

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.

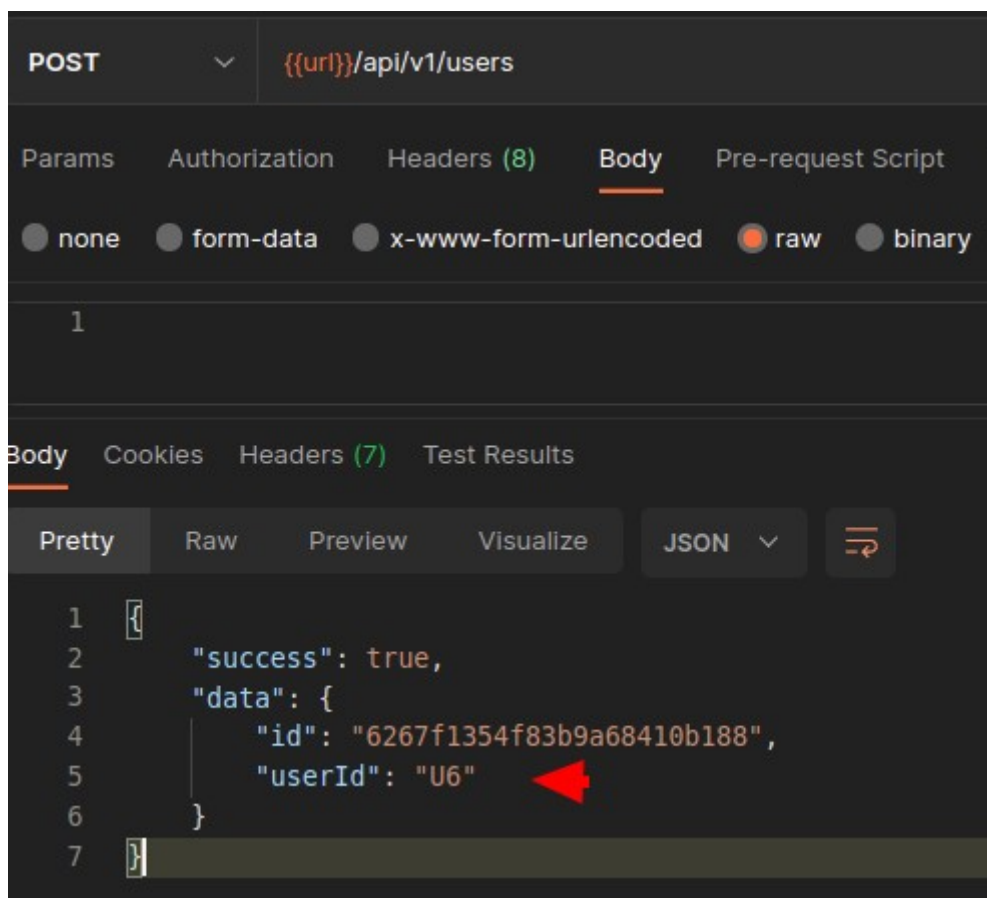
offset: Holds the offset value of the runtime(node) for better user experience, we can make the offset value respective to the user/client for a more precise user experience.

In this project, a host user can schedule meetings. There are certain constraints while creating a meeting,

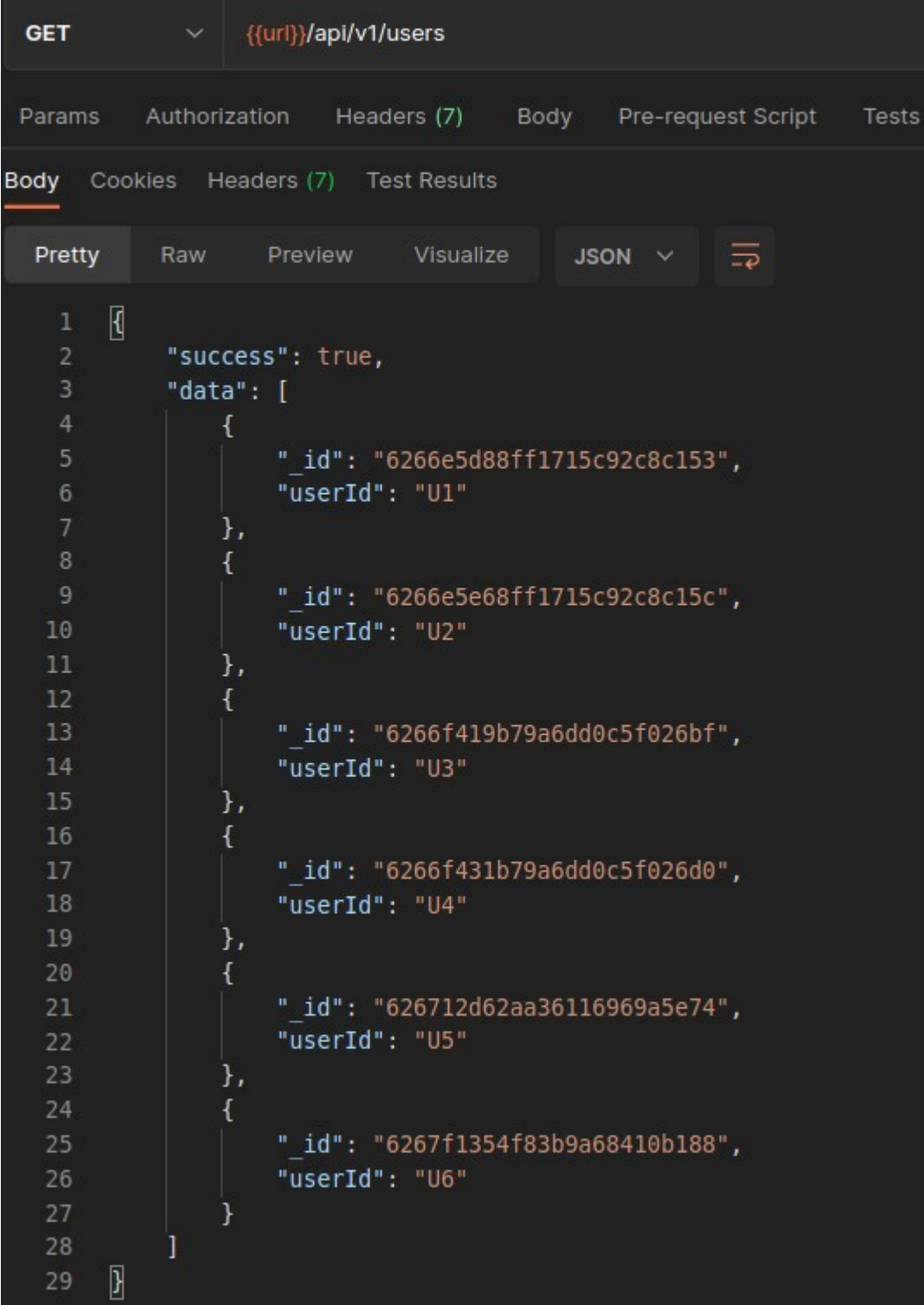
- 1) the host user must not have any other meetings on that day in that specific time slot.
- 2) The meeting room should also be free in that specific time slot.
- 3) If there are added guests they also have to be free on that day in that specific time slot.

