**AI Professor Documentation**

**AI-Professor API Documentation**

**Overview**

The AI-Professor API provides a comprehensive platform for educational institutions to manage courses, content, user accounts, and AI-powered features. This RESTful API enables seamless integration with client applications, allowing for the creation and management of educational resources, student enrollments, and interactive AI chat capabilities.

**Base URL**

All API endpoints are relative to the base URL of your AI-Professor installation.

**Authentication**

The API uses OAuth 2.0 Password Bearer authentication. To access protected endpoints, you need to:

1. Obtain an access token by authenticating with the /auth/login endpoint
2. Include the token in the Authorization header of subsequent requests: Authorization: Bearer YOUR\_TOKEN

For long-running applications, you can refresh your access token using the /auth/refresh-token endpoint.

**API Tokens**

For integration with third-party services, long-running applications, or programmatic access, dedicated API tokens can be generated and managed via the /api-tokens/ endpoint.

**Rate Limiting and Quotas**

API calls may be subject to rate limiting and usage quotas, particularly for AI-powered features. Consult with your system administrator for specific limits applied to your installation.

**Table of Contents**

* [Authentication](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#authentication)
* [User Management](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#user-management)
* [Institutions](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#institutions)
* [File Management](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#file-management)
* [Courses](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#courses)
* [Class Content](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#class-content)
* [Enrollments](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#enrollments)
* [AI Content Processing](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#ai-content-processing)
* [Chat](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#chat)
* [API Tokens](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#api-tokens)
* [Notifications](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#notifications)
* [Email](https://claude.ai/chat/bc9691de-7cba-4a33-9209-855c40976e71#email)

**Authentication**

**Register a new student or teacher account**

POST /auth/register/student-teacher

Creates a new user account with student or teacher role.

**Request Body:**

{

"expire\_date": "YYYY-MM-DDTHH:MM:SS"

}

**Response:**

{

"course\_id": "string",

"enrollment\_active": true,

"expire\_date": "YYYY-MM-DDTHH:MM:SS",

"updated\_at": "string"

}

**Class Content**

**Create Content**

POST /class\_content/

Create new content with optional file uploads.

**Response:**

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

**List Content**

GET /class\_content/

List all content with optional filtering and pagination.

**Query Parameters:**

* title: Filter by title
* course\_id: Filter by course ID
* teacher\_id: Filter by teacher ID
* created\_after: Filter by creation date (after)
* created\_before: Filter by creation date (before)
* page: Page number (default: 1)
* size: Items per page (default: 10)

**Response:**

{

"items": [

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

],

"total": 0,

"page": 1,

"size": 10,

"pages": 0

}

**Get Content**

GET /class\_content/{content\_id}

Get content by ID.

**Response:**

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

**Update Content**

PUT /class\_content/{content\_id}

Update existing content. Only the content owner or admin can update content.

**Request Body:**

{

"title": "string" | null,

"description": "string" | null,

"files\_to\_add": ["string"] | null,

"files\_to\_remove": ["string"] | null

}

**Response:**

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

**Delete Content**

DELETE /class\_content/{content\_id}

Delete content. Only the content owner or admin can delete content.

**Response:** HTTP 204 No Content

**Upload Content With Files**

POST /class\_content/upload

Upload content with files to a course and optionally to specific sections.

**Query Parameters:**

* title: Content title (required)
* course\_id: Course ID (required)
* description: Content description
* visibility: Content visibility (default: "ACTIVE")

**Request Body (multipart/form-data):**

* files: Array of files to upload
* section\_ids: Array of section IDs to add content to

**Response:**

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

**Add Files To Content**

POST /class\_content/{content\_id}/files

Add existing files to content.

**Request Body:**

{

"file\_ids": ["string"]

}

**Response:**

["string"] // Array of file IDs that were added

**Remove Files From Content**

DELETE /class\_content/{content\_id}/files

Remove files from content.

**Request Body:**

{

"file\_ids": ["string"]

}

**Response:** HTTP 204 No Content

**Upload File To Existing Content**

POST /class\_content/upload-file-to-content/{content\_id}

Upload new files to existing content.

**Request Body (multipart/form-data):**

* files: Array of files to upload

**Response:**

["string"] // Array of file IDs that were added

**Add Content To Section**

POST /class\_content/section

Add existing content to a section.

**Request Body:**

{

"section\_id": "string",

"content\_id": "string",

"visibility": "ACTIVE" | "INACTIVE"

}

**Response:**

{

"id": "string",

"section\_id": "string",

"content\_id": "string",

"visibility": "ACTIVE" | "INACTIVE",

"added\_at": "string",

"updated\_at": "string"

}

**Add Content To Multiple Sections**

POST /class\_content/section/bulk

Add existing content to multiple sections at once.

**Request Body:**

{

"section\_ids": ["string"],

"content\_id": "string",

"visibility": "ACTIVE" | "INACTIVE"

}

**Response:**

["string"] // Array of section-content association IDs

**Update Section Content**

PUT /class\_content/section/{section\_id}/{content\_id}

Update content visibility in a section.

**Request Body:**

{

"visibility": "ACTIVE" | "INACTIVE"

}

**Response:**

{

"id": "string",

"section\_id": "string",

"content\_id": "string",

"visibility": "ACTIVE" | "INACTIVE",

"added\_at": "string",

"updated\_at": "string"

}

**Remove Content From Section**

DELETE /class\_content/section/{section\_id}/{content\_id}

Remove content from a section.

