BSCS FINAL PROJECT

A Torrent Client (Named Magneto)



Project Advisor

**Dr. GHULAM MUSTAFA**

Presented by:

**Group ID: S25BS006**

Student Reg# Student Name

**L1F20BSCS0603 Ali Anser**

**L1S21BSCS0065 Minhaal Ali Hafeez**

**L1S23BSCS0257 Faheem Raza**

**Faculty of Information Technology & Computer Science**

**University of Central Punjab**

# Software Requirements Specification

# Version # 1

# A Torrent Client (Named Magneto)

# Advisor: DR. Ghulam Mustafa

# Group S25BS006

|  |  |
| --- | --- |
| Member Name | Primary Responsibility |
| **Ali Anser** | **BitTorrent Protocol Implementation and VPN Integration** |
| **Minhaal Ali Hafeez** | **Graphical User Interface** |
| **Faheem Raza** | **Parsing** |

Table of Contents

Table of Contents [i](#__RefHeading___Toc193286255)

Revision History [ii](#__RefHeading___Toc193286256)

Previous Phases Feedback [ii](#__RefHeading___Toc193286257)

Abstract [iii](#__RefHeading___Toc193286258)

1. Introduction and Background [1](#__RefHeading___Toc193286259)

1.1 Product (Problem Statement) [1](#__RefHeading___Toc193286260)

1.2 Background [1](#__RefHeading___Toc193286261)

1.3 Scope [1](#__RefHeading___Toc193286262)

1.4 Objective(s)/Aim(s)/Target(s) [1](#__RefHeading___Toc193286263)

1.5 Challenges 2

1.6 Learning Outcomes 2

1.7 Nature of End Product 2

1.8 Completeness Criteria 2

1.9 Business Goals 3

1.10 Related Work/ Literature Survey/ Literature Review 3

1.11 Document Conventions 3

2. Overall Description 4

2.1 Product Features 4

2.2 User Classes and Characteristics 4

2.3 Operating Environment 5

2.4 Design and Implementation Constraints 5

2.5 Assumptions and Dependencies 5

3. Functional Requirements 6

3.1 Add Torrent via File or Magnet Link 7

3.2 View Torrent Download Progress 7

3.3 Enable or Disable VPN Connection 8

3.4 Configure Torrent Download Settings 8

3.5 Verify File Integrity 9

3.6 View Peer Details 10

3.7 Set File Priority 10

3.8 Requirements Analysis and Modeling 11

4. Nonfunctional Requirements 13

4.1 Performance Requirements 13

4.2 Safety Requirements 13

4.3 Security Requirements 13

4.4 Additional Software Quality Attributes 13

5. Other Requirements 13

6. Revised Project Plan 14

7. References 14

Appendix A: Glossary 15

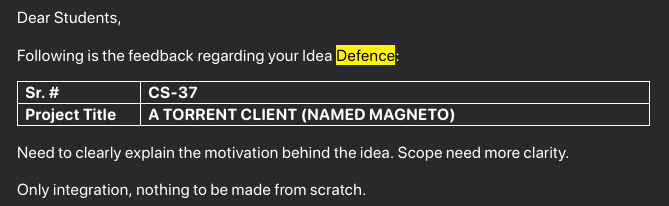
Appendix B: IV & V Report 16

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

Previous Phases Feedback

**Idea Defence Feedback (Screenshot)**

****

# Abstract

Magneto is a cross-platform torrent client with a built-in VPN module, designed to offer secure, private, and seamless peer-to-peer file sharing. Unlike existing torrent clients that require external VPN configurations, Magneto embeds VPN functionality directly into the application, eliminating complexity and enhancing user privacy. Attempts to modify existing clients or integrate torrenting and VPN libraries proved unfeasible due to architectural limitations, necessitating a from-scratch implementation. Magneto aims to fill a gap in the market by combining torrenting with integrated privacy features, catering to users who value both functionality and anonymity.

# Introduction and Background

Magneto is a torrent client built from the ground up with integrated VPN functionality to provide a private and secure torrenting experience. While torrenting remains a popular method for distributing large files efficiently, it inherently exposes user IP addresses to peers, compromising privacy. Magneto solves this by embedding a VPN module directly within the application, streamlining configuration and eliminating the need for third-party VPN tools.

