



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
UE19CS353**

Mini Project

Report on

Secure File Sharing System

By:

PES1UG19CS519 – Sumukh Raju Bhat

PES1UG19CS497 – Sowmya M

PES1UG19CS469 – Shreyas Vinayaka Basri KS

6th Semester, Section 'H'

Project Description

The aim of the project is to build an end to end secure file sharing system which mimics the P2P architecture. Files are shared via a room which is created by the client. 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. The files being sent are encrypted with AES (Advanced Encryption Standard) encryption algorithm. On the sender side the file is encrypted using the secret key before sending the file.

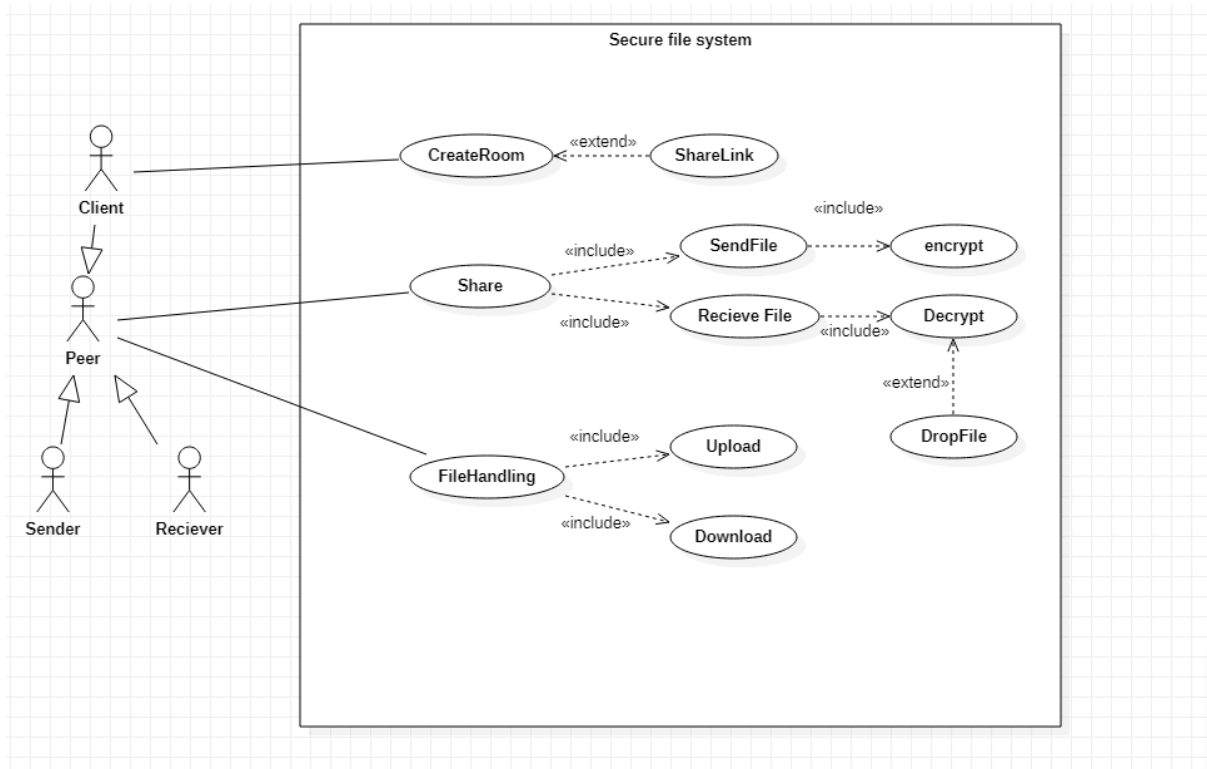
The encrypted files are decrypted on the receiving end using the secret key available in the room to validate the file. After the decryption the file is downloaded on the receiving end. In order to account for the file not being corrupted, the concept of checksum is used to verify the integrity of the 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 Repository:

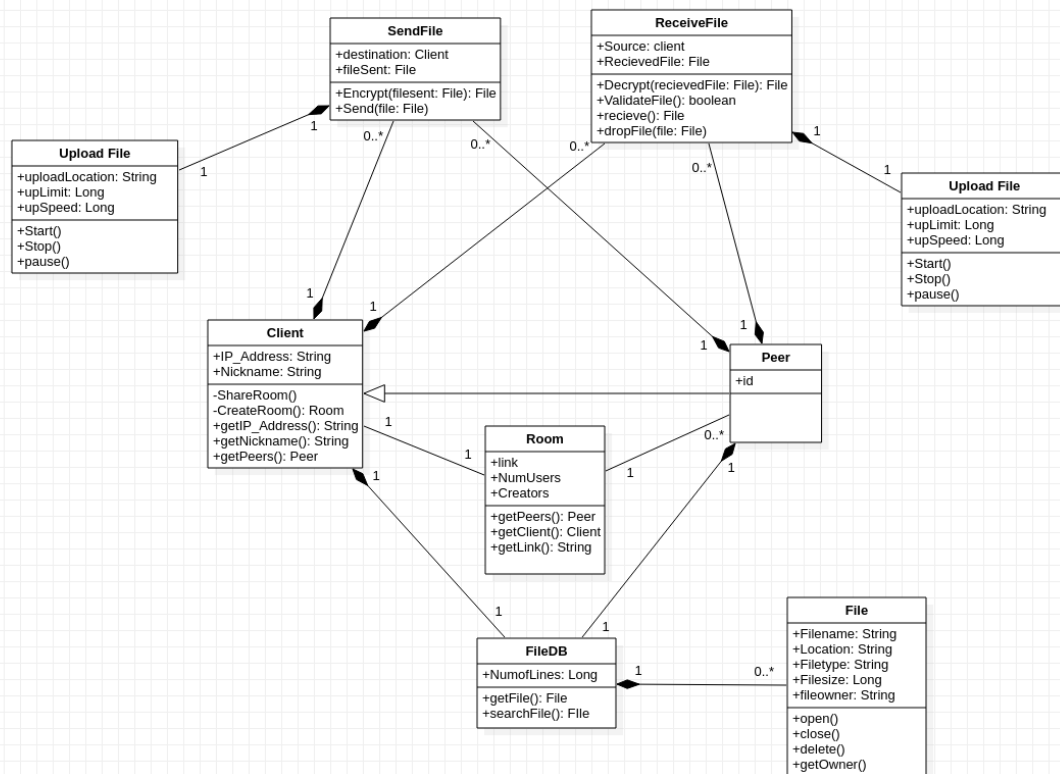
<https://github.com/sumukhbhat2701/secure-file-sharing-springMVC>