**Response:** HTTP 204 No Content

**Get Section Content**

GET /class\_content/section/{section\_id}

Get all content in a section with optional visibility filtering.

**Query Parameters:**

* visibility: Filter by visibility ("ACTIVE" | "INACTIVE")

**Response:**

[

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

]

**Get Content Sections**

GET /class\_content/sections/{content\_id}

Get all sections that contain a specific content item.

**Response:**

["string"] // Array of section IDs

**List Section Contents**

GET /class\_content/section-contents

List all section-content associations with optional filtering and pagination.

**Query Parameters:**

* section\_id: Filter by section ID
* content\_id: Filter by content ID
* visibility: Filter by visibility
* course\_id: Filter by course ID
* page: Page number (default: 1)
* size: Items per page (default: 10)

**Response:**

{

"items": [

{

"id": "string",

"section\_id": "string",

"content\_id": "string",

"visibility": "ACTIVE" | "INACTIVE",

"added\_at": "string",

"updated\_at": "string"

}

],

"total": 0,

"page": 1,

"size": 10,

"pages": 0

}

**Enrollments**

**Create Email Enrollment**

POST /enrollments/email

Create email enrollment for a course.

**Request Body:**

{

"course\_id": "string",

"section\_id": "string" | null,

"expire\_days": 30 | null

}

**Response:**

{

"id": "string",

"course\_id": "string",

"section\_id": "string" | null,

"enrollment\_key": "string",

"expire\_date": "string",

"created\_at": "string",

"created\_by": "string"

}

**Create Public Link Enrollment**

POST /enrollments/public-link

Create public link enrollment for a course.

**Request Body:**

{

"course\_id": "string",

"section\_id": "string" | null,

"expire\_days": 30 | null

}

**Response:**

{

"id": "string",

"course\_id": "string",

"section\_id": "string" | null,

"enrollment\_key": "string",

"enrollment\_link": "string",

"expire\_date": "string",

"created\_at": "string",

"created\_by": "string"

}

**Bulk Invite Students**

POST /enrollments/invite

Send invitations to multiple students at once.

**Query Parameters:**

* course\_id: Course ID (required)
* section\_id: Section ID
* emails: List of email addresses to invite (required)
* create\_accounts: Create accounts for non-existent students (default: false)

**Response:**

{

"sent": ["string"], // Emails to which invitations were sent

"failed": ["string"], // Emails to which invitations could not be sent

"invalid": ["string"] // Invalid email addresses

}

**Verify Enrollment Token**

GET /enrollments/token/verify?token={token}

Verify enrollment token and enroll student in course.

**Response:**

{

"success": true,

"course\_id": "string",

"course\_title": "string",

"section\_id": "string" | null,

"message": "Successfully enrolled in course"

}

**Enroll With Key**

POST /enrollments/enroll-with-key

Enroll student in course using enrollment key.

**Request Body:**

{

"enrollment\_key": "string"

}

**Response:**

{

"success": true,

"course\_id": "string",

"course\_title": "string",

"section\_id": "string" | null,

"message": "Successfully enrolled in course"

}

**Check Course Enrollment**

GET /enrollments/course/{course\_id}

Check if a course is available for enrollment and verify key if provided.

**Query Parameters:**

* key: Enrollment key to validate

**Response:**

{

"is\_available": true,

"course\_info": {

"id": "string",

"title": "string",

"description": "string" | null,

"teacher": {

"id": "string",

"name": "string"

}

},

"key\_valid": true | false | null,

"already\_enrolled": true | false

}

**Unenroll From Course**

DELETE /enrollments/course/{course\_id}

Unenroll yourself from a course.

**Query Parameters:**

* section\_id: Section ID

**Response:**

{

"success": true,

"message": "Successfully unenrolled from course"

}

**Verify Enrollment Invitation**

GET /enrollments/invitation/verify/{token}

Verify enrollment invitation token and enroll student in course.

**Response:**

{

"success": true,

"course\_id": "string",

"course\_title": "string",

"section\_id": "string" | null,

"message": "Successfully enrolled in course"

}

**Get Sent Invitations**

GET /enrollments/invitations

Get list of invitations sent by the teacher.

**Query Parameters:**

* course\_id: Filter by course ID
* email: Filter by email address
* page: Page number (default: 1)
* size: Items per page (default: 10, max: 100)

**Response:**

{

"items": [

{

"id": "string",

"course\_id": "string",

"course\_title": "string",

"section\_id": "string" | null,

"section\_name": "string" | null,

"email": "string",

"token": "string",

"used": true | false,

"expire\_date": "string",

"created\_at": "string"

}

],

"total": 0,

"page": 1,

"size": 10,

"pages": 0

}

**Delete Invitation**

DELETE /enrollments/invitation/{invitation\_id}

Delete an invitation that hasn't been used yet.

**Response:**

{

"success": true,

"message": "Invitation deleted successfully"

}

**Resend Invitation**

GET /enrollments/invitation/{invitation\_id}/resend

Resend an invitation email.

**Response:**

{

"success": true,

"message": "Invitation resent successfully"

}

**Get Student Enrollments**

GET /enrollments/student/courses

Get courses the student is enrolled in.

**Query Parameters:**

* page: Page number (default: 1)
* size: Items per page (default: 10, max: 100)

**Response:**

{

"items": [

{

"course\_id": "string",

"course\_title": "string",

"course\_code": "string",

"section\_id": "string" | null,

"section\_name": "string" | null,

"enrolled\_at": "string",

"teacher": {

"id": "string",

"name": "string"

}

}

],

"total": 0,

"page": 1,

"size": 10,

"pages": 0

}

**Get Course Students**

GET /enrollments/course/{course\_id}/students

Get students enrolled in a course.

