


Database

👤 Created by	 Ray Hung
🕒 Created time	@March 13, 2024 2:44 PM
🏷️ Tags	

Profile Model

user (foreign key to auth.user): Mapping to the user who owns this profile

profile_picture (Image): Image field that represents user's profile picture

Calendar Model

owner (foreign key to auth.user): Mapping to the user who owns this calendar

name (Text): Name of the calendar, nonNull

start_date (Date): Start date of the calendar, nonNull

end_date (Date): End date of the calendar, nonNull

finalized (Boolean): Whether or not the calendar has a finalized schedule

Invitee Model

calendar (foreign key to calendar): Mapping to the calendar which stores this invitee

invitee (foreign key to auth.user): Mapping to the user that was invited to this calendar

deadline (DateTime): The deadline for this invitee to respond, nonNull

Availability Model

calendar (foreign key to calendar): Mapping to the calendar which stores this availability

user (foreign key to auth.user): Mapping to the user who provided this availability for this calendar

start_period (DateTime): Start date and time for this availability provided by user, nonNull

end_period (DateTime): End date and time for this availability provided by user, nonNull

preference (Integer): One of the three options in (1, 2, 3), 1 being not preferred, 3 being most preferred

Meets Model

calendar (foreign key to calendar): Mapping to the calendar which stores this meeting time

meeter (foreign key to auth.user): Mapping to the user that the owner of the calendar is meeting

start_period (DateTime): Start time for the meeting with this user by the owner of the calendar

end_period (DateTime): End time for the meeting with this user by the owner of the calendar

Contacts Model

requester (foreign key to auth.user): Mapping to the user that has made the contacts request

requested (foreign key to auth.user): Mapping to the user that receives this contacts request

accepted (Boolean): Signaling whether or not this contact has been established

Authentication	none
Source Code	shell

OneOnOne API

```
# Install the command line client
$ pip install coreapi-cli
```

accounts

login > create

POST /accounts/login/

Interact

Login the user with given credentials

Input Format

```
{ "username": <username>, "password": <password> }
```

Responses

- 200 OK - Successful login
- 400 Bad Request - Any field is missing or invalid

Output Format when successful

```
{ "refresh": <refresh-token>, "access": <access-token> }
```

Output Format when unsuccessful

```
{ "non_field_errors": <error_message> }
```

With the following error messages:

- User not activated
- Username and password do not match
- Both fields are required

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
username required	
password required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action accounts login create -p username=... -p password=...
```

profile > list

GET /accounts/profile/

 Interact

Get the profile details for the logged-in user through the access token

Responses

200 OK - Successful retrieval of profile

401 Unauthorized - Access token is invalid

Output Format when successful

```
{ "user": { "id": <user-id>, "username": <username>, "first_name": <first-name>,
"last_name": <last-name>, "email": <email> }, "profile_picture": <profile-picture-path>, }
```

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action accounts profile list
```

profile > update

PUT /accounts/profile/

 Interact

Update the profile with given data shown below, only the fields that are given will be updated. If a user wants to update the password, the `current_password`, `new_password` and `confirm_password` fields are required.

Input Format

```
{ "first_name": <first-name>, "last_name": <last-name>, "email": <email>,
"current_password": <password>, "new_password": <confirm-password>, "confirm_password":
<confirm-password>, "profile_picture": <profile-picture-upload> }
```

Responses

200 OK - Successful update to profile

401 Unauthorized - Access token is invalid, or certain fields are invalid

Output Format when successful

```
{ "user": { "id": <user-id>, "username": <username>, "first_name": <first-name>,
"last_name": <last-name>, "email": <email> }, "profile_picture": <profile-picture-path>, }
```

Output Format when unsuccessful

```
{ "non_field_errors": <error_message> }
```

With the following error messages:

- Enter current password first

- Current password is incorrect
- Password must match

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action accounts profile update
```

register > create

POST /accounts/register/

⇌ Interact

Register a new user with given user data

Input Format

```
{ "username": <username>, "first_name": <first-name>, "last_name": <last-name>, "email":
<email>, "password": <password>, "confirm_password": <confirm-password> }
```

Responses

201 Created - Successful creation

400 Bad Request - Any field is missing or invalid

Output Format when successful

```
{ "username": <username>, "first_name": <first-name>, "last_name": <last-name>, "email":
<email> }
```

Output Format when unsuccessful

```
{ "<field-name>": <error_message> }
```

With the following error messages:

- This field is required
- A user with username already exists

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
username required	Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
first_name required	
last_name required	
email required	
password required	
confirm_password required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action accounts register create -p username=... -p first_name=... -p last_name=...
```

api

token > create

POST /api/token/

⇌ Interact

Takes a set of user credentials and returns an access and refresh JSON web token pair to prove the authentication of those credentials.

