→ A meeting scheduler software allow organization to schedule & book meetings for a group of participants.

→ System allow users to book & Cancel meeting

→ Organizer can add new participants to a meeting

→ The scheduler determines a meeting time & Location depending upon the availability for a the participants.

Expectation from the interviewer

→ How does the system determine available rooms?

→ How important is the capacity of a room when assigning room for a meeting

→ How does the system check the availability of the attendees?

→ How does the system access the meeting information of all attendees?

Requirement Collection

R1: There should be n number of meeting Rooms

R2: Each meeting room should have a specific capacity to accomodate the desired number of people

R3: If not reserved already, each meeting room should have the ability to be booked, along with setting an interval, a start time, & end time of meeting

R4: A Notification should be sent to all the people invited to the meeting

R5: Users will receive an invite regarding of whether they are available at the interval or not. Users can respond to the invitation by either accepting or rejecting the invite.

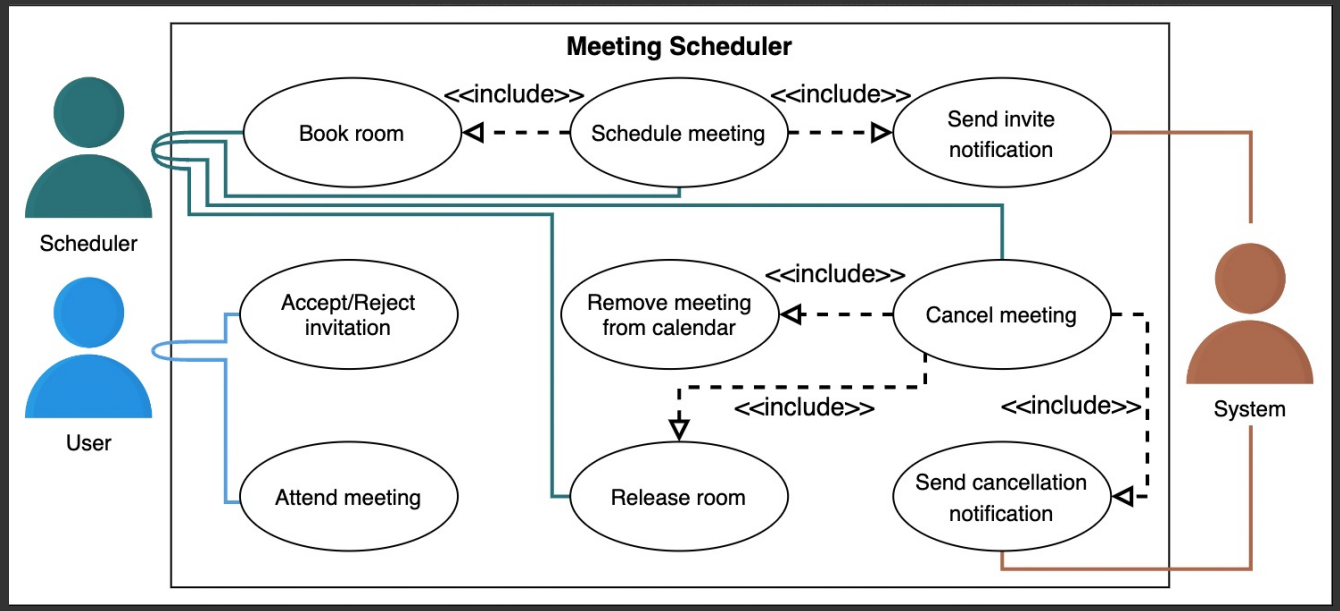
R6: Each User should have access to a calender that is used to track the date & time, as well as to schedule or cancel meetings.

Actors

Primary Actor → • Scheduler
• User

Secondary Actor → • System

Scheduler	User	System
Schedule/Cancel meeting	Attend meeting	Send invite notification
Book/Release room	Accept/Reject meeting	Send cancelation notification



Class Diagram for the Meeting Scheduler

User

User
- name : string - email : string
+ respondInvitation(invite) : void

Interval

Interval
- startTime : date/time - endTime : date/time

Meeting Room

Meeting Room
- id : int - capacity : int - bookedIntervals : Interval {list} - isAvailable : bool

Meeting

Meeting
- id : int - participants : User {list} - participantsCount : int - interval : Interval - room : MeetingRoom - subject : string
+ addParticipants(participants {list}) : void

Calendar

Calendar
- meetings : Meeting {list}

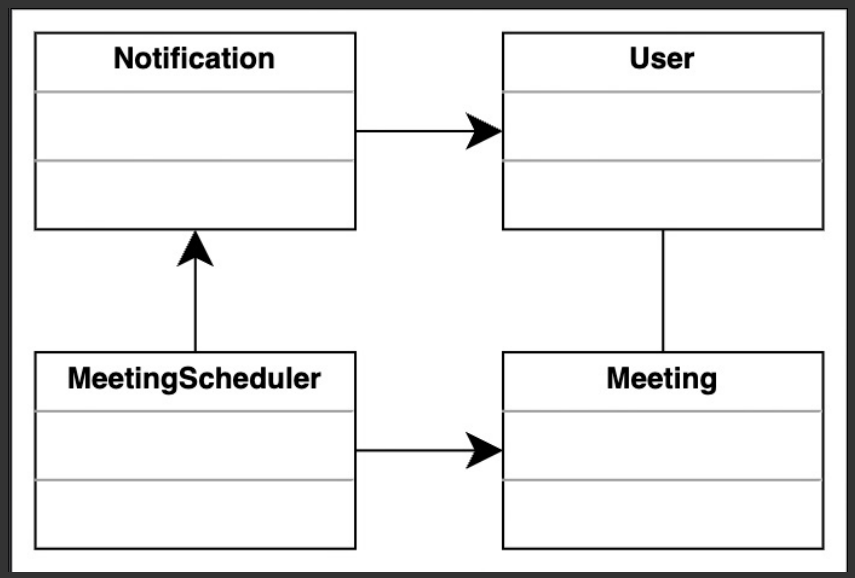
Meeting Scheduler

MeetingScheduler
- organizer : User - calendar : Calendar - rooms : Room {list}
+ scheduleMeeting(user {list}, interval) : bool + cancelMeeting(user {list}, interval) : bool + bookRoom(room, numberOfPersons, interval) : bool + releaseRoom(room, interval) : bool + checkRoomsAvailability(numberOfPersons, interval) : MeetingRoom

Notification

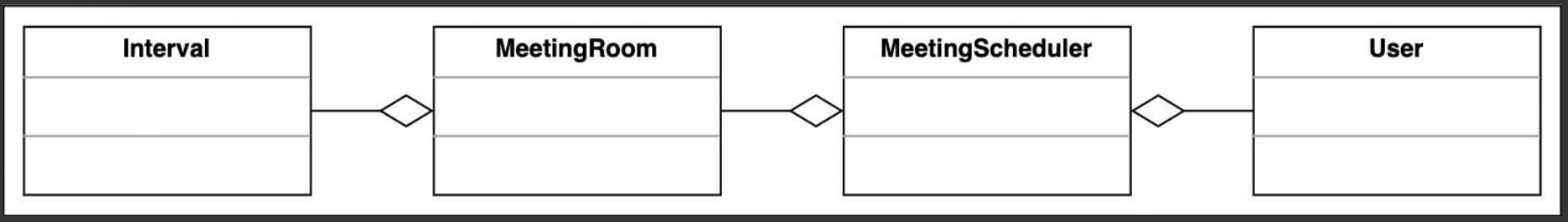
Notification
- notificationId : int - content : string - creationDate : date/time
+ sendInvite(user) : void + cancelNotification(user) : void

