**FILE TRANSFER APPLICATION**

1. **Requirement elicitation**
   1. **Description**

**The File Transfer Application** is designed to enable users to efficiently share and download files over a network using a **Peer-to-Peer (P2P)** and **Client-Server** architecture. The application aims to replicate functionalities similar to popular file-sharing applications like **BitTorrent** but with simplified features. Users can upload files to the network, which will then be distributed among peers for downloading. The application will ensure data integrity through hashing mechanisms, manage multiple peer connections, and provide a user-friendly interface for file sharing.

* 1. **Functional requirements**
     1. **User Authentication**

-Users should be able to register and log in to the application.

-Users should be able to manage their account settings.

* + 1. **File Uploading**

-Users should be able to select and upload files to share with others.

-The application should validate file types and sizes before uploading.

* + 1. **File Downloading**

-Users should be able to search for and download files from multiple peers simultaneously.

-The application should support resuming interrupted downloads.

* + 1. **File Integrity Verification**

-The application should calculate and verify file hashes (e.g., SHA-256) to ensure data integrity after download.

-Users should be notified if the file integrity check fails.

* + 1. **Peer Management**

-The application should allow users to connect to multiple peers for efficient file sharing.

-The application should display a list of available peers sharing a specific file.

* + 1. **Error Handling**

-The application should handle download errors gracefully and provide feedback to the user.

-Users should have the ability to retry downloading failed file segments.

* + 1. **Progress Monitoring**

-Users should be able to monitor upload and download progress through a visual interface.

-The application should provide notifications upon completion of file transfers.

* + 1. **File Metadata Management**

-The application should store and display metadata for shared files, including file name, size, type, and upload date.

* 1. **Non-functional requirements**
     1. **Performance**

-The application should support simultaneous downloads from at least five peers without significant performance degradation.

-File uploads and downloads should be completed within a reasonable time, ideally under five minutes for typical file sizes (e.g., 500 MB).

* + 1. **Scalability**

-The application should be able to accommodate an increasing number of users and files without a decline in performance.

* + 1. **Reliability**

-The application should ensure high availability, with minimal downtime during file transfers.

-Data integrity must be guaranteed throughout the file transfer process.

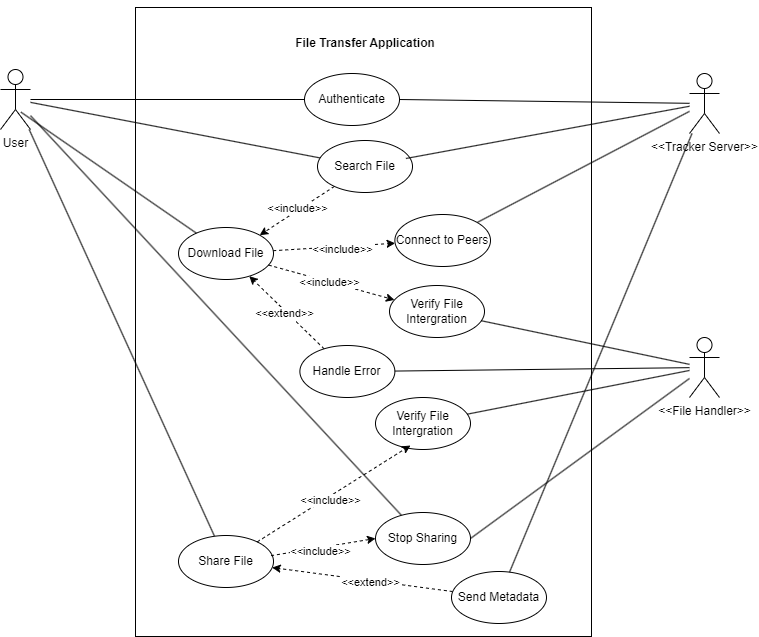
* 1. **Use case diagrams**
     1. **Actors**

|  |  |  |
| --- | --- | --- |
| **Actor ID** | **Actor** | **Description** |
| 1 | User | The end user of the application who interacts with the system to upload, download, and manage files. |
| 2 | Tracker Server | A server that manages the connections between peers, facilitates file sharing, and maintains a directory of available files and peers. |

* + 1. **Use case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Use case ID** | **Use case name** | **Description** | **Notes** |
| UC-1 | Authenticate | User register, log in or log out to the service |  |
| UC-2 | Share file | Users can upload files to the system, making them available for other peers to download. This is a critical function of the application. | Important |
| UC-3 | Search file | Users can search for specific files within the network, enabling them to find files shared by other peers. |  |
| UC-4 | Download file | Users can download files from multiple peers simultaneously, ensuring fast and efficient file transfer. This function is vital for the application's performance. | Important |
| UC-5 | Stop Sharing | Users can stop their file sharing immediately |  |

