LOCAL FILE SHARER

WEB ESSENTIAL

MINI-PROJECT REPORT

Submitted by

JASWANTH REDDY B 211701009

AJAY KRISHNA C R 211701004

in partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND DESIGN

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

(AUTONOMOUS)

MAY, 2024

BONAFIDE CERTIFICATE

Certified that this project **"LOCAL FILE SHARER"** is the bonafide work of **"JASWANTH REDDY B (211701009), AJAY KRISHNA C R (211701004)"** who carried out the project work under my supervision.

SIGNATURE

Prof. Uma Maheshwar Rao,

Head of the Department,

Computer Science & Design

Rajalakshmi Engineering College

Thandalam, Chennai -602105.

SIGNATURE

Mr. Duraimurugan,

Assistant Professor,

Computer Science & Engineering

Rajalakshmi Engineering College

Thandalam, Chennai -602105.

Submitted for the End semester practical examination to be held on ____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

I express my sincere thanks to my beloved and honourable chairman **MR.S.MEGANATHAN** and the chairperson **DR.M.THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to my respected and honourable principal **Dr. S.N.MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by my head of the department **Prof. UMA MAHESHWAR RAO,** and my Assistant Professor **Mr.DURAIMURUGAN,** for being ever supporting force during my project work.

I also extend my sincere and hearty thanks to my internal guide **Mr. DURAIMURUGAN** fill for his valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering.

AJAY KRISHNA C R (211701004)
JASWANTH REDDY B (211701009)

TABLE OF CONTENTS

CHAPTER NO. **PAGE**

TITLE

LIST OF FIGURES

LIST OF ABBREVIATION

ABBREVIATION	ACRONYMS
SRS	Software Requirement Specification
CLIENT/SERVER	The entity who will be using the interviewfeedback system.
SERVER	A system that runs in Linux
RAM	Random access memory
MONDODB	A document based database managementsystem
HTTP	Hyper Text Transfer Protocol
PDF	Portable Document Format
Username	Unique name given to each account ofdigital library
Password	Unique word set by each user as a secret code.

ABSTRACT

The main goal of this site is to share files locally. Each user in a network that are connected to each other can share files locally without the use of internet. It also allows users to share files at a faster rate. But there is a problem in this kind of software. We usually create applications specifically to the type of device or the operating system. It therefore highly reduces the portability and installation of the software. To overcome this we introduce this feature as a website. It is implemented with the help of signaling servers which are usually connected with the help of websocket and then use a api that is used in browsers to connect systems as peers called WebRTC. Therefore we can have an additional feature of sending files also over the internet. We could also see the users in the same network automatically which gives easy access to determine the user.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

1.2 In an increasingly interconnected world, the need for efficient and convenient file sharing solutions has never been more vital. However, the traditional reliance on the internet for such tasks is not always ideal, especially in local network scenarios where privacy, speed, and accessibility are paramount. To address this, we introduce a cutting-edge web-based file sharing platform designed to facilitate seamless file transfers within a local network.

1.3 SCOPE OF THE WORK

The scope of work for developing our web-based local file sharing platform encompasses the following key areas: project planning and design, platform development using WebRTC for peer-to-peer file sharing, local and internet file transfer capabilities, user management and security measures, thorough testing and quality assurance, documentation creation, deployment and hosting setup with scalability considerations, user support and training resources, ongoing maintenance and updates, compliance with legal and data protection regulations, marketing and promotion strategies, feedback collection for iterative improvements, and budget and resource management.

1.4 PROBLEM STATEMENT

In today's interconnected world, the need for efficient and secure local file sharing within a network is evident. Traditional methods often rely on the internet, resulting in privacy concerns, slower transfer speeds, and potential network limitations. Moreover, existing solutions typically require device-specific applications, limiting accessibility and hindering cross-platform compatibility.

1.5 AIM AND OBJECTIVES OF THE PROJECT

Aim:
The primary aim of this project is to develop and implement a web-based local file sharing platform that facilitates efficient, secure, and cross-platform file transfers within a local network while offering the flexibility of internet-based file sharing.

Objectives:

Platform Development: Develop a user-friendly web-based platform that allows users to share files within a local network and over the internet.

WebRTC Integration: Implement WebRTC technology for direct peer-to-peer file transfers, ensuring fast and secure communication between users.

Cross-Platform Compatibility: Ensure that the platform is accessible from various devices and operating systems, eliminating the need for device-specific applications.

Local File Sharing: Enable users to share files locally within the same network without relying on internet connectivity, ensuring privacy and speed.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

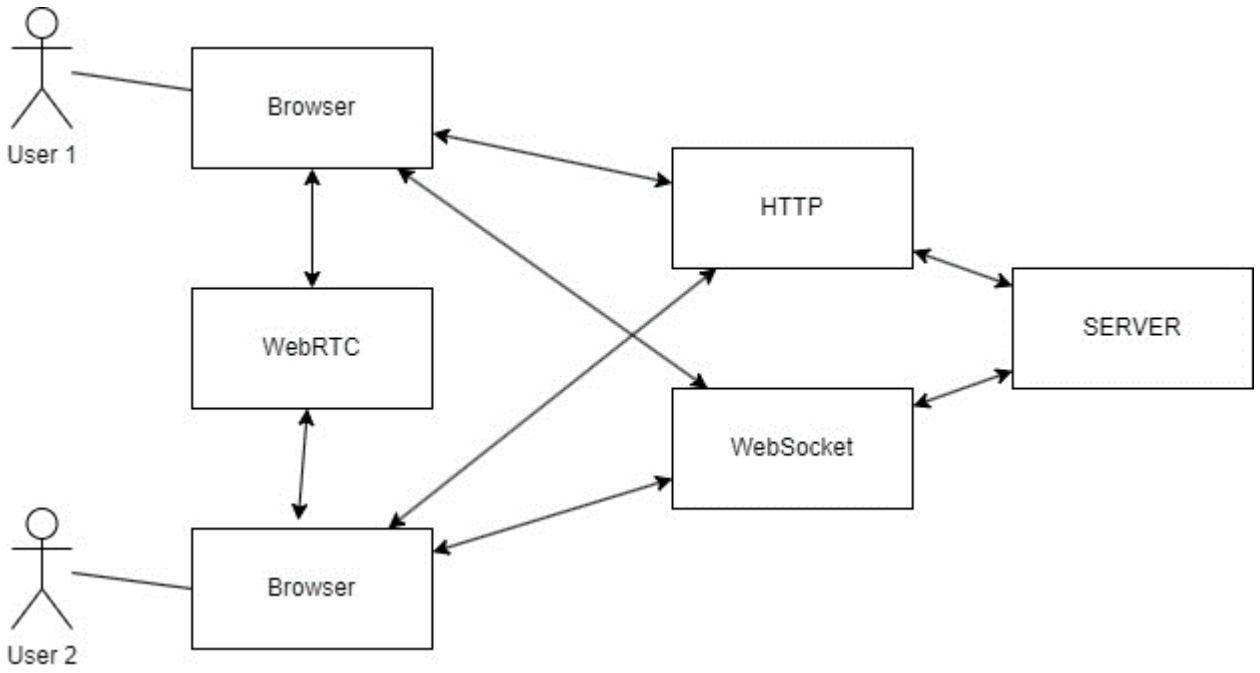
Processor	:	Pentium IV Or Higher
Memory Size	:	4 GB (Minimum)
HDD	:	40 GB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10+
Front – End	:	JavaScript, CSS, HTML
Back – End	:	Node,MongoDB

