.
/vpnStatus	GET	-	Endpoint for fetching available VPN country list and latency.
/vpnConfig	GET	-	Endpoint for request VPN configuration.
/validateM	GET	-	Validate email exist in database, and initiate reset password flow.
/validateO	POST	{email,otp}	Validate One Time password for reset password scenario
/resetPass	POST	{email,password	Endpoint for reset password, and update the record.
/resendO	GET	-	Resend One Time Password

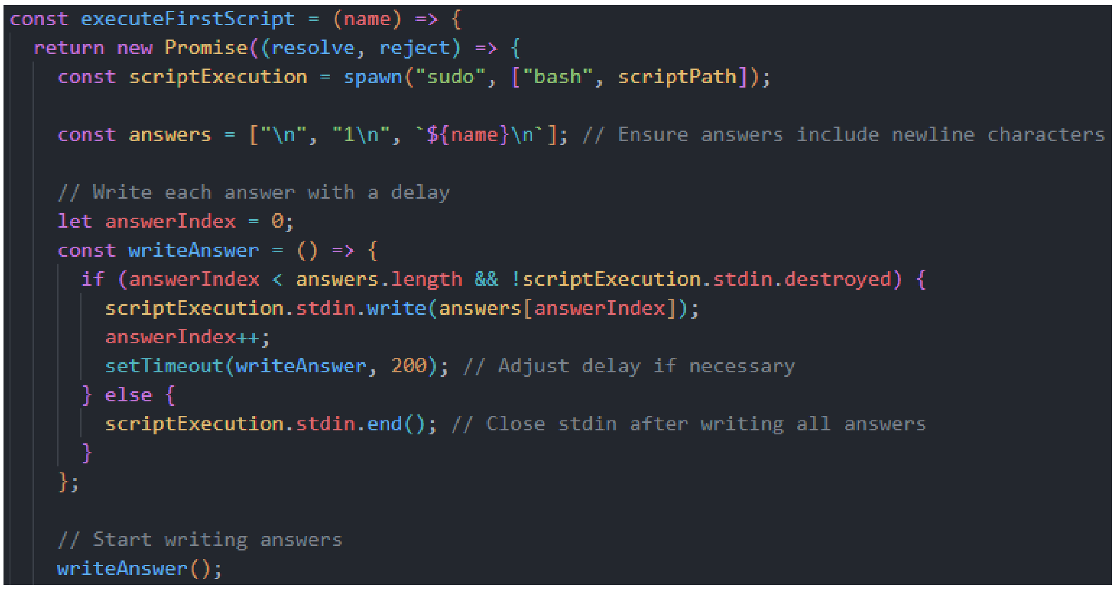
/goPremium	GET	-	Endpoint for upgrade user to premium status.
/connect	POST	{token,disconnectTime}	Endpoint use for checking user status and allow usage to determine granted or denied connect to VPN serives.

6.4 Microservice on VPN Server

A microservices built with Node.js and express.js is deploy with each VPN server. The purpose of the microservices is to listen request from centralized backend, generate VPN configurations and read the configuration.

6.4.1 Generate Configuration File

The microservice have the ability to generate VPN configurations by using the **child-process** module from Node.js. This module able to invoke the Linux command to read the autogenerate script to generate configuration and passing parameter for necessary modifications on the configuration.

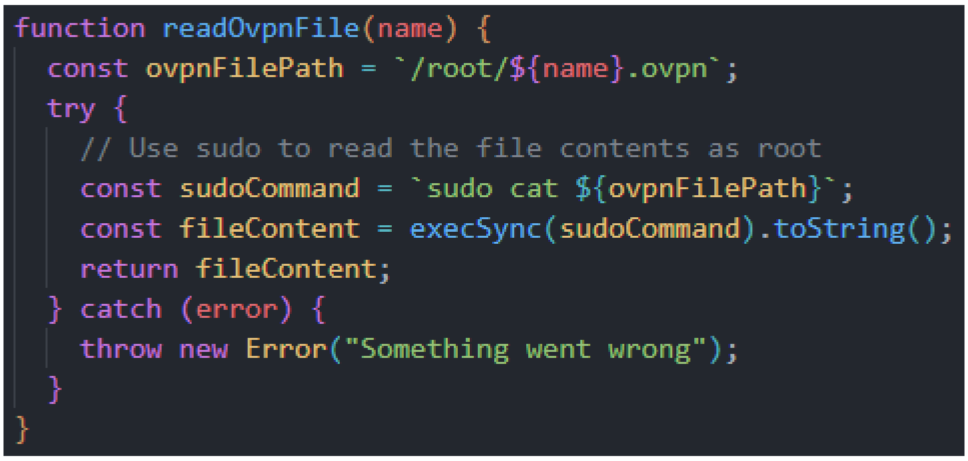


```
const executeFirstScript = (name) => {
  return new Promise((resolve, reject) => {
    const scriptExecution = spawn("sudo", ["bash", scriptPath]);

    const answers = ["\n", "1\n", `${name}\n`]; // Ensure answers include newline characters

    // Write each answer with a delay
    let answerIndex = 0;
    const writeAnswer = () => {
      if (answerIndex < answers.length && !scriptExecution.stdin.destroyed) {
        scriptExecution.stdin.write(answers[answerIndex]);
        answerIndex++;
        setTimeout(writeAnswer, 200); // Adjust delay if necessary
      } else {
        scriptExecution.stdin.end(); // Close stdin after writing all answers
      }
    };

    // Start writing answers
    writeAnswer();
```

Figure 6.7 Generate Configuration File

6.4.2 Read Configuration File

Child-process module from Node.js provide ability to use Linux command. After generate configuration ,it able to read the file through Linux command and return the result as string for further processing.



```
function readOvpnFile(name) {
  const ovpnFilePath = `/root/${name}.ovpn`;
  try {
    // Use sudo to read the file contents as root
    const sudoCommand = `sudo cat ${ovpnFilePath}`;
    const fileContent = execSync(sudoCommand).toString();
    return fileContent;
  } catch (error) {
    throw new Error("Something went wrong");
  }
}
```

Figure 6.8 Read Configuration File

6.4.3 API List

The microservice only have 2 endpoints since it designed for generate and read configuration file. Below is the API list of microservice.

Table 6.6 Microservices API List

Endpoint	**HTTP method**	**Parameter**	**Description**
/getConfig	POST	{clientName}	Invoked system to run script to generate VPN configuration and return the configuration
/readConfig	GET	-	Read configuration and return the configuration

6.5 VPN Server

The setup of the VPN Server will be discussed in this section. Our VPN server hosted on EC2 Amazon Web Service for public access. The preparation, installation and configuration step and details will be explained in this section.

6.5.1 Preparation for Installation

OpenVPN library are the main component for providing VPN services in this project. Ubuntu LTS 22, Amazon Web Services and the SSH client MobaXterm will be the use in this project.Prerequisite for configure OpenVPN are as below:

1. A remote instance with Linux as operating system.
2. Attach public IP to the instance.
3. Planning and configuration firewall of the instance.
4. A SSH client for connect to the instance.

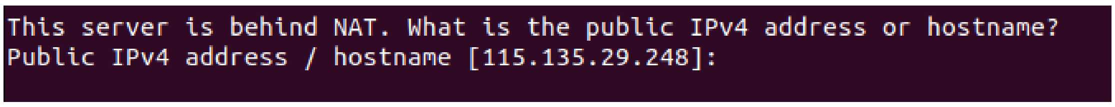
6.5.2 Installing OpenVPN

Due to complexity of manual installation of the OpenVPN, is not recommend to manual install the OpenVPN library, we will be installing OpenVPN with the help of installation script which found on the GitHub. This script will automatically setup all the necessary configuration for OpenVPN. The steps are as below:

1. Download script and run the script

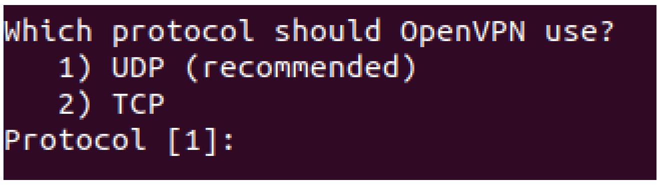
   sudo wget https://git.io/vpn -O openvpn-install.sh && sudo bash openvpn-install.sh

2. Answering all the prompt questions



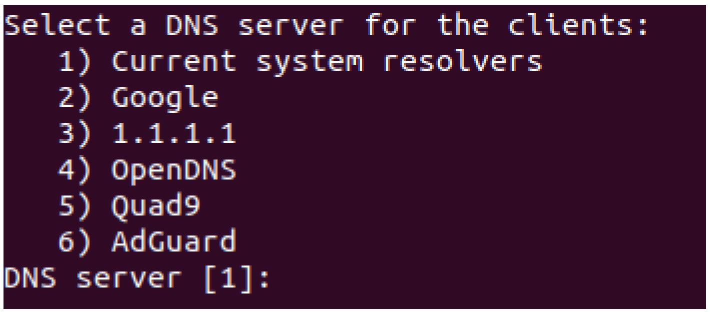
```
This server is behind NAT. What is the public IPv4 address or hostname?
Public IPv4 address / hostname [115.135.29.248]:
```

Figure 6.9 Configure IP address



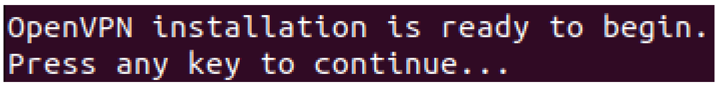
```
Which protocol should OpenVPN use?
   1) UDP (recommended)
   2) TCP
Protocol [1]:
```

Figure 6.10 Select Protocol



```
Select a DNS server for the clients:
   1) Current system resolvers
   2) Google
   3) 1.1.1.1
   4) OpenDNS
   5) Quad9
   6) AdGuard
DNS server [1]:
```

Figure 6.11 Assign DNS



```
OpenVPN installation is ready to begin.
Press any key to continue...
```

3. OpenVPN Installation Done.

The script will setup 2 item. One is OpenVPN library, and the other is the certificate required by the OpenVPN in order to establishing secure connection. The script will first create a Certificate Authority (CA) and generated a server key and certificate request for setting up an OpenVPN server using Easy-RSA. The CA certificate was saved at **/etc/openvpn/server/easy-rsa/pki/ca.crt.** The server certificate request at **/etc/openvpn/server/easy-rsa/pki/reqs/server.req,** and the server key at **/etc/openvpn/server/easy-rsa/pki/private/server.key**. It then signed the server certificate request using the CA, resulting in the creation of the server certificate at **/etc/openvpn/server/easy-rsa/pki/issued/server.crt**. Additionally, an inline file containing both the certificate and key was created at **/etc/openvpn/server/easy-rsa/pki/inline/server.inline**.

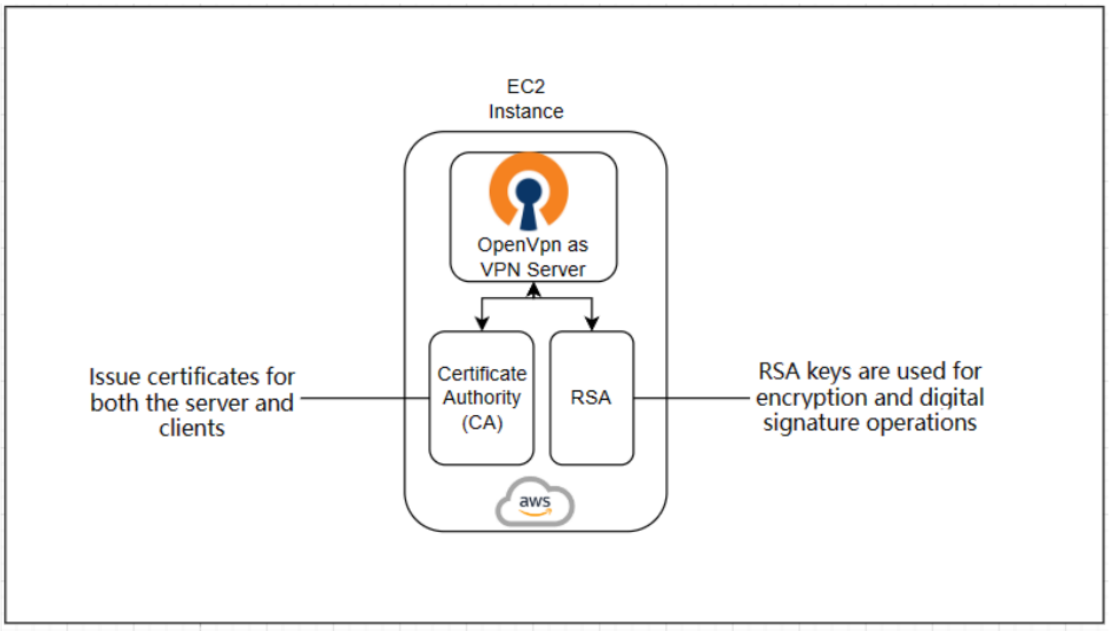
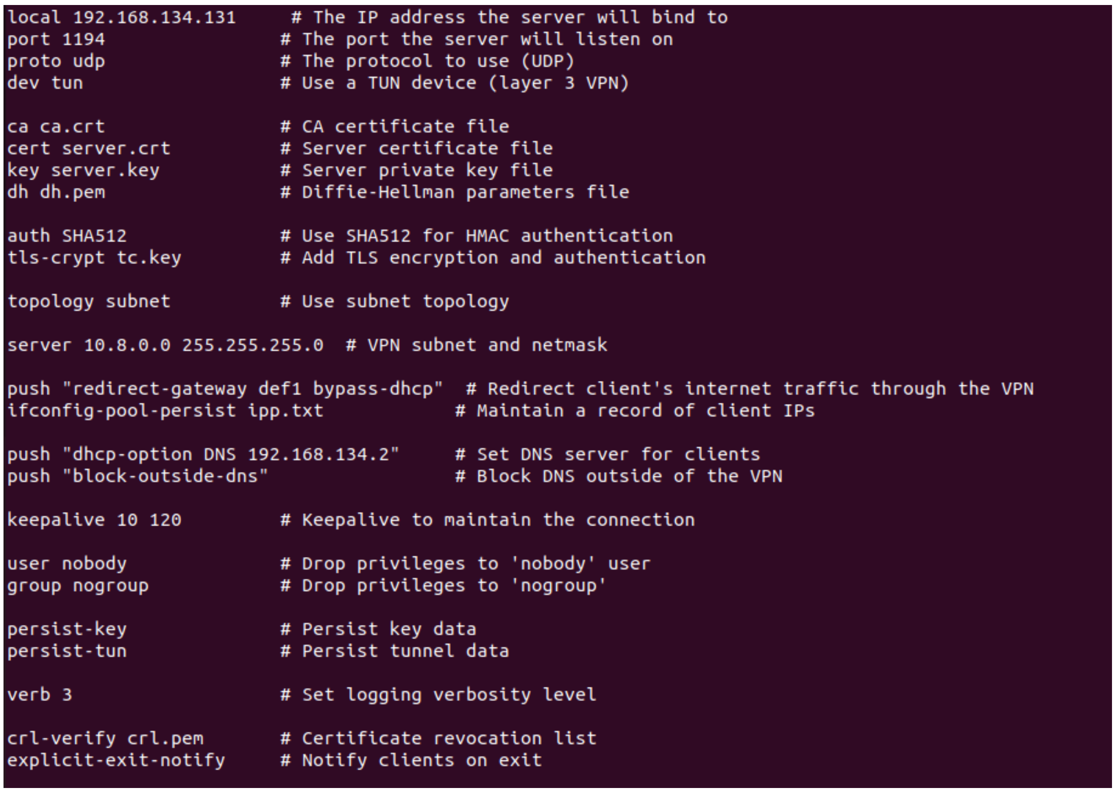


Figure 6.12 OpenVPN Server

Setting up certificates is crucial for establishing a secure and trusted communication channel. Certificates ensure that the server and clients can authenticate each other, preventing unauthorized access. The CA certificate acts as a trusted root which will verifying the authenticity of server and client certificates. The server certificate and key encrypt data transmitted over the VPN, safeguarding it from eavesdropping and tampering.

6.5.3 Post-Installation (Server Configuration)

The OpenVPN server configuration file will locate at **/etc/openvpn/server/server.conf.** The **server.conf** file is the configuration file for the OpenVPN server. It specifies all the necessary settings and parameters to establish and manage the VPN connections. This includes network settings, security configurations, and paths to certificate and key files. Essentially, it defines how the OpenVPN server should operate, including the port to listen on, encryption methods, authentication mechanisms, and other essential server behaviors. We able to modify the file to meet own requirement if necessary. Below is the content of the configuration file.

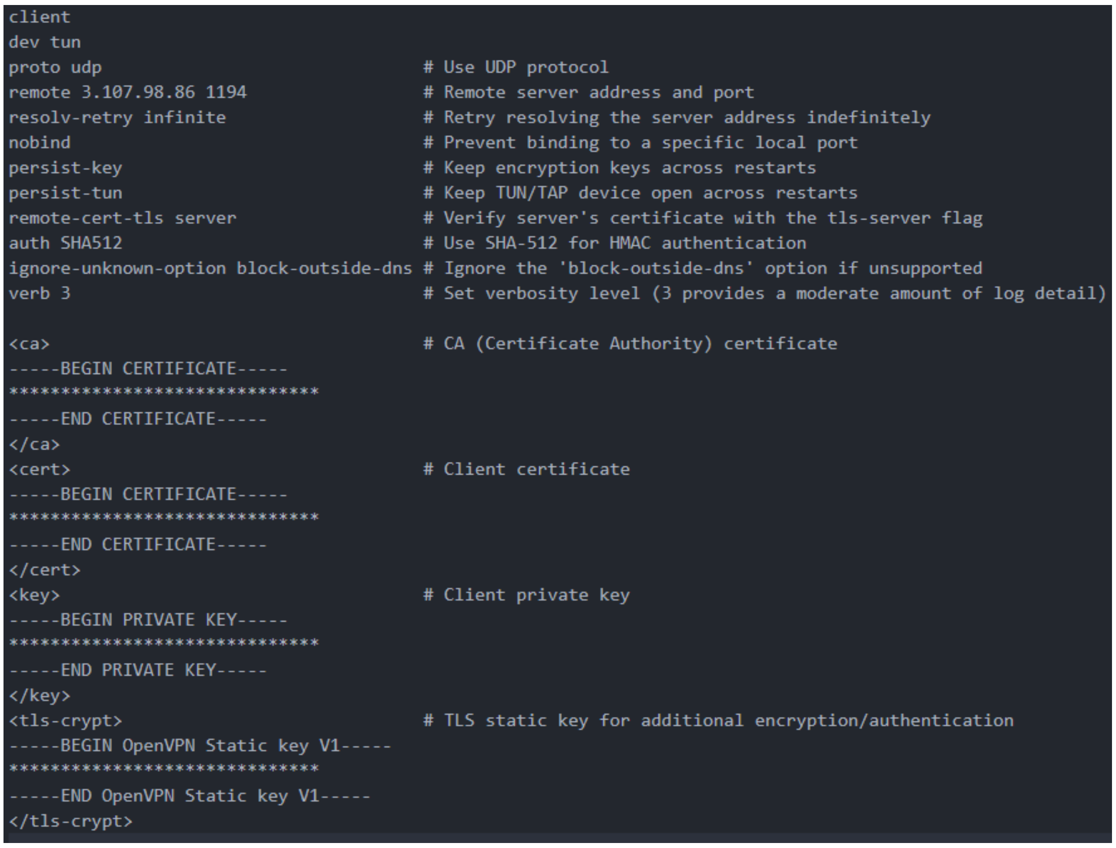


```
local 192.168.134.131     # The IP address the server will bind to
port 1194                # The port the server will listen on
proto udp                # The protocol to use (UDP)
dev tun                  # Use a TUN device (layer 3 VPN)

ca ca.crt                # CA certificate file
cert server.crt          # Server certificate file
key server.key           # Server private key file
dh dh.pem                # Diffie-Hellman parameters file

auth SHA512              # Use SHA512 for HMAC authentication
tls-crypt tc.key         # Add TLS encryption and authentication

topology subnet          # Use subnet topology

server 10.8.0.0 255.255.255.0  # VPN subnet and netmask

push "redirect-gateway def1 bypass-dhcp"  # Redirect client's internet traffic through the VPN
ifconfig-pool-persist ipp.txt           # Maintain a record of client IPs

push "dhcp-option DNS 192.168.134.2"    # Set DNS server for clients
push "block-outside-dns"                # Block DNS outside of the VPN

keepalive 10 120         # Keepalive to maintain the connection

user nobody              # Drop privileges to 'nobody' user
group nogroup            # Drop privileges to 'nogroup'

persist-key              # Persist key data
persist-tun              # Persist tunnel data

verb 3                   # Set logging verbosity level

crl-verify crl.pem       # Certificate revocation list
explicit-exit-notify     # Notify clients on exit
```

Figure 6.13 OpenVPN Server Configuration

6.5.4 Post-Installation (Client Configuration)

To create a new configuration file for client, simply rerun the auto installation script and it will generate a client configuration file automatically. Similar with the server certificate,the client certificate request was generated at **/etc/openvpn/client/easy-rsa/pki/reqs/client.req**, and the client key at /**etc/openvpn/client/easy-rsa/pki/private/client.key**. The client certificate request was then signed using the CA, resulting in the creation of the client certificate at **/etc/openvpn/client/easy-rsa/pki/issued/client.crt**. Additionally, an inline file containing both the client certificate and key was created at **/etc/openvpn/client/easy-rsa/pki/inline/client.inline.** After certificate generated, the client configuration will also be generated,the certificate will be attached to the configuration file and client now able to establish connection to VPN server with this configuration file. The content of the client configuration file are as below.



```
client
dev tun
proto udp                            # Use UDP protocol
remote 3.107.98.86 1194              # Remote server address and port
resolv-retry infinite                # Retry resolving the server address indefinitely
nobind                               # Prevent binding to a specific local port
persist-key                          # Keep encryption keys across restarts
persist-tun                          # Keep TUN/TAP device open across restarts
remote-cert-tls server               # Verify server's certificate with the tls-server flag
auth SHA512                          # Use SHA-512 for HMAC authentication
ignore-unknown-option block-outside-dns # Ignore the 'block-outside-dns' option if unsupported
verb 3                               # Set verbosity level (3 provides a moderate amount of log detail)

<ca>                                 # CA (Certificate Authority) certificate
-----BEGIN CERTIFICATE-----
******************************
-----END CERTIFICATE-----
</ca>
<cert>                               # Client certificate
-----BEGIN CERTIFICATE-----
******************************
-----END CERTIFICATE-----
</cert>
<key>                                # Client private key
-----BEGIN PRIVATE KEY-----
******************************
-----END PRIVATE KEY-----
</key>
<tls-crypt>                          # TLS static key for additional encryption/authentication
-----BEGIN OpenVPN Static key V1-----
******************************
-----END OpenVPN Static key V1-----
</tls-crypt>
```

Figure 6.14 OpenVPN Client Configuration

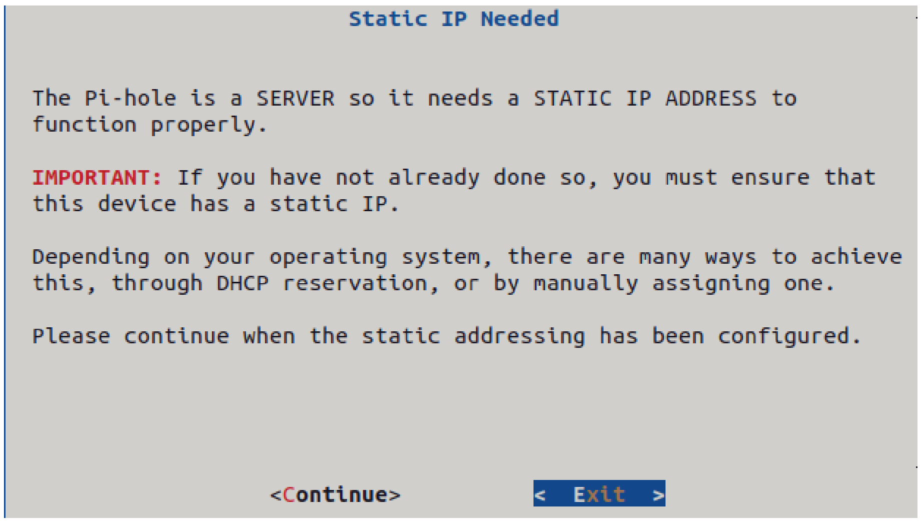
6.5.5 Integration with DNS Ads Block Pihole

Pi-hole is a network-wide ad blocker that acts as a DNS sinkhole, preventing ads and tracking scripts from being downloaded and displayed on any device connected to the network. By functioning as a DNS server, Pi-hole intercepts domain requests and blocks those associated with ads or tracking, effectively filtering unwanted content before it reaches your devices. It offers privacy protection, customizable blocklists, and a web interface for monitoring and managing network activity. Pi-hole is efficient, requires minimal resources, and provides ad-blocking benefits to all devices on the network without the need for individual software. Below is the configuration and installation steps.

1. First we need to install the pihole using an installation command.

   curl -sSL https://install.pi-hole.net | bash

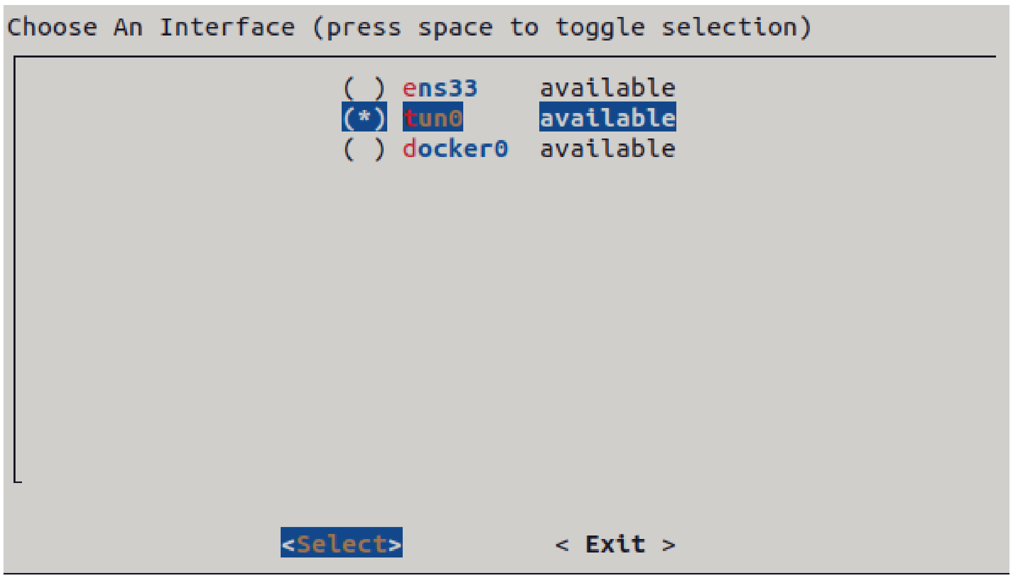
2. Ensure Static IP



```
                         Static IP Needed


 The Pi-hole is a SERVER so it needs a STATIC IP ADDRESS to
 function properly.

 IMPORTANT: If you have not already done so, you must ensure that
 this device has a static IP.

 Depending on your operating system, there are many ways to achieve
 this, through DHCP reservation, or by manually assigning one.

 Please continue when the static addressing has been configured.



                    <Continue>          <  Exit  >
```

Figure 6.15 PiHole Installation

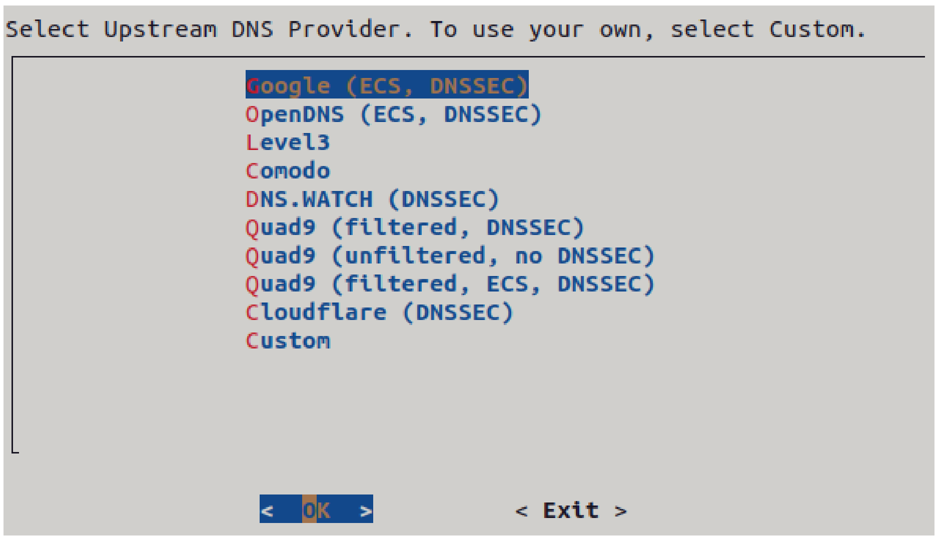
3. Select tun0



```
Choose An Interface (press space to toggle selection)

                     ( ) ens33     available
                     (*) tun0      available
                     ( ) docker0   available


                 <Select>            < Exit >
```

Figure 6.16 Interface Selection

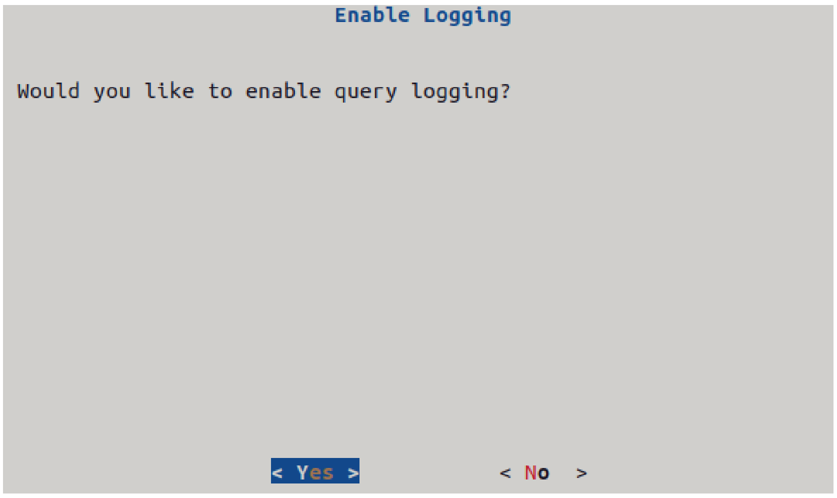
4. Select upstream DNS provider



```
Select Upstream DNS Provider. To use your own, select Custom.

                    Google (ECS, DNSSEC)
                    OpenDNS (ECS, DNSSEC)
                    Level3
                    Comodo
                    DNS.WATCH (DNSSEC)
                    Quad9 (filtered, DNSSEC)
                    Quad9 (unfiltered, no DNSSEC)
                    Quad9 (filtered, ECS, DNSSEC)
                    Cloudflare (DNSSEC)
                    Custom

                     <  OK  >          < Exit >
```

Figure 6.17Select DNS Provider

5. Enable Query login based on own requirement. In this project, we will choose no



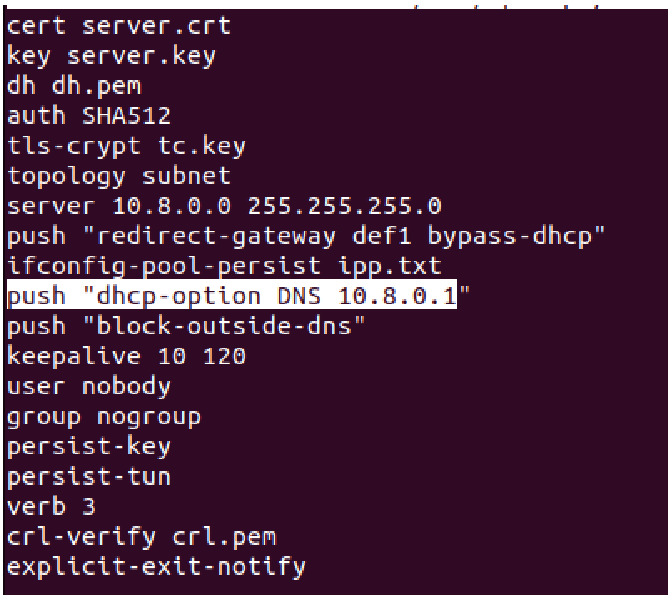
```
                         Enable Logging

 Would you like to enable query logging?

                      < Yes >          < No  >
```

Figure 6.18 Logging selection

6. Modify OpenVPN server configuration file.

    sudo vim /etc/openvpn/server/server.conf

7. Modify the line of push "dhcp-option to 10.8.0.1"



```
cert server.crt
key server.key
dh dh.pem
auth SHA512
tls-crypt tc.key
topology subnet
server 10.8.0.0 255.255.255.0
push "redirect-gateway def1 bypass-dhcp"
ifconfig-pool-persist ipp.txt
push "dhcp-option DNS 10.8.0.1"
push "block-outside-dns"
keepalive 10 120
user nobody
group nogroup
persist-key
persist-tun
verb 3
crl-verify crl.pem
explicit-exit-notify
```

Figure 6.18 OpenVPN server configuration

8. Restart VPN server

   sudo systemctl restart openvpn-server@server

Now the ads block have successfully integrated with the VPN server and our VPN have the capability to filter ads content based on DNS.

6.5.6 Security configuration

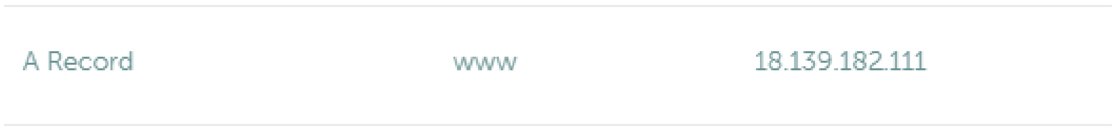
Securing our application is crucial to protect user data and ensure reliable service. Proper configuration of HTTPS and firewall rules is a key part of this security process. In this section, we will discuss why these measures are important for our application, and then we will dive into the detailed steps to configure HTTPS and set up a firewall for our AWS instances.

6.5.7 Configuration for HTTPS using CertBot

In order to host our application using HTTPS, our application need a valid SSL/TLS certificate and a domain name. Below is the step to configure HTTPS communication

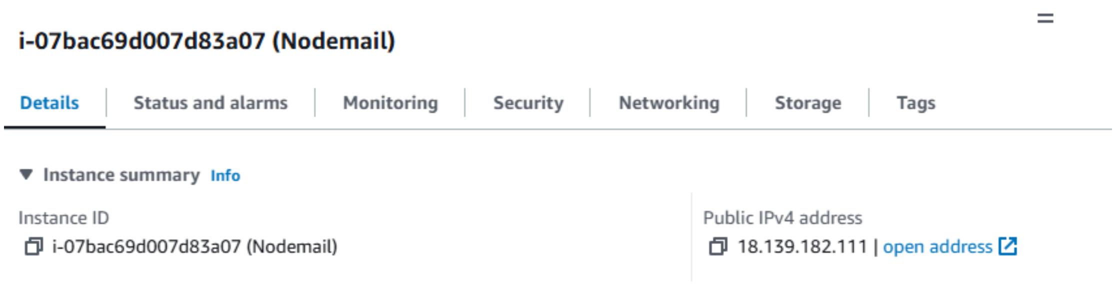
Figure 6.20 Example Domain Name

First, we need to select a domain name for our application, domain name can be acquired from domain name provider such as (GoDaddy, Namecheap, Google Domains). After acquired a domain name, we need to configure DNS Record, we need to set up a prefix for our domain name and the IP address the domain will point to our remote instance which has a Public IP.



