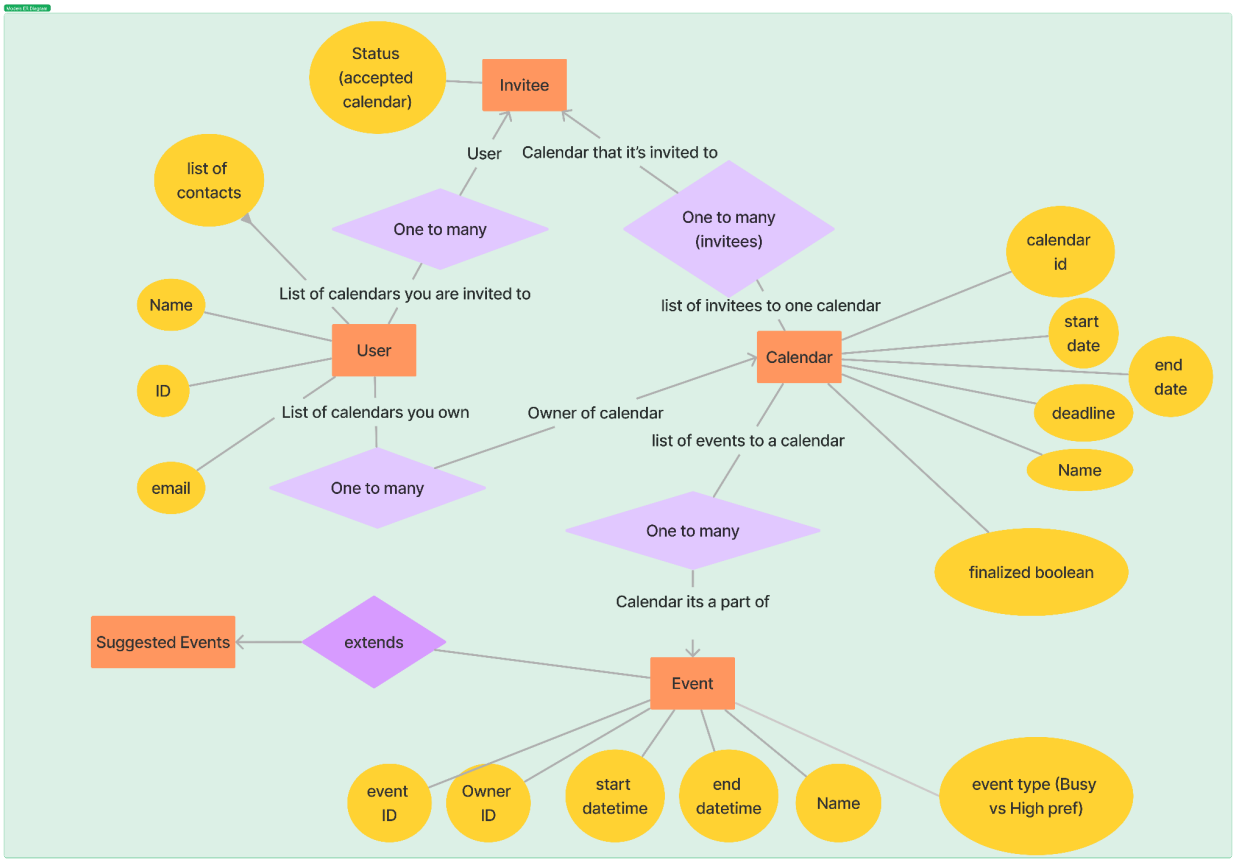
CSC309: OneOnOne API Documentation

Chloe Lam, Roy Lin, Koji Wong, Iris Myung

ER Diagram



[Link to Full Figma Diagrams]

