PES University, Bengaluru

(Established under Karnataka Act 16 of 2013)

Mini project report on

Secure File Sharing System

Submitted in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE20CS352 –OOADJ Project

Submitted by:

SRN	NAME	SEC	Roll No
PES1UG20CS825	Prem Sagar J S	“H”	61
PES1UG20CS472	Tushar J	“H”	10
PES1UG20CS479	Vaibhav Vijay	“H”	25
PES1UG20CS808	Bhavana N G	“C”	61

Under the guidance of

Prof. Priya Badrinath

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

TABLE OF CONTENTS

NOTE: Please add appropriate description for all diagrams where ever required. Only important class implementation needs to be added to IMPLEMNTATION SECTION.

INTRODUCTION

A safe file sharing system is an essential tool in today's communication and collaboration. It enables individuals and organisations to securely exchange files and information over the internet while safeguarding sensitive data from unauthorised access, modification, or theft.

The primary goal of a secure file sharing system is to ensure the shared data's confidentiality, integrity, and availability. The protection of data from unauthorised access is referred to as confidentiality. The term "integrity" refers to the fact that the data remains unchanged during transmission or storage. The availability of data means that authorised users can access it when they need it.

A secure file sharing system achieves these goals by combining encryption, access controls, authentication, and other security mechanisms. Encryption ensures that the data is unreadable and unusable to anyone who does not have the correct key. Access controls limit authorised users' access and prevent unauthorised access. The authentication process ensures that only authorised users have access to the data.

Project Description

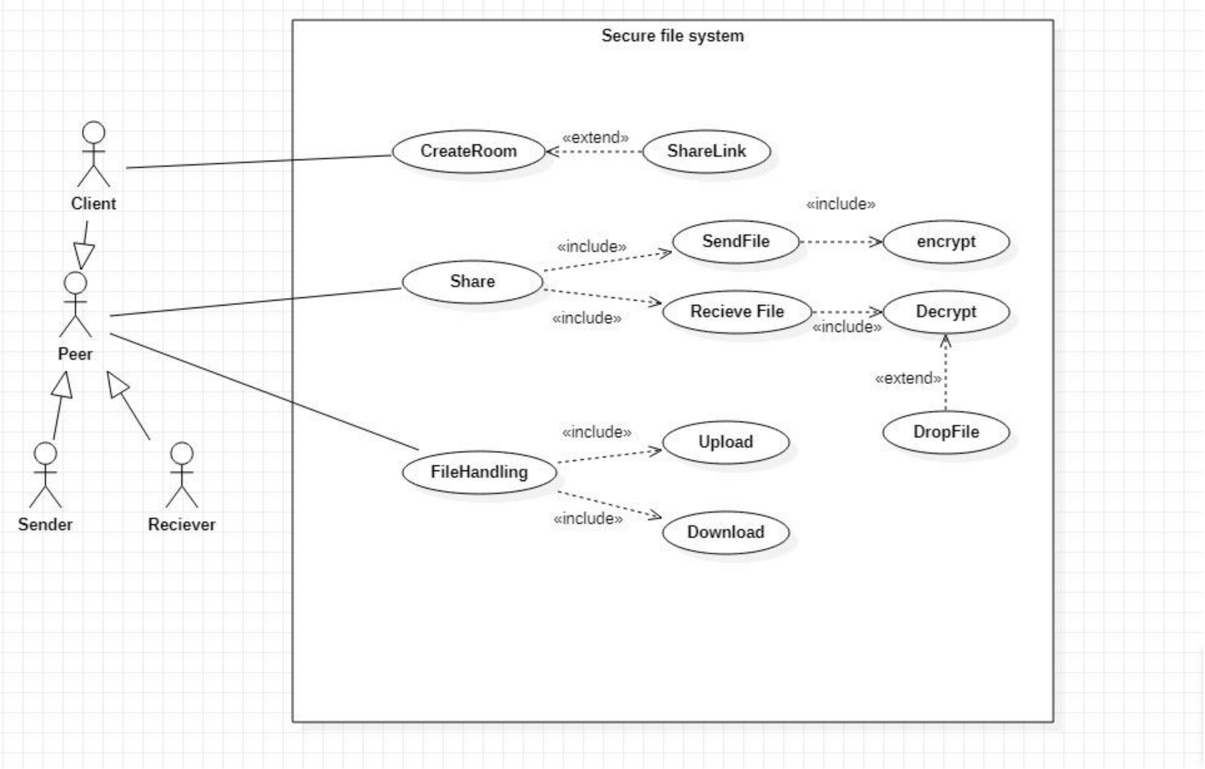
The aim of the project is to build an end-to-end secure file sharing system which mimics the P2P architecture. Files are shared via a room which is created by the client. The invite link of the sharing platform which in this case is a room is created for the peers who want to access and download files. 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.

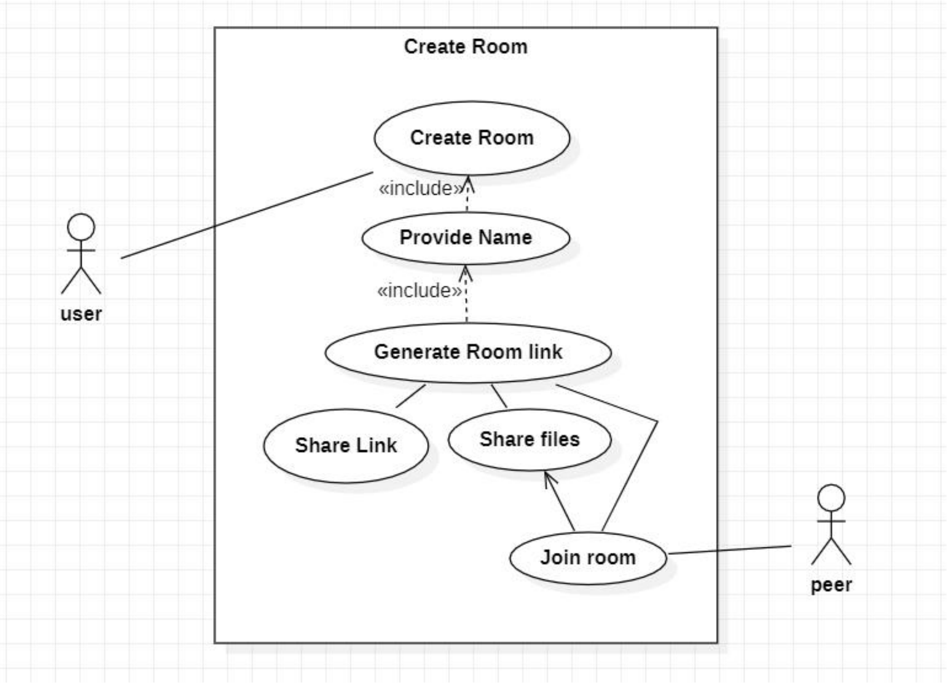
The encrypted files are decrypted on the receiving end using the secret key available in the room to validate the file. After the decryption the file is downloaded on the receiving end. In order to account for the file not being corrupted, the concept of checksum is used to verify the integrity of the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

Analysis and Design Models

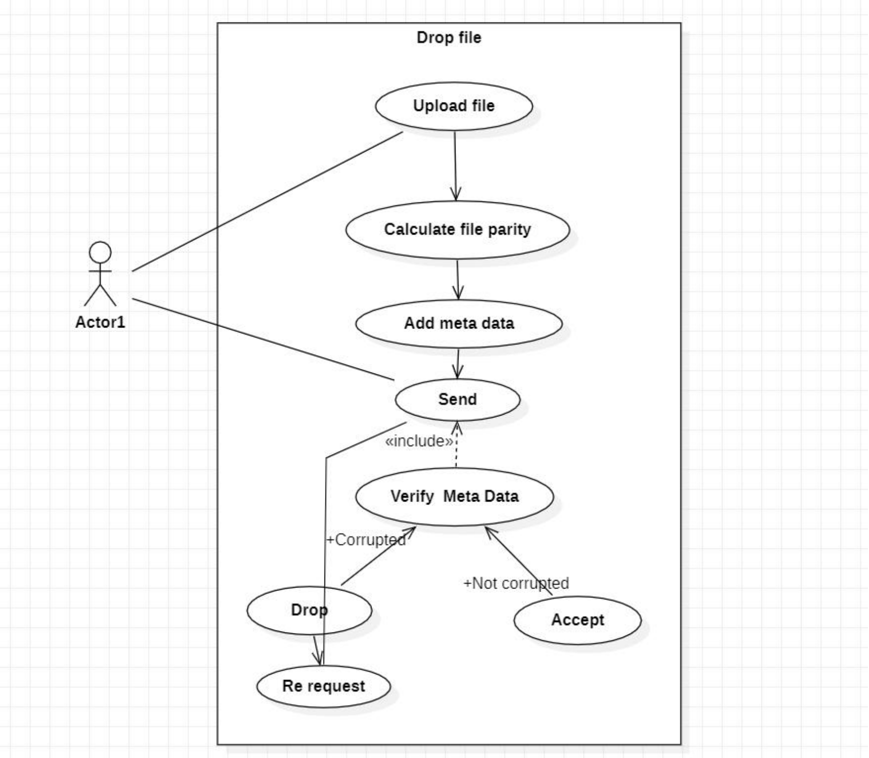
1. Use Case Diagram:



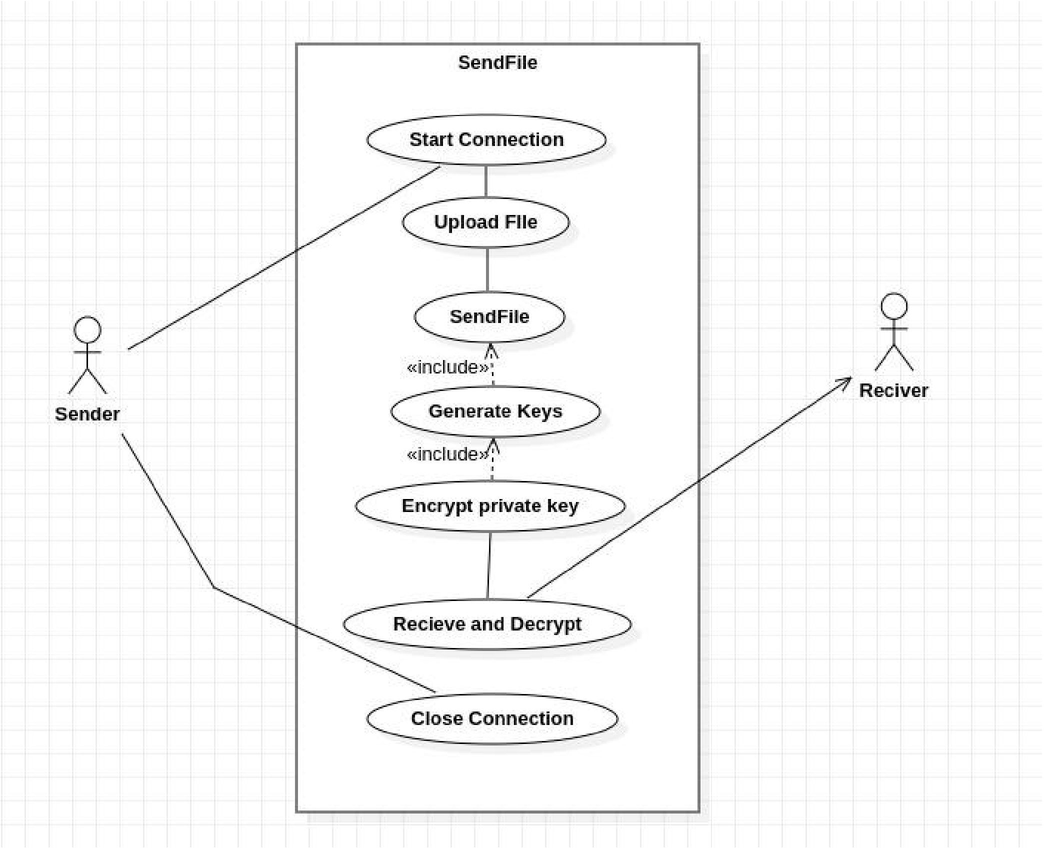
(a) Create Room Use Case:



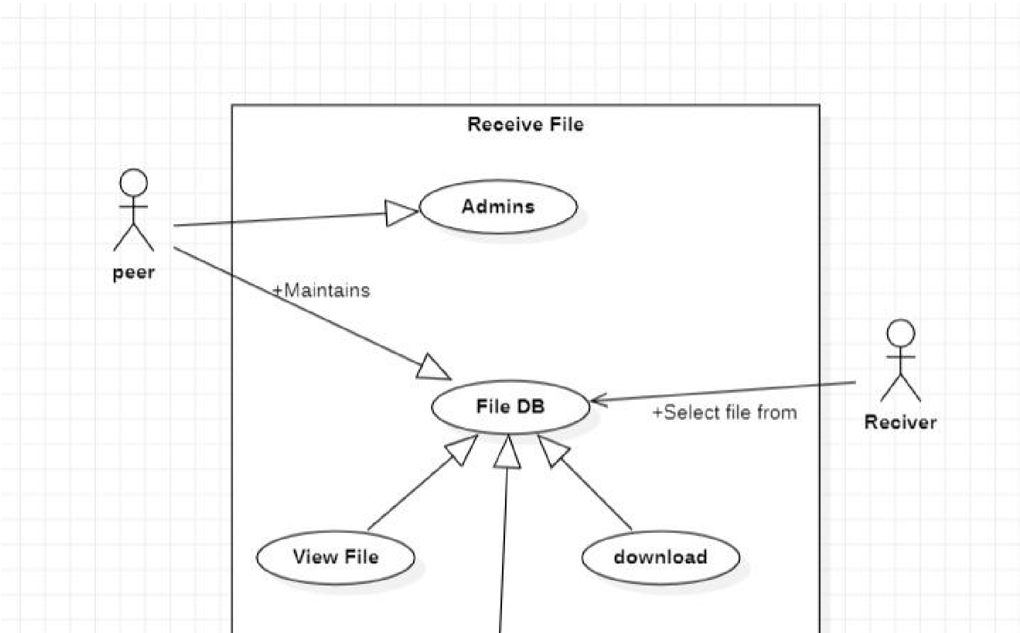
(b) Drop File Use Case:



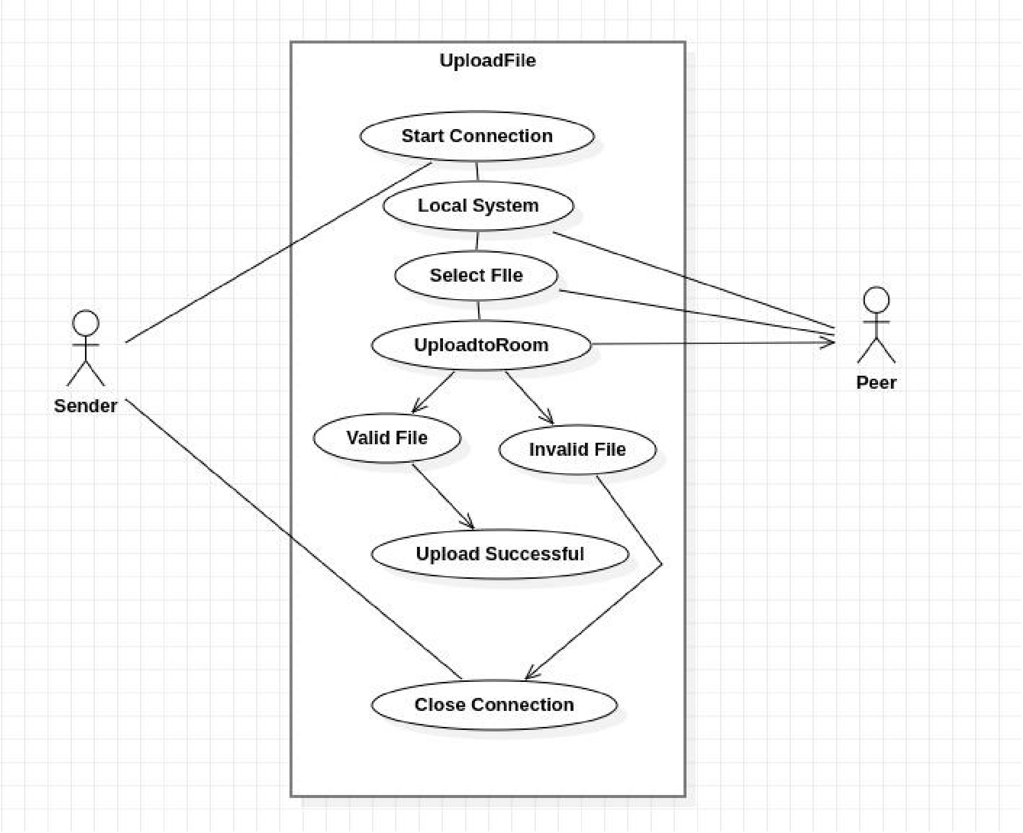
(c) Send File Use Case:



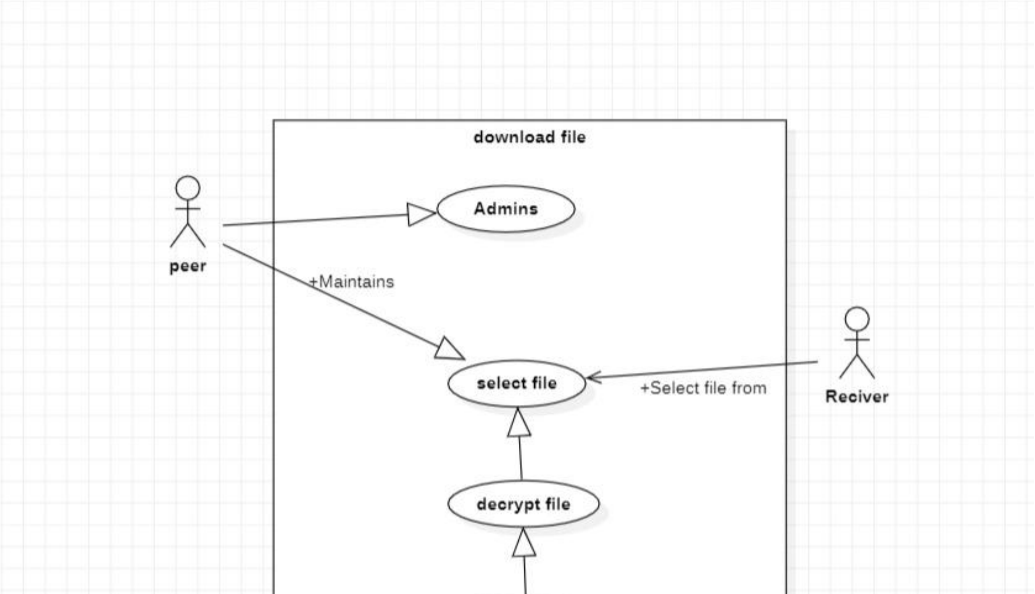
(d) Receive File Use Case:



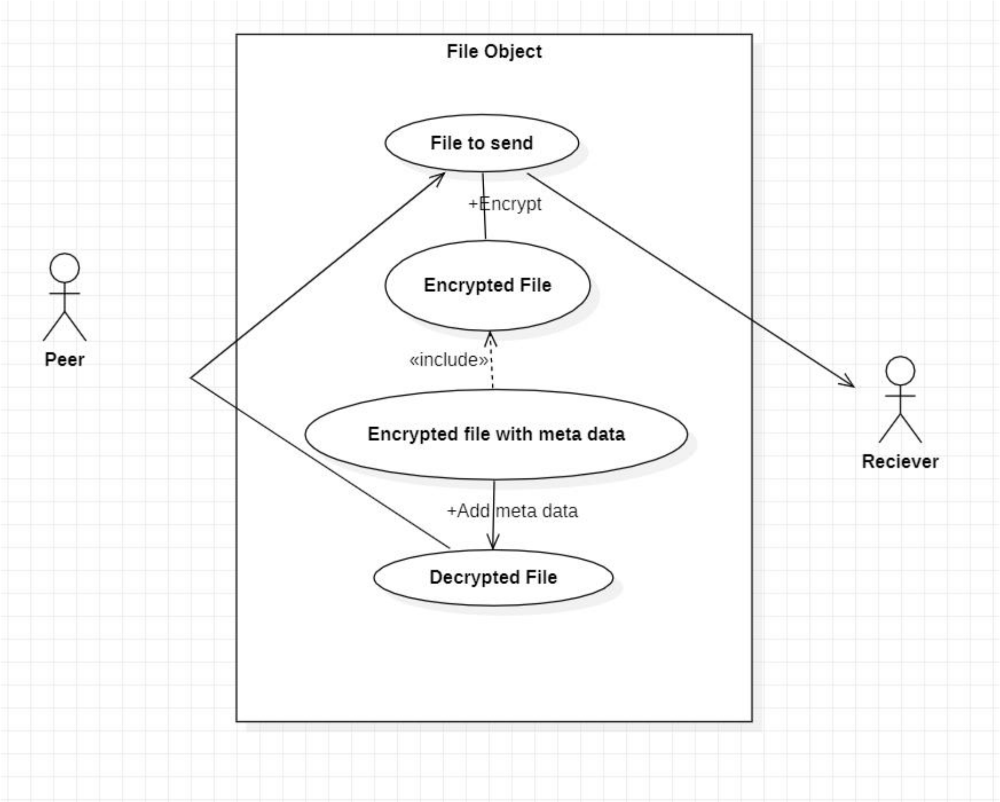
(e) Upload File Use Case:



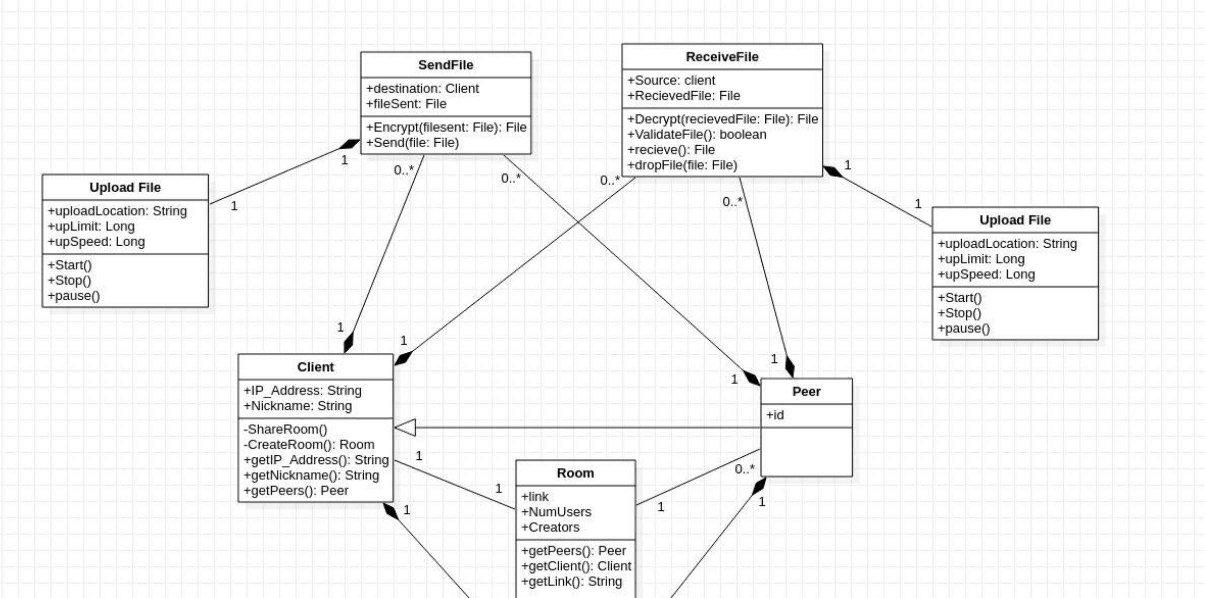
(f) Download File Use Case:



(g) File Object

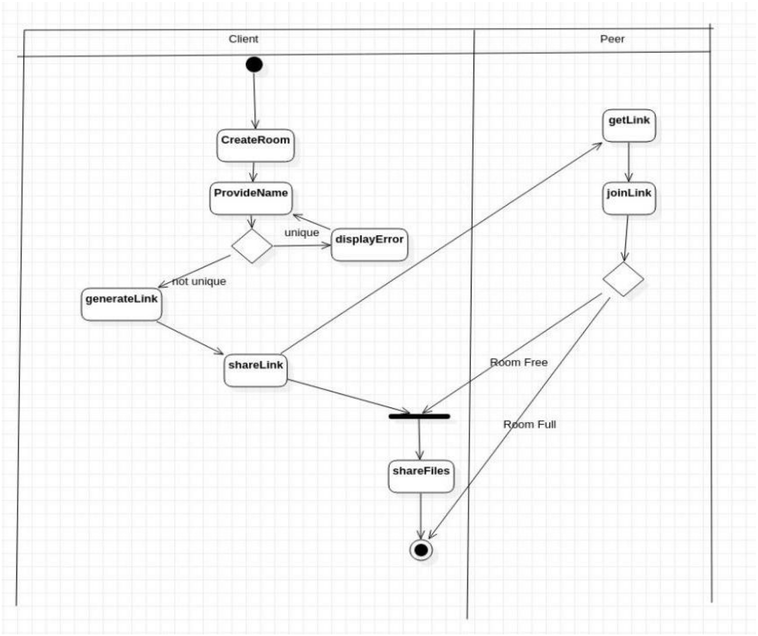


2. Class Modelling diagram:



3.Activity diagram for the use-cases implemented

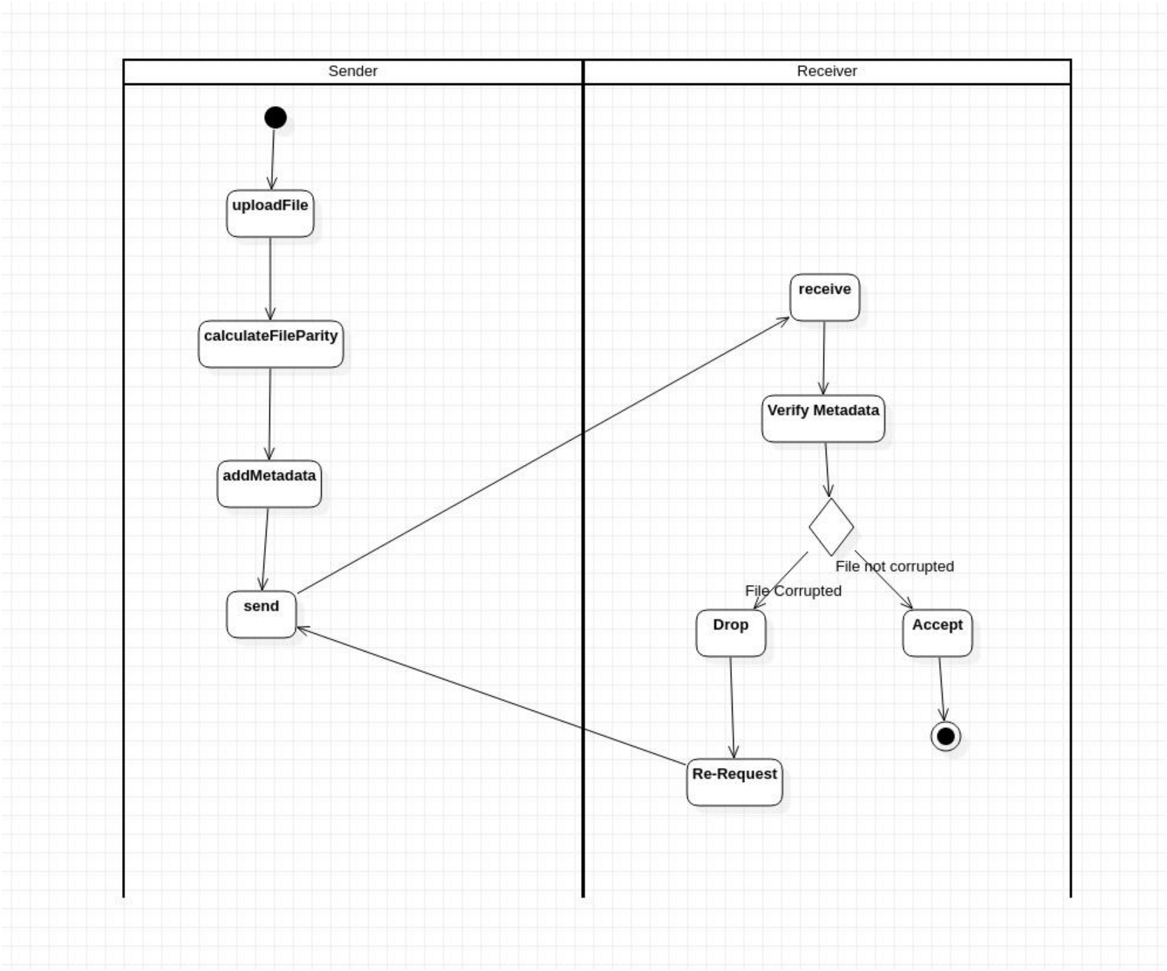
(a) Create Room Use Case:



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)Drop File Use Case



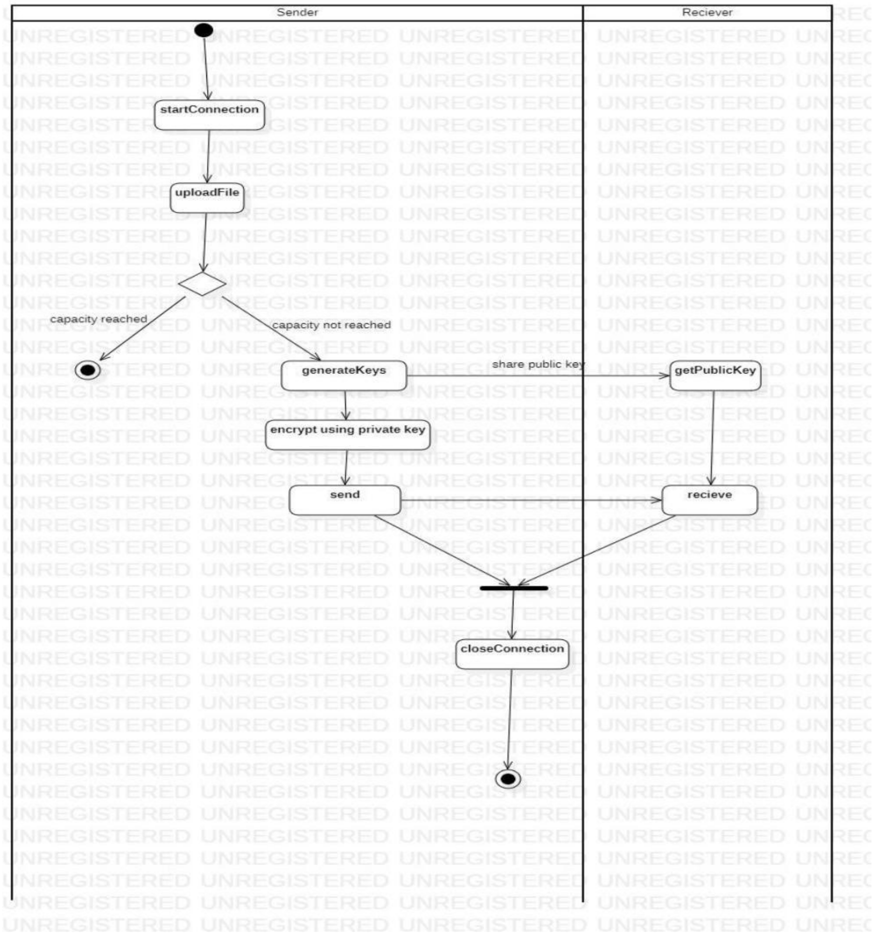
1. Name: DropFile
2. Summary: On receiving the file, drop the file if it was corrupted during transfer.
3. Actor: Receiver/Peer
4. Preconditions: File is uploaded and is ready to be sent over the network

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

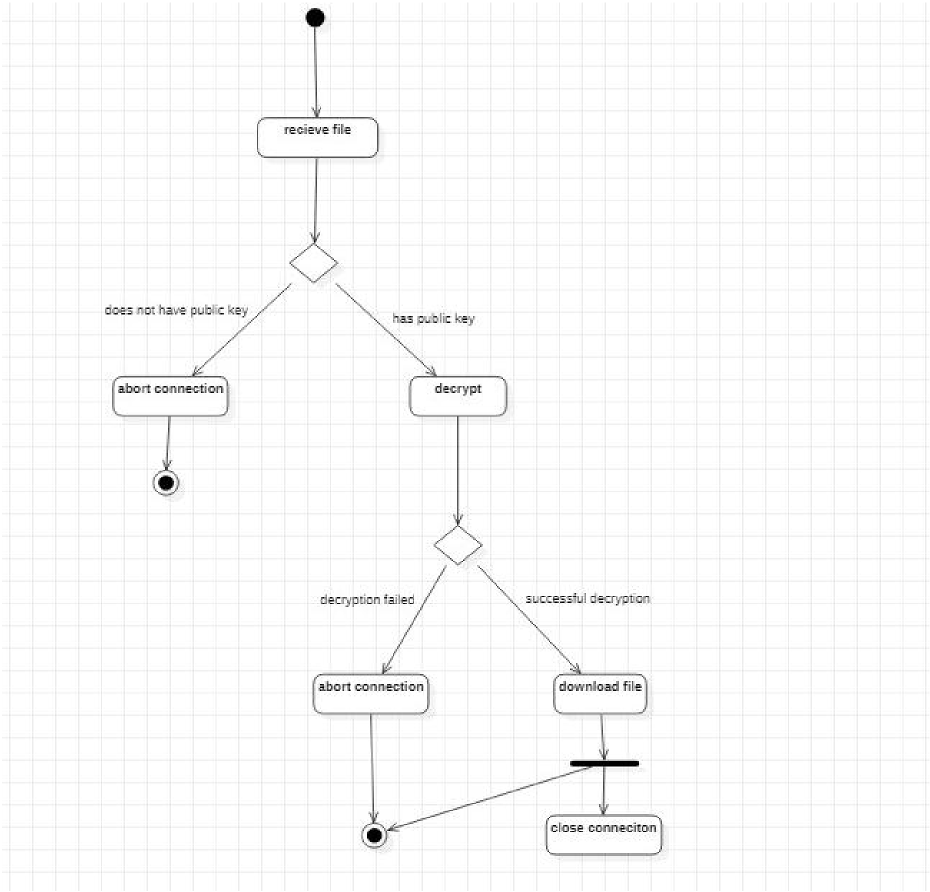
Send File Use Case:



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

Post-condition: Receiver can receive the sent file

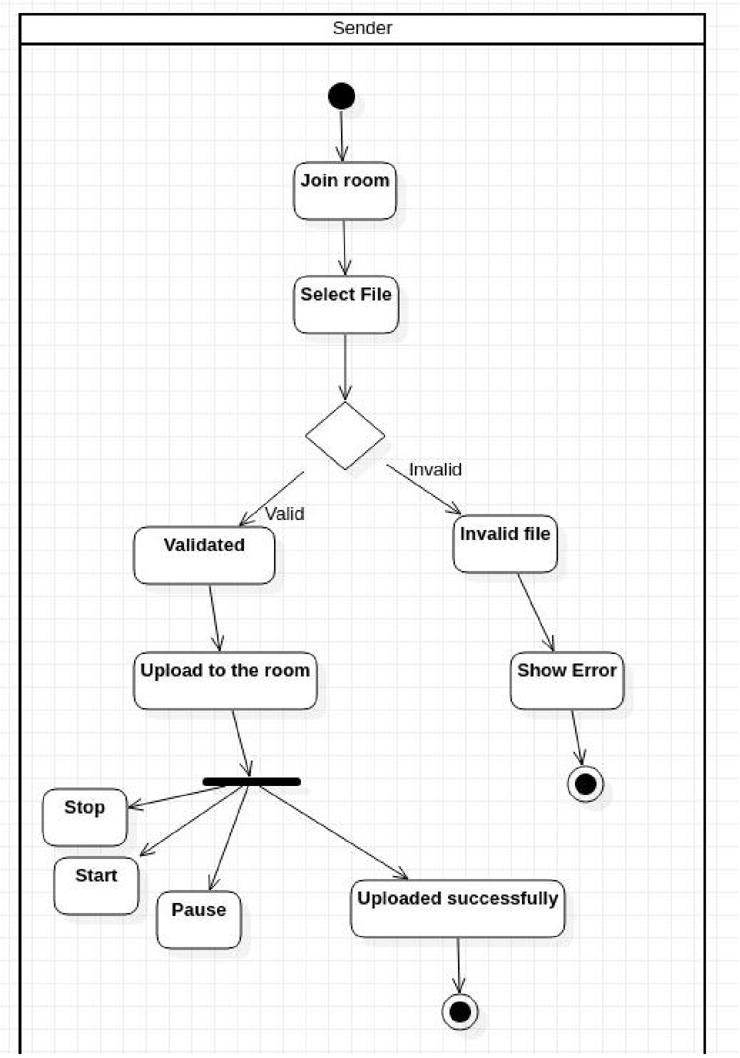
Receive File Use Case:



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.