A Record	www	18.139.182.111

Figure 6.21 DNS Record with prefix **www** point to 18.139.182.111



i-07bac69d007d83a07 (Nodemail)

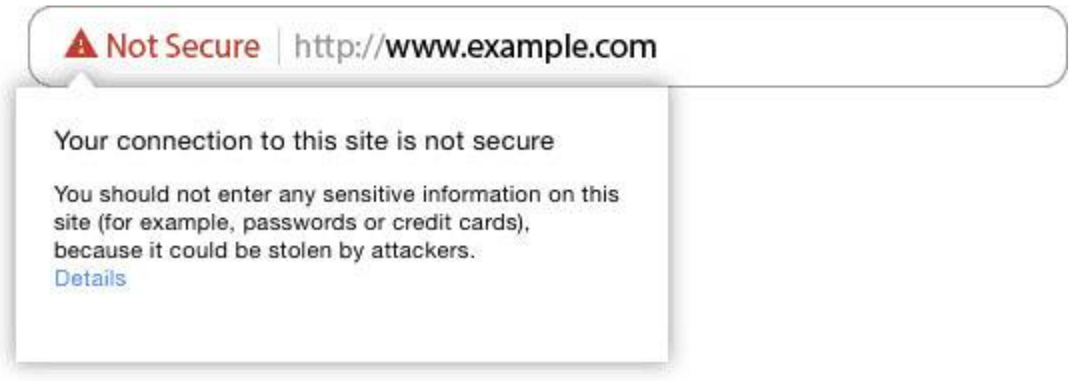
Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

▼ Instance summary Info

Instance ID	Public IPv4 address
i-07bac69d007d83a07 (Nodemail)	18.139.182.111 | open address

Figure 6.22 Instance on AWS with IP address 18.139.182.111

After the configuration, the domain name www.magicconchxhell.xyz will point to the IP address 18.139.182.111. But currently our application is still considered as not secure since we still lack valid SSL/TLS certificate to establish HTTPS connection.



Not Secure | http://www.example.com

Your connection to this site is not secure

You should not enter any sensitive information on this site (for example, passwords or credit cards), because it could be stolen by attackers.
Details

Figure 6.23 Browser warning

Next, we need to generate valid SSL/TLS certificate with the help of Certbot. Certbot is a free, open-source software tool for automatically using Let's Encrypt certificates on manually administrated websites to enable HTTPS. Our remote server using Linux Ubuntu, below are the step to configure certbot with Ubuntu.

1. Install Certbot

```
$ sudo snap install --classic certbot
```

Figure 6.24 Certbot Installation

2. Execute the following instruction on the command line on the machine to ensure that the certbot command can be run.

```
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Figure 6.25 Command Validation

3. Run the Certbot

```
$ sudo certbot certonly --standalone
```

Figure 6.26 Execute Certbot

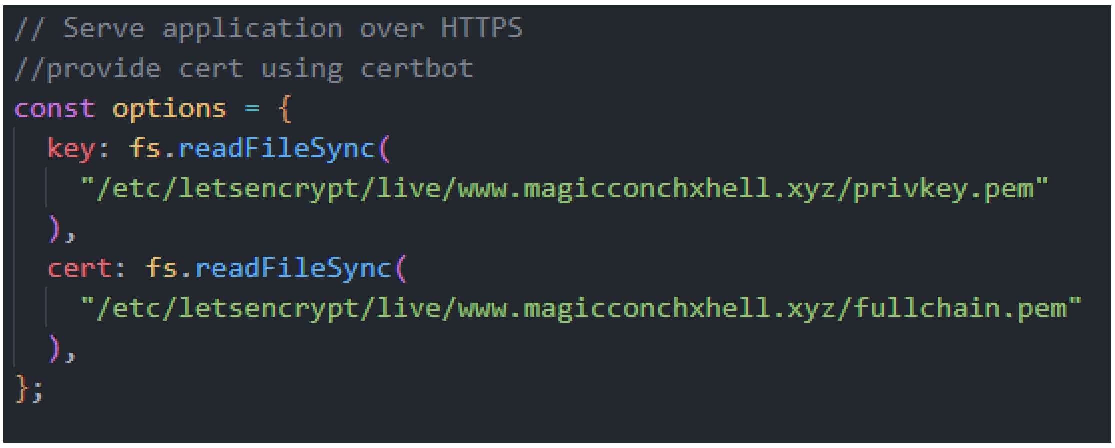
4. Enter Domain Name

   Enter the domain name for the certificate, and answer other questions prompt by

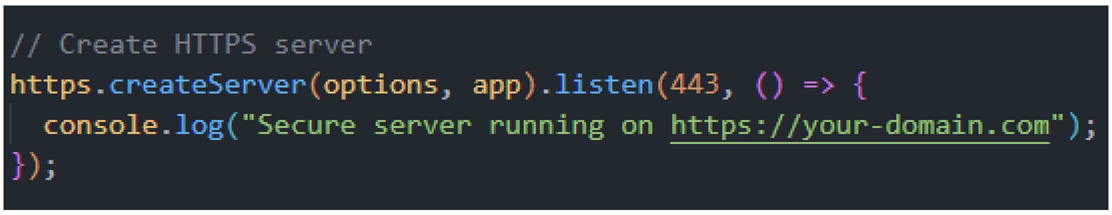
Certbot, after that Certbot will be able to generate a valid SSL/TLS Certificate for HTTPS hosting purpose.

5. Host the certificate

   Next, we need, host the certificate generated by certbot in our application.



```
// Serve application over HTTPS
//provide cert using certbot
const options = {
  key: fs.readFileSync(
    "/etc/letsencrypt/live/www.magicconchxhell.xyz/privkey.pem"
  ),
  cert: fs.readFileSync(
    "/etc/letsencrypt/live/www.magicconchxhell.xyz/fullchain.pem"
  ),
};
```

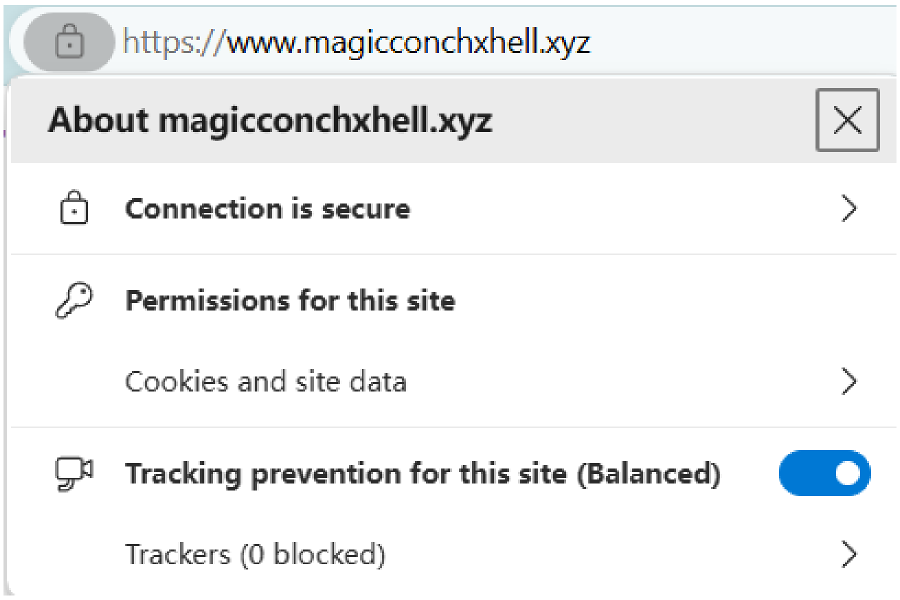


```
// Create HTTPS server
https.createServer(options, app).listen(443, () => {
  console.log("Secure server running on https://your-domain.com");
});
```

Figure 6.27 Serve Certificate

6. Test the application

   Now our application is hosting using HTTPS.



https://www.magicconchxhell.xyz

About magicconchxhell.xyz

Connection is secure

Permissions for this site

Cookies and site data

Tracking prevention for this site (Balanced)

Trackers (0 blocked)

Figure 6.28 Https Validation

6.5.8 Configuration for Firewall on AWS Security Group

Firewall is important for ensuring the security and integrity of our application (instance). In AWS, firewall configurations are managed through the Security Groups which are a set of traffic inbound and outbound rules attached to an instance. Below are the rules attached to the VPN server and Centralized Backend.

Centralized Backend

Table 6.7 Server Inbound Rules:

Type	**Protocol**	**Port Range**	**Source**
HTTP	TCP	80	0.0.0./0
HTTPS	TCP	443	0.0.0./0
SSH	TCP	22	0.0.0./0

Table 6.8 Server Outbound Rules:

Type	**Protocol**	**Port Range**	**Source**
ALL	ALL	ALL	0.0.0./0

VPN Server

Table 6.9 VPN Inbound Rules

Type	**Protocol**	**Port Range**	**Source**
HTTP	TCP	80	0.0.0./0
HTTPS	TCP	443	0.0.0./0
UDP	UDP	1194	0.0.0./0
UDP	UDP	1194	::/0
SSH	TCP	22	0.0.0./0

Table 6.10 VPN Outbound Rules:

Type	**Protocol**	**Port Range**	**Source**

ALL	ALL	ALL	0.0.0./0

Explanation of each Rules

1. HTTP (Hypertext Transfer Protocol)

   HTTP is the foundation of any data exchange on the Web, it used for transmitting hypertext request and information on internet. The reason for exposing HTTP port in our application is ensure that the request which send using HTTP can be capture and we able to redirect that request to HTTPS automatically for secure the communication. This method able to improve and provide a smoother request experience.

2. HTTPS (Hypertext Transfer Protocol Secure)

   HTTPS ensures that the data exchanged between user's browser and web server is encrypted, which able to protect against data breaches and man in the middle attack. HTTPS is the critical component for secure communication between client and server. In Our application, all the requests are transmit using HTTPS protocol between client, centralized backend, and VPN Server.

3. SSH (Secure Shell)

   Secure Shell is the protocol which used to securely access and manage the remote servers over network. It able to provides secure channel for communication between client and server, by encrypting data transmitted between them. SSH allow us to remote manage and configure our server over the network

4. UDP 1194 (Port for OpenVPN Connection)

   The port 1194 which using UDP protocol is the port for establish connection with the VPN server. By default, OpenVPN use UDP protocol listen on port 1194 for communication. OpenVPN can be configured to use TCP or other port while UDP

1194 is the standard and the most commonly used port for OpenVPN connections.

Chapter 7
System Testing

7.1 Introduction

This chapter documents the testing activities of the project. It includes the test plan which all tests have been complete and compare to actual and expected results. The testing cover for this project included the frontend application, backends APIs and VPN testing.

7.2 Test Plan

Test Plan as a critical document which aims to enhance the efficiency and effectiveness of testing process. It includes details test scope, test item, test strategy, entry and exit criteria. The test plan will outline the testing process, ensuring that all activities are carried out accurately and comprehensively. The main goal of this test plan is to detect system errors, failures, correct the error and ensure stability of the VPN application. Upon test closure no critical bug reports should be issued.

7.2.1 Test Scope

The test scope encompasses the frontend and backend component of the application. The testing for frontend and backend will be tested independently which frontend application will focuses the test on the interaction and functionality while backend will focus test on the functionality of the backend API endpoint. The following are the test activity that will be conducted for this project:

1. Unit testing of the mobile application.
2. Unit testing of the API.
3. VPN testing

7.2.2 Test Strategy

The testing strategy for this project is focused on behavioral testing strategy and functional testing strategy. Behavioral strategy focuses on the external behavioral of the application which test on the frontend mobile application while functional testing will focus on the functionality of the VPN connection, Server, and API endpoint of entire system.

7.2.3 Test Condition

Unit test was planned for this project which will be carried out by developers with various testing tools.

Entry Criteria

The testing can begin when the entry criteria listed are met.

1. The entire test project has already been scheduled in detail.
2. All features to be tested were completed and functioned stably.
3. All relevant resources, such as tools and devices, have been set up in the test environment.

Exit Criteria

The testing can end when the exit criteria listed are met.

1. All test cases have been executed and passed.
2. All defects found during the test phase are corrected and closed.
3. No critical defects were outstanding.
4. No changes to the codes and design.

7.3 Unit Test

Unit testing is a software testing method where individual units or components of an application which are tested in isolation. The primary purpose of unit testing is to ensure that each part of the software behaves as expected. In this project, unit test will perform on both frontend application and backend API endpoint, purpose of this unit test to ensure the functionality and quality of the system. Both components will be tested individually which able to narrow the scope of defect detection and identifying possible root cause of problem efficiently and effectively.

7.3.1 Mobile Application Unit Test

The project performed unit test by manually inserting values clicking buttons and conducting various interaction with the mobile application within emulator. This testing is purpose to test the functionality and error handling of the frontend mobile application.6 test case were designed to test the functionality of the mobile application. The actual and expected results of the tests were compared and documented.

Table 7.1 Test Case APP-1 - Register

Test Case	APP1		**Test Case Title**	Register	
Test Case Overview	Test register functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Register an account with a valid email, password and confirm password	1. Enter email	test@gmail.com	HTTP request sent.	HTTP request sent.	Pass
	2. Enter password	Test123@			
	3.Enter confirm password	Test123@			
	4. Tap register button	-			
Register account with empty input field.	1. Tap register button	-	Error message display without sending HTTP request.	Error message display without sending HTTP request	Pass
Register account with valid email, password	1.Enter email	test@gmail.com	Error message display without sending HTTP	Error message display without sending HTTP	Pass
	2. Enter password	Test123@			

but not match confirm password.	3. Enter confirm password	Test122@	request.	request.	
	4.Tap register password	-			

Table 7.2 Test Case APP-2 - Login

Test Case	APP2		**Test Case Title**	Login	
Test Case Overview	Test login functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Login account with valid email and password.	1. Enter email	test@gmail.com	HTTP request sent.	HTTP request sent.	Pass
	2. Enter password	Test123@			
	3. Tap register button	Test123@			
		-			
Login account with empty input field.	1. Tap register button	-	Error message display without sending HTTP request.	Error message display without sending HTTP request	Pass

Table 7.3 Test Case APP-3 – Forget Password

Test Case	APP3		**Test Case Title**	Forget Password	
Test Case Overview	Test edit password functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Valid email and new	1. Enter email	test@gmail.com	HTTP request sent.	HTTP request sent.	Pass

password match with confirm password	2. Tap proceed button	-			
	3. Enter confirm password	Test123@			
	4.Enter new confirm password	Test123@			
	5. Tap reset button	-			
Not registered email	1. Enter email	test@gmail.com	Error message display without sending HTTP request.	Error message display without sending HTTP request	Pass
	2. Tap proceed button	-			
Valid email and new password but not match with confirm password	1. Enter email	test@gmail.com	Error message display without sending HTTP request.	Error message display without sending HTTP request.	Pass
	2. Tap proceed button	-			
	3. Enter confirm password	Test123@			
	4.Enter new confirm password	Test1234@			
	5. Tap reset button	-			

Table 7.4 Test Case APP-4 –Logout

Test Case	APP4		**Test Case Title**	Logout	
Test Case Overview	Test logout functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Logout Button	1. Press logout button	-	Navigate to login page	Navigate to login page	Pass

Table 7.5 Test Case APP-5 –Select Country

Test Case	APP5		**Test Case Title**	Select Country	
Test Case Overview	Test select country functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Select country with backend server in working condition	1. Press select country button	-	HTTP request sent.	HTTP request sent.	Pass
	2. Tap selected country	-			
Select country with backend server error condition	1. Press select country button	-	Error message display	Error message display	Pass
	2. Tap selected country	-			

Table 7.6 Test Case APP-6 –Connect to VPN

Test Case	APP6		**Test Case Title**	Connect to VPN	
Test Case Overview	Test connect to vpn functionality of mobile application				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Connect to VPN and the daily quota does not reach	1. Press connect button	-	HTTP request sent. Established connection to VPN	HTTP request sent. Established connection to VPN	Pass
Connect to VPN and the account already reach daily quota	1. Press select country button	-	HTTP request sent. A message prompt to informed quota	HTTP request sent. A message prompt to informed quota	Pass

			information. Does not establish connection to VPN	information. Does not establish connection to VPN	

7.3.2 APIs Unit Test

For testing of backend APIs, Postman APIs development tools was used for this testing. Postman allows to send HTTP request to API to be tested and obtained the responses.11 test case are design for the backend APIs. The actual and expected result of tests were compared and documented.

Table 7.7 Test Case API-1 –Register

Test Case	API-1		**Test Case Title**	Register	
Test Case Overview	Test Register functionality of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Register with valid email and password.	1. Select Post method	-	New record inserted to database and one time password send to register email.	New record inserted to database and one time password send to register email.	Pass
	2. Enter email	Test@gmail.com			
	3. Enter password	Test123@			
	4. Send Http Request	-			
Register with valid email registered before	1. Select Post method	-	API response Email have been registered	API response Email have been registered	Pass
	2. Enter email	Test@gmail.com			
	3. Enter password	Test123@			
	4. Send Http Request	-			
Server error	1. Select Post method	-	API response Error during registration	API response Error during registration	Pass
	2. Enter email	Test@gmail.com			
	3. Enter password	Test123@			

	4. Send Http Request	-			

Table 7.8 Test Case API-2 –Validate One Time Password for register

Test Case	API-2		**Test Case Title**	Validate One Time Password for register	
Test Case Overview	Test one time password verification of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Validate with valid one-time password	1. Select Post method	-	New record inserted to database and API response Registration successful	New record inserted to database and API response Registration successful	Pass
	2. Enter email	Test@gmail.com			
	3. Enter one time password	123456			
	4. Send Http Request	-			
Validate with invalid one-time password	1. Select Post method	-	API response Invalid One-time password	API response Invalid One-time password	Pass
	2. Enter email	Test@gmail.com			
	3. Enter one time password	123456			
	4. Send Http Request	-			
Server error	1. Select Post method	-	API response Error during OTP validation	API response Error during OTP validation	Pass
	2. Enter email	Test@gmail.com			
	3. Enter one time password	123456			
	4. Send Http Request	-			

Table 7.9 Test Case API-3 –User Login

Test Case	API-3		**Test Case Title**	User Login	
Test Case Overview	Test login functionality of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Login with valid email and password	1. Select Post method	-	API response Login successful and return a token	API response Login successful and return a token	Pass
	2. Enter email	Test@gmail.com			
	3. Enter password	123456			
	4. Send Http Request	-			
Login with email not registered before	1. Select Post method	-	API response Email not registered	API response Email not registered	Pass
	2. Enter email	Test1@gmail.com			
	3. Enter password	123456			
	4. Send Http Request	-			
Login with valid email but invalid password	1. Select Post method	-	API response Password Error	API response Password Error	Pass
	2. Enter email	Test1@gmail.com			
	3. Enter password	12345677			
	4. Send Http Request	-			
Server error	1. Select Post method	-	API response Error during login	API response Error during login	Pass
	2. Enter email	Test1@gmail.com			
	3. Enter password	123456			
	4. Send Http Request	-			

Table 7.10 Test Case API-4 –Resend One time password

Test Case	API-4		**Test Case Title**	Resend One time password	
Test Case Overview	Test login functionality of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Resend one time password	1. Select GET method	-	One time password sends to email. API response One time password resend to the entered email	One time password sends to email. API response One time password resend to the entered email	Pass
	2. Enter email	Test@gmail.com			
	3.Send Http request	-			
Server error	1. Select GET method	-	API response Error during resend one time password	API response Error during resend one time password	Pass
	2. Enter email	Test@gmail.com			
	3.Send Http request	-			

Table 7.11 Test Case API-5 –Fetch VPN country information

Test Case	API-5		**Test Case Title**	Fetch VPN country information	
Test Case Overview	Test Fetch VPN country information functionality of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Query the VPNstatus endpoint	1. Select GET method	-	API response status of VPN country	API response status of VPN country	Pass
	2. Send Http request	-			
Server error	1. Select GET method	-	API response Error getting information	API response Error getting information	Pass
	2. Send Http request	-			

Table 7.12 Test Case API-6 –Fetch VPN configuration

Test Case	API-6		**Test Case Title**	Fetch VPN configuration	
Test Case Overview	Test Fetch VPN configuration of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
Query the getconfig endpoint with valid token	1. Select GET method	-	API response VPN configuration	API response VPN configuration	Pass
	2. Enter country	Test@gmail.com			
	3. Enter token	token			
	4. Send Http Request	-			
Query the getconfig endpoint with invalid token	1. Select GET method	-	API response invalid token	API response invalid token	Pass
	2. Enter country	Test@gmail.com			
	3. Enter token	Invalid token			
	4. Send Http Request	-			

Table 7.13 Test Case API-7 –Validate Email for reset password

Test Case	API-7		**Test Case Title**	Validate Email for reset password	
Test Case Overview	Test Validate Email for reset password of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
With valid email	1. Select GET method	-	API response status 200 and send one time password to email	API response status 200 and send one time password to email	Pass
	2. Enter email	Test@gmail.com			
	3. Send Http Request	-			
With invalid email	1. Select GET method	-	API response 409	API response 409	Pass

	2. Enter email	Test11@gmail.com	Email not registered	Email not registered	
	3. Send Http Request	-			
Server Error	1. Select GET method	-	API response error occur during validate	API response error occur during validate	Pass
	2. Enter email	Test@gmail.com			
	3. Send Http Request	-			

Table 7.14 Test Case API-8 –Validate One time password for reset password

Test Case	API-8		**Test Case Title**	Validate One time password for reset password	
Test Case Overview	Test Validate One time password for reset password of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
With valid email and one time password	1. Select POST method	-	API response validation successful	API response validation successful	Pass
	2. Enter email	Test@gmail.com			
	3. Enter One time password	123456			
	4. Send Http Request	-			
With valid email and invalid one-time password	1. Select POST method	-	API response invalid one-time password	API response invalid one-time password	Pass
	2. Enter email	Test@gmail.com			
	3. Enter One time password	1234567			
	4. Send Http Request	-			

Server error	1. Select POST method	-	API response error during validation	API response error during validation	Pass
	2. Enter email	Test@gmail.com			
	3. Enter One time password	123456			
	4. Send Http Request	-			

Table 7.15 Test Case API-9 –Reset password

Test Case	API-9		**Test Case Title**	Reset password	
Test Case Overview	Test Validate One time password for reset password of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
With email and new password	1. Select POST method	-	API response reset successful	API response reset successful	Pass
	2. Enter email	Test@gmail.com			
	3. Enter password	124456			
	4. Send Http Request	-			
Server error	1. Select POST method	-	API response error occur during reset	API response error occur during reset	Pass
	2. Enter email	Test@gmail.com			
	3. Enter One time password	1234567			
	4. Send Http Request	-			

Table 7.16 Test Case API-10 –Change to premium status

Test Case	API-10		**Test Case Title**	Change to premium status	
Test Case Overview	Test Change to premium status of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
With valid token	1. Select GET method	-	API response upgrade successful	API response upgrade successful	Pass
	2. Enter token	token			
	3. Send Http Request	-			
With invalid token	1. Select GET method	-	API response invalid token	API response invalid token	Pass
	2. Enter token	token			
	3. Send Http Request	-			
Server error	1. Select GET method	-	API response error occur during upgrade	API response error occur during upgrade	Pass
	2. Enter token	token			
	3. Send Http Request	-			

Table 7.17 Test Case API-11 –Connect VPN

Test Case	API-11		**Test Case Title**	Connect VPN	
Test Case Overview	Test Connect VPN of API				
Test Detail	**Test Instruction**	**Test Data**	**Expected Result**	**Actual Result**	**Condition**
With valid token and disconnect time	1. Select POST method	-	API response allow or denied based on remaining quota	API response allow or denied based on remaining quota	Pass
	2. Enter token	token			
	3. Enter disconnect time	disconnect time			

	4. Send Http Request	-			
With invalid token and disconnect time	1. Select POST method	-	API response invalid token	API response invalid token	Pass
	2. Enter token	token			
	3. Enter disconnect time	disconnect time			
	4. Send Http Request	-			

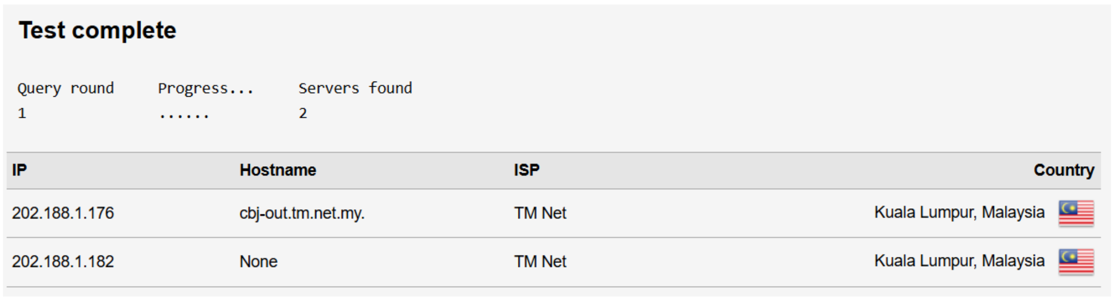
7.4 VPN Test

In this section, we will cover various VPN tests such as DNS leak tests, IP address and location tests, traffic routing tests, speed tests, and network packet analysis using Wireshark. These tests are designed to ensure that the VPN is functioning correctly by verifying that it effectively masks the user's IP address, prevents DNS leaks, routes traffic securely, maintains satisfactory connection speeds, and properly encrypts data. By performing these tests, we can confirm the VPN's ability to protect user privacy, maintain anonymity, and ensure data security while using the internet.

7.4.1 DNS Leak Test

DNS stands for Domain Name System, which translating human-readable domain names into IP addresses that computer use to communicate with each other. When using VPN services, every request as well as the DNS queries should also be routed through the VPN server instead of our internet service provider (ISP) ‘s DNS server instead. If the DNS queries is sent to our local internet service provider (ISP), this is considered as DNS leak. By performing a DNS leak test, we able to tell the effectiveness of the VPN services and ensures privacy protection for online activities.

We will test the DNS leak with the help of **DNSleaktest** website, this website offers simple test to determine if our DNS request is being leaked. Below shows 2 tests for the DNS leak, the first image is the test before we establish connection to VPN server and as the test results shows that, our DNS queries is route to our Internet Service Provider (ISP) which also indicates that the ISP are aware of our online activities.



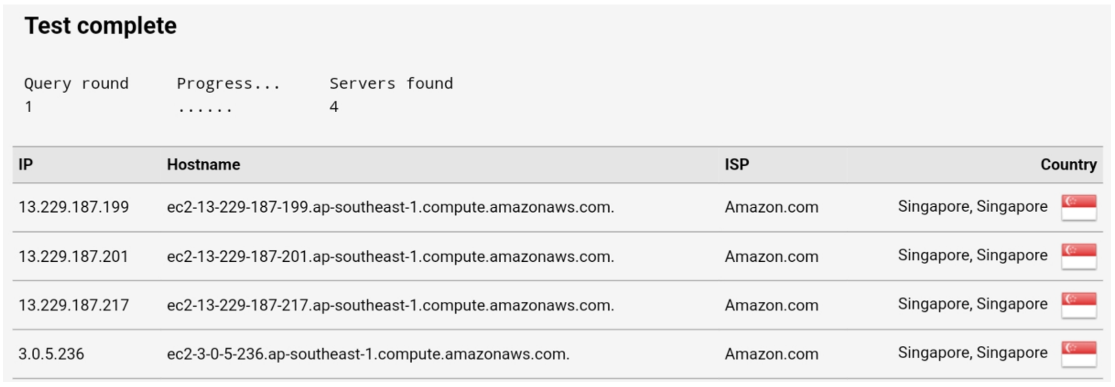
Test complete

```
Query round    Progress...    Servers found
1              ......         2
```

IP	Hostname	ISP	Country
202.188.1.176	cbj-out.tm.net.my.	TM Net	Kuala Lumpur, Malaysia
202.188.1.182	None	TM Net	Kuala Lumpur, Malaysia

Figure 7.1 DNS Leak Test

Second test is tested when we connected to the VPN services. The results show that our DNS queries are routed to the DNS servers provided by the same entity that operates the VPN server. This indicates that our DNS requests are being handled securely within the VPN's encrypted tunnel, confirming that there is no DNS leak. By routing DNS queries through the VPN's DNS servers, it able ensures our online activity is kept private and is not visible to our ISP or any external observers. This setup maintains our online privacy and anonymity by keeping all aspects of our internet traffic secure.



Test complete

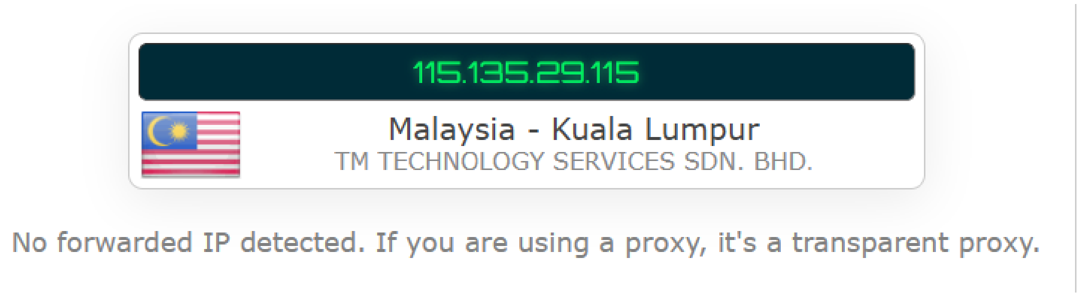
```
Query round     Progress...     Servers found
1               ......          4
```

IP	Hostname	ISP	Country
13.229.187.199	ec2-13-229-187-199.ap-southeast-1.compute.amazonaws.com.	Amazon.com	Singapore, Singapore
13.229.187.201	ec2-13-229-187-201.ap-southeast-1.compute.amazonaws.com.	Amazon.com	Singapore, Singapore
13.229.187.217	ec2-13-229-187-217.ap-southeast-1.compute.amazonaws.com.	Amazon.com	Singapore, Singapore
3.0.5.236	ec2-3-0-5-236.ap-southeast-1.compute.amazonaws.com.	Amazon.com	Singapore, Singapore

Figure 7.2 DNS Leak Test Result

7.4.2 IP Address and Location Test

IP address and Location test is an effective way to determine if the VPN is effectively masking our real IP address and location. The main goal of VPN is to mask the real IP address and location which helps protect the privacy and precenting websites advertisers and other entities from tacking real identity and location. When connected to VPN, our traffic all are route through the VPN server first, then the VPN server will make the request for us and this will resulting the web server only received request from our VPN server and shows the VPN server IP address instead our real IP address.



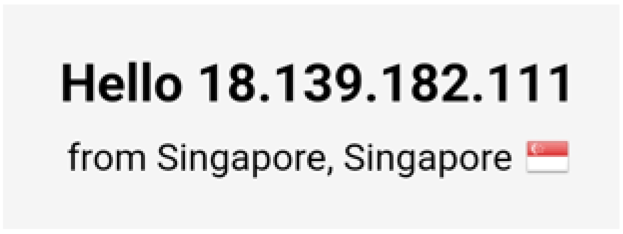
115.135.29.115

Malaysia - Kuala Lumpur

TM TECHNOLOGY SERVICES SDN. BHD.

No forwarded IP detected. If you are using a proxy, it's a transparent proxy.

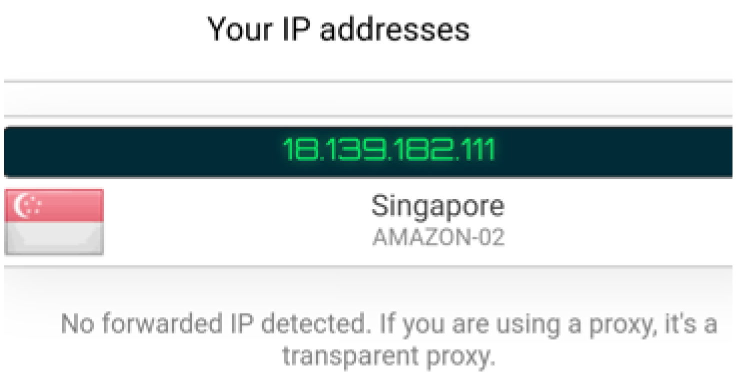
Figure 7.3 IP address before connecting to VPN



Hello 18.139.182.111

from Singapore, Singapore

Figure 7.4 IP address After connecting to VPN



Your IP addresses

18.139.182.111

Singapore
AMAZON-02

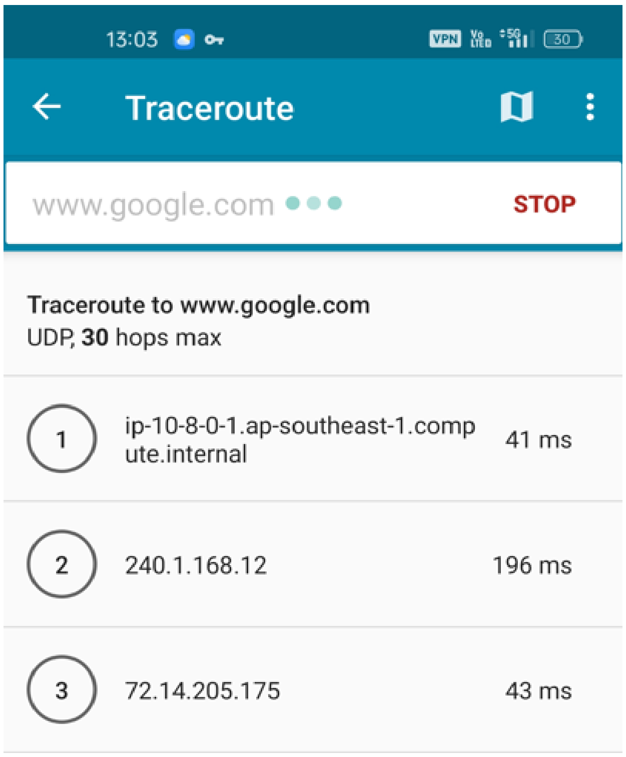
No forwarded IP detected. If you are using a proxy, it's a transparent proxy.

Figure 7.5 IP address and Location after connected to VPN

After connecting to VPN server, we will run the test using **IPleak** website to detect our current IP address. The result shows our IP address has changes to the VPN server's IP address and location which indicates the VPN is masking our real address effectively.

7.4.3 Traffic Routing Test

The traceroute test is conducted after establishing connection to VPN. The purpose of this test is to understand the path that data packets take from source device to destination after connecting to VPN.



Traceroute to www.google.com
UDP, 30 hops max

1	ip-10-8-0-1.ap-southeast-1.compute.internal	41 ms
2	240.1.168.12	196 ms
3	72.14.205.175	43 ms

Figure 7.6 Traffic Routing Test

Traceroute Results:

1. First Hop (VPN Server):

➢ Address: ip-10-8-0-1.ap-southeast-1.compute.internal (VPN Server)

➢ Latency: 41 ms

2. Subsequent Hops:

➢ Not detailed as they are not the focus of this summary.

The traceroute results confirms that after establishing the VPN connection, the data packets are first routed through the VPN server. This is evidenced by the result of the testing which the first hop of the data packets.

Through the traceroute testing, we can claim that all internet traffic from the source device is encrypted which protects the data from potential eavesdropping or interception by unauthorized parties through VPN tunnel. Beside that VPN server able mask, the sources device's IP address and providing anonymity and preventing tracking of online activities. By routing through VPN server, it allows user to bypass geographical restriction and access content that may be blocked or restricted in original region, since all the data packets are first routed to VPN server first before performing the actual request as we seen in the testing result.

7.4.4 Speed Test

The speed test is conducted by comparing the speed of network before and after connecting to VPN services. This aim to analyze the impact and the performance of our VPN application.

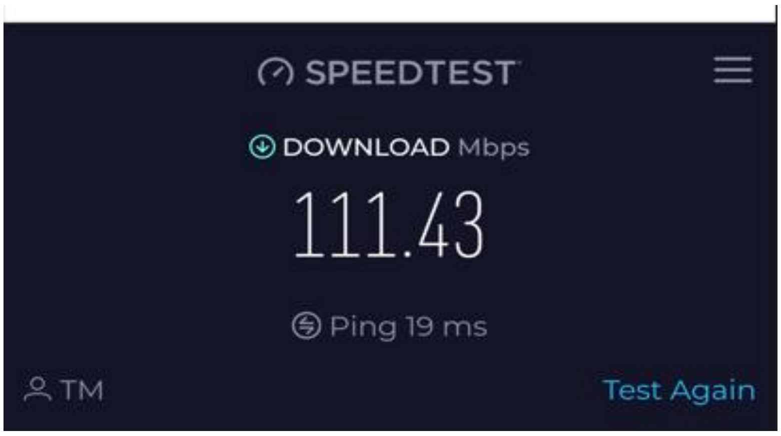


Figure 7.7 Speed Test Before Connect to VPN

Before Connecting to VPN:

- **Download Speed:**111.43 Mbps
- **Ping :**19ms
- **Server:** TM

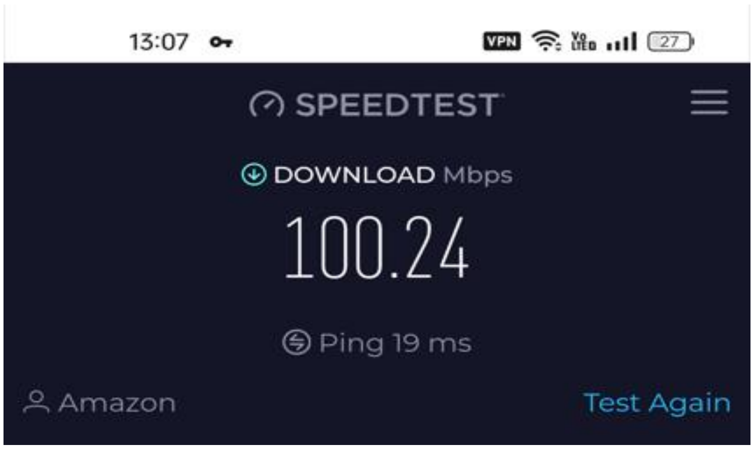


Figure 7.8 Speed Test after Connect to VPN

After Connecting to VPN

- **Download Speed:**100.24 Mbps
- **Ping :**19ms
- **Server:** Amazon

Based on the test results, before connecting to VPN, the test was conducted through local network. The download speed measured was 111.43 Mbps with a ping of 19 MS, while after connecting to VPN the speed measured was 100.24 Mbps with a ping of 19 and there is a slight reduction in download speed when connected to the VPN, which is expected due to the additional overhead of encryption and routing through the VPN server.

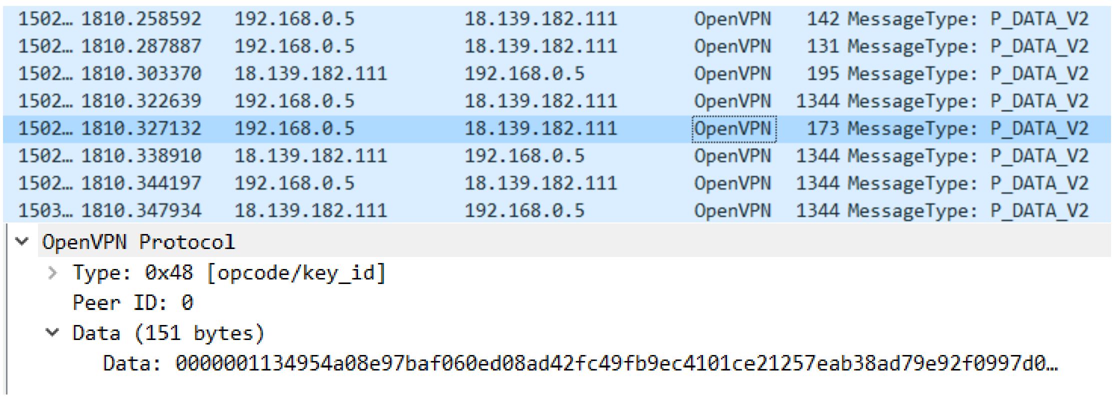
From the result we can say that our VPN application performance and speed is consider as effective. Our application has the minimal impact on speed indicates that the VPN maintains a high level of performance which providing security and encryption without significantly compromising download speeds or increasing latency.

Important Consideration

While the performance of speed of the VPN is consider effective, but the actual speed can be vary significantly depending on the VPN server's location and distance from the original location. Other factors such as server load and network congestion can also affect the speed and performance of the connection.

7.4.5 Capturing Network Traffic using Wireshark

To verify that the packet during the transmission is encrypted, we will capture and analyze the packet after connected to the VPN. From the screenshot, we can saw that the **Data** payload column contains a brunch of hexadecimal string which represents that, the content we transmit between the VPN is encrypted and only the intended recipient, either the VPN client and server can decrypt and read the data.



```
1502… 1810.258592   192.168.0.5        18.139.182.111     OpenVPN    142 MessageType: P_DATA_V2
1502… 1810.287887   192.168.0.5        18.139.182.111     OpenVPN    131 MessageType: P_DATA_V2
1502… 1810.303370   18.139.182.111     192.168.0.5        OpenVPN    195 MessageType: P_DATA_V2
1502… 1810.322639   192.168.0.5        18.139.182.111     OpenVPN   1344 MessageType: P_DATA_V2
1502… 1810.327132   192.168.0.5        18.139.182.111     OpenVPN    173 MessageType: P_DATA_V2
1502… 1810.338910   18.139.182.111     192.168.0.5        OpenVPN   1344 MessageType: P_DATA_V2
1502… 1810.344197   192.168.0.5        18.139.182.111     OpenVPN   1344 MessageType: P_DATA_V2
1503… 1810.347934   18.139.182.111     192.168.0.5        OpenVPN   1344 MessageType: P_DATA_V2
v OpenVPN Protocol
  > Type: 0x48 [opcode/key_id]
    Peer ID: 0
  v Data (151 bytes)
      Data: 0000001134954a08e97baf060ed08ad42fc49fb9ec4101ce21257eab38ad79e92f0997d0…
```

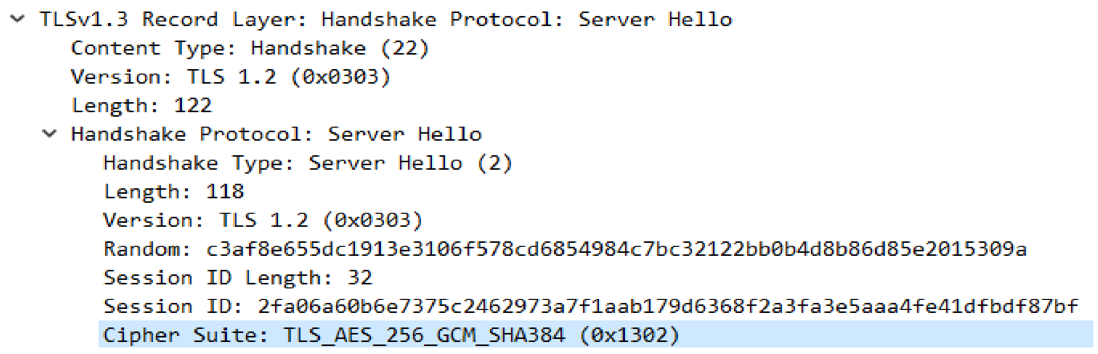
Figure 7.9 Result in WireShark

7.4.5.1 Secure Connection Verification

In order to verify the security of our VPN connection, we will continue the capture packets during the establishment of the connection. By analyzing the packets, we can ensure that the communication between the client and server is properly encrypted and secure. Two critical components in this verification process are the use of the TLS 1.2 protocol and the specific cipher suite employed during the connection.

Wireshark can capture various handshake messages, including the "Server Hello" packet. This packet is part of the TLS handshake process, where the server communicates the chosen encryption parameters to the client. By examining the packets, we can identify the protocols and cipher suites being used by our VPN server to ensure it meet our security requirements.

From the screenshot we can focus on 2 parameters. The Handshake protocols and the Cipher suite.



