****

**Software Engineering Department  
ORT Braude College**

**Capstone Project Phase Ae**

**Data Pulse**

**D-9 Decentralized P2P File Sharing Platform**

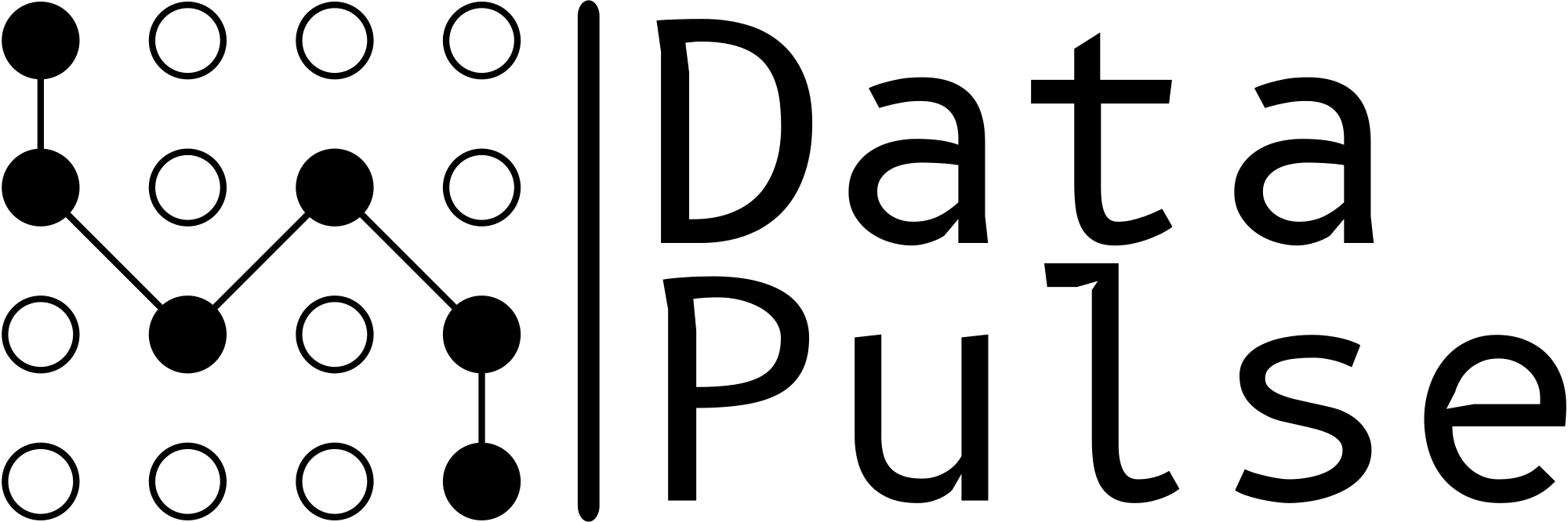
**Submitted by:**

**Ofek Ben Avraham**

**Rotem Porat**

**Supervised by:**

**Ronen Zilber**

****

[**Abstract 4**](#_55ijwo9wiw9l)

[**Introduction 4**](#_bwk9j5ow131b)

[**Related Work 5**](#_gxped0czaseg)

[**Background 5**](#_ivkxvxrw7ia8)

[1. BitTorrent and its development 6](#_s8ddktle1hhs)

[1.1. Torrent file 6](#_u5t1b13dvugg)

[2. Criteria for Evaluating Distributed File Sharing Platforms 7](#_f1nfs7di0epn)

[3. Beyond File Sharing: Other Applications of Distributed Sharing 7](#_vphwn72983bd)

[4. Common Terms and Protocols 8](#_9pv1lkxw8bbr)

[4.1. Communication protocols 8](#_pr7573xupreg)

[4.2. Encryption Protocols 9](#_p4wkq3lyo19u)

[4.3. Advanced Protocol Features 9](#_1gynny1phr98)

[5. Error Handling and Redundancy in Distributed File Sharing 9](#_6edwor78bp2d)

[5.1.Error Handling Mechanisms 9](#_va2p4k3wv9lf)

[5.2. Redundancy Strategies 9](#_g5ylzbqak0hk)

[5.3. Data Availability and Uptime 10](#_lphtqulis8vm)

[6. Network Traffic Optimization in Distributed File Sharing 10](#_banlv0com29m)

[6.1. Congestion Control Mechanisms 10](#_l4nrr3vwoo35)

[6.2. Bandwidth Management 11](#_x9j6oxoqjet)

[6.3. Latency Reduction Techniques 11](#_t1qo7ahgkn1q)

[**Expected Achievements 11**](#_j32sskhle5sa)

[1.Results 11](#_bn983q6ypsn)

[2. Unique features 12](#_mkbfk5m88tdd)

[3. Success criteria 12](#_spqa6hpaname)

[**Engineering Process 13**](#_g5eqat1nhb7h)

[1. Research/Engineering Process 13](#_6tj80udh4n3w)

[1.2. Research - Distributed File Sharing 13](#_8gbhw8ehovgf)

[2. Constraints and Challenges - Distributed File Sharing 13](#_ae0du6hlrzl2)

[2.1. Scalability and Efficiency 14](#_wwiwhj3joz6v)

[2.2. Security Concerns 14](#_9khiqk4h77aj)

[2.3. User Privacy 14](#_knzlr8t3fb0x)

[3. Central Requirements of the Project 14](#_bie991x4qx59)

[3.1. Functional requirements 14](#_ui76qfyy2amv)

[3.2. Non-functional requirements 15](#_v7o6nfqbkcxg)

[4. Description of the Requirements Collection 16](#_cbhrwj15ke4k)

[5. Motivation 16](#_u33z4scbpk68)

[6. Suggested Hardware and Software Tools for the development 17](#_5m6bz57gat84)

[7. Algorithms and Data Structures 17](#_uumd1ud2epcb)

[7.1. Peer-to-Peer Network Model 17](#_60lnaguhkwcv)

[7.2. Segmented file transfer 17](#_k0af37y7i4ti)

[7.3. Encryption and security 17](#_i2bmlenrsm3l)

[7.4. Data Structures 17](#_oso5411m863a)

[8. Future User Interface Design 18](#_bcevtvtafjz5)

[File Selection Interface 19](#_mvrw6ioh2p0b)

[9. System Architecture 19](#_ma49833h7wx4)

[10. Diagrams 21](#_bsufdq471i5p)

[10.1. Use Case 21](#_vhnv5gjyu3px)

[10.2. Sequence Diagram 22](#_iw6ogdm7kise)

[10.2.1. Sequence Diagram of file upload 22](#_tt3zazydtjzk)

[10.2.2. Sequence Diagram of file download 23](#_wprv3a8i09zx)

[10.2.3. Sequence Diagram of file search 24](#_hzm9217w1asz)

[10.3. Activity Diagram 25](#_vd9t0hevya4q)

[10.3.1. Expanded Activity Diagram for “check storage available” 26](#_i8yoz9f46iz0)

[10.3.2. Expanded Activity Diagram for “locate file segments” 27](#_6hw8s4m0i7nz)

[**Verification and Evaluation 28**](#_a2iu7pjs19ji)

[1. Test strategy 28](#_mk0bdn3vzpwo)

[2. Constraints and assumptions 29](#_jaypb8zdkd7q)

[3. Work environment testing 29](#_1u1ynu1rlon7)

[3.1. Hardware 29](#_zavinhc6ivvk)

[3.2. Software 29](#_s2mefgq9t0p6)

[3.3. Security 29](#_d5ocstmqtsyv)

[3.4. User Review 29](#_fuehgz1xbfl6)

[4. Test cases and execution 30](#_fr08d57zyc7p)

[4.1. Unit tests 30](#_dd9nfafna8ak)

[4.2. Integration tests 30](#_mclw6m92smpb)

[4.3. System tests 30](#_c779hh50xj5g)

[4.4. Performance tests 30](#_l2x14y4tfsua)

[4.5. Security checks 30](#_xioqugatczej)

[4.6. Usability tests 31](#_2axrt44oxypf)

[5. Leveraging AI Tools for Research and Development in DataPulse 31](#_y7z94t6qk486)

[**References 32**](#_h1063b4l505t)

# **Abstract**

DataPulse is a decentralized file-sharing platform created to overcome the challenges of traditional single-server downloading methods, which are often ineffective due to limited bandwidth. As digital files grow in size and the demand for efficient sharing solutions increases, DataPulse leverages peer-to-peer (P2P) networking to enable seamless and rapid file distribution. The primary goal of DataPulse is to facilitate the efficient and reliable sharing of large files through a community-driven approach, where each participant acts as both a client and a server. This model enhances download speeds and reduces the load on any single server by allowing simultaneous downloads from multiple peers. In DataPulse, designated peers known as trackers manage groups of connected peers. This tracker-based system optimizes the file-sharing process within the group, ensuring scalable and secure data distribution. By addressing the bottlenecks of traditional server-based methods, DataPulse offers a robust solution for large file sharing, providing a practical platform that guarantees quick, reliable, and efficient file distribution.

# **Introduction**

The rapid growth in digital file sizes and the increasing demand for efficient file sharing have posed significant challenges for traditional single-server download methods. These challenges include bandwidth limitations, reduced download speeds during high traffic, and server upload capacity constraints. The disparity between client download speed and server upload speed often prevents the full potential of client download speeds from being realized, leading to inefficient file distribution. Additionally, when multiple clients attempt to access the same file simultaneously, servers can become overwhelmed, resulting in even slower download times.