**Query Parameters:**

* section\_id: Filter by section ID
* page: Page number (default: 1)
* size: Items per page (default: 10, max: 100)

**Response:**

{

"items": [

{

"user\_id": "string",

"first\_name": "string",

"last\_name": "string",

"email": "string",

"section\_id": "string" | null,

"section\_name": "string" | null,

"enrolled\_at": "string"

}

],

"total": 0,

"page": 1,

"size": 10,

"pages": 0

}

**Remove Student From Course**

DELETE /enrollments/student/{student\_id}/course/{course\_id}

Remove a student from a course.

**Query Parameters:**

* section\_id: Section ID

**Response:**

{

"success": true,

"message": "Student removed from course successfully"

}

**Get Course Enrollments**

GET /enrollments/course/{course\_id}/enrollments

Get active enrollment methods for a course.

**Response:**

{

"email\_enrollments": [

{

"id": "string",

"enrollment\_key": "string",

"expire\_date": "string",

"created\_at": "string",

"created\_by": "string"

}

],

"public\_link\_enrollments": [

{

"id": "string",

"enrollment\_key": "string",

"enrollment\_link": "string",

"expire\_date": "string",

"created\_at": "string",

"created\_by": "string"

}

]

}

**AI Content Processing**

**Process Content**

POST /ai-content/process

Process content by extracting text, generating embeddings, and creating a summary. This is an asynchronous operation. Use the /status endpoint to check progress.

**Request Body:**

{

"content\_id": "string",

"force\_reprocess": false

}

**Response:**

{

"task\_id": "string",

"content\_id": "string",

"status": "QUEUED" | "PROCESSING" | "COMPLETED" | "FAILED",

"message": "Content processing task has been queued"

}

**Get Processing Status**

GET /ai-content/status/{content\_id}

Get the current status of content processing.

**Response:**

{

"id": "string",

"content\_id": "string",

"status": "QUEUED" | "PROCESSING" | "COMPLETED" | "FAILED",

"processing\_started\_at": "string" | null,

"processing\_completed\_at": "string" | null,

"error\_message": "string" | null,

"created\_at": "string",

"updated\_at": "string",

"file\_count": 0 | null,

"chunk\_count": 0 | null,

"total\_tokens": 0 | null

}

**Get Content Summary**

GET /ai-content/summary/{content\_id}

Get the summary for a specific content item.

**Response:**

{

"id": "string",

"content\_id": "string",

"summary": "string",

"one\_line\_description": "string",

"summary\_model": "string",

"created\_at": "string"

}

**Get Token Usage**

GET /ai-content/token-usage

Get token usage with optional filtering.

**Query Parameters:**

* content\_id: Filter by content ID
* user\_id: Filter by user ID
* operation\_type: Filter by operation type
* model\_name: Filter by model name
* start\_date: Filter by start date
* end\_date: Filter by end date
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

{

"items": [

{

"id": "string",

"content\_id": "string" | null,

"user\_id": "string",

"operation\_type": "string",

"model\_name": "string",

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"created\_at": "string"

}

],

"total\_items": 0,

"total\_input\_tokens": 0,

"total\_output\_tokens": 0,

"total\_cost": 0.0

}

**Get Aggregated Token Usage**

GET /ai-content/token-usage/aggregate

Get aggregated token usage statistics.

**Query Parameters:**

* start\_date: Filter by start date
* end\_date: Filter by end date
* user\_id: Filter by user ID

**Response:**

{

"by\_operation": {

"string": {

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"count": 0

}

},

"by\_model": {

"string": {

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"count": 0

}

},

"by\_user": {

"string": {

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"count": 0

}

},

"by\_content": {

"string": {

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"count": 0

}

},

"total\_input\_tokens": 0,

"total\_output\_tokens": 0,

"total\_cost": 0.0,

"date\_range": {

"start": "string",

"end": "string"

}

}

**Search Content Direct Text**

POST /ai-content/search/direct-text

Search for content using direct text matching instead of vector similarity.

**Request Body:**

{

"query": "string",

"content\_ids": ["string"] | null,

"limit": 5,

"min\_similarity": 0.7

}

**Response:**

{

"results": [

{

"content\_id": "string",

"content\_title": "string",

"file\_id": "string",

"file\_name": "string",

"chunk\_id": "string",

"text": "string",

"similarity": 0.0

}

]

}

**Get Content Chunks**

GET /ai-content/content-chunks/{content\_id}

Get all content chunks for a specific content ID to debug search issues.

**Response:**

{

"content\_id": "string",

"chunks": [

{

"id": "string",

"content\_id": "string",

"file\_id": "string" | null,

"text": "string",

"metadata": {},

"embedding": [] | null,

"created\_at": "string"

}

]

}

**Chat**

**Create Chat Session**

POST /chat/sessions

Create a new chat session.

**Request Body:**

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"content\_id": "string" | null

}

**Response:**

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"id": "string",

"user\_id": "string",

"content\_id": "string" | null,

"is\_active": true,

"last\_message\_at": "string",

"created\_at": "string",

"updated\_at": "string",

"message\_count": 0

}

**Get User Chat Sessions**

GET /chat/sessions

Get all chat sessions for the current user with optional filtering.

**Query Parameters:**

* content\_id: Filter by content ID
* active\_only: Only return active sessions (default: true)
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

{

"items": [

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"id": "string",

"user\_id": "string",

"content\_id": "string" | null,

"is\_active": true,

"last\_message\_at": "string",

"created\_at": "string",

"updated\_at": "string",

"message\_count": 0

}

],

"total": 0,

"page": 1,

"size": 20,

"pages": 0

}

**Get Chat Session**

GET /chat/sessions/{session\_id}

Get a chat session by ID.

**Query Parameters:**

* include\_messages: Include messages in response (default: false)

**Response:**

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"id": "string",

"user\_id": "string",

"content\_id": "string" | null,

"is\_active": true,

"last\_message\_at": "string",

"created\_at": "string",

"updated\_at": "string",

"message\_count": 0,

"messages": [] // If include\_messages=true

}

**Update Chat Session**

PATCH /chat/sessions/{session\_id}

Update a chat session.

**Request Body:**

{

"title": "string" | null,

"description": "string" | null,

"is\_active": true | false | null,

"settings": {} | null

}

**Response:**

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"id": "string",

"user\_id": "string",

"content\_id": "string" | null,

"is\_active": true,

"last\_message\_at": "string",

"created\_at": "string",

"updated\_at": "string",

"message\_count": 0

}

**Delete Chat Session**

DELETE /chat/sessions/{session\_id}

Delete a chat session.

**Response:** HTTP 204 No Content

**Send Message**

POST /chat/sessions/{session\_id}/messages

Send a message to a chat session.

**Request Body:**

{

"message": "string",

"system\_prompt": "string" | null,

"session\_settings": {} | null

}

**Response:**

{

"session\_id": "string",

"message": {

"role": "USER" | "ASSISTANT" | "SYSTEM" | "FUNCTION" | "TOOL",

"content": "string",

"name": "string" | null,

"function\_call": {} | null,

"tool\_calls": [] | null,

"tool\_call\_id": "string" | null,

"id": "string",

"chat\_session\_id": "string",

"sequence": 0,

"created\_at": "string",

"token\_count": 0

},

"token\_usage": {

"input": 0,

"output": 0,

"total": 0

},

"cost": 0.0,

"created\_at": "string"

}

**Get Session Messages**

GET /chat/sessions/{session\_id}/messages

Get all messages for a chat session with pagination.

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 50, max: 100)

**Response:**

[

{

"role": "USER" | "ASSISTANT" | "SYSTEM" | "FUNCTION" | "TOOL",

"content": "string",

"name": "string" | null,

"function\_call": {} | null,

"tool\_calls": [] | null,

"tool\_call\_id": "string" | null,

"id": "string",

"chat\_session\_id": "string",

"sequence": 0,

"created\_at": "string",

"token\_count": 0

}

]

**Get Chat Token Usage**

GET /chat/token-usage

Get token usage for chat sessions.

**Query Parameters:**

* session\_id: Filter by session ID
* start\_date: Filter by start date
* end\_date: Filter by end date
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

[

{

"id": "string",

"chat\_session\_id": "string",

"user\_id": "string",

"operation\_type": "string",

"model\_name": "string",

"input\_tokens": 0,

"output\_tokens": 0,

"cost": 0.0,

"created\_at": "string"

}

]

**Get Content Chat Sessions**

GET /chat/content/{content\_id}/sessions

Get all chat sessions for a specific content item.

**Query Parameters:**

* active\_only: Only return active sessions (default: true)
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

{

"items": [

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"id": "string",

"user\_id": "string",

"content\_id": "string" | null,

"is\_active": true,

"last\_message\_at": "string",

"created\_at": "string",

"updated\_at": "string",

"message\_count": 0

}

],

"total": 0,

"page": 1,

"size": 20,

"pages": 0

}

**Debug RAG**

GET /chat/debug-rag/{session\_id}

Debug the RAG functionality for a specific chat session.

**Response:**

{

"session\_id": "string",

"content\_id": "string",

"chat\_mode": "RAG",

"retrieval\_history": [

{

"message\_id": "string",

"query": "string",

"retrieved\_chunks": [

{

"chunk\_id": "string",

"text": "string",

"similarity": 0.0,

"file\_id": "string" | null,

"file\_name": "string" | null

}

],

"timestamp": "string"

}

]

}

**API Tokens**

**Create API Token**

POST /api-tokens/

Create a new API token.

The token will be shown only once in the response. Make sure to copy it as it won't be displayed again for security reasons.

**Request Body:**

{

"name": "string",

"expires\_in\_days": 30 | null

}

**Response:**

{

"id": "string",

"name": "string",

"token": "string",

"user\_id": "string",

"expires\_at": "string" | null,

"created\_at": "string",

"last\_used\_at": "string" | null

}

**List API Tokens**

GET /api-tokens/

List API tokens.

* Regular users: See only their own tokens
* Admins: Can filter by user\_id to see tokens for any user

For security reasons, only a preview of each token is shown.

**Query Parameters:**

* user\_id: Filter tokens by user ID (admin only)
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

[

{

"id": "string",

"name": "string",

"token\_preview": "string",

"user\_id": "string",

"expires\_at": "string" | null,

"created\_at": "string",

"last\_used\_at": "string" | null,

"is\_active": true | null,

"username": "string" | null

}

]

**Get API Token**

GET /api-tokens/{token\_id}

Get details of a specific API token.

Regular users can only view their own tokens. Admins can view any token.

For security reasons, only a preview of the token is shown.

**Response:**

{

"id": "string",

"name": "string",

"token\_preview": "string",

"user\_id": "string",

"expires\_at": "string" | null,

"created\_at": "string",

"last\_used\_at": "string" | null,

"is\_active": true | null,

"username": "string" | null

}

**Revoke API Token**

DELETE /api-tokens/{token\_id}

Revoke (deactivate) an API token.

Regular users can only revoke their own tokens. Admins can revoke any token.

Once revoked, a token cannot be used anymore and will not appear in the token list.

**Response:**

{

"message": "Token revoked successfully"

}

**Get All Tokens (Admin)**

GET /api-tokens/admin/all

Get all API tokens in the system (admin only).

This endpoint is restricted to admin users.

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)
* include\_inactive: Include inactive/revoked tokens (default: false)

**Response:**

[

{

"id": "string",

"name": "string",

"token\_preview": "string",

"user\_id": "string",

"expires\_at": "string" | null,

"created\_at": "string",

"last\_used\_at": "string" | null,

"is\_active": true | null,

"username": "string" | null

}

]

**Notifications**

**Subscribe to Server-Sent Events**

GET /notifications/sse

Subscribe to Server-Sent Events for real-time notifications.

**Response:** Event stream with notification data.

**Get Notifications**

GET /notifications

Get user notifications with optional filtering.

**Query Parameters:**

* unread\_only: Get only unread notifications (default: false)
* notification\_type: Filter by notification type
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

{

"items": [

{

"id": "string",

"user\_id": "string",

"message": "string",

"notification\_type": "ACCOUNT" | "SECURITY" | "ADMIN" | "SYSTEM" | "SESSION" | "INFO" | "WARNING" | "TEST" | "INSTITUTION",

"is\_read": false,

"created\_at": "string",

"data": {} | null

}

],

"total": 0,

"unread\_count": 0,

"page": 1,

"size": 20,

"pages": 0

}

**Create Notification (Admin)**

POST /notifications

Create a new notification for a specific user.

**Query Parameters:**

* user\_id: User ID (required)
* message: Notification message (required)
* notification\_type: Notification type (default: "SYSTEM")

**Request Body:** Optional data object to include with the notification.

**Response:**

{

"id": "string",

"user\_id": "string",

"message": "string",

"notification\_type": "ACCOUNT" | "SECURITY" | "ADMIN" | "SYSTEM" | "SESSION" | "INFO" | "WARNING" | "TEST" | "INSTITUTION",

"is\_read": false,

"created\_at": "string",

"data": {} | null

}

**Delete All Notifications**

DELETE /notifications

Delete all notifications for the current user.

**Query Parameters:**

* read\_only: Delete only read notifications (default: false)

**Response:**

{

"deleted\_count": 0,

"message": "Notifications deleted successfully"

}

**Mark Notification as Read**

PATCH /notifications/{notification\_id}/read

Mark a specific notification as read.

**Response:**

{

"id": "string",

"is\_read": true,

"message": "Notification marked as read"

}

**Mark All Notifications as Read**

PATCH /notifications/read-all

Mark all notifications for the current user as read.

**Query Parameters:**

* notification\_type: Filter by notification type

**Response:**

{

"updated\_count": 0,

"message": "All notifications marked as read"

}

**Delete Notification**

DELETE /notifications/{notification\_id}

Delete a specific notification.

**Response:**

{

"message": "Notification deleted successfully"

}

**Email**

**Send Welcome Email**

POST /email/send-welcome/{user\_id}

Send welcome email to user (admin only).

**Response:**

{

"success": true,

"message": "Welcome email sent successfully"

}

**Send Custom Notification Email**

POST /email/notification

Send custom notification email (admin only).

**Request Body:**

{

"user\_id": "string",

"message": "string",

"notification\_type": "ACCOUNT" | "SECURITY" | "ADMIN" | "SYSTEM" | "SESSION" | "INFO" | "WARNING" | "TEST" | "INSTITUTION"

}

**Response:**

{

"success": true,

"message": "Notification email sent successfully"

}

**Schema Definitions**

**UserCreate**

{

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "SUPERADMIN" | "ADMIN" | "OFFICER" | "TEACHER" | "STUDENT",

"password": "string",

"country": "string" | null,

"date\_of\_birth": "YYYY-MM-DD" | null,

"phone\_number": "string" | null,

"institution\_email": "string" | null,

"bio": "string" | null,

"status": "string" | null

}

**UserLogin**

{

"email": "string",

"password": "string"

}

**UserUpdateSchema**

{

"first\_name": "string" | null,

"last\_name": "string" | null,

"username": "string" | null,

"email": "string" | null,

"bio": "string" | null,

"profile\_image\_url": "string" | null,

"country": "string" | null,

"date\_of\_birth": "YYYY-MM-DD" | null,

"phone\_number": "string" | null,

"institution\_email": "string" | null

}

**UserStatusUpdate**

{

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "PENDING" | "ARCHIVED"

}

**SessionLimitUpdate**

{

"session\_limit": 1-10

}

**PasswordChangeSchema**

{

"new\_password": "string"

}

**InstitutionCreate**

{

"name": "string",

"type": "UNIVERSITY" | "COLLEGE" | "SCHOOL" | "TRAINING\_CENTER",

"email": "string" | null,

"country\_code": "string" | null,

"phone\_number": "string" | null,

"website": "string" | null,

"address": "string" | null,

"university\_category": "PUBLIC" | "PRIVATE" | "INTERNATIONAL" | null,

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED"

}

**InstitutionUpdate**

{

"name": "string" | null,

"type": "UNIVERSITY" | "COLLEGE" | "SCHOOL" | "TRAINING\_CENTER" | null,

"email": "string" | null,

"country\_code": "string" | null,

"phone\_number": "string" | null,

"website": "string" | null,

"address": "string" | null,

"university\_category": "PUBLIC" | "PRIVATE" | "INTERNATIONAL" | null,

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED" | null

}

**FileResponse**

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

**CourseCreate**

{

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null

}

**ContentResponse**

{

"title": "string",

"description": "string" | null,

"course\_id": "string",

"id": "string",

"teacher\_id": "string",

"created\_at": "string",

"updated\_at": "string",

"teacher": {

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

}

**ChatSessionCreate**

{

"title": "string",

"description": "string" | null,

"chat\_mode": "STANDARD" | "ADVANCED" | "RAG",

"settings": {} | null,

"content\_id": "string" | null

}

**APITokenCreate**

{

"name": "string",

"expires\_in\_days": 0 | null

}

**Error Handling**

**Error Responses**

The API uses standard HTTP status codes to indicate the success or failure of requests:

* **2xx**: Success
  + 200: OK
  + 201: Created
  + 204: No Content
* **4xx**: Client Errors
  + 400: Bad Request
  + 401: Unauthorized
  + 403: Forbidden
  + 404: Not Found
  + 422: Unprocessable Entity
* **5xx**: Server Errors
  + 500: Internal Server Error
  + 503: Service Unavailable

**Validation Errors**

Validation errors return a 422 status code with a response body containing details about the validation failure:

{

"detail": [

{

"loc": ["body", "field\_name"],

"msg": "Error message",

"type": "error\_type"

}

]

}

**Conclusion**

This documentation provides a comprehensive overview of the AI-Professor API. For specific implementation details, please refer to the source code or contact the system administrator.

For any issues, bugs, or feature requests, please submit them through the appropriate channels defined by your organization's support process.

© 2025 AI-Professor. All rights reserved. { "first\_name": "string", "last\_name": "string", "username": "string", "email": "string", "role": "TEACHER" | "STUDENT", "password": "string", "country": "string" | null, "date\_of\_birth": "YYYY-MM-DD" | null, "phone\_number": "string" | null, "institution\_email": "string" | null, "bio": "string" | null, "status": "string" | null }

\*\*Responses:\*\*

- 201: Account created successfully

- 422: Validation error

### Register a new admin or officer account

POST /auth/register/admin-officer

Creates a new user account with admin or officer role. Requires admin privileges.

\*\*Request Body:\*\*

Same as student/teacher registration.

\*\*Responses:\*\*

- 201: Account created successfully

- 401: Unauthorized

- 403: Forbidden

- 422: Validation error

### Register a new superadmin account

POST /auth/register/superadmin

Creates a new superadmin account. May require special authorization.

\*\*Request Body:\*\*

Same as student/teacher registration.

\*\*Responses:\*\*

- 200: Account created successfully

- 401: Unauthorized

- 403: Forbidden

- 422: Validation error

### Login

POST /auth/login

Authenticate and receive access and refresh tokens.

\*\*Request Body:\*\*

```json

{

"email": "string",

"password": "string"

}

**Response:**

{

"access\_token": "string",

"refresh\_token": "string",

"token\_type": "bearer",

"expires\_in": 3600,

"user": {

// User details

}

}

**Logout**

POST /auth/logout

Invalidate the current session.

**Request Body:**

{

"refresh\_token": "string"

}

**Response:**

{

"message": "Successfully logged out"

}

**Refresh Token**

POST /auth/refresh-token

Get a new access token using a refresh token.

**Request Body:**

{

"refresh\_token": "string"

}

**Response:**

{

"access\_token": "string",

"token\_type": "bearer",

"expires\_in": 3600

}

**Request Password Reset**

POST /auth/password/request-reset

Request a password reset email.

**Request Body:**

{

"email": "string"

}

**Response:**

{

"message": "If a user with this email exists, a password reset link has been sent"

}

**Reset Password**

POST /auth/password/reset

Reset password using token received via email.

**Request Body:**

{

"token": "string",

"new\_password": "string"

}

**Response:**

{

"message": "Password reset successful"

}

**Terminate Session**

POST /auth/terminate-session/{session\_id}

Terminate a specific user session.

**Response:**

{

"message": "Session terminated successfully"

}

**Change Temporary Password**

POST /auth/change-temporary-password

Change a temporary password assigned by an administrator.

**Request Body:**

{

"new\_password": "string"

}

**Response:**

{

"message": "Password changed successfully"

}

**User Management**

**Get Users**

GET /users

Get a list of all users with optional filtering.

**Query Parameters:**

* search: Search term for user fields
* role: Filter by user role
* userStatus: Filter by user status
* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)

**Response:**

{

"items": [

{

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string",

"status": "string",

// Additional user fields

}

],

"total": 0,

"page": 1,

"size": 20,

"pages": 0

}

**Get User Profile**

GET /users/{user\_id}

Get detailed user profile by ID.

**Response:**

{

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string",

"status": "string",

// Additional user fields

}

**Update User Profile**

PATCH /users/{user\_id}

Update user profile information.

**Request Body:**

{

"first\_name": "string" | null,

"last\_name": "string" | null,

"username": "string" | null,

"email": "string" | null,

"bio": "string" | null,

"profile\_image\_url": "string" | null,

"country": "string" | null,

"date\_of\_birth": "YYYY-MM-DD" | null,

"phone\_number": "string" | null,

"institution\_email": "string" | null

}

**Response:**

{

"id": "string",

"first\_name": "string",

"last\_name": "string",

// Updated user details

}

**Delete User**

DELETE /users/{user\_id}

Delete a user account (soft delete).

**Response:**

{

"message": "User deleted successfully"

}

**Get Current User Profile**

GET /users/profile

Get the currently authenticated user's profile.

**Response:**

{

"id": "string",

"first\_name": "string",

"last\_name": "string",

// User details

}

**Get User Sessions**

GET /users/sessions

Get all active sessions for the current user.

**Response:**

{

"sessions": [

{

"id": "string",

"created\_at": "string",

"last\_active\_at": "string",

"ip\_address": "string",

"user\_agent": "string",

"is\_current": true

}

]

}

**Change User Status**

PATCH /users/{user\_id}/status

Change a user's status (requires appropriate permissions).

**Request Body:**

{

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "PENDING" | "ARCHIVED"

}

**Response:**

{

"id": "string",

"status": "string",

"message": "User status updated successfully"

}

**Resend Verification Email**

POST /users/verification/resend

Resend email verification link.

**Request Body:**

{

"email": "string"

}

**Response:**

{

"message": "Verification email resent successfully"

}

**Verify Email**

GET /users/verification/verify?token={token}

Verify email using token.

**Response:**

{

"message": "Email verified successfully"

}

**Archive Own Account**

POST /users/archive-account

Archive your own account.

**Response:**

{

"message": "Account archived successfully"

}

**Permanently Delete User**

DELETE /users/{user\_id}/permanent

Permanently delete a user account (admin only).

**Response:**

{

"message": "User permanently deleted"

}

**Get User Sessions by ID**

GET /users/{user\_id}/sessions

Get all active sessions for a user.

**Response:**

{

"sessions": [

{

"id": "string",

"created\_at": "string",

"last\_active\_at": "string",

"ip\_address": "string",

"user\_agent": "string"

}

]

}

**Update Session Limit**

PATCH /users/{user\_id}/session-limit

Update a user's session limit (admin only).

**Request Body:**

{

"session\_limit": 5

}

**Response:**

{

"message": "Session limit updated successfully"

}

**Change User Password (Admin)**

POST /users/{user\_id}/change-password

Change a user's password (admin only).

**Request Body:**

{

"new\_password": "string"

}

**Response:**

{

"message": "Password changed successfully"

}

**Institutions**

**Create Institution**

POST /institutions/

Create a new institution (Admin-only).

**Request Body:**

{

"name": "string",

"type": "UNIVERSITY" | "COLLEGE" | "SCHOOL" | "TRAINING\_CENTER",

"email": "string" | null,

"country\_code": "string" | null,

"phone\_number": "string" | null,

"website": "string" | null,

"address": "string" | null,

"university\_category": "PUBLIC" | "PRIVATE" | "INTERNATIONAL" | null,

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED"

}

**Response:**

{

"id": "string",

"name": "string",

"type": "string",

// Institution details

}

**Get Institution by ID**

GET /institutions/{institution\_id}

Retrieve an institution by its ID.

**Response:**

{

"id": "string",

"name": "string",

"type": "string",

// Institution details

}

**Update Institution**

PATCH /institutions/{institution\_id}

Update an existing institution (Admin-only).

**Request Body:**

{

"name": "string" | null,

"type": "UNIVERSITY" | "COLLEGE" | "SCHOOL" | "TRAINING\_CENTER" | null,

"email": "string" | null,

"country\_code": "string" | null,

"phone\_number": "string" | null,

"website": "string" | null,

"address": "string" | null,

"university\_category": "PUBLIC" | "PRIVATE" | "INTERNATIONAL" | null,

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED" | null

}

**Response:**

{

"id": "string",

"name": "string",

// Updated institution details

}

**Archive Institution**

DELETE /institutions/{institution\_id}

Archive an institution by its ID (Admin-only).

**Response:**

{

"message": "Institution archived successfully"

}

**Get Institution by Name**

GET /institutions/name/{name}

Retrieve an institution by its name.

**Response:**

{

"id": "string",

"name": "string",

// Institution details

}

**Get Institutions**

GET /institutions

Retrieve a paginated list of institutions.

**Query Parameters:**

* page: Page number (default: 1)
* limit: Number of items per page (default: 15)

**Response:**

{

"items": [

{

"id": "string",

"name": "string",

"type": "string",

// Institution details

}

],

"total": 0,

"page": 1,

"size": 15,

"pages": 0

}

**Get Institution Users**

GET /institutions/{institution\_id}/users

Retrieve all users associated with an institution.

**Response:**

{

"users": [

{

"id": "string",

"first\_name": "string",

"last\_name": "string",

"username": "string",

"email": "string",

"role": "string"

}

]

}

**Batch Update Institution Status**

POST /institutions/batch-status-update

Update the status of multiple institutions at once (Admin-only).

**Request Body:**

{

"institution\_ids": [

"string"

],

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED"

}

**Response:**

{

"updated": [

"string" // IDs of updated institutions

],

"not\_found": [

"string" // IDs of institutions not found

]

}

**Update Institution Status**

PATCH /institutions/{institution\_id}/status

Update only the status of an institution (Admin-only).

**Request Body:**

{

"status": "ACTIVE" | "INACTIVE" | "SUSPENDED" | "ARCHIVED"

}

**Response:**

{

"id": "string",

"name": "string",

"status": "string",

// Institution details

}

**Universal Search for Institutions**

GET /institutions/search/universal?search\_term={search\_term}

Universal search for institutions.

**Query Parameters:**

* search\_term: Universal search term for ID, name, or type
* page: Page number (default: 1)
* limit: Number of items per page (default: 15)

**Response:**

{

"items": [

{

"id": "string",

"name": "string",

"type": "string",

// Institution details

}

],

"total": 0,

"page": 1,

"size": 15,

"pages": 0

}

**File Management**

**Upload File**

POST /files/

Upload a new file to the system.

**Request Body (multipart/form-data):**

* file: The file to upload (required)
* directory: Optional directory to store the file in
* visibility: Whether the file should be publicly accessible (default: false)

**Response:**

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

**Get File**

GET /files/{file\_id}

Get file details by ID.

**Response:**

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

**Update File**

PUT /files/{file\_id}

Update file details.

**Request Body:**

{

"filename": "string" | null,

"file\_rename": "string" | null,

"file\_type": "string" | null,

"file\_size": 0 | null,

"file\_path": "string" | null,

"directory": "string" | null,

"visibility": true | false | null,

"user\_id": "string" | null

}

**Response:**

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

**Delete File**

DELETE /files/{file\_id}

Delete a file.

**Response:**

{

"message": "File deleted successfully"

}

**Upload Multiple Files**

POST /files/upload-multiple

Upload multiple files at once.

**Request Body (multipart/form-data):**

* files: Array of files to upload (required)
* directory: Optional directory to store the files in
* visibility: Whether the files should be publicly accessible (default: false)

**Response:**

[

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

]

**Delete Multiple Files**

DELETE /files/batch-delete

Delete multiple files by their IDs.

**Request Body:**

[

"string" // File IDs

]

**Response:**

{

"deleted": [

"string" // IDs of deleted files

],

"not\_found": [

"string" // IDs of files not found

]

}

**Get User Files**

GET /files/me/files

Get a paginated list of the current user's files.

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)
* visibility: Filter by visibility (true/false)
* directory: Filter by directory
* file\_type: Filter by file type
* search: Search term for filename

**Response:**

{

"items": [

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": false,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

],

"total": 0,

"page": 1,

"size": 20,

"pages": 0

}

**Get All Files (Admin)**

GET /files/admin/files

Get a paginated list of all files (admin/superadmin only).

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)
* user\_id: Filter by user ID
* visibility: Filter by visibility (true/false)
* directory: Filter by directory
* file\_type: Filter by file type
* search: Search term for filename

**Response:** Similar to Get User Files.

**Get Public Files**

GET /files/public

Get a paginated list of all public files.

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)
* file\_type: Filter by file type
* search: Search term for filename

**Response:** Similar to Get User Files.

**Get Files By Directory**

GET /files/directory/{directory\_path}

Get a paginated list of files in a specific directory.

**Query Parameters:**

* page: Page number (default: 1)
* page\_size: Items per page (default: 20, max: 100)
* user\_id: Filter by user ID
* public\_only: Only return public files (default: false)

**Response:** Similar to Get User Files.

**Set File Visibility**

PATCH /files/{file\_id}/visibility

Set a file's visibility status (public or private).

**Request Body (x-www-form-urlencoded):**

* visibility: New visibility status (true/false)

**Response:**

{

"filename": "string",

"file\_rename": "string",

"file\_type": "string",

"file\_size": 0,

"file\_path": "string",

"directory": "string" | null,

"visibility": true,

"user\_id": "string",

"id": "string",

"created\_at": "string"

}

**Download File**

GET /files/download/{file\_id}

Download a file.

**Response:** File data with appropriate Content-Type and Content-Disposition headers.

**Stream File**

GET /files/stream/{file\_id}

Stream a file for in-browser viewing.

**Headers:**

* Range: HTTP Range header for partial content (video/audio streaming)

**Response:** File data with appropriate Content-Type and partial content headers if Range was specified.

**View Public File**

GET /files/public/view/{file\_id}

View a public file without authentication.

**Headers:**

* Range: HTTP Range header for partial content (video/audio streaming)

**Response:** File data if the file is public.

**Download Public File**

GET /files/public/download/{file\_id}

Download a public file without authentication.

**Response:** File data with appropriate Content-Type and Content-Disposition headers if the file is public.

**Courses**

**Create Course**

POST /courses

Create a new course without sections. Only teachers can create courses.

**Request Body:**

{

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null

}

**Response:**

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string"

}

**Create Course With Sections**

POST /courses/sections

Create a new course with multiple sections. Only teachers can create courses.

**Request Body:**

{

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"sections": [

{

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

]

}

**Response:**

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string",

"sections": [

{

"id": "string",

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

]

}

**Get Course**

GET /courses/{course\_id}

Get a course by its ID with all its sections.

**Response:**

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string",

"sections": [

{

"id": "string",

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

]

}

**Update Course**

PUT /courses/{course\_id}

Update a course. Only the teacher who created the course or an admin can update it.

**Request Body:**

{

"title": "string" | null,

"code": "string" | null,

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED" | null,

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null

}

**Response:**

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string",

"updated\_at": "string"

}

**Delete Course**

DELETE /courses/{course\_id}

Delete a course and all its sections. Only the teacher who created the course or an admin can delete it.

**Response:**

{

"message": "Course deleted successfully"

}

**Get Teacher Courses**

GET /courses/course

Get all courses created by the current teacher.

**Query Parameters:**

* skip: Number of records to skip (default: 0)
* limit: Maximum number of records to return (default: 100)
* include\_sections: Whether to include sections in the response (default: true)

**Response:**

{

"courses": [

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string",

"sections": [

{

"id