CHAPTER 3

ARCHITECTURE DIAGRAM



Architecture Diagram for local file sharing

CHAPTER 4

MODULE DESCRIPTION

Home Module

Home module has all the other modules. We have access to the entire website from this module, it is the first page opens when we start the website. This is the only module that does all the work like filesharing

About Module

This module provides basic information about our website, it has all the contents and the creator details.

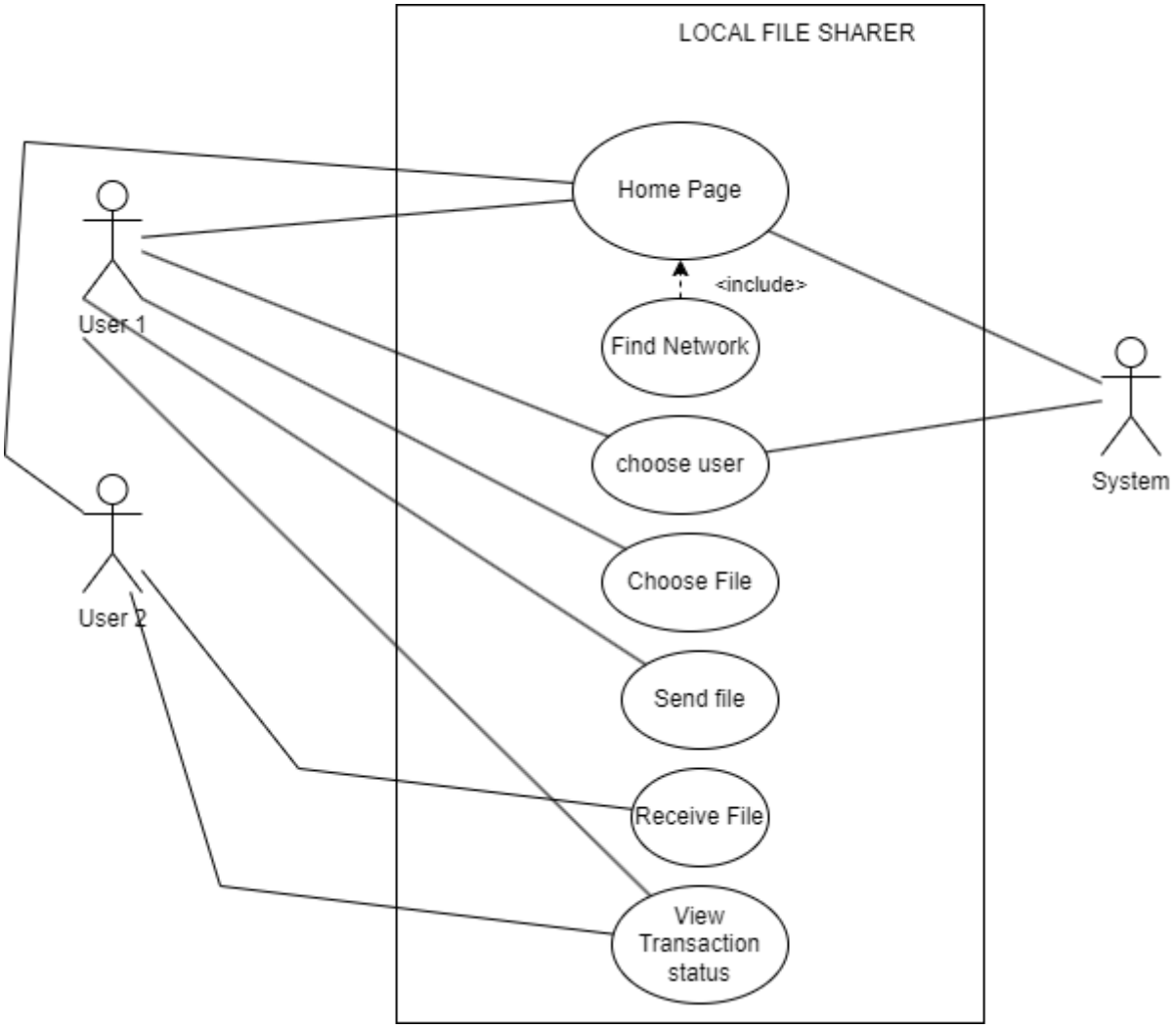
Privacy Policy Module

This module has the details of statement or legal document (in privacy law) that discloses some or all of the ways a party gathers, uses, discloses, and manages a customer or client's data.

CHAPTER 5

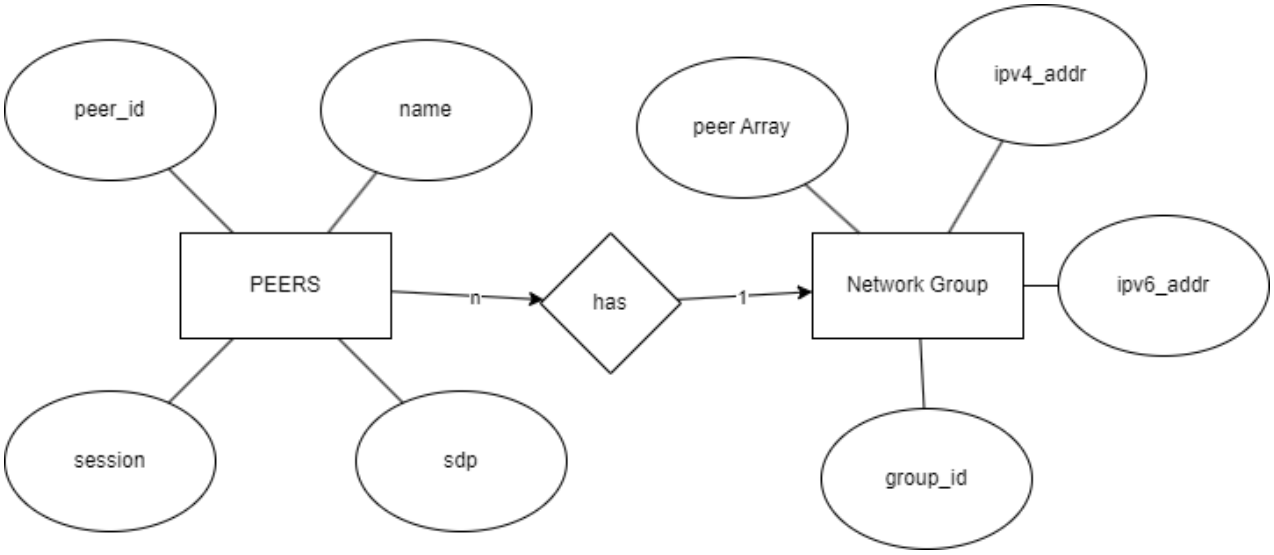
SYSTEM DESIGN

5.1 USE CASE DIAGRAM



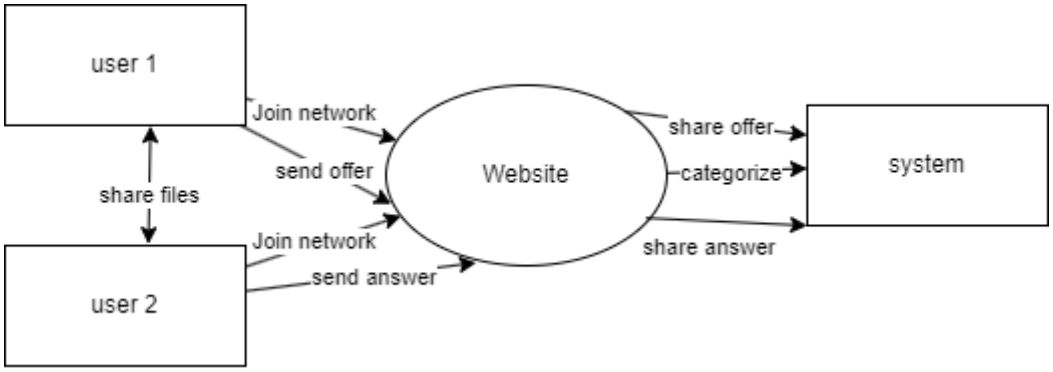
USE CASE DIAGRAM FRO LOCAL FILE SHARING

5.2 ER DIAGRAM

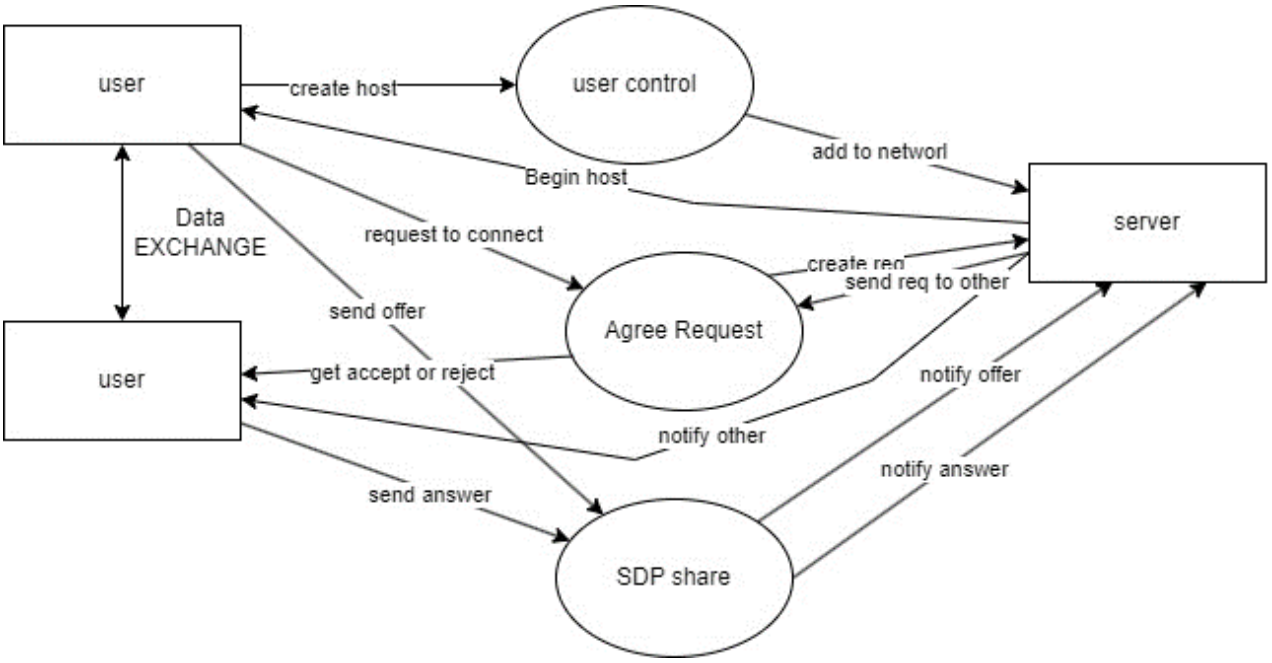


ER DIAGRAM FOR LOCAL FILE SHARING

5.3 DFD DIAGRAM

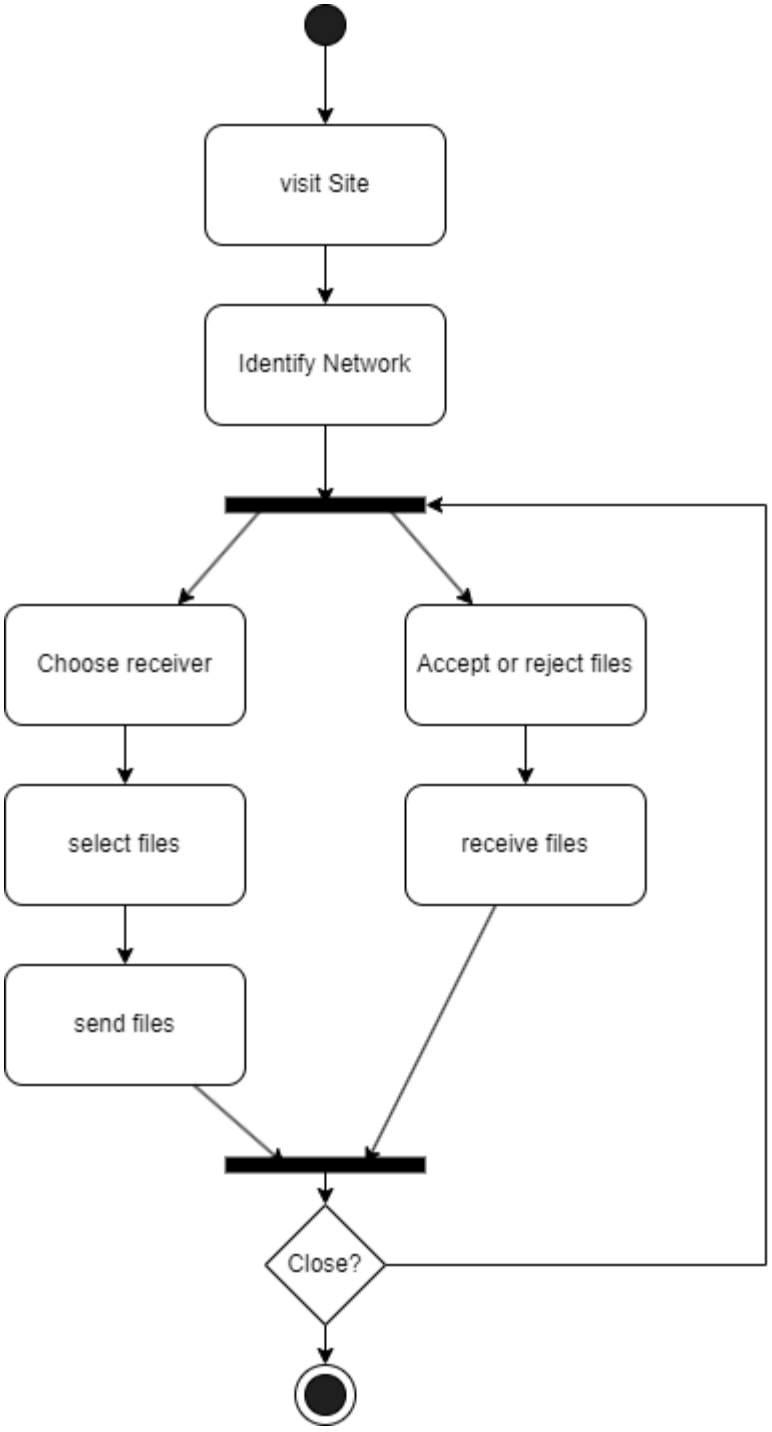


DFD level-0 Diagram



DFD Level-1 Diagram

5.4 ACTIVITY DIAGRAM



ACTIVITY DIAGRAM FOR LOCAL FILE SHARING

CHAPTER 6

SAMPLE CODING

Index.js backend

```
const { app, DBACTION } = require('./app')
const WebSocketServer = require('ws')
const http = require('http')
const ipv6Expand = require("./ipv6expand")
const db = require('./db')

const server = http.createServer()
const ws = new WebSocketServer.Server({
   server: server,
   perMessageDeflate: false
})

server.on('request', app)
ws.peers = {}
let DBAction = new DBACTION();
DBAction.deleteAll()
ws.on('connection', (conn) => {
   console.log("recieved a connection");

   conn.on('message', async (message) => {
      try {
         message = message.toString()
         message = JSON.parse(message)
         conn.pause()
         if (message.operation == 'reqpair') {
            let peerSession = (await DBAction.getPeer(message.name)).session
```

```
            ws.peers[peerSession].send(JSON.stringify({ operation: 'reqpair', name:
conn.peer.name, filedata: message.filedata }))
        }
        else if (message.operation == 'acceptreq') {
            let peerSession = (await DBAction.getPeer(message.name))
            ws.peers[peerSession.session].send(JSON.stringify({ operation:
'acceptreq', name: conn.peer.name }))
        }
        else if (message.operation == 'rejectreq') {
            let peerSession = (await DBAction.getPeer(message.name))
            ws.peers[peerSession.session].send(JSON.stringify({ operation:
'rejectreq', name: conn.peer.name }))
        }
        else if (message.operation == 'sendoffer') {
            let peerSession = (await DBAction.getPeer(message.name))
            ws.peers[peerSession.session].send(JSON.stringify({ operation:
'recvoffer', name: conn.peer.name, offer: message.offer }))
        }
        else if (message.operation == 'sendanswer') {
            let peerSession = (await DBAction.getPeer(message.name))
            ws.peers[peerSession.session].send(JSON.stringify({ operation:
'recvanswer', name: conn.peer.name, answer: message.answer }))
        }
        else if (message.operation == 'createhost') {
            if (message.network['ipv6'] != 400) {
                message.network['ipv6'] =
ipv6Expand(message.network['ipv6']).slice(0, 19);
            }
            res = await DBAction.addUser(message.network)
            conn.peer = res.peer
```

