Models

Apps: accounts

User (django default user)

Apps: profile_info

Dashboard

- **owner:** points to Django's user model
- **welcome message** (a string that does not exceed 300 char)
- **language** (a string that does not exceed 25 char)
- **country** (a string that does not exceed 50 char)
- **timezone** (a string that does not exceed 25 char)

Apps: calendars

Calendar:

- **name:** A concise title or name for the calendar (maximum length of 50 characters)
- **description**: A brief explanation or summary of the calendar's purpose (maximum length of 120 characters)
- **owner:** id of the owner of the calendar (Null means calendar belongs to invitee)
- **contacts**: a list of contacts associated with a particular calendar

Availability:

- **date**: DateField, the date of an available meeting interval
- **start_time:** TimeField, the start time of an available meeting interval
- **end_time:** TimeField, the end time of an available meeting interval
- **preference:** CharField with max length of 10, indicates if this meeting interval is HIGH_PREFERENCE, MEDIUM_PREFERENCE, LOW_PREFERENCE
- **invitee:** Foreign key to invitation model, points to a invitation this availability refers to (delete on cascade)
- **calendar:** ForeignKey to Calendar model, available times for this particular calendar
- **owner:** ForeignKey to User model, represents the owner of this availability

Meeting:

- **name:** The title or name of the booked meeting, CharField with a maximum length of 50 characters.
- **description:** A brief description or agenda for the booked meeting, CharField with a maximum length of 120 characters.
- **Calendar:** The calendar that this meeting is for, foreign key to calendar
- **Contacts:** Represents the contacts associated with this meeting, ManyToManyField to the Contact model
- **duration**: planned length of the meeting, DurationField
- **last_modified:** The date and time when the booked meeting details were last modified, DateTimeField

- **start_date**: the earliest date the meeting should occur, TimeField
- **end_date**: the latest date the meeting should occur, TimeField
- **confirmed:** BooleanField , True iff meeting is confirmed
- **date:** date of the meeting, DateField

BoundedTime:

- start_time: the daily start time for the time period that the start time of the meeting could take place within
- end_time: the daily end time for the time period that the start time of the meeting could take place within
- duration: the duration of the meeting that user plans to organize
- models.DurationField()
- start_date: the start date for the date period that the start date of the meeting could take place within
- end_date: the end date for the date period that the end date of the meeting could take place within

Invitation

- **Meeting:** ForeignKey to meetings (the meeting this invite is for)
- **Invitee:** ForeignKey to contacts (person being invited)
- **Confirmed**: BooleanField if invitee has filled out the RSVP

SuggestedMeeting

- **Owner_availability:** IntegerField, represents teh ID of the owner's availability associated with this meeting.
- **Invitee_availability:** IntegerField, represents the ID of the invitee's availability associated witht this suggested meeting
- **Invitee:** ForeignKey to the User model (delete on cascade), represents the user invited to this suggested meeting
- **Start_time:** TimeField, represents the start time of the suggested meeting
- **End_time:** TimeField, represents the end time of the suggested meeting
- **Date:** DateField, represents the date of the suggested meeting
- **Owner_preference:** CharField with a maximum length of 10 characters, represents the preference level of the owner for this suggested meeting

SuggestedSchedule

- **Bounded_time:** ForeignKey tto the BoundedTime model (delete on cascade), represents the bounded time associated with this suggested schedule
- **Suggested_meeting:** ForeignKey to the SuggestedMeeting model (delete on cascade), represents the suggested meeting associated wihth this suggested schedule.

Apps: contacts

Contact

- **User:** points to Django's user model (delete on cascade)
- **First_name:** first name of the contact, 'CharField' with maximum length of 255 characters
- **Last_name:** last name of contact, 'CharField' with maximum length of 255 characters
- **Email:** email address of a contact, 'EmailField' type ensures entered value is a valid email address
- **Phone_number:** phone number of the contact, 'CharField' with maximum 15 characters. Is optional, so can be left blank. If no value is provided, will be stored as NULL in the database

Authentication

Endpoint: `accounts/register/`
Description:

- Endpoint where new users can register for a new account.
- Performs the following valiations:
    - If password and password2 are equal
    - If user is already registered
    - If passwords have length < 8
    - If email is valid

Methods: POST
Field/Payload: username, password, password2, email, first_name, last_name

- Optional fields: email, first_name, last_name

Endpoint: `accounts/login/`
Description:

- Endpoint where can log in
- If unable to authenticate, return "No active account found with the given credentials", Status 401

Methods: POST
Field/Payload: username, password

Endpoint: `accounts/logout/`
Description:

- Endpoint where can log out
- Refresh token is sent as payload to move it into blacklist, so that new access tokens can not be generated after logging out

Methods: POST
Field/Payload: refresh

Endpoint: `api/token/refresh`
Description:

- Refreshes the current access token so that user does not have to log in again
- Receives refresh token in request data and returns a new access token if refresh token is valid

Methods: POST
Field/Payload: refresh

Profile

Endpoint: `profile/view/`
Description:

- Endpoint to get user information

Methods: GET
Field/Payload: username, email, first_name, last_name

Endpoint: `profile/edit/`
Description:

- Endpoint to edit user information
- Performs the validations:
  - If password and password2 are equal
  - If passwords have length > 8
  - If password not given, ignore the change

Methods: PUT
Field/Payload: email, first_name, last_name, password, password2

Endpoint: `profile/dashboard/view`
Description:

- Endpoint to get information about their user dashboard

Methods: GET
Field/Payload: language, country, timezone, message

Endpoint: `profile/dashboard/edit`
Description:

- Endpoint to update information about their user dashboard

Methods: PUT
Field/Payload: language, country, timezone, message

Calendars

Endpoint: `calendars/`
Describe: get details of the calendars

- If no calendar, respond with empty JSON
- If there are multiple calendars, respond with each calendar's name and description

Methods: GET
Authentication Required: NO
Field/Payload: `name, description`

Endpoint: `calendars/create/`
Describe: create a calendar
Methods: POST
Authentication Required: YES
Field/Payload:

- name (string): Name of the calendar.
- description (string): Description of the calendar.

```
SAMPLE PAYLOAD:
{
    "name": "conference",
    "description": "AI"
}
```

Endpoint: `calendars/calendar/<int:id>`
Describe: Retrieve details of a calendar by its id
Methods: GET
Authentication Required: YES
Field/Payload: None

```
SAMPLE QUERY: "http://127.0.0.1:8000/calendars/calendar/1/"
```

Endpoint: `calendars/<int:id>/availability/select/`
Describe: Add time slots to the calendar with calendar id
Methods: POST

Authentication Required: YES
Field/Payload:

- availability_set (array of objects): Array containing time slot objects with the following fields:
    - date (string): Date of the time slot (format: YYYY-MM-DD).
    - start_time (string): Start time of the time slot (format: HH:MM:SS).
    - end_time (string): End time of the time slot (format: HH:MM:SS).
    - preference (string): Preference level of the time slot (high, medium, or low).

```
SAMPLE PAYLOAD:
{
  "availability_set": [
    {
      "date": "2024-03-15",
      "start_time": "09:00:00",
      "end_time": "10:00:00",
      "preference": "high"
    },
    {
      "date": "2024-03-16",
      "start_time": "14:00:00",
      "end_time": "15:30:00",
      "preference": "medium"
    },
    {
      "date": "2024-03-17",
      "start_time": "10:30:00",
      "end_time": "12:00:00",
      "preference": "low"
    }
  ]
}
```

Endpoint: `calendars/<int:id>/contacts`
Describe: adding contacts to the meeting

- Get: get the list of contacts

Methods: GET
Field/Payload: contacts

Endpoint: `calendars/<int:id>/contacts/add/`
Describe: adding contacts to the meeting

- Post: Add contact to the list

Methods: POST
Field/Payload: id

```
Sample Payload
{
   "contacts": [
       {
           “id:” 1
       },
       {
          “id:” 2
       }
   ]
}
```

Endpoint: `calendars/<int:id>/meetings/suggest_schedules/`
Describe:

- Get: returns a list of suggested schedules
- Post: user indicates the suggested schedule choice
    - Django willi then create meeting models based on each scheduled meeting

Methods: GET, POST
Field/Payload: id

- Id = the id of the suggested schedule

Endpoint: `calendars/<int:id>/meetings/invite/<int:invite_id>/`
Describe: page where invitation with invite-id gets to respond

- GET: gets the times that the user is available, and shows the user ID that is inviting
- POST: responds with invitee availability (in similar format to `calendars/<id>/meetings/create/)`

Methods: POST, GET
Field/Payload:

- availability_set (array of objects): Array containing time slot objects with the following fields:
    - date (string): Date of the time slot (format: YYYY-MM-DD).
    - start_time (string): Start time of the time slot (format: HH:MM:SS).
    - end_time (string): End time of the time slot (format: HH:MM:SS).
    - preference (string): Preference level of the time slot (high, medium, or low).

Endpoint: `calendars/<int:id>/meetings/invite/status/`
Describe: page where user gets to see who has responded and who hasn't
- GET

Methods: GET
response:
```
{
    "responsed": [
        {
            "first_name": "andrew",
            "last_name": "mc"
        }
    ],
    "not_responsed": []
}
```

Endpoint: `calendars/<int:id>/meetings/invite/remind/`

Describe: remind users who have not responded
- POST

Methods: POST
response:
```
{
    "users_reminded": []
}
```

Contacts

Endpoint: `contacts/contacts_index/`
Description:
- Displays a list of contacts for the logged-in user
- GET to retrieve the contacts

Methods: GET
Field/Payload: none

Endpoint: `contacts/add_contact/`
Description:
- Endpoint that allows users to add a new contact.
- Uses POST to submit the form and add a new contact
    - Phone_number = optional field
    - Will check to ensure non-duplicates of first_name, last_name, email

Methods: POST

Field/Payload: first_name, last_name, email, phone_number

Endpoint: `contacts/edit_contact/<int:id>/`
Description:

- Endpoint that allows logged-in user to edit existing contacts.

Methods: PUT
Field/Payload: first_name, last_name, email, phone_number

Endpoint: `contacts/delete_contact/<int:id>/`
Description:

- Endpoint that allows logged-in user to delete existing contacts.

Methods: DELETE
Field/Payload: none