



PES University, Bengaluru
(Established under Karnataka Act 16 of 2013)

Department of Computer Science & Engineering

Session: Jan – May 2022

Object Oriented Analysis and Design with Java - Laboratory
UE20CS352

Mini Project Report

Secure File Sharing System

Authors:

PES1UG20CS583 – S.Niharika

PES1UG20CS588 – Sahana Rao

PES1UG20CD 591 – Shreya

(6th Semester, Section J)

PROBLEM STATEMENT

The aim of the project is to build an end-to-end secure file sharing system which mimics the P2P architecture. Files are securely shared via a room which is created by the client through Advanced encryption algorithm. The invite link of the sharing platform which in this case is a room is created for the peers who want to access and download files. The link has to be submitted by each peer to join the network to send or receive the files.

DETAILED ANALYSIS

There are two actors in the scene namely Client and Peer. The client creates a room. The peer joins the room with the link provided by the client. Both the client and the peer can send the files. The files in the room can be downloaded or deleted by any peer in the room. On the sender side the file is encrypted (AES Encryption algorithm) using the secret key before sending the file. The encrypted files are decrypted on the receiving end using the secret key available in the room to validate the file. After the decryption the file is downloaded on the receiving end.

In order to account for the file not being corrupted, the concept of checksum (calculated using SHA-256 algorithm) is used to verify the integrity of the file. The encrypted file here is being sent to the peers in the room using different sockets in the local system presently. All the files being sent/transmitted in the room are stored in the mongodb database for future reference of the ongoing session. Each user is assigned a database to keep track of his files used in the session.

GITHUB LINK

https://github.com/Sahana-L-Rao-M/OOAD_SecureFileSharing_583_588_591

MOTIVATION

To address the growing concerns about the security of online data sharing. With the increasing reliance on digital communication and the sharing of sensitive information online, there is a growing need for secure and reliable methods of file sharing.

Existing file sharing solutions are often limited in their security features, leaving them vulnerable to cyber threats such as data breaches, hacking, and identity theft.

WHY THIS PROJECT?

Protects Sensitive Information: A secure file sharing system ensures that sensitive information, such as personal or financial data, intellectual property, or confidential business information, is protected from unauthorized access or data breaches.

Control: With a secure file sharing system, users have more control over the information they share, who they share it with, and how it is accessed. This allows organizations to implement access controls and permission levels, ensuring that files are only accessed by authorized parties.

OBJECTIVES AND GOALS

1. Develop a secure and reliable file sharing system that utilizes advanced encryption algorithms, user authentication, access control to ensure the confidentiality, integrity, and availability of shared files.
2. Provide a user-friendly experience that enables users to easily and securely share files with authorized parties.
3. Implement access controls and permission levels that allow organizations to control who can access and share files, and provide detailed logs and audit trails of file access and sharing activity.

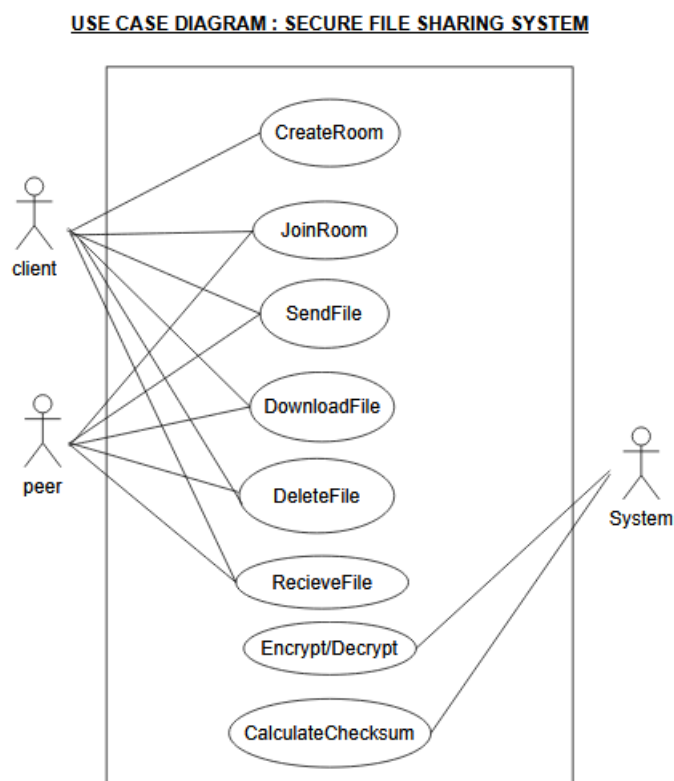
4. Provides user control where user can control over the information they share and who they share it with

BACKGROUND

Secure file sharing is an essential aspect of modern-day communication and collaboration. With the increasing reliance on digital communication, there is a growing need for secure file sharing tools that enable users to exchange information safely and efficiently. Secure file sharing tools ensure that files are transmitted and stored securely, protecting them from unauthorized access and theft.