To meet these challenges, DataPulse offers a decentralized file-sharing platform that leverages peer-to-peer (P2P) networking and a tracker-based approach. DataPulse designates specific peers as trackers, each managing a group of connected peers. This method simplifies peer discovery and reduces latency, as peers can quickly find others within the group managed by their tracker. By acting as both clients and servers, DataPulse network peers can download files from multiple sources simultaneously, maximizing download speeds and minimizing the load on individual servers. This ensures a scalable, secure, and efficient file distribution system that maintains high performance even under heavy use.

Existing distributed file-sharing solutions, such as BitTorrent and uTorrent, rely on torrent files-small files containing metadata about the files to be distributed, including the creator, creation date, and encoding. While these platforms improve download speeds by allowing users to download files from multiple sources simultaneously, they often feature complex user interfaces that compromise usability. DataPulse improves upon these solutions by streamlining the process and providing a more intuitive user interface, making it easier for users to share and download large files efficiently.

The main users of DataPulse include individuals and businesses-essentially anyone interested in downloading files efficiently and reliably. For individuals, DataPulse offers faster download speeds and a more user-friendly file-sharing experience than existing solutions. Businesses benefit from the scalable and secure distribution of large files, reducing the load on their servers. By addressing the limitations of conventional file-sharing methods and existing P2P solutions, DataPulse provides a robust decentralized file-sharing platform that ensures large files can be shared quickly, reliably, and securely.

# **Related Work**

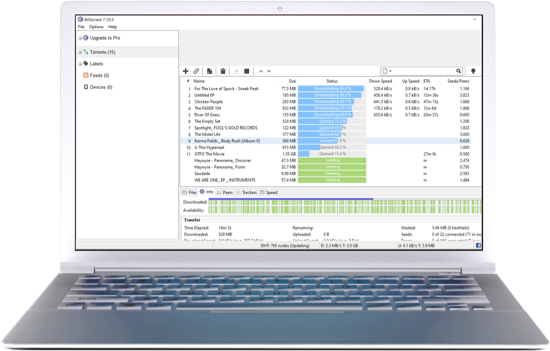
The evolution of distributed file sharing platforms has been significantly influenced by a series of historical developments and technological advancements, in tandem with advances in various technology-driven sectors. Initially, platforms like Usenet introduced decentralized sharing but ran into scalability and bandwidth issues. Next came Napster, which, despite pioneering a central index for broader file sharing, faced legal challenges that highlighted the vulnerabilities of centralized systems. After the fall of Napster, Gnutella emerged, adopting a fully decentralized peer-to-peer (P2P) network that improved legal and operational robustness but highlighted problems with efficiency and scalability. These early systems set the stage for more sophisticated solutions like BitTorrent, which optimized bandwidth usage and improved download speeds with segmented file transfer techniques. Continuous technological innovations have constantly reshaped the file sharing landscape. Modern platforms increasingly harness machine learning to improve system efficiency through predictive algorithms that optimize network routing and load distribution. In addition, blockchain technology has begun to play a central role in further decentralizing file storage and increasing security against tampering and censorship. This journey from Usenet to today's systems shows a dynamic evolution driven by technological advances, aimed at creating more robust and user-friendly systems capable of meeting the complex demands of digital file distribution in the Internet age. Each stage of this evolution was based on lessons learned and limitations of previous systems, and showed a clear trend to integrate advanced technologies to solve historical challenges and improve the efficiency and security of file sharing.

# **Background**

Distributed file sharing enables data to be shared among multiple users without the need for a centralized server. By decentralizing the distribution process, these platforms enhance scalability, resilience, and efficiency. Users can download and upload files concurrently from multiple sources, optimizing bandwidth utilization and accelerating the transfer process. This decentralized approach effectively addresses the limitations of traditional file-sharing methods, such as bandwidth bottlenecks and server overloads.

## BitTorrent and its development

Introduced by Baram Cohen in 2001, BitTorrent revolutionized file sharing by addressing the inefficiencies of previous systems. BitTorrent breaks files into smaller segments, allowing users to download and upload parts at the same time, thus maximizing bandwidth utilization and reducing download times.

BitTorrent uses a decentralized approach with tracking that helps peers find each other. Over time, improvements such as Distributed Hash Table (DHT) for traceless operation, Peer Exchange (PEX) and Local Peer Discovery (LPD) have further improved its efficiency and durability.

## 

### 1.1. Torrent file

Torrent file is a small file that contains metadata about the files to be shared in a peer-to-peer (P2P) network, specifically using the BitTorrent protocol. It includes information such as the names and sizes of the files being distributed, the structure of the data divided into smaller pieces, and the URL of a tracker—a server that helps coordinate the communication between peers in the network. The torrent file does not contain the actual content but rather the information needed for peers to locate and exchange the data efficiently. This mechanism allows users to download and upload file segments from multiple sources simultaneously, optimizing bandwidth usage and speeding up the transfer process. Over time, advancements like Distributed Hash Table (DHT), Peer Exchange (PEX), and Local Peer Discovery (LPD) have enhanced the effectiveness and resilience of torrent-based file sharing.

## Criteria for Evaluating Distributed File Sharing Platforms

When evaluating the effectiveness of a distributed file-sharing platform, several key criteria should be considered:

**Download and Upload Speeds**: The platform should be designed to maximize download speeds by enabling simultaneous downloads from multiple sources. Additionally, it should optimize upload speeds to ensure users contribute effectively to the network.

**Scalability**: The platform must maintain efficiency as the number of users and the size of files increase, avoiding significant delays or crashes even under heavy load.

**Reliability and Resilience**: The system should be capable of handling high traffic volumes and remain operational even if some peers disconnect or if there are network interruptions.

**Security and Privacy**: The platform should implement strong encryption and security protocols to protect user data and ensure that shared files are not tampered with or intercepted by unauthorized parties.

**User Experience**: The platform should offer a user-friendly and intuitive interface, making it accessible to both technical and non-technical users.

## Beyond File Sharing: Other Applications of Distributed Sharing

While distributed file sharing is commonly associated with large file transfers, its principles can be applied to various other domains:

**Content Distribution Networks (CDNs)**: Similar technologies are employed to distribute web content globally, allowing users to access websites and media quickly by downloading from the nearest server or peer.

**Blockchain Technology**: Distributed sharing concepts are fundamental to blockchain systems, where data such as transaction records are distributed across multiple nodes, ensuring transparency, security, and resilience.

**Streaming Services**: Some streaming platforms leverage distributed sharing to manage bandwidth and provide smooth streaming experiences by distributing the load across multiple servers.

**Collaborative Platforms**: Distributed sharing is also used in collaborative platforms, enabling multiple users to work on the same files or projects in real-time, ensuring synchronization and data consistency across all peers.

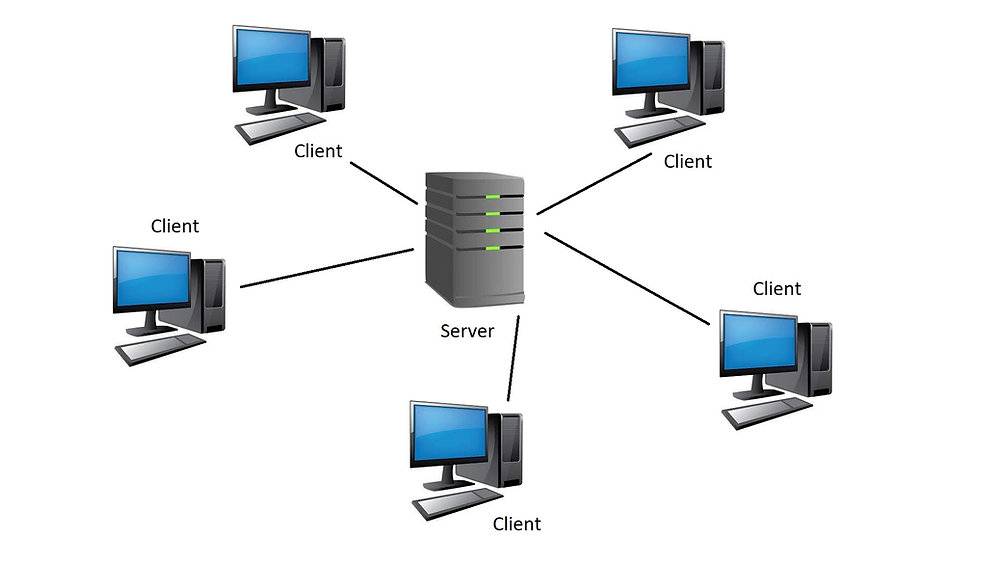
## Common Terms and Protocols

### 4.1. Communication protocols

**TCP (Transmission Control Protocol):** TCP is a core protocol of the Internet Protocol Suite, providing reliable, ordered, and error-checked delivery of data between applications. In BitTorrent, TCP is commonly used for creating connections and transferring data between peers.

**HTTP/HTTPS Trackers:** HTTP and HTTPS trackers are server-based systems used for peer discovery in file sharing. HTTP trackers use the HTTP protocol to manage peer lists, while HTTPS trackers add encryption for enhanced security and privacy. Although HTTP/HTTPS trackers have higher overhead compared to UDP trackers, they are valued for their reliability and compatibility with web technologies.

**Star Trackers:** Star trackers represent an advanced concept in distributed systems, particularly in the context of P2P networks. Unlike traditional trackers that simply coordinate peer communication, star trackers dynamically select the most stable and reliable peers to act as central coordinators or "stars" within the network. This selection is based on criteria such as peer uptime, bandwidth, and historical reliability. By designating these star peers, the system can reduce the likelihood of connection drops, optimize the discovery process, and enhance overall network stability. Star trackers essentially create a more efficient and resilient network topology by ensuring that the most capable nodes are leveraged to maintain the integrity and performance of the distributed system.



### 4.2. Encryption Protocols

**Advanced Encryption Standard (AES):** symmetric encryption algorithm that is widely regarded as one of the most secure and efficient methods for encrypting data. Developed by the U.S. National Institute of Standards and Technology (NIST) in 2001, AES uses a block cipher approach with key sizes of 128, 192, or 256 bits to encrypt and decrypt data in fixed-size blocks of 128 bits. Due to its strong security and speed, AES is commonly used in various applications, including securing sensitive data, protecting communications, and encrypting files in distributed systems.

**SSL/TLS:** These protocols secure communication between the qBittorrent Web UI and the browser, ensuring that the data being transferred is encrypted.

### 4.3. Advanced Protocol Features

**Peer Exchange (PEX):** PEX enables peers to share information about other peers they are connected to, which helps in discovering additional peers within the network.

**Local Peer Discovery (LPD):** LPD allows peers in the same local network to discover each other without the need for a tracker, optimizing local network traffic.

**Rarest First Algorithm:** This algorithm prioritizes downloading the rarest pieces of a file first to ensure that all pieces are available among peers, reducing the risk of pieces becoming unavailable.

**Endgame Mode:** Endgame mode ensures the fast download of the last pieces of a file by requesting them from multiple peers simultaneously.

## Error Handling and Redundancy in Distributed File Sharing

### 5.1.Error Handling Mechanisms

Distributed file-sharing platforms must incorporate robust error-handling mechanisms to maintain data integrity and ensure successful file transfers, even when network issues or peer disconnections occur. These mechanisms can include:

**Checksum Verification**: Each piece of the file being transferred is accompanied by a checksum value, which allows the receiving peer to verify the integrity of the data. If the checksums don't match, the data can be requested again from another peer.

**Automatic Retransmission**: If a piece of a file fails to transfer correctly, the system can automatically attempt to retransmit the data from the same or different peers.

### 5.2. Redundancy Strategies

Redundancy in distributed file sharing ensures that files remain available even if certain peers are offline or data is corrupted. This can be achieved through:

**Replication**: Multiple copies of each file segment are stored across different peers. This way, if one peer becomes unavailable, others can still provide the necessary data, ensuring uninterrupted file access.

**Forward Error Correction (FEC)**: FEC algorithms can be used to reconstruct lost or corrupted data segments without needing retransmissions. This method encodes the data in such a way that if parts of the data are lost, they can be recovered from the remaining data and the error correction code.

### 5.3. Data Availability and Uptime

Maintaining high data availability is crucial for the reliability of distributed file-sharing networks. Strategies to ensure this include:

**Peer Selection Algorithms**: These algorithms prioritize connections to peers that are more likely to stay online longer, thus ensuring that data remains available for extended periods.

**Load Balancing**: Distributing requests evenly across available peers prevents any single peer from being overwhelmed, which enhances the overall stability and availability of the network.

## Network Traffic Optimization in Distributed File Sharing

### 6.1. Congestion Control Mechanisms

In distributed file-sharing networks, managing network congestion is crucial to maintain smooth data flow and prevent bottlenecks. Techniques such as:

**Rate Limiting**: Setting limits on the upload and download speeds for each peer helps prevent overwhelming the network, ensuring a fair distribution of bandwidth among all users.

**Traffic Shaping**: This involves controlling the type and amount of traffic sent across the network to optimize performance and reduce latency. By prioritizing certain types of traffic (e.g., more critical data packets), the network can operate more efficiently.

### 6.2. Bandwidth Management

Effective bandwidth management ensures that the available bandwidth is utilized optimally, balancing the load across the network. Strategies include:

**Dynamic Bandwidth Allocation**: Allocating bandwidth dynamically based on current network conditions and peer requirements helps to adapt to changes in traffic load, ensuring that resources are used efficiently.

**Peer Prioritization**: Prioritizing peers with higher bandwidth or better connectivity can improve overall network performance, as these peers can handle more data and facilitate faster transfers.

### 6.3. Latency Reduction Techniques

Reducing latency is essential for improving the responsiveness of file-sharing networks. Techniques include:

**Locality-Aware Peer Selection**: Selecting peers that are closer can speed up data transfers by reducing the time it takes for data to travel across the network.

**Caching**: Storing frequently requested data at strategic locations within the network can reduce the distance that data needs to travel, thereby lowering latency and speeding up access times.

# **Expected Achievements**

## 1.Results

In the DataPulse project, we aim to develop a decentralized file sharing platform that addresses the limitations of traditional single server download methods.

Developing a strong P2P network system:   
DataPulse will create a decentralized network where peers can share files efficiently without relying on a central server. This system will leverage advanced protocols such as P2P to improve connectivity and TCP for optimal data transfer.

Improved user interface: a central emphasis will be on creating an intuitive and user-friendly interface that simplifies the file sharing process for both technical and non-technical users. The interface will include features that make peer discovery and file management simple, improving the overall user experience.

Scalability and performance optimization: The platform will be designed to handle large numbers of users and file sizes efficiently. This will be achieved by implementing load balancing algorithms and ensuring that the system can scale seamlessly as demand increases.

Security and privacy measures: DataPulse will incorporate strong encryption protocols, such as SSL/TLS and Protocol Encryption (PE), to protect user data and ensure secure file transfers. The platform will also offer features to anonymize user activity and prevent unauthorized access.

## 2. Unique features

Decentralized Tracking System: DataPulse will implement a decentralized tracking system where dedicated peers act as trackers, managing the peer network and enabling faster discovery of peers. This approach will reduce latency and improve network resilience.

Effective data redundancy and error handling: The platform will include features for automatic data redundancy and error correction, ensuring that files remain available even if some peers are offline. This will improve the reliability and resilience of the file sharing network.

Integration with modern web technologies: The platform will be built using modern web technologies such as Angular for the frontend and MongoDB for the backend, which ensures a scalable and maintainable system architecture.

## 3. Success criteria

**Efficient file sharing:** The platform should enable fast and reliable file sharing, with minimal interruptions, over a distributed network.

**User-friendly interface:** The platform should include an intuitive and accessible user interface that simplifies the file sharing process.

**Scalability:** The system must handle an increasing number of users and large file sizes without compromising performance or reliability.

**Security:** Strong encryption and privacy protocols should be in place to protect user data and ensure secure communication within the network.

**Cross-platform functionality:** DataPulse should run smoothly, providing a consistent user experience.

**High Availability and High Redundancy:** The system should maintain high data availability, with effective redundancy mechanisms and error handling to ensure uninterrupted service.

# **Engineering Process**

## 1. Research/Engineering Process

The development of DataPulse, a decentralized file sharing platform, involves a carefully structured research and engineering process. This strategic approach enables a deep understanding of the theoretical foundations of peer-to-peer technology, as well as the practical considerations for its future implementation.

### 1.2. Research - Distributed File Sharing

The research phase commenced with a thorough exploration of the history and evolution of file sharing protocols and technologies. This examination placed particular emphasis on the following areas:

**Historical Development:**An in-depth analysis was conducted on early file sharing systems, such as Usenet and Napster, to understand their underlying mechanisms, the challenges they faced, and their influence on the design of newer models like BitTorrent.

**Current Technologies:**The team extensively researched modern file sharing systems, focusing on the efficiencies introduced by advanced protocols. In addition, the potential role of emerging technologies, such as machine learning and blockchain, in improving the efficiency and security of file distribution systems is examined.

**Legal and Ethical Considerations:**The research also involved an assessment of the legal implications and security concerns that may affect the future deployment of a distributed file sharing system, ensuring that the development of DataPulse aligns with relevant regulations and best practices.

This comprehensive research phase included a thorough review of academic literature, an examination of related projects, and consultations with industry experts. Regular team meetings were instrumental in discussing these findings and formulating the implementation strategy for DataPulse.

## 2. Constraints and Challenges - Distributed File Sharing

The research process identified several key constraints and challenges that must be addressed in the development of DataPulse. These challenges are critical to ensuring the platform's success and require carefully designed strategies to mitigate potential risks.

### 2.1. Scalability and Efficiency

Challenge: Designing a system capable of handling a large and growing number of users and files without compromising performance or stability.

Strategy: To address scalability, DataPulse will implement a modular architecture that allows for horizontal scaling. This involves distributing the workload across multiple servers or nodes, ensuring that no single point becomes a bottleneck. Load balancing algorithms will be employed to distribute network traffic evenly, preventing any one node from becoming overwhelmed.

### 2.2. Security Concerns

Challenge: Developing a robust strategy to mitigate potential vulnerabilities that could enable unauthorized access or data breaches, while maintaining the overall security and integrity of the platform.

Strategy: Security will be prioritized through a multi-layered approach, incorporating both proactive and reactive measures. DataPulse will employ end-to-end encryption using advanced protocols like SSL/TLS for all data transmissions. Additionally, a regular security audit process will be established, including penetration testing and code reviews to identify and patch vulnerabilities promptly. The platform will also integrate a role-based access control system, ensuring that users have only the necessary permissions to perform their tasks, thus minimizing the risk of internal threats.

