Mini Project Report

On

Social Networking System

Problem Statement:

In the digital age, social networking platforms have become an integral part of people's lives. The problem at hand is to develop a basic social networking system that includes a server and a client for simulating user interactions.

Abstract:

The provided codes consist of a Social Network Server and a Social Network Client. The server facilitates user registration, friend requests, messaging, and message retrieval, while the client allows users to interact with the server through a text-based menu. This system emulates a basic social networking platform where users can perform actions like creating an account, connecting with friends, sending messages, and viewing their messages. The client connects to the server via sockets, enabling communication between the two.

This system aims to provide the following key functionalities:

User Registration:

- Users can create new accounts by providing a unique username.
- The system assigns each user a unique user ID.
- The server stores user data, including usernames, friends, friend requests, and messages.

Friendship Management:

- Users can send friend requests to other users by specifying the recipient's user ID.
- Users can accept friend requests from other users.
- The system tracks the friendships between users and updates their friend lists accordingly.

Messaging:

- Users who are friends can send text messages to each other.
- Messages are timestamped and stored for retrieval.

Message Retrieval:

- Users can retrieve their messages.

- The system provides users with their messages, including sender information and message content.

Client Interaction:

- A client application allows users to interact with the server.
- Users can perform the above actions through a menu-driven interface.

This social networking system is designed as a simplified prototype and does not include advanced features or security measures commonly found in real-world social networking platforms. The problem statement assumes that the primary goal is to demonstrate basic server-client communication and interaction for social networking purposes. In a real-world application, additional features such as user profiles, privacy settings, multimedia support, and enhanced security would be necessary.

Data Structures used in this Project:

In the project (SocialServer and SocialClient) for the simple social networking system, several data structures are used to manage and store various pieces of information. list of data structures:

User Class (In SocialServer):

This is a custom class that represents a user in the social network.

Attributes:

- int userId: A unique identifier for each user.
- String username: The username chosen by the user.
- Set<User> friends: A set of User objects representing the user's friends.
- Set<User> friendRequests: A set of User objects representing friend requests.
- List<Message> messages: A list of Message objects representing the messages sent and received by the user.

Message Class (In SocialServer):

Another custom class to represent a message.

Attributes:

- String sender: The username of the sender.
- String content: The content of the message.
- Date timestamp: A timestamp for when the message was created.

SocialServer Class:

The main class representing the social networking server.

Data Structures:

- Map<Integer, User> users: A HashMap that stores user objects with their unique userId as keys. It allows efficient retrieval and management of user data.
- AtomicInteger userIdCounter: An AtomicInteger used to generate unique user IDs.

ClientHandler Class (In SocialServer):

A class responsible for handling client requests.

Data Structures:

- SocialServer socialNetwork: A reference to the main SocialServer instance.
- Socket clientSocket: The socket for communication with the client.
- PrintWriter out and BufferedReader in: Input and output streams for communicating with the client.

SocialClient Class:

The main class representing the social networking client application.

Data Structures:

- int userId: A variable to store the current user's ID once registered.

The data structures used in these programs are essential for managing users, their friendships, messages, and the client-server communication. Here's a brief explanation of their purposes:

- Map and Set data structures are used for efficient storage and retrieval of user information and relationships (friends and friend requests).
- List is used to maintain a chronological order of messages.
- AtomicInteger is used to ensure unique user IDs are generated during user registration.
- In the client application, simple variables (int userId, String username) are used to temporarily store user input and the current user's state.
- These data structures collectively allow the server to manage user interactions, friendships, and messaging within the social network, and the client to facilitate user interaction with the server.

Steps for executing

Assuming you have Java installed on your system:

- **Create Files:**
  Copy and paste the code for Server.java, Client.java, and ClientHandler.java into three separate files with their respective names.

- **Open Command Prompt:**
  Open a command prompt or terminal window on your system.

- **Navigate to the Directory**:
  Use the cd command to navigate to the directory where you saved your Java files. For example:
  cd path/to/your/directory

- **Compile the Code:**
  Compile the Java files using javac:
  javac Server.java
  javac Client.java

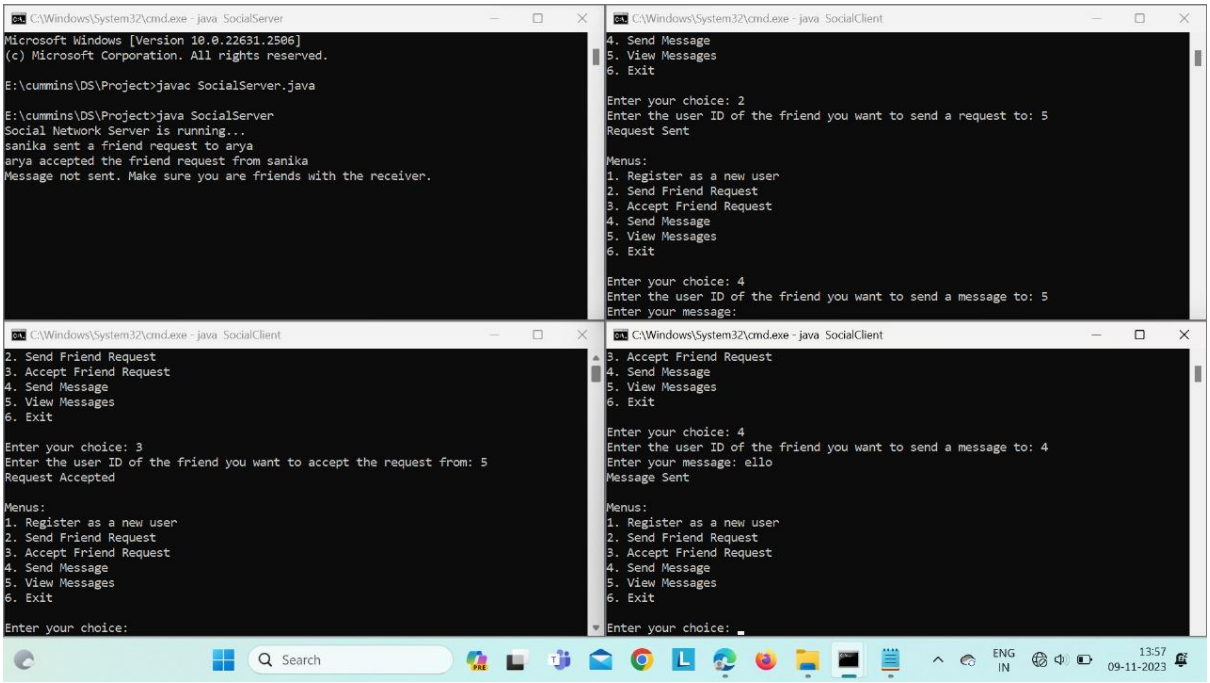
- **Run the Server:**
  Start the server by executing the following command in the command prompt:
  java Server
  The server will start and wait for client connections.

- **Run Multiple Clients:**
  Open new command prompt windows for each client you want to simulate.
  In each client window, run the client program:
  java Client
  You can run multiple client instances simultaneously.

- **Observe Output:**
  In the server command prompt, you should see messages indicating that clients have connected.
  In each client command prompt, you should see messages indicating the server's response.
  Terminate Programs:

  To stop the server and clients, you can usually use Ctrl+C in the respective command prompt windows.

Output:



Group Members:
1.Arya Patil – UIT2023001
2.Sanika Paware – UIT2023006
3.Pooja Shirsat – UIT2023003
4.Swamini Bhnad – UIT2023004