MODELS (Use Case and Class Models)

USE CASE DIAGRAM:



USE CASE DESCRIPTIONS

1. CREATE ROOM

1. Name: CreateRoom
2. Summary: Create a room where peers can share files
3. Actor: Client/Peer
4. Preconditions: The Client has entered the platform
5. Description:
 - Click on “create room”
 - Click on “generate room link”
 - Click on “share link” and share it with other peers and the peers can join the room if the room limit has not been exceeded
6. Exceptions: The room limit exceeds
7. Post-condition: Peers can join and share files
8. Alternate flow: Restart the system and re-login with valid credentials
9. Assumptions: The user is authorized

2. DELETE FILE

1. Name: DeleteFile
2. Summary: On receiving the file, drop the file if it was corrupted during transfer.
3. Actor: Receiver/Peer
4. Preconditions: File is uploaded and is ready to be sent over the network
5. Description:
 - Calculate the file parity at the sender side, which upon received at the receiver side is used to verify the integrity of the file
 - It is added as metadata of the file
 - It is encrypted, sent over the network and received on the other side, and decrypted.

- The metadata appended to the file is used to check the integrity of the file
 - If the file is found to be tampered, it is dropped immediately and a new request to resend the file goes to the sender, else accept the file.
6. Exceptions: The file doesn't reach the receiver side
 7. Post-condition: Receiver can download the files
 8. Alternate flow: Secure file preview that allows users to view files without downloading them
 9. Assumptions:
 - User Authentication
 - Secure storage of files

3. SendFile

1. Name: SendFile
2. Summary: To send file over the network
3. Actor: Sender/Peer
4. Preconditions: Upload the file to be sent
5. Description:
 - On starting the connection, keys generated for secure transmission i.e., encryption
 - Share the public key to the receiver
 - Encrypt the file with the key already generated
 - Send the file to the receiver
 - Receive at the receiver side
 - Close connection
6. Exceptions: File doesn't exist in the DB
7. Post-condition: Receiver can receive the sent file
8. Alternate Flows: The system can use a secure transfer protocol such as SFTP or HTTPS to ensure that the file is transmitted securely over the internet.
9. Assumption: Encryption technique is common throughout the groups.

4. Receive File

1. Name: Receive File
2. Summary: Receive the file sent over the network.
3. Actor: Peer/Receiver
4. Preconditions: The Sender has pushed a file to the fileDB
5. Description:
 - Click on “receive” file
 - Click on “verify” to verify the contents of the file by decrypting the file using the public key of the sender.
 - Click on “download” to download the file after successful verification.
6. Exceptions: Verification fails.
7. Post-condition: The file can now be downloaded by the receiver.
8. Alternate flow: Create a new account and send request
9. Assumptions: No third-party access, File is not corrupted

5. Upload File

1. Name: Upload File use case
2. Summary: Select the file and upload the file in the FileDB
3. Actor: Client/Peer
4. Pre-conditions: The Client has entered the platform
5. Description:
 - Click on “upload” to upload the file
 - Select the file from the peer local system.
 - Click on “OK” to start uploading.
 - The uploading actions have start, pause and stop options.
6. Exception: File is invalid or corrupted.
7. Post-condition: Send the file for encryption.
8. Alternate Flow: The uploader can be prompted for a second form of authentication before uploading the file.

9. Assumptions: Only one file is sent over the network

6.Download File

1. Name: Download File use case

2. Summary: Download the shared file in the room

3. Actor: Receiver/Peer

4. Pre-conditions: The file had been received and verified by decryption.

5. Description:

- Click on “download” to download the file.
- Select the destination to download in the peer’s local system.
- Click on “OK” to start downloading.
- The downloading actions have start, pause and stop options.

