Contents

INTRODUCTION:

PEER TO PEER FILE SHARING

P2P file sharing has revolutionized the way we exchange data, enabling users from all over the world to connect and share files directly with one another.

P2P file sharing differs from traditional file sharing methods as it eliminates the need for a centralized server. Instead, it allows users to connect directly to each other's computers, creating a decentralized network where files can be shared and downloaded seamlessly.

PROBLEM STATEMENT

Designing a Peer-to-Peer Distributed File Sharing System in Java, emphasizing Java IO and Thread concepts. Enable users to share files directly, supporting concurrent transfers. Implement Peer Discovery for network connection, Socket-based File Transfer, thread-based Concurrency for efficient uploads/downloads, and a user-friendly File Management interface.

METHODOLOGY: P2P FILE SHARING SYSTEM

Objective Definition:

Develop a Peer-to-Peer (P2P) Distributed File Sharing System in Java.
Enable users to share and download files directly without a central server

Directory Structure:

Define shared and downloads directories (SHARED_DIRECTORY and DOWNLOADS_DIRECTORY).
Ensure existence of the downloads directory.

System Architecture:

Utilize a client-server model with sender and receiver roles.
Communication established through sockets.

Peer Simulation:

Simulate sender and receiver functionality on separate threads.
Utilize local IP addresses for sender and receiver (127.0.0.1).

File Transfer Mechanism:

Employ ObjectOutputStream and ObjectInputStream for efficient file serialization and deserialization.
Utilize byte buffers for reading and writing file content.

Concurrency Management:

Leverage Java threads to enable concurrent uploads and downloads.
Separate sender and receiver functionalities into distinct threads

IMPLEMENTATION

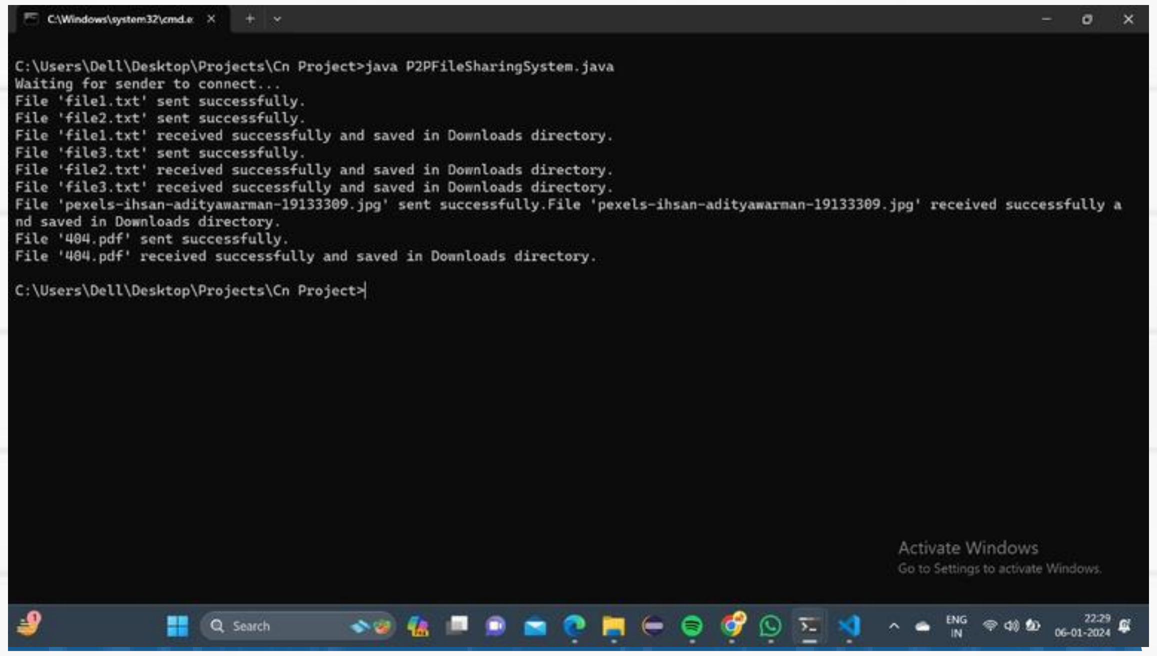
SENDER

The sender connects to the receiver, sequentially transmitting file names and contents via ObjectOutputStream. File content is read in 4096-byte chunks from the shared directory, concluding with an "exit" signal.

RECIEVER

The receiver, upon ServerSocket connection, establishes a Downloads directory and uses ObjectInputStream to continuously receive file names and contents. Files are saved, with ongoing success updates until completion.

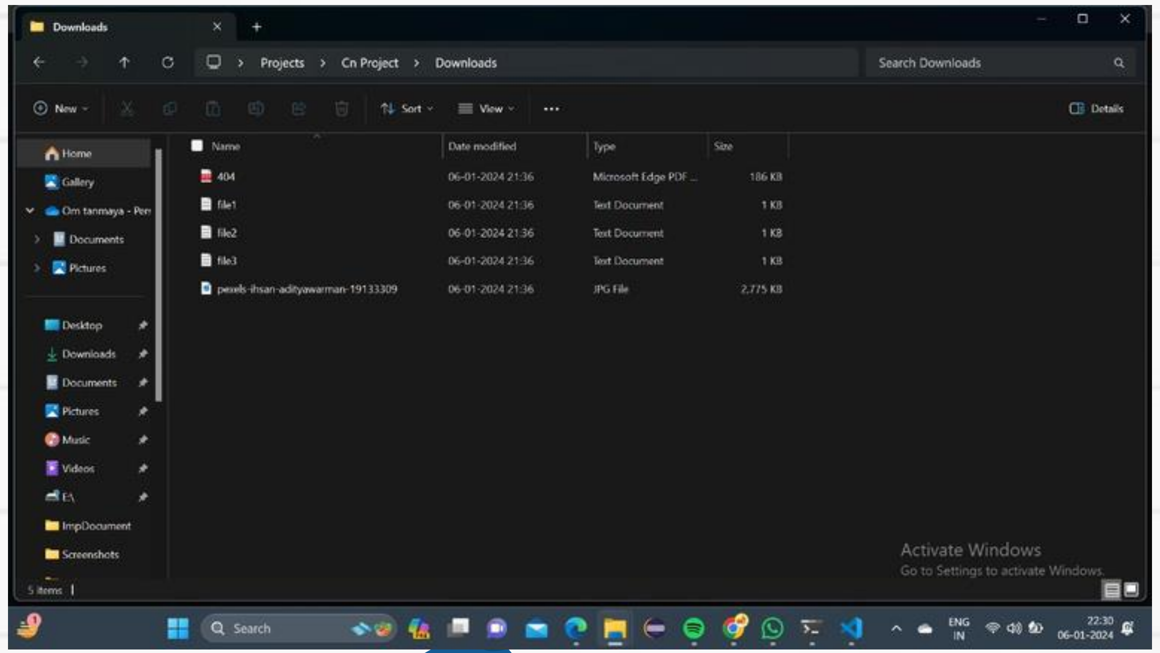
RESULT INTERPRETATION



```
C:\Users\Dell\Desktop\Projects\Cn Project>java P2PFileSharingSystem.java
Waiting for sender to connect...
File 'file1.txt' sent successfully.
File 'file2.txt' sent successfully.
File 'file1.txt' received successfully and saved in Downloads directory.
File 'file3.txt' sent successfully.
File 'file2.txt' received successfully and saved in Downloads directory.
File 'file3.txt' received successfully and saved in Downloads directory.
File 'pexels-ihsan-adityawarman-19133309.jpg' sent successfully.File 'pexels-ihsan-adityawarman-19133309.jpg' received successfully a
nd saved in Downloads directory.
File '404.pdf' sent successfully.
File '404.pdf' received successfully and saved in Downloads directory.

C:\Users\Dell\Desktop\Projects\Cn Project>
```

The image illustrates seamless file upload and download completion, marking a successful exchange in the P2P system.

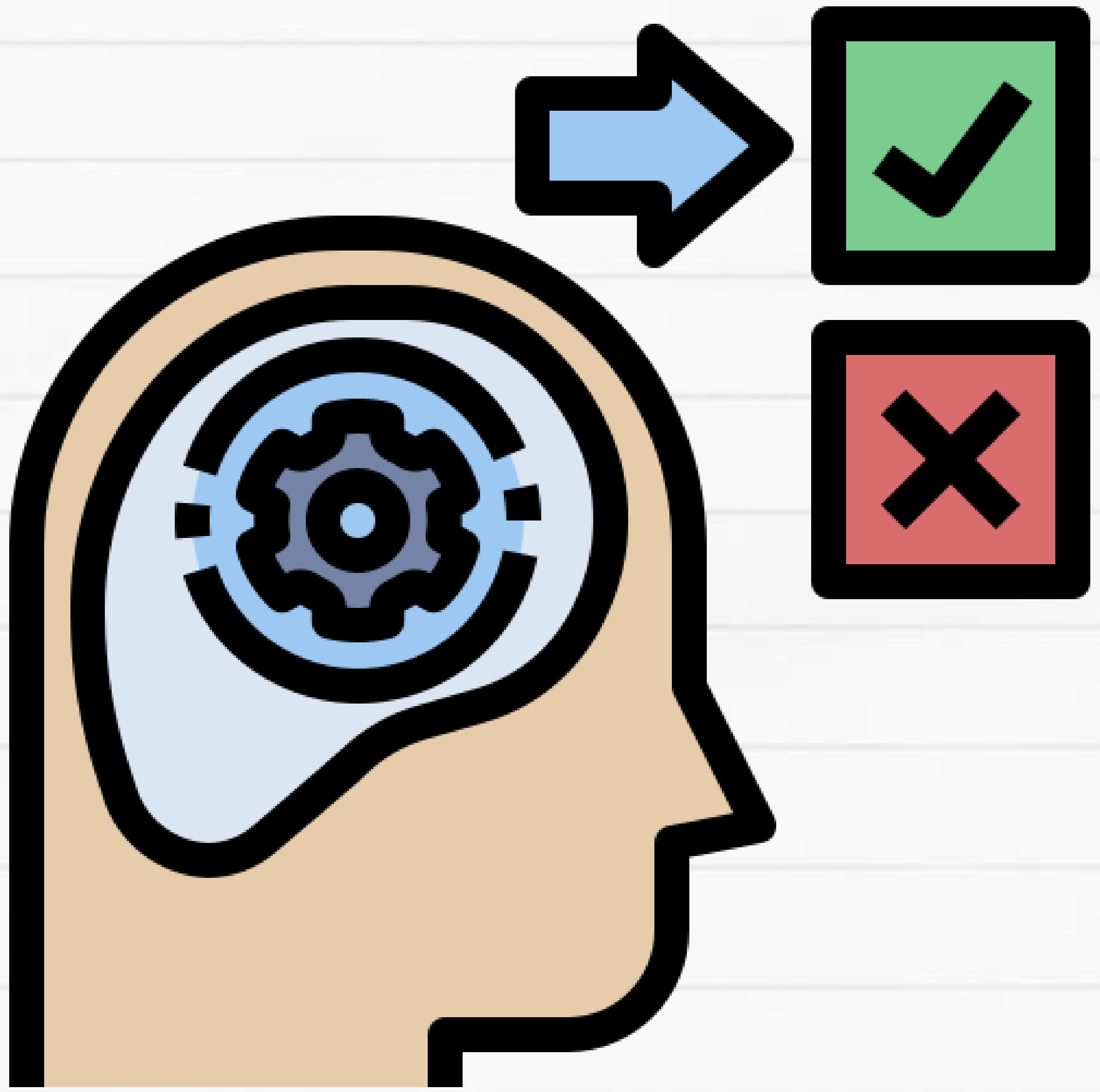


Projects > Cn Project > Downloads

Name	Date modified	Type	Size
404	06-01-2024 21:36	Microsoft Edge PDF ...	186 KB
file1	06-01-2024 21:36	Text Document	1 KB
file2	06-01-2024 21:36	Text Document	1 KB
file3	06-01-2024 21:36	Text Document	1 KB
pexels-ihsan-adityawarman-19133309	06-01-2024 21:36	JPG File	2,775 KB

The image illustrate Files Downloaded Successfully in the Local Machine through Peer to Peer System.

CONCLUSION



In conclusion, our P2P Distributed File Sharing System showcases the power of Java IO and threading, enabling direct file sharing between peers without a central server. With robust features like Peer Discovery, efficient File Transfer, Concurrency support, and a user-friendly File Management interface, we've embraced decentralized collaboration seamlessly.

ANY QUESTIONS