Models

- CustomUser(AbstractUser)
    - Username
    - Password
    - Email
    - First_name
    - Last_name
    - List of contacts (other CustomUsers)
- Attendee(Model)
    - Status (for checking if they uploaded their info)
    - Accepted (for accepting calendar invites)
    - User (which User this attendee is)
    - Calendar (which calendar is tied to this Attendee)
- Calendar(Model)
    - Name
    - Description
    - Start_date (when does the calendar start)
    - End_date (when does it end)
    - Deadline (when do the invitees need to submit their availabilities by)
    - Owner (which user owns this calendar)
    - Is_finalized (checking if the calendar is finalized or not)
- Event(Model)
    - Name
    - Start_time (when does this event start)
    - End_time (when does this event end)
    - Last_modified
    - Calendar (which calendar does this event belong to)
    - Owner (who owns this event)
    - Event_type (high preference [HP], low preference [LP], or finalized [FN])

API Endpoints

User Model

/users/register/
Register a new user.
Method: POST
Fields/payload: username, password1, password2, email, first_name, last_name
Example Payload:

```
{
    “username” : “user1”,
    “email” : “user1@gmail.com”,
    “first_name” : “user1_first_name”,
    “last_name” : “user1_last_name”,
    “password1” : “12345678”,
    “password2”: “12345678”,
}
```

Validation errors:

- The username field is required.
- The password field is required.
- The password field is required.
- The first name field is required.
- The last name field is required.
- The email field is required.
- The passwords do not match.
- The password must be at least 8 characters long.
- A user with that username already exists.
- A user with that email already exists.

/**users/login/**
Note: implemented with Django Rest Framework's TokenObtainPairView.
Login the user.
Methods: POST
Fields/payload: username, password
Example Payload:

```
{
    “username” : “user1”,
    “password” : “12345678”,
}
```

Validation errors:

- The username field is required.
- The password field is required.
- A user with that username does not exist.

- The password is incorrect.

/users/logout/
Log out the user.
Methods: POST
Fields/payload: refresh_token
Validation errors: None

/users/profile/view/
Get the user's profile.
Methods: GET
Fields/payload: None
Validation errors: None

/users/profile/edit/
Edit the user's profile.
Methods: POST
Fields/payload: first_name, last_name, email, password1, password2
Example Payload:

```
{
        “email” : “edited_user1@gmail.com”,
        “first_name” : “new_user1_first_name”,
        “last_name” : “new_user1_last_name”,
        “password1” : “A12345678”,
        “password2”: “A12345678”,
}
```

Validation errors:
- The passwords do not match.
- The password must be at least 8 characters long.
- A user with that username already exists.
- A user with that email already exists.

Note: The user’s password only updates if password1 and password2 both exist, are equal, and valid in length.

/users/contacts/view/
View all of the user's contacts.
Methods: GET
Fields/payload: None
Validation errors: None

/users/contacts/add/
Add a contact to the user's profile (via email).
Methods: POST
Fields/payload: contact_email

Example Payload:

```
{
        “contact_email” : “invitee1@gmail.com”,
}
```

Validation errors:

- The contact email field is required.
- A user with that email does not exist.
- You cannot add yourself as a contact.
- You cannot add the same contact twice.

/users/contacts/remove
Remove a contact from the user's profile (via email).
Methods: POST
Fields/payload: contact_email
Example Payload:

```
{
        “contact_email” : “invitee1@gmail.com”,
}
```

Validation errors:

- The contact email field is required.
- A user with that email does not exist.
- You cannot remove yourself as a contact.
- You cannot remove a contact that is not in your contacts.