## Product (Problem Statement)

Magneto is a cross-platform torrent client with built-in VPN integration. It provides users with a seamless way to share files securely over the BitTorrent protocol without relying on external VPN software.

## Background

Torrenting uses peer-to-peer communication, where users download pieces of files from multiple sources. This efficient model comes with significant privacy risks, such as exposure to surveillance and ISP throttling. While VPNs can mitigate these risks, configuring them separately adds complexity. Current torrent clients are not designed for such integration, necessitating a new solution that merges both functionalities.

## Scope

This SRS covers the development of Magneto, including implementation of the BitTorrent protocol, VPN integration, a desktop GUI, torrent/magnet parsers, performance optimization, and complete documentation. The scope includes development, testing, and deployment of a functional cross-platform client, but excludes mobile platform support and third-party plugin extensibility.

## Objective(s)/Aim(s)/Target(s)

* Implement a BitTorrent-compliant torrent client from scratch.
* Integrate a secure and robust VPN module within the application.
* Design a clean, user-friendly desktop interface.
* Support torrent file and Magnet URL parsing.
* Maintain high performance with minimal resource usage.
* Ensure complete platform independence across Windows, Linux, and macOS.
* Eliminate dependency on third-party VPN software.

## Challenges

* Low-level implementation of the BitTorrent protocol.
* Seamless and secure VPN integration.
* Ensuring stable cross-platform GUI performance.
* Avoiding IP leaks under all network conditions.
* Maintaining high download/upload throughput while tunneling traffic.
* Complying fully with protocol standards.

## Learning Outcomes

* Experience in cross-platform development using Qt.
* Deep understanding of VPN technologies and BitTorrent internals.
* Proficiency in secure network programming and cryptography.
* Skills in performance optimization and testing under load.
* Experience with collaborative development and documentation practices.

## Nature of End Product

A fully functional torrent client with integrated VPN, offering secure, anonymous, and efficient P2P file sharing, packaged with technical and user documentation and deployable across major desktop operating systems.

## Completeness Criteria

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Criteria** | **Weightage %** |
| 1 | Desktop GUI | 10 |
| 2 | **VPN Integration and Security Features** | 20 |
| 3 | **Core BitTorrent Protocol Implementation** | 40 |
| 4 | **Torrent File and Magnet URL Parsers** | 15 |
| 5 | Documentation | 5 |
| 6 | Performance | 10 |

## Business Goals

* Provide a privacy-first alternative to existing torrent clients.
* Tap into the niche of privacy-conscious torrent users.
* Reduce friction for users who currently rely on separate VPN tools.
* Set a precedent for integrated privacy in P2P applications.

## Related Work/ Literature Survey/ Literature Review

Clients like uTorrent and qBittorrent provide robust torrenting features but lack native privacy safeguards. Users depend on external VPNs, leading to poor UX and potential misconfiguration. Research into P2P privacy confirms the need for embedded security features to increase user safety and adoption. Magneto builds on these insights to deliver a secure, all-in-one client.

## Document Conventions

* Monospaced text indicates technical terms or code components.
* Bold is used for emphasis on important terms.

# Overall Description

## Product Features

* **BitTorrent Protocol Implementation**: Supports all core functionalities including peer discovery, piece exchange, and seeding.
* **Integrated VPN Module**: Provides encrypted, anonymous torrenting without external VPN tools.
* **Magnet and Torrent File Support**: Parses and handles .torrent files and Magnet links seamlessly.
* **Cross-Platform Desktop GUI**: User-friendly interface built with Qt, supporting torrent management and VPN settings.
* **Download/Upload Management**: Handles multiple concurrent torrents with speed control and progress tracking.
* **Privacy Protection**: Ensures no IP leaks, with automatic VPN reconnection and kill-switch mechanisms.
* **Performance Optimization**: Efficient resource usage and bandwidth management for high-speed transfers.

## User Classes and Characteristics

* **General Users**: Privacy-focused torrent users with basic technical knowledge. They will primarily use torrenting and VPN toggle features.
* **Advanced Users**: Tech-savvy individuals who may adjust VPN settings, monitor peer details, and tweak performance options.
* **Developers and Contributors**: May inspect or extend the codebase. Require access to detailed technical documentation and system architecture.

Favored users are General and Advanced Users, as they form the primary user base. Developer satisfaction is secondary but still supported.