Association

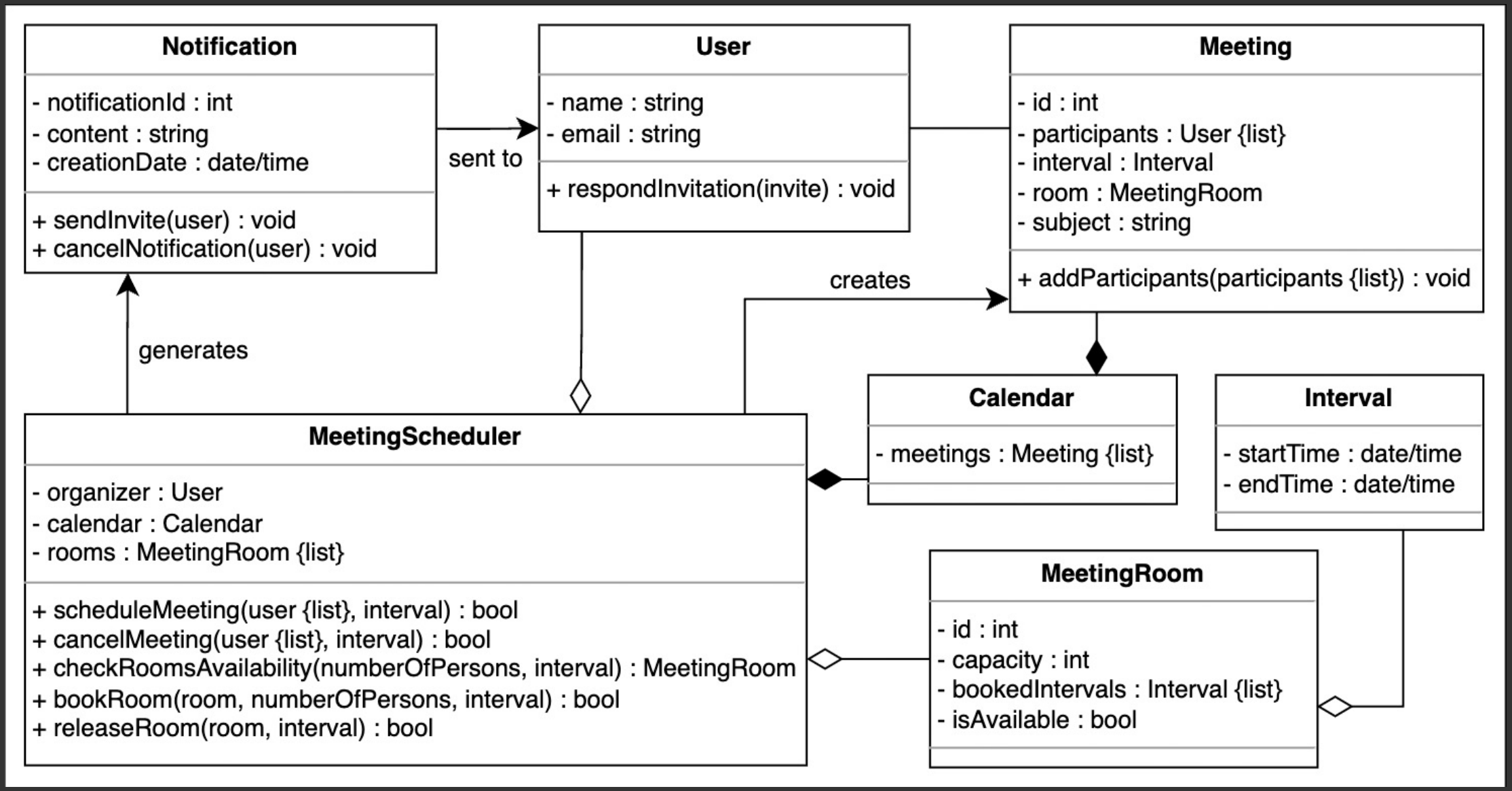


Composition

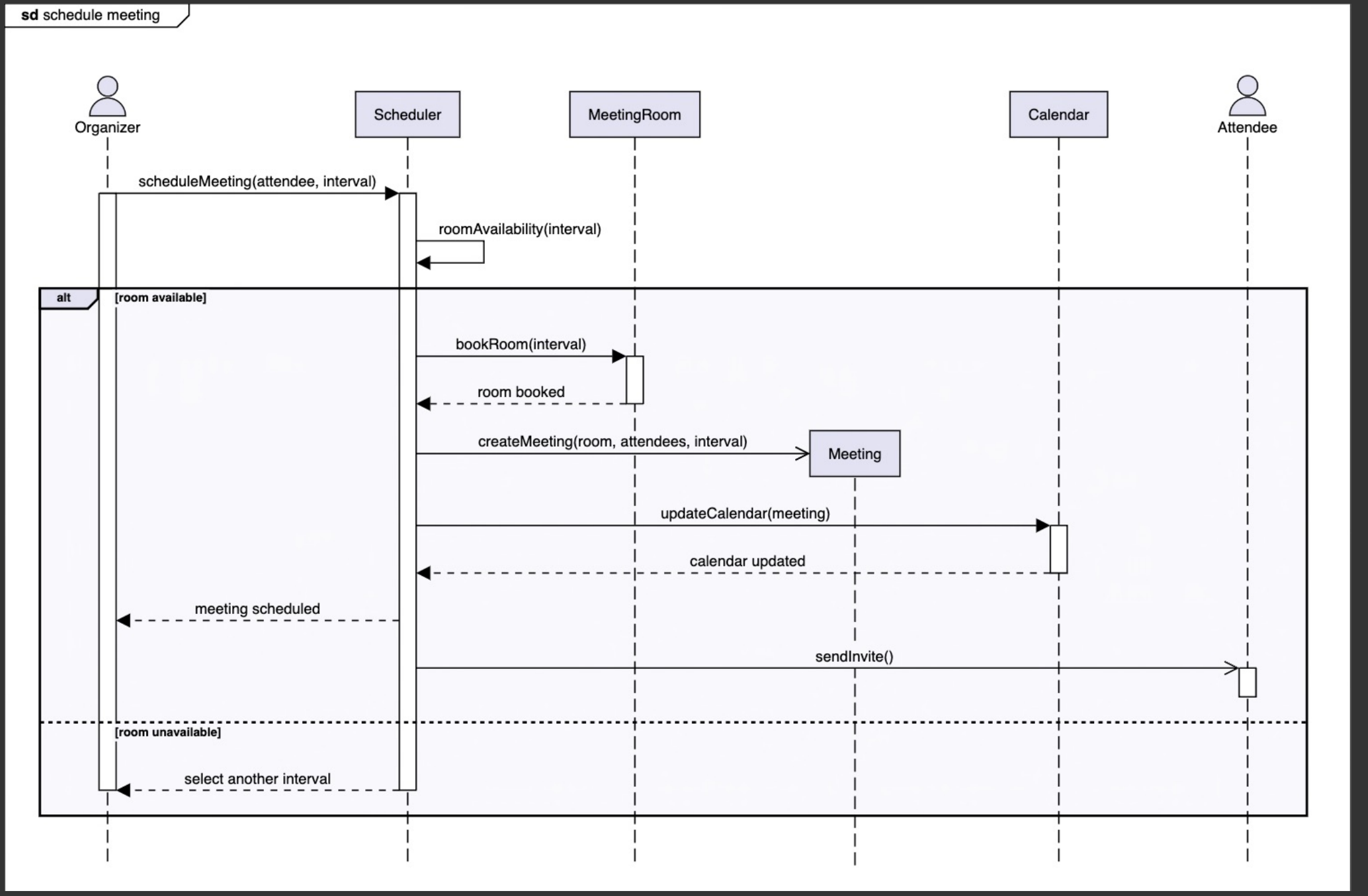
Aggregation

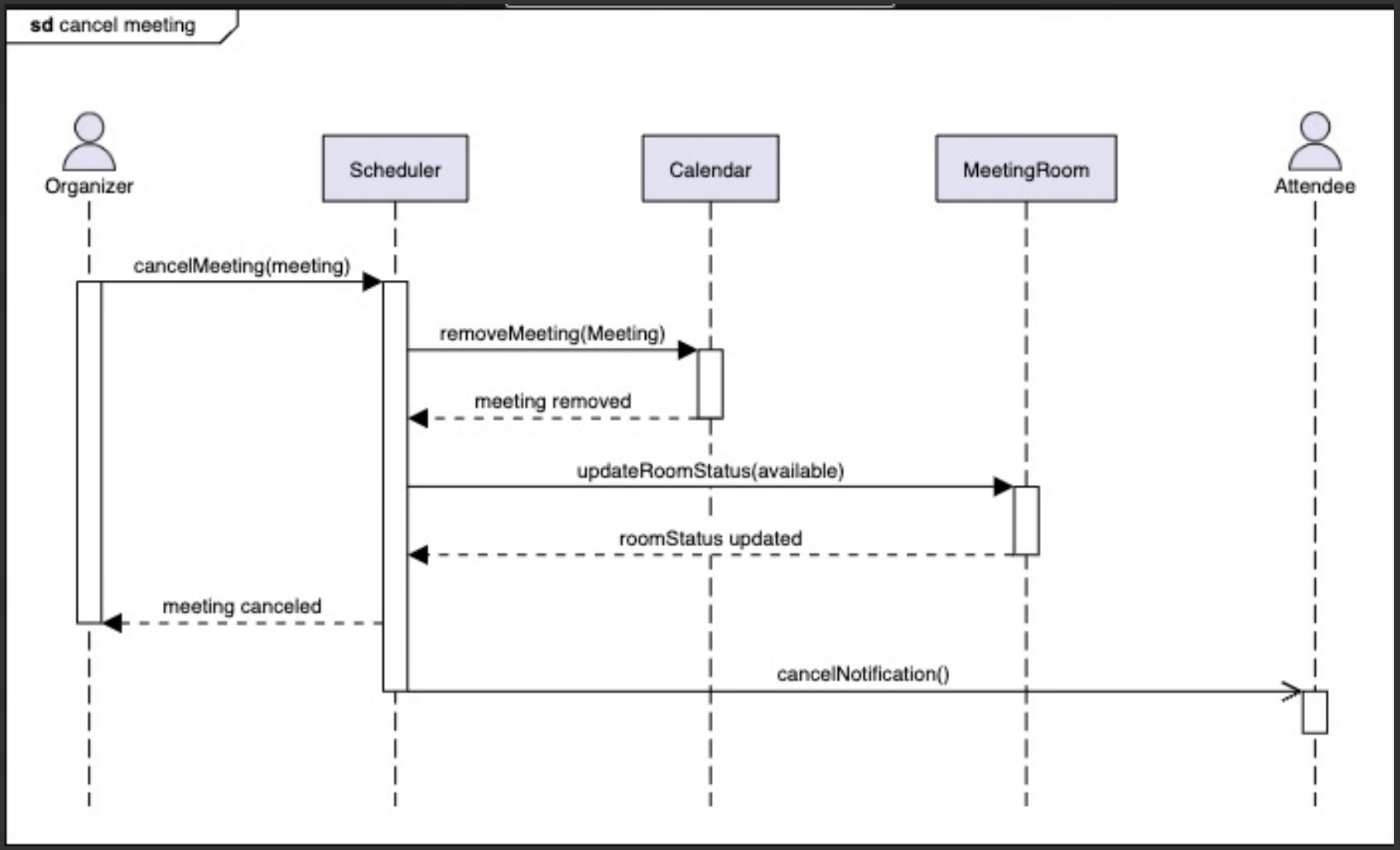


Class Diagram

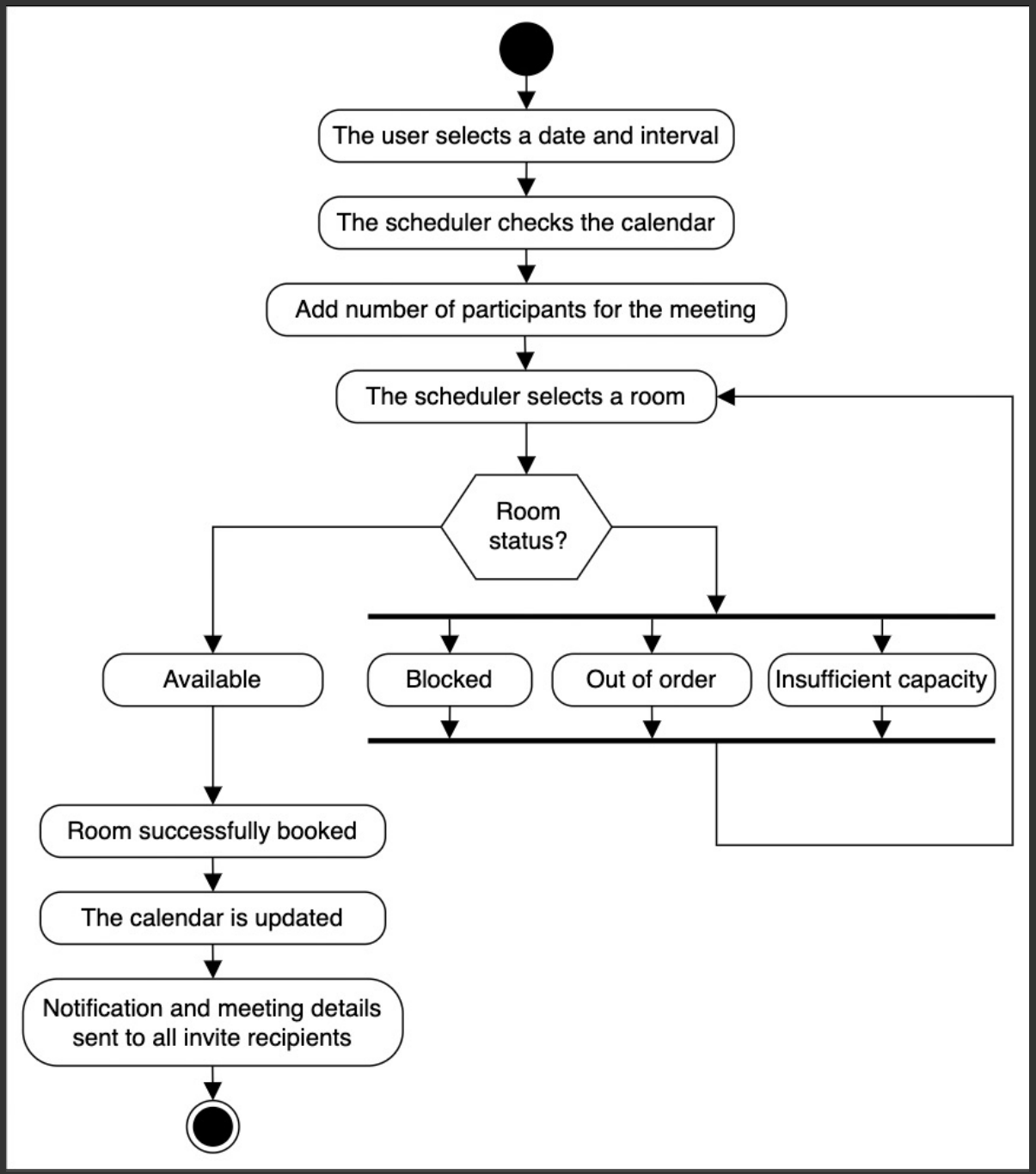


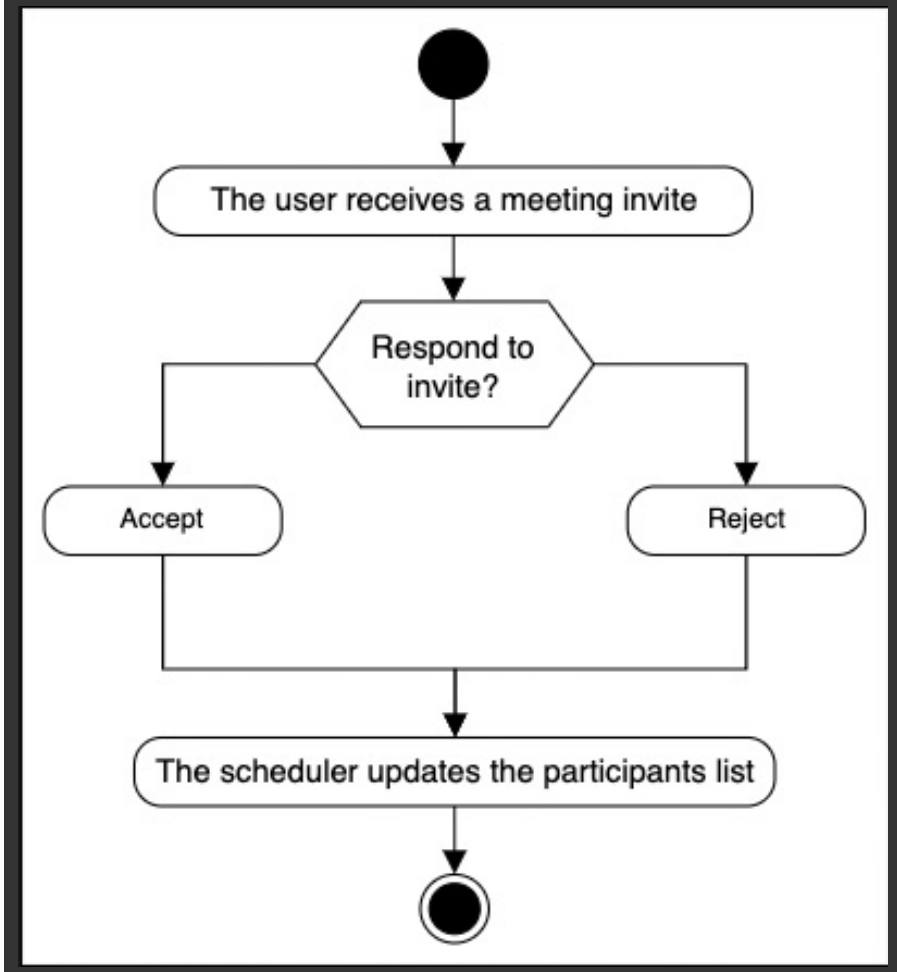
Schedule Meeting




sd cancel meeting
Organizer
Scheduler
Calendar
MeetingRoom
Attendee
cancelMeeting(meeting)
removeMeeting(Meeting)
meeting removed
updateRoomStatus(available)
roomStatus updated
meeting canceled
cancelNotification()

