Software Requirements Specification (SRS)

Peer-to-Peer (P2P) File Sharing Platform

Done By :- Abhijith S.L - B22CS02

Aman Ajeeth Mohammad - B22CS12

Shifad Shaji - B22CS63

Table of Contents

1. Introduction

The **Peer-to-Peer (P2P) File Sharing Platform** is aiming to be a decentralized platform designed to enable users to share files directly with one another. Unlike traditional file-sharing systems that rely on centralized servers, this platform utilizes **P2P networking protocols** to establish direct connections between users, enhancing file transfer speed, security, and efficiency. This document outlines the functional and nonfunctional requirements to develop a web-based **Peer-to-Peer (P2P) File Sharing Platform.** The primary goal of the platform is to develop a platform to share files using **P2P networking protocols** without the need of a centralized server.

1.1 Purpose

This document specifies the software requirements for a Peer-to-Peer (P2P) File Sharing Platform. The platform allows users to share files directly with one another using P2P networking protocols, ensuring secure and decentralized file transfer.

1.2 Scope

The P2P file sharing platform will facilitate secure file transfers without relying on a central server. It will feature file encryption, user authentication, and real-time file sharing updates. This platform is designed for scalability, ensuring that users can share large files efficiently.

1.3 Definitions, Acronyms, and Abbreviations

- **P2P**: Peer-to-Peer
- **WebRTC**: Web Real-Time Communication
- **SSL/TLS**: Secure Sockets Layer/Transport Layer Security
- **Backend**: Server-side software
- **Frontend**: Client-side software
- **Database**: A structured collection of data
- **JSON**: JavaScript Object Notation
- **API**: Application Programming Interface

1.4 References

- WebRTC Documentation
- Flask Documentation
- Django Documentation

- React Documentation
- Vue.js Documentation

2. Overall Description

2.1 Product Perspective

This product is a stand-alone decentralized platform designed to allow users to transfer files securely and efficiently without the need for a centralized server. The platform will rely on a peer-to-peer (P2P) architecture for file sharing, utilizing WebRTC for real-time communication. The system environment is designed to support a variety of devices, operating systems, and network configurations to ensure the platform is accessible to a broad range of users.

2.2 Product Features

- **File Sharing**: Allow users to upload and download files to/from peers directly.
- **File Encryption**: Encrypt files before sharing to ensure data privacy.
- **User Authentication**: Ensure that users are authenticated using JWT for secure access.
- **Real-Time Updates**: Provide users with real-time file upload and download status (e.g., percentage progress).

2.3 User Classes and Characteristics

- **End User**: A person who wishes to share files securely with other users.
    - Requirements: Ability to upload, download, and manage files.

2.4 Operating Environment

The platform will be designed to work seamlessly across various devices and operating systems, ensuring broad compatibility and accessibility for users worldwide.

Client-Side:

- **Devices**: Desktop PCs, laptops, tablets, smartphones.
- **Operating System**:
    - Windows 10 or newer
    - macOS 10.13 (High Sierra) or newer

    - Linux (Ubuntu 18.04 or newer)
    - Android 8.0 or newer
    - iOS 12 or newer
- **Hardware Requirements**:
    - CPU: Any modern processor (Intel i3 or higher / AMD Ryzen)
    - RAM: 512MB minimum (4GB or more recommended for heavy file transfers)
    - Storage: Sufficient disk space for file uploads/downloads
    - Network Interface: A stable internet connection (wired or wireless)

Server-Side:

For hosting the backend services, the platform will leverage cloud providers like **AWS**, **Azure**, or other free cloud hosting services. These services offer the flexibility to scale server resources as needed, providing high availability and reliability.

Network Environment:

- **WebRTC for P2P Communication**: The platform utilizes WebRTC for direct peer-to-peer communication, enabling users to share files without the need for a central server. This reduces latency and improves the efficiency of file transfers.
- **Secure Communication**: All interactions between users and the server (including authentication and real-time status updates) will be conducted over HTTPS and WebSockets for secure and real-time communication.
- **Bandwidth**: The platform will require stable, high-speed internet connections to facilitate seamless file transfers, especially for large files.