* + 1. **Diagram**



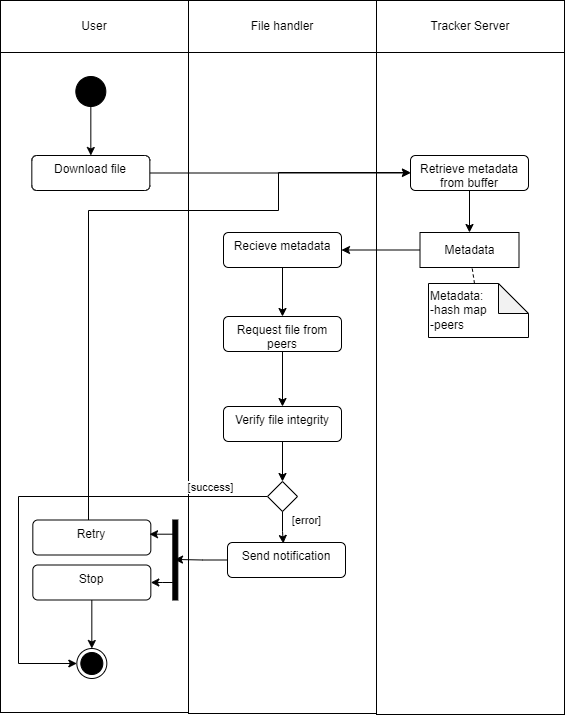
* + 1. **Use case: Share File**

|  |  |  |  |
| --- | --- | --- | --- |
| Use case ID: | UC-2 | | |
| Use case name: | Share File | | |
| Created by: | Lê Văn Anh Khoa | Last updated by: | Lê Văn Anh Khoa |
| Date Created: | 06/10/2024 | Date Last Updated: | 06/10/2024 |
| Actors: | User, Tracker Server, File Handler | | |
| Description: | User uploads a file to the application, making it available for other peers to download. The application verifies the file's integrity during the sharing process to ensure data reliability. Metadata regarding the shared file is communicated to the tracker server. | | |
| Trigger: | User upload and initiates the sharing process. | | |
| Includes use case: | 1. Verify File Integrity 2. Stop Sharing | | |
| Extends use cases: | Share Metadata | | |
| Preconditions: | 1. The user must be authenticated (logged in) to share files.   2. The file to be shared must be selected by the user and must comply with any size or format restrictions. | | |
| Post-conditions: | 1. The file is successfully uploaded to the application. 2. The integrity of the file is verified. 3. Metadata about the shared file is stored in the tracker server. 4. When the user stop sharing file, the buffer in server is cleaned and notification to other clients are send. | | |
| Normal Flow: | 1. The user selects a file to share. 2. The application verifies the file format and size. 3. The user confirms the upload action. 4. The application verifies the file's integrity using a hashing algorithm. 5. Upon successful verification, the application sends the file metadata to the tracker server. 6. The tracker server updates its records to include the newly shared file. 7. The user stop sharing file 8. The tracker server update its records and send notification for the clients who are downloading this file | | |
| Alternative Flow: | 2a. If the file format or size does not meet the application’s requirements, the user receives an error message and must select a different file.  4a. If the file integrity check fails, the application notifies the user, and the upload process is aborted.  5a. If there is an issue with the tracker server during metadata sharing, the application informs the user and may retry the process. | | |
| Notes and issues: |  | | |

