Name	Mayur Solankar, Manish Jadhav, Vishesh Savani, Shreyansh Salvi
UID	2023301018, 2023301005, 2022300100, 2022300091
Subject	Distributed Computing
Experiment No.	7
Project title	Social Media System
Problem Statement	To implement clock synchronization using Lamport's Logical Clock Algorithm.
Objectives	The objective of Lamport's logical clock algorithm in a social media platform is to provide a consistent and ordered timestamping mechanism for posts and comments, ensuring accurate sequencing in a distributed environment.
Theory	**What are Lamport clocks ?** Lamport clocks represent time logically in a distributed system. They are also known as logical clocks. The idea behind Lamport clocks is to disregard physical time and capture just a "happens-before" relationship between a pair of events. **Why use Lamport clocks ?** Time synchronization is a key problem in distributed systems. Time is used to order events across servers. Using physical clocks to order events is challenging because real synchronization is impossible and clocks experience skew. A clock skew is when different clocks run at different rates, so we cannot assume that time t on node a happened before time t + 1 on node b. Instead of employing physical time, Leslie Lamport proposed logical clocks that capture events' orderings through a "happens-before" relationship. Implementing Lamport's logical clock algorithm in a social media platform is necessary for maintaining order of events in a distributed environment. In the context of a social media platform, users constantly create posts and comments. Lamport's logical clocks allow these events to be time stamped consistently, ensuring that the order in which they occurred is accurate. When a user makes a post or a comment, Lamport's algorithm assigns a logical timestamp to the event, representing its position in the sequence of events across the platform. This logical timestamp confirms that events are related in the correct order, regardless of the physical time they occurred.