Software Requirements:

- **Frontend**: React.js or Vue.js for UI development, WebRTC for real-time communication.
- **Backend**: Flask or Django (Python), WebRTC for peer-to-peer communication, PostgreSQL or MongoDB for storing user credentials and file metadata.
- **Encryption**: AES-256 encryption for securing file transfers.
- **Authentication**: JWT (JSON Web Tokens) for secure user authentication.

2.5 Constraints

- **Decentralized Nature**: The platform's P2P architecture reduces reliance on central servers, which can sometimes introduce challenges in managing peer connectivity and file availability.

- **File Size**: The platform must handle large file transfers and ensure efficient data transmission, which can be impacted by user bandwidth and network conditions.

2.6 Assumptions and Dependencies

- The platform relies on WebRTC for P2P communication, which requires browser compatibility.
- Secure file transfer is reliant on robust encryption algorithms.
- The system environment assumes users have access to modern browsers and internet connections to support WebRTC.

3. System Requirements

The system requirements for the **Peer-to-Peer (P2P) File Sharing Platform** are divided into **Functional Requirements (FRs)** and **Non-Functional Requirements (NFRs)** to define the system's capabilities, constraints, and performance expectations.

3.1 Functional Requirements (FRs)

3.1.1 User Authentication & Authorization

- The platform must allow users to **register** and **log in** using an email and password.
- The platform must support **JWT-based authentication** for secure session management.

3.1.2 File Sharing

- The platform must enable users to **upload files** for sharing.
- The platform must enable users to **download files** from peers.
- The platform must support **drag-and-drop functionality** for easy file uploads.
- The platform must allow **multiple simultaneous file uploads/downloads**.
- The platform must support **resumable downloads** in case of network disruptions.
- The platform must allow users to **set permissions** on shared files (e.g., read-only, full access).

3.1.3 Peer-to-Peer Communication

- The platform must use **WebRTC for direct P2P file transfers**.

- The platform must automatically handle **NAT traversal** using STUN/TURN servers if required.
- The platform must **prioritize local network connections** when possible for faster file transfers.

3.1.4 Security & Encryption

- The platform must implement **AES-256 encryption** for file security before transmission.
- The platform must use **HTTPS and WSS (Secure WebSockets)** for communication.
- The platform must ensure **end-to-end encryption** for sensitive data exchanges.
- The platform must store user authentication data securely using **hashed passwords (bcrypt/argon2)**.

3.1.5 Real-Time File Transfer Status

- The platform must display **real-time upload and download progress** to users.
- The platform must notify users upon **successful file transfers** or **failed attempts**.
- The platform must provide a **history/log of transferred files**.

3.1.6 User Interface & Experience

- The platform must have an **intuitive and responsive UI** built with React.js or Vue.js.
- The platform must provide a **file search feature** for users to find shared files quickly.
- The platform must allow users to **preview certain file types** (e.g., images, PDFs) before downloading.

3.1.7 System Administration & Monitoring

- Admin users must be able to **view logs of file transfers**.
- Admin users must have the ability to **ban or block suspicious users**.

3.2 Non-Functional Requirements (NFRs)

3.2.1 Performance & Scalability

- The platform should handle at least **100 concurrent file transfers per server instance**.
- The platform should support **large file transfers (>5GB)** with optimized memory usage.
- The platform should minimize **latency for real-time P2P connections (<200ms delay)**.

3.2.2 Security & Compliance

- The platform must follow **GDPR** compliance for data privacy.
- The platform must log **user actions and file transfer history** securely.

3.2.3 Availability & Reliability

- The platform should have **99.9% uptime**, with fallback mechanisms for server failures.
- The platform should use **distributed servers** to prevent single points of failure.
- The platform should retry **failed file transfers automatically** within 3 attempts.

3.2.4 Usability & Accessibility

- The platform should be **responsive** and function smoothly on **desktop.**
- The platform should support **multi-language support** (at least English, Spanish, and French).

3.2.5 Maintainability & Extensibility

- The platform should use a **modular codebase** for easy maintenance and updates.
- The platform should allow **plug-and-play support** for adding new encryption methods or authentication providers.

