This doc elaborates on the design pattern and features of the projects with screenshots.

To get detailed descriptions and installation steps to run the app you can view the readme.md file.

Models/Interfaces:

1) UserModel:

```
{
  userId: {
    type: String,
    unique: true,
    trim: true,
  },
}
```

We've just kept the userId for the user. Which is generated by us, Starting from U1 and increasing the suffix number by 1 for each user.
We could also add name, age, etc fields. Or we can establish an authentication system also, but for the sake of this assignment, we've kept it simple.

2) ScheduleModel:

```
{
    hostUserId: {
      type: ObjectId,
      ref: "User",
      required: true,
    },
    guestUsers: {
      type: [ObjectId],
      ref: "User",
    },
    roomId: {
      type: String,
      required: true,
    },
    meetingDate: {
      type: Date,
      required: true,
    },
    startTime: {
      type: Date,
      required: true,
    },
    endTime: {
      type: Date,
      required: true,
    },
    offset:{
      type:Number
    }
  }
```

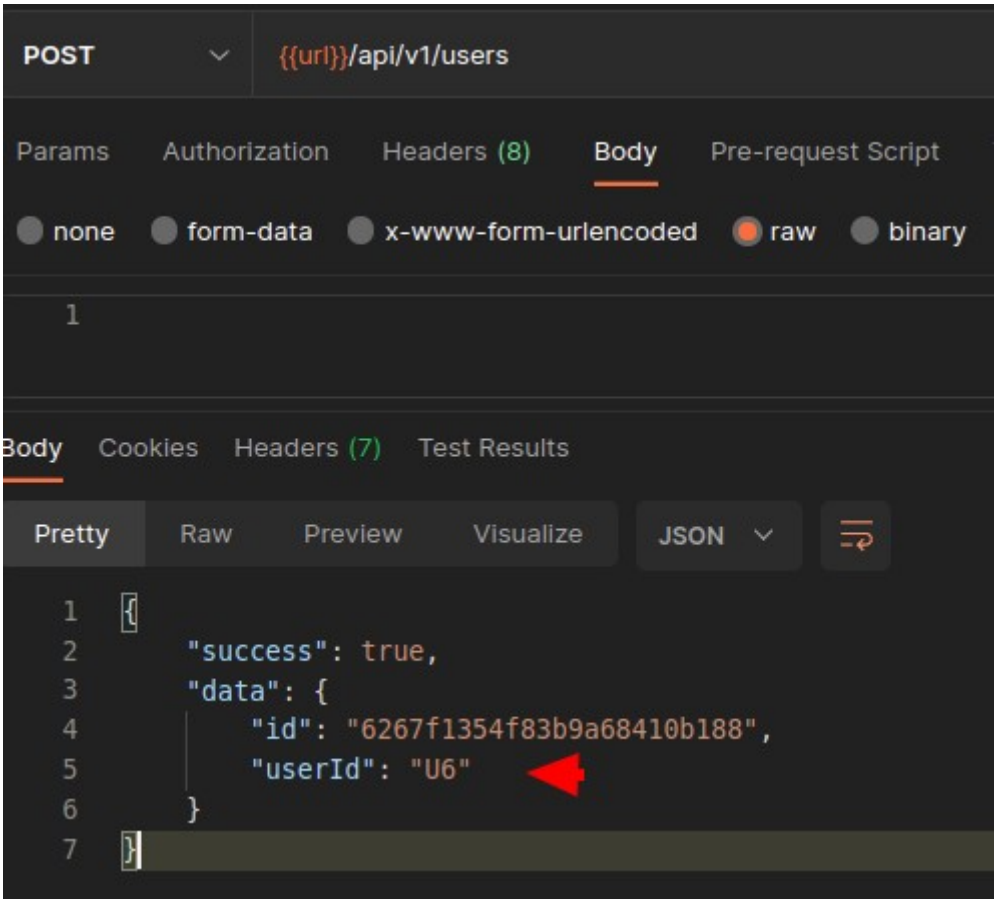
hostUserId: Holds the userid of the **host**.
guestUsers: An **array** of userIds of the **guests**.
roomId: The room ID which is from a fixed set of rooms. **["R1", "R2", "R3", "R4", "R5"]**, We can add more room in **/utils/RoomList.js**.

meetingDate: Date of the meeting in **YYYY-MM-DD** format
startTime: Start time of the meeting in **HH:MM** format.
endTime: End time of the meeting in **HH:MM** format.
offset: Holds the offset value of the runtime(node) for better user experience, we can make the offset value respective to the user/client for a more precise user experience.