### 2.3. User Privacy

Challenge: Balancing the need for effective data sharing with the imperative to protect user privacy, which is essential for maintaining trust and compliance with data protection laws.

Strategy: To protect user privacy, DataPulse will implement strict data anonymization techniques, ensuring that personally identifiable information is not stored or transmitted without encryption. Additionally, privacy by design principles will be embedded into the platform's architecture, meaning that privacy considerations will be integrated into every aspect of the system from the outset.

## 3. Central Requirements of the Project

### 3.1. Functional requirements

| 1 | The system enables secure and efficient file uploads, downloads, and sharing in a distributed network. |
| --- | --- |
| 2 | The system implements strong authentication methods to ensure that only authorized users can access the platform. |
| 3 | The system allows peer-to-peer connections for direct communication between users. |
| 4 | The system provides real-time monitoring, allowing users to track file transfer progress with details on upload/download status. |
| 5 | The system implements a file versioning feature to manage and track different versions of shared files. |
| 6 | The system encrypts data transfers to protect user information and ensure the security of file sharing. |
| 7 | The system can detect and handle errors during file transfers, automatically retrying or recovering from failures to maintain data integrity. |

### 3.2. Non-functional requirements

| 1 | The system is optimized to effectively support an increasing number of users and manage large amounts of data without experiencing performance issues. |
| --- | --- |
| 2 | The system provides an easy-to-use graphical interface that ensures accessibility for both technically skilled users and novice users. |
| 3 | The system employs advanced encryption techniques to protect user data during transmission and storage, ensuring robust security. |
| 4 | The system is designed to maintain consistent operation and data integrity, even in the event of node failures or network disruptions. |
| 5 | The system is optimized for high performance, ensuring that file transfers and other operations are completed efficiently, even under heavy load. |
| 6 | The system ensures a steady and reliable data transfer rate, minimizing fluctuations and maintaining a consistent user experience during file sharing. |

## 4. Description of the Requirements Collection

The requirements for DataPulse were meticulously gathered through a combination of methods to ensure a comprehensive understanding of user needs and system constraints. Document analysis involved reviewing technical documentation and existing case studies on distributed systems to identify key requirements and potential limitations. Interviews with potential users and stakeholders provided valuable insights into the desired features and functionalities, ensuring the platform would meet real-world needs. Additionally, observations of user interactions with existing file-sharing platforms helped identify common pain points and opportunities for improvement, allowing the team to tailor the system for enhanced user experience.

## 5. Motivation

Product Development The primary motivation behind the DataPulse project is to create a decentralized file sharing platform that addresses the limitations of traditional systems, such as server overload and privacy concerns. By leveraging the latest advancements in peer-to-peer technology, the aim is to develop a more efficient, secure, and user-friendly solution.

The selected work phases, ranging from the initial research to the development of the planned product, are designed to ensure that DataPulse not only meets the current needs of file sharing but also anticipates and adapts to future technological trends and user requirements. For instance, the focus on user-friendly interfaces reflects the team's commitment to making the platform accessible and intuitive for a wide range of users, ultimately encouraging wider adoption and community engagement.

## 6. Suggested Hardware and Software Tools for the development

* **Programming Languages:** Python for the backend development, leveraging its powerful libraries and frameworks that enable efficient network operations.
* **Front-end Development:** Angular, with its efficient data binding and modular architecture, to enhance the user interface design.
* **Database Management:** MongoDB, known for its scalability and flexibility in handling large data sets typical of file sharing applications.

## 7. Algorithms and Data Structures

The DataPulse project utilizes a modular architecture that combines several core algorithms and data structures to enable its distributed file sharing functionality.

### 7.1. Peer-to-Peer Network Model

At the core of the system is the peer-to-peer network model, which is responsible for managing the connections between the various nodes (users) in the network.

### 7.2. Segmented file transfer

The file transfer process in DataPulse is based on a segmented approach, inspired by the BitTorrent protocol. Each file is divided into smaller pieces, which are then distributed and shared among the connected peers. This enables simultaneous uploads and downloads, improving the overall efficiency and resilience of the network.

### 7.3. Encryption and security

To ensure the security and privacy of data transfers, DataPulse incorporates advanced encryption algorithms and custom encryption methods for the file data itself, providing an additional layer of protection.

### 7.4. Data Structures

The system's data architecture includes the following key components:

* Peer Registry: Maintains a decentralized database of connected peers, storing relevant information such as IP addresses, file lists, and connection status.
* File Index: Manages the metadata for all the files available in the network, including file segments, their availability, and integrity verification checksums.
* Network Graph: Represents the dynamic topology of the peer-to-peer network, allowing the system to adapt to changes as peers join or leave the network.

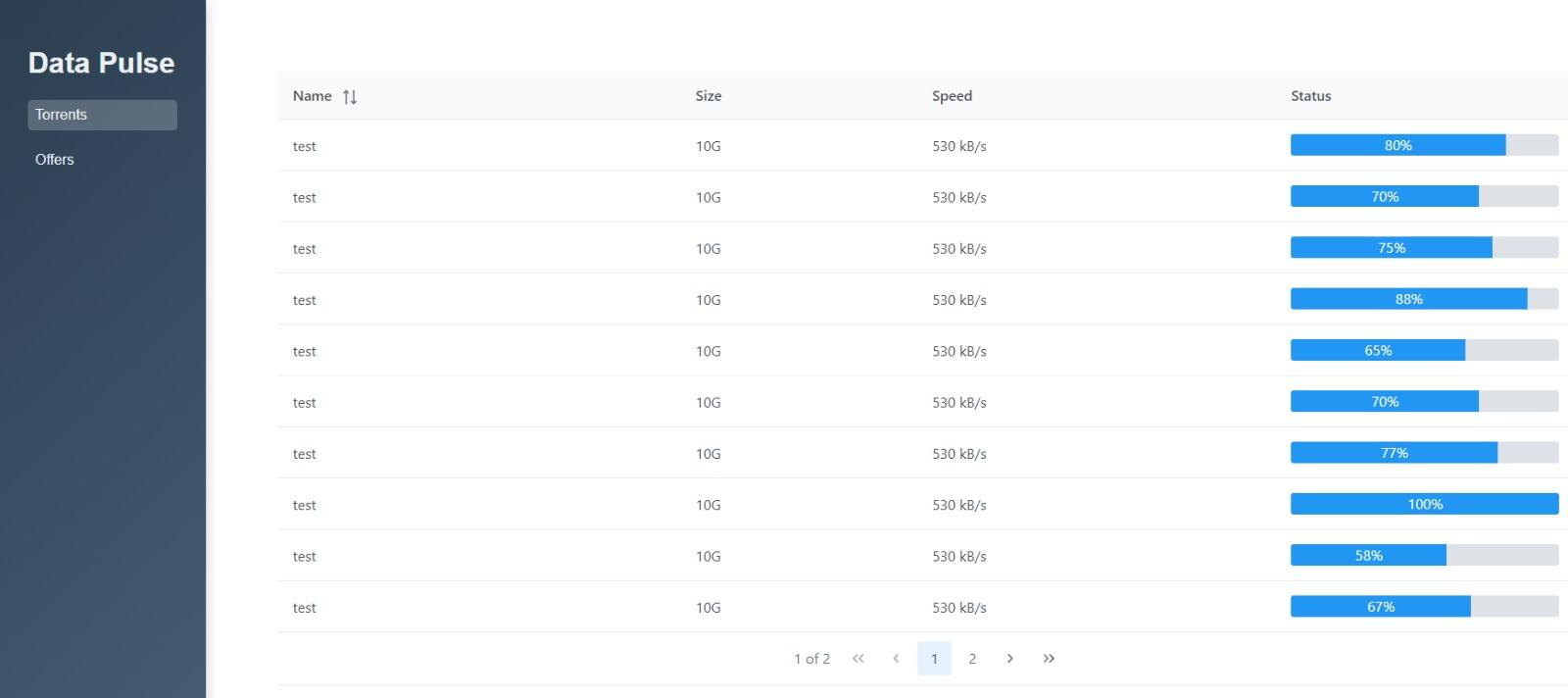
These data structures work in tandem to enable the efficient discovery, transfer, and tracking of files within the DataPulse network.

## 8. Future User Interface Design

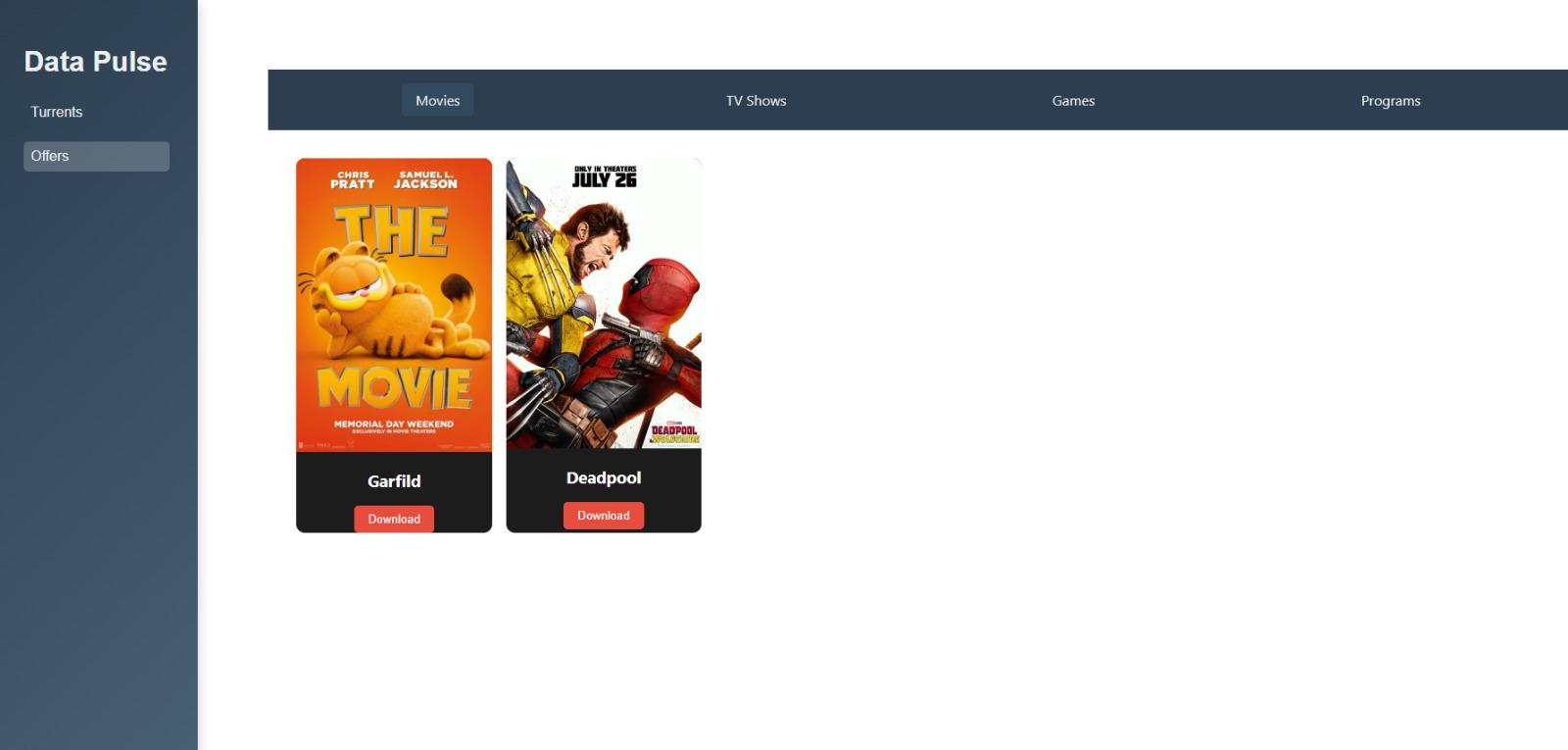
The DataPulse user interface is designed to be intuitive and accessible, providing a seamless experience for both novice and experienced users. The interface incorporates the following key features:

* File Management Dashboard: Users will have a centralized view of their shared files, with options to upload, download, pause, resume, and cancel transfers.
* Drag-and-Drop Uploads: The interface will support easy file uploads by allowing users to simply drag and drop files into designated areas of the application.
* Real-Time Download Status: Users will be able to monitor the progress of their file downloads and uploads, with detailed information on transfer speeds, remaining time, and completed percentages.

By combining the efficiency of decentralized file sharing with an intuitive and user-friendly interface, DataPulse aims to create a seamless and accessible experience for its users, encouraging broader adoption and community engagement.

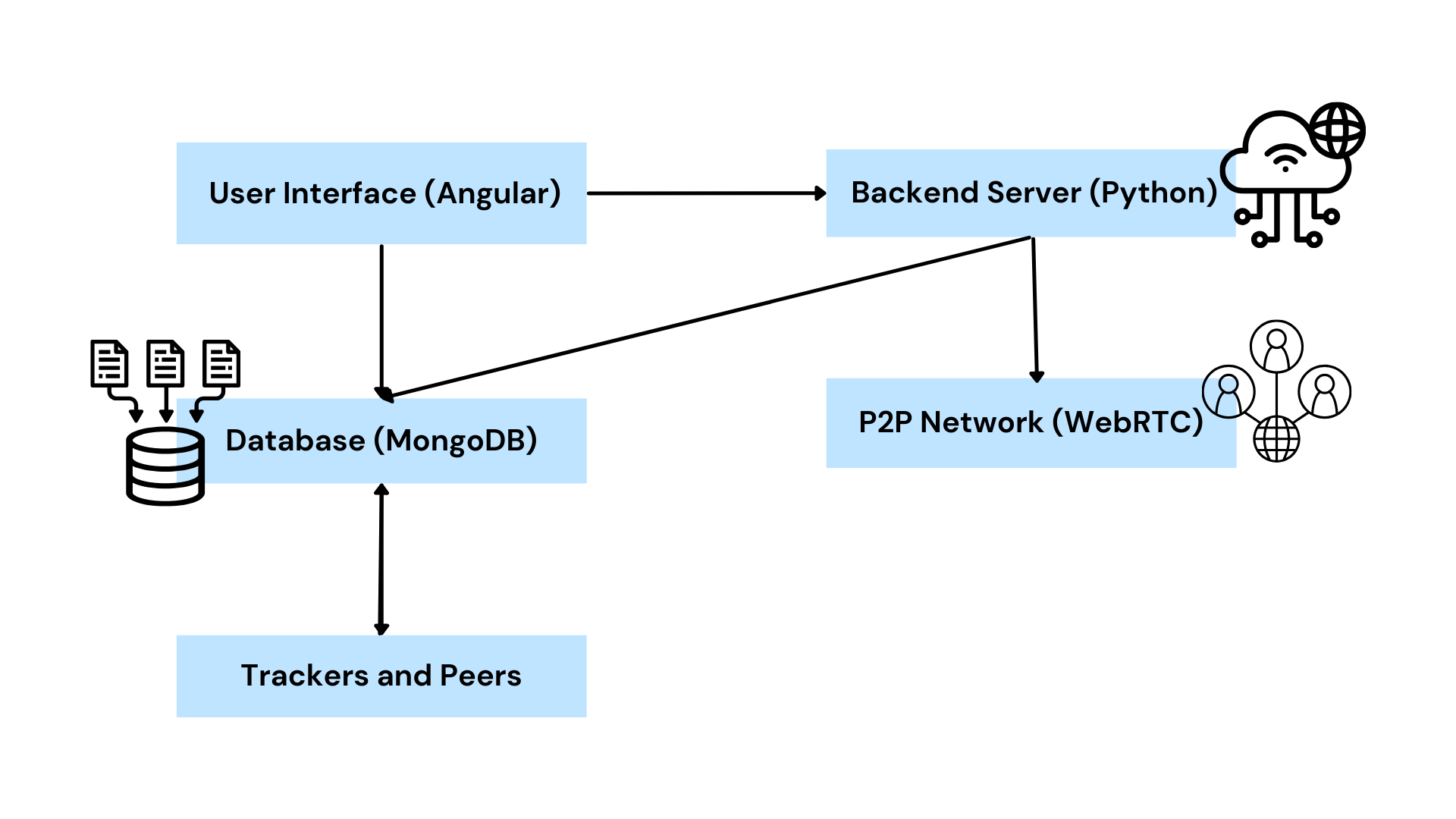


File Management Dashboard



## File Selection Interface

## 9. System Architecture

The overall system architecture of DataPulse follows a modular design, allowing for scalability and easier integration of future enhancements. The key components and their relationships are illustrated in the system architecture diagram:  


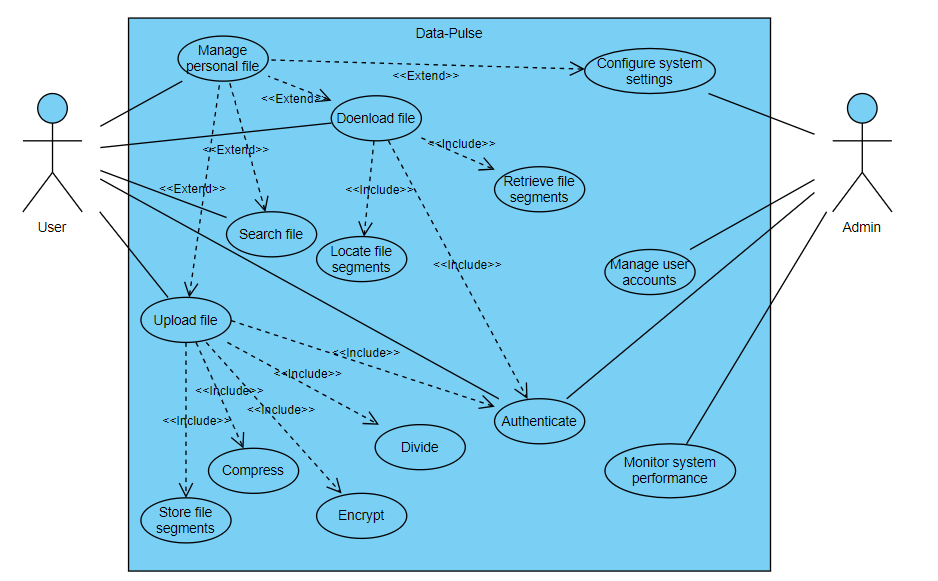
The Peer Registry and File Index work together to manage the discovery and tracking of peers and shared files within the decentralized network. The Segmented Transfer module is responsible for breaking down files into smaller pieces and coordinating their distribution among the connected peers.