4. External Interface Requirements

4.1 User Interfaces

The user interface should be simple and intuitive. Key pages will include:

- **File Upload/Download Page**: Allows users to select and transfer files.
- **Dashboard**: Displays user activity, file progress, and file sharing history.
- **Login/Registration Page**: For user authentication and account creation.

4.2 Hardware Interfaces

The platform does not require special hardware beyond standard computing devices (PCs, laptops, or mobile devices with internet connectivity).

4.3 Software Interfaces

- **Frontend**: React for the UI, connecting to the backend via RESTful APIs.
- **Backend**: Django with WebRTC for real-time **P2P** Connection..
- **Database**: PostgreSQL or MongoDB for user management and metadata storage.
- **Encryption Libraries**: OpenSSL or similar libraries for file encryption.

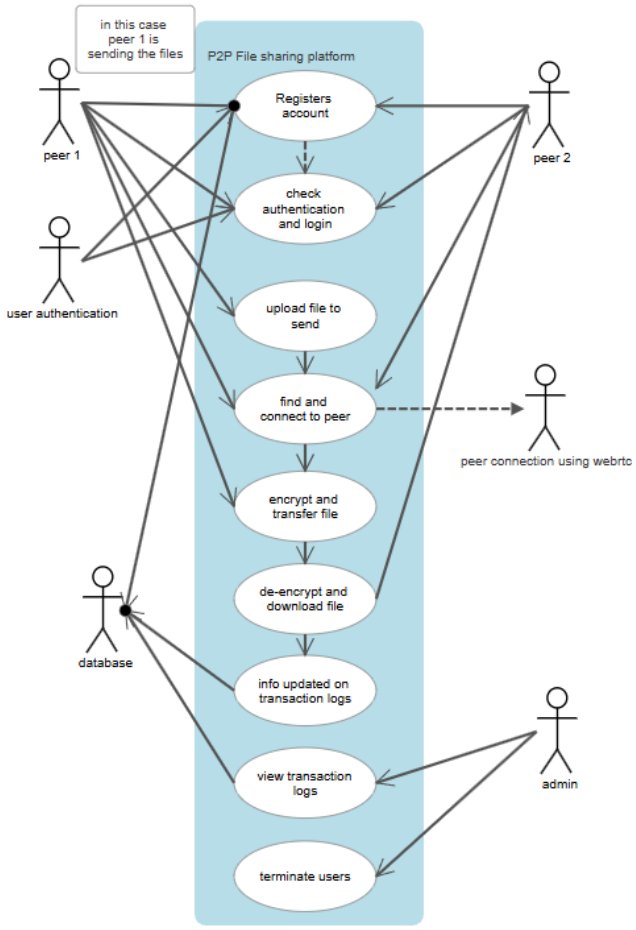
4.4 Communication Interfaces

- **WebRTC** for peer-to-peer communication.
- **REST API** for user interaction with the backend (e.g., file metadata retrieval, authentication).
- **WebSockets** for real-time updates and notifications.