6. Exception: File is invalid or corrupted (verification failed).

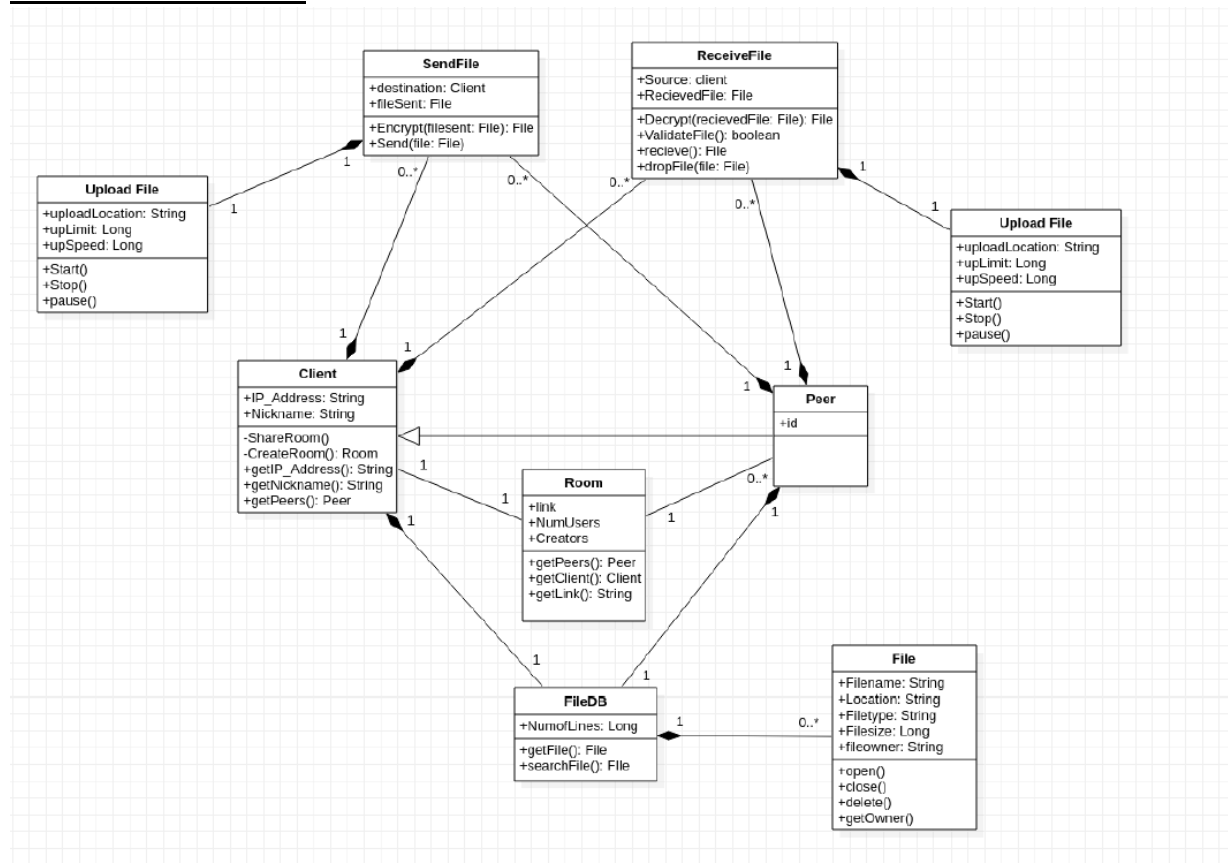
7. Post-condition: The peer can continue being in the room to share and receive files.

8. Alternate flow: Download Notifications that alert the file owner when file has been downloaded successfully

9. Assumptions:

- Secure file download
- User authentication
- Secure storage of files

CLASS DIAGRAM



ARCHITECTURE PATTERNS, DESIGN PRINCIPLES AND DESIGN PATTERNS

Design principles:

The following SOLID principles were kept in mind while coding the project:

- **S - Single Responsibility Principle (SRP)**

All files in the model have only one logical responsibility. E.g.: Classes like `sendFile` and `receiveFile` only take the responsibility of sending and receiving files respectively. Hence encrypting and decrypting is delegated to another class called `Crypto`.

- **O - Open-Closed Principle (OCP)**

Class `peer` inherits the properties of `client` class by extending its properties.

Class CustomMultipartFile implements MultipartFile with additional functionalities.

- **L - The Liskov Substitution Principle (LSP)**

A client object can be replaced with a peer object without any flaws.

- **I - Interface Segregation Principle (ISP)**

There are no interfaces or classes with functionality which overburden them or ones which are not used by the client.

- **D - Dependency Inversion Principle (DIP)**

There is no high-level class that directly depends on classes at lower levels.

Design Patterns:

- **Singleton - classes that use singleton:**

- a. SendFile
- b. ReceiveFile

- **Factory -**

- a. UserFactory that creates:
- b. Peer
- c. Client

TOOLS AND FRAMEWORK

Tools and Frameworks Used:

MVC: model-view-controller

The Model-View-Controller (MVC) framework separates an application into three main logical components Model, View, and Controller.

Three important MVC components are:

- **Model** - It includes all the data and its related logic
- **View** - Present data to the user or handles user interaction
- **Controller** - An interface between Model and View components

Using: Spring MVC Framework -

Model - Classes:

- - Client
- - Crypto
- - ReceiveFile
- - SendFile
- - Peer
- - Room

Controller - Classes

- - FileController

Views

- HTML files with minimal vanilla JavaScript.

Maven

- Maven is used to handle the dependencies in the java project using POM (project object model).
- It helps in downloading the dependencies, which refers to the libraries or JAR files specific to the versions required in the project.

