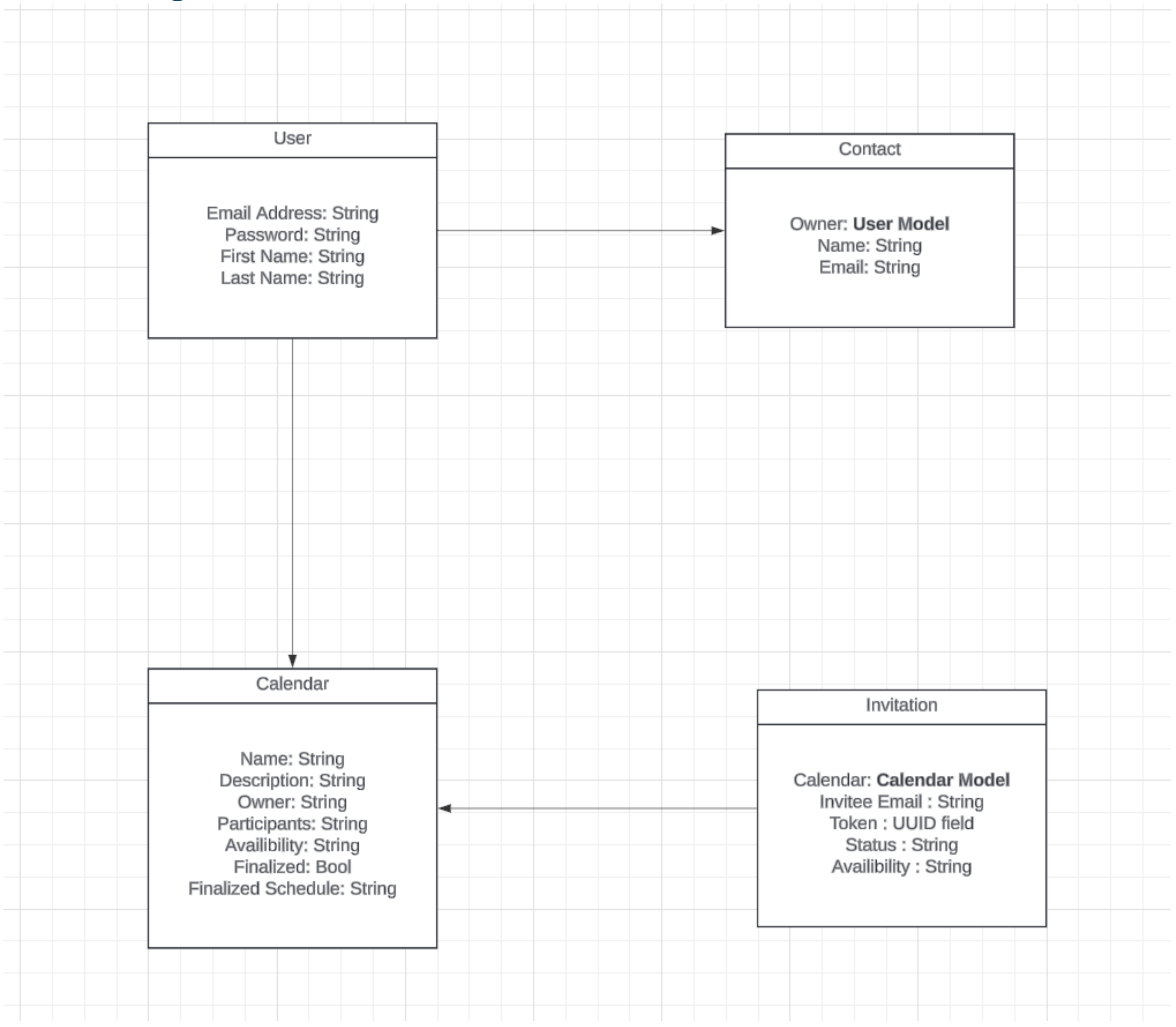
# CSC309 – P2
# API EndPoints

## Model Diagram:

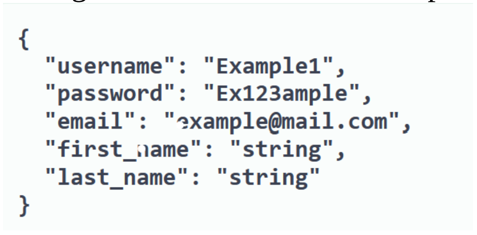


## Accounts:

### Register:

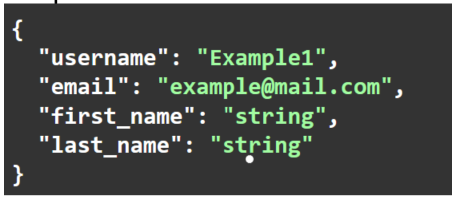
**POST:**/accounts/register/

To register/create an account. Expects a JSON request body with the following fields:



```
{
  "username": "Example1",
  "password": "Ex123ample",
  "email": "example@mail.com",
  "first_name": "string",
  "last_name": "string"
}
```

Response:

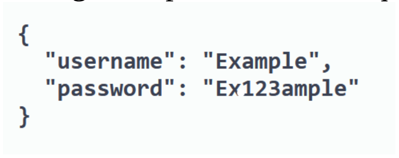


```
{
  "username": "Example1",
  "email": "example@mail.com",
  "first_name": "string",
  "last_name": "string"
}
```

### Login:

**POST**:/accounts/login/

To login. Expects a JSON request body with the following fields:



```
{
  "username": "Example",
  "password": "Ex123ample"
}
```

Response:

```
{
    "refresh":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoicmVmcmVzaCIsImV4cCI6MTcxMDQxO
TQ4MiwiaWF0IjoxNzEwMzMzMDgyLCJqdGkiOiI0YmZlZmRjZmM1MDY0M2U3ODJhMzQ3ZmM5MDk3ZmFhMCIsInVz
ZXJfaWQiOjV9.vdcceTvJ6eJ8TWRVuvAGBMMQb-EGMWVtF6E4hM-HQDw",
    "access":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90eXBlIjoiYWNjZXNzIiwiZXhwIjoxNzEwMzMzM
zgyLCJpYXQiOjE3MTAzMzMwODIsImp0aSI6ImEwOTE5YzYyMzM5ZjQ4Y2ZiZmZmODBmNGVkZTQ1NDM4IiwidXNl
