Account

Register:

Endpoint: accounts/register/

Description: This endpoint allows users to register a new account. The request body should be in form-data format.

Method: POST

Fields/payload:

- username (text): The username of the new account.
- email (text): The email address of the new account.
- password (text): The password for the new account.
- password2 (text): The confirmation of the password

JSON responses:

- Successfully registered

```
{
    "message": "User created successfully."
}
```

- Duplicate Email

```
{
    "email": "Email already exists."
}
```

- Duplicate username

```
{
    "username": [
        "A user with that username already exists."
    ]
}
```

- Two password fields doesn't match

```
{
    "password": "Passwords must match."
}
```


- Invalid username


```
{
    "username": [
        "Enter a valid username. This value may contain only letters,
numbers, and @/./+/-/_ characters."
    ]
}
```


- Username, password, and repeat password are required.


```
{
    "username": [
        "This field may not be blank."
    ],
    "password": [
        "This field may not be blank."
    ]
}
```


- Missing email


```
{
    "email": "This field may not be blank."
}
```


- Missing fields


```
{
    "error": "missing fields"
}
```

Login:

Endpoint: accounts/api/login/
Description: This endpoint allows users to log in to their accounts. The request should be made with the raw request body type.
Method: POST
Fields/payload:

- username
- password

JSON response:

- Successfully login in

```
{
    "refresh": "...",
    "access": "..."
}
```

- Username or password invalid

```
{
    "error": "Invalid credentials"
}
```

- Missing fields

```
{
    "error": "missing fields"
}
```

Calendar

Create calendar:

Endpoint: /calendar/create/

Description: This endpoint simply initiates a new calendar for the currently logged in user. If the calendar is successfully created, the endpoint will return 200 status code. Otherwise, it will return a 403 status code, and the only case this can happen is when the user already has a calendar.

Method: POST

Update calendar:

Endpoint: /calendar/update/

Description: To make the backend more uniform and easier to use for other apps, this endpoint handles both updating the time block information as well as events. Distinguishing between the two types of calls is done by analyzing which field is unspecified/null. The accepted combinations of unspecified/null values are:

- permanent change of availability to not available: [event_id, yr, wk, pref]
- permanent change to the user's preference: [event_id, yr, wk]
- temporary change of preference: [event_id]
- delete temporary change of preference (payload values must match an existing time block, leaving pref unspecified or be null): [event_id, pref]
- an event takes place at this time block: [pref]

Please see the following examples for more details.

Method: POST

1. **Event related calls**
   To ensure that the user's temporary preference changes are kept intact when an event is scheduled in that time block (with custom preference/availability for that week and day only), that time block will not be removed and will persist in the database even when an event is registered in that time block.
   - If an event is created, for example, with ID 1 happening on Jan 8, 2024 at 13:00, please make the call to the endpoint with the following payload (`pref` field must be left as null or unspecified as the following example):

     ```
     {
           "event_id" : 1,
           "yr" : 2024,
           "wk" : 2,
           "day" : 1,
           "time" : 13.5
     }
     ```

     day is the day of the week, starting at 1 being Monday and 7 being Sunday
   - If an event is modified, make the same call as above with the new event's info

- If the event is deleted, you don't need to call this endpoint. The time block is deleted automatically by the CASCADE on_delete option

2. **Availability preference change related calls**
   a. **Permanent change to availability/preference (i.e., changing default preferences)**
      - **Modifying an existing time block's preference**
        For example, if I'm available on Wednesdays at 13:00 as one of my first choices, but now I want to rank this time block as my second choice, the call to the endpoint should have the following payload:

```
{
      "day" : 3,
      “time” : 13.0,
      “pref” : 2
}
```

      - **Making an available time block permanently unavailable**
        Using the previous example, if now I'm unavailable at that time, the call to the endpoint should have the following payload (i.e., setting pref to null or leave it out of the payload):

```
{
      "day" : 3,
      “time” : 13.0,
}
```

   b. **Temporary change to availability/preference**
      The steps are identical with the inclusion of the yr and wk now being required. So using the previous example again. Suppose I'm temporarily ranking it as second choice on Wednesday at 13:00 in the 10th week of 2024, the endpoint should be called with:

```
{
      “yr” : 2024,
      “wk” : 2,
      “day” : 1,
      “time” : 13.5,
      “pref” : 2
}
```

      Now suppose I no longer need this temporary change, i.e., reverting back to the default where this time is ranked as 1st choice. I can send the endpoint with the following payload:

```
{
      “yr” : 2024,
      “wk” : 2,
      “day” : 1,
      “time” : 13.5
}
```

The call for marking a time as unavailable is the same as the payload used to delete a temporary change of preference. Suppose I'm temporarily unavailable on that day. This should be the payload I am sending:

```
{
      “yr” : 2024,
      “wk” : 2,
      “day” : 1,
      “time” : 13.5
}
```

If however, I changed my mind and I am available or don't need the temporary change in preference anymore, I can delete the temporary change in preference by simply sending the above payload again.