## Operating Environment

**Hardware**: x86\_64 architecture systems with at least 2GB RAM and 200MB storage.

**Operating Systems**:

* Windows 8 or later
* Linux distributions (Ubuntu 20.04+, Fedora, Arch)
* macOS 11 or later**.**

**Dependencies:**

* Qt Framework for GUI
* OpenSSL for encryption
* Git for version control

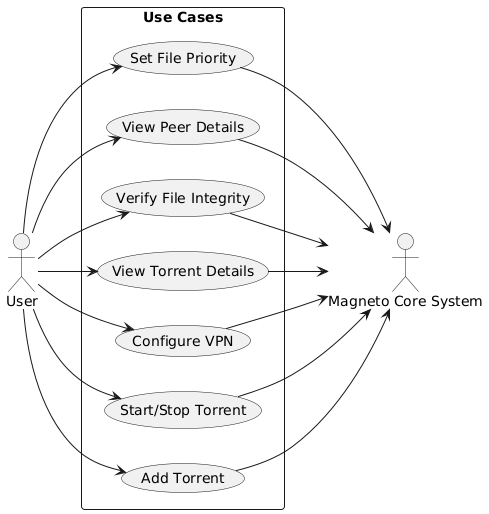
## Design and Implementation Constraints

* **Language**: Development will be in C++ with Qt for GUI.
* **VPN Integration**: Must use secure, well-supported VPN protocol.
* **Cross-Platform Compatibility**: GUI and networking stack must behave consistently across different operating systems.
* **Security**: High priority on leak-proof traffic routing and safe data handling.
* **Licensing**: Only use libraries with permissive licenses (e.g., MIT, BSD, Apache 2.0).

## Assumptions and Dependencies

The development of Magneto assumes that users will have administrative access to their systems to permit VPN tunneling and network-level configurations. It is also assumed that the target operating systems—Windows, Linux, and macOS—will provide stable and consistent support for required system libraries and networking features. The project depends on the availability and reliability of the Qt framework for cross-platform GUI development, as well as the continued compatibility of encryption and networking libraries across supported platforms. Additionally, successful implementation relies on stable internet connectivity for real-time peer-to-peer communication and version control operations through Git. Any significant changes to these assumptions or external dependencies may impact development timelines or functional integrity.

# Functional Requirements



## Add Torrent via File or Magnet Link

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-1 | |
| **Purpose** | | To allow the user to add a torrent for downloading using either a .torrent file or a magnet link | |
| **Priority** | | High | |
| **Pre-conditions** | | Application is running, User is connected to internet | |
| **Post-conditions** | | Torrent is added to the download list, Download starts if peers are available | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User clicks "Add Torrent" button | | System opens file chooser and input field for magnet link |
| **2** | User selects a .torrent file or pastes URL | | System validates the input and parses torrent metadata |
| **3** | User confirms | | System adds torrent to the list and initiates connection to peers |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User inputs invalid file or magnet URL | | System displays an error message |
| **2** | User cancels input | | System closes dialog and returns to main interface |

Table 1: UC-1

## View Torrent Download Progress

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-2 | |
| **Purpose** | | To allow the user to monitor the real-time status and progress of active torrent downloads | |
| **Priority** | | High | |
| **Pre-conditions** | | At least one torrent is currently being downloaded | |
| **Post-conditions** | | The user sees updated download progress and related metrics for each torrent | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User navigates to the active torrents list | | System displays all current torrents being downloaded |
| **2** | — | | System shows progress bar, download/upload speeds, peer count, and ETA |
| **3** | — | | System updates progress in real time as data is received |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | Torrent is paused | | System stops updating progress and marks status |
| **2** | Torrent download fails | | System notifies user of failure and logs reason |

Table 2: UC-2

## Enable or Disable VPN Connection

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-3 | |
| **Purpose** | | To allow the user to control the VPN module, enabling or disabling the secure connection as needed | |
| **Priority** | | High | |
| **Pre-conditions** | | Application is running, VPN module is configured | |
| **Post-conditions** | | VPN connection status is updated according to user action, All torrent traffic is routed through the VPN when enabled | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User clicks “Enable VPN” | | System initiates VPN connection |
| **3** | User clicks “Disable VPN” | | System terminates VPN connection |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | VPN connection fails to establish | | System displays error and suggests retry or troubleshooting steps |
| **2** | VPN disconnects unexpectedly | | System attempts automatic reconnection or alerts user |

