Problem statement2: Event management System

CODE IN JAVA:

```
import java.util.*;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

// Event class that holds event details
class Event {
    private String title;
    private String description;
    private LocalDateTime dateTime;
    private List<User> attendees = new ArrayList<>();
    private boolean reminderSent = false;

    public Event(String title, String description, LocalDateTime dateTime) {
        this.title = title;
        this.description = description;
        this.dateTime = dateTime;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public LocalDateTime getDateTime() {
        return dateTime;
    }
```

```
    public List<User> getAttendees() {
        return attendees;
    }

    public boolean isReminderSent() {
        return reminderSent;
    }

    public void setReminderSent(boolean reminderSent) {
        this.reminderSent = reminderSent;
    }

    public void addAttendee(User user) {
        attendees.add(user);
        System.out.println("User " + user.getName() + " has RSVP'd to the event '" + title + "'.");
    }

    public void removeAttendee(User user) {
        attendees.remove(user);
        System.out.println("User " + user.getName() + " has been removed from the event '" + title +
"'.");
    }
}

// User class representing attendees
class User {
    private String name;
    private String email;

    public User(String name, String email) {
```

```
        this.name = name;
        this.email = email;
    }

    public String getName() {
        return name;
    }

    public String getEmail() {
        return email;
    }
}

// Event Manager class to manage events
class EventManager {
    private List<Event> events = new ArrayList<>();

    // Create a new event
    public void createEvent(String title, String description, LocalDateTime dateTime) {
        Event event = new Event(title, description, dateTime);
        events.add(event);
        System.out.println("Event '" + title + "' has been created.");
    }

    // List all events
    public void listEvents() {
        if (events.isEmpty()) {
            System.out.println("No events found.");
            return;
        }
        System.out.println("List of events:");
```

```
    for (Event event : events) {
        System.out.println("- " + event.getTitle() + " on " + event.getDateTime());
    }
}

// View event details
public Event viewEvent(String title) {
    for (Event event : events) {
        if (event.getTitle().equalsIgnoreCase(title)) {
            System.out.println("Event Details: ");
            System.out.println("Title: " + event.getTitle());
            System.out.println("Description: " + event.getDescription());
            System.out.println("Date: " + event.getDateTime());
            System.out.println("Attendees: " + event.getAttendees().size());
            return event;
        }
    }
    System.out.println("Event not found.");
    return null;
}

// Delete an event
public void deleteEvent(String title) {
    Event eventToRemove = null;
    for (Event event : events) {
        if (event.getTitle().equalsIgnoreCase(title)) {
            eventToRemove = event;
            break;
        }
    }
    if (eventToRemove != null) {
```

```
            events.remove(eventToRemove);
            System.out.println("Event '" + title + "' has been deleted.");
        } else {
            System.out.println("Event not found.");
        }
    }

    // Send a reminder for upcoming events
    public void sendReminders() {
        LocalDateTime now = LocalDateTime.now();
        for (Event event : events) {
            if (event.getDateTime().isAfter(now) && !event.isReminderSent()) {
                System.out.println("Reminder: Event '" + event.getTitle() + "' is scheduled for " +
event.getDateTime());
                event.setReminderSent(true);  // Assuming reminder is sent
            }
        }
    }

    // RSVP to an event
    public void rsvpToEvent(Event event, User user) {
        if (event != null) {
            event.addAttendee(user);
        }
    }

    // Manage attendees (list, add, remove)
    public void manageAttendees(Event event) {
        if (event == null) return;

        System.out.println("Attendees for event '" + event.getTitle() + "':");
```

```
        for (User attendee : event.getAttendees()) {
            System.out.println("- " + attendee.getName());
        }

        // Example: Removing an attendee
        if (!event.getAttendees().isEmpty()) {
            User userToRemove = event.getAttendees().get(0);  // For example, removing the first attendee
            event.removeAttendee(userToRemove);
        }
    }

    // Track user activity (Good to have)
    public void trackActivity(User user, String action) {
        System.out.println("Tracking activity: User '" + user.getName() + "' performed action: " + action);
    }
}

public class Main {
    public static void main(String[] args) {
        EventManager eventManager = new EventManager();

        // Create some users
        User user1 = new User("Alice", "alice@example.com");
        User user2 = new User("Bob", "bob@example.com");

        // Create an event
        LocalDateTime eventDateTime = LocalDateTime.of(2024, 9, 15, 10, 30);
        eventManager.createEvent("Tech Conference", "A conference on latest technology trends", eventDateTime);

        // List events
```

```
        eventManager.listEvents();

        // RSVP to the event
        Event techConference = eventManager.viewEvent("Tech Conference");
        eventManager.rsvpToEvent(techConference, user1);
        eventManager.rsvpToEvent(techConference, user2);

        // Manage attendees
        eventManager.manageAttendees(techConference);

        // Send reminders for upcoming events
        eventManager.sendReminders();

        // Track user activity (Good to have)
        eventManager.trackActivity(user1, "RSVP'd to Tech Conference");
    }
}
```