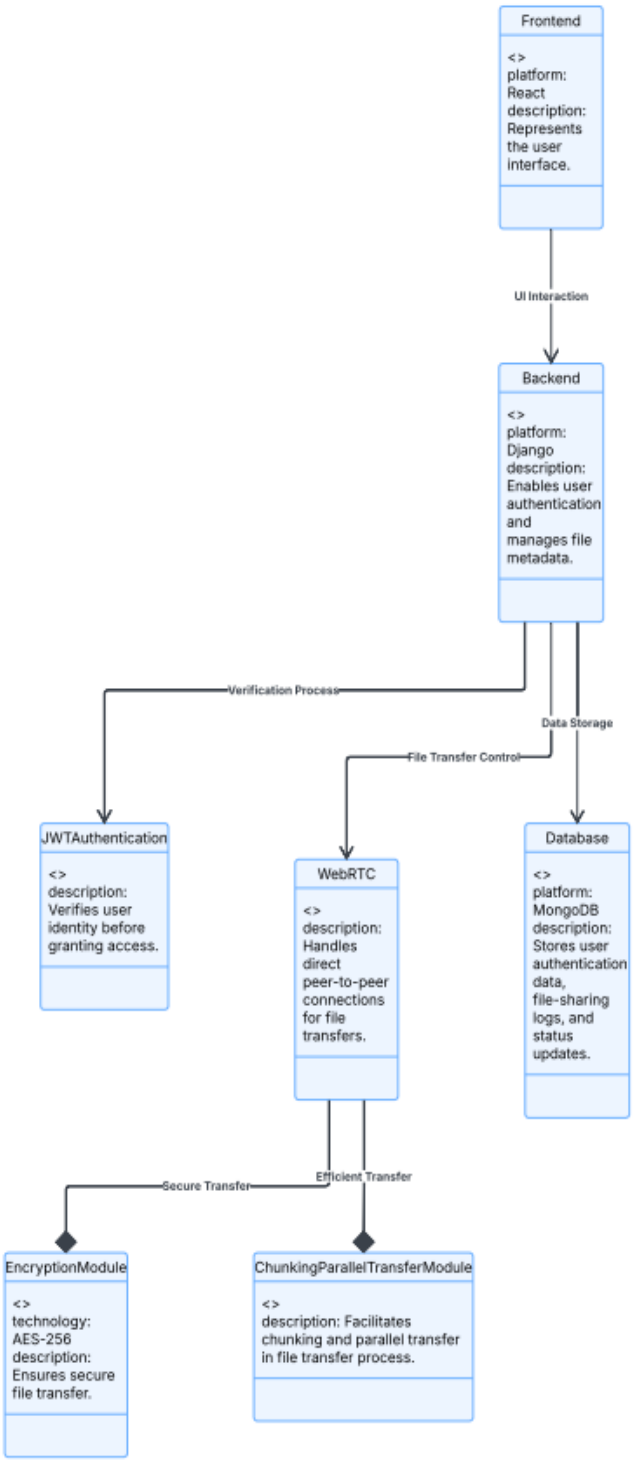
5. Appendices

Appendix A: Analysis models

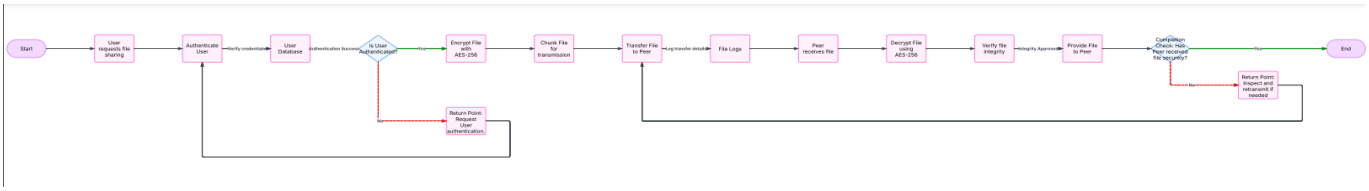
A.1 Use Case Diagram



A.2 System Architecture Diagram



A.3 Data Flow Diagram



6.Conclusion

The **Peer-to-Peer (P2P) File Sharing Platform** is designed to provide a **secure, efficient, and decentralized** solution for file transfers. By eliminating the reliance on centralized servers, the platform ensures **faster data exchange, reduced bandwidth costs, and enhanced privacy**.

This **Software Requirements Specification (SRS)** document outlines the **functional and nonfunctional requirements**, system architecture, and key design considerations necessary to develop a **scalable and user-friendly** P2P file-sharing system. The platform leverages **WebRTC and encryption techniques** to facilitate seamless and protected file transfers across a global network of users.

By implementing this platform, users can benefit from:

- **Decentralized and resilient file-sharing**
- **End-to-end encryption for security**
- **Scalable peer discovery and connectivity**
- **Optimized performance for high-speed transfers**
- **Cross-platform accessibility for ease of use**

Moving forward, the development of this platform will focus on **enhancing security, improving peer discovery mechanisms, and optimizing data transfer speeds** to ensure a robust and reliable file-sharing experience. This SRS serves as a foundation for developers, stakeholders, and system architects to build a platform that meets user needs while maintaining **efficiency, security, and scalability**.