|  |  |
| --- | --- |
| **Name** | Mayur Solankar, Manish Jadhav, Vishesh Savani, Shreyansh Salvi |
| **UID** | 2023301018, 2023301005, 2022300100, 2022300091 |
| **Subject** | Distributed Computing |
| **Experiment No.** | 3 |
| **Project title** | Social Media System |
| **Problem Statement** | Implementation of Multithreading in Distributed System |
| **Objectives** | The objective is to use multithreading to increase the real-time interactivity and responsiveness of social media platforms for actions like chatting, liking, commenting, and posting. |
| **Theory** | **What is Multithreading?**  A distributed system involves many computers or nodes working together to achieve a common goal. Multithreading means that several threads are executing simultaneously in one process. The social media chat feature allows for real time messaging and the most readable messages. Each conversation runs on a separate thread, allowing multiple users to send and receive messages simultaneously. Social media platforms manage various user interactions such as liking, commenting, sharing and responding to posts. Allowing users to see updates in their posts and comments. Notifications must be sent immediately when a user receives likes, comments or messages. It ensures that notifications are delivered without delay. Users upload photos and videos to share with others.  **Benefits of Multithreading in Distributed Systems Concurrency: Multithreading** allows a distributed system to execute multiple tasks concurrently, improving overall system efficiency and performance. Multithreading enables distributed systems to remain responsive to external requests and events by handling tasks in parallel. |
| **Code:** | 1. **Client Side:**   import socket  import threading  class User:  def \_init\_(self, username):  self.username = username  self.friends = []  def add\_friend(self, friend\_username):  self.friends.append(friend\_username)  class SocialMediaPlatform:  def \_init\_(self):  self.posts = []  self.users = {}  self.lock = threading.Lock()  def add\_user(self, username):  if username not in self.users:  self.users[username] = User(username)  return f"User '{username}' created successfully."  return f"User '{username}' already exists."  def add\_friend(self, username, friend\_username):  user = self.users.get(username)  friend = self.users.get(friend\_username)  if user and friend:  user.add\_friend(friend\_username)  friend.add\_friend(username)  return f"Friend '{friend\_username}' added to '{username}'."  return "User or friend not found."  def post(self, username, message):  with self.lock:  post\_index = len(self.posts)  post = f"{username}: {message}"  self.posts.append(post)  return f"Post created successfully. Index: {post\_index}\nPost Content: {post}"  def comment(self, username, post\_index, comment):  with self.lock:  if 0 <= post\_index < len(self.posts):  original\_post = self.posts[post\_index]  updated\_post = f"{original\_post}\n- Comment by {username}: {comment}"  self.posts[post\_index] = updated\_post  return f"Comment added successfully.\nComment Content: - Comment by {username}: {comment}"  return "Invalid post index."  def get\_posts(self):  return self.posts  def handle\_client(client\_socket, platform):  while True:  data = client\_socket.recv(1024).decode()  if not data:  break  # Parse the client's request  parts = data.split()  response = ""  if parts[0] == '1':  # Create User  username = parts[1]  response = platform.add\_user(username)  elif parts[0] == '2':  # Add Friend  username = parts[1]  friend\_username = parts[2]  response = platform.add\_friend(username, friend\_username)  elif parts[0] == '3':  # Make a Post  username = parts[1]  message = ' '.join(parts[2:])  response = platform.post(username, message)  elif parts[0] == '4':  # Comment on a Post  username = parts[1]  post\_index = int(parts[2])  comment = ' '.join(parts[3:])  response = platform.comment(username, post\_index, comment)  elif parts[0] == '5':  # Exit  # Send all posts and summary  posts = platform.get\_posts()  num\_posts = len(posts)  num\_comments = sum(post.count('\n- Comment by ') for post in posts)  response = f"Posts Summary:\nNumber of posts: {num\_posts}\nNumber of comments: {num\_comments}\n"  for post in posts:  response += f"{post}\n"  client\_socket.send(response.encode())  break  else:  response = "Invalid choice."  client\_socket.send(response.encode())  client\_socket.close()  def main():  server\_host = '192.168.0.103' # Use the IP address of 'enp3s0' interface  server\_port = 12345  platform = SocialMediaPlatform()  server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  server\_socket.bind((server\_host, server\_port))  server\_socket.listen(5) # Listen for up to 5 incoming connections  print(f"Server is listening on {server\_host}:{server\_port}")  while True:  client\_socket, addr = server\_socket.accept()  client\_handler = threading.Thread(target=handle\_client, args=(client\_socket, platform))  client\_handler.start()  main()   1. **Server Side:**   import socket  def main():  server\_host = '192.168.0.103' # Use the IP address of 'enp3s0' interface  server\_port = 12345  # Set up a socket to communicate with the server  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  client\_socket.connect((server\_host, server\_port))  while True:  print("1. Create User")  print("2. Add Friend")  print("3. Make a Post")  print("4. Comment on a Post")  print("5. Exit")    choice = int(input("Enter your choice: "))    if choice == 1:  username = input("Enter your username: ")  client\_socket.send(f"1 {username}".encode())    elif choice == 2:  username = input("Enter your username: ")  friend\_username = input("Enter your friend's username: ")  client\_socket.send(f"2 {username} {friend\_username}".encode())    elif choice == 3:  username = input("Enter your username: ")  message = input("Enter your post: ")  client\_socket.send(f"3 {username} {message}".encode())    elif choice == 4:  username = input("Enter your username: ")  post\_index = int(input("Enter the post index you want to comment on: "))  comment = input("Enter your comment: ")  client\_socket.send(f"4 {username} {post\_index} {comment}".encode())    elif choice == 5:  client\_socket.send("5".encode())  # Receive and print all posts and summary  data = client\_socket.recv(4096).decode()  print("Server Response:")  print(data)  break    else:  print("Invalid choice. Please try again.")  # Close the client socket  client\_socket.close()  main() |
| **Output:** |  |
| **Conclusion:** | Hence by completing we came to about implementation of Multithreading in Distributed Systems. |