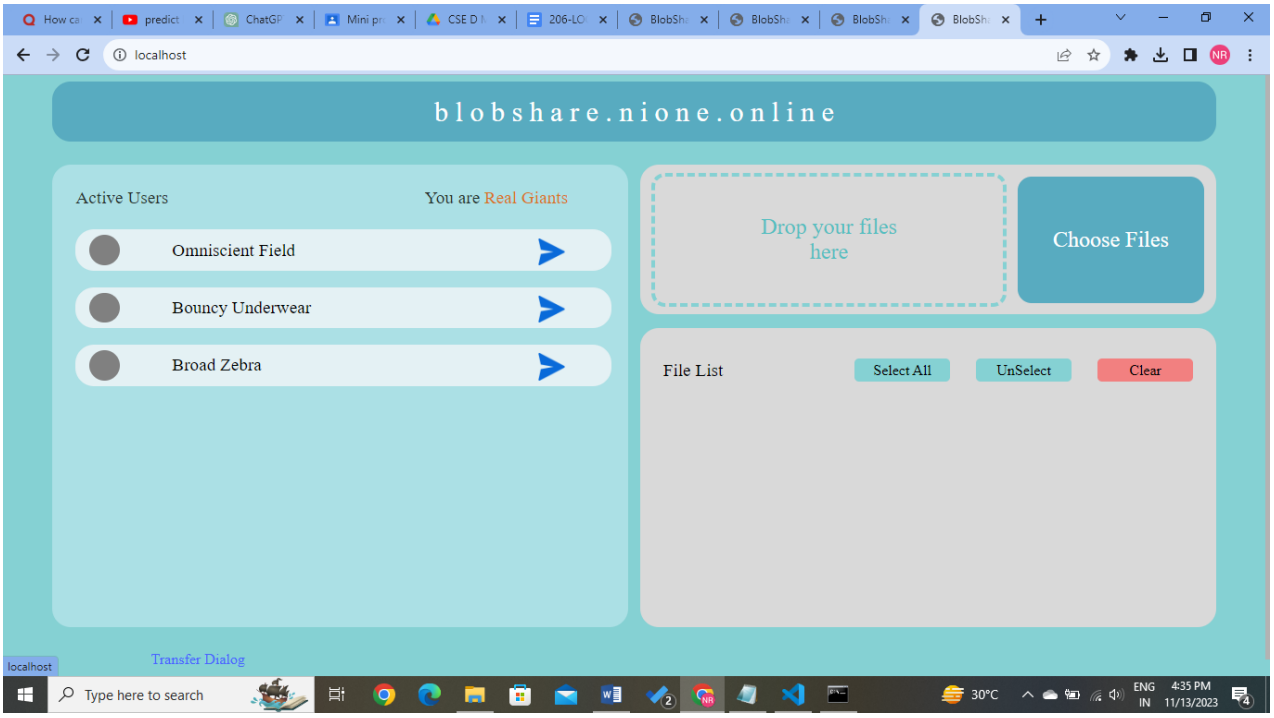
```
            ws.peers[conn.peer.session] = conn
            let otherHosts = await DBAction.getNetworkPeers(conn.network,
conn.peer);
            let namelist = []
            otherHosts.forEach(element => {
               ws.peers[element.session].send(JSON.stringify({

                  operation: 'newhost',
                  name: conn.peer.name
               }))
               namelist.push(element.name)
            });
            conn.send(JSON.stringify({
               operation: 'beginhosts',
               namelist: namelist,
               name: conn.peer.name
            }))
         }
         else if (message.operation == 'onicecandidate') {

            let peerSession = (await DBAction.getPeer(message.name))
            ws.peers[peerSession.session].send(JSON.stringify({ operation:
'updateIce', name: conn.peer.name, ice: message.sdp }))

         }
         conn.resume()
      }
      catch (err) {
      }
   })
```

```
    conn.on('close', async () => {
        try{
        if (conn.network != undefined) {
            let otherHosts = await DBAction.getNetworkPeers(conn.network,
conn.peer);
            otherHosts.forEach(element => {
                ws.peers[element.session].send(JSON.stringify({
                    operation: 'hostleft',
                    name: conn.peer.name
                }))
            });

            delete ws.peers[conn.peer.session]
            DBAction.popPeer(conn.network, conn.peer)
        }
    }
    catch(err){

    }
    })
})

server.listen(80);
```