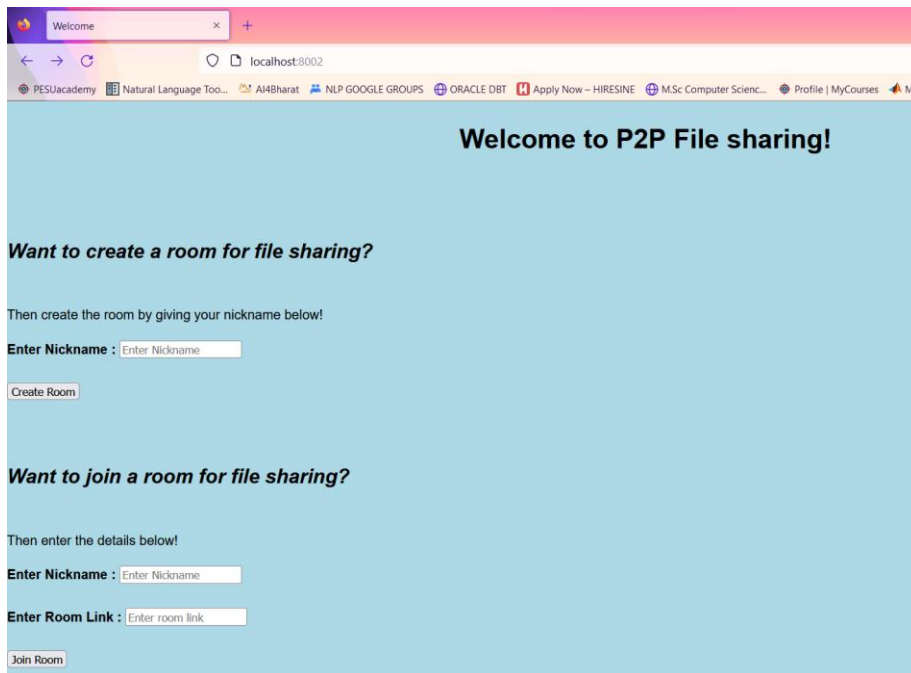
OUTPUT SCREENSHOTS:

Terminal:

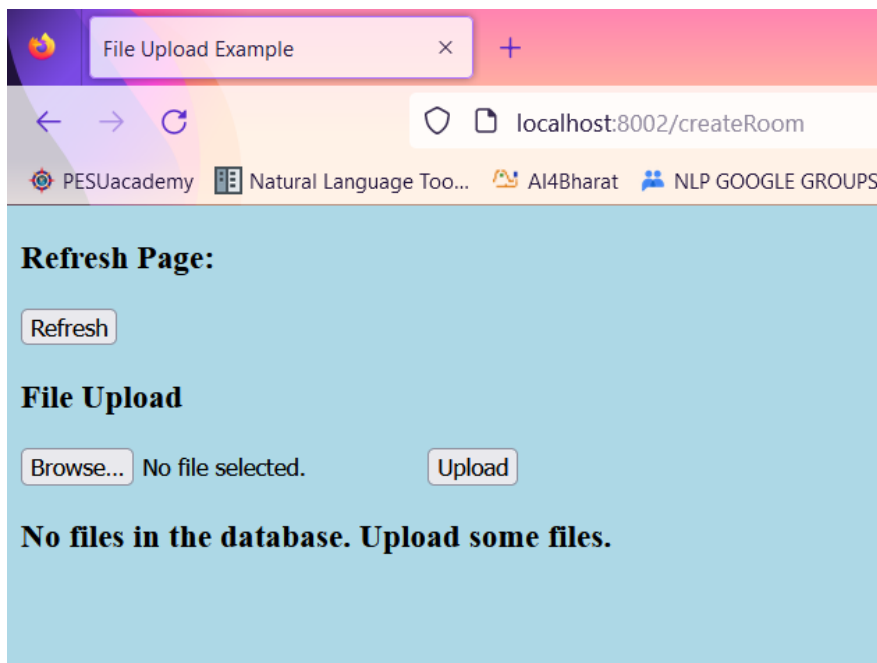
```
D:\PES1UG20CS588\SEM6\OOAD\PROJECT\secure-file-sharing-springMVC>mvnw spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.filesharingsystem:filesharingsystem >-----
[INFO] Building filesharingsystem 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] >>> spring-boot-maven-plugin:2.5.11:run (default-cli) > test-compile @ filesharingsystem >>>
[INFO]
[INFO] --- maven-resources-plugin:3.2.0:resources (default-resources) @ filesharingsystem ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Using 'UTF-8' encoding to copy filtered properties files.
[INFO] Copying 1 resource
[INFO] Copying 5 resources
[INFO]
```

Loading page:

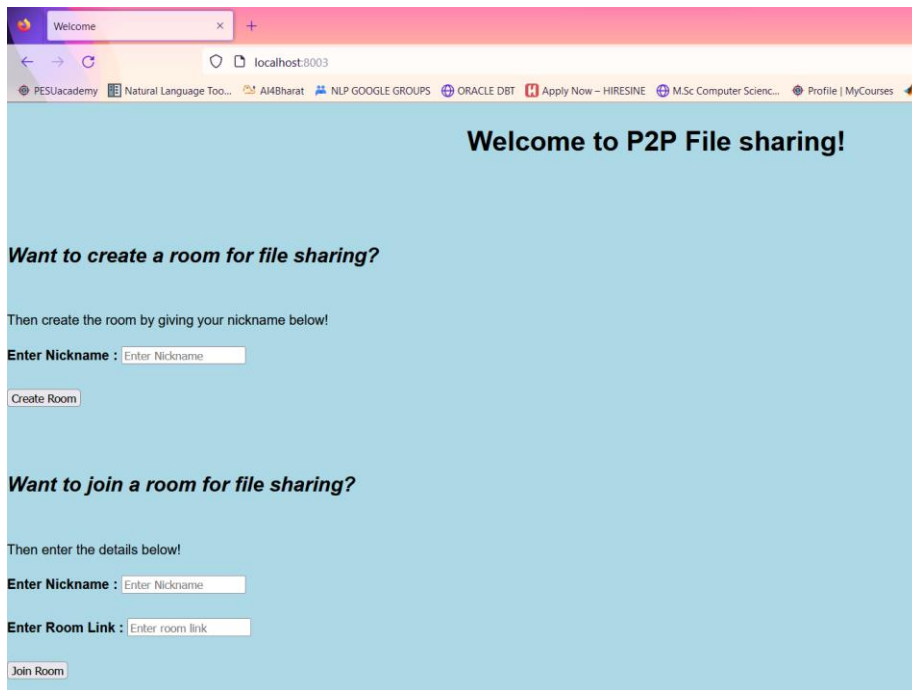
Client (localhost:8002)



Creates Room:



Peer1(localhost:8003)



The screenshot shows a web browser window with the address bar set to localhost:8003. The page has a light blue background and a pink header. The main heading is "Welcome to P2P File sharing!". Below this, there are two sections: "Want to create a room for file sharing?" and "Want to join a room for file sharing?". Each section includes a prompt to enter a nickname, a text input field, and a button to either "Create Room" or "Join Room".

Welcome to P2P File sharing!

Want to create a room for file sharing?

Then create the room by giving your nickname below!

Enter Nickname :

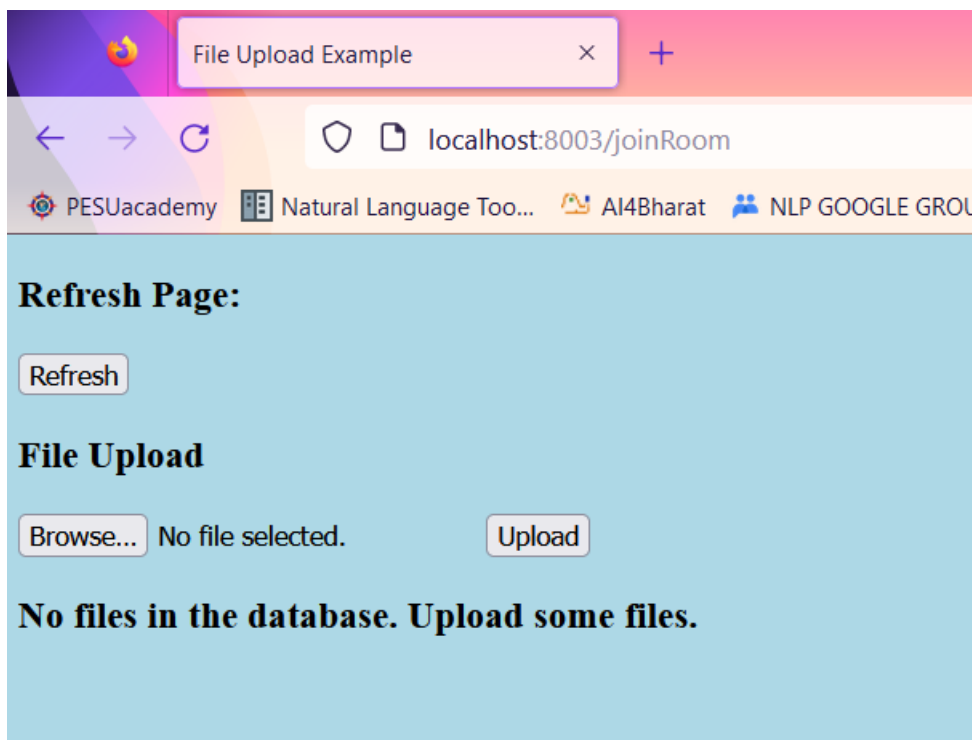
Want to join a room for file sharing?

Then enter the details below!

