BHARATIYA VIDYA BHAVAN'S
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Empowered Autonomous Institute Affiliated to Mumbai University)
Department Of Computer Engineering

Name	Mayur Solankar, Manish Jadhav, Vishesh Savani, Shreyansh Salvi
UID	2023301018, 2023301005, 2022300100, 2022300091
Subject	Distributed Computing
Experiment No.	6
Project title	Social Media System
Problem Statement	To implement mutual exclusion algorithm
Objectives	Ensures only one user or process accesses a shared resource at a time, preventing conflicts and maintaining data consistency in a social media platform.
Theory	**MUTUAL EXCLUSION** What is a Mutual Exclusion ? In the context of a social media platform, mutual exclusion refers to a synchronization mechanism that ensures only one user or process can access and modify shared resources, such as posts, comments, or user profiles, at a given time. This prevents conflicts and inconsistencies that might arise if multiple users attempt to modify the same data simultaneously. By using mutual exclusion, the social media platform maintains data integrity, preserves the order of user interactions, and prevents race conditions, ensuring a seamless and coherent user experience. Mutual exclusion is a fundamental problem in distributed computing systems. Mutual exclusion ensures that concurrent access of processes to a shared resource or data is serialized, that is, executed in a mutually exclusive manner. Mutual exclusion in a distributed system states that only one process is allowed to execute the critical section (CS) at any given time. In a distributed system, shared variables (semaphores) or a local kernel cannot be used to implement mutual exclusion. Message passing is the sole means for implementing distributed mutual exclusion. The decision as to which process is allowed access to the CS next is arrived at by message passing, in which each process learns about the state of all other processes in some consistent way. The design of distributed mutual exclusion algorithms is complex because these algorithms must deal with unpredictable message delays and incomplete knowledge of the system state. There are three basic approaches for implementing distributed mutual exclusion:

1. Token-based approach.
2. Non-token-based approach.
3. Quorum-based approach.

In the token-based approach, a unique token (also known as the PRIVILEGE message) is shared among the sites. A site is allowed to enter its CS if it possesses the token, and it continues to hold the token until the execution of the CS is over. Mutual exclusion is ensured because the token is unique.

Requirements of Mutual exclusion Algorithm:
•No Deadlock: Two or more sites should not endlessly wait for any message that will never arrive.
• No Starvation: Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site are repeatedly executing critical section
•Fairness: Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e. Critical section execution requests should be executed in the order of their arrival in the system.
•Fault Tolerance: In case of failure, it should be able to recognize it by itself to continue functioning without any disruption.

Code

Server Code :-

```
import socket
import threading

# Data for social media posts (post_id: [post_content, likes, comments])
posts = {
    1: ["Post 1: Hello World!", 0, []],
    2: ["Post 2: Python is great!", 0, []]
}

# Lock for mutual exclusion
server_lock = threading.Lock()


def handle_client(client_socket, client_address):
    print(f"Client {client_address} connected.")

    while True:
        try:
            message = client_socket.recv(1024).decode()
            if not message:
```

```
                break

            msg_parts = message.split(":")
            action = msg_parts[0]
            client_id = msg_parts[1]
            clock = msg_parts[2]

            if action == "comment":
                post_id = int(msg_parts[3])
                comment = msg_parts[4]
                with server_lock:  # Mutual exclusion for adding a comment
                    posts[post_id][2].append(f"Client {client_id}:
{comment}")
                response_message = f"Comment added to Post {post_id}:
{comment}"
                client_socket.send(response_message.encode())

            elif action == "like":
                post_id = int(msg_parts[3])
                with server_lock:  # Mutual exclusion for liking a post
                    posts[post_id][1] += 1
                response_message = f"Post {post_id} liked by Client
{client_id}"
                client_socket.send(response_message.encode())

            elif action == "create_post":
                content = msg_parts[3]
                new_post_id = max(posts.keys()) + 1
                with server_lock:  # Mutual exclusion for creating a post
                    posts[new_post_id] = [content, 0, []]
                response_message = f"Client {client_id} created a new post:
{content}"
                client_socket.send(response_message.encode())

            elif action == "view_posts":
                with server_lock:
                    response_message = "Current Posts:\n"
                    for post_id, post_data in posts.items():
                        response_message += f"Post {post_id}: {post_data[0]}
| Likes: {post_data[1]} | Comments: {post_data[2]}\n"
                client_socket.send(response_message.encode())

            elif action == "exit":
                print(f"Client {client_id} has exited.")
```

```
                break

        except Exception as e:
            print(f"Error with client {client_address}: {e}")
            break

    client_socket.close()
    print(f"Client {client_address} disconnected.")

# Start the server
def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(("0.0.0.0", 8080))
    server_socket.listen(5)
    print("Server is listening on port 8080...")

    while True:
        client_socket, client_address = server_socket.accept()
        client_thread = threading.Thread(target=handle_client,
args=(client_socket, client_address))
        client_thread.start()

if __name__ == "__main__":
    start_server()
```

Client 1

```
import socket

logical_clock = 0
my_id = 1  # Client ID (should be unique per client)

def increment_clock():
    global logical_clock
    logical_clock += 1

def send_request_to_server(action, post_id=None, content=None):
    global logical_clock, my_id
    server_address = "127.0.0.1"
    server_port = 8080

    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((server_address, server_port))

    increment_clock()
```

```
    if action == "comment":
        comment_message =
f"comment:{my_id}:{logical_clock}:{post_id}:{content}"
        client.send(comment_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "like":
        like_message = f"like:{my_id}:{logical_clock}:{post_id}"
        client.send(like_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "create_post":
        create_message = f"create_post:{my_id}:{logical_clock}:{content}"
        client.send(create_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "view_posts":
        view_message = f"view_posts:{my_id}:{logical_clock}"
        client.send(view_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "exit":
        exit_message = f"exit:{my_id}:{logical_clock}"
        client.send(exit_message.encode())
        client.close()
        return

    client.close()

def client_run():
    while True:
        print("\nSocial Media Activities")
        print("1. Comment on a friend's post")
        print("2. Like a friend's post")
        print("3. Create a post")
        print("4. View all posts")
        print("5. Exit")
        choice = input("Enter your choice: ")
```

```
        if choice == "1":
            post_id = input("Enter post ID to comment on: ")
            comment = input("Enter your comment: ")
            send_request_to_server(action="comment", post_id=post_id,
content=comment)
        elif choice == "2":
            post_id = input("Enter post ID to like: ")
            send_request_to_server(action="like", post_id=post_id)
        elif choice == "3":
            content = input("Enter your post content: ")
            send_request_to_server(action="create_post", content=content)
        elif choice == "4":
            send_request_to_server(action="view_posts")
        elif choice == "5":
            send_request_to_server(action="exit")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    client_run()
```

Client 2

```
import socket

logical_clock = 0
my_id = 2  # Client ID (should be unique per client)

def increment_clock():
    global logical_clock
    logical_clock += 1

def send_request_to_server(action, post_id=None, content=None):
    global logical_clock, my_id
    server_address = "127.0.0.1"
    server_port = 8080

    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect((server_address, server_port))

    increment_clock()

    if action == "comment":
        comment_message =
f"comment:{my_id}:{logical_clock}:{post_id}:{content}"
```

```
        client.send(comment_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "like":
        like_message = f"like:{my_id}:{logical_clock}:{post_id}"
        client.send(like_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "create_post":
        create_message = f"create_post:{my_id}:{logical_clock}:{content}"
        client.send(create_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "view_posts":
        view_message = f"view_posts:{my_id}:{logical_clock}"
        client.send(view_message.encode())
        response = client.recv(1024).decode()
        print(response)

    elif action == "exit":
        exit_message = f"exit:{my_id}:{logical_clock}"
        client.send(exit_message.encode())
        client.close()
        return

    client.close()

def client_run():
    while True:
        print("\nSocial Media Activities")
        print("1. Comment on a friend's post")
        print("2. Like a friend's post")
        print("3. Create a post")
        print("4. View all posts")
        print("5. Exit")
        choice = input("Enter your choice: ")

        if choice == "1":
            post_id = input("Enter post ID to comment on: ")
            comment = input("Enter your comment: ")
```

```
            send_request_to_server(action="comment", post_id=post_id,
content=comment)
        elif choice == "2":
            post_id = input("Enter post ID to like: ")
            send_request_to_server(action="like", post_id=post_id)
        elif choice == "3":
            content = input("Enter your post content: ")
            send_request_to_server(action="create_post", content=content)
        elif choice == "4":
            send_request_to_server(action="view_posts")
        elif choice == "5":
            send_request_to_server(action="exit")
            break
        else:
            print("Invalid choice. Try again.")

if __name__ == "__main__":
    client_run()
```

Output:

Server Side

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\Experiment 6> python mutual_exclusion_server.py
Server is listening on port 8080...
Client ('127.0.0.1', 49737) connected.
Client ('127.0.0.1', 49737) disconnected.
Client ('127.0.0.1', 49738) connected.
Client ('127.0.0.1', 49738) disconnected.
Client ('127.0.0.1', 49749) connected.
Client ('127.0.0.1', 49749) disconnected.
Client ('127.0.0.1', 49750) connected.
Client ('127.0.0.1', 49750) disconnected.
Client ('127.0.0.1', 49751) connected.
Client ('127.0.0.1', 49751) disconnected.
Client ('127.0.0.1', 49752) connected.
Client ('127.0.0.1', 49752) disconnected.
```

Client 1

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd '.\Experiment 6\'
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\Experiment 6> python mutual_exclusion_client.py

Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 3
Enter your post content: hi guys myself vishesh
Client 1 created a new post: hi guys myself vishesh

Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 4
Current Posts:
Post 1: Post 1: Hello World! | Likes: 0 | Comments: []
Post 2: Post 2: Python is great! | Likes: 0 | Comments: []
Post 3: hi guys myself vishesh | Likes: 0 | Comments: []


Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 5
```

Client 2

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\Experiment 6> python client2.py

Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 2
Enter post ID to like: 1
Post 1 liked by Client 2

Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 4
Current Posts:
Post 1: Post 1: Hello World! | Likes: 1 | Comments: []
Post 2: Post 2: Python is great! | Likes: 0 | Comments: []
Post 3: hi guys myself vishesh | Likes: 0 | Comments: []


Social Media Activities
1. Comment on a friend's post
2. Like a friend's post
3. Create a post
4. View all posts
5. Exit
Enter your choice: 2
Enter post ID to like: 3
```

Conclusion:

Hence by completing we came to about implementation of mutual exclusion in social media platform.