cl9pZCI6NX0.5QBLOtEBYVdUgI6ewSPBgdRSiJuVLykYzScJrcmHoz0"
}
```

The token returned is required for authentication in other endpoints of the app, by putting ‘Bearer <Token>’ in the header

If a user inputs incorrect credentials, they will be met with a Bad Request error 400:

```
{
  "non_field_errors": [
    "Incorrect credentials"
  ]
}
```

## Logout:

**DELETE**:/accounts/logout/

To logout. The user should be authenticated. Response:

```
{
    "detail": "Logout successful"
}
```

If the user isn't authenticated, they will be met with Error 401 Unauthorized:

```
{
    "detail": "Authentication credentials were not provided."
}
```

## Add Contact:

**POST**:/accounts/profile/addcontact/

To add a contact. The user must enter both a name and email. Response:

```
HTTP 200 OK
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Contact added successfully",
    "contacts": {
        "Example Man": "example@example.example"
    }
}
```

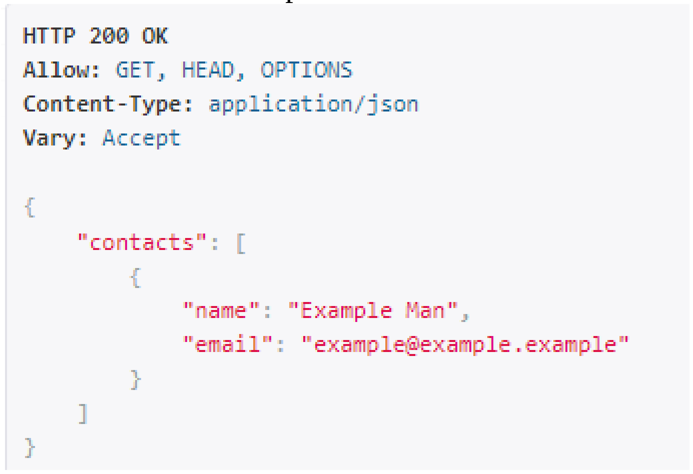
If at least one field isn't entered, the user will be met with Error 400 Bad Request:

```
HTTP 400 Bad Request
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Both fields are required"
}
```