In other words, the first time you send the above payload, i.e., when the backend database doesn't contain the entry for this time block on this specific day, the backend will treat it as you declaring that you are temporarily unavailable at that time.

If the time block does exist, the backend will treat it as "I want to remove this temporary change in availability/preference and delete the time block

---

View calendar:

Endpoint: /calendar/view/[yr]/[wk]

Description: This endpoint retrieves the time blocks of the given year and week number. These time blocks will be used to build the calendar the user will see in phase 3, which consists of a week view of 30 min increment time blocks. Only the time blocks that the user indicated preferences or has an event occupied will be reported. If a time block is not reported, the user is not available in that time block.

Method: GET

JSON response:

- First time block is the default preference
- Second one is where an event occupies that time block
- Third is a temporary change in time block preference

```
{
  "id": 1,
```

```
  "start_dt": "2024-03-03",
  "end_dt": "2024-03-09",
  "wk": 10,
  "timeblocks": [
      {
            "id": 1,
            "event": null,
            "time": 10.0,
            "yr": null,
            "wk": null,
            "day": 1,
            "preference": 1
      },
      {
            "id": 2,
            "event": 1,
            "time": 10.5,
            "yr": 2024,
            "wk": 11,
            "day": 2,
            "preference": null
      },
      {
            "id": 3,
            "event": 1,
            "time": 10.5,
            "yr": 2024,
            "wk": 10,
            "day": 5,
            "preference": 1
      }
  ]
}
```

Create Event:

Endpoint: /calendar/[yr]/[wk]/event/add/

Description: This endpoint allows users to create an event for the specified year and week. They must already have an account and they will already have a default calendar. The view requires the users to input a name for the event, an optional description, date of the event, a

start time, a duration (default=30mins) and finally select the participants for the event which can be none. If required fields are missing and error is shown.

Fields/payload: `name, description, date, start_time, duration, participants`

Validation errors:

- This field is required (Everything except description)
- Name already used
- Participant not found

Method: POST

JSON response:

```
{
    "message": "Event Created"
}
```

Edit Event:

Endpoint: /calendar/event/<event_id>/edit/

Description: This endpoint allows users to update an existing event. The exact same fields as the event creation view are presented with data retrieved. The same error handling occurs if required fields are missing. Upon successful event creation an automatic email is generated to all participants notifying of the update and the CalendarUpdateView endpoint is called.

Fields/payload: `name, description, date, start_time, duration, participants`

Validation errors:

- This field is required (Everything except description)
- Name already used
- Participant not found

Method: POST

JSON response:

```
{
    "message": "Email send successfully"
```

}

Event Reminder:

Endpoint: /calendar/event/<event_id>/remind/

Description: This endpoint allows users to send a reminder to participants that have **not** selected their default preference. Upon successful redirection an auto generated email with a link is sent to the participants with no default preference. This link will be unique to the receiver prompting them to login in order to access the calendar update page unique to their account. The user can send reminders until all participants have provided preferences.

Method: POST

JSON response:


```
{
    "message": "Reminder sent successfully.",
    "finalized_schedule": {
        "date": "2024-03-12",
        "start_time": "13:00"
    }
}
```

Event List:

Endpoint: /calendar/event/list/

Description: This endpoint allows users to see all upcoming events linked to their account. This endpoint is mainly for our frontend page that shows users all upcoming events.

Method: GET

JSON response:


```
[
    {
        "id": 1,
        "name": Meeting with Team A,
```

```
        "description": This is a description,
        "date": 2024-01-01,
        "start_time": 23:23,
        "duration": 30,
        "participants": mustafa, daniel, donna
    },
    {
        "id": 2,
        "name": Meeting with Team B,
        "description": This is a description,
        "date": 2024-02-02,
        "start_time": 22:22,
        "duration": 35,
        "participants": mustafa, donna
    },
    {
        "id": 3,
        "name": Meeting with Team C,
        "description": This is a description,
        "date": 2024-03-03,
        "start_time": 21:13,
        "duration": 40,
        "participants": murphy
    },
]
```

Suggested schedule:

Endpoint: /calendar/view/<event_id>/suggested_schedule

Description: This endpoint retrieves the suggested schedule for a specific event. It assumes all participants have accounts and have set their default preferences upon event creation. So the case for someone who still hasn't submitted their availabilities will be handled when the event is created. When invitees are invited, our web app automatically generates a suggested schedule based on the availability of all participants. It then returns this schedule, ordered from the most to the least preferred, according to participants' preferences marked as either first or second choices. Also, the error is shown when all availabilities are conflicted.

Method: GET

JSON response:

- Suggested schedule

```
{
    "suggestions": [
        {
            "date": "2024-03-12",
            "yr": 2024,
            "wk": 11,
            "day": 2,
            "start_time": 6.0,
            "end_time": 6.5
        },
        {
            "date": "2024-03-12",
            "yr": 2024,
            "wk": 11,
            "day": 2,
            "start_time": 6.5,
            "end_time": 7.0
        }
    ]
}
```


- No available schedule


```
{
    "error": "No available schedules because some participants are not
available."
}
```


- User unauthorized (did not login)


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


- User who logged in but is not the event owner can’t view the suggested schedule


```
{
    "error": "You do not have permission to view the suggested schedules
for this event."
}
```


- No such event

```
{
    "error": "Event not found."
}
```

Finalize schedule:

Endpoint: /calendar/view/<event_id>/finalize_schedule

Description: This endpoint is used to finalize a schedule for a specific event.

Method: POST

Fields/payload:

- date: the date for the event takes place
- start_time: the start_time the user want to finalize the event

JSON response:

- Successfully select a schedule and finalize it for the event

```
{
    "message": "Schedule finalized successfully.",
    "finalized_schedule": {
        "date": "2024-03-12",
        "start_time": "06:00"
    }
}
```

- Not select the schedule provided

```
{
    "error": "No selected schedule provided."
}
```

- Same JSON responses as suggested_schedule
    - User unauthorized (did not login)
    - User who logged in but is not the event owner can't finalize event
    - No such event

Contacts

Create Contact:

Endpoint: /contacts/create/

Description: This endpoint allows authenticated users to create a new contact entry. The request body should contain the contact details in JSON format.

Method: POST

Fields/Payload:

- name(text): the name of the new contact
- email(text): the email address of the new contact
- phone (text): the phone number of the contact
- preferred_contact(text): the preferred method of contact
- Accessibility (text, optional): any accessibility requirements
- pronouns(text, optional): the pronouns of the new contact
- tag_line(text, optional): a tag associated with the contact
- notes(text, optional): any notes for the new contact

JSON responses:

- Successfully created new contact


```
{
    “Message”: “Contact created successfully.”
}
```


- User unauthorized (did not login)


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


- Validation error (missing required field):


```
{
```

```
    "name": [
        "This field may not be blank."
    ]
}
```


List Contacts:

Endpoint: /contact/

Description: This endpoint retrieves all contacts created by the authenticated user

Method: GET

JSON response:

- Example of listing contacts:


```
[
    {
        "name": "John Doe",
        "email": "john@example.com",
        "phone": "123-456-7890",
        "preferred_contact_method": "Email",
        "accessibility": "",
        "pronouns": "he/him",
        "tag_line": "UofT."
    },
    {
        "name": "Jane Smith",
        "email": "jane@example.com",
```

```
        "phone": "098-765-4321",
        "preferred_contact_method": "Phone",
        "accessibility": "Requires wheelchair access",
        "pronouns": "she/her",
        "tag_line": “UofT"
    }
]
```


- User unauthorized(did not log in)


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


Contact Detail:

Endpoint: /contact/<int:id>/

Description: This endpoint allows users to retrieve, update or delete a specific contact entry.

The actions depend on the method used: GET for retrieval, PUT for update, DELETE for removal.

Methods: GET, PUT, DELETE

JSON response:

- GET JSON response


```
{
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "123-456-7890",
```

```
    "preferred_contact_method": "Email",
    "accessibility": "",
    "pronouns": "he/him",
    "tag_line": "Friend from college."
}
```


- PUT JSON response (on successful update):


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


- DELETE JSON response (on successful deletion):


```
{
    "message": "Contact deleted successfully."
}
```


- User unauthorized (did not login)


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


- No such contact:


```
{
    "error": "Contact not found."
}
```

A UML diagram of out backend infrastructure: