Meeting Application

You have to build an application that maintains meetings for a user (say, yourself). This application helps you manage your meetings - you can filter to view meetings (past, present and future), or search for meetings based on the meetings description. You can add (i.e. create) meetings. You will by default be part of a meeting you create. The users added when creating a meeting will automatically be part of the meeting (no concept of accepting a meeting!). However, they can excuse themselves from the meeting (drop off from a scheduled meeting).

The application helps you view your meetings for a day in a calendar view.

Optional part:

Additionally, you can create teams consisting of yourself and other users. You will by default be part of a team you create. You can view all teams that you are part of and add members of these teams. You can excuse yourself from the team (leave the team). Once you do so you will not be able to view the team details, nor be part of new meetings where the team has been added (i.e. team members are attendees).

Features :-

1. Register new user
2. Login
3. Create a meeting
4. View meetings
5. Search meetings
6. Drop off from meeting

Feature 1 : Register

Scenario Outline: Successful Registration
When user enters valid name as "<name>"
And valid email as "<email>"
And email is not already registered
And password as "<password>"
Then payload given

```
{
        message: String                 e.g. ”User Registered Successfully”,
        registration-name: String       e.g. ”Ram”
}
```

Input

```
{
        registration-name: String       e.g. ”Ram”,
        userid: String                  e.g. ram@success.com,
        password: String                e.g. ”borntowin”,
```

}

Method Type: POST

Return Status: 201

URL: http://localhost:4000/user

Feature 2 : Login

Scenario (Valid):

When user enters valid email as "<email>"

And password as "<password>"

And credentials entered are correct

Then payload given

```
{
        message: String
}
```

Input

```
{
        userid: String,
        password: String
}
```

Method Type: POST

Return Status: 201

Scenario Outline (invalid):

When user entered unregistered email as "<email>"

And password as "<password>"

Then payload given

```
{
        message: String
}
```

Input

```
{
        userid:String,
        password: String
}
```

Method Type: POST

Return Status: 401

URL: http://localhost:4000/login

Feature 3 : Creating a Meeting

Scenario Outline (Valid): User creates a meeting

When user enters valid meeting details – date of meeting, start and end times , description and email ids

Then payload returned

```
{
        message: String
        meeting-id: String
}
```

Input

```
{

    user-details: {
        "userid:String,
        "password:String
    },
    meeting: {
        date-of-meeting: Date
        start-time: String  => (in hours 0-23 and minutes 0-59)
        end-time: String  => (in hours 0-23 and minutes 0-59)
        description: String
        email-ids-of-attendees: String  =>multiple email ids separated by comma
    }
}
```

Note: The application should take valid user details, and from the server side it must be validated and then the meeting should be crated.
Method Type: POST
Return Status: 201
URL: http://localhost:4000/meeting

Feature 4: View Meetings

Scenario Outline (Valid):

To get all the details of meetings , the end point should return an array of all the meetings

```
[

{
```

meeting-id: String
date-of-meeting: Date
start-time: String => (in hours 0-23 and minutes 0-59)
end-time: String => (in hours 0-23 and minutes 0-59)
description: String
email-ids-of-attendees: String =>multiple email ids separated by comma
},
{
meeting-id: String
date-of-meeting: Date
start-time: String => (in hours 0-23 and minutes 0-59)
end-time: String => (in hours 0-23 and minutes 0-59)
description: String
email-ids-of-attendees: String =>multiple email ids separated by comma
}

]

Note: The application should take valid user details, and from the server side it must be validated and then the meetings for a user should be returned
Method Type: GET
Return Status: 200
URL: http://localhost:4000/user/meetings

Feature 5: Search for a specific meeting

Scenario Outline (Valid):

To get details of specific meeting, user should provide his user-id.

If meeting-id is valid, should return the meeting details

{
meeting-id: String
date-of-meeting: Date
start-time: String => (in hours 0-23 and minutes 0-59)
end-time: String => (in hours 0-23 and minutes 0-59)
description: String
email-ids-of-attendees: String =>multiple email ids separated by comma
}

Scenario Outline (Invalid):

{

message: String e.g. “Meeting id doesn’t exist”

}

Note: The application should take valid user details, and from the server side it must be validated and then the meetings for a user should be returned
Method Type: GET
Return Status: 200
URL: http://localhost:4000/user/meetings

Feature 6: Drop off from meeting

To get dropped off from a meeting, user should provide his user-id and meeting-id.

Scenario Outline (Valid):

{

message: String e.g. "You are dropped off from the meeting"

}

Scenario Outline (Invalid):

{

message: String e.g. "Meeting id doesn't exist"

}

Note: Meeting-id will be given from valid user and user will be dropped off from the meeting
Method Type: DELETE
Return Status: 200
URL: http://localhost:4000/user/user-id/meetings/meeting-id