

Model

Model 1: User

Default User

Model 2: Contact

- User 1
- User 2

Model 3: Node

- meeting id
- Timestamp
- Sender user

Model 3.1: Remind node (inherits Node)

- Receiver user
- Message

Model 3.2: Join node (inherits Node)

- Receiver user

Model 3.3: Submit node (inherits Node)

- Calendar_id

Model 3.4: Poll node (inherits Node)

- Calendar_id (intersected)
- Four selected datetime

Model 3.5: State node (inherits Node)

- Date time
- State (ready or final)

Model 3.6: Remove node (inherits node)

- Initiator
- Kicked out user

Model 4: meeting

- meeting id
- Calendar (mutual between users)
- meeting -> List of users
- meeting -> List of nodes
- State (enum)

Model 5: Calendar

- Owner
- meeting
- Start date
- End date
- List of events
- calendar_id

Model 6: Event

- Name
- Description
- Availability (enum)
- Start DateTime
- End DateTime
- Repeat & time (extra)

Model 7: Member

- User
- Meeting
- Role

API

Root: <http://localhost:8000/api/>

- GET: Return API document
 - Header:X
 - Param:X
 - Body:X
 - Response:X

Accounts APP

Users: <http://localhost:8000/api/accounts/>

- GET: Return a list of all users
 - Header: Authorization(admin)
 - Param:X
 - Body:X
 - Response: id, username, email
- POST: X
- PUT: X
- DELETE: X

Register: <http://localhost:8000/api/accounts/register/>

- GET: X
- POST: Create new users
 - Header: X
 - Param:X
 - Body: username, password, email
 - Response: message
- PUT: X
- DELETE: X

Login: <http://localhost:8000/api/token/>

- GET: X
- POST: User login
 - Header: X
 - Param:X
 - Body: username, password
 - Response: access key
- PUT: X
- DELETE: X

Profile: <http://localhost:8000/api/accounts/profile/>

- GET: Return user information
 - Header: Authorization(user)
 - Param:X
 - Body:X
 - Response: id, username, email
- POST: X
- PUT: Update user information

- Header: Authorization(user)
 - Param: X
 - Body: username, password, email
 - Response: message
- DELETE: X

Contacts: <http://localhost:8000/api/accounts/contacts/>

- GET: List of all contacts of user <ID>
 - Header: Authorization(user)
 - Param:X
 - Body: X
 - Response: user1, alias1, user2, alias2
- POST: Create a new contact for user <ID>
 - Header: Authorization
 - Param:X
 - Body: user2, alias2
 - Response: user1, alias1, user2, alias2
- PUT: X
- DELETE:X

Contact ID of User ID: http://localhost:8000/api/accounts/contacts/<contact_id>/

- GET: Return contact <contactID> of user <userID>
 - Header:Authorization(user)
 - Param:X
 - Body:X
 - Response: id, user_id, alias
- POST: X
- PUT: Return updated information of contact <contactID> of user <userID>
 - Header:Authorization(user)
 - Param:X
 - Body:alias2
 - Response:message
- DELETE: Delete contact <contactID> and Return an empty object
 - Header:Authorization(user)
 - Param:X
 - Body:X
 - Response:message

Meetings APP

meetings: <http://localhost:8000/api/meetings/>

- GET: Return a list of all meetings

- Header: Authorization (Admin)
- Param: X
- Body: X
- Response: List of meeting models
- POST: Create a new meeting
 - Header: Authorization (User)
 - Param: X
 - Body: name, description
 - Response: The new meeting model
- PUT: X
- DELETE: X

meeting ID: http://localhost:8000/api/meetings/<meeting_id>/

- GET: Return the meeting <meetingID>
 - Header: Authorization (IsMember | IsAdminUser)
 - Param: X
 - Body: X
 - Response: Meeting model of specific meeting id
- POST: X
- PUT: Return an updated meeting
 - Header: Authorization (IsMember | IsAdminUser)
 - Param:X
 - Body: name, description
 - Response: meeting model got updated
- DELETE: Return an empty object and delete
 - Header: Authorization (IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response:message

Members in meeting ID: <http://localhost:8000/api/meetings/<meetingID>/members/>

- GET: Return a list of members in meeting <meetingID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param: X
 - Body: X
 - Response: List of member models
- POST: X
- PUT: X
- DELETE: X

Member ID in meeting ID:

http://localhost:8000/api/meetings/<meetingID>/members/<user_id>/

- GET: Return member
 - Header: Authorization(IsMember | IsAdminUser)

- Param:X
- Body:X
- Response: Show a specific member model
- POST: Invite a new member in meetings who is a contactor
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body: user2_id
 - Response: newly invited a member model
- PUT: Update member information of <memID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param: X
 - Body: user2_id
 - Response: updated member model
- DELETE: Return an empty object and delete member <memID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param: X
 - Body: X
 - Response: message

Calendars in meeting ID:

http://localhost:8000/api/meetings/<meetingID>/members/<user_id>/

- Authentication: Yes (Users in the meeting)
- GET: Return a list of calendars in meeting <meetingID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:
 - Body:
 - Response: a list of calendar model
- POST: X
- PUT: X
- DELETE: X

Calendar ID in meeting ID:

http://localhost:8000/api/meetings/<meetingID>/members/<user_id>/Calendar/

- GET: Return calendar <calendarID> in meeting <meetingID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: get a member's calendar model
- POST: Return a newly created calendar

- Header: Authorization(IsMember | IsAdminUser)
- Param:X
- Body:X
- Response: new calendar model
- PUT: Return an updated calendar <calendarID>
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: Updated calendar model
- DELETE: X

Events in Calendar:

http://localhost:8000/api/meetings/<meeting_id>/members/<member_id>/calendar/events/

- Authentication: Yes (Users in the meeting)
- GET: Return a list of events of meeting <meetingID> and member <memberID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response:List of event model
- POST: Create a new event
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body: name, availability, start_time, end_time
 - Response: Newly created event model
- PUT: X
- DELETE: X

Event ID in Calendar:

http://localhost:8000/api/meetings/<meeting_id>/members/<member_id>/calendar/events/event_id

- GET: Return event of <eventID> in meeting <meetingID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: event model of specific event ID
- POST: X
- PUT: Return an updated event <eventID>
 - Header: Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body: name, availability, start_time, end_time
 - Response: updated event model

- DELETE: Return an empty object and delete the event ID
 - Header: Authentication(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: message

Nodes in meeting ID: http://localhost:8000/meetings/<meeting_id>/nodes/

- GET: Return a list of timeline nodes of meeting <meeting_id>
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: List of all types of Node model
- POST: X
- PUT: X
- DELETE: X

Specific node type in meeting ID:

http://localhost:8000/meetings/<meeting_id>/nodes/<node_type>/

- GET: Return specific node <node_type> list of meeting <meetingID>
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: List of <node_type> model
- POST: Create specific node <node_type> of meeting <meetingID>
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: created <node_type> model
- PUT: X
- DELETE: X

Specific node ID in meeting ID:

http://localhost:8000/meetings/<meeting_id>/nodes/<node_type>/<node_id>/

- GET: Return specific node <node_id> of meeting <meetingID>
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: Node of node_id model
- POST: X
- PUT: Update specific node <node_type> of meeting <meetingID> (Poll node only)
 - Header:Authorization(IsMember | IsAdminUser)
 - Param:X
 - Body:X
 - Response: Updated node model
- DELETE: X

Appendix

1. API Rules

GET /collection: 返回资源对象的列表 (数组)

GET /collection/resource: 返回单个资源对象

POST /collection: 返回新生成的资源对象

PUT /collection/resource: 返回完整的资源对象

PATCH /collection/resource: 返回完整的资源对象

DELETE /collection/resource: 返回一个空文档

2. Request Inputs

Header:

- metadata about the request
- HTTP Headers are NOT part of the URL
- if it's information about the request or the client, then the header is appropriate
- headers are hidden from end-users
- globally data
- restrict Attacks by detecting authorization on its header because a header can be accessed before the body is downloaded

Param:

- the query params are within the URL
- like this "tag=networking&order=newest"
- if it's the content of the request itself, then it's a parameter
- The product ID and requested image size are examples of "some detail" (or parameter) being supplied as part of the content of a request
- parameters can be seen by end-users (query parameters) on URL

Body:

- data of business logic
- important information
- unlike the body, proxy servers are allowed to modify headers
- data in specific kinds of requests
- you can pass tokens by the body as encoding & decoding in servers

3. Authentication & Permission

Token generated by user login.

AllowAny

IsAdminUser

IsAuthenticated

IsMember

- Permission:
 - This user's profile
 - This user's contact list
 - This user's contact
 - This user's meeting list
 - This user's meeting
 - This user's meeting member list
 - This user's meeting calendar list
 - This user's meeting calendar's event list
 - This user's meeting node list

4. User

Super User

username: csc309

Password: csc309yes!!

5. 2 pointer version

```
Def find intersection (calendar1, calendar2):
    assert(len calendar1, len calendar 2 > 0)
    l, j = 0, 0
    Intersection = []
    While (i < len(c1) and j < len(c2)):
        Start1, end1 = c1[i].start, c1[i].end
        Start2, end2 = c2[j].start, c2[j].end
        // determine whether intersect
```

```
If start1 <= end2 and start2 <= end1:  
    intersection.append(max(start1, start2), min(end1, end2))
```

```
If end1 < end2:  
    I += 1;
```

```
Else:  
    J += 1;
```

Return intersection

```
Def find(calendars):  
    Curr_intersection = calendars[0]  
    For calendar in calendars[1:]:  
        Curr_intersection = find intersection (currintersection, calendar)  
        If curr_intersection.length == 0:  
            Return []  
    Return current_intersection
```

6. set of test data

- Register:

POST ▼ | <http://localhost:8000/api/accounts/register/>

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	Key		Value
<input checked="" type="checkbox"/>	username	Text ▼	jjiaoxingrun
<input checked="" type="checkbox"/>	password	Text ▼	Jxr20010103
<input checked="" type="checkbox"/>	email	Text ▼	xingrun.jiao@mail.utoronto.ca

● Login Token:

POST

http://localhost:8000/api/token/

Send

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Textjiaoxingrun			
<input checked="" type="checkbox"/>	password	TextJxr20010103			
<input type="checkbox"/>	email	Textxingrun.jiao@mail.utoronto.ca			

odyCookiesHeaders (10)Test ResultsStatus: 200 OKTime: 962 msSize: 794 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "refresh": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2t1b190eXB1IjoicmVmcmVzaCI6ImV4cCI6MTc3MDI3MzE1NCwiaWF0IjoxNzEwMTg2NzU0LCJqdGkiOiI1ZDNiZTZhOTZkODc0MDI0YmI1OWE3OTAyNDcxYWU0YyIsInVzZXJfaWQiOiJ9.UGy4FLZasZNPLQ8uzrBZojLq6rmdp8WZf4W9xY1UvUw",
3   "access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2t1b190eXB1IjoicmVmcmVzaCI6ImV4cCI6MTc3MDI3MzE1NCwiaWF0IjoxNzEwMTg2NzU0LCJqdGkiOiI1ZDNiZTZhOTZkODc0MDI0YmI1OWE3OTAyNDcxYWU0YyIsInVzZXJfaWQiOiJ9.UGy4FLZasZNPLQ8uzrBZojLq6rmdp8WZf4W9xY1UvUw"
4 }
```

● Profile info:

GET

http://localhost:8000/api/accounts/profile/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

Headers

9 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2t1b190eXB1IjoicmVmcmVzaCI6ImV4cCI6MTc3MDI3MzE1NCwiaWF0IjoxNzEwMTg2NzU0LCJqdGkiOiI1ZDNiZTZhOTZkODc0MDI0YmI1OWE3OTAyNDcxYWU0YyIsInVzZXJfaWQiOiJ9.UGy4FLZasZNPLQ8uzrBZojLq6rmdp8WZf4W9xY1UvUw				
	Key	Value	Description			

odyCookiesHeaders (10)Test ResultsStatus: 200 OKTime: 12 msSize: 387 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 2,
3   "username": "jiaoxingrun",
4   "email": "xingrun.jiao@mail.utoronto.ca"
5 }
```

● Profile update:

PUT

http://localhost:8000/api/accounts/profile/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	username	Textjiaoxingrun			
<input checked="" type="checkbox"/>	password	TextJxr20010103			
<input type="checkbox"/>	email	Textxingrun.jiao@mail.utoronto.ca			
	Key	Value	Description		

odyCookiesHeaders (10)Test ResultsStatus: 200 OKTime: 856 msSize: 349 BSave as example

PrettyRawPreviewVisualizeJSON

```
1 [
2   "message: Update Profile success"
3 ]
```

● Create contact:

POST

http://localhost:8000/api/accounts/contacts/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettings

noneform-data x-www-form-urlencodedrawbinaryGraphQL

<input checked="" type="checkbox"/>	user2	Text	1	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 27 msSize: 367 BSave as example

PrettyRawPreviewVisualize

```
{"id":1,"user1":2,"user2":1,"alias1":"","alias2":""}
```

● View contact list:

GET

http://localhost:8000/api/accounts/contacts/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettings

noneform-data x-www-form-urlencodedrawbinaryGraphQL

<input checked="" type="checkbox"/>	user2	Text	1	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 11 msSize: 369 BSave as example

PrettyRawPreviewVisualize

```
[{"id":1,"user1":2,"user2":1,"alias1":"","alias2":""}]
```

● View specific contact info:

GET

http://localhost:8000/api/accounts/contacts/1/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettings

noneform-data x-www-form-urlencodedrawbinaryGraphQL

<input checked="" type="checkbox"/>	user2	Text	1	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 11 msSize: 353 BSave as example

PrettyRawPreviewVisualize

```
{"id":1,"user_id":1,"alias":""}
```

● Change contact alias:

PUT

http://localhost:8000/api/accounts/contacts/1/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

<input checked="" type="checkbox"/>	alias2	Text	hhh	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 13 msSize: 357 BSave as example

PrettyRawPreviewVisualize

{"message":"Contact info updated!"}

● Delete contact:

DELETE

http://localhost:8000/api/accounts/contacts/1/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

<input checked="" type="checkbox"/>	alias2	Text	hhh	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 33 msSize: 365 BSave as example

PrettyRawPreviewVisualize

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

● Create meeting:

POST

http://localhost:8000/api/meetings/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

<input checked="" type="checkbox"/>	name	Text	sample	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 201 CreatedTime: 102 msSize: 476 BSave as example

PrettyRawPreviewVisualize

{"id":6,"name":"sample","description":null,"state":"edit","created_time":"2024-03-11T20:08:25.386205Z","modified_time":"2024-03-11T20:08:25.386241Z"}

● View meeting list:

GET

http://localhost:8000/api/meetings/

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

Headers

7 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...				
	Key	Value	Description			

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 9 msSize: 523 BSave as example

PrettyRawPreviewVisualize

{

"count":1,"next":null,"previous":null,"results":[{"id":6,"name":"sample","description":null,"state":"edit",

"created_time":"2024-03-11T20:08:25.386205Z","modified_time":"2024-03-11T20:08:25.386241Z"}]}

● View specific meeting:

GET

http://localhost:8000/api/meetings/6/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCookies

Headers

9 hidden

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...				
	Key	Value	Description			

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 22 msSize: 485 BSave as example

PrettyRawPreviewVisualize

{

"id":6,"name":"sample","description":null,"state":"edit","created_time":"2024-03-11T20:08:25.386205Z","modified_time":"2024-03-11T20:08:25.386241Z"}

}

● Update specific meeting:

PUT

http://localhost:8000/api/meetings/6/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettingsCook

noneform-datax-www-form-urlencodedorawbinaryGraphQL

<input checked="" type="checkbox"/>	name	Text	sample_new	
<input type="checkbox"/>		Text	Jxr20010103	
<input type="checkbox"/>	email	Text	xingrun.jiao@mail.utoronto.ca	
	Key	Text	Value	Description

bodyCookiesHeaders (10)Test Results

Status: 200 OKTime: 17 msSize: 489 BSave as example

PrettyRawPreviewVisualize

{

"id":6,"name":"sample_new","description":null,"state":"edit","created_time":"2024-03-11T20:08:25.386205Z","modified_time":"2024-03-11T20:12:46.881795Z"}

}

●