Analysis and Design Models

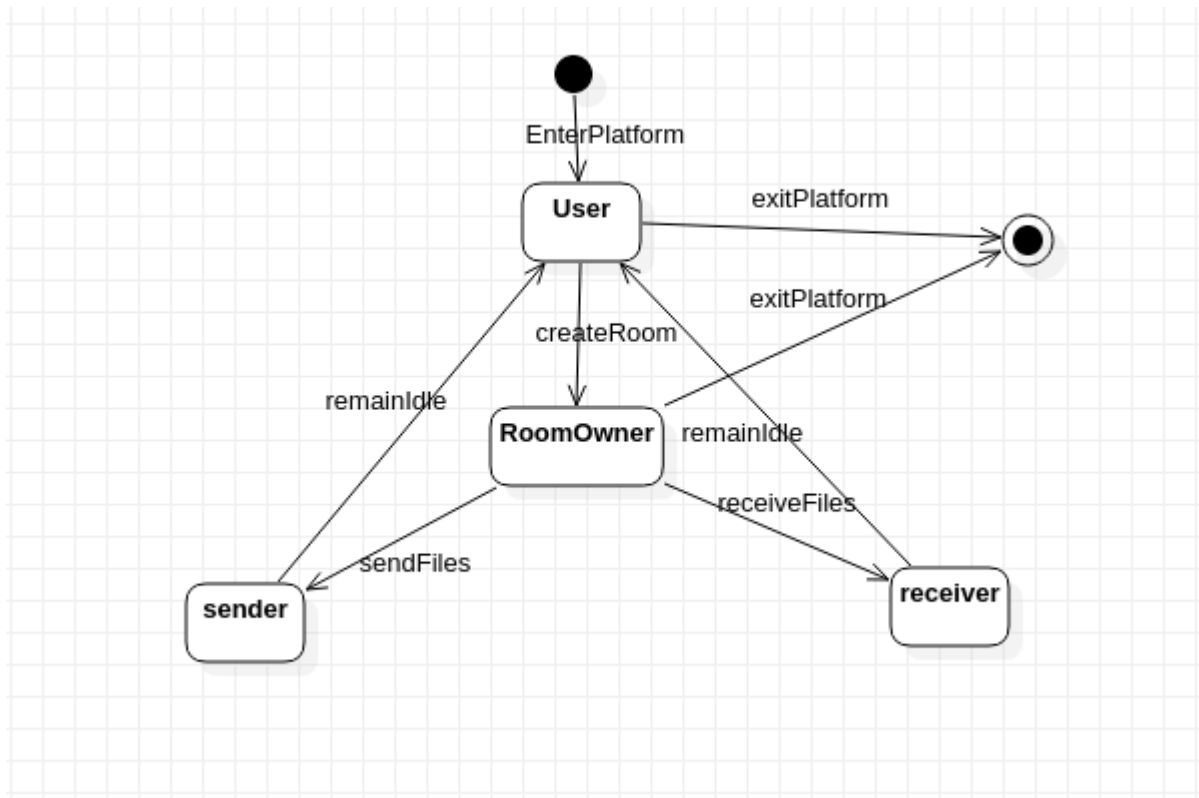
1. Use Case Diagram:



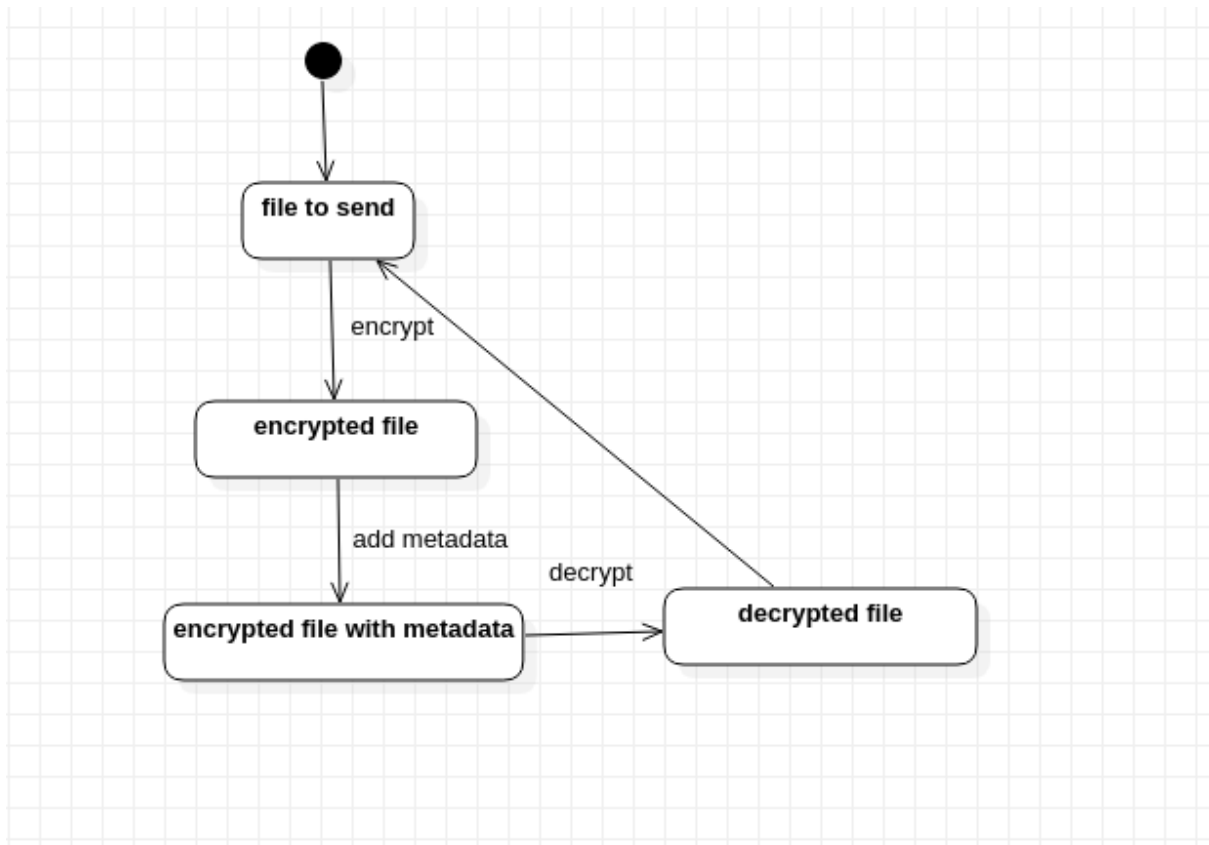
2. Class diagram:



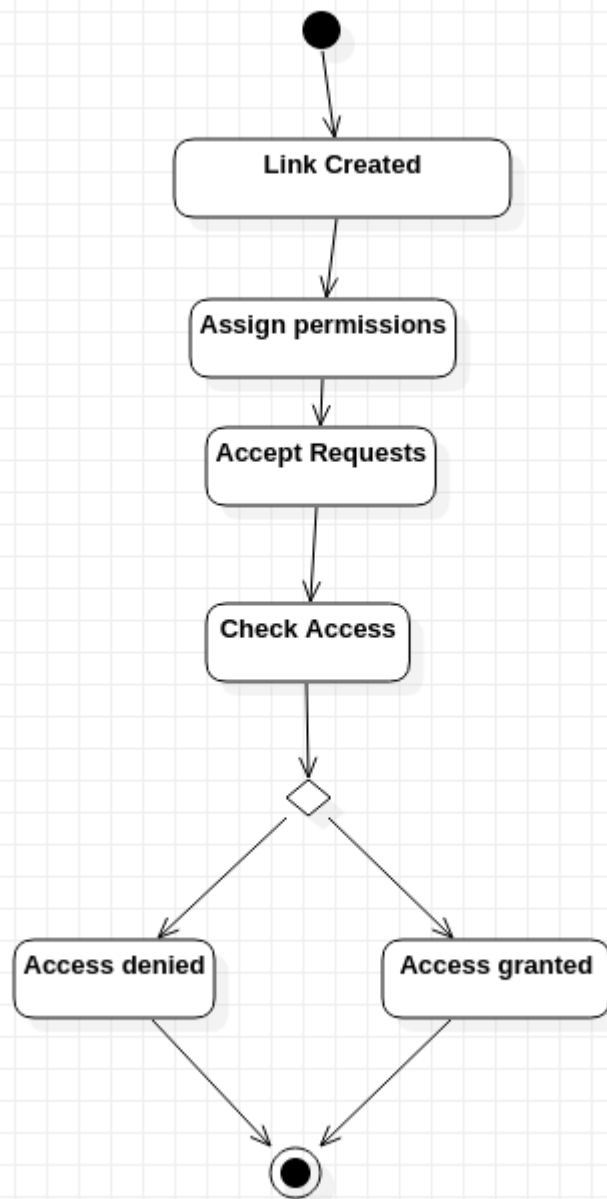
3. State diagram for the classes with temporal behaviour Client Object



File Object

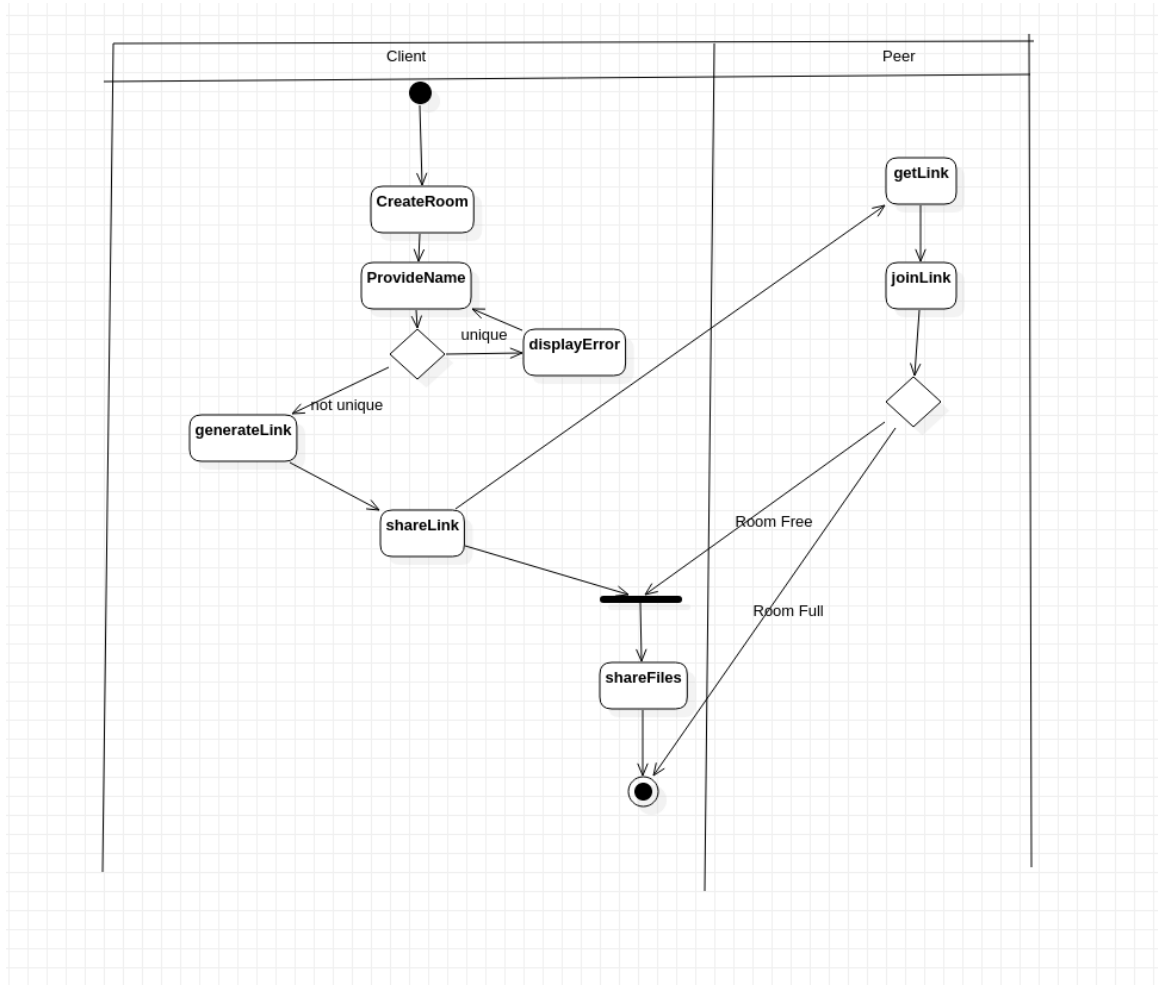


Link accessibility



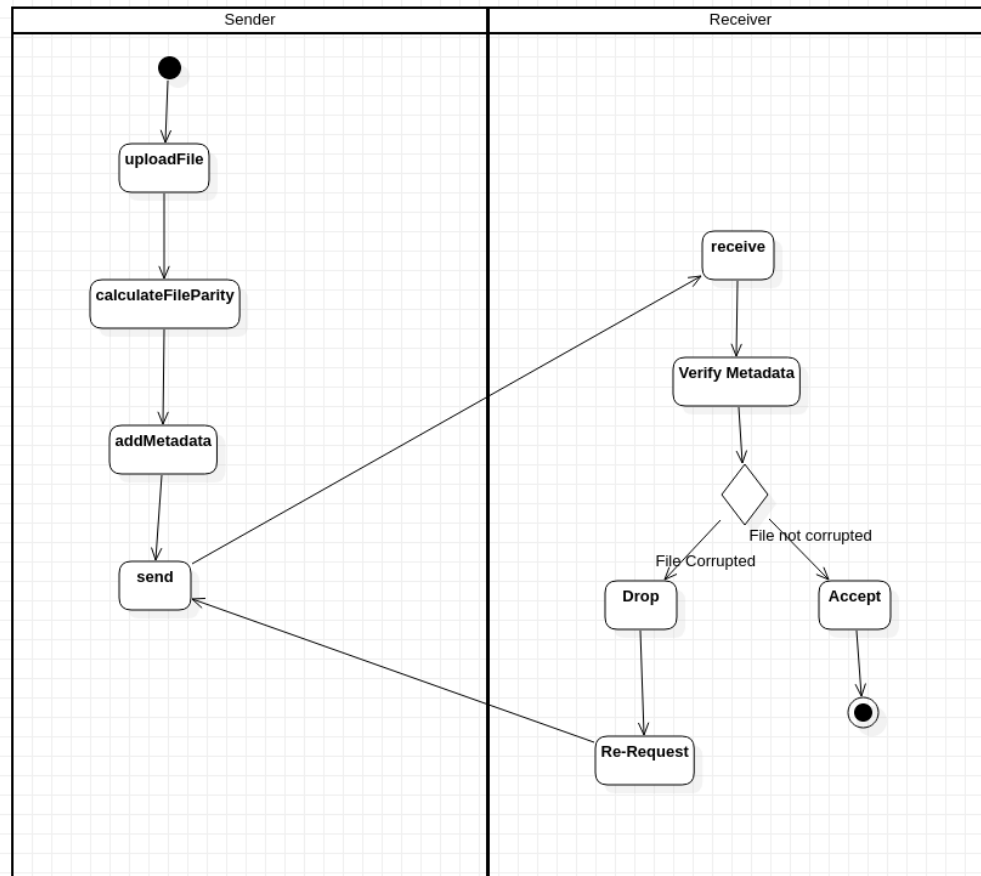
4. Activity diagram for the use-cases implemented

(a) CreateRoom Use Case:



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

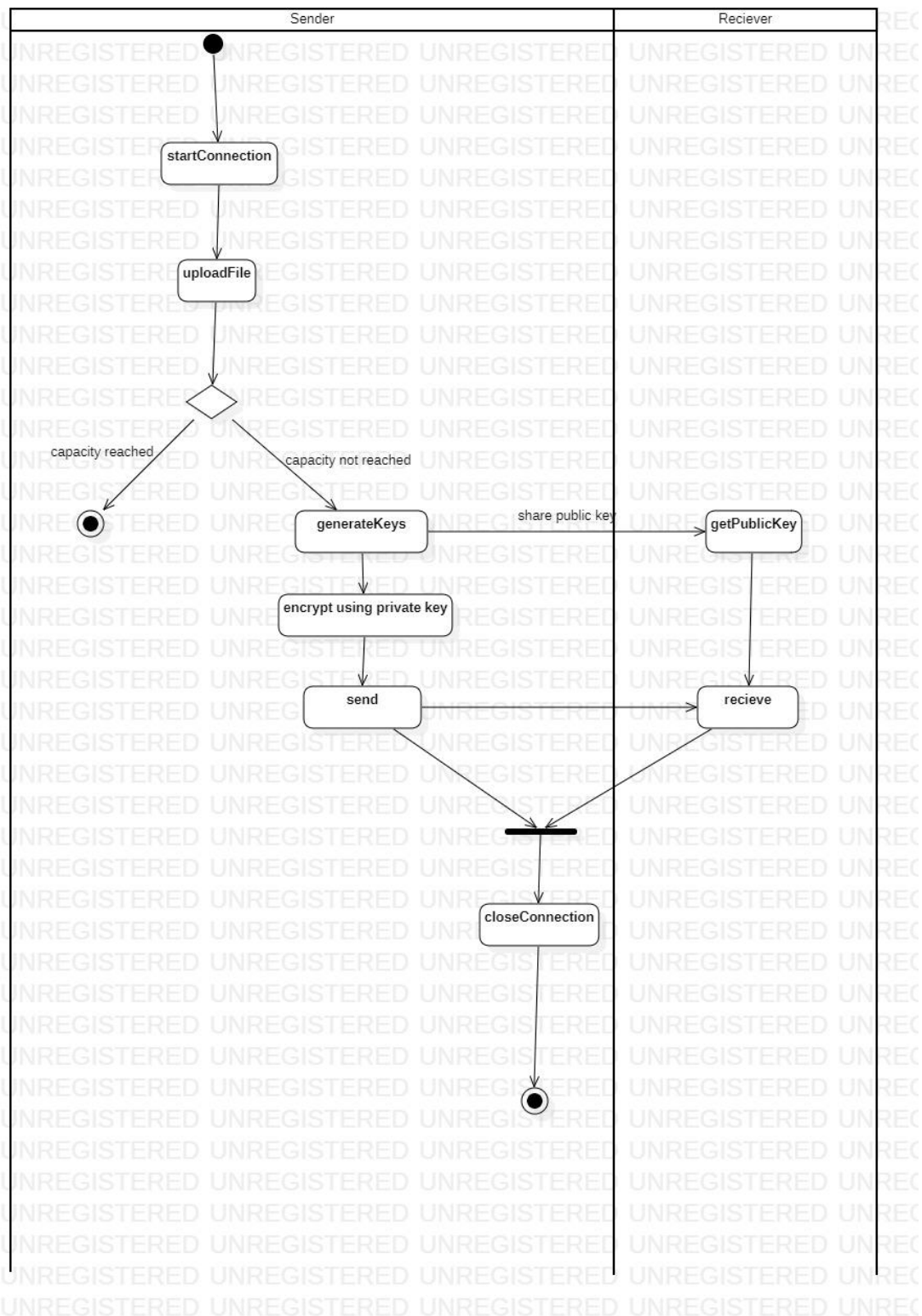
(a) DropFile Use Case



1. Name: DropFile
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

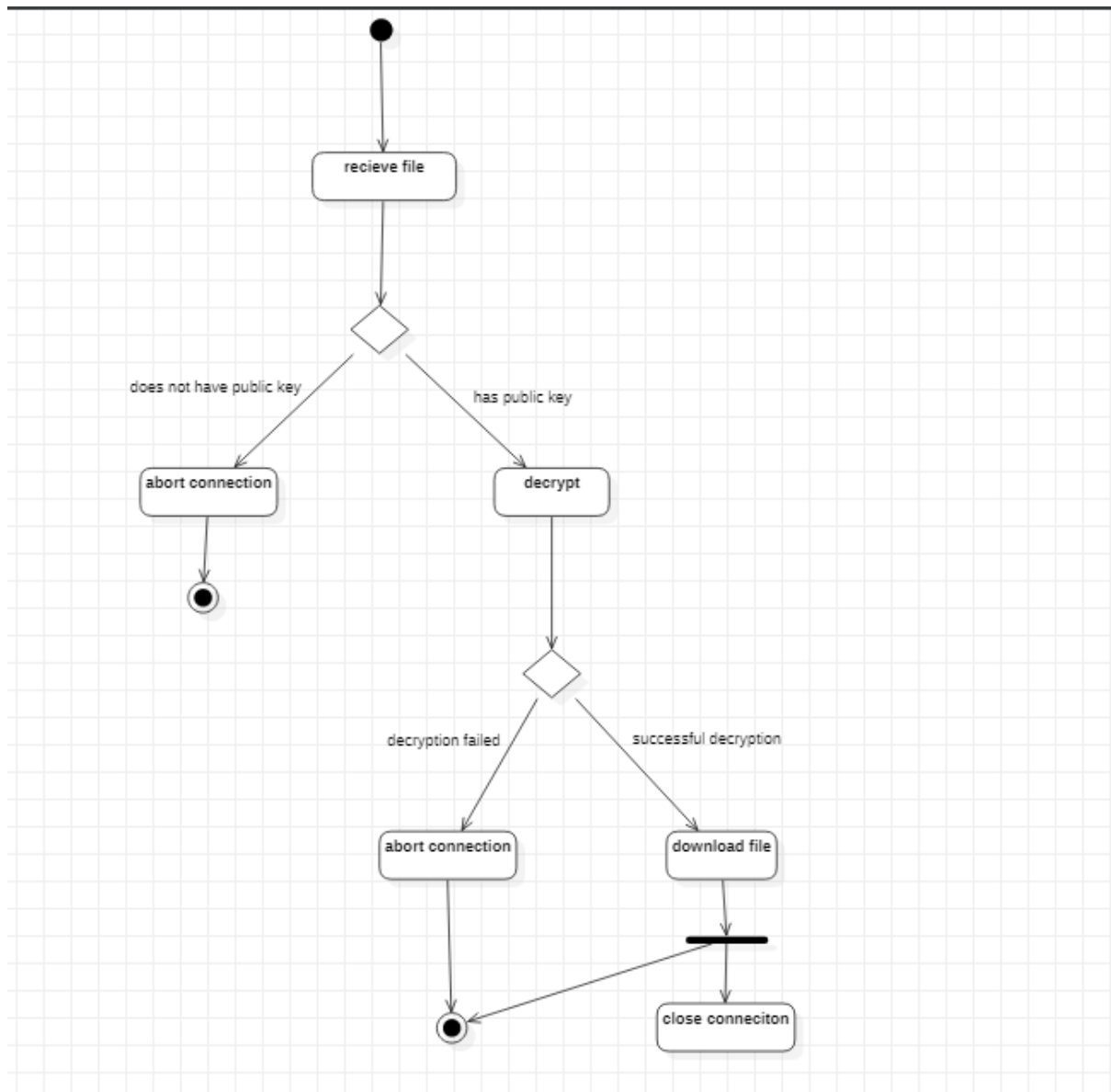
SendFile Use Case:



1. Name: SendFile
2. Summary: Prepare the file to be sent over the network

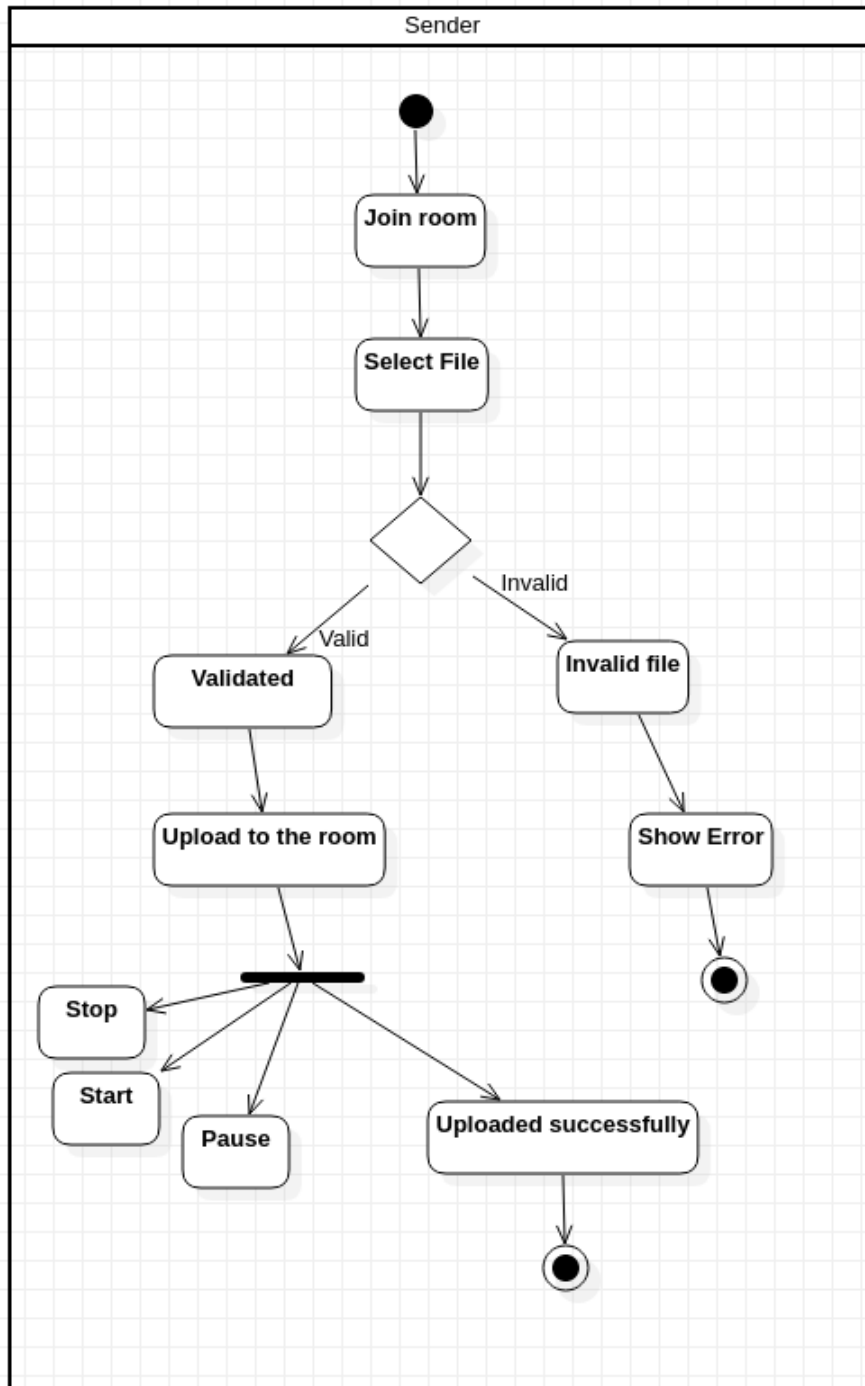
3. Actor: Sender/Peer
4. Preconditions: The file is uploaded onto the platform
5. Description:
 - On starting the connection, keys generated for secure transmission i.e encryption
 - Share the public key to the receiver
 - Encrypt the file with the key already generated
 - Send the file to the receiver
 - 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

ReceiveFile Use Case:



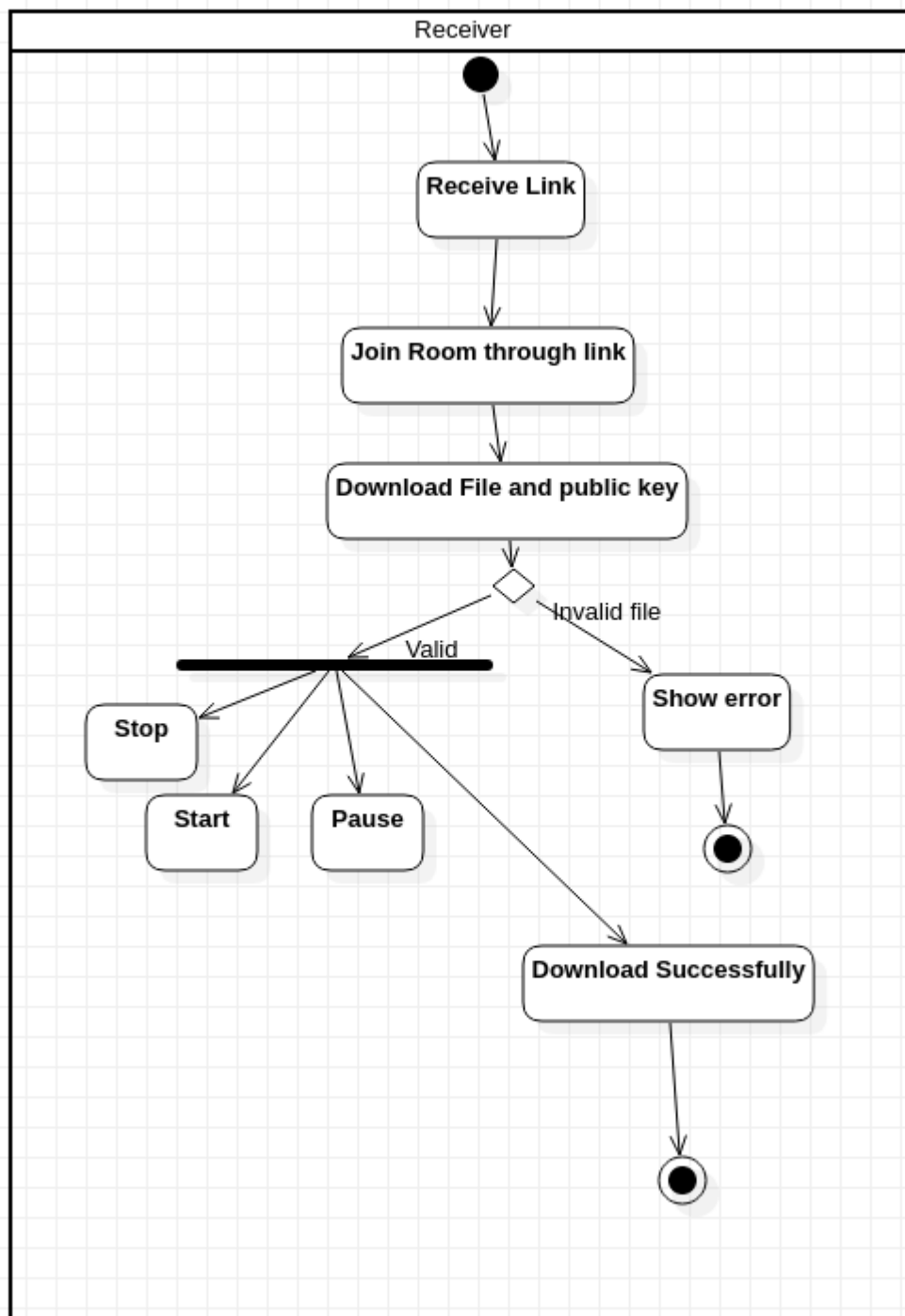
1. Name: Receive File
2. Summary: Receive the file sent over the network.
3. Actor: Peer/Receiver
4. Preconditions: The Sender has pushed a file to the fileDB
5. Description:
 - Click on “receive” file
 - Click on “verify” to verify the contents of the file by decrypting the file using the public key 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.

Upload file use case:



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.

DownloadFile use Case:



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.

Tools and Frameworks Used:

1. 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.

3. 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.

4. Database - MongoDB

- MongoDB is a NoSQL database that uses JSON-like documents with optional schemas.
- As a document database, MongoDB makes it easy for developers to store structured or unstructured data.
- In this project it is used to store the files. Each peer has a database that stores the file sent to him.

Design Principles and Design Patterns Applied:

1. 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. Eg: 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)**

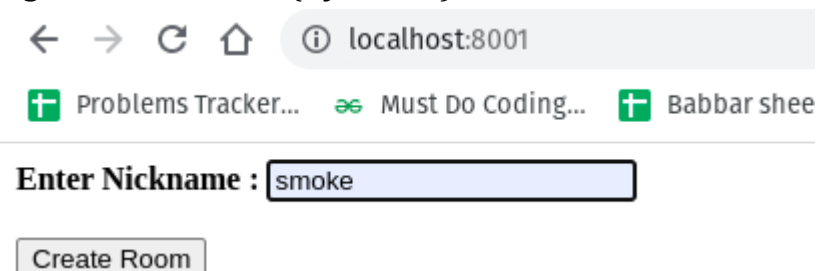
- There is no high level class that directly depends on classes at lower levels.

Design Patterns:

- **Singleton - classes that use singleton:**
 - a. SendFile
 - b. ReceiveFile
- **Factory -**
 - a. UserFactory that creates:
 - b. Peer
 - c. Client

Application Screenshots:

Fig1 - Create Room(By Client):



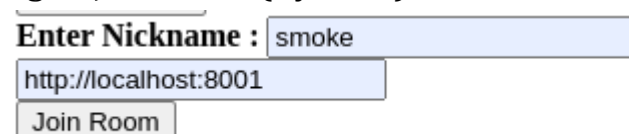
← → ↻ 🏠 ⓘ localhost:8001

📌 Problems Tracker... 🔗 Must Do Coding... 📌 Babbar shee

Enter Nickname :

Create Room

Fig2 - Join Room(By Peer):

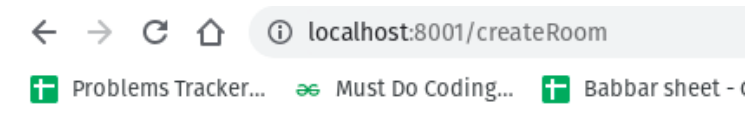


Enter Nickname :

Join Room

Fig3 - Upload File:

Fig3.1 - Before upload:



Refresh Page:

Refresh

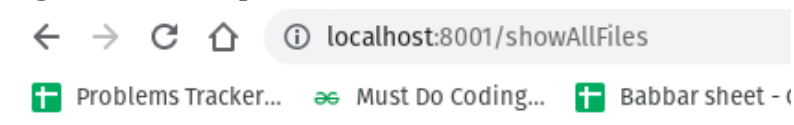
File Upload

Choose file No file chosen

Upload

No files in the database. Upload some files.

Fig3.2 - After upload:



Refresh Page:

Refresh

File Upload

Choose file No file chosen

Upload

todo.txt

delete

send

download

Fig4 - Before Sending a file:

Fig4.1 (Peer 1 - localhost:8001) [Sender - Already has 1 file]

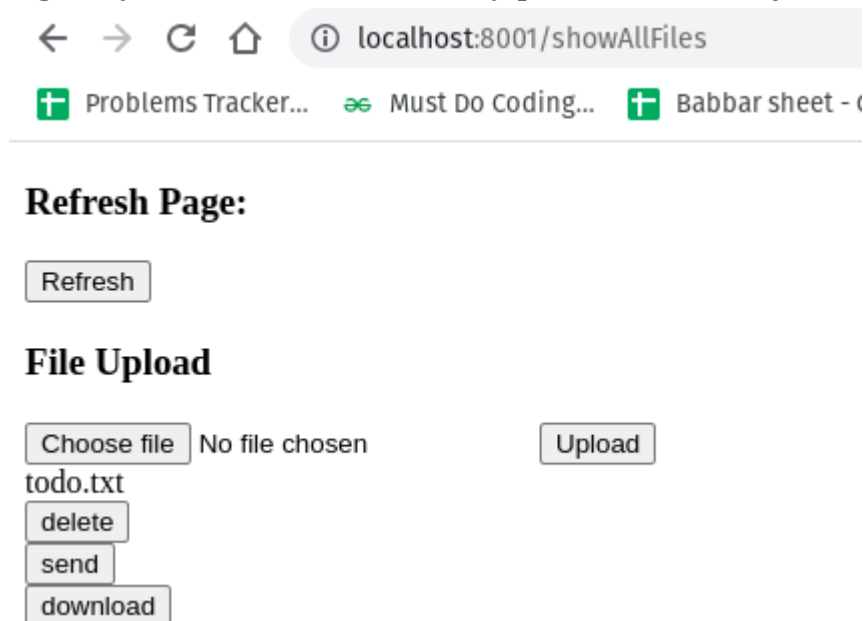


Fig4.2 (Peer2 - localhost:8002) [Receiver - has 0 files]

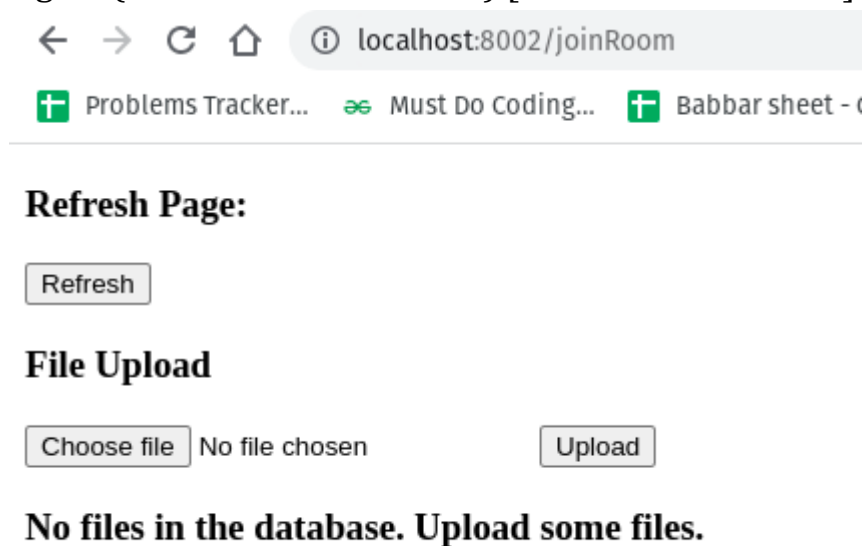


Fig4.3 (Peer3 - localhost:8003) [Receiver - has 0 files]

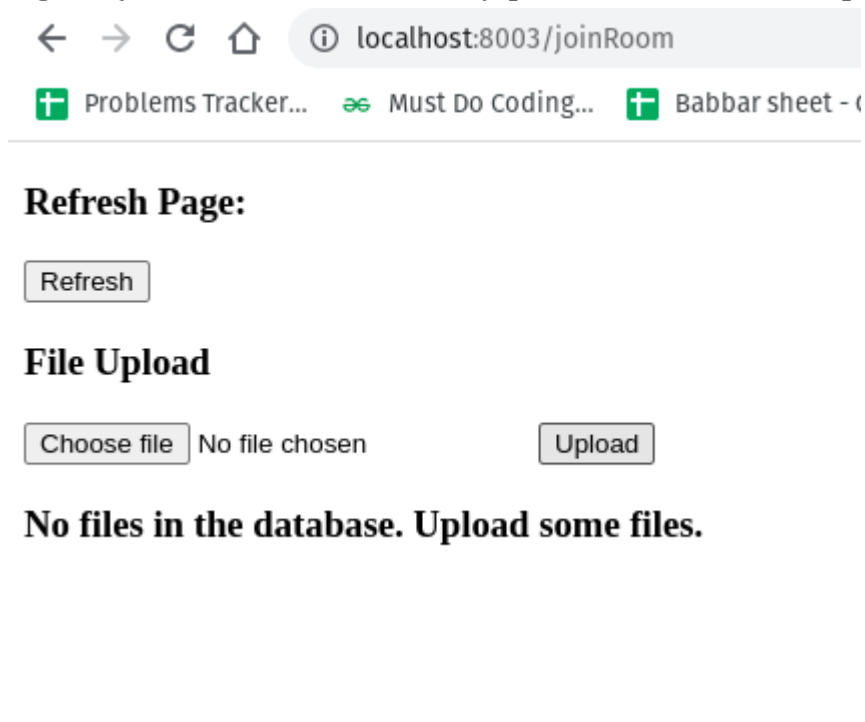


Fig5 - After Sending a file:

Fig5.1 (Peer 1 - localhost:8001) [Sender - No change]

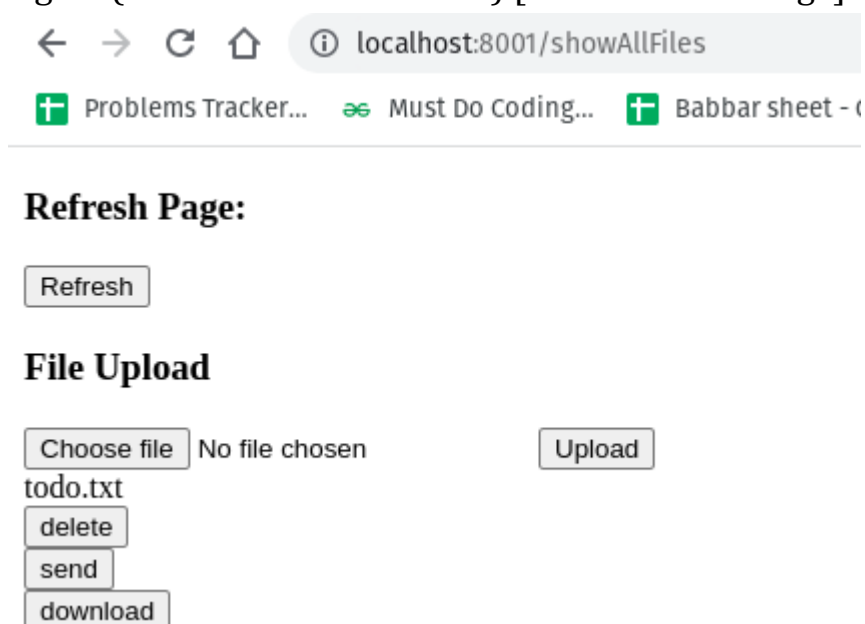


Fig5.2 (Peer2 - localhost:8002) [Receiver - gets 1 new file added, the one sent by Peer1]

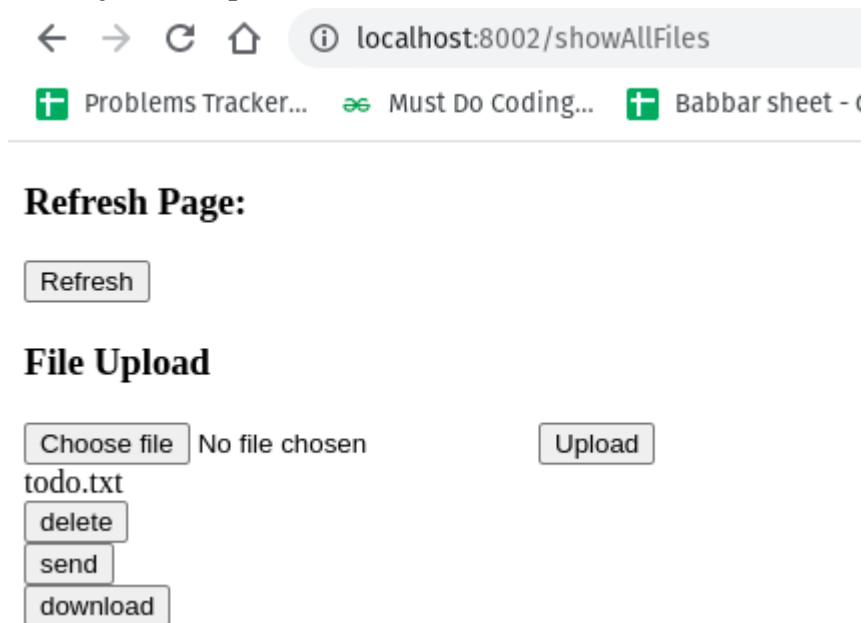


Fig5.3 (Peer3 - localhost:8003) [Receiver - gets 1 new file added, the one sent by Peer1]

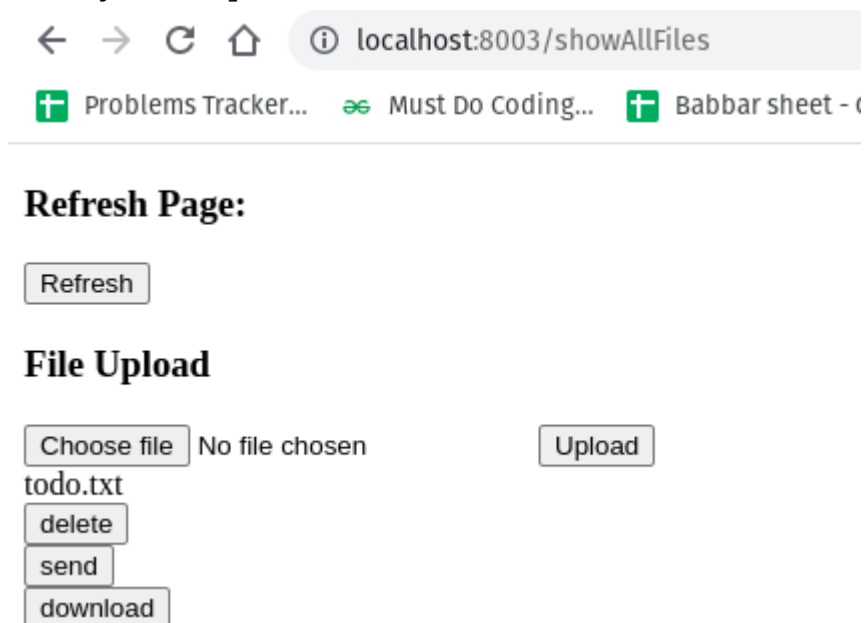


Fig6 - Download File:

←

→

↻

🏠

📄 localhost:8002/showAllFiles

📄 Problems Tracker...

🔗 Must Do Coding...

📄 Babbar sheet - G...

Refresh Page:

Refresh

File Upload

Choose file

No file chosen

Upload

todo.txt

delete

send

download

📄 todo.txt

^

Fig7 - Delete File:

Fig7.1: Before Delete:

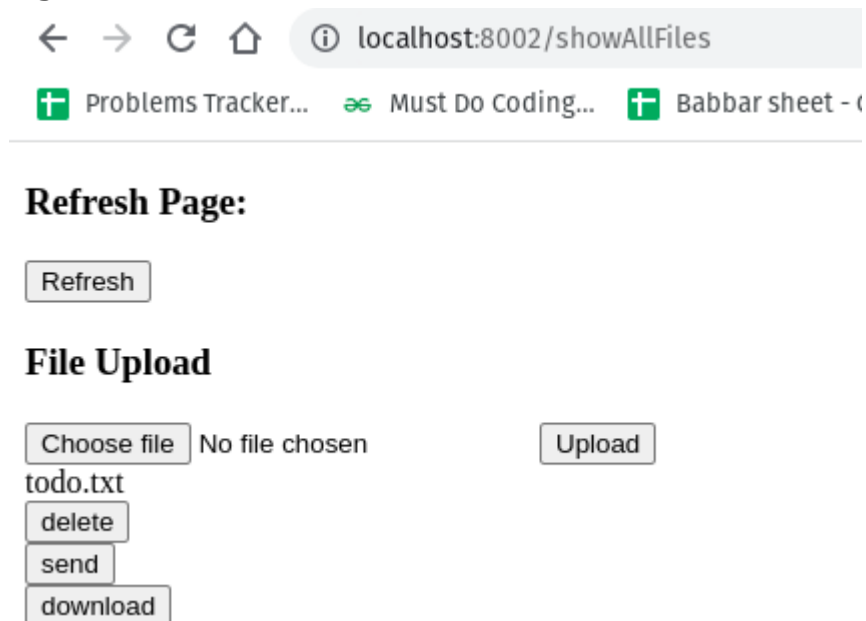
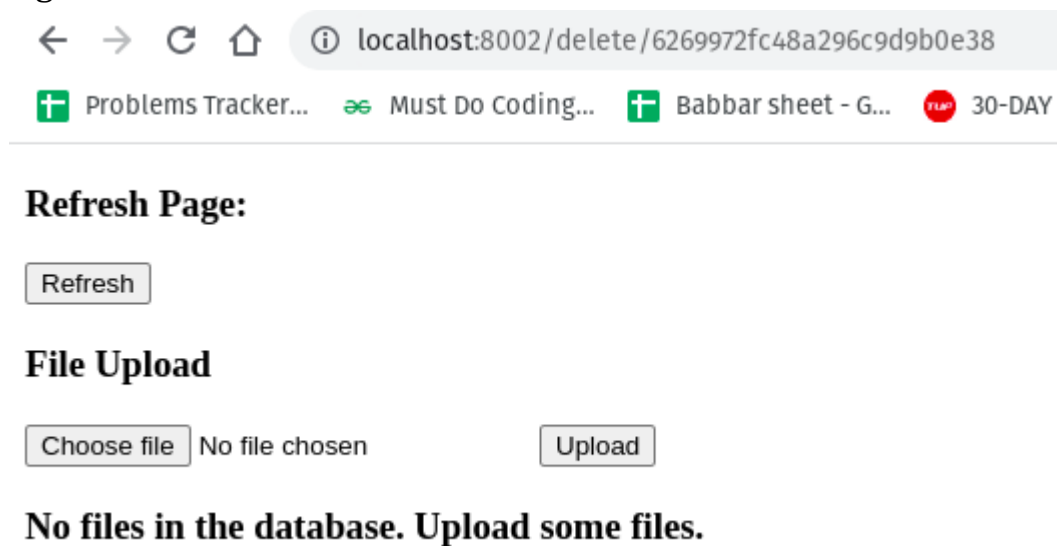


Fig7.2: After Delete:



Team Contributions: (Classes/Useases Implemented)

Sumukh Raju Bhat (CS519)	Room, SendFile, ReceiveFile, FileController.
Sowmya M(CS497)	Crypto, DBService, myFile, CustomMultiPartiteFile.
Shreyas VBKS(CS469)	UserFactory, Peer, Client, FileController.