Enter Nickname :

Enter Room Link :

After joining room created by client:



The screenshot shows the same web browser window, but the address bar now shows localhost:8003/joinRoom. The page content has changed to show a "Refresh Page:" section with a "Refresh" button. Below this is a "File Upload" section with a "Browse..." button, the text "No file selected.", and an "Upload" button. At the bottom, there is a message: "No files in the database. Upload some files.".

File Upload Example

localhost:8003/joinRoom

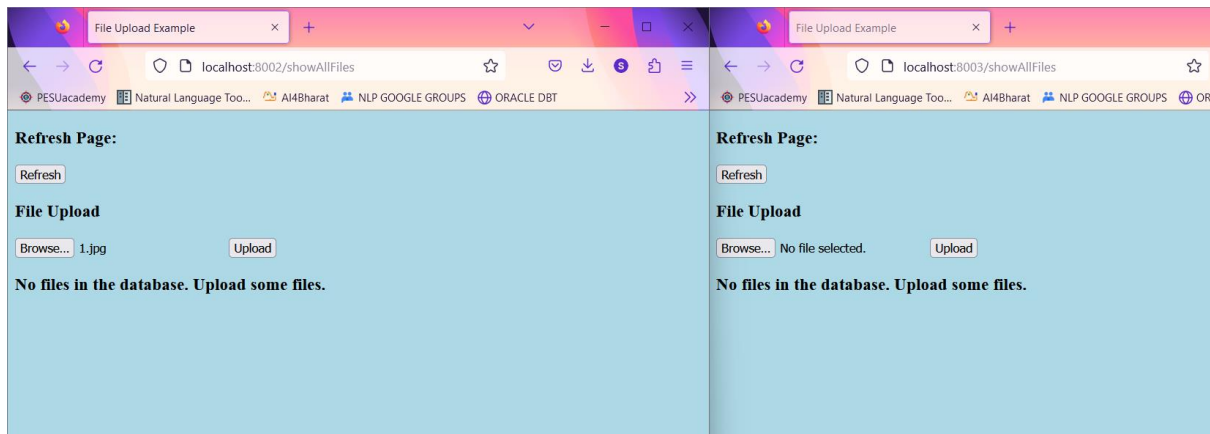
Refresh Page:

File Upload

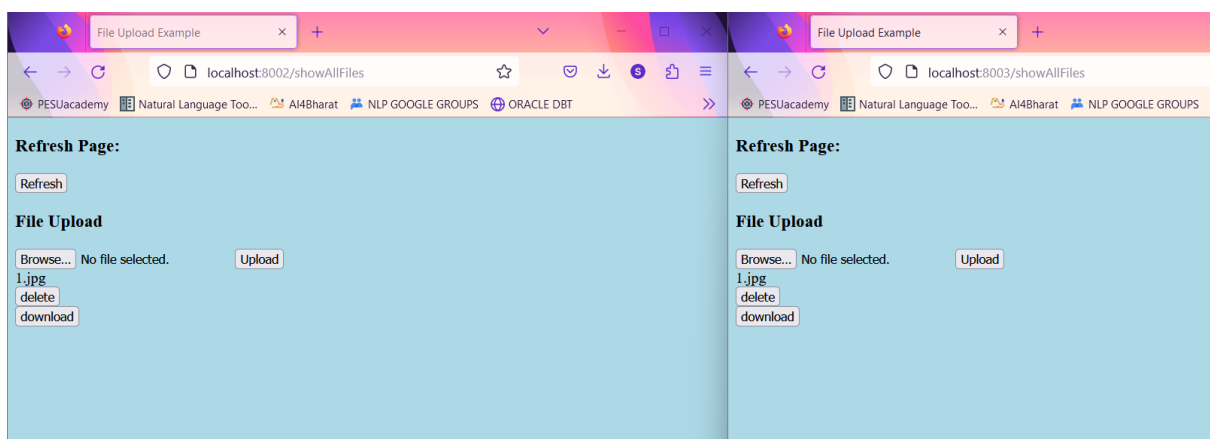
No file selected.

No files in the database. Upload some files.