The encryption module ensures the security of data transfers by applying advanced encryption techniques. The user interface provides the visual and interactive layer, allowing users to seamlessly upload, download, and manage files, as well as monitor the overall system status.

This modular architecture allows DataPulse to be flexible, scalable, and capable of incorporating future advances in peer-to-peer technology and user experience improvements.

## 10. Diagrams

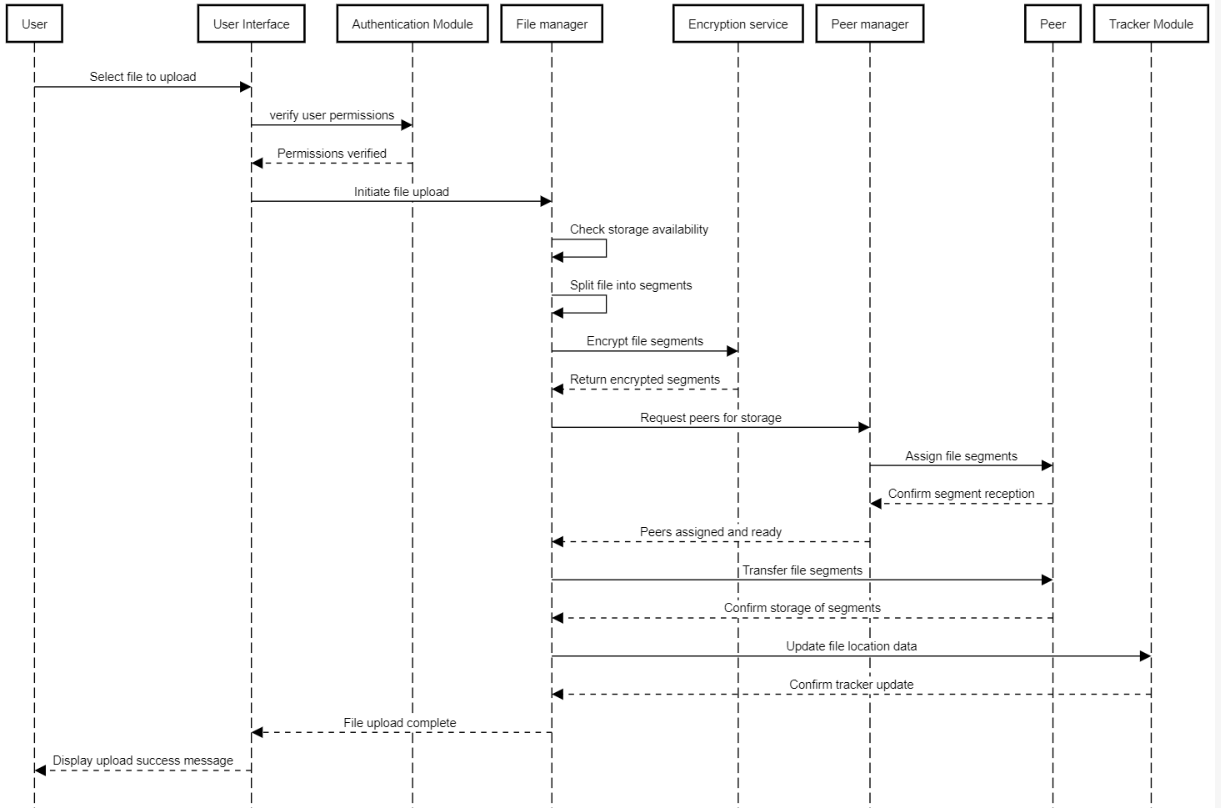
### 10.1. Use Case



The flowchart describes the interactions between different users (User and Admin) and the system functionalities within the DataPulse platform. Users can authenticate, manage personal files, search for files, upload, download. Admins, on the other hand, have additional capabilities such as monitoring system performance and managing user accounts. The central process for handling requests includes various steps such as authentication, compression, encryption, file division, and the storage or retrieval of file segments. The system is designed to efficiently manage and secure file sharing operations, ensuring both user accessibility and system integrity.

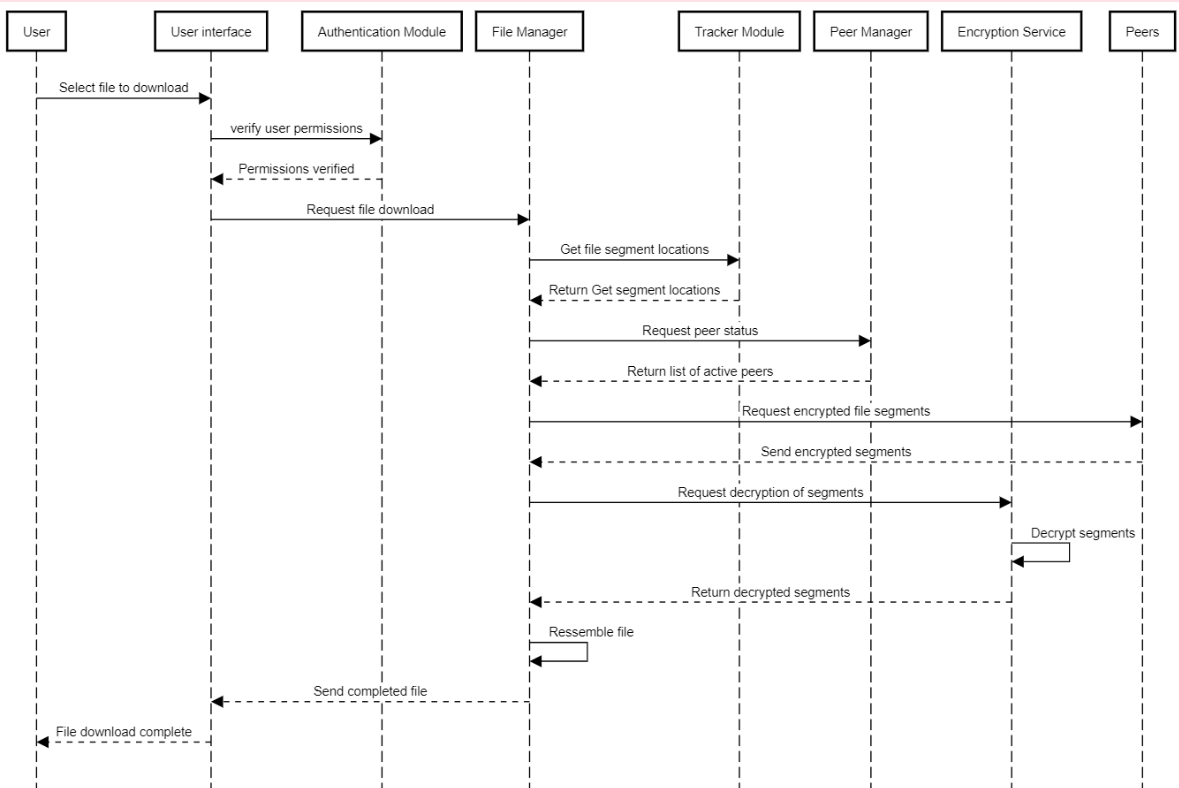
### 10.2. Sequence Diagram

#### 10.2.1. Sequence Diagram of file upload



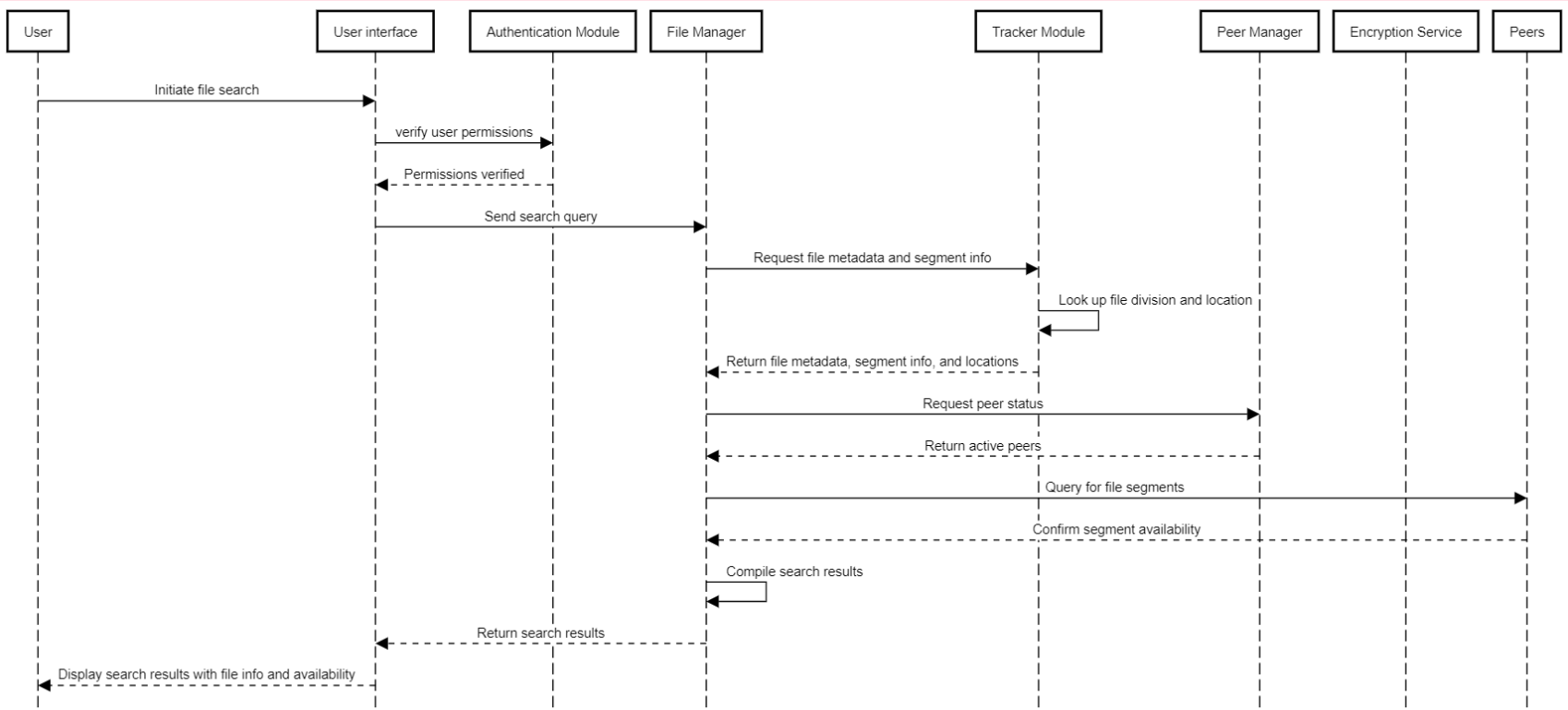
The sequence diagram for the DataPulse system illustrates a structured flow of the file upload process, beginning with the user's interaction with the system and culminating in the successful storage of the file across multiple peers. Initially, the user selects a file for upload, and the User Interface (UI) verifies the user's permissions through the Authentication Module. Once verified, the File Manager takes over, splitting the file into segments and sending them to the Encryption Service for secure encryption. These encrypted segments are then distributed to available peers by the Peer Manager, which manages peer assignments and confirms the storage. Finally, the File Manager updates the Tracker Module with the location data of the file, ensuring that the system knows where each segment is stored. The process concludes with the UI notifying the user of the successful upload, completing the sequence.