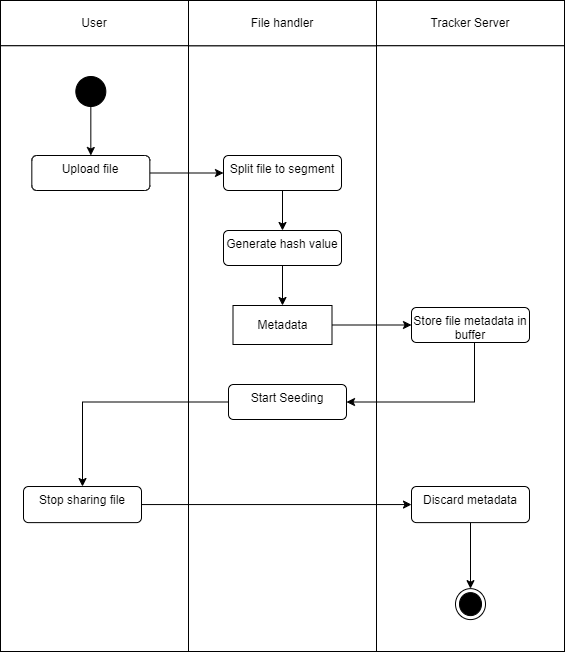
* + 1. **Use case: Download File**

|  |  |  |  |
| --- | --- | --- | --- |
| Use case ID: | UC-4 | | |
| Use case name: | Download File | | |
| Created by: | Lê Văn Anh Khoa | Last updated by: | Lê Văn Anh Khoa |
| Date Created: | 06/10/2024 | Date Last Updated: | 06/10/2024 |
| Actors: | User, Tracker Server, File Handler | | |
| Description: | User searches for a file and downloads it from multiple peers simultaneously. The file's integrity is verified during and after the download, ensuring that the file is complete. The application handles potential errors. | | |
| Trigger: | User searches for a file and initiates a download through the application. | | |
| Includes use cases: | 1. Connect to peers 2. Verify File Integrity | | |
| Extends use cases: | 1. Handle error | | |
| Preconditions: | The user must have successfully searched for the file they wish to download. | | |
| Post-conditions: | 1. The file is successfully downloaded and stored on the user’s device. 2. If any errors occur during the download, they are handled appropriately, and the user is notified. | | |
| Normal Flow: | 1. The user selects the file and initiates the download. 2. The application connects to multiple peers to download file segments concurrently. 3. As file segments are received, the application verifies their integrity using a hashing mechanism. 4. The application assembles the file from the downloaded segments. 5. Once the download is complete, the application verifies the integrity of the entire file. 6. The application notifies the user that the download is complete and the file is ready to use. | | |
| Alternative Flow: | 2a. If a peer disconnects during the download, the application attempts to reconnect or find alternative peers to complete the download.  3a. If the file integrity verification fails, the application discards the corrupted segments and attempts to re-download the missing or corrupted pieces. | | |
| Notes and issues: |  | | |

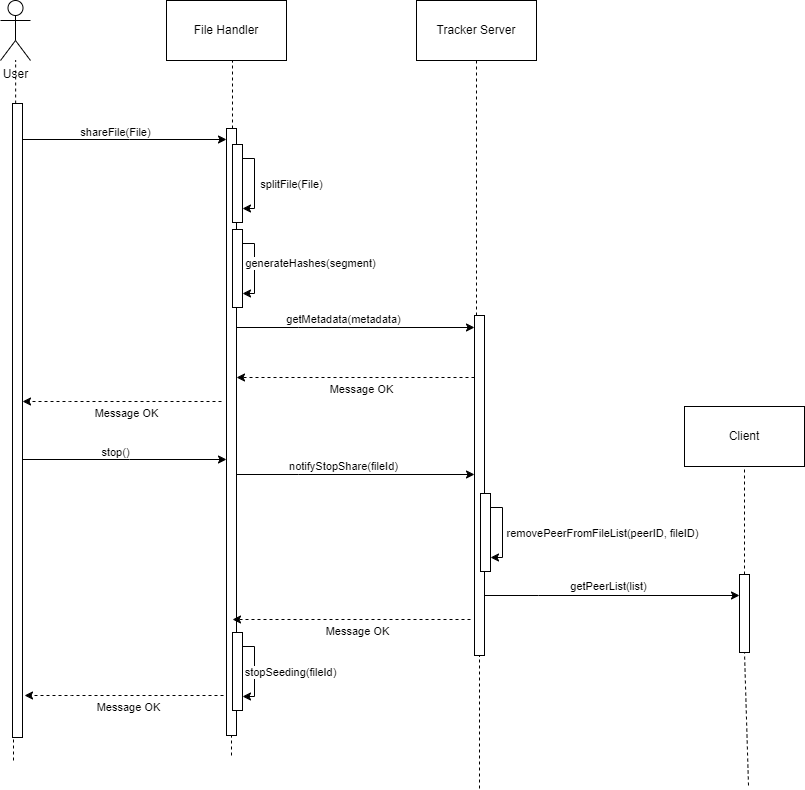
1. **System modeling**
   1. **Activity diagram**
      1. **Download file**



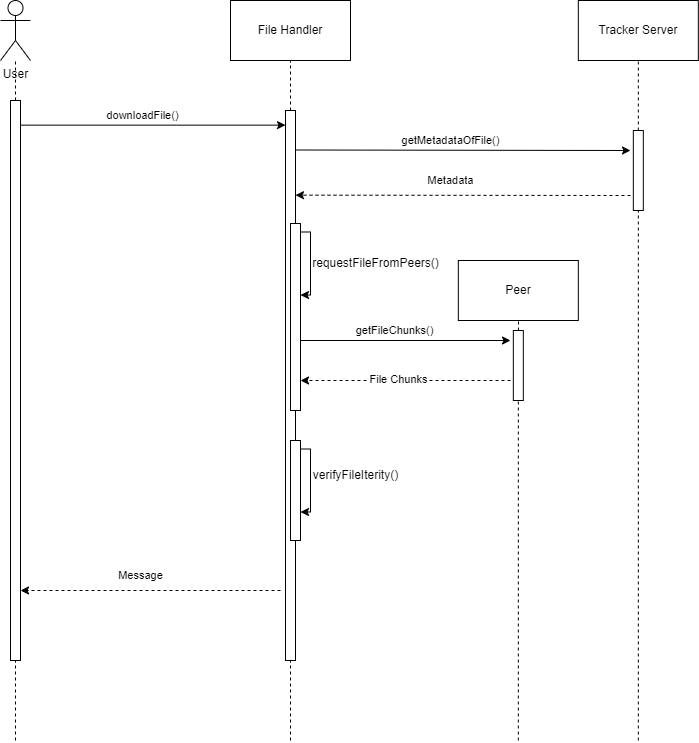
* + 1. **Share file**



* 1. **Sequence diagram**
     1. **Share file**

****

* + 1. **Download file**

****

* 1. **Class diagram**

1. **Data requirements**
2. **Implementation**