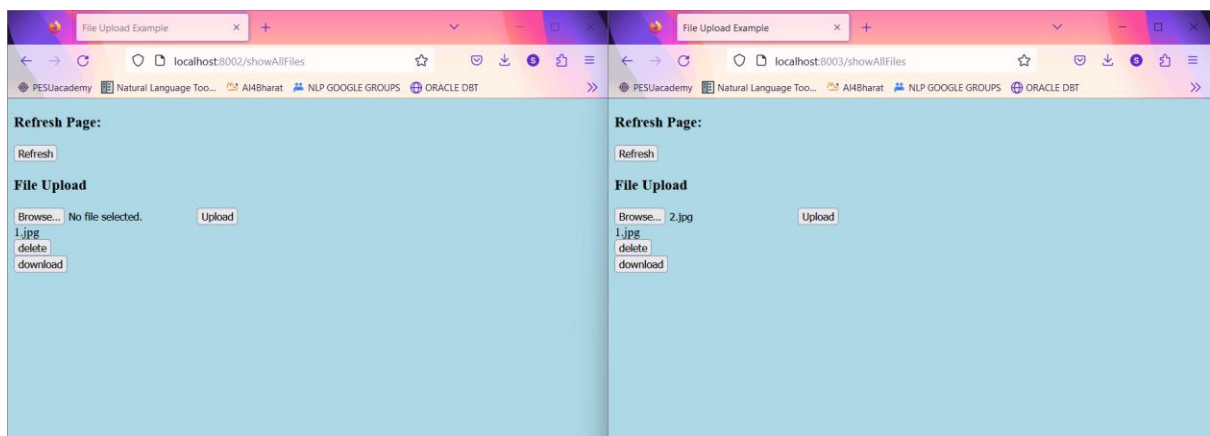
Client browses file to upload(1.jpg):



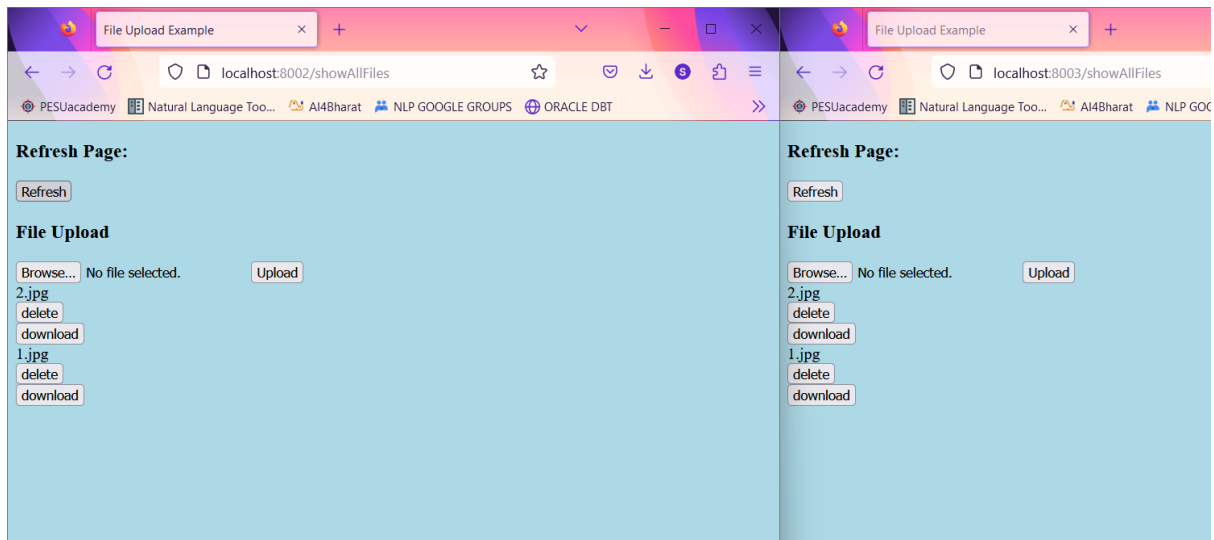
Client sends and Peer receives file:



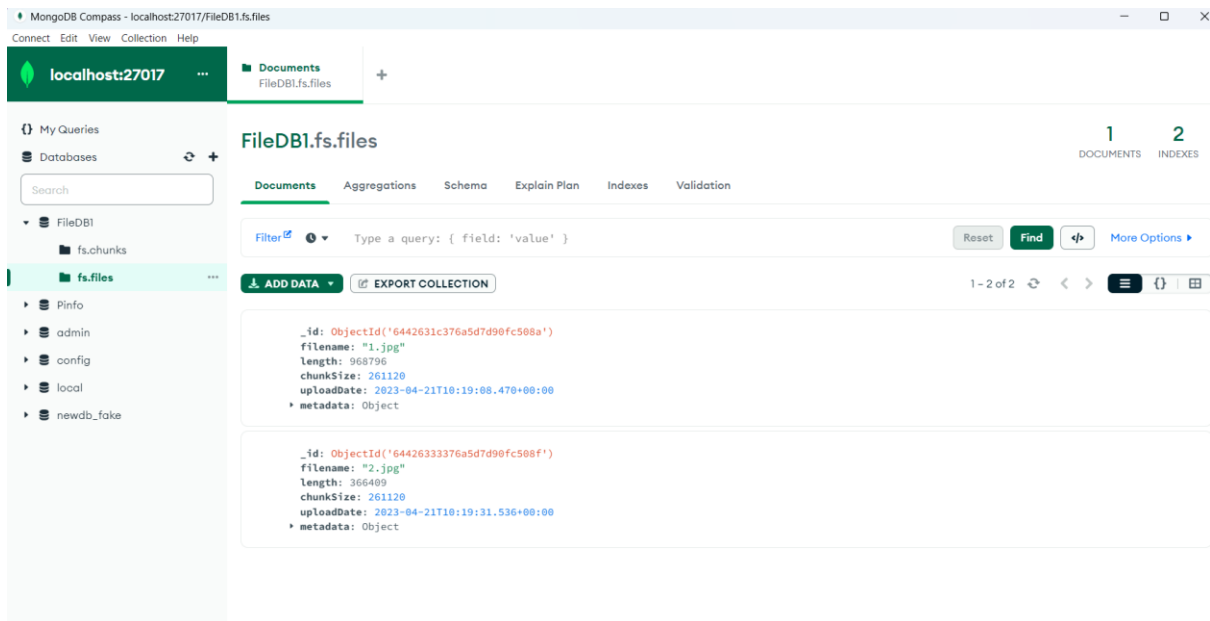
Peer browses file(2.jpg)