Code	Server

Server

```python
import socket
import threading

class LamportClock:
    def __init__(self):
        self.timestamp = 0

    def tick(self):
        self.timestamp += 1

    def update(self, received_time):
        self.timestamp = max(self.timestamp, received_time) + 1

class Server:
    def __init__(self):
        self.clock = LamportClock()
        self.users = {}
        self.posts = []
        self.comments = {}
        self.likes = {}

    def log_activity(self, activity, details=""):
        """Log server activity along with the Lamport timestamp."""
        print(f"[Lamport Timestamp: {self.clock.timestamp}] {activity}
{details}")

    def handle_request(self, data):
        # Increment Lamport timestamp on each request
        self.clock.tick()

        parts = data.split(':')
        command = parts[0].strip()

        # Handling different commands
        if command == "CREATE_USER":
            username = parts[1].strip()
            if username not in self.users:
                self.users[username] = []
                response = f"User '{username}' created successfully!"
            else:
                response = f"User '{username}' already exists."
            self.log_activity("CREATE_USER", f"User: {username}")
```

```python
        elif command == "POST":
            username = parts[1].strip()
            post_content = parts[2].strip()
            post_id = len(self.posts) + 1
            self.posts.append((post_id, username, post_content))
            self.comments[post_id] = []
            self.likes[post_id] = []
            response = f"Post created with ID {post_id} by '{username}'!"
            self.log_activity("POST", f"Post ID: {post_id}, User: {username},
Content: {post_content}")

        elif command == "COMMENT":
            username = parts[1].strip()
            post_id = int(parts[2].strip())
            comment_content = parts[3].strip()
            if post_id in self.comments:
                self.comments[post_id].append((username, comment_content))
                response = f"Comment added to Post ID {post_id} by
'{username}'."
                self.log_activity("COMMENT", f"Post ID: {post_id}, User:
{username}, Comment: {comment_content}")
            else:
                response = f"Post ID {post_id} does not exist."
                self.log_activity("COMMENT_FAIL", f"Post ID: {post_id} not
found")

        elif command == "LIKE":
            username = parts[1].strip()
            post_id = int(parts[2].strip())
            if post_id in self.likes:
                if username not in self.likes[post_id]:
                    self.likes[post_id].append(username)
                    response = f"'{username}' liked Post ID {post_id}."
                    self.log_activity("LIKE", f"Post ID: {post_id}, User:
{username}")
                else:
                    response = f"'{username}' has already liked Post ID
{post_id}."
                    self.log_activity("LIKE_FAIL", f"Post ID: {post_id},
User: {username} already liked")
            else:
                response = f"Post ID {post_id} does not exist."
                self.log_activity("LIKE_FAIL", f"Post ID: {post_id} not
found")
```

```python
        elif command == "VIEW_ALL_POSTS":
            if self.posts:
                response = "All Posts:\n"
                for post_id, username, content in self.posts:
                    response += f"Post ID: {post_id}, User: {username}, 
Content: {content}\n"
                self.log_activity("VIEW_ALL_POSTS", "Displayed all posts")
            else:
                response = "No posts available."
                self.log_activity("VIEW_ALL_POSTS", "No posts available")

        elif command == "VIEW_POST_DETAILS":
            post_id = int(parts[1].strip())
            if post_id in [post[0] for post in self.posts]:  # Check if post
ID exists
                post = next(post for post in self.posts if post[0] ==
post_id)
                response = f"Details of Post ID {post_id}:\n"
                response += f"Content: {post[2]}\n"
                response += f"Likes: {len(self.likes[post_id])}\n"
                response += "Liked by: " + ", ".join(self.likes[post_id]) +
"\n"
                response += "Comments:\n"
                for username, comment in self.comments[post_id]:
                    response += f" - {username}: {comment}\n"
                self.log_activity("VIEW_POST_DETAILS", f"Post ID: {post_id}
details viewed")
            else:
                response = f"Post ID {post_id} does not exist."
                self.log_activity("VIEW_POST_DETAILS_FAIL", f"Post ID
{post_id} not found")

        else:
            response = "Unknown command."
            self.log_activity("UNKNOWN_COMMAND", data)

        return response

def handle_client(conn, addr, server):
    """Handle client connection and request processing."""
    print(f"New connection from {addr}")

    with conn:
```
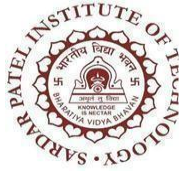
```python
            data = conn.recv(1024).decode('utf-8')
            if data:
                response = server.handle_request(data)
                conn.send(response.encode('utf-8'))
                print(f"Response sent to {addr}")

def main():
    server = Server()
    host = '127.0.0.1'
    port = 12345

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen()
        print("Server is listening...")

        while True:
            conn, addr = s.accept()
            threading.Thread(target=handle_client, args=(conn, addr,
server)).start()

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

Client Side

```python
import socket
import time

class Client:
    def __init__(self):
        self.host = '127.0.0.1'
        self.port = 12345

    def send_event(self, event):
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.connect((self.host, self.port))
            s.send(event.encode('utf-8'))
            data = s.recv(1024).decode('utf-8')
            print(f"\nServer Response:\n{data}")

    def create_user(self, username):
        event = f"CREATE_USER:{username}"
        self.send_event(event)
```

```python
    def post(self, username, content):
        event = f"POST:{username}:{content}"
        self.send_event(event)

    def comment(self, username, post_id, content):
        event = f"COMMENT:{username}:{post_id}:{content}"
        self.send_event(event)

    def like_post(self, username, post_id):
        event = f"LIKE:{username}:{post_id}"
        self.send_event(event)

    def view_all_posts(self):
        event = "VIEW_ALL_POSTS"
        self.send_event(event)

    def view_post_details(self, post_id):
        event = f"VIEW_POST_DETAILS:{post_id}"
        self.send_event(event)


def main():
    client = Client()

    # User registration
    user1 = input("\nEnter your username: ")
    client.create_user(user1)

    while True:
        # Menu for user actions
        print("\n--- Social Media Actions ---")
        print("1. Create a post")
        print("2. Comment on a post")
        print("3. Like a post")
        print("4. View all posts")
        print("5. View details of a specific post (comments & likes)")
        print("6. Exit")
        choice = input("Select an option (1-6): ")

        if choice == '1':
            post_content = input("Enter your post content: ")
            client.post(user1, post_content)
            time.sleep(1)

        elif choice == '2':
```
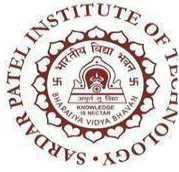
```python
            post_id = input("Enter the post ID to comment on: ")
            comment_content = input("Enter your comment: ")
            client.comment(user1, post_id, comment_content)
            time.sleep(1)

        elif choice == '3':
            post_id = input("Enter the post ID to like: ")
            client.like_post(user1, post_id)
            time.sleep(1)

        elif choice == '4':
            print("Fetching all posts...")
            client.view_all_posts()
            time.sleep(1)

        elif choice == '5':
            post_id = input("Enter the post ID to view details (comments &
likes): ")
            print(f"Fetching details for Post ID: {post_id}")
            client.view_post_details(post_id)
            time.sleep(1)

        elif choice == '6':
            print("Exiting the application. Goodbye!")
            break

        else:
            print("Invalid choice, please select a valid option.")


if __name__ == "__main__":
    main()
```
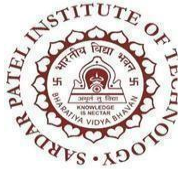
Output:	**Server Side**

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd .\8_Experiment\
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\8_Experiment> python server.py
Server is listening...
New connection from ('127.0.0.1', 49849)
[Lamport Timestamp: 1] CREATE_USER User: vishesh
Response sent to ('127.0.0.1', 49849)
New connection from ('127.0.0.1', 49850)
[Lamport Timestamp: 2] CREATE_USER User: manish
Response sent to ('127.0.0.1', 49850)
New connection from ('127.0.0.1', 49851)
[Lamport Timestamp: 3] POST Post ID: 1, User: vishesh, Content: hi guys myself vishesh from 3 year of engineer
ing
Response sent to ('127.0.0.1', 49851)
New connection from ('127.0.0.1', 49852)
[Lamport Timestamp: 4] COMMENT Post ID: 1, User: manish, Comment: hi vishesh, manish this side
Response sent to ('127.0.0.1', 49852)
New connection from ('127.0.0.1', 49854)
[Lamport Timestamp: 5] POST Post ID: 2, User: manish, Content: I thrilled to announce I solved 1000 problems o
n leetcode
Response sent to ('127.0.0.1', 49854)
New connection from ('127.0.0.1', 49856)
[Lamport Timestamp: 6] LIKE Post ID: 2, User: vishesh
Response sent to ('127.0.0.1', 49856)
New connection from ('127.0.0.1', 49857)
[Lamport Timestamp: 7] VIEW_POST_DETAILS Post ID: 1 details viewed
Response sent to ('127.0.0.1', 49857)
New connection from ('127.0.0.1', 49858)
[Lamport Timestamp: 8] VIEW_POST_DETAILS Post ID: 2 details viewed
Response sent to ('127.0.0.1', 49858)
New connection from ('127.0.0.1', 49859)
[Lamport Timestamp: 9] VIEW_ALL_POSTS Displayed all posts
Response sent to ('127.0.0.1', 49859)
New connection from ('127.0.0.1', 49861)
```

```
Response sent to ('127.0.0.1', 49854)
New connection from ('127.0.0.1', 49856)
[Lamport Timestamp: 6] LIKE Post ID: 2, User: vishesh
Response sent to ('127.0.0.1', 49856)
New connection from ('127.0.0.1', 49857)
[Lamport Timestamp: 7] VIEW_POST_DETAILS Post ID: 1 details viewed
Response sent to ('127.0.0.1', 49857)
New connection from ('127.0.0.1', 49858)
[Lamport Timestamp: 8] VIEW_POST_DETAILS Post ID: 2 details viewed
Response sent to ('127.0.0.1', 49858)
New connection from ('127.0.0.1', 49859)
[Lamport Timestamp: 9] VIEW_ALL_POSTS Displayed all posts
Response sent to ('127.0.0.1', 49859)
New connection from ('127.0.0.1', 49861)
[Lamport Timestamp: 10] VIEW_ALL_POSTS Displayed all posts
Response sent to ('127.0.0.1', 49861)
```

Client 1

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd .\8_Experiment\
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\8_Experiment> python client.py

Enter your username: vishesh

Server Response:
User 'vishesh' created successfully!