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  ↺

1  [
2    "success": true,
3    "data": [
4      {
5        "_id": "6266e5d88ff1715c92c8c153",
6        "userId": "U1"
7      },
8      {
9        "_id": "6266e5e68ff1715c92c8c15c",
10       "userId": "U2"
11     },
12     {
13       "_id": "6266f419b79a6dd0c5f026bf",
14       "userId": "U3"
15     },
16     {
17       "_id": "6266f431b79a6dd0c5f026d0",
18       "userId": "U4"
19     },
20     {
21       "_id": "626712d62aa36116969a5e74",
22       "userId": "U5"
23     },
24     {
25       "_id": "6267f1354f83b9a68410b188",
26       "userId": "U6"
27     }
28   ]
29 ]
```

3) Create a meeting schedule:

The screenshot displays a REST client interface with a POST request to `{{url}}/api/v1/schedule`. The request body is a JSON object with the following fields: `userId` ("U1", labeled **host**), `roomId` ("R2"), `guestUsers` (["U2", "U3", labeled **guests**]), `meetingDate` ("2022-04-26"), `startTime` ("12:30"), and `endTime` ("13:30").

The response body is a JSON object with the following fields: `success` (true), `data` (an object containing `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"), and `__v` (0)).

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

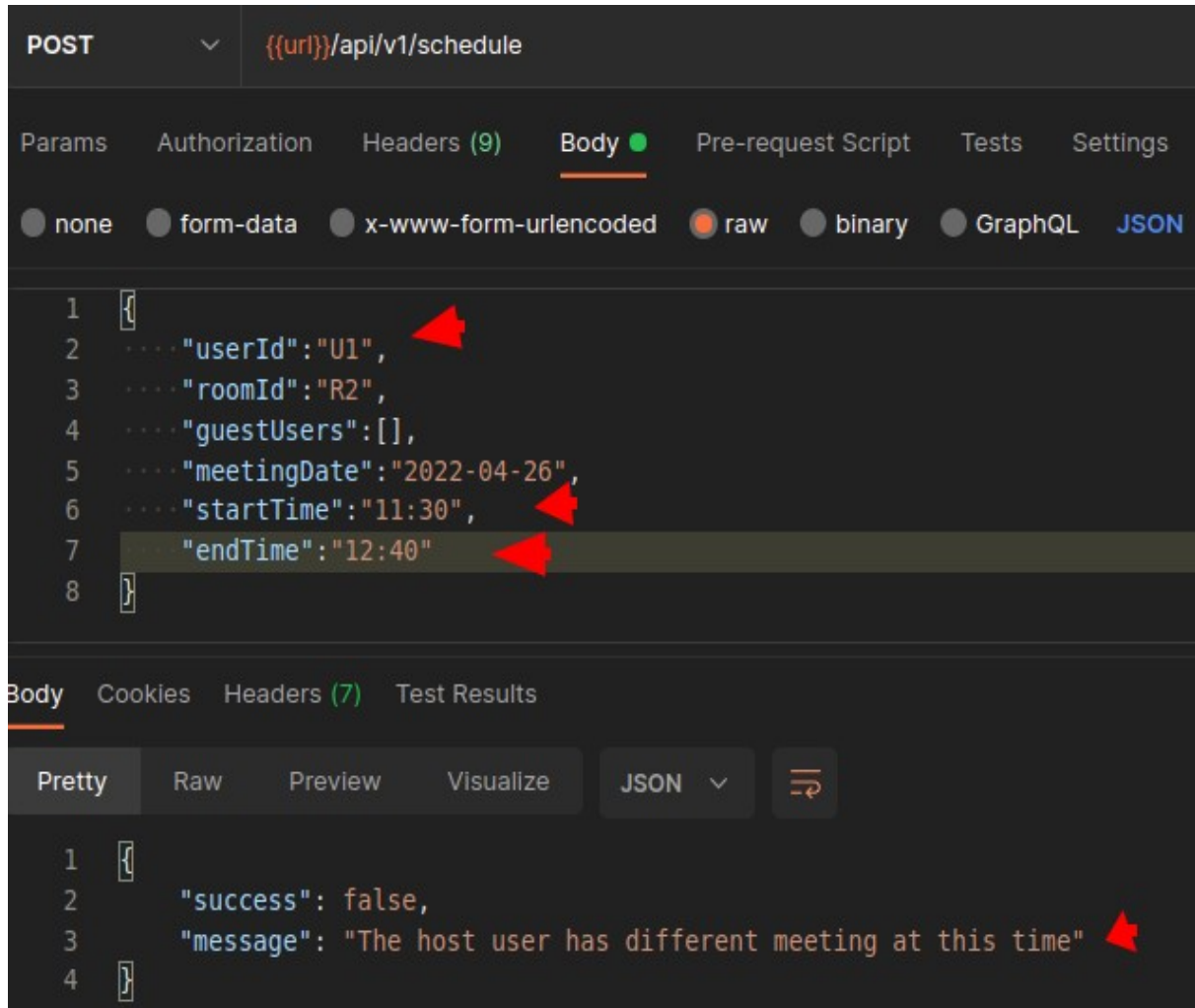
{
  "userId": "U1",
  "roomId": "R2",
  "guestUsers": ["U2", "U3"],
  "meetingDate": "2022-04-26",
  "startTime": "12:30",
  "endTime": "13:30"
}
```

```
{
  "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 ☐ GraphQL