Activity Diagram





Code for Meeting Scheduler

1. User

```
public class User {
  private String name;
  private String email;
  public void respondInvitation(Notification invite);
}
```

2. Interval

```
public class Interval {
  private Date startTime;
  private Date endTime;
}
```

3. Meeting Room

```
public class MeetingRoom {
  private int id;
  private int capacity;
  private boolean isAvailable;
  private List<Interval>
bookedIntervals;
}
```

4. Calendar

```
public class Calendar {
  private List<Meeting> meetings;
}
```

5. Meeting

```
public class Meeting {
  private int id;
  private int participantsCount;
  private List<User> participants;
  private Interval interval;
  private MeetingRoom room;
  private String subject

  public void addParticipants(List<User> participants);
}
```

6. Muting Scheduler

```
public class MeetingScheduler {
  private User organizer;
  private Calendar calendar;
  private List<MeetingRoom> rooms;

  // Scheduler is a singleton class that ensures it will have only one active instance at a time
  private static MeetingScheduler scheduler = null;

  // Created a static method to access the singleton instance of Scheduler class
  public static MeetingScheduler getInstance() {
    if (scheduler == null) {
      scheduler = new MeetingScheduler();
    }
    return scheduler;
  }

  public boolean scheduleMeeting(List<User> users, Interval interval);
  public boolean cancelMeeting(List<User> users, Interval interval);
  public boolean bookRoom(MeetingRoom room, int numberOfPersons, Interval interval);
  public boolean releaseRoom(MeetingRoom room, Interval interval);
  public MeetingRoom checkRoomsAvailability(int numberOfPersons, Interval interval);
}
```

7. Notification

```
public class Notification {
  private int notificationId;
  private string content;
  private Date creationDate;

  public boolean sendNotification(User user);
  public boolean cancelNotification(User user);
}
```