Mentoring Tool Documentation

Release 0.1

Praekelt Foundation

Aug 07, 2017

Contents

1 Instructions for local dev **3**
- 1.1 Installation 3
- 1.2 Account creation 3

2 Moving Parts **5**
- 2.1 Comms API 5
- 2.2 Web Interface 5

3 Model Reference **7**
- 3.1 Mentorship Models 7

4 Comms API Reference **11**
- 4.1 Calls 11
- 4.2 Messages 12
- 4.3 Feedback 13
- 4.4 OnionDev custom API endpoints 14

5 Mentor API Reference **17**
- 5.1 Mentees 17
- 5.2 Mentors 18
- 5.3 Calls 20
- 5.4 Messages 21
- 5.5 Events 22
- 5.6 Tasks 22
- 5.7 Schedule 23

6 Indices and tables **25**

HTTP Routing Table **27**

Python Module Index **29**

This Mentoring Tool is designed to allow mentors to contact mentees via voice calls or text messages without exposing the contact details of either participant.

Contents:

CHAPTER 1

Instructions for local dev

Installation

- Install and run Redis
- Install VirtualEnv

1. Clone the Mentoring Tool repo:

```
$ git clone git@github.com:praekelt/mentorship.git
$ cd mentorship
```

2. Setup a VirtualEnv:

```
$ virtualenv ve
$ source ve/bin/activate
```

3. Install requirements:

```
$ cd mentorship
$ pip install -r requirements.txt
$ pip install -r requirements-dev.txt
```

4. Setup database and run:

```
$ ./manage.py migrate
$ ./manage.py createsuperuser
$ ./manage.py runserver
```

Account creation

If you're using a fresh database then a few objects will need to be created through the Django Admin interface in order to get things working smoothly (remember you'll need to log into the Admin with a superuser account for this):

Voice API

Many of the sections of this project require access to an external voice api. For testing purposes create an instance of ExampleApi.

The urls entered are not called so you can enter any string that will pass the url format checks.

Mentee User

1. Create a new user. Make sure to fill in the *first name* field
2. Create a mentee instance linked to the new user.

Mentor User

1. Create a Mentee User as above.
2. Create a new user. Make sure to fill in the *first name* field
3. Create a mentor instance linked to the new user and mentee.

CHAPTER 2

Moving Parts

The Mentoring Tool has a number of moving parts described below:

Comms API

REST API to handle communications to and from messaging and voice call providers. The base API can be inherited and extended to facilitate the inclusion of new providers. API end points provided:

- Receiver for voice call termination notification
- Receiver for incoming messages
- Receiver for incoming call feedback (json string)

Web Interface

A crude interface used during development to simulate the behaviour of, and requests made by, the Android app. Features include:

- Login for mentors
- Text inputs and send buttons for messages
- Call and end call buttons to initiate calls to the Comms API
- Event log showing recent events
- Display of incoming messages (shown in the event log)
- Call scheduling page for scheduling future calls
- To Do list for display and editing of tasks the mentor has yet to complete.

CHAPTER 3

Model Reference

Mentorship Models

Contact

This is the base model used to create a Mentee and Mentor. It has the following fields:

`user` this is a reference to a django user

`contact_number` a valid mobile number

`external_id` used to link to the same users on other platforms

Mentee

One to One relation with the contact model.

Mentor

Inherits from the contact model and adds the following attributes:

`mentee` a foreign key to a mentee model

`profile_pic` a picture associated with the mentor

As well as other information about the specific mentor such as job title, motivation and inspiration.

Call

This models the call that happens between a Mentor and Mentee. It has the following attributes:

`start_time` the time the mentor clicks start_call on the app `partner_start_time` the time the call is started from the partner (Onion Dev)

`end_time` the time that our API received a notification saying that the call has ended

`partner_end_time` the time the partner has ended the call

`caller` the mentor that started the call

`receiver` the mentee that received the call

`feedback_task` a foreign key to the feedback task associated with the call

`recording_url` a url to the recording of the call

`partner_id` the id of the instance of the call on the call partner's system

`activity` a foreign key to the activity associated with the call

`scheduled_call` a foreign key to the scheduled call associated with the call

`call_id` the call ID

`status` this can only be successful, scheduled, inactive or inprogress.

The call model has the following methods:

`from_partner_id(self, partner_id)` takes in the parter id and returns the call instance for that id if any.

`status(self)` returns the status of the call.

`status(self, new_status)` sets the status of the calls

`get_duration(self)` returns how long the call lasted according to the call partner

`get_lag(self)` returns the time difference between when the mentor started the call and when the call partner started the call

`get_status_display(self)` returns the status that should be displayed

`has_timed_out(self)` returns whether not the call has timed out based on the partner end time.

`dial(self)` sets the call_id and creates the call started event.

`ended(self)` updates the call as ended

`handle_ended_successful(self, data)` sets the end time, sends a notification that the call as ended and creates an event for feedback to take place.

CallDataRecord

A record of attempted calls. It has the following attributes:

`call` a foreign key to a call instance

`partner_id` the id sent from the partner of the call

`start_time` the time the call was attempted

`status` the status of the attempted call

ScheduledCall

A call that has been scheduled to take place between a Mentor and Mentee at a specific date and time. It has the following attribtues:

`created_at` the date and time the scheduled call was created at `call_time` the scheduled time for the call to take place

`caller` the mentor that scheduled the call

`callee` the mentee that shoudl receive the call

`activity` the activity that will be discussed in the call

The scheduled call also has the following methods:

`notify_callee(self, message)` sends, creates and returns a Message object sent to the mentee reminding them of the call.

`notify_callee_scheduled(self)` sends a standard message when the scheduled call was created to the mentee.

`notify_callee_rescheduled(self, new_time)` sends a message to the mentee informing them that their scheduled call has been rescheduled

`notify_callee_cancelled(self)` sends a message to the mentor informing them that their scheduled call has been cancelled

`remind_caller(self, reminder_type, text=None)` reminds the Mentor of their scheduled call via a push notification and created a CallReminder object.

CallReminder

An object to record when reminders are sent. It has the following attributes:

`scheduled_call` a foreign key to the scheduled call

`reminder_type` a charfield describing the type of reminder.

BaseCallNote

This is a polymorphic model that serves as a base for call note models. It has the following properties:

`version` the version of the call note instance i.e 1 or 2

`mentor` the mentor that created the call note

`call` the call associated with the call note

`created_at` the date and time the call note was created at.

CallNoteV1

Inherits from BaseCallNote and adds on the required call notes questions such as mentee state as well as following properties:

`version` this is set to one as it is the older version of Call Notes and is only here for older version of the app.

CallNoteV2

Inherits from BaseCallNote and adds on the required call notes questions such as rating, call quality as well as following properties:

`version` this is set to 2 as is is the latest implementation of call notes.

Feedback

This is used to store data given by mentee via IVR about a call with their mentor. It has the following attributes:

`received_at` the date and time the feedback was received

`start_time` the time that the feedback call started

`sender` the mentee that completed the feedback

`call` the call the feedback is about

`questions` there are four questions (each its own field) these hold the answers to each question

`completed` boolean field. True if all four questions have been answered. Else False.

Message

This is used to send and receive text between mentor and mentee. The messages can be sent via the app to the mentee from the mentor. Or they can be sent via the system. Messages from mentee to mentor will be received by the app through this model as well. A message has the following attributes:

`time_sent` the time the message was sent

`sender` the mentee or mentor that sent the message

`recipient` the mentor or mentor that received the message

`message_type` the type of message sent. This is either 'system' or 'user'

`content` the message text

The message model has the following methods:

`send_and_create` creates and sends the message

`send` sends the message

`receive_message` creates a message received event and sends a push notification to the Mentor's device

Event

The event model is used to record when events like a message has been sent or a call has been scheduled. It has the following attributes:

`occured_at` the date and time the event occurred

`event_type` a short description of the event_type

`relevant_mentor`

`object_id` the id of the message or call that triggered the event.

CHAPTER 4

Comms API Reference

The Comms module of the mentoring tool provides API endpoints for communication with messaging and voice call providers.

All communication with these endpoints must be authenticated. Any registered user can authenticate to the API using their username and password. Currently the authentication scheme used is Basic Auth.

If you're developing locally remember to create the necessary accounts *Account creation*

A full and up to date version of the api docs can be found by adding */docs/* onto your root url.

Calls

`GET /comms/call/`
List all calls

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
      {
          "id": 1,
          "start_time": "2016-01-14T08:56:59.152335Z",
          "end_time": "2016-01-14T08:57:01.749749Z",
          "caller": 1,
          "receiver": 2
      }
  ]
}
```

`GET /comms/call/`(**call_id:** *str*)
Return the call details.

Example response:

```
{
  "id":1,
  "start_time":"2016-01-14T08:56:59.152335Z",
  "end_time":"2016-01-14T08:57:01.749749Z",
  "caller":1,
  "receiver":2
}
```

`PUT /comms/call/`(**call_id:** *str*)
Update the details of a call. Currently this will ignore the data sent and automatically update the *end_time* of the call to be the current time.

Parameters

- **`id`** (*int*) – The call id.
- **`start_time`** (*str*) – The date and time the call started in ISO 8601 format.”
- **`end_time`** (*str*) – The date and time the call ended in ISO 8601 format.”
- **`caller`** (*int*) – The id of the call initiator.
- **`receiver`** (*int*) – The id of the receiver of the call.

`PATCH /comms/call/`(**call_id:** *str*)
Update the details of a call. Currently this will ignore the data sent and automatically update the *end_time* of the call to be the current time.

Accepts the same parameters as *`PUT /comms/call/(call_id:str)`*. Only the parameters provided are updated. Others retain their original values.

Messages

`GET /comms/message/`
List all messages

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "time_sent": "2016-01-14T12:49:12.509174Z",
      "sender_contact": "1234567890",
      "recipient_contact": "1234567891",
      "content": "hi there"
    }
  ]
}
```

`POST /comms/message/`
Create a new message

Parameters

- **`sender_contact`** (*str*) – The contact number for the sender of the message.
- **`recipient_contact`** (*str*) – The contact number for the receiver of the message.
- **`content`** (*str*) – The content of the message"

GET /comms/message/(**message_id:** *str*)

Return the message details.

Example response:

```
{
  "id": 1,
  "time_sent": "2016-01-14T12:49:12.509174Z",
  "sender_contact": "sdfsdfsdf",
  "recipient_contact": "sdfsdfsdfsd",
  "content": "hi there"
}
```

Feedback

GET /comms/feedback/

List all feedback objects

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
      {
          "id": 1,
          "sender_id": 2,
          "call_id": 1,
          "complete": true,
          "content": "\"[{u'question 6': u'Purple', u'question 4': u'No', u
↪'question 5': u'4', u'question 2': u'Yes', u'question 3': u'Yes', u'question 1
↪': 1}]\""
      }
  ]
}
```

POST /comms/feedback/

Create a new feedback object

Parameters

- **`sender_id`** (*int*) – The id for the user that created the feedback.
- **`call_id`** (*int*) – The id for the call that the feedback relates to.
- **`complete`** (*bool*) – A status flag indicating whether the feedback survey was fully completed. It should be set to *true* if the survey was fully completed and *false* if the survey was submitted incomplete.
- **`content`** (*str*) – The content of the feedback. This must be a valid JSON string.

GET /comms/feedback/(**feedback_id:** *str*)

Return the feedback object details.

Example response:

```
{
    "id": 1,
    "sender_id": 2,
    "call_id": 1,
    "complete": false,
    "content": "\"[{u'question 6': u'Purple', u'question 4': u'No', u'question 5
↪': u'4', u'question 2': u'Yes', u'question 3': u'Yes', u'question 1': 1}]\""
}
```

OnionDev custom API endpoints

POST /comms/onion_dev_call_end/

Report the end of a call.

Parameters

- **handler** (*str*) – The MSISDN of the caller.
- **conversationEndTime** (*str*) – The time of the end of the call as recorded by OnionDev. This is expected to be in the Asia/Calcutta timezone and have the format "%Y-%m-%d %H:%M:%S.%f".
- **conversationStartTime** (*str*) – The time of the start of the call as recorded by OnionDev. This is expected to be in the Asia/Calcutta timezone and have the format "%Y-%m-%d %H:%M:%S.%f".
- **conversationId** (*str*) – The ID used by OnionDev to identify the call.
- **recordingUrl** (*str*) – URL linking to a recording of the call.
- **cdrs.id** (*str*) – ID of call attempt.
- **cdrs.startTime** (*str*) – Start time of call attempt as recorded by OnionDev.
- **cdrs.errorCause** (*str*) – Error Code for failed attempt.
- **scheduledCallId** (*str*) – Identifier for the call, should match to the same identifier given in the call initiation response as *scheduled_call_id*

Example response:

```
{
    "conversationId": "23",
    "conversationEndTime": "2017-04-20 17:33:42.899903",
    "feedbackSubscriptionStatus": True,
    "cdrs": [{
        "id": "123",
        "startTime": "2017-04-20 17:31:18",
        "errorCause": "NO_ANSWER"
    }, {
        "id": "456",
        "startTime": "2017-04-20 17:32:18",
        "errorCause": "NORMAL_CLEARING"
    }],
    "totalTries": 2,
    "conversationSuccess": True,
    "beneficiary": "911234567890",
    "conversationStartTime": "2017-04-20 17:32:33.593300",
    "handler": "919876543321",
    "recordingUrl": "https://example.org/123/456.mp3",
    "scheduledCallId": "123"
}
```

POST /comms/onion_dev_feedback/

Create a call feedback report.

Parameters

- **cdr** (*dict*) – An object containing the details of the call. See fields below for required attributes.
- **cdr.callerid** (*str*) – The MSISDN of the caller.
- **complete** (*boolean*) – True if this is a complete call feedback report. False if it is a partial feedback report.
- **records** (*str*) –

The feedback content given as an object, with an item for each answer to a question asked on the IVR feedback call. Each key is an identifier for the question asked, and the values are objects with the following properties:

```
answer: The user's answer to a question as a string
questionTime: The IST-based time at which the answer was recorded
order: number indicating the ordering of this question
```

Example response:

```
{
    "conversationId": "123",
    "surveyId": "484",
    "completed": True,
    "cdr": {
        "start_time": "2017-04-21 07:00:06.432989",
        "id": "23",
        "callerid": "919876543321"
    },
    "records": {
        "127123": {
            "answer": "Happy",
            "questionTime": "2017-04-21 07:00:54",
            "order": "1"
        }
    }
}
```

POST /comms/onion_dev_message/

Register a text message (SMS) that has been received.

Parameters

- **mobile** (*str*) – The MSISDN of the sender (mentee).
- **text** (*str*) – The content of the message.

Example response:

```
{"message": "Hi Mentor", "mobile": "911234567890"}
```

CHAPTER 5

Mentor API Reference

The Mentor module of the mentoring tool provides API endpoints that allow front end applications to access and manipulate the data.

All communication with these endpoints must be authenticated. Mentor and Mentee endpoints can only be accessed by admin users. All other endpoints require usernames and passwords for a Mentor account for access.

Currently the authentication scheme used is Basic Auth.

If you're developing locally remember to create the necessary accounts *Account creation*. Otherwise, Please contact an admin with superuser permissions to get the relevant accounts created.

A full and up to date version of the api docs can be found by adding */docs/* onto your root url.

Mentees

GET /mentor/mentee
List all mentees.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": "2",
      "username": "tony",
      "first_name": "tony",
      "last_name": "",
      "email": "",
      "password": "pbkdf2_sha256$20000$FEIdxqmdW7av$GEB/
↪BUnbRUFj8f7CdNYKaWxGobM2uCJ9sD2/bNSUbn0=",
      "contact_number": "+27123456789"
    }
  ]
}
```

POST /mentor/mentee

Create a new user with a mentee account. Returns a 201 on success.

Parameters

- **id** (*int*) – The id for the new user.
- **username** (*str*) – The username for the new user. This will be used to authenticate them.
- **first_name** (*str*) – The first name of the new user.
- **last_name** (*str*) – The surname for the new user.
- **email** (*str*) – The email address for the new user.
- **password** (*str*) – The password for the new user. This will be used to authenticate them.
- **contact_number** (*str*) – The contact number for the new user. This will be used to initiate calls and send messages

GET /mentor/mentee/(mentee_id: *str*)

Return the details for the mentee.

Example response:

```
{
  "id": "2",
  "username": "tony",
  "first_name": "tony",
  "last_name": "",
  "email": "",
  "password": "pbkdf2_sha256$20000$FEIdxqmdW7av$GEB/
↪BUnbRUFj8f7CdNYKaWxGobM2uCJ9sD2/bNSUbn0=",
  "contact_number": "+27123456789"
}
```

PUT /mentor/mentee/(mentee_id: *str*)

Edit the details for a mentee account.

Accepts the same parameters as *POST /mentor/mentee*. Returns the same response as *GET /mentor/mentee/(mentee_id:str)*.

Mentors

GET /mentor/mentor

List all mentors.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": "1",
      "username": "adam",
      "first_name": "adam",
      "last_name": "",
      "email": "",
      "password": "pbkdf2_sha256$20000$8aa2Qh1GJFhX
↪$RV3WqQoN17KoH7SS8r0nUbapHCGqa8hG3I/37xeGuqw=",
      "contact_number": "+27987654321",
      "mentee": 2
    }
  ]
}
```

POST /mentor/mentor

Create a new user with a mentor account. Returns a 201 on success.

Parameters

- **id** (*int*) – The id for the new user.
- **username** (*str*) – The username for the new user. This will be used to authenticate them.
- **first_name** (*str*) – The first name of the new user.
- **last_name** (*str*) – The surname for the new user.
- **email** (*str*) – The email address for the new user.
- **password** (*str*) – The password for the new user. This will be used to authenticate them.
- **contact_number** (*str*) – The contact number for the new user. This will be used to initiate calls and send messages
- **mentee** (*int*) – The id for the mentee that has been assigned to this mentor.

GET /mentor/mentor/(mentor_id: *str*)

Return the details for the mentor.

Example response:

```
{
  "id": "1",
  "username": "adam",
  "first_name": "adam",
  "last_name": "",
  "email": "",
  "password": "pbkdf2_sha256$20000$8aa2Qh1GJFhX
↪$RV3WqQoN17KoH7SS8r0nUbapHCGqa8hG3I/37xeGuqw=",
  "contact_number": "+27987654321",
  "mentee": 2
}
```

PUT /mentor/mentor/(mentor_id: *str*)

Edit the details for a mentor account.

Accepts the same parameters as *POST /mentor/mentor*. Returns the same response as *GET /mentor/mentor/(mentor_id:str)*.

Calls

GET /mentor/call/

List all calls made by the authenticated user.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
      {
          "id": 1,
          "start_time": "2016-01-14T08:56:59.152335Z",
          "end_time": "2016-01-14T08:57:01.749749Z",
          "caller": 1,
          "receiver": 2
      }
  ]
}
```

POST /mentor/call/

Create a call, with a start_time of the present time, from the currently authenticated mentor to their mentee in the database, then inform the API in use to initiate a call between them. Currently this does not accept any parameters and will ignore any data sent.

Example response:

```
{
  "id":1,
  "start_time":"2016-01-14T08:56:59.152335Z",
  "end_time":null,
  "caller":1,
  "receiver":2
}
```

GET /mentor/call/ (**call_id:** *str*)

Return the call details if the call was made by the authenticated user. Otherwise, return 404.

Example response:

```
{
  "id":1,
  "start_time":"2016-01-14T08:56:59.152335Z",
  "end_time":"2016-01-14T08:57:01.749749Z",
  "caller":1,
  "receiver":2
}
```

PUT /mentor/call/ (**call_id:** *str*)

This access is provided for development and testing purposes only. It should not be used in production. Update the details of a call that was made by the authenticated user. Currently this does not accept any parameters and will ignore any data sent. If the *end_time* of the call is empty then the request will trigger an automatic update of the *end_time* of the call to be the current time.

Returns the same response as `GET /mentor/call/(call_id:str)`.

PATCH /mentor/call/(call_id: *str*)

This access is provided for development and testing purposes only. It should not be used in production. Update the details of a call that was made by the authenticated user. Currently this does not accept any parameters and will ignore any data sent. If the *end_time* of the call is empty then the request will trigger an automatic update of the *end_time* of the call to be the current time.

Returns the same response as *GET /mentor/call/(call_id:str)*.

Messages

GET /mentor/message/

List all messages sent or received by the currently authenticated user.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "time_sent": "2016-01-14T12:49:12.509174Z",
      "sender_contact": "1234567890",
      "recipient_contact": "1234567891",
      "content": "hi there"
    }
  ]
}
```

POST /mentor/message/

Create a new message from the currently authenticated mentor to their mentee in the database, then inform the API in use to send this message.

Parameters

- **sender_id** (*int*) – The if of user who is sending the message.
- **recipient_id** (*int*) – The if of user who should receive the message.
- **content** (*str*) – The content of the message

GET /mentor/message/(message_id: *str*)

Return the message details if the message was sent or received by the authenticated user. Otherwise, return 404.

Example response:

```
{
  "id": 1,
  "time_sent": "2016-01-14T12:49:12.509174Z",
  "sender": "sdfsdfsdf",
  "recipient": "sdfsdfsdfsd",
  "content": "hi there"
}
```

Events

GET /mentor/event/

List all events relevant to the authenticated user.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
      "occured_at": "2016-02-08T16:00:34.635564Z",
      "event_type": "start_call",
      "description": "Call started",
      "relevant_mentor": 1
    }
  ]
}
```

GET /mentor/event/ (**event_id:** *str*)

Return the event details if the event is relevant to the authenticated user. Otherwise, return 404.

Example response:

```
{
    "occured_at": "2016-02-08T16:00:34.635564Z",
    "event_type": "start_call",
    "description": "Call started",
    "relevant_mentor": 1
}
```

Tasks

GET /mentor/task/

List all tasks for the authenticated user.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
    {
        "id": 1,
        "mentor": 1,
        "description": "Complete feedback for call at 16:13 on 2016-02-08",
        "link": "",
        "due_date": null,
        "status": "done"
    }
  ]
}
```

GET /mentor/task/(task_id: *str*)
Return the details for the task if it is for the authenticated user. Otherwise, return 404.

Example response:

```
{
  "id": 5,
  "mentor": 1,
  "description": "Complete feedback for call at 16:26 on 2016-02-08",
  "link": "",
  "due_date": null,
  "status": "pending"
}
```

PUT /mentor/task/(task_id: *str*)
Update the status for a task relevant to the authenticated user.

Parameters

- **status** (*str*) – The new status of the task. Valid values: * "pending" * "in-progress" * "done"

PATCH /mentor/task/(task_id: *str*)
Update the status for a task relevant to the authenticated user.

Accepts the same parameters as *PUT /mentor/task/(task_id:str)*.

Schedule

GET /mentor/schedule/
List all scheduled calls relevant to the authenticated user.

Example response:

```
{
  "count": 1,
  "next": null,
  "previous": null,
  "results": [
      {
          "id": 1,
          "created_at": "2016-02-11T15:25:46.291152Z",
          "call_time": "2016-12-23T17:30:00Z",
          "caller": 1,
          "callee": 2
      }
  ]
}
```

POST /mentor/schedule/
Create a new schedule item for the authenticated mentor to call their mentee. This also immediately sends a message to the mentee to notify them of the call.

Parameters

- **call_time** (*str*) – The time at which the call should take place. Accepts ISO 8601 format.

GET /mentor/schedule/(**schedule_id:** *str*)

Returns the details for the scheduled call if it is relevant to the authenticated user. Otherwise, returns 404.

Example response:

```
{
  "id": 1,
  "created_at": "2016-02-11T15:25:46.291152Z",
  "call_time": "2016-12-23T17:30:00Z",
  "caller": 1,
  "callee": 2
}
```

PUT /mentor/schedule/(**schedule_id:** *str*)

Change the time of a scheduled call between the authenticated mentor and their mentee. This also immediately sends a message to the mentee to notify them of the change.

Parameters

- **call_time** (*str*) – The time at which the call should take place. Accepts ISO 8601 format.

PATCH /mentor/schedule/(**schedule_id:** *str*)

Change the time of a scheduled call between the authenticated mentor and their mentee. This also immediately sends a message to the mentee to notify them of the change.

Parameters

- **call_time** (*str*) – The time at which the call should take place. Accepts ISO 8601 format.

DELETE /mentor/schedule/(**schedule_id:** *str*)

Delete a scheduled call between the authenticated mentor and their mentee. This also immediately sends a message to the mentee to notify them of the change.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

HTTP Routing Table

/comms

GET /comms/call/, 11
GET /comms/call/(call_id:str), 11
GET /comms/feedback/, 13
GET /comms/feedback/(feedback_id:str), 13
GET /comms/message/, 12
GET /comms/message/(message_id:str), 13
POST /comms/feedback/, 13
POST /comms/message/, 12
POST /comms/onion_dev_call_end/, 14
POST /comms/onion_dev_feedback/, 15
POST /comms/onion_dev_message/, 15
PUT /comms/call/(call_id:str), 12
PATCH /comms/call/(call_id:str), 12

/mentor

GET /mentor/call/, 20
GET /mentor/call/(call_id:str), 20
GET /mentor/event/, 22
GET /mentor/event/(event_id:str), 22
GET /mentor/mentee, 17
GET /mentor/mentee/(mentee_id:str), 18
GET /mentor/mentor, 18
GET /mentor/mentor/(mentor_id:str), 19
GET /mentor/message/, 21
GET /mentor/message/(message_id:str), 21
GET /mentor/schedule/, 23
GET /mentor/schedule/(schedule_id:str), 23
GET /mentor/task/, 22
GET /mentor/task/(task_id:str), 22
POST /mentor/call/, 20
POST /mentor/mentee, 18
POST /mentor/mentor, 19
POST /mentor/message/, 21
POST /mentor/schedule/, 23
PUT /mentor/call/(call_id:str), 20
PUT /mentor/mentee/(mentee_id:str), 18
PUT /mentor/mentor/(mentor_id:str), 19
PUT /mentor/schedule/(schedule_id:str), 24
PUT /mentor/task/(task_id:str), 23
DELETE /mentor/schedule/(schedule_id:str), 24
PATCH /mentor/call/(call_id:str), 20
PATCH /mentor/schedule/(schedule_id:str), 24
PATCH /mentor/task/(task_id:str), 23

Python Module Index

c

comms.models, 10

Index

C

comms.models (module), 10