## View Contacts:

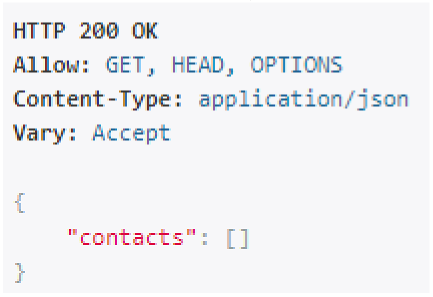
**GET**:/accounts/profile/contacts/

To view contacts. Response:



```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "contacts": [
        {
            "name": "Example Man",
            "email": "example@example.example"
        }
    ]
}
```

If no contacts exist, list will be empty:

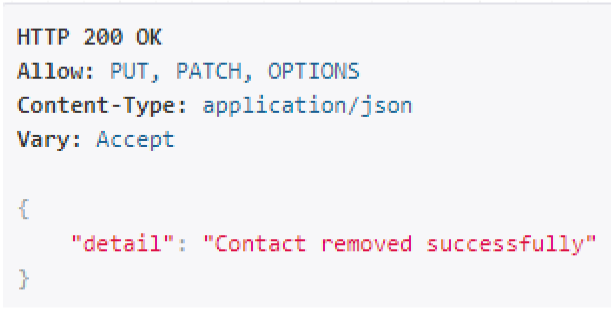


```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "contacts": []
}
```

## Delete Contact:

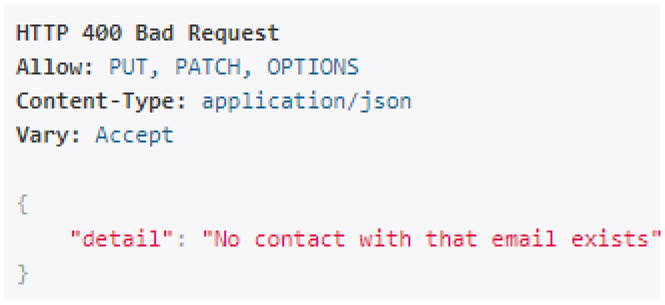
**POST**:/accounts/profile/deletecontact/

To delete contact. The user must enter an email. Response if successful:



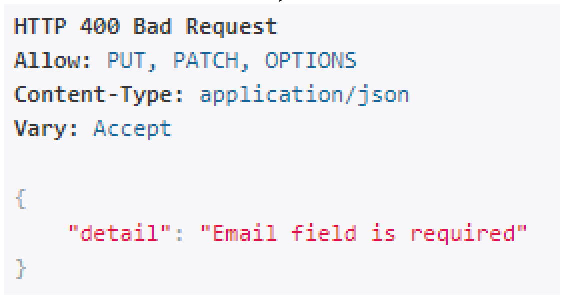
```
HTTP 200 OK
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Contact removed successfully"
}
```

If no contact is found with the given email, user will be met with Error 400 Bad Request and told there is no contact to remove:



```
HTTP 400 Bad Request
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "No contact with that email exists"
}
```

If no email is entered, user will be met with Error 400 Bad Request.



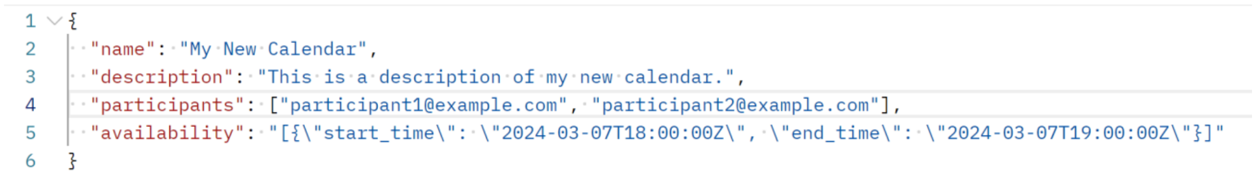
```
HTTP 400 Bad Request
Allow: PUT, PATCH, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Email field is required"
}
```

## Calendar:

### Create:

**POST**:/calendars/create/

To create a calendar. User should be authenticated. Expects a JSON request body with the following fields:

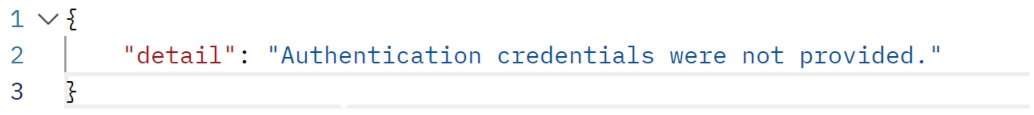


```
{
  "name": "My New Calendar",
  "description": "This is a description of my new calendar.",
  "participants": ["participant1@example.com", "participant2@example.com"],
  "availability": "[{\"start_time\": \"2024-03-07T18:00:00Z\", \"end_time\": \"2024-03-07T19:00:00Z\"}]"
}
```

An email will be sent to the invitees to choose an available time to meet.

The created calendar will be assigned an ID (used to view the calendar, see below).

If the user isn't authenticated, they will be met with Error 401 Unauthorized:

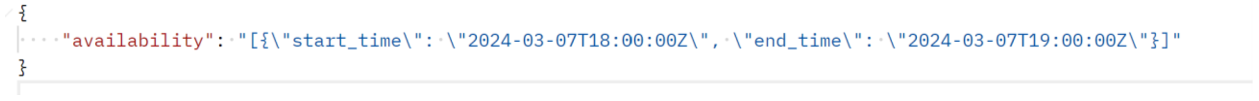


```
{
    "detail": "Authentication credentials were not provided."
}
```

### Update:

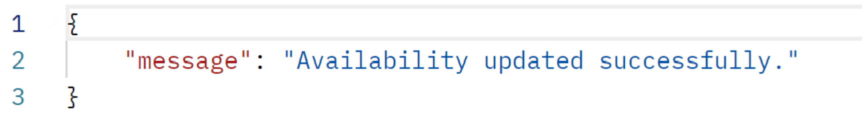
**PATCH** :/calendars/{id}/update-availability

For the Calendar host to choose their availability. The {id} is the id of their calendar. Expects a JSON request body with the following field for availability:



```
{
    "availability": "[{\"start_time\": \"2024-03-07T18:00:00Z\", \"end_time\": \"2024-03-07T19:00:00Z\"}]"
}
```

If successful, returns a 200 status and the following message:



```
{
    "message": "Availability updated successfully."
}
```

## View:

**GET**:/calendars/{id}/

To view a calendars information. The user should be authenticated. The response has information about the calendar, the invitees, and their response status:

```
{
    "id": 13,
    "name": "My New Calendar",
    "description": "This is a description of my new calendar.",
    "owner_username": "Rabah",
    "availability": [
        {
            "start_time": "2024-03-07T18:00:00Z",
            "end_time": "2024-03-07T19:00:00Z"
        }
    ],
    "invitations": [
        {
            "invitee_email": "participant1@example.com",
            "status": "Pending",
            "availability": "[]"
        },
        {
            "invitee_email": "participant1@example.com",
            "status": "Pending",
            "availability": "[]"
        },
        {
            "invitee_email": "participant2@example.com",
            "status": "Pending",
            "availability": "[]"
        },
```

If the user isn't authenticated, they will be met with Error 401 Unauthorized (only the creator of the calendar can view it):

```
{
    "detail": "Authentication credentials were not provided."
}
```

Trying to access a calendar that doesn't exist will be met with error 404 Not Found:

```
{
    "detail": "Not found."
}
```

## Recommendations:

**GET**: /calendars/<int:calendar_id>/recommendations/
Given a calendar ID, returns a list of 3 recommended calendars by scheduling each invitee into a timeslot based on the inviter's and the invitees' availability.

Return body contains a 2D array where each internal array represents a possible calendar combination. The time slots contain the email of the invitee scheduled in that time slot, and the start/end time of the slot. (In order to actually select a recommended calendar, see the “calendars/<int:calendar id>/finalize” endpoint below.)

```
[
  [
    {
      "invitee": "participant1@example.com",
      "meeting_times": [
        "2024-03-07T18:00:00",
        "2024-03-07T18:30:00"
      ]
    },
    {
      "invitee": "participant2@example.com",
      "meeting_times": [
        "2024-03-07T20:00:00",
        "2024-03-07T20:30:00"
      ]
    }
  ],
  [
    {
      "invitee": "participant1@example.com",
      "meeting_times": [
        "2024-03-07T18:30:00",
        "2024-03-07T19:00:00"
      ]
    },
    {
      "invitee": "participant2@example.com",
      "meeting_times": [
        "2024-03-07T20:30:00",
        "2024-03-07T21:00:00"
      ]
    }
  ],
  [
    {
      "invitee": "participant1@example.com",
      "meeting_times": [
        "2024-03-07T18:00:00",
        "2024-03-07T18:30:00"
      ]
    },
    {
      "invitee": "participant2@example.com",
      "meeting_times": [
        "2024-03-07T21:00:00",
        "2024-03-07T21:30:00"
      ]
    }
  ]
]
```