```
1 {
2   ... "userId": "U4",
3   ... "roomId": "R2",
4   ... "guestUsers": [],
5   ... "meetingDate": "2022-04-26",
6   ... "startTime": "11:30",
7   ... "endTime": "12:40"
8 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↺

```
1 {
2   "success": false,
3   "message": "The room is not empty in this time slot"
4 }
```

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:

The screenshot displays a REST client interface for a POST request to `{{url}}/api/v1/schedule`. The request body is a JSON object with the following fields: `userId` ("U4"), `roomId` ("R3"), `guestUsers` (an array containing "U1"), `meetingDate` ("2022-04-26"), `startTime` ("11:30"), and `endTime` ("12:40"). A red arrow points to the `guestUsers` field. The response body, shown in JSON format, indicates a failure: `"success": false` and `"message": "guest user: U1 is not available at this time"`. A red arrow points to the message text.

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

{
  "userId": "U4",
  "roomId": "R3",
  "guestUsers": ["U1"],
  "meetingDate": "2022-04-26",
  "startTime": "11:30",
  "endTime": "12:40"
}
```

```
{
  "success": false,
  "message": "guest user: U1 is not available at this time"
}
```


4) List all meetings for specific user:

The screenshot shows a REST client interface with a GET request to the endpoint `{{url}}/api/v1/schedule/byUser?userId=U2`. The 'Params' tab is active, showing a query parameter `userId` with the value `U2`. The 'Body' tab is also active, displaying the JSON response in 'Pretty' format. The response indicates a successful request with a list of meeting data for user `U2`.

KEY	VALUE
<input checked="" type="checkbox"/> <code>userId</code>	<code>U2</code>

```
1 {
2   "success": true,
3   "data": [
4     {
5       "_id": "626712b02aa36116969a5e6b",
6       "hostUserId": "U3",
7       "guestUsers": [
8         "U2",
9         "U4"
10      ],
11      "roomId": "R2",
12      "meetingDate": "2022-04-25T05:30:00.000Z",
13      "startTime": "2022-04-25T12:30:00.000Z",
14      "endTime": "2022-04-25T12:35:00.000Z",
15      "offset": -330,
16      "createdAt": "2022-04-25T21:29:20.643Z",
17      "updatedAt": "2022-04-25T21:29:20.643Z",
18      "_v": 0
19    },
20  ],
21 }
```

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
<input checked="" type="checkbox"/>	roomId	R2
	Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1  [
2    "success": true,
3    "data": [
4      {
5        "_id": "626712b02aa36116969a5e6b",
6        "hostUserId": "U3",
7        "guestUsers": [
8          "U2",
9          "U4"
10       ],
11       "roomId": "R2",
12       "meetingDate": "2022-04-25T05:30:00.000Z",
13       "startTime": "2022-04-25T12:30:00.000Z",
14       "endTime": "2022-04-25T12:35:00.000Z",
15       "offset": -330,
16       "createdAt": "2022-04-25T21:29:20.643Z",
17       "updatedAt": "2022-04-25T21:29:20.643Z",
18       "__v": 0
19     },
```