Client receives the file

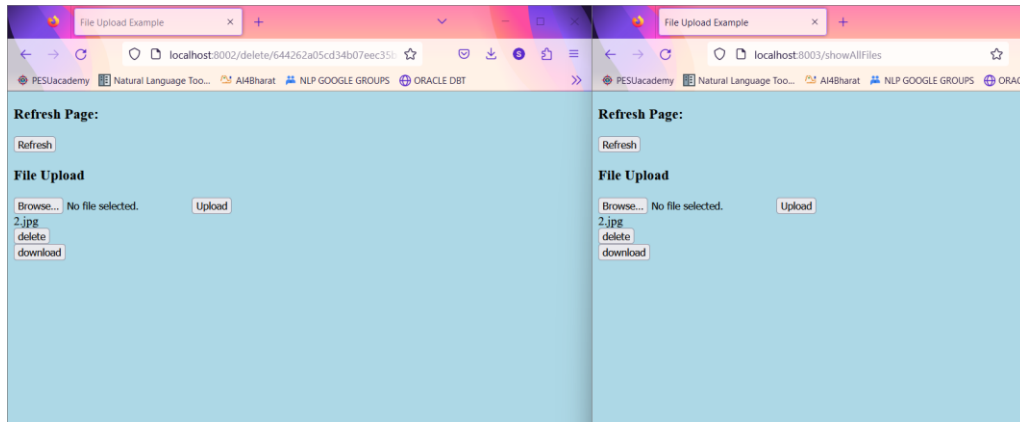


After uploading 2 files



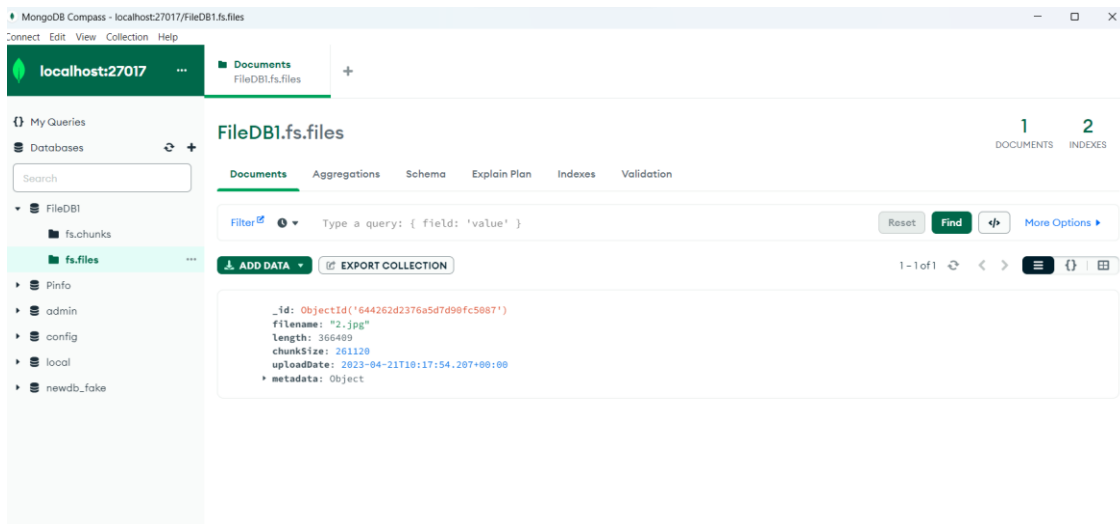
Deleting file:

Delete 1.jpg:



After deleting:

In database:



TEAM CONTRIBUTIONS

S.NIHARIKA	Room, SendFile, ReceiveFile, FileController.
SAHANA RAO	Crypto, DBService, myFile, CustomMultiPartiteFile.
SHREYA	UserFactory, Peer, Client, FileController