Design of Meeting Scheduling Application

Requirement Gathering –

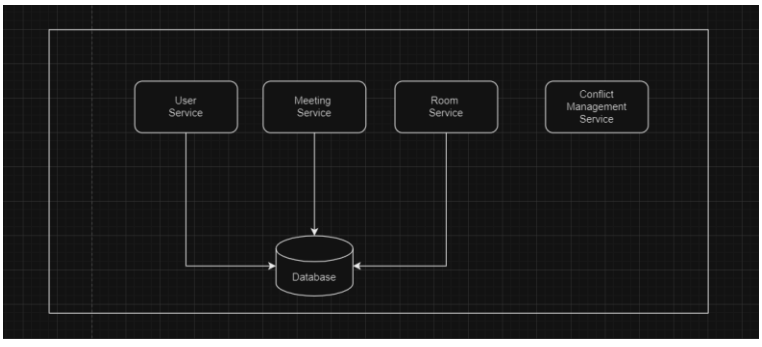
i) User should be able to schedule a meeting using start (from) and end (to) time.
ii) We need to check if any collisions are there while scheduling the meeting for both single-person and multi-person meetings.
iii) Add other persons to meeting.
iv) Check availability of person.
v) We need to have room accommodation and avoid room conflicts.

High Level Architecture –

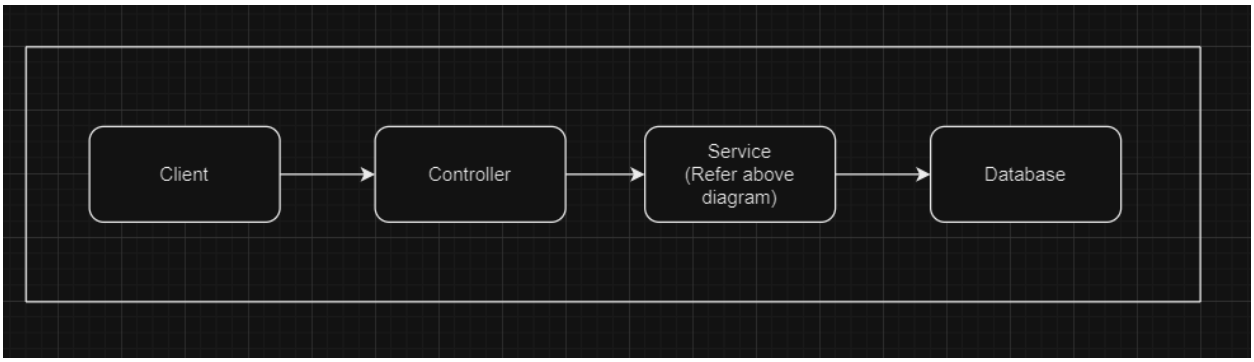
Components -

1. User Service – Manages all the operations regarding user activities.
2. Meeting Service – Manages all information for meetings.
3. Room Service – Manages room availability and capacity checks.
4. Conflict Management Service – To check the meeting conflicts.
5. Database – Stores all the metadata.

Service Level Diagram –



Overall High Level Architecture Diagram –



Database Selection –

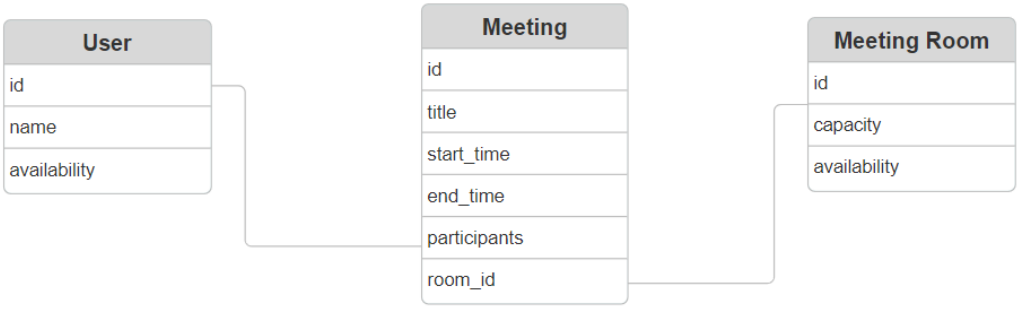
We will prefer selecting Relational Database (RDBMS) as here we are dealing with structured data. So DB like Mysql or Postgres will be a good choice here.

Tables:

- User: Stores user information and availability.
- MeetingRoom: Stores room information and availability.
- Meeting: Stores meeting details, linked to Users and MeetingRooms.

Schema:

- User(id, name, availability)
- MeetingRoom(id, capacity, availability)
- Meeting(id, title, startTime, endTime, participants, roomId)



API Contract –

- User Management:

- POST /users/create-user: Create user.

  Request –

```
{
  "name": "Shubh",
  "availability": [
    {
      "start": "2024-09-28T12:47:31.768Z",
      "end": "2024-09-28T13:47:31.768Z"
    }
  ]
}
```

  Response –

```
{
  "id": 3,
  "name": "Shubh",
  "availability": [
    {
      "start": "2024-09-28T12:47:31.768",
      "end": "2024-09-28T13:47:31.768"
    }
  ]
}
```

- Meeting Room Management:

- POST /rooms/create-room: Create a new meeting room.

  Request –

```
{
  "name": "scrum room",
  "capacity": 4,
  "availability": [
    {
      "start": "2024-09-28T09:48:52.161Z",
      "end": "2024-09-28T12:48:52.161Z"
    }
  ]
}
```

Response –


```
{
  "id": 2,
  "name": "scrum room",
  "capacity": 4,
  "availability": [
    {
      "start": "2024-09-28T09:48:52.161",
      "end": "2024-09-28T12:48:52.161"
    }
  ]
}
```


- Meeting Management:

- POST /meetings/create-meeting: Schedule a new meeting.

  Request –


```
{
  "title": "Tech talks",
  "startTime": "2024-09-28T12:51:55.833Z",
  "endTime": "2024-09-28T13:40:55.833Z",
  "participantIds": [1, 2],
  "roomId": 1
}
```


Response –

```
Meeting - Tech talks scheduled successfully.
```

Implementation Details –

I) Entity Classes –

- User.java
- Meeting.java
- MeetingRoom.java

II) Service Classes –

- UserService.java – createUser(), checkAvailability(), updateAvailability()
- MeetingService.java – createMeeting(), updateMeeting()

- MeetingRoomService.java – createMeetingRoom(), checkRoomAvailability()
- ConflictManagementService.java – checkOverallAvailability(roomId, paricipants, time), checkRoomConflict(), checkUserConflict()

III) Repository Classes/Interfaces –

- UserRepository.java
- MeetingRepository.java
- MeetingRoomRepository.java