```
v TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
  v Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 118
      Version: TLS 1.2 (0x0303)
      Random: c3af8e655dc1913e3106f578cd6854984c7bc32122bb0b4d8b86d85e2015309a
      Session ID Length: 32
      Session ID: 2fa06a60b6e7375c2462973a7f1aab179d6368f2a3fa3e5aaa4fe41dfbdf87bf
      Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
```

Figure 7.10 Cipher suite

1. Handshake Protocol TLS 1.2

TLS 1.2 is the widely recognized and robust version of the Transport Layer Security protocol. It plays a crucial role in securing the OpenVPN connection by establishing a secure channel between the client and server. It will perform a series of cryptographic handshakes between client and server which include key exchange, certificate verification, and the negotiation of encryption parameters to ensures that the data transmitted between the client and server is encrypted and protected against eavesdropping and tampering.

2. Cipher Suite TLS_AES_256_GCM_SHA384

The cipher suite **TLS_AES_256_GCM_SHA384** is of paramount importance in ensuring a secure connection. This suite specifies the algorithms used for encryption, integrity checking, and hashing. **AES_256_GCM**- the Advanced Encryption Standard (AES) with a 256-bit key in Galois/Counter Mode (GCM) ensures that the data is encrypted, providing confidentiality and integrity. While **SHA384**-Secure Hash Algorithm 384-bit is used for hashing, ensuring that the data has not been altered during transmission.

AES_256_GCM is known for its robust encryption capabilities, while GCM mode offers additional integrity checks. SHA384 ensures that messages are authenticated and have not been tampered with, providing a comprehensive security framework.

By capturing and analyzing the "Server Hello" packet with Wireshark, we can verify that **TLS 1.2** is being used along with the **TLS_AES_256_GCM_SHA384** cipher suite. These elements are critical in confirming that our OpenVPN connection is secure by the TLS handshake which provide authentication, encryption, and integrity verification for packet transmit during the session. By confirming these components through packet analysis, we can confidently assert that our OpenVPN connection is secure, with encrypted and authenticated data transmission, protecting the communication from potential threats.

7.5 Conclusion

In summary, the extensive software testing has successfully identified and addressed potential errors in the application. This testing has significant enhanced the system’s quality and reliability. The testing phase has proven to be a crucial component of the software development lifecycle, ensuring that the final product adheres to the necessary standards and specifications.

Chapter 8

CONCLUSION AND RECOMMENDATION

8.1 Conclusion

This project has successfully developed a reliable VPN application for Android platform. During the project, few challenges such as VPN module integration, system architecture design had been faced during early stage of the project. However, with careful planning, troubleshooting, and problem-solving, these challenges were effectively addressed, ensuring the smooth progress and successful completion of the application. The final product meets the intended objectives, providing a secure and user-friendly VPN solution for Android users.

The first objective, implement and integration of robust encryption standards and reliable protocols has been achieved through deployment of VPN using OpenVPN protocol and the configuration of the VPN with using strong encryption standards. This ensures that user data and activities is securely transmitted and protected from potential threat by providing a high level of security and privacy for end users

The second objective which was enhance user authentication mechanisms by integrated with two-factor-authentication has been successfully accomplished by integrating the two-factor-authentication(2FA). The integration of 2FA provides an additional layer of security and ensuring that only authorized users able to access our VPN application by requiring second form of verification in addition to the standard username and password. This mechanism has significantly enhanced the overall security of the application and have capabilities protecting against unauthorized access.

The final objective was optimized user interface design by providing user friendly interface has also been successfully achieved. The application features a streamlined and user-friendly interface that allows users to easily navigate and interact with the VPN features within the application his focus on user-centered design enhances accessibility and ensures that users can efficiently manage their VPN connections with minimal effort which contributing to a positive overall experience.

8.2 Limitations

While the project has successfully achieved its primary objectives, there are still few limitations.

8.2.1 Geographical Limitations Due to Insufficient Server Coverage

The VPN application have geographical limitations allow user to select due to insufficient server coverage. The application currently only provided service at 3 countries. The limited server availability may cause reduced connection speeds or face difficulty accessing the VPN altogether. This limitation restricts the global accessibility and effectiveness of the VPN when particularly in regions where internet censorship or restrictions are prevalent, and where a reliable VPN connection is most needed.

8.2.2 Limited Customization Options

The second limitation comes to the customization options of the VPN application. The application did not provide much space allow user to make their own customization such as encryption protocols, customizing DNS setting or other VPN features. The project is configured to follow a set of fixed predefined configuration which reduce the flexibility of the application. This limitation could potentially restrict the application's appeal to tech-savvy users who desire greater control over their VPN configurations. Although the application using fixed configuration but all the configuration claim to provide the most optimal balance between security, performance, and ease of use, while users may not have the ability to customize settings extensively due to this, the default configurations have been chosen to meet the needs of most users.

8.2.3 Server Overload Due to Low-Spec Infrastructure

The VPN application might encounter issues related to server overload especially during peak usage times. This is due to the low-spec infrastructure of the server selected due to lack of sufficient budget. As a result, users may experience slower connection speeds, increased latency, or even temporary connection drops. This limitation impacts the overall performance and reliability of the VPN service, especially as the user base grows.

8.3 Recommendation for Future Work

There are still have a lot of space for future work that could exploit from existing project.

8.3.1 Provide more countries coverage

One of the key areas for improvement is expanding the server network to cover more countries. By increasing number of servers in diverse geographic locations can enhance global accessibility of the VPN but also can improve speeds and reliability for user in underserved regions. This improvement able make the application more attractive to broader user base and help mitigate current limitation of insufficient server coverage

8.3.2 Enhance the current cloud platform infrastructure

To address issues related to server overload and performance, it is highly recommended to enhance current cloud platform infrastructure such as add more server instance within same region to provide VPN services in terms to achieve high availability and implement load balancing to reduce the traffic stress and provide capabilities of disaster recovery. This would lead to improved connection stability, faster speeds, and better overall user experience especially during peak usage times

8.3.3 Integration of other VPN protocols

The current project only supports for one VPN protocols which is OpenVPN protocol. By integrating additional VPN protocols support such as Wire Guard, IKEV2 or SSTP could provide users more option, better experience, and potentially better performance in certain scenarios. By Offering a variety of protocols would allow users to choose the one that best fits their needs, whether they prioritize speed, security, or compatibility with different network environments.

8.3.4 Provide support for customization

Introducing more customization options would significantly enhance the user experience, particularly for advanced users. By allowing users to modify settings such as encryption levels, protocol selection, DNS configurations, and connection rules would give them greater control over their VPN experience. This flexibility could cater to a wider range of user preferences and use cases, making the application more versatile and appealing to a broader audience.

Reference

1. Gershwin, A. A. (2019, June 14). The evolution of VPNs: From business security to privacy protection. Bitcoin Insider. https://www.bitcoininsider.org/article/70244/evolution-vpns-business-security-privacy-protection
2. Shweta. (2024, April 12). What is a remote access VPN? Forbes. https://www.forbes.com/advisor/business/what-is-remote-access-vpn/
3. Paloalto. (2024, April 12). What is a site-to-site VPN? Palo Alto Networks. https://www.paloaltonetworks.com/cyberpedia/what-is-a-site-to-site-vpn
4. Fortinet. (2024, April 12). What is peer-to-peer (P2P) VPN? Are P2P VPNs safe? Fortinet. https://www.fortinet.com/resources/cyberglossary/peer-to-peer-p2p-vpn
5. OpenVPN. (2018, March 6). OpenVPN brings tools for businesses to securely and easily access Microsoft Cloud Services. PR Newswire. https://www.prnewswire.com/news-releases/openvpn-brings-tools-for-businesses-to-securely-and-easily-access-microsoft-cloud-services-300536944.html
6. BlessingGeek, B. (2019, February 13). WireGuard: Erste Testversion für Windows verfügbar. RandomBrick.de. https://www.randombrick.de/wireguard-erste-testversion-fuer-windows-verfuegbar/
7. OpenVPN. (2018, March 6). OpenVPN brings tools for businesses to securely and easily access Microsoft Cloud Services. PR Newswire. https://www.prnewswire.com/news-releases/openvpn-brings-tools-for-businesses-to-securely-and-easily-access-microsoft-cloud-services-300536944.html
8. Eddy, M. S. (2023, April 12). My start in Surfshark VPN. PCMag Australia. https://au.pcmag.com/vpn/62274/surfshark-vpn
9. i2Coalition. (2020). I2COALITION member spotlight Q&A: Hotspot Shield. i2Coalition. https://i2coalition.com/i2coalition-member-spotlight-qa-hotspot-shield/
10. Cisco. (2024). What is a virtual private network (VPN)? Cisco. https://www.cisco.com/c/en/us/products/security/vpn-endpoint-security-clients/what-is-vpn.html
11. Cloudflare. (2024). What is IPsec? | How IPsec VPNs work. Cloudflare. https://www.cloudflare.com/learning/network-layer/what-is-ipsec/

12. Fortinet. (2024a). What is a remote access VPN? Fortinet. https://www.fortinet.com/resources/cyberglossary/remote-access-vpn
13. Fortinet. (2024b). What is peer-to-peer (P2P) VPN? Are P2P VPNs safe? Fortinet. https://www.fortinet.com/resources/cyberglossary/peer-to-peer-p2p-vpn
14. Microsoft. (2024). What is a VPN? Why should I use a VPN? Microsoft Azure. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-vpn/
15. OpenVPN. (2022). What is OpenVPN? OpenVPN. https://openvpn.net/faq/what-is-openvpn/
16. OpenVPN. (2024). What is a VPN? VPN definition. OpenVPN. https://openvpn.net/what-is-a-vpn/
17. Paloalto. (2024). Site-to-site VPN overview. Palo Alto Networks. https://docs.paloaltonetworks.com/network-security/ipsec-vpn/administration/get-started-with-ipsec-vpn-site-to-site/site-to-site-vpn-overview
18. Tailscale. (2024). What you need to know about Internet Protocol Security (IPsec). Tailscale. https://tailscale.com/learn/ipsec
19. Zola, A. (2021). What is L2TP and how does it work? Networking. https://www.techtarget.com/searchnetworking/definition/Layer-Two-Tunneling-Protocol-L2TP
20. Awati, R., Bernstein, C., & Cobb, M. (2024). What is the Advanced Encryption Standard (AES)? TechTarget Security. https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard
21. Cobb, M. (2023). What is triple DES and why is it being disallowed? TechTarget Security. https://www.techtarget.com/searchsecurity/tip/Expert-advice-Encryption-101-Triple-DES-explained
22. Dmitry. (2023). VPN protocols compared: Advantages and disadvantages. WebsiteHosting.com. https://websitehosting.com/blog/vpn-protocols-compared-advantages-and-disadvantages/
23. Donenfeld, J. A. (2024). Fast, modern, secure VPN tunnel. WireGuard. https://www.wireguard.com/
24. Jack. (2023). What is WireGuard? A new VPN protocol explained. Cybernews. https://cybernews.com/what-is-vpn/wireguard-protocol/

25. Miklos. (2024). IKEv2 VPN protocol explained: What it is and how it works. Privacy Affairs. https://www.privacyaffairs.com/ikev2-vpn-protocol/
26. Nagaraj, K. (2023). Understanding ChaCha20 encryption: A secure and fast algorithm for data protection. Medium. https://cyberw1ng.medium.com/understanding-chacha20-encryption-a-secure-and-fast-algorithm-for-data-protection-2023-a80c208c1401
27. Perimeter 81. (2023). What is secure Socket Tunneling Protocol (SSTP)? Perimeter 81. https://www.perimeter81.com/blog/network/secure-socket-tunneling-protocol
28. Proofpoint. (2024). What is SSTP? - VPN protocol. Proofpoint. https://www.proofpoint.com/us/threat-reference/sstp
29. Williams, M. (2020). Hola VPN review. TechRadar. https://www.techradar.com/reviews/hola-free-vpn
30. Webster, E., & Contributor, T. (2021). What is the Extensible Authentication Protocol (EAP)? TechTarget Security. https://www.techtarget.com/searchsecurity/definition/Extensible-Authentication-Protocol-EAP
31. Okta. (2023). HMAC (hash-based message authentication codes) definition. Okta. https://www.okta.com/identity-101/hmac/
32. Millares, L. (2024). Hotspot Shield VPN review 2024: Pricing, features & security. TechRepublic. https://www.techrepublic.com/article/hotspot-shield-review/
33. Gillis, A. S. (2022). What is Diffie-Hellman key exchange? TechTarget Security. https://www.techtarget.com/searchsecurity/definition/Diffie-Hellman-key-exchange
34. Cobb, M. (2021). What is the RSA algorithm? Definition from SearchSecurity. TechTarget Security. https://www.techtarget.com/searchsecurity/definition/RSA
35. Tomaschek, A. (2024). Surfshark VPN gives you blazing speeds at competitive pricing. CNET. https://www.cnet.com/tech/services-and-software/surfshark-vpn-review/
36. NIKI, S. (2023). VPN growth highlights global crackdown on internet freedom. Nikkei Asia. https://asia.nikkei.com/Business/Technology/VPN-growth-highlights-global-crackdown-on-internet-freedom
37. Sablah, W. (2024). Worst VPN warning list: Avoid VPN scams in 2024. Cloudwards. https://www.cloudwards.net/worst-free-vpn/

38. Skendzic, A., & Kovačić, B. (2017). Open source system OpenVPN in a function of Virtual Private Network. ResearchGate. https://www.researchgate.net/publication/317159271_Open_source_system_OpenVPN_in_a_function_of_Virtual_Private_Network
39. Iqbal, M. I. M. I., & Riadi, I. R. (2019). Analysis of security virtual private network (VPN) using OpenVPN. ResearchGate. https://www.researchgate.net/publication/333198144_Analysis_of_Security_Virtual_Private_Network_VPN_Using_OpenVPN
40. McGrew, D., & Viega, J. (2005). The Galois/Counter mode of operation (GCM). NIST. https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-revised-spec.pdf