**Project Report: Chat Application using Sockets in Java**

**Project Title:** **Chat Application using Sockets in Java**

**Student Name: Sariga Rajam S**

**Department: B.E-CSE-Artificial Intelligence & Machine Learning**

**Project Link: https://github.com/sariga17/Chat-application.git**

**Duration of the Project:**

* **Start Date**: 27-9-2024
* **End Date**: 3-10-2024

**Project Overview:**

This project is a simple chat application built using Java sockets. It demonstrates the fundamentals of network programming, allowing communication between a client and a server over a TCP connection. The application supports bidirectional message exchange, where both the client and the server can send and receive messages in real-time.

**Key Components:**

1. **ChatClient Class**:
   * **Functionality**:
     + The ChatClient class connects to the server using the provided server address and port.
     + The client reads user input from the console and sends it to the server, while simultaneously receiving messages from the server.
   * **Core Methods**:
     + Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT): Establishes the connection to the server.
     + BufferedReader input: Reads incoming messages from the server.
     + PrintWriter output: Sends messages to the server.
     + A separate thread is used to handle incoming server messages asynchronously.
2. **ChatServer Class**:
   * **Functionality**:
     + The ChatServer class listens for client connections on a specified port.
     + Once a client connects, the server can send and receive messages from the client.
   * **Core Methods**:
     + ServerSocket serverSocket = new ServerSocket(PORT): Opens a port and listens for incoming connections.
     + BufferedReader input: Reads incoming messages from the client.
     + PrintWriter output: Sends messages to the client.
     + The server also handles incoming client messages asynchronously using a thread.

**Project Workflow:**

1. **Client-Server Communication**:
   * The server listens on a specified port (in this case, 1234).
   * The client connects to the server using the localhost address and the same port.
   * Both the server and the client can send and receive messages in real-time using sockets and input/output streams.
2. **Multithreading**:
   * Both the client and the server utilize multithreading to allow non-blocking message reception, ensuring smooth communication while both parties can continue typing.

**Technological Infrastructure:**

* **Language**: Java
* **Libraries Used**:
  + java.io.\* for handling input and output streams.
  + java.net.\* for socket communication.
* **IDE**: CMD prompt
* **Development Environment**: Windows

**Challenges:**

* Handling asynchronous communication between the client and server.
* Ensuring that both the client and server maintain connection stability while sending and receiving data.
* Managing input/output streams for concurrent reading and writing of messages.

**Future Enhancements:**

* **Multi-Client Support**: Extend the server to support multiple clients connecting simultaneously.
* **GUI Integration**: Develop a graphical user interface for better user experience instead of console-based input/output.
* **Encryption**: Implement message encryption to enhance security during communication.

**Conclusion:**

This project demonstrates the core concepts of socket programming in Java, enabling real-time communication between a client and server. It provides a foundation for building more complex network-based applications, such as chat applications with multiple clients or adding encryption for secure communication.

**Program Output**