--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 1
Enter your post content: hi guys myself vishesh from 3 year of engineering

Server Response:
Post created with ID 1 by 'vishesh'!

--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 3
Enter the post ID to like: 2

Server Response:
'vishesh' liked Post ID 2.
```

```
--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 5
Enter the post ID to view details (comments & likes): 1
Fetching details for Post ID: 1

Server Response:
Details of Post ID 1:
Content: hi guys myself vishesh from 3 year of engineering
Likes: 0
Liked by:
Comments:
 - manish: hi vishesh, manish this side


--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 4
Fetching all posts...

Server Response:
All Posts:
```

```
--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 5
Enter the post ID to view details (comments & likes): 1
Fetching details for Post ID: 1

Server Response:
Details of Post ID 1:
Content: hi guys myself vishesh from 3 year of engineering
Likes: 0
Liked by:
Comments:
 - manish: hi vishesh, manish this side


--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 4
Fetching all posts...

Server Response:
All Posts:
```

```
--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 4
Fetching all posts...

Server Response:
All Posts:
Post ID: 1, User: vishesh, Content: hi guys myself vishesh from 3 year of engineering
Post ID: 2, User: manish, Content: I thrilled to announce I solved 1000 problems on leetcode


--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 6
Exiting the application. Goodbye!
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\8_Experiment>
                                    Ln 139, Col 1    Spaces: 4    UTF-8    CR
```
**Client 2**

```
PS C:\Users\vishe\OneDrive\Desktop\DC Codes> cd .\8_Experiment\
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\8_Experiment> python client2.py

Enter your username: manish

Server Response:
User 'manish' created successfully!

--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 2
Enter the post ID to comment on: 1
Enter your comment: hi vishesh, manish this side

Server Response:
Comment added to Post ID 1 by 'manish'.

--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 1
Enter your post content: I thrilled to announce I solved 1000 problems on leetcode

Server Response:
```
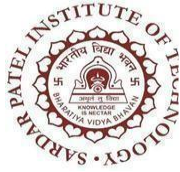
```
Post created with ID 2 by 'manish'!

--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 5
Enter the post ID to view details (comments & likes): 2
Fetching details for Post ID: 2

Server Response:
Details of Post ID 2:
Content: I thrilled to announce I solved 1000 problems on leetcode
Likes: 1
Liked by: vishesh
Comments:


--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 4
Fetching all posts...

Server Response:
All Posts:
```
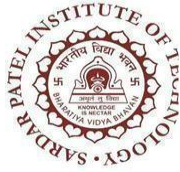
```
--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 4
Fetching all posts...

Server Response:
All Posts:
Post ID: 1, User: vishesh, Content: hi guys myself vishesh from 3 year of engineering
Post ID: 2, User: manish, Content: I thrilled to announce I solved 1000 problems on leetcode


--- Social Media Actions ---
1. Create a post
2. Comment on a post
3. Like a post
4. View all posts
5. View details of a specific post (comments & likes)
6. Exit
Select an option (1-6): 6
Exiting the application. Goodbye!
PS C:\Users\vishe\OneDrive\Desktop\DC Codes\8_Experiment> []
```

Conclusion:	In conclusion, implementing Lamport's logical clock algorithm in a social media platform ensures correct event sequencing, accurate tracking of user interactions and maintaining a consistent user experience in a distributed and concurrent environment.