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
username required	
password required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action api token create -p username=... -p password=...
```

token > refresh > create

POST /api/token/refresh/

⇌ Interact

Takes a refresh type JSON web token and returns an access type JSON web token if the refresh token is valid.

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
refresh required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action api token refresh create -p refresh=...
```

calendars

list

GET /calendars/

 Interact

Obtain the list of all calendars created by the currently authenticated user

Response

200 OK - Success
401 Unauthorized - Not authenticated

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars list
```

create

POST /calendars/

 Interact

Create a new calendar

Input Format

{ "name": "<calendar-name>", "start_date": "<start-date>", "end_date": "<end-date>" }

Response

200 OK - Success
400 Bad Request - Invalid input
401 Unauthorized - Not authenticated

Output Format when successful

{ "id": <calendar-id>, "name": <calendar-name>, "start_date": <start-date>, "end_date": <end-date>, "finalized": <true/false>, "owner": <owner-userid> }

Output Format when unsuccessful

{ "<field-name>": "<error-message>" }

With the following error messages:

- This field is required

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
name required	A name for the calendar
start_date required	The starting date for the calendar

Parameter	Description
end_date required	The ending date for the calendar

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars create -p name=... -p start_date=... -p end_date=...
```

read

GET /calendars/{id}

⇌ Interact

Obtain calendar details

Response

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
id required	A unique integer value identifying this calendar.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars read -p id=...
```

update

PUT /calendars/{id}

⇌ Interact

Updates the name of a calendar, which is defined by the serializer.

Users cannot modify a start or end date once the calendar has been created.

The calendar is identified by the API endpoint's parameter `pk`.

Input Format

```
{ "name": "<new-calendar-name>" }
```

Response

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

Output Format when successful

```
{ "name": "<new-calendar-name>" }
```

Output Format when unsuccessful

```
{ "name": "<error-message>" }
```

With the following error messages:

- This field is required

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
id required	A unique integer value identifying this calendar.

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
name required	A name for the calendar
start_date required	The starting date for the calendar
end_date required	The ending date for the calendar

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/
```

```
# Interact with the API endpoint
$ coreapi action calendars update -p id=... -p name=... -p start_date=... -p end_date=...
```

delete

DELETE /calendars/{id}

⇌ Interact

Deletes the calendar, and removes all associated content with the calendar

The calendar is identified by the API endpoint's parameter `pk`

It does not return any message upon successful in the JSON response

Response

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
id required	A unique integer value identifying this calendar.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars delete -p id=...
```

finalize

PUT /calendars/{id}/finalize/

 Interact

Finalizes the calendar, preventing any further changes to the calendar

Does not require any input

Reponses

200 OK - Success

400 Bad Request - Already finalized

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

Response when successful

```
{ "detail": "Calendar has been successfully finalized." }
```

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
id required	A unique integer value identifying this calendar.

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
name required	A name for the calendar
start_date required	The starting date for the calendar
end_date required	The ending date for the calendar

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars finalize -p id=... -p name=... -p start_date=... -p end_date=...
```

availabilities > list

GET /calendars/{calendar_id}/availabilities/

 Interact

Obtain the list of all availabilities provided for a calendar.

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner or invitee
- 404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars availabilities list -p calendar_id=...
```

availabilities > create

POST /calendars/{calendar_id}/availabilities/

 Interact

Create a new availability for a calendar

Input Format

```
{ "start_period": "<start-period>", "end_period": "<end-period>", "preference": "<preference>" }
```

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner or invitee
- 404 Not Found - Invalid calendar
- 410 Gone - Deadline passed

Output Format when successful

```
{ "id": <availability-id>, "start_period": "<start-period>", "end_period": "<end-period>",  
"preference": "<preference>", "calendar": <calendar-id>, "user": <user-id> }
```

Output Format when unsuccessful

```
{ "detail": "<error-message>" }
```

With the following error messages: - You do not have permission to perform this action. - Deadline has passed
- This field is required - Not owner or invitee

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
start_period required	The start time for the availability
end_period required	The ending time for the availability
preference required	The preference level for this availability

```
# Load the schema document  
$ coreapi get http://127.0.0.1:8000/docs/  
  
# Interact with the API endpoint  
$ coreapi action calendars availabilities create -p calendar_id=... -p start_period=...
```

availabilities > read

GET /calendars/{calendar_id}/availabilities/{id}/

⇌ Interact

Obtain details of a specific availability provided by a user

Responses

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner or invitee

404 Not Found - Invalid calendar or availability

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	


Parameter	Description
id required	A unique integer value identifying this availability.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars availabilities read -p calendar_id=... -p id=...
```

availabilities > update

PUT /calendars/{calendar_id}/availabilities/{id}/

 Interact

Updates the time period or preference level for a specific availability

Input Format

```
{ "start_period": "<new-start-period>", "end_period": "<new-end-period>", "preference": "<new-preference>" }
```

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner or invitee
- 404 Not Found - Invalid calendar or availability
- 410 Gone - Deadline passed

Output Format when successful

```
{ "id": <availability-id>, "start_period": "<new-start-period>", "end_period": "<new-end-period>", "preference": "<new-preference>", "calendar": <calendar-id>, "user": <user-id> }
```

Output Format when unsuccessful

```
{ "detail": "<error-message>" }
```

With the following error messages: - You do not have permission to perform this action. - Deadline has passed - This field is required - Not owner or invitee

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	
id required	A unique integer value identifying this availability.

Request Body

The request body should be a "application/json" encoded object, containing the following items.


Parameter	Description
start_period required	The start time for the availability
end_period required	The ending time for the availability
preference required	The preference level for this availability

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars availabilities update -p calendar_id=... -p id=... -p start_
```

availabilities > delete

DELETE /calendars/{calendar_id}/availabilities/{id}/

 Interact

Deletes the availability period

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner or invitee
- 404 Not Found - Invalid calendar or availability
- 410 Gone - Deadline passed

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	
id required	A unique integer value identifying this availability.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars availabilities delete -p calendar_id=... -p id=...
```

invitee > list

GET /calendars/{calendar_id}/invitee/

 Interact

Obtain the list of all invitees of a calendar.

Responses

- 200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars invitee list -p calendar_id=...
```

invitee > create

POST /calendars/{calendar_id}/invitee/

 Interact

Invite a new user to be in a calendar

Input Format

```
{ "invitee": <invitee-userid>, "deadline": <deadline> }
```

Responses

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

410 Gone - Finalized

Output Format when successful

```
{ "id": <invitee-id>, "calendar": <calendar-id>, "invitee": <invitee-userid>, "deadline":
<deadline>, "has_availability": <true/false> }
```

Output Format when unsuccessful (404, when user is not invited)

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

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
invitee required	The user id of the invited user to the calendar
deadline required	The last time that the invitee can add an availability

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars invitee create -p calendar_id=... -p invitee=... -p deadline=...
```

invitee > read

GET /calendars/{calendar_id}/invitee/{id}

⇌ Interact

Obtain details of a specific invitee

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner
- 404 Not Found - Invalid calendar or invitee

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	
id required	A unique integer value identifying this invitee.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars invitee read -p calendar_id=... -p id=...
```

invitee > update

PUT /calendars/{calendar_id}/invitee/{id}

⇌ Interact

Updates the deadline for the invitee to add an availability (no other fields can be modified)

Input Format

```
{ "deadline": <new-deadline> }
```

Responses

- 200 OK - Success

401 Unauthorized - Not authenticated
403 Forbidden - Not owner
404 Not Found - Invalid calendar or invite
410 Gone - Finalized

Output Format when successful

```
{ "id": <invitee-id>, "calendar": <calendar-id>, "invitee": <invitee-userid>, "deadline":  
<deadline>, "has_availability": <true/false> }
```

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	
id required	A unique integer value identifying this invitee.

Request Body

The request body should be a "application/json" encoded object, containing the following items.

Parameter	Description
invitee required	The user id of the invited user to the calendar
deadline required	The last time that the invitee can add an availability

```
# Load the schema document  
$ coreapi get http://127.0.0.1:8000/docs/  
  
# Interact with the API endpoint  
$ coreapi action calendars invitee update -p calendar_id=... -p id=... -p invitee=... -
```

invitee > delete

DELETE /calendars/{calendar_id}/invitee/{id}

⇌ Interact

Deletes the invite, and removes all associated content with the invitee

The calendar is identified by the API endpoint's parameter `pk`

It does not return any message upon successful in the JSON response

Responses

200 OK - Success
401 Unauthorized - Not authenticated
403 Forbidden - Not owner
404 Not Found - Invalid calendar or invite
410 Gone - Finalized

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	
id required	A unique integer value identifying this invitee.

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars invitee delete -p calendar_id=... -p id=...
```

schedule > list

GET /calendars/{calendar_id}/schedule/

Interact

Obtain the list of meets that make up a schedule for the calendar

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner
- 404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars schedule list -p calendar_id=...
```

schedule > create

POST /calendars/{calendar_id}/schedule/

Interact

DELETES OLD SCHEDULE IF EXISTS, then generates a new suggested schedule

Calendar is identified by the API endpoint's parameter calendar_id

Responses

- 200 OK - Success
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not owner

404 Not Found - Invalid calendar

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars schedule create -p calendar_id=...
```

schedule > update

PUT /calendars/{calendar_id}/schedule/

[⇌ Interact](#)

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars schedule update -p calendar_id=...
```

schedule > delete

DELETE /calendars/{calendar_id}/schedule/

[⇌ Interact](#)

Deletes the meeting period for a schedule

Returns a 200 OK response if the deletion is successful, with empty JSON response

Responses

200 OK - Success

401 Unauthorized - Not authenticated

403 Forbidden - Not owner

404 Not Found - Invalid calendar

410 Gone - Finalized

Path Parameters

The following parameters should be included in the URL path.

Parameter	Description
calendar_id required	

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action calendars schedule delete -p calendar_id=...
```

contacts

list

GET /contacts/

[↔ Interact](#)

Obtain the list of username of friends of the user.

Response

200 OK - Successful request

Output Format when successful

```
{ "friends": [<username1>, <username2>, ...] }
```

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action contacts list
```

friendRequests > list

GET /contacts/friendRequests/

[↔ Interact](#)

Obtain the list of friend requests that the user has received but not accepted yet.

Response

200 OK - Successful request

Output Format when successful

```
{ "requester_username": <requester-username>, }
```

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action contacts friendRequests list
```

friendRequests > request > create

POST /contacts/friendRequests/request/

 Interact

Accept or reject a friend request.

Input Format

```
{ "username": <username>, "action": <True/False> }
```

Response

200 OK - Successful request

400 Bad Request - Any error is encountered, shown below

Output Format when successful

```
{ "message": "User with username: <username> has been <accepted/rejected>" }
```

Output Format when unsuccessful

```
{ "error": <error-message> }
```

With the following error messages:

- User does not exist
- No friend request from this user

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action contacts friendRequests request create
```

friendRequests > user > create

POST /contacts/friendRequests/user/

 Interact

Send a friend request to the user with the username specified.

Input Format

```
{ "requested": "<requested-username>" }
```

Response

201 Created - Successfully created

400 Bad Request - Any error is encountered, shown below

Output Format when successful

```
{ "message": "Friend request sent to <requested-username>" }
```

Output Format when unsuccessful

```
{ "error": <error-message> }
```

With the following error messages:

- You cannot add yourself to your contacts
- User does not exist
- You have already sent a request to this user

- This user has already sent you a request

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action contacts friendRequests user create
```

friendRequests > user > delete

DELETE /contacts/friendRequests/user/

 Interact

Delete a friend from user's contact list.

Input Format

```
{ "username": <username> }
```

Response

204 No Content - Successfully deleted

400 Bad Request - Any error is encountered, shown below

Output Format when successful

```
{ "message": "Friend <username> deleted" }
```

Output Format when unsuccessful

```
{ "error": <error-message> }
```

With the following error messages:

- User does not exist
- You are not friends with this user

```
# Load the schema document
$ coreapi get http://127.0.0.1:8000/docs/

# Interact with the API endpoint
$ coreapi action contacts friendRequests user delete
```