In this project, a host user can schedule meetings. There are certain constraints while creating a meeting,
1) the host user must not have any other meetings on that day in that specific time slot.
2) The meeting room should also be free in that specific time slot.
3) If there are added guests they also have to be free on that day in that specific time slot.

Screenshots:

1) Creating users:



2) List all users:

GET {{url}}/api/v1/users

Params Authorization Headers (7) Body Pre-request Script Tests

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "data": [
        {
            "_id": "6266e5d88ff1715c92c8c153",
            "userId": "U1"
        },
        {
            "_id": "6266e5e68ff1715c92c8c15c",
            "userId": "U2"
        },
        {
            "_id": "6266f419b79a6dd0c5f026bf",
            "userId": "U3"
        },
        {
            "_id": "6266f431b79a6dd0c5f026d0",
            "userId": "U4"
        },
        {
            "_id": "626712d62aa36116969a5e74",
            "userId": "U5"
        },
        {
            "_id": "6267f1354f83b9a68410b188",
            "userId": "U6"
        }
    ]
}
```

3) Create a meeting scedule:

POST {{url}}/api/v1/schedule

Params Authorization Headers (9) Body Pre-request Script Tests Se

none form-data x-www-form-urlencoded raw binary GraphQL

```
{
    "userId":"U1",
    "roomId":"R2",
    "guestUsers":["U2", "U3"],
    "meetingDate":"2022-04-26",
    "startTime":"12:30",
    "endTime":"13:30"
}
```

host

guests

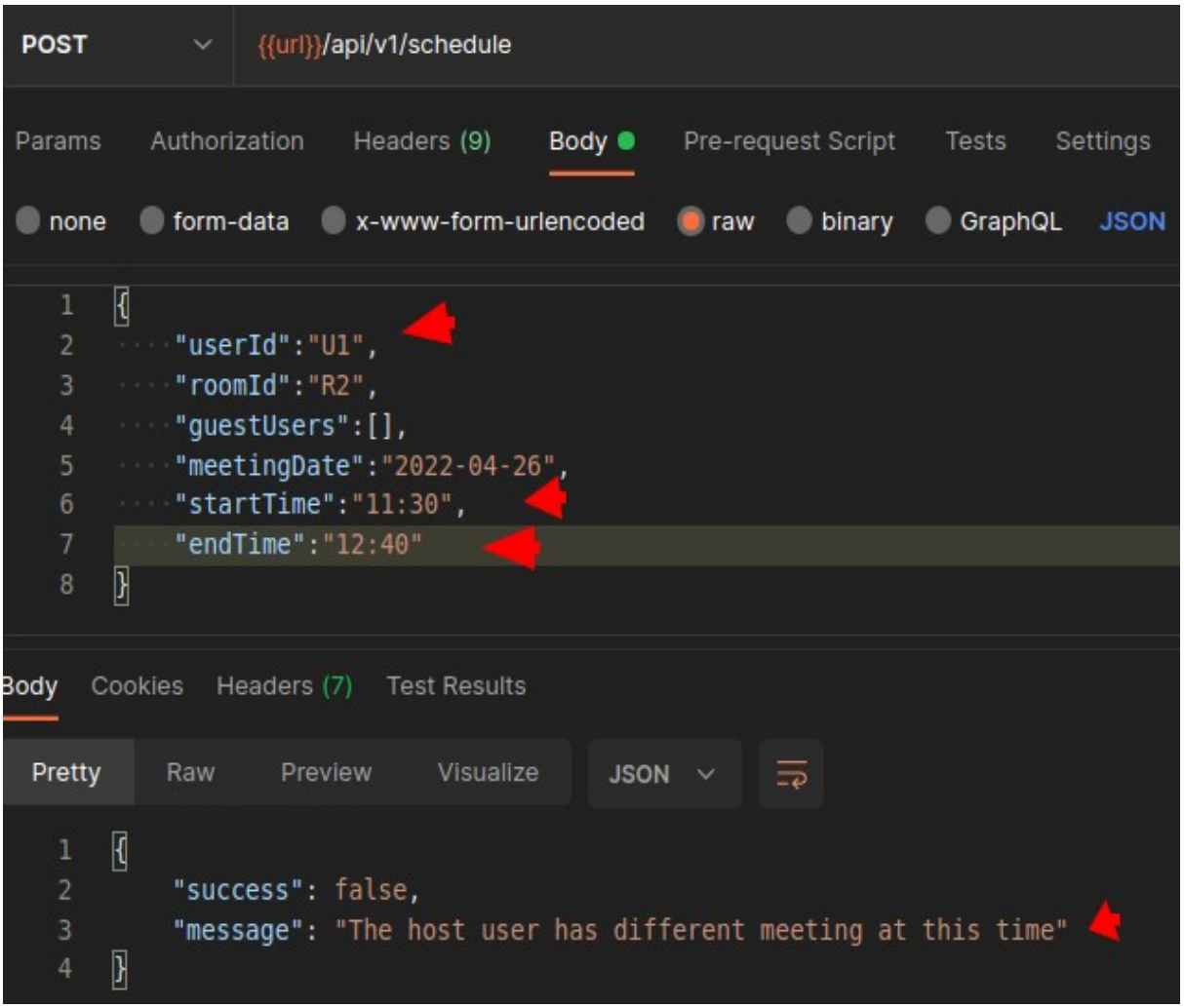
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "success": true,
    "data": {
        "hostUserId": "U1",
        "guestUsers": [
            "U2",
            "U3"
        ],
        "roomId": "R2",
        "meetingDate": "2022-04-26T05:30:00.000Z",
        "startTime": "2022-04-26T12:30:00.000Z",
        "endTime": "2022-04-26T13:30:00.000Z",
        "offset": -330,
        "_id": "6267f1e64f83b9a68410b193",
        "createdAt": "2022-04-26T13:21:42.296Z",
        "updatedAt": "2022-04-26T13:21:42.296Z",
        "__v": 0
    }
```

Now the User U1, U2 and U3 are already in a meeting on 2022-04-26 from 12:30 to 13:30 in Room R2.

Error case 1:
Now suppose User U1 wanted to create another meeting where the end time is more than 12:30.



Error case 2:
Now, if we change the host to U4 and try to book a meeting, where the time overlaps with the first meeting in room R2.

POST {{url}}/api/v1/schedule

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary Grap

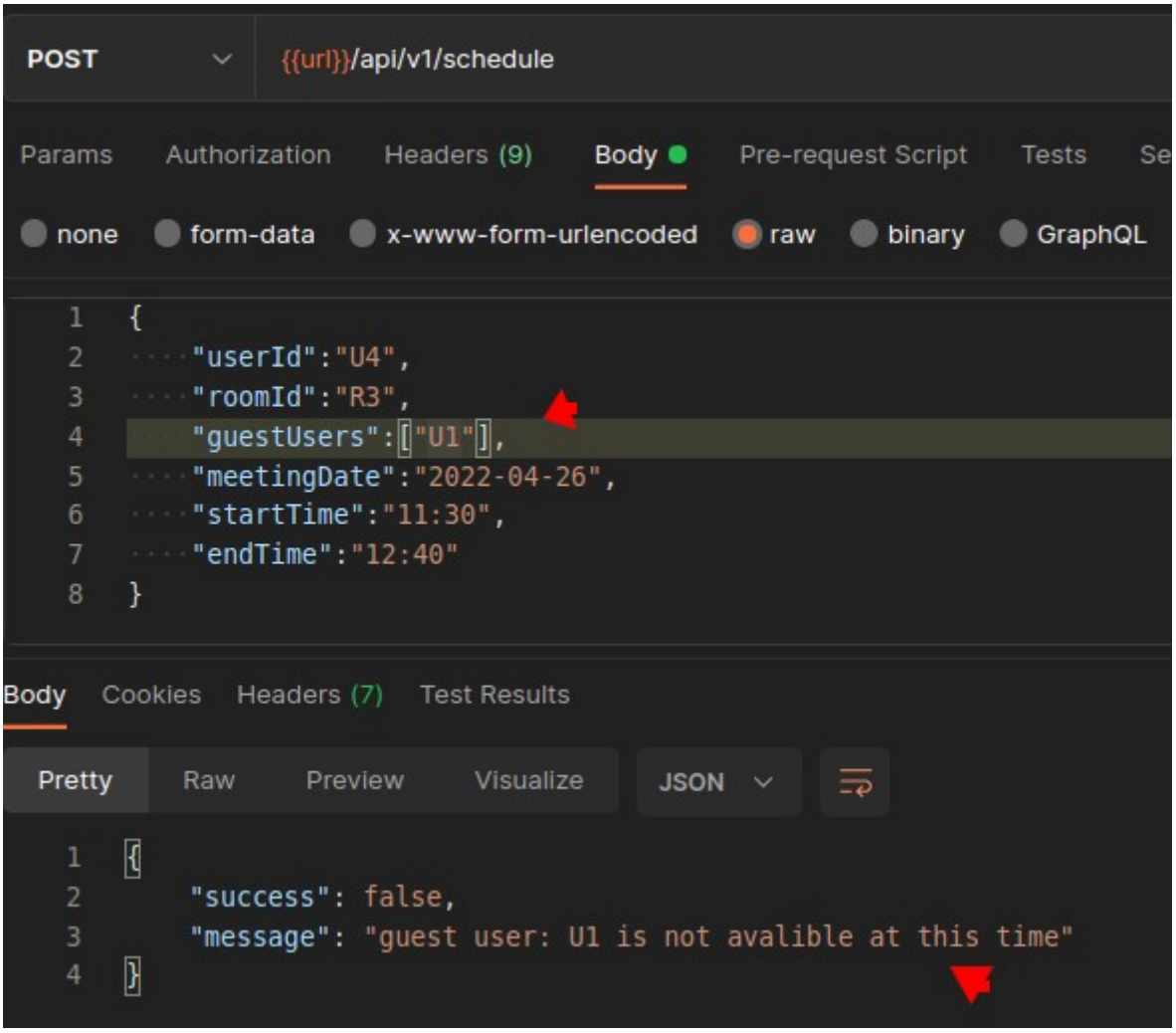
```
{
    "userId":"U4",
    "roomId":"R2",
    "guestUsers":[],
    "meetingDate":"2022-04-26",
    "startTime":"11:30",
    "endTime":"12:40"
}
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "success": false,
    "message": "The room is not empty in this time slot"
}
```

Error case 3:
If we create another meeting where U4 is the host but he's keeping U1 as a guest, but U1 already has a overlapped meeting, in that case:



4) List all meetings for specific user:

GET {{url}}/api/v1/schedule/byUser?userId=U2

Params Authorization Headers (7) Body Pre-request Script Tests

Query Params

KEY	VALUE
☑ userId	U2

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "data": [
        {
            "_id": "626712b02aa36116969a5e6b",
            "hostUserId": "U3",
            "guestUsers": [
                "U2",
                "U4"