#### 10.2.2. Sequence Diagram of file download



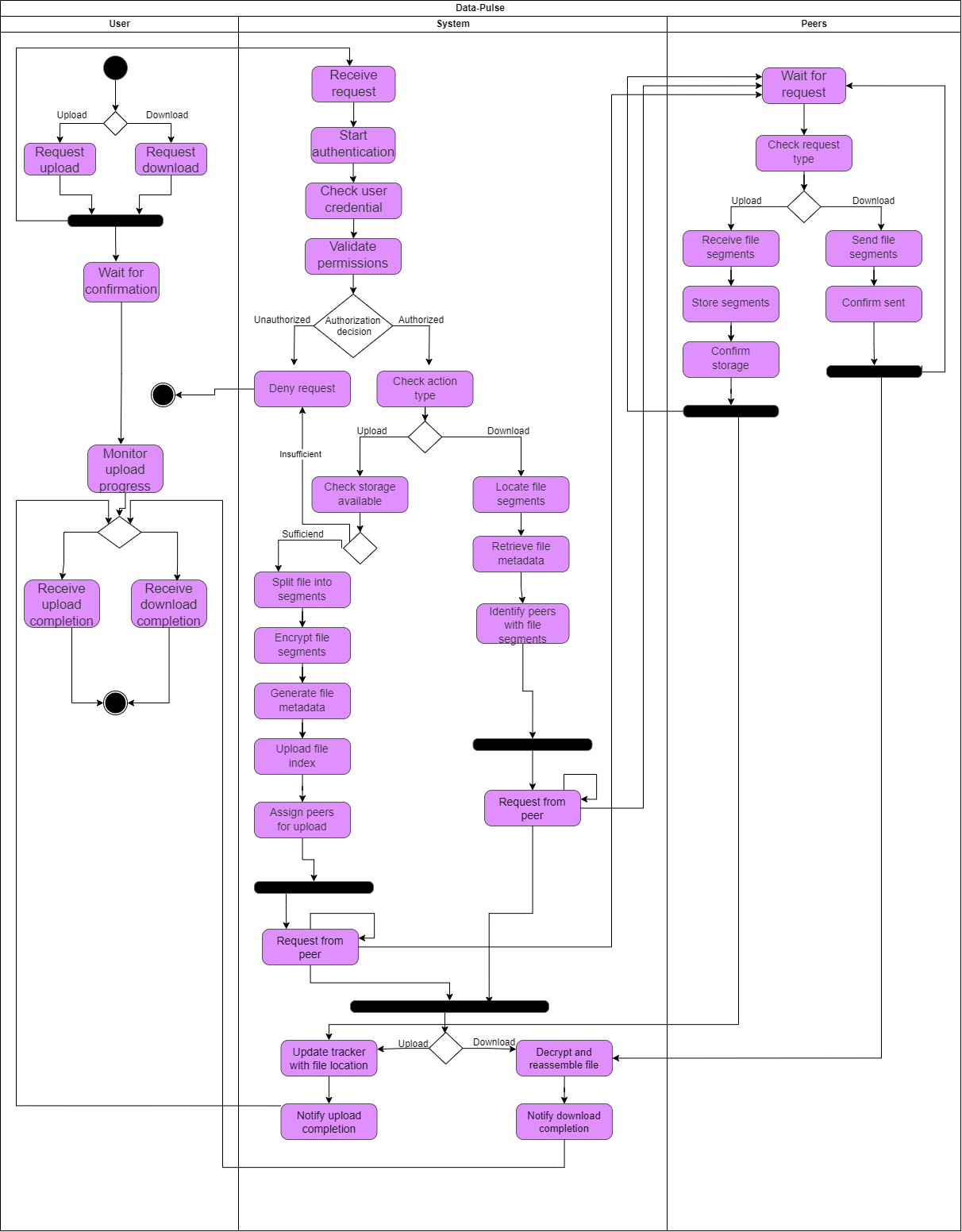
The sequence diagram illustrates the process of a user downloading a file through a series of interactions between different components. The user initiates the process by selecting the file to download via the user interface, which then verifies user permissions with the Authentication Module. Once permissions are verified, the File Manager is tasked with managing the download. It first queries the Tracker Module to obtain the locations of file segments, and then checks the Peer Manager for the status of available peers. The File Manager requests the encrypted file segments from the active peers, which are then sent back to the File Manager. These segments are passed to the Encryption Service for decryption. After successfully decrypting the segments, the File Manager reassembles the complete file and sends it to the user interface, which then confirms to the user that the file download is complete. The flow emphasizes the coordinated interaction between multiple modules to ensure secure and efficient file downloading.

#### 10.2.3. Sequence Diagram of file search



The sequence diagram depicts the flow of a file search within a system, where the user initiates the search through the user interface. The process begins with the user interface verifying user permissions with the Authentication Module. Once permissions are confirmed, the user interface sends the search query to the File Manager. The File Manager then requests file metadata, segment information, and their locations from the Tracker Module, which retrieves this data from its records. After receiving the information, the File Manager queries the Peer Manager to check the status of available peers. The File Manager then sends queries to the identified peers to confirm the availability of the requested file segments. Once all the data is gathered, the File Manager compiles the search results, including file metadata and availability information, and sends it back to the user interface. Finally, the user interface displays the search results to the user, providing detailed file information and its availability for download.

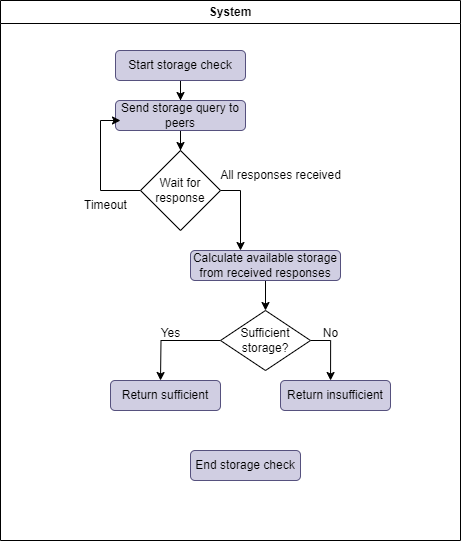
### 10.3. Activity Diagram



The DataPulse system begins when a user initiates an upload or download. After authentication, the system processes the request: for uploads, it checks storage, splits and encrypts the file, generates metadata, and assigns peers; for downloads, it locates file segments and identifies peers. The system then enters a parallel processing phase, distributing or requesting segments from multiple peers simultaneously. Once complete, the system finalizes the process by updating the tracker (for uploads) or reassembling the file (for downloads), and notifies the user. Throughout, peers wait for requests, handle file segments, and return to a waiting state. This decentralized approach emphasizes efficient file sharing, security, and scalability.

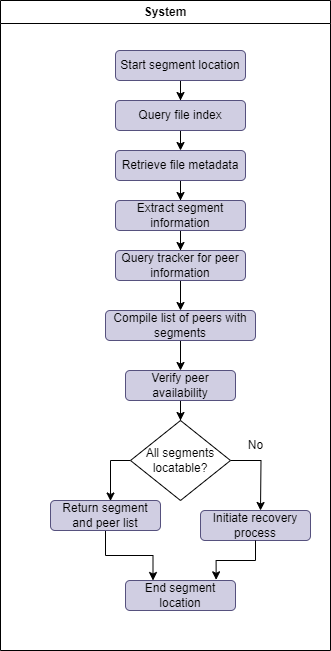
This flow demonstrates the decentralized nature of DataPulse, where file storage and retrieval are distributed across multiple peers. The central system manages the overall process of file handling, tracking, and security, while the user interacts primarily with the system interface.