/users/<int:contact_id>/profile/
View the profile of a specific contact.
Methods: GET
Fields/payload: contact_email
Example Payload:

```
{
        “contact_email” : “invitee1@gmail.com”,
}
```

Validation errors:

- Contact not found
- Contact is not in your contacts list

/users/accept/<int:attendee_id>/
An attendee can accept a calendar event
Method: POST
Payload: None

/users/calendars/view/
List all the calendars of the owner
Method: GET
Payload: None

/users/invitedcalendars/view/unaccepted/
View all of the user's unaccepted calendars they are invited to.
Methods: GET
Payload: None
Validation Errors: None

/users/invitedcalendars/view/accepted/
View all of the user's accepted calendars they are invited to.
Methods: GET
Payload: None
Validation errors: None

/users/invitedcalendars/view/finalized/
View all of the user's calendars.
Methods: GET
Payload: None
Validation errors: None

/users/<str: user_email>/id/
Get a user's id by their email
Methods: GET
Payload: user_email
Validation errors: None

/users/<int: user_id>/nameAndEmail/
Get a user's name and email by their user id
Methods: GET
Payload: user_id
Validation errors: None

Calendar Model

/calendars/create/
Creates calendar with name, description, deadline, start_date, end_date
Method: POST
Example Payload:

```
{
    "name": "calendar 100",
    "description": "example description",
    "deadline": "2023-04-15"
    "start_date": "2023-04-16",
    "end_date": "2023-04-20",
}
```

/calendars/<int:calendar_id>/view/
View calendar by calendar_id
Method: GET
Payload: None, gets by calendar_id in URLConf

/calendars/<int:calendar_id>/edit/
Method: POST
Payload:

```
{
    "name": "new calendar name",
    "description": "new description",
    "deadline": "2023-06-15"
    "start_date": "2023-06-16",
    "end_date": "2023-06-20",
}
```

/calendars/events/add/
Add availability to a calendar, where events represent the times of availability
Method: POST
Payload:

```
{
    "Start_time": 2024-03-10T10:20:00Z,
    "end_time":  2024-03-12T10:50:00Z,
    “Event_type”: HP (High Priority)
    “Calendar”: 1
    "name": "test event",
}
```

/calendars/invite/
Invites a user to a calendar
Method: POST
Payload:

```
{
    “User”: user_id,
    “Calendar”: calendar_id
}
```

/calendars/int:calendar_id>/list/status/
List invite statuses for attendees of a calendar
Method: GET
Payload:

```
[
 {
    "user_id": 1,
    "username":  invitee1,
    “status”: true,
 },
 {
    "user_id": 2,
    "username":  invitee2,
    “status”: false,
 }
]
```

/calendars/<int:calendar_id>/event/<int:event_id>/update/
Add availability to a calendar, where events represent the times of availability
Method: POST
Payload:

```
{
    “event_id” : 1,
    “name” : “test event 2”,
    "start_time": 2024-03-10T10:20:00Z,
    "end_time": 2024-03-12T10:30:00Z ,
    “event_type”: HP, (High Priority)
    “calendar”: 1,
}
```

/calendars/<int:calendar_id>/contact/<user_id>/view/
Get availability of a specified contact
Method: GET
Payload: None

/calendars/<int:attendee_id>/reminder/
Send a reminder email to an attendee of a calendar event
Method: POST
Payload: attendee_id

/calendars/<int:calendar_id>/suggested/
Retrieve suggested events for a calendar
Method: GET
Payload: calendar_id

/calendars/<int:calendar_id>/finalize/
Finalizes a calendar by creating and saving the final meeting events to the database.
Method: POST
Example Payload:

```
{
        events: [
                {
                        start_time: 2024-03-15T12:20:00Z ,
                        end_time: 2024-03-15T12:40:00Z,
                        owner: 1 ,
                        calendar: 2,
                        event_type: FN
                }
        ]
}
```

/calendars/<int:calendar_id>/get_events/
Get a list of all events associated with the specified calendar
Method: GET
Payload: calendar_id

/calendars/<int:calendar_id>/get_events/<int:invitee_id>
Retrieve the events added by the specified invitee for the given calendar
Method: GET
Payload: calendar_id, invitee_id

/calendars/events/<int:event_id>/delete/
Delete an event from the calendar
Method: DELETE
Payload: event_id