Post-condition: The file can now be downloaded by the receiver.

Upload file use case:



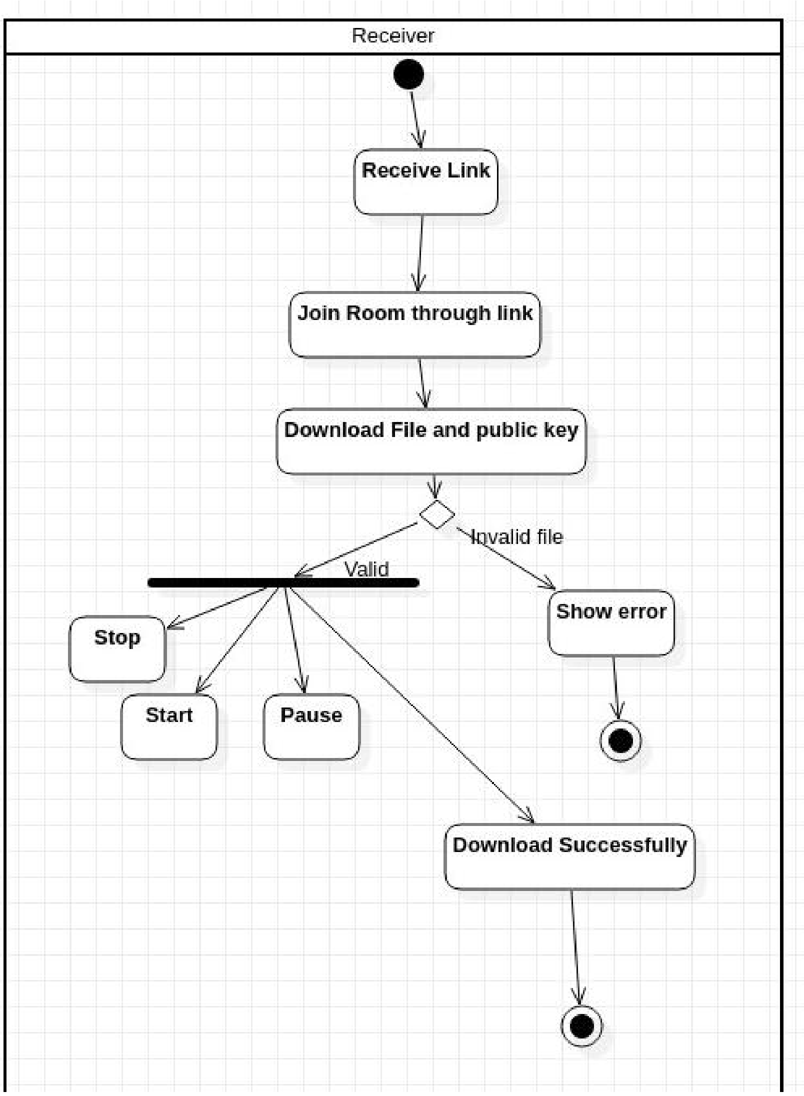
1. Name: Upload File use case
2. Summary: Select the file and upload the file in the FileDB
3. Actor: Client/Peer
4. Pre-conditions: The Client has entered the platform
5. Description:

- 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.

   Post-condition - Send the file for encryption.

Download File use Case:

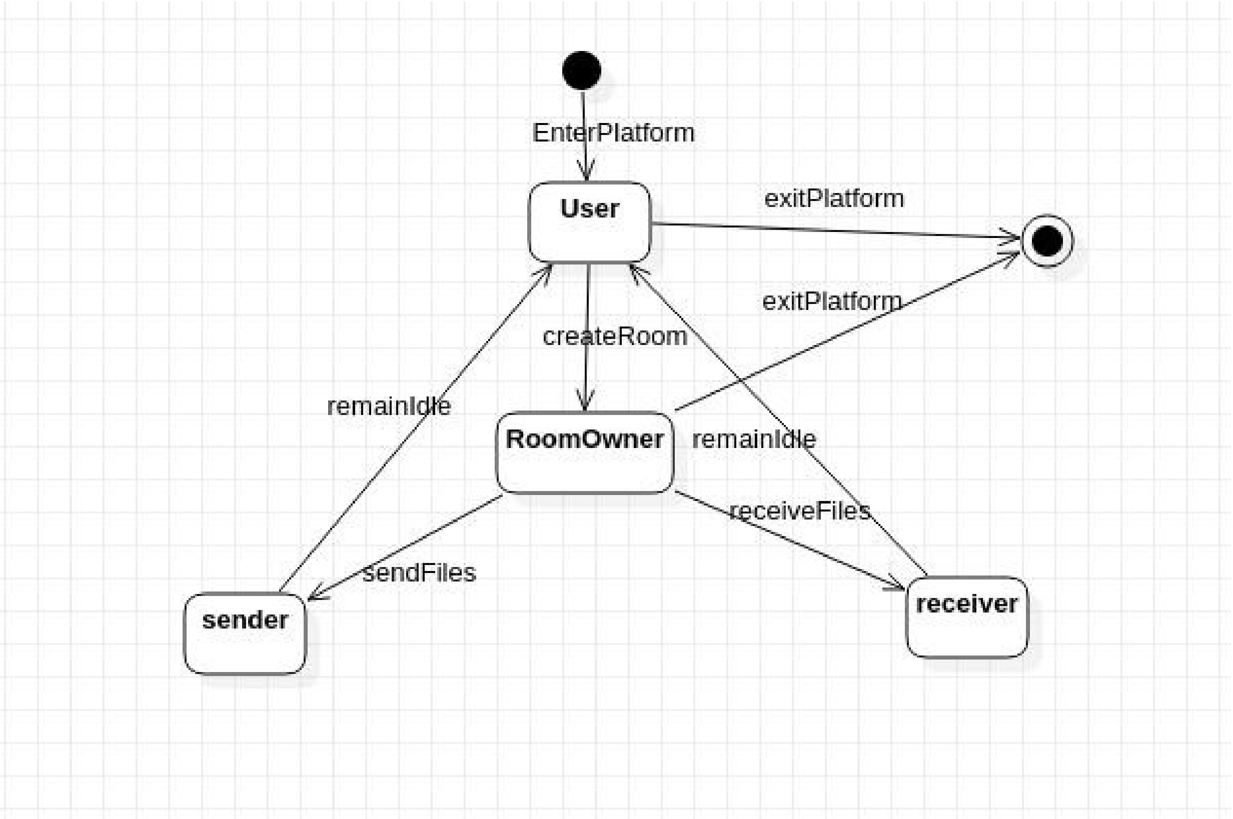


1. Name: Download File use case
2. Summary: Download the shared file in the room
3. Actor: Receiver/Peer
4. Pre-conditions: The file had been received and verified by decryption.
5. Description:

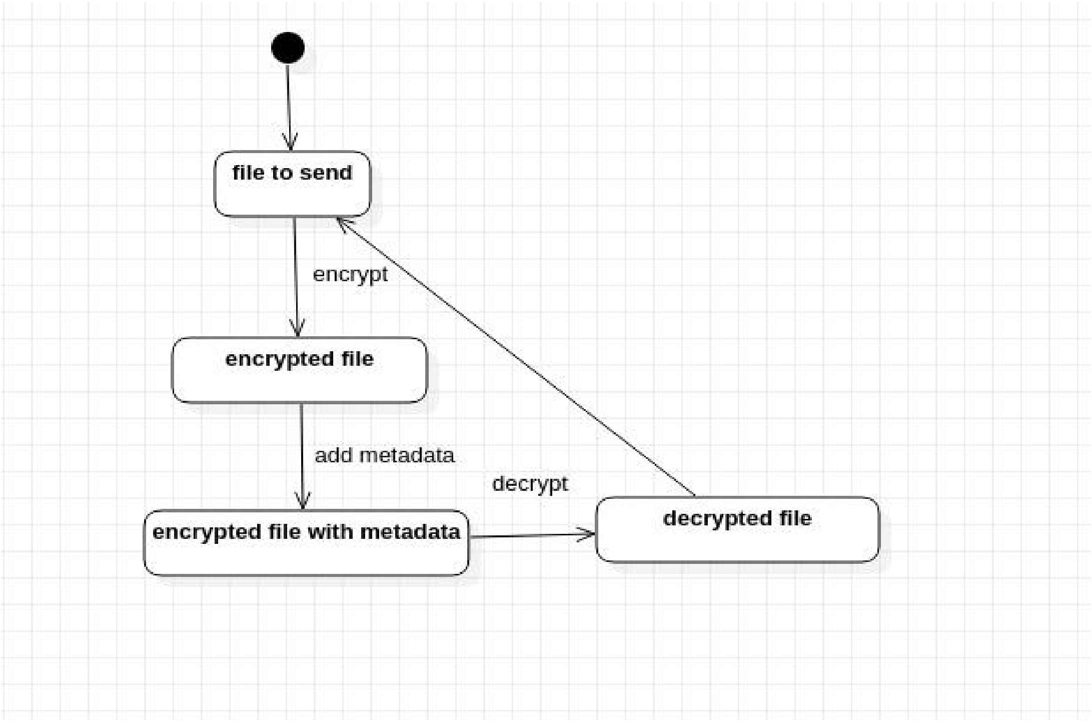
- Click on “download” to download the file.
- Select the destination to download in the peer’s local system.
- Click on “OK” to start downloading.
- The downloading actions have start, pause and stop options.

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.

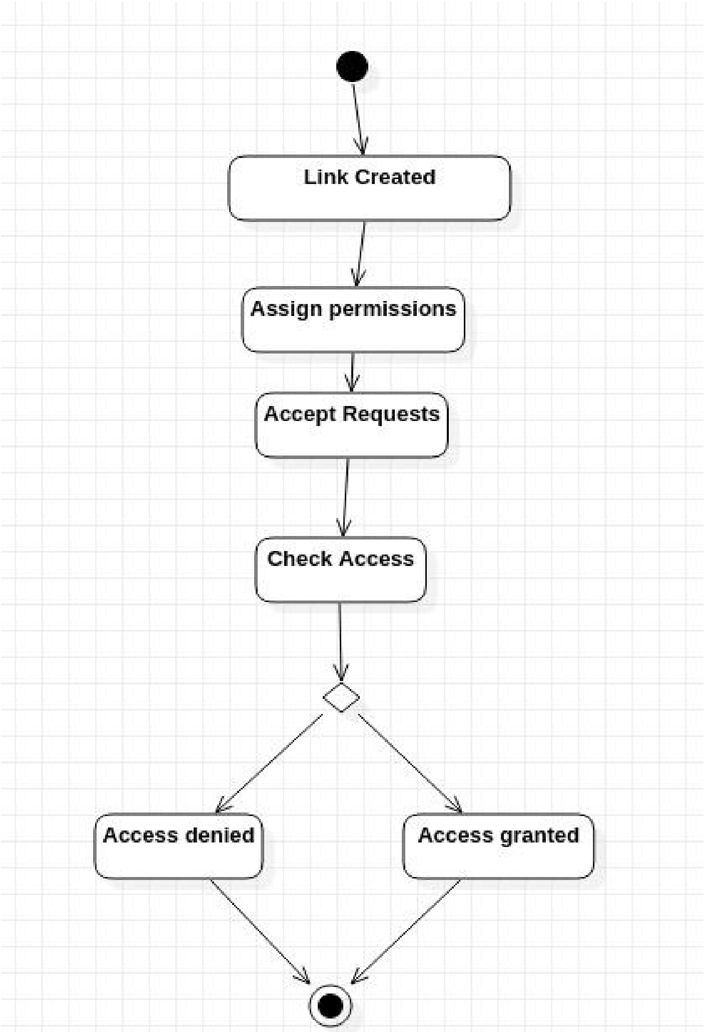
State diagram for the classes with temporal behaviour Client Object



File Object



Link accessibility



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 send File and receive File 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 Custom Multipart File implements Multipart File 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:

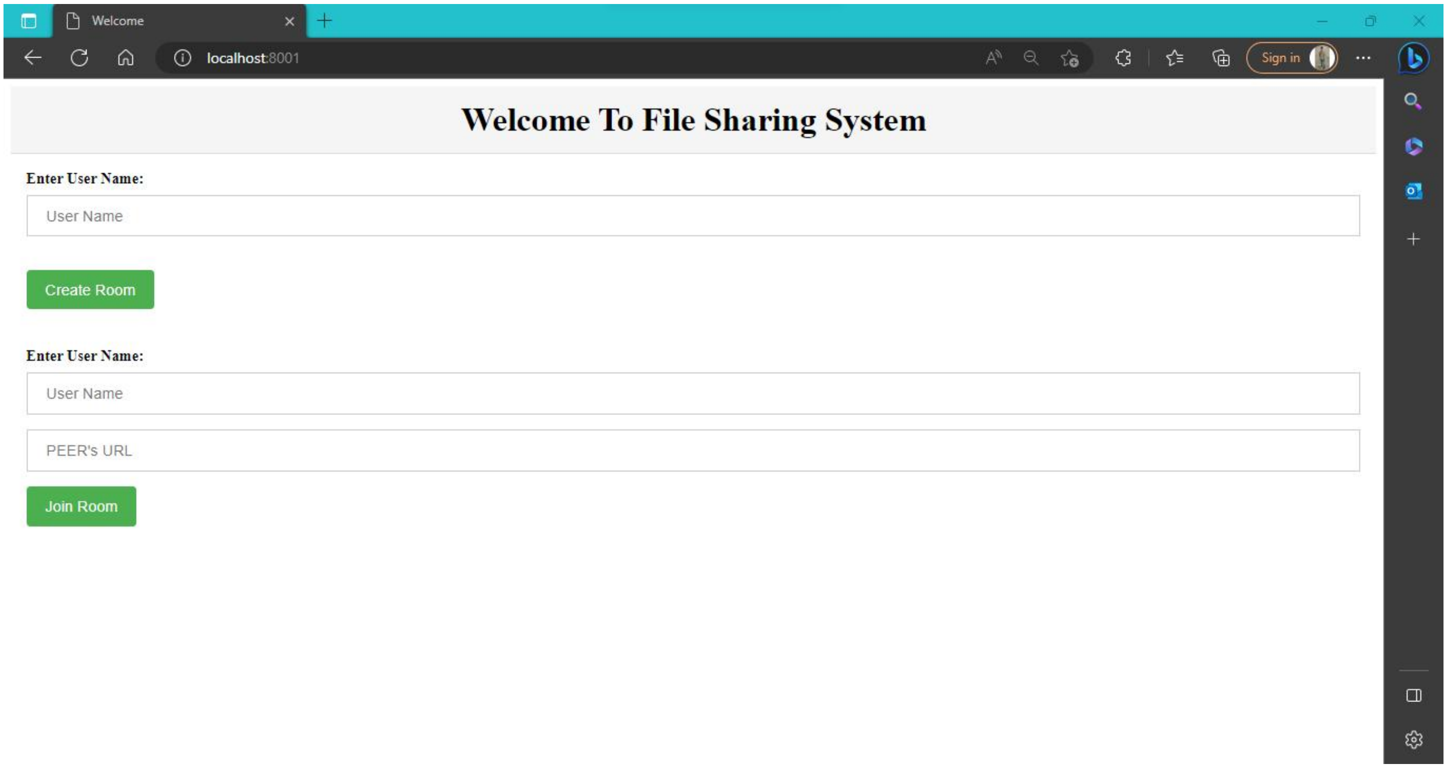
- **Singleton - classes that use singleton:**
    a. SendFile
    b. ReceiveFile
- **Factory -**
    a. UserFactory that creates:
    b. Peer
    c. Client

GITHUB REPOSITORY DETAILS

https://github.com/premXsagarXsonu/Secure-File-Sharing-Sytem-OOADJ.git

SCREENSHOTS OF APPLICATION

Index Page :



Welcome To File Sharing System

Enter User Name:

User Name

Create Room

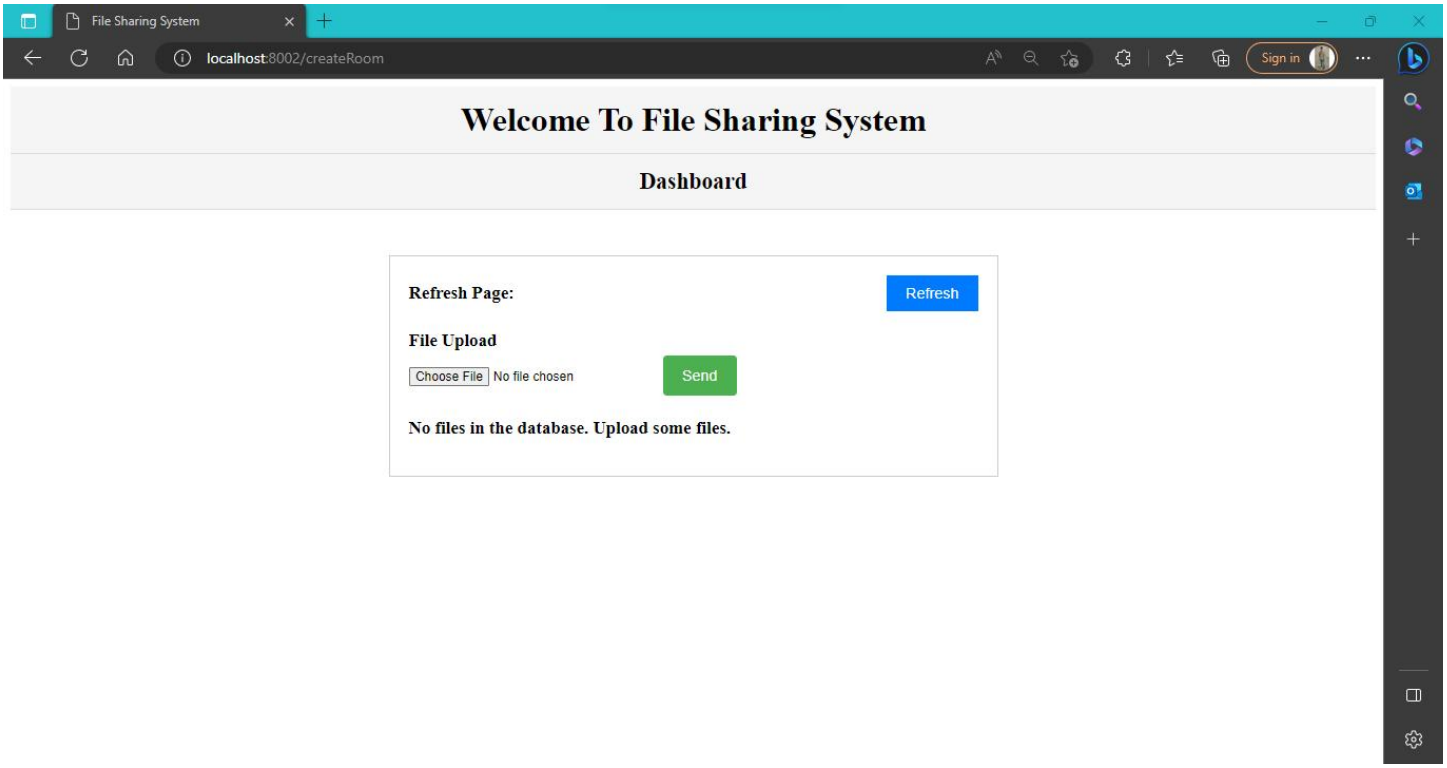
Enter User Name:

User Name

PEER's URL

Join Room

Dashboard Page:



Welcome To File Sharing System

Dashboard

Refresh Page: Refresh

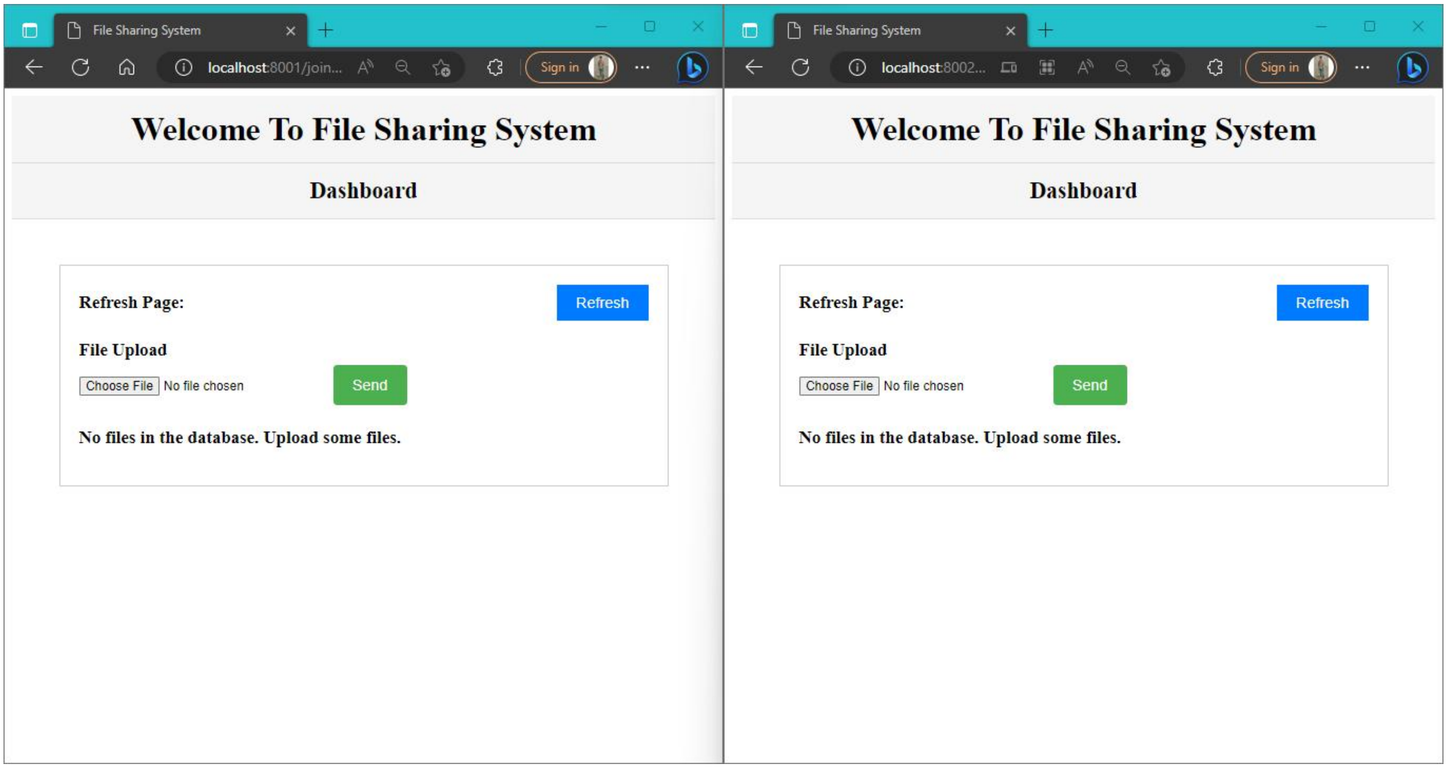
File Upload

Choose File No file chosen Send

No files in the database. Upload some files.

Dashboard Page :

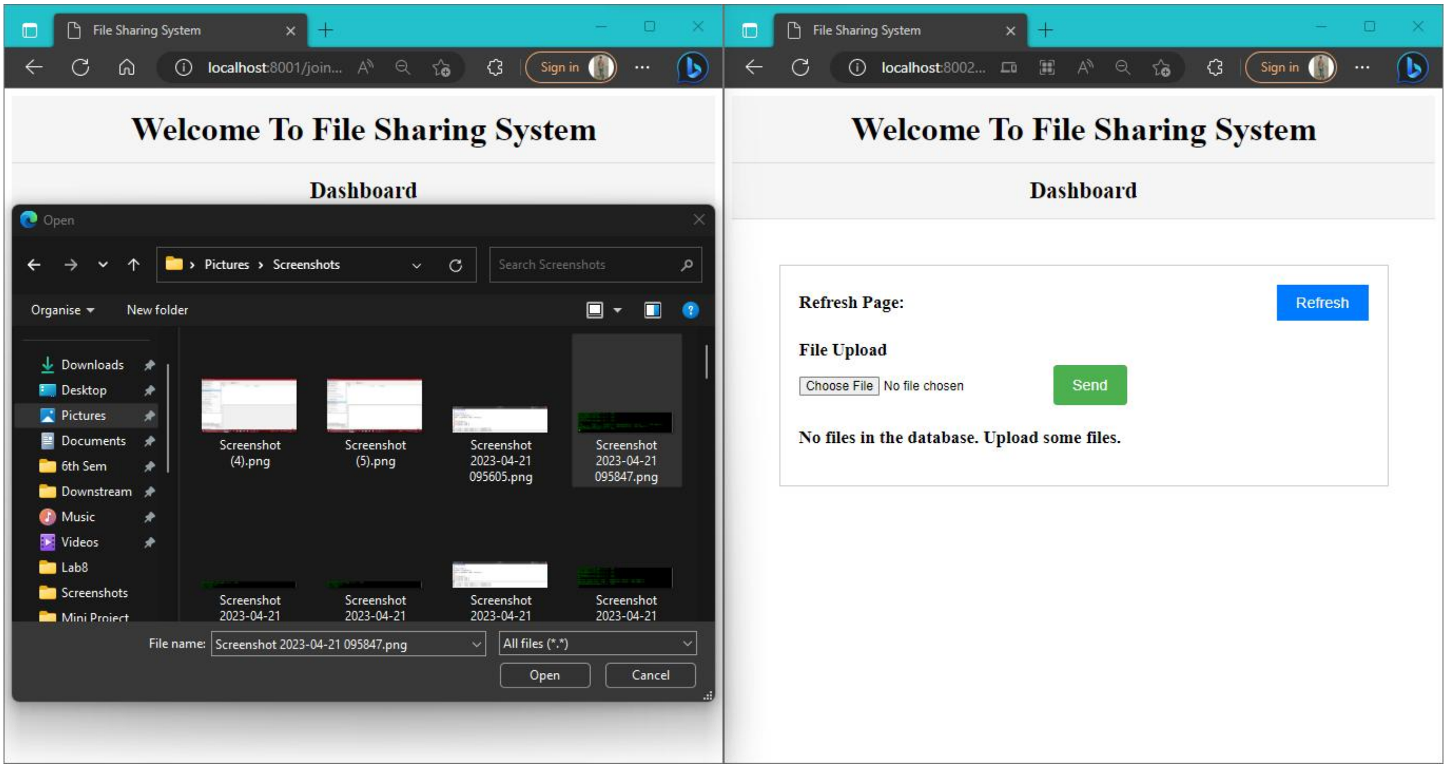
(dashboard page with two peers with different port number 8001 and 8002)



No files in the database. Upload some files.

No files in the database. Upload some files.

Peer1 selecting file to send :

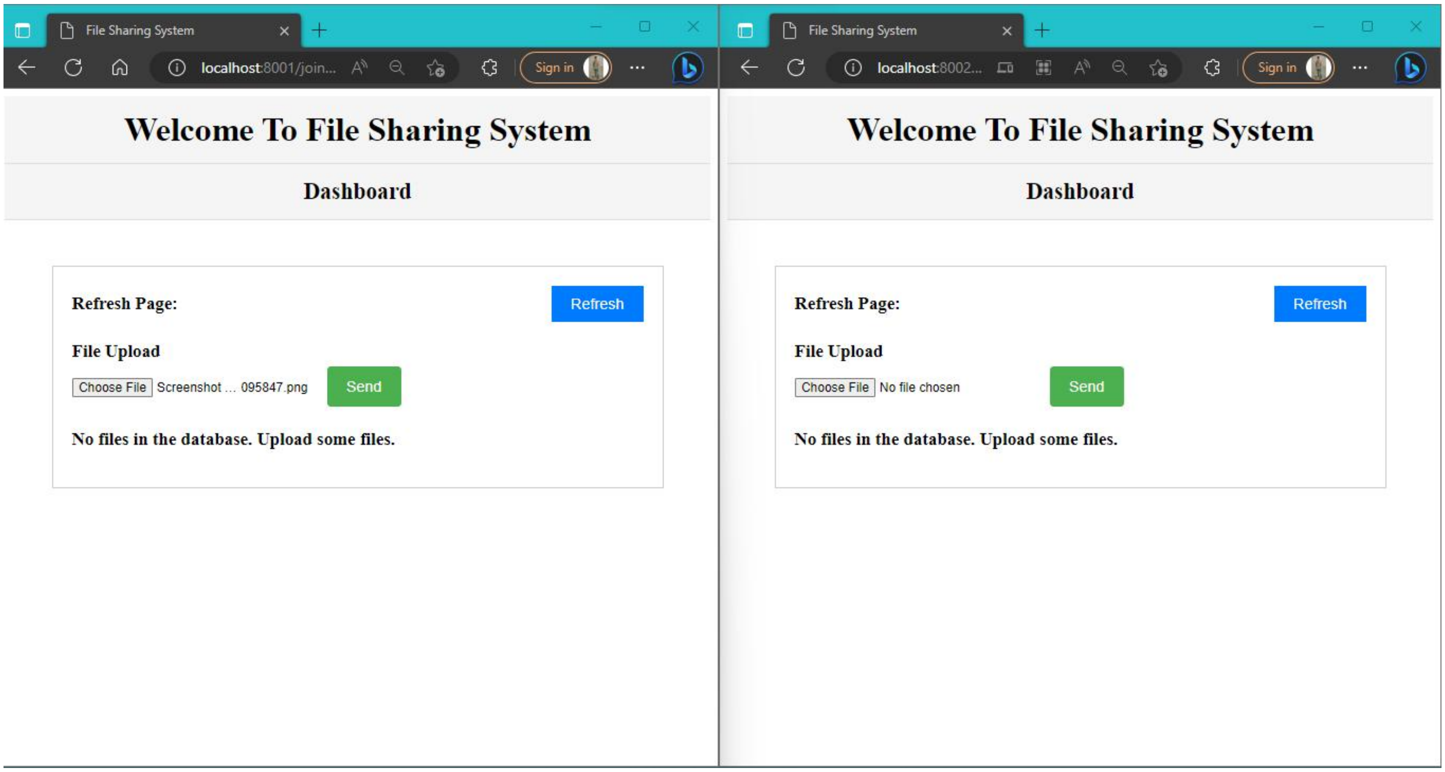


File name: Screenshot 2023-04-21 095847.png

All files (*.*)

No files in the database. Upload some files.

Peer1 selected file to send :



Welcome To File Sharing System

Dashboard

Refresh Page:

File Upload

Screenshot ... 095847.png

No files in the database. Upload some files.

Welcome To File Sharing System

Dashboard

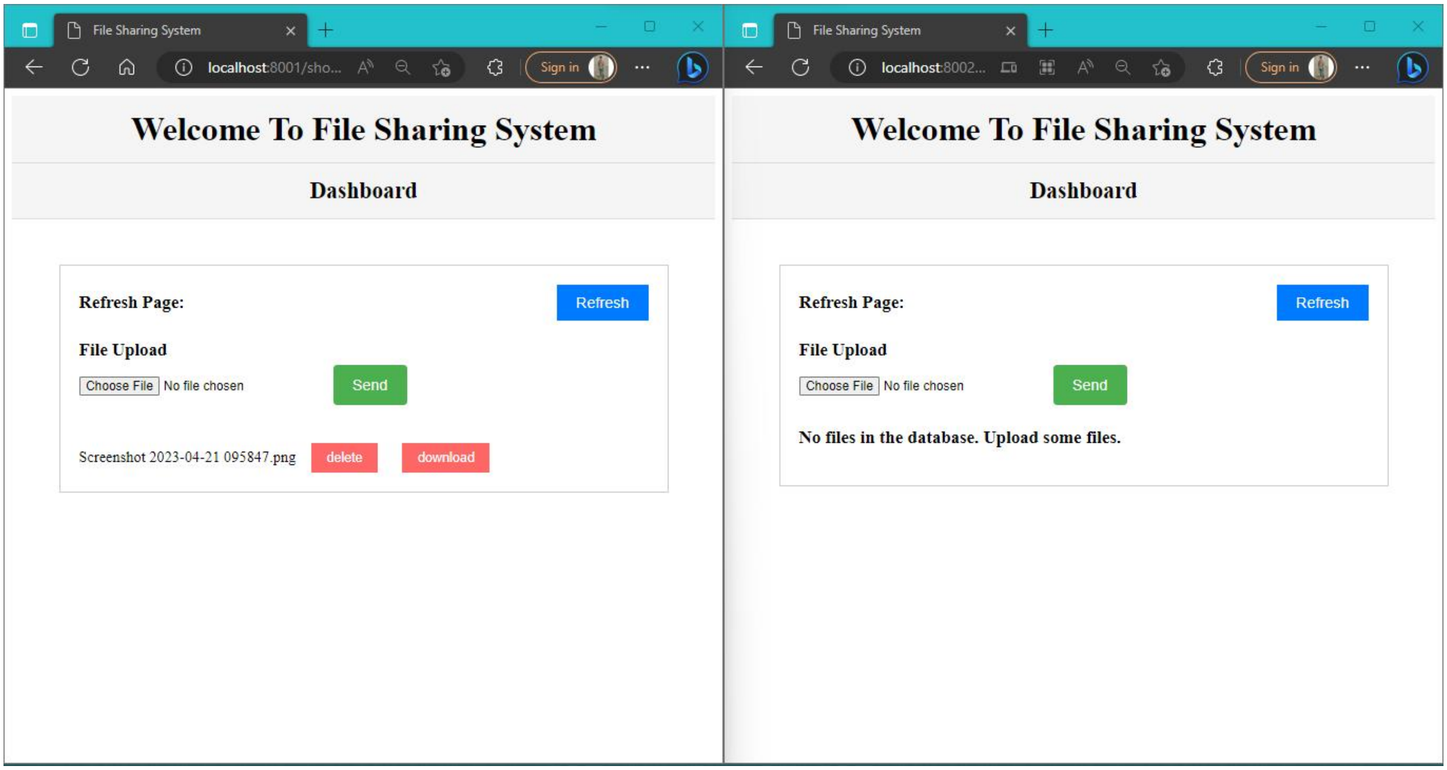
Refresh Page:

File Upload

No file chosen

No files in the database. Upload some files.

Peer1 sent file to peer 2 :



Welcome To File Sharing System

Dashboard

Refresh Page:

File Upload

No file chosen

Screenshot 2023-04-21 095847.png

Welcome To File Sharing System

Dashboard

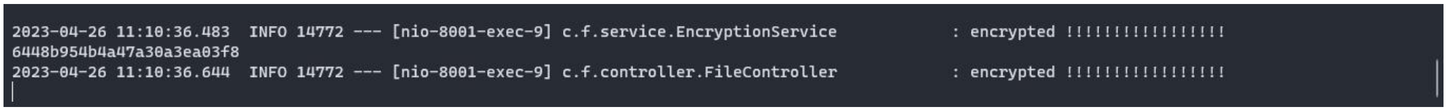
Refresh Page:

File Upload

No file chosen

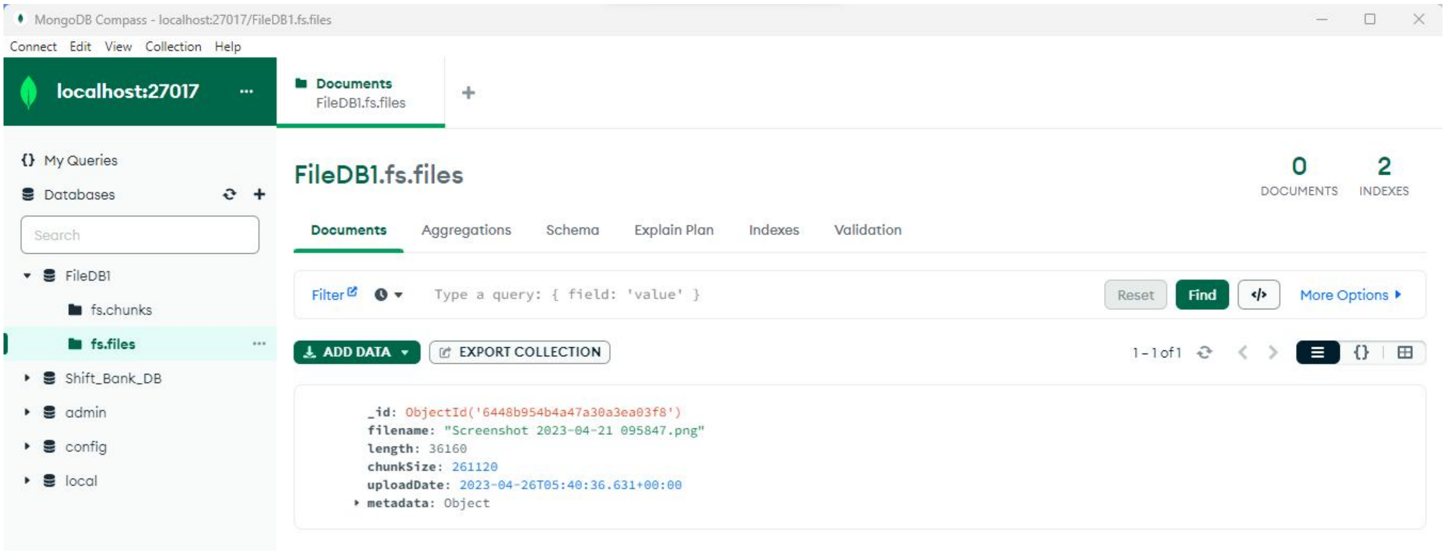
No files in the database. Upload some files.

Log Message saying file is encrypted in Spring-Boot console



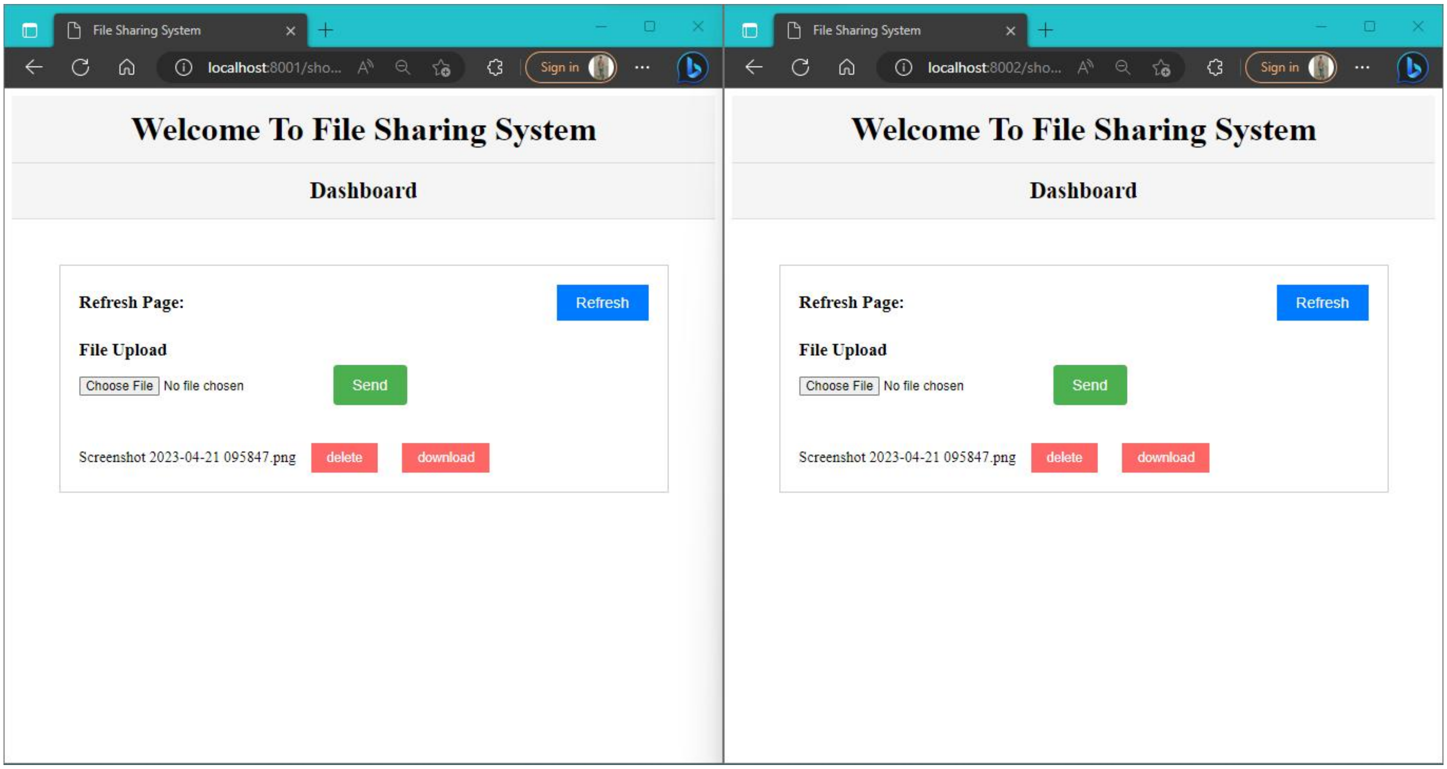
```
2023-04-26 11:10:36.483  INFO 14772 --- [nio-8001-exec-9] c.f.service.EncryptionService            : encrypted !!!!!!!!!!!!!!!!!
6448b954b4a47a30a3ea03f8
2023-04-26 11:10:36.644  INFO 14772 --- [nio-8001-exec-9] c.f.controller.FileController            : encrypted !!!!!!!!!!!!!!!!!
```

File sent by Peer1 encrypted and stored in the MongoDB Database

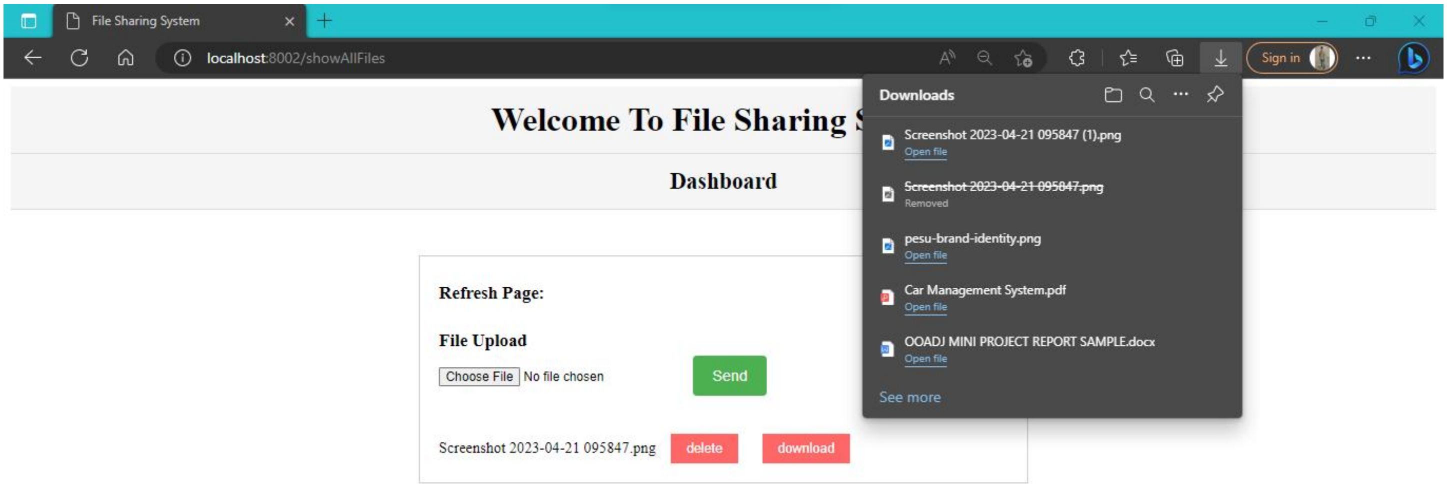


```
_id: ObjectId('6448b954b4a47a30a3ea03f8')
filename: "Screenshot 2023-04-21 095847.png"
length: 36160
chunkSize: 261120
uploadDate: 2023-04-26T05:40:36.631+00:00
metadata: Object
```

After Page refresh at peer2 side file has appeared in the peer2 dashboard



Peer2 downloading the file that is received received



Welcome To File Sharing S

Dashboard

Refresh Page:

File Upload

Screenshot 2023-04-21 095847.png

Downloads

Screenshot 2023-04-21 095847 (1).png
Open file

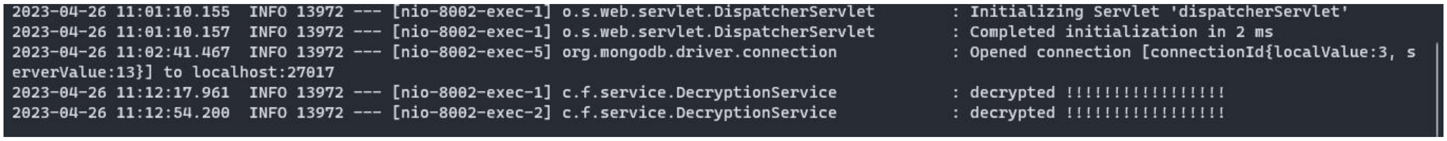
~~Screenshot 2023-04-21 095847.png~~
Removed

pesu-brand-identity.png
Open file

Car Management System.pdf
Open file

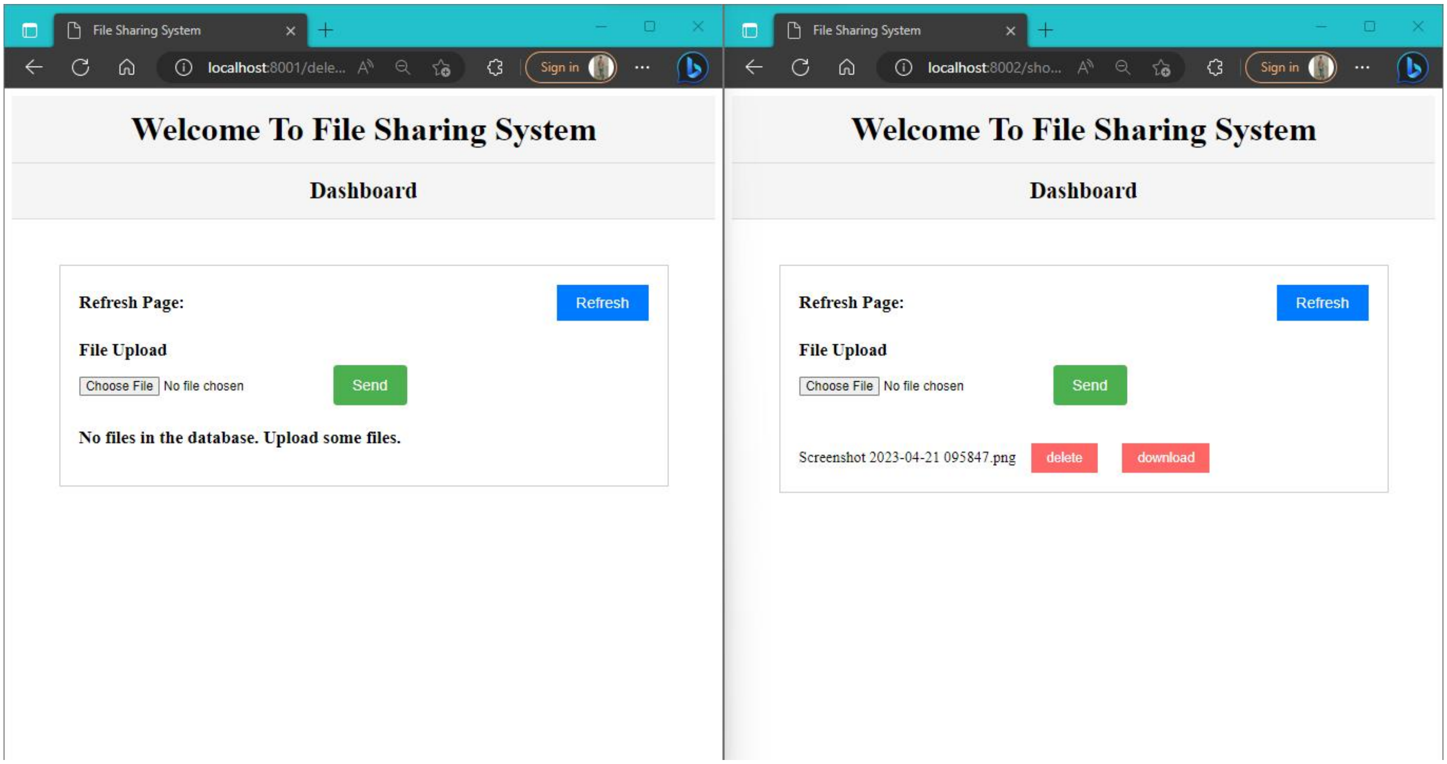
OOADJ MINI PROJECT REPORT SAMPLE.docx
Open file

Log Message saying file is decrypted before downloading in Spring-Boot console



```
2023-04-26 11:01:10.155  INFO 13972 --- [nio-8002-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-04-26 11:01:10.157  INFO 13972 --- [nio-8002-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
2023-04-26 11:02:41.467  INFO 13972 --- [nio-8002-exec-5] org.mongodb.driver.connection            : Opened connection [connectionId{localValue:3, s
erverValue:13}] to localhost:27017
2023-04-26 11:12:17.961  INFO 13972 --- [nio-8002-exec-1] c.f.service.DecryptionService            : decrypted !!!!!!!!!!!!!!!!!
2023-04-26 11:12:54.200  INFO 13972 --- [nio-8002-exec-2] c.f.service.DecryptionService            : decrypted !!!!!!!!!!!!!!!!!
```

Peer 1 deleted the file which they sent :



Welcome To File Sharing System

Dashboard

Refresh Page:

File Upload

No files in the database. Upload some files.

Welcome To File Sharing System

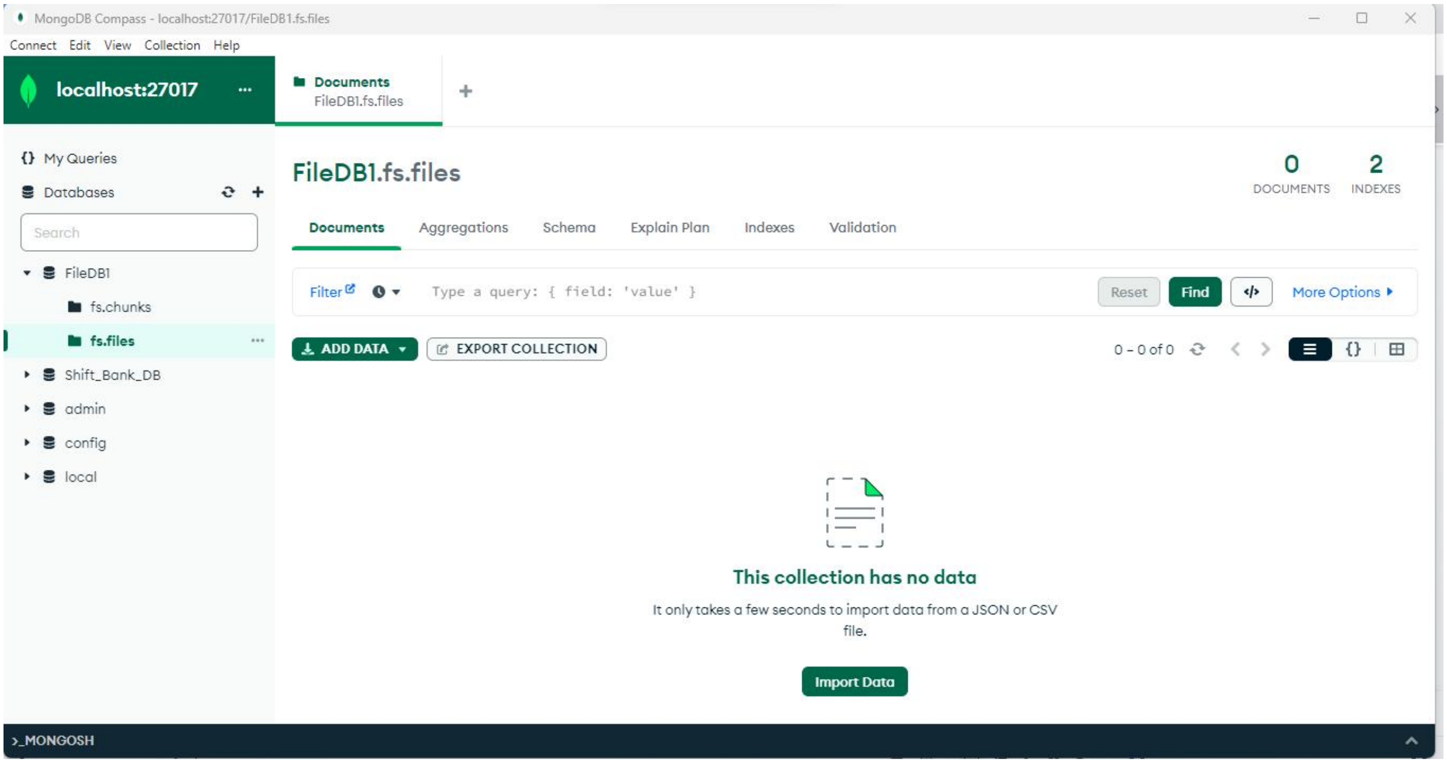
Dashboard

Refresh Page:

File Upload

Screenshot 2023-04-21 095847.png

File has been dropped in the DataBase that peer 1 deleted



MongoDB Compass - localhost:27017/FileDB1.fs.files

FileDB1.fs.files

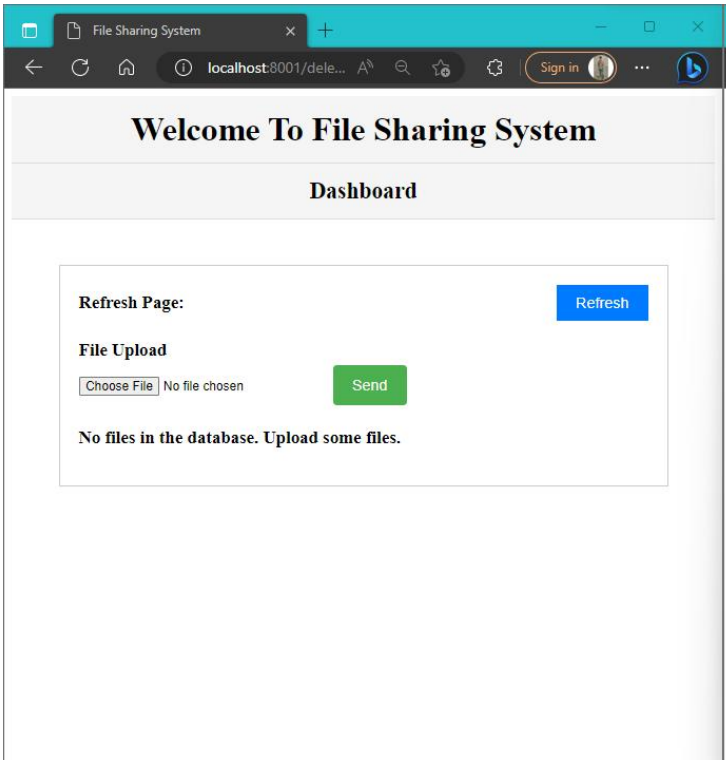
0 DOCUMENTS 2 INDEXES

0 - 0 of 0

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

After Page refresh at peer2 side file has disappeared in the peer2 dashboard



Welcome To File Sharing System

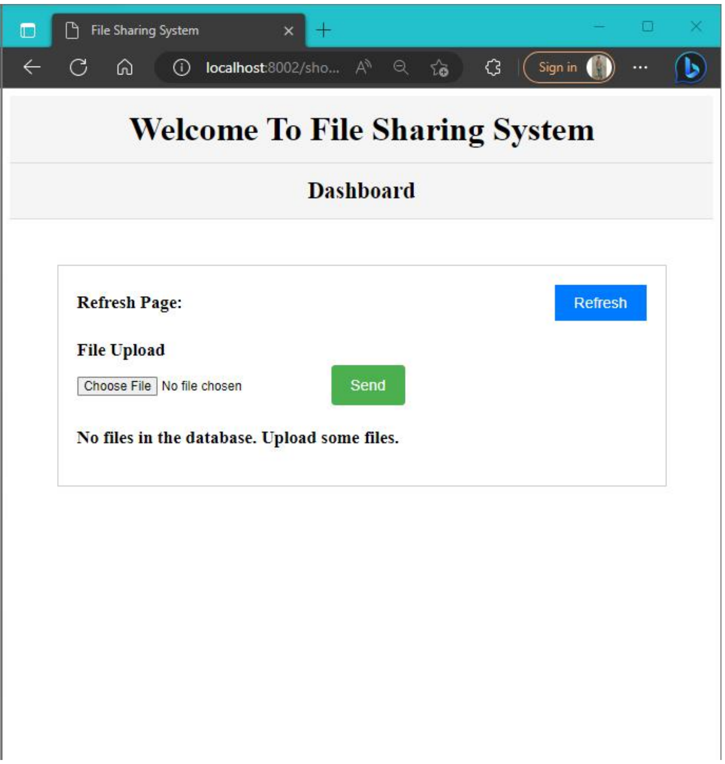
Dashboard

Refresh Page:

File Upload

No file chosen

No files in the database. Upload some files.



Welcome To File Sharing System

Dashboard

Refresh Page:

File Upload

No file chosen

No files in the database. Upload some files.