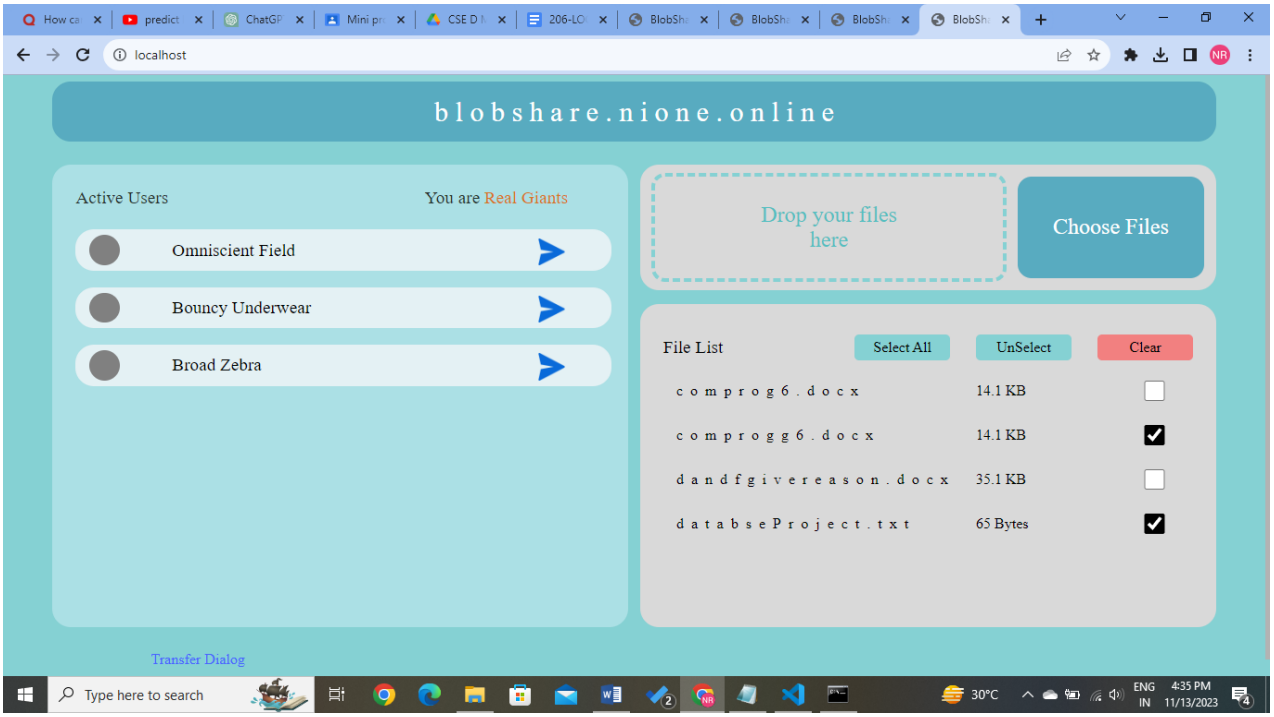
CHAPTER 7

SCREEN SHOTS

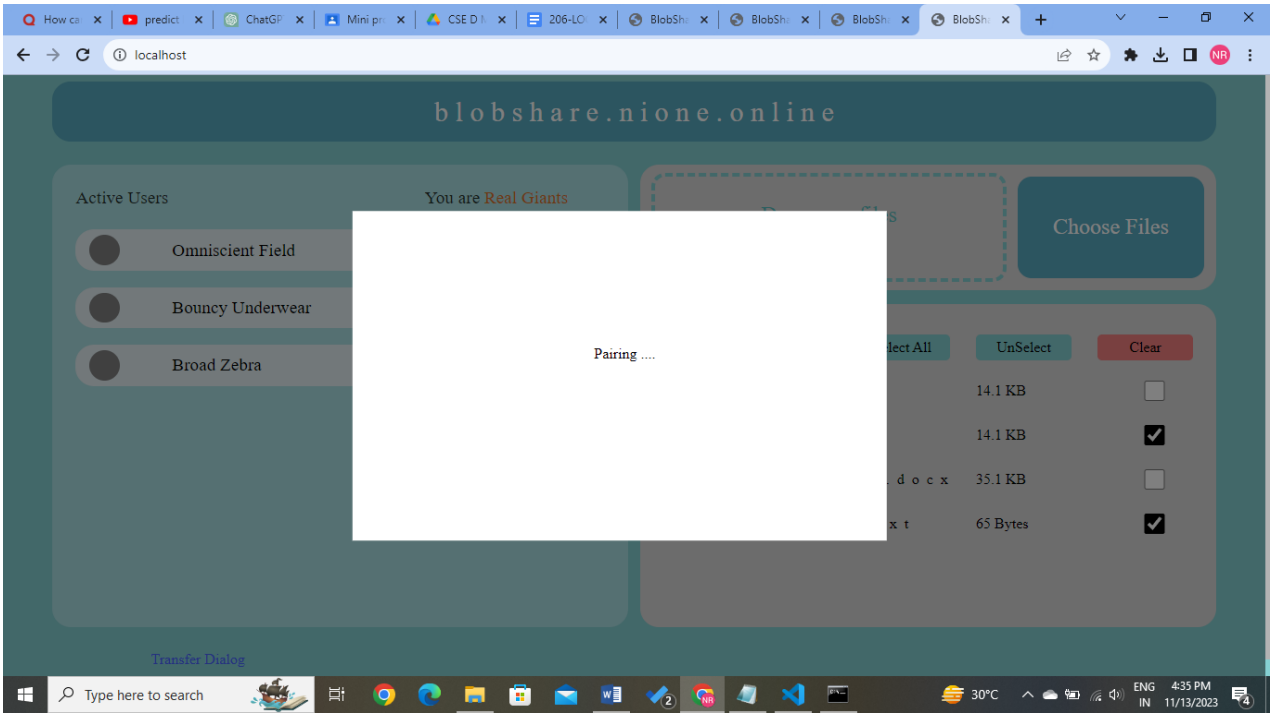
Home screen



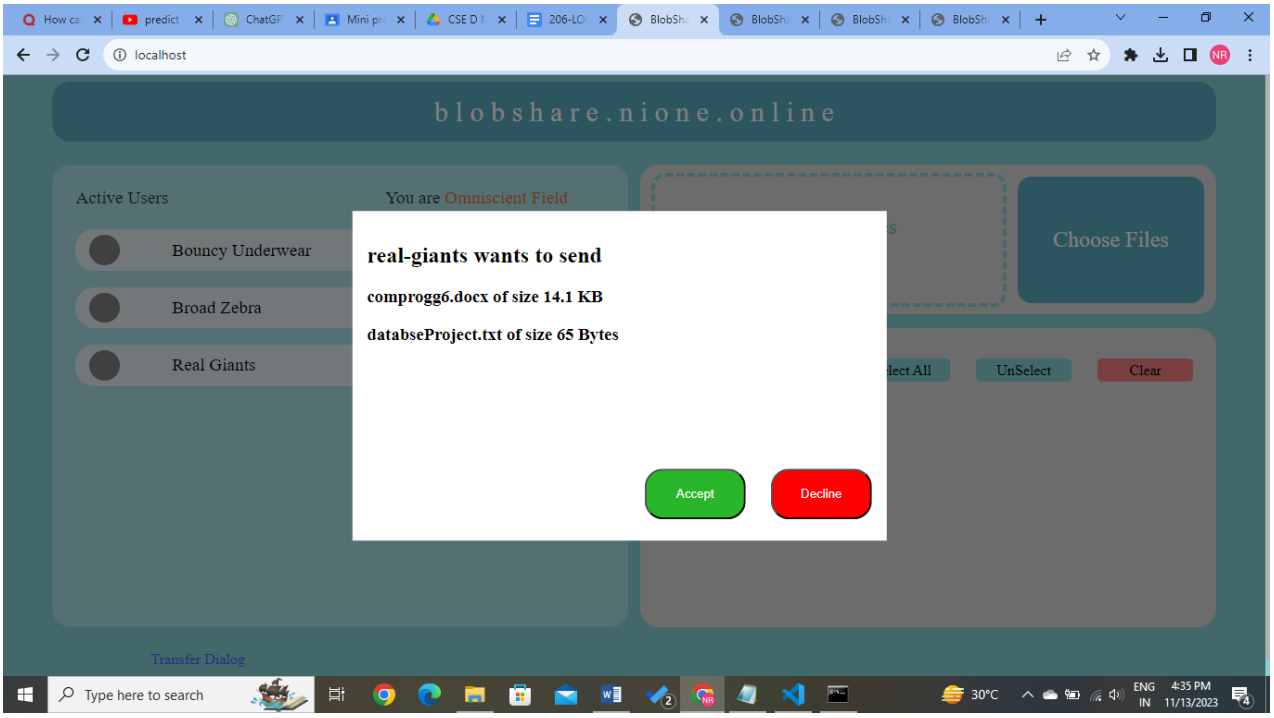
File Selection



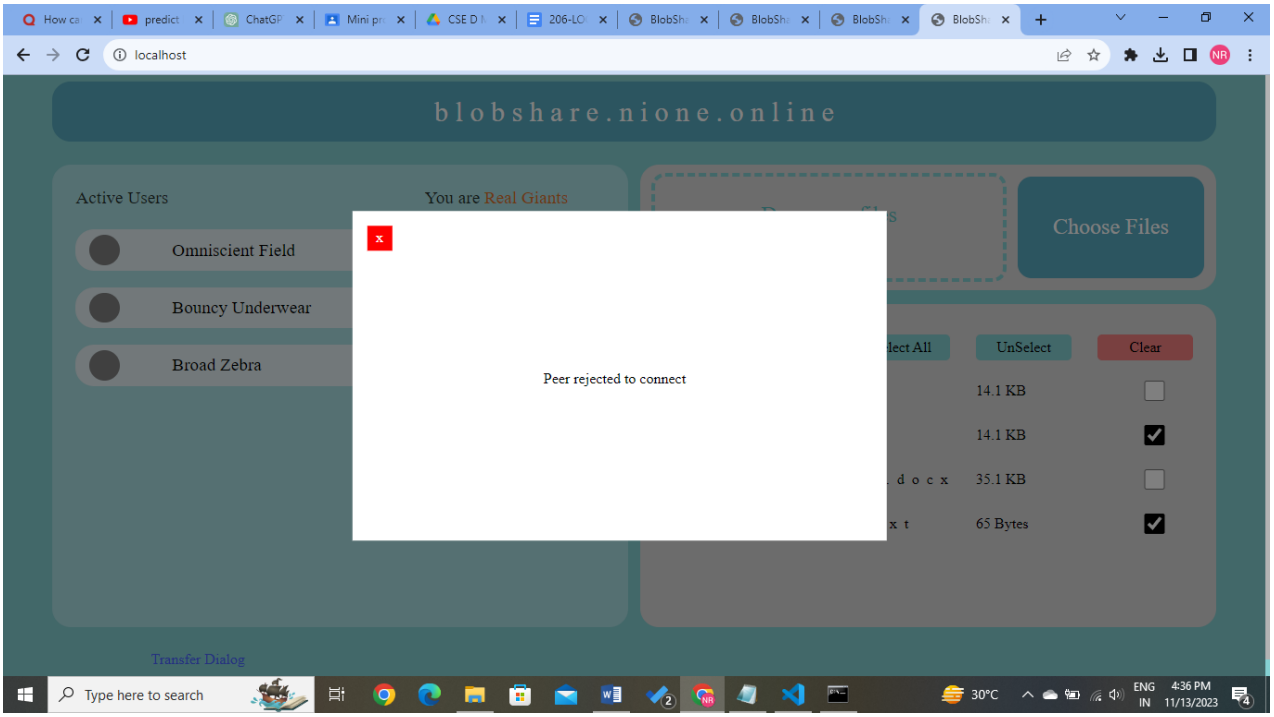
Request to pair



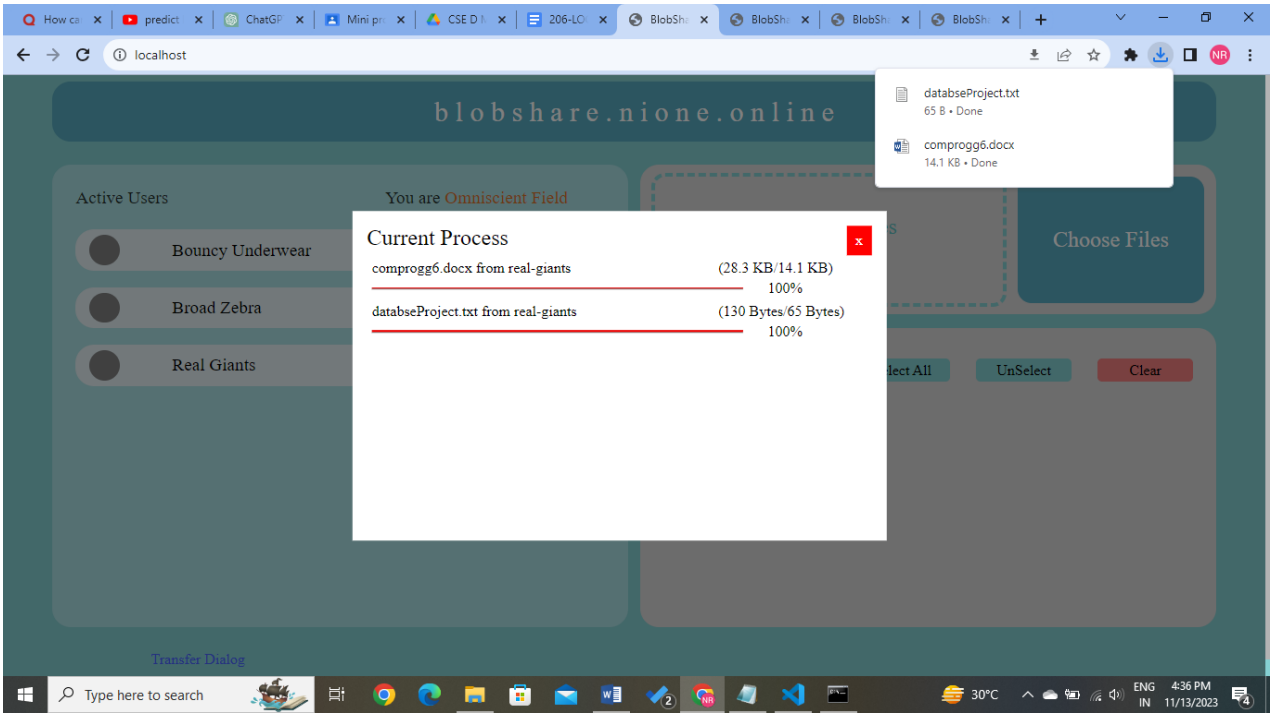
ACCEPT OR REJECT SCREEN



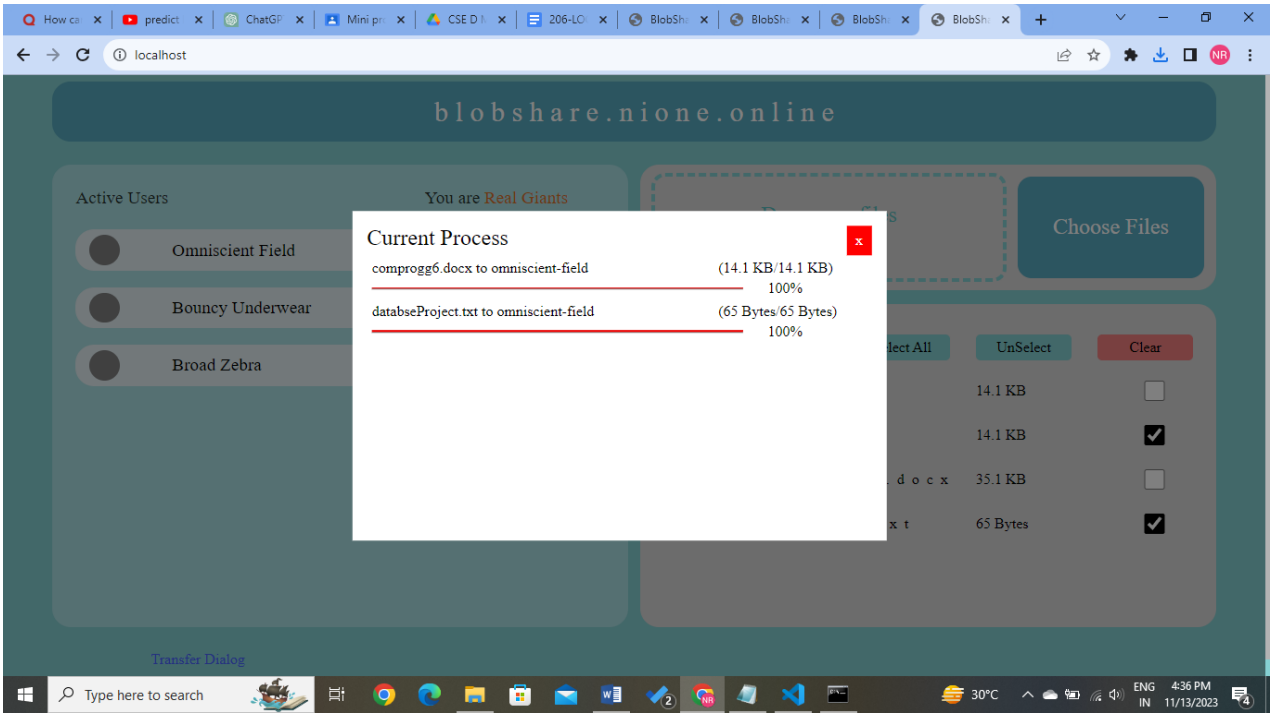
DECLINED SCREEN



RECIEVER TRANSFER DIALOG



SENDER TRANSFER DIALOG



CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

In conclusion, the development of the web-based local file sharing platform represents a significant stride toward addressing the challenges associated with secure, swift, and versatile file sharing within local networks. By leveraging WebRTC technology, we've successfully created a user-friendly solution that enables seamless file exchanges within the same network while offering the flexibility of internet-based file sharing when necessary. The platform's implementation of automatic user detection, robust security measures, and cross-platform compatibility ensures a holistic approach to meet user needs.

Future Enhancements:

Enhanced Security Features: Implement advanced encryption methods to further fortify the security of file transfers.

Collaborative Features: Introduce collaborative functionalities, allowing simultaneous editing and real-time collaboration on shared files.

File Synchronization: Enable automatic synchronization of specified folders across devices within the network.

Improved User Interface: Continuously enhance the platform's user interface to ensure a more intuitive and engaging experience for users.

Integration with Cloud Services: Incorporate integration with cloud storage services for extended file management options.

Machine Learning for Content Organization: Utilize machine learning algorithms to categorize and organize shared content more efficiently.

REFERENCES

- WebRTC documentation
- Express js in geek for geeks
- WebSockets documentation