If a user cannot be scheduled for a meeting, returns “No available time slot” inside the meeting_times field. [
```
    {
        "invitee": "participant1@example.com",
        "meeting_times": [
            "2024-03-07T18:00:00",
            "2024-03-07T18:30:00"
        ]
    },
    {
        "invitee": "participant2@example.com",
        "meeting_times": "No available time slot"
    }
  ],
```

## Finalize:

**POST**: /calendars/<int:calendar_id>/finalize/
Given a calendar ID, sets the “*Finalized*” boolean to *True*, then populates the *finalized_schedule* field with the passed in schedule.
Requires auth, and the requesting user must be the owner of the calendar.

Expects a JSON request body with the following field. (See the “/calendars/1/recommendations” endpoint above for how to query recommended schedules, then use that as input for this endpoint)
```
{
  "finalized_schedule": [
    {
      "invitee": "participant1@example.com",
      "meeting_times": [
        "2024-03-07T18:00:00",
        "2024-03-07T18:30:00"
      ]
    },
    {
      "invitee": "participant2@example.com",
      "meeting_times": [
        "2024-03-07T20:00:00",
        "2024-03-07T20:30:00"
      ]
    }
  ]
}
```

If successful, returns with a 200 status and the following message.
```
{
  "message": "Calendar finalized successfully."
}
```

### Send Reminder:

**POST**: /calendars/send-reminder/
Sends an email reminder to the user with the specified email and for a calendar ID, reminding them to submit their availability, and with a unique link for them to do so.
Requires auth.

Expects a JSON request body with the following fields:

```
{
  “email”: “bobglob@gmail.com”,
  “calendar_id”: 1
}
```

If successful, returns with a 200 status and the following message.

```
{'message': 'Reminder sent to bobglob@gmail.com.'}
```

## Invitations:

### Update Invitee Availability:

**PATCH**:/calendars/invitations/{token}/update/
For the invitees to choose their availability. The {token} is sent with the link to their emails.
Expects a JSON request body with the following field for availability:

```
{
    "availability": "[{\"start_time\": \"2024-03-07T18:00:00Z\", \"end_time\": \"2024-03-07T19:00:00Z\"}]"
}
```

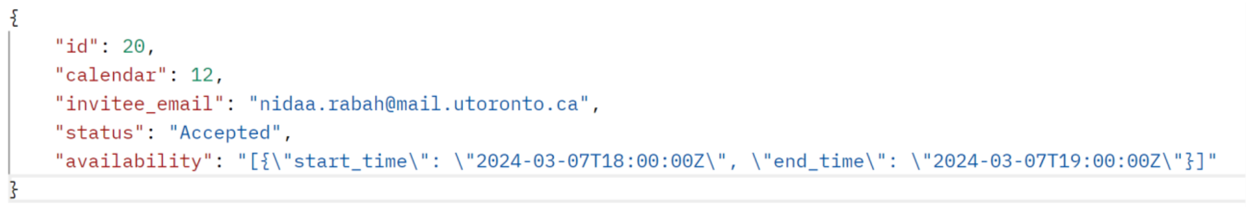
If successful, returns a 200 status and the following message:

```
{
    "message": "Availability updated successfully."
}
```

### View:

**GET**:/calendars/invitations/{token}/view/
For the invitees to view their response.

Returns a 200 status and the following information:



```
{
    "id": 20,
    "calendar": 12,
    "invitee_email": "nidaa.rabah@mail.utoronto.ca",
    "status": "Accepted",
    "availability": "[{\"start_time\": \"2024-03-07T18:00:00Z\", \"end_time\": \"2024-03-07T19:00:00Z\"}]"
}
```

In both update and view, if the token is incorrect, Error 404 Not Found will be raised.