#### 10.3.1. Expanded Activity Diagram for “check storage available”



The process begins with the system initiating a storage check by sending a query to all connected peers to assess available storage space. The system then waits for responses, and if a timeout occurs, it proceeds to calculate the available storage based on the responses it has received. If all responses are received within the expected timeframe, the system immediately calculates the available storage. Once the available storage is calculated, the system evaluates whether there is sufficient space to proceed with the file upload. If the storage is sufficient, the system returns a confirmation that there is enough space. If not, it returns an insufficient storage message. The process concludes after this evaluation, ensuring that the system only proceeds with file uploads when adequate storage is confirmed, thereby maintaining system efficiency and reliability.

#### 10.3.2. Expanded Activity Diagram for “locate file segments”



The process begins with the system initiating the segment location phase, where it queries the file index to retrieve metadata about the requested file. This metadata includes details on how the file is divided into segments. The system then extracts specific information about these segments, such as their sizes and identifiers, and queries the tracker to identify which peers hold the required segments. A list of peers with the necessary file segments is compiled, and the system verifies the availability of these peers. The system then checks whether all segments can be located; if they are, it returns the segment and peer list. If not, it initiates a recovery process to find any missing segments. The process concludes after either successfully locating all segments.

# **Verification and Evaluation**

Testing is a crucial step in the development lifecycle of DataPulse, a distributed file sharing platform. The goal is to ensure that the platform works as expected, maintains data integrity, is secure and provides a smooth user experience. This test plan describes the strategies, methodologies, and environment setup for testing DataPulse.

## Test strategy

DataPulse's testing strategy will include a combination of manual and automated testing to cover all aspects of the platform, including functionality, performance, security and usability. The main types of tests to be performed include:

* Unit Testing: Each module (eg, peer registration, file indexing, segmented file transfer) will be independently tested using frameworks like unittest to make sure they work correctly in isolation.
* Integration Testing: Once individual components have been tested, they will be integrated and tested as a whole to ensure they interact correctly.
* System testing: The entire system will be tested to make sure it meets the requirements and functions properly under different conditions.
* Performance Testing: The platform will be tested for speed, scalability and responsiveness under varying loads using Apache JMeter to ensure it can handle a large number of users and files.
* Security Testing: This will include testing for vulnerabilities, data encryption and user authentication mechanisms to ensure the platform is secure.
* Usability testing: The user interface and user experience will be tested to make sure the platform is intuitive and accessible.
* Regression Testing: After each update or bug fix, regression testing will be performed to ensure that new changes do not adversely affect existing functionality.

## Constraints and assumptions

* Network Variation: Given that DataPulse is a distributed platform, testing will be performed in environments with varying network conditions (eg, high latency, low bandwidth) to simulate real-world scenarios.
* Scalability limitations: The test will assume that the system can be expanded within a certain range of users and data volume. Extreme scalability testing may be limited by available resources.
* Decentralized Environment: The decentralized nature of the platform assumes that peers may join and leave the network frequently, affecting the testing process. The tests will address the dynamic nature of peer participation.

## Work environment testing

The DataPulse test environment will closely mirror the production environment to ensure accurate test results. The following components will be part of the test environment:

### 3.1. Hardware

- A number of machines configured to impersonate different colleagues in the network.

- Monitoring of simulation servers and central databases for certain test scenarios.

### 3.2. Software

- Testing tools: automated test frameworks for user interface testing, JMeter for performance testing and custom scripts for integration and unit testing.

- Database Management Systems: MongoDB instances will be configured to simulate database interactions.

### 3.3. Security

- Crypto Libraries: Testing different crypto protocols to ensure data security across the platform.

### 3.4. User Review

- A small group of beta testers will be invited to use the platform in a controlled environment to gather feedback on usability and identify issues that were not caught in the initial testing phases.

## Test cases and execution

The test plan will include a comprehensive set of test cases covering the different types of tests (unit, integration, system, performance, security, usability). Sample test cases for each category are provided below:

### 4.1. Unit tests

- Make sure the `PeerRegistry` module correctly handles peer registration and deregistration.

- Make sure the 'FileIndexer' module accurately tracks file metadata and availability.

- Check `EncryptionModule` for correct implementation of SSL/TLS and custom encryption algorithms.

### 4.2. Integration tests

- Validate interaction between `PeerRegistry` and `NetworkManager` during peer discovery and connection establishment.

- Check the integration of `UserInterface` with the underlying platform components.

### 4.3. System tests

- Verify end-to-end file sharing workflow, including upload, download, pause and resume functionality.

- Verify system behavior under changing network conditions, such as peer abandonment and bandwidth fluctuations.

- Check the system's ability to handle large file sizes and a high number of simultaneous users.

### 4.4. Performance tests

- Measure the throughput and response times of the platform under various user and file loading scenarios using Apache JMeter.

- Evaluation of system scalability by simulating a

### 4.5. Security checks

- Try exploiting potential vulnerabilities in EncryptionModule and NetworkManager using a security testing tool.

- Verify the effectiveness of user authentication mechanisms and access controls.

- Make sure the platform is resistant to common attacks, such as man-in-the-middle (MITM).

### 4.6. Usability tests

- Observe and collect feedback from beta users on the intuitiveness and ease of use of the user interface.

- Identify areas for improvement in the overall user experience of the platform.

- Verify the accessibility of the platform to users with diverse backgrounds and technical expertise.

DataPulse's testing program is designed to be thorough, covering every aspect of the platform from its underlying infrastructure to the user interface. By testing in different conditions and environments, we aim to identify and resolve issues before they reach the end user, ensuring a robust, secure and user-friendly platform.

## Leveraging AI Tools for Research and Development in DataPulse

During the research phase of the DataPulse project, the AI-driven features of Copilot and Google Scholar were instrumental in guiding our initial design decisions. Copilot, by creating and reviewing code snippets, has allowed us to explore practical implementation strategies for distributed file sharing systems, such as how different network protocols can be efficiently combined. For example, while investigating the implementation of peer-to-peer architectures, Copilot suggested specific configurations for data segmentation and transmission protocols, helping us conceptualize how these might be structured in the final system. On the other hand, Google Scholar's AI-driven search capabilities allowed us to conduct comprehensive literature reviews, identifying key academic papers and patents that advanced our understanding of distributed architectures. This research highlighted the benefits of using a trace-based systems approach, which we identified as critical to ensuring scalability and efficiency at DataPulse. These insights, derived from the tools, were fundamental in designing the theoretical framework of the platform.

# **References**

1. Bharambe, A. R., Agrawal, C., & Seshan, S. (2006). Analyzing and Improving BitTorrent Performance. <https://www.researchgate.net/profile/Ashwin-Bharambe/publication/245396846_Analyzing_and_Improving_BitTorrent_Performance/links/54ef331d0cf2495330e1b8ac/Analyzing-and-Improving-BitTorrent-Performance.pdf>
2. Qiu, D., & Srikant, R. (2004). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. Proceedings IEEE INFOCOM 2004, 1, 367-377. <https://doi.org/10.1109/INFCOM.2004.1354521>
3. Pouwelse, J. A., Garbacki, P., Epema, D. H. J., & Sips, H. J. (2005). An introduction to the BitTorrent peer-to-peer file-sharing system. Proceedings of the IEEE International Symposium on Network Computing and Applications, 4(1), 2-9. Retrieved from<https://www.researchgate.net/publication/228960534_An_introduction_to_the_BitTorrent_Peer-to-Peer_File-Sharing_System>
4. Qiu, D., & Srikant, R. (2010). Modeling and performance analysis of BitTorrent-like peer-to-peer networks. Peer-to-Peer Networking and Applications, 1(2), 85-91. <https://doi.org/10.1007/s12083-010-0070-6>
5. Knežević, P. (2005). Design and analysis of a peer-to-peer distributed system for dynamic networks (Doctoral dissertation). Technical University of Darmstadt. <https://tuprints.ulb.tu-darmstadt.de/854/1/knezevic_thesis.pdf>
6. Roussopoulos, M., Baker, M., & Rosenthal, D. (2003). 2 P2P or Not 2 P2P? IEEE Computer Society 22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings. <https://doi.org/10.1109/RELDIS.2003.1284159>
7. Gopal, S., Mukherjee, S., Srivastava, A., & Biswas, K. (2020). Intelligent Data Sharing System for Smart Cities Using Blockchain and IPFS. 2020 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER), 113-118. <https://doi.org/10.1109/DISCOVER50404.2020.9045515>
8. Legout, A., & Nardelli, L. (2002). Rarest First and Choke Algorithms Are Enough. INRIA Research Report RR-4725. Retrieved from<https://inria.hal.science/inria-00000156>
9. Neumann, F., & Witt, C. (2013). On the runtime of randomized local search and simple evolutionary algorithms for dynamic makespan scheduling. Computers & Operations Research, 40(4), 1010-1020. <https://doi.org/10.1016/j.cor.2012.09.021>
10. Li, Z., Zhou, S., Guan, X., & Ma, D. (2010). A BitTorrent-like P2P file sharing network performance evaluation: A case study. 2010 International Conference on E-Business and E-Government, 543-546. https://doi.org/10.1109/ICEE.2010.5434378