Table 3: UC-3

## Configure Torrent Download Settings

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-4 | |
| **Purpose** | | To allow the user to customize download options such as save location, bandwidth limits, and connection preferences | |
| **Priority** | | Medium | |
| **Pre-conditions** | | Application is running, User has atleast one torrent added | |
| **Post-conditions** | | User’s download preferences are saved and applied to current and future torrents | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User opens settings or preferences menu | | System displays configurable download options |
| **3** | User changes save path, bandwidth limits, or max peers | | System validates and saves the new settings |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User inputs invalid path or limit | | System displays error message and requests correction |
| **2** | VPN disconnects unexpectedly | | User cancels settings change |

Table 4: UC-4

## Verify File Integrity

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-5 | |
| **Purpose** | | To allow the user to manually or automatically verify the integrity of downloaded files using piece hashes | |
| **Priority** | | Medium | |
| **Pre-conditions** | | At least one torrent has completed downloading | |
| **Post-conditions** | | System confirms file integrity or flags corrupted/incomplete pieces | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User selects “Verify Integrity” from torrent menu | | System begins hash check of downloaded pieces |
| **3** | System computes and compares piece hashes | | System reports verification results (pass/fail) |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | File is missing or altered | | System flags specific pieces/files as corrupted |
| **2** | User cancels the process | | System halts verification and restores previous state |

Table 5: UC-5

## ****View Peer Details****

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-6 | |
| **Purpose** | | To display detailed information about peers connected to a torrent | |
| **Priority** | | Medium | |
| **Pre-conditions** | | Torrent is actively downloading or seeding, Peers are connected to the client | |
| **Post-conditions** | | User sees peer-specific data such as IP, country, client name, and speed | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User selects a torrent | | System loads and displays peer list |
| **3** | User expands “Peer Details” view | | System shows IP, port, client, country, download/upload speed |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | No peers connected | | System shows “No active peers” message |
| **2** | Peer disconnects mid-view | | System dynamically updates peer list in real time |

Table 6: UC-6

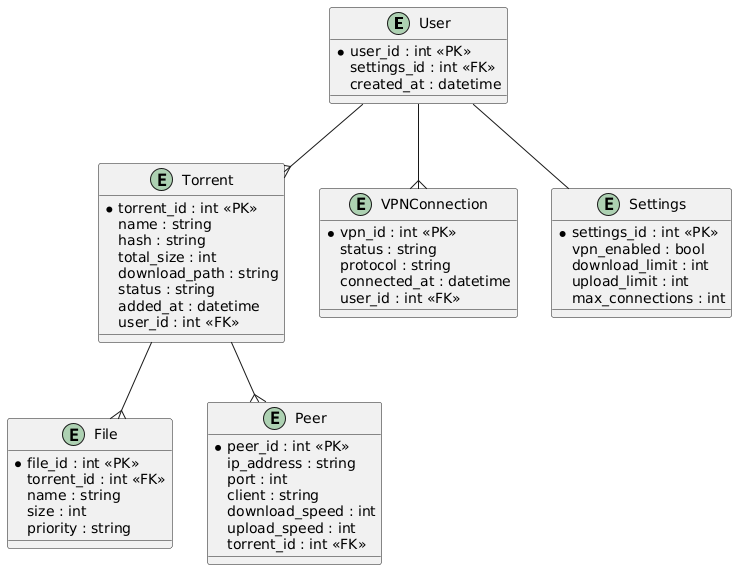
## ****Set File Priority****

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-7 | |
| **Purpose** | | To allow users to assign different download priorities to individual files within a torrent | |
| **Priority** | | Medium | |
| **Pre-conditions** | | A multi-file torrent has been added to the client | |
| **Post-conditions** | | Selected files are downloaded according to assigned priorities (e.g., high, normal, low, or skipped) | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User opens file list for a torrent | | System displays all files in the torrent |
| **3** | User sets priority for each file | | System marks files accordingly (high/normal/low/skipped) |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | User marks a file as “Do not download” | | System skips downloading pieces related to that file |
| **2** | User changes file priority mid-download | | System adjusts scheduling dynamically based on new priorities |

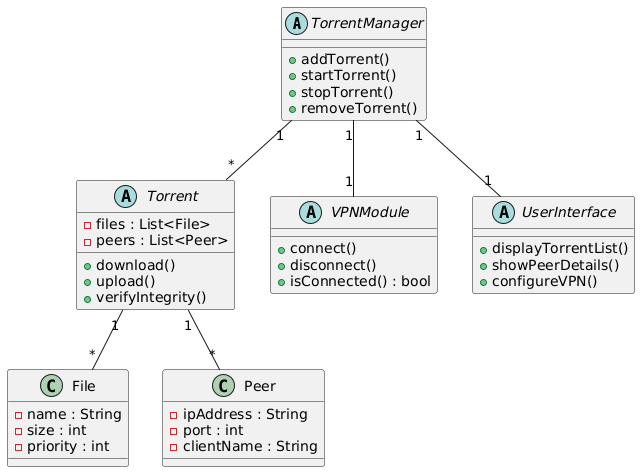
Table 7: UC-7

## Requirements Analysis and Modeling

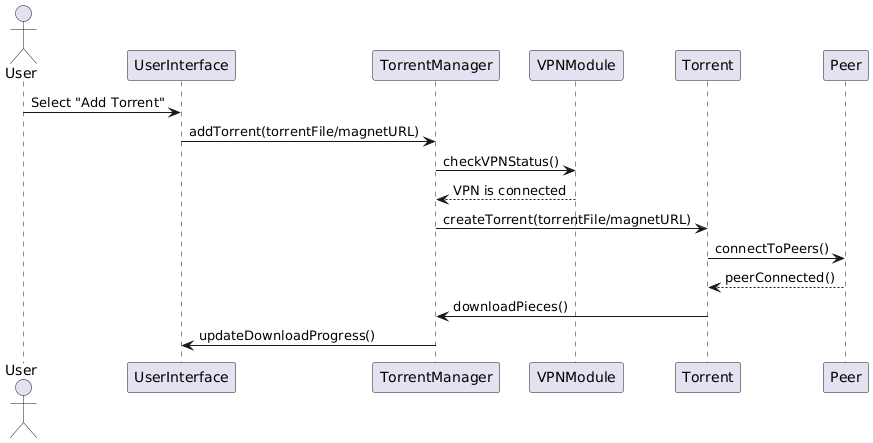
**Entity Relationship Diagram:**



**Abstract Class Diagram:**



**Sequence Diagram:**

**

# Nonfunctional Requirements

## Performance Requirements

Magneto must deliver fast and efficient torrent downloads and uploads, even when multiple torrents are active simultaneously. The integration of the VPN should not noticeably degrade network throughput; the system should optimize bandwidth usage to maintain speeds comparable to traditional torrent clients without built-in VPNs. Memory and CPU consumption must be kept within reasonable limits to ensure smooth operation on typical user machines, avoiding excessive resource drain during peak activity. Responsiveness in the user interface is critical, with minimal delays when interacting with torrents, peers, or VPN settings.

## Safety Requirements

The system must safeguard against any misuse or vulnerabilities that could compromise the user’s device or data. Since Magneto handles network traffic and file transfers, it should prevent unauthorized access or injection attacks. Any failures in VPN connectivity should trigger safeguards that halt torrent activity to avoid accidental IP exposure. Users must be protected from accidental data loss by ensuring proper file handling and integrity verification before marking downloads as complete. The client must comply with applicable software safety and privacy regulations, especially regarding user data protection.

## Security Requirements

Security is paramount for Magneto, given the sensitive nature of torrenting and VPN use. The application must implement strong encryption for all VPN traffic to prevent interception or IP leaks. User authentication may be required for accessing VPN configuration settings to prevent unauthorized tampering. All network communications must adhere to secure protocols, and any data stored locally should be protected against unauthorized access. The software should regularly audit and update its security features to address emerging threats and comply with privacy laws and standards relevant to its user base.

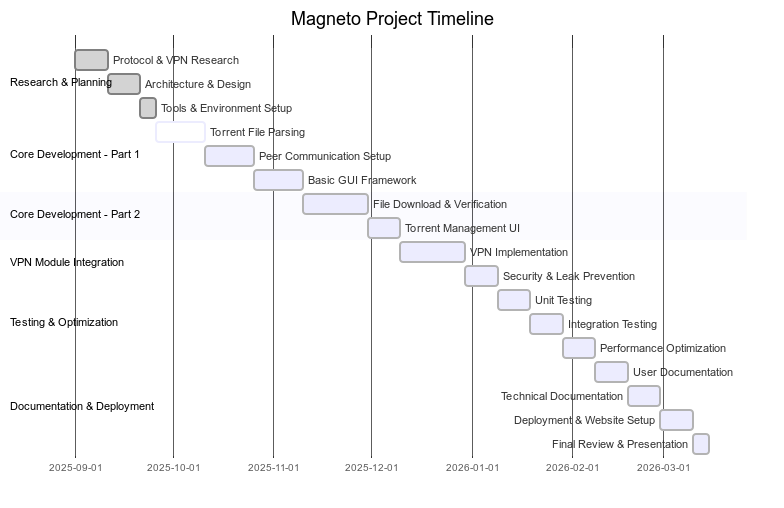
## Additional Software Quality Attributes

Magneto should be highly portable and compatible across major operating systems such as Windows, macOS, and Linux, ensuring consistent functionality and interface behavior. Usability is critical; the interface must balance simplicity for novice users with enough control for advanced users to customize torrenting and VPN settings. The system should be robust, gracefully handling errors and network interruptions without crashing or corrupting data. Maintainability is essential; the codebase should be clean, modular, and well-documented to facilitate future updates and debugging. The software must also be testable, allowing thorough verification of both core torrent functions and security features.

# Other Requirements

Magneto does not rely on any complex database system but will need a lightweight local storage solution to manage torrent metadata, user preferences, and VPN configurations securely. The software must interface smoothly with underlying network hardware and standard operating system networking stacks without requiring special drivers or elevated privileges beyond those necessary for VPN operations. Internationalization support should be considered for future versions, with the ability to add multiple languages and locale settings to widen user accessibility. Legal compliance is critical, particularly in respecting copyright laws and privacy regulations across jurisdictions where the software is distributed. The project aims to produce reusable components where feasible, especially in the VPN and torrent protocol modules, to allow future projects or extensions to build upon this foundation with minimal redevelopment.

# Revised Project Plan



# References

* Cohen, Bram. BitTorrent Specification. Theory.org Wiki. Available at: <https://wiki.theory.org/BitTorrentSpecification>
* NordVPN. Interesting VPN Uses. NordVPN Blog. Available at: <https://nordvpn.com/blog/interesting-vpn-uses/>s
* The Tor Project. Miscellaneous Topics: VPN and Anonymity. Tor Support. Available at: <https://support.torproject.org/misc/misc-4/>
* Smith, John, et al. "Research on VPN and Privacy in Peer-to-Peer Networks." Journal of Secure Computing, vol. 15, no. 3, 2023, pp. 102-115.
* uTorrent Official Documentation. Available at: https://www.utorrent.com/support
* qBittorrent Official Documentation. Available at: https://www.qbittorrent.org/documentation/
* Qt Project. Qt Framework Documentation. Available at: [https://doc.qt.io](https://doc.qt.io/)

Appendix A: Glossary

* **BitTorrent Protocol**: A peer-to-peer (P2P) file-sharing protocol that enables efficient distribution of large files by splitting them into smaller pieces.
* **Client**: Software that accesses a service made available by a server. In this context, the torrent client handles file sharing.
* **Magnet Link**: A type of hyperlink that contains a unique identifier for a torrent, allowing clients to locate peers without needing a .torrent file.
* **Peer**: A participant in the BitTorrent network that both downloads and uploads pieces of a file.
* **Seeding**: The act of sharing a fully downloaded file with other peers on the network.
* **Swarm**: The group of all peers sharing a specific torrent.
* **VPN (Virtual Private Network)**: A service that encrypts a user’s internet connection and masks their IP address to enhance privacy and security.
* **Encryption**: The process of encoding data to prevent unauthorized access.
* **Bencode**: A data encoding format used by the BitTorrent protocol for storing and transmitting information like torrent metadata.
* **IP Leak**: The unintended exposure of a user’s IP address despite the use of privacy tools like VPNs.
* **Cross-Platform**: Software that is compatible with multiple operating systems such as Windows, macOS, and Linux.

Appendix B: IV & V Report

**(Independent verification & validation)**

**IV & V Resource**

Name Signature

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S#** | **Defect Description** | **Origin Stage** | **Status** | **Fix Time** | |
| **Hours** | **Minutes** |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| … |  |  |  |  |  |

**Table 8: List of non-trivial defects**