5.

Title	Implement Client/Server UDP Application for Current Time Retrieval
Description	**As a** developer **I want to** create a server application that allows receiving client requests via UDP **So that** it responds with the current system time. **As a** programmer, **I want to** create a client application that receives the server's location via the command line and communicates with it via UDP **So that** it displays the received time
Acceptance Criteria	1. The server must be able to receive client requests via UDP and respond with the current system time. 2. The client must receive the server's location via the command line. 3. The client must send a request to the server, receive the current time, display it on standard output, and terminate execution.
Technical Requirements	• The application must be developed in Java. • Communication between client and server must be based on the UDP protocol.

6.

Title	Implement Client/Server UDP Application for File transfer
Description	**As a** programmer, **I want to** create a client/server application using the UDP protocol that allows clients to obtain a file stored on the server **So that** clients can transfer files from the server to a specified local directory, ensuring that files are transferred correctly and efficiently.
Acceptance Criteria	• **Server**: 1. It must be able to receive client requests via UDP. 2. It must be launched by passing the listening port and the local directory containing files that clients can request via the command line. 3. It should allow access only to files located in the specified directory or subdirectories. 4. It must be able to transfer files of any size, sending data blocks with a maximum size (e.g., MAX_DATA = 4000 bytes), with the last block being a UDP datagram with a content size of zero bytes. • **Client**: 1. It must be launched by passing the server's location, the desired file name, and the local directory where the file should be stored via the command line. 2. It must send a datagram to the server with the desired file name. 3. It must be able to receive and store the file in the specified local directory. 4. Any issue during the transfer, including timeout, should lead to aborting the operation.
Technical Requirements	• The application must be developed in Java.

	• Communication between client and server must be based on the UDP protocol. • It should be possible to transfer files of any size, with a maximum block size of 4000 bytes. • The application must correctly handle issues during the transfer, aborting the operation in case of error.

7.

Title	Implement Client/Server TCP Application for Current Time Retrieval
Description	**As a** programmer **I want to** create a server application that allows receiving client requests via TCP **So that** it responds with the current system time. **As a** programmer **I want to** create a client application that receives the server's location via the command line and communicates with it via TCP **So that** it displays the received time.
Acceptance Criteria	1. The server must be able to receive client requests via TCP and respond with the current system time. 2. The client must receive the server's location via the command line. 3. The client must send a request to the server, receive the current time, display it on standard output, and terminate execution.
Technical Requirements	• The application must be developed in Java. • Communication between client and server must be based on the TCP protocol.

8.

Title	Implement Client/Server TCP Application for File Transfer
Description	**As a** programmer **I want to** create a client/server application using the TCP protocol that allows clients to obtain a file stored on the server **So that** clients can transfer files from the server to a specified local directory, ensuring that files are transferred correctly and efficiently.
Acceptance Criteria	• **Server**: 1. It must be able to receive client requests via TCP. 2. It must be launched by passing the listening port and the local directory containing files that clients can request via the command line. 3. It should allow access only to files located in the specified directory or subdirectories. 4. It must be able to transfer files of any size, sending data blocks with a maximum size (e.g., MAX_DATA = 4000 bytes), with the last block being a UDP datagram with a content size of zero bytes. 5. It should signal the completion of the transfer by closing the TCP connection. • **Client:** 1. It must be launched by passing the server's location, the desired file name, and the local directory where the file should be stored via the command line. 2. It must send a datagram to the server with the desired file name. 3. It must be able to receive and store the file in the specified local directory. 4. Any issue during the transfer, including timeout, should lead to aborting the operation.

Technical Requirements	• The application must be developed in Java. • Communication between client and server must be based on the TCP protocol. • It should be possible to transfer files of any size, with a maximum block size of 4000 bytes. • The application must correctly handle issues during the transfer, aborting the operation in case of error.

11.

Title	Implement Peer-to-Peer Application for Message Exchange via Multicast
Description	**As an** developer **I want to** create a peer-to-peer application that allows message exchange within a group of users using multicast IP addresses **So that** I can communicate with other users in the chat group efficiently.
Acceptance Criteria	**Application launch**: 1. The application must be launched by passing the user's name, the multicast group address, and the desired listening port via the command line. **Message sending and receiving**: 1. The application must continuously wait for messages entered via standard input. 2. The entered messages must be sent to the chat group via multicast. 3. Messages received via the network must be displayed on standard output. 4. Messages must be exchanged in the form of strings. **Special command "LIST"**: 1. Upon receiving a message containing the sequence "LIST", the application must resend the local user's name to the origin of the message.
Technical Requirements	• The application must be developed in Java. • Communication should use multicast IP addresses (Class D).