            ],
            "roomId": "R2",
            "meetingDate": "2022-04-25T05:30:00.000Z",
            "startTime": "2022-04-25T12:30:00.000Z",
            "endTime": "2022-04-25T12:35:00.000Z",
            "offset": -330,
            "createdAt": "2022-04-25T21:29:20.643Z",
            "updatedAt": "2022-04-25T21:29:20.643Z",
            "__v": 0
        },
```


It returns all the meetings for a specific user, **sorted by date and time**, so that the user can see the closest meeting first. **A user can also see the meetings in which they are a guest**.

5) List all meetings for specific room:

GET {{url}}/api/v1/schedule/byRoom?roomId=R2

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
roomId	R2
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "success": true,
    "data": [
        {
            "_id": "626712b02aa36116969a5e6b",
            "hostUserId": "U3",
            "guestUsers": [
                "U2",
                "U4"
            ],
            "roomId": "R2",
            "meetingDate": "2022-04-25T05:30:00.000Z",
            "startTime": "2022-04-25T12:30:00.000Z",
            "endTime": "2022-04-25T12:35:00.000Z",
            "offset": -330,
            "createdAt": "2022-04-25T21:29:20.643Z",
            "updatedAt": "2022-04-25T21:29:20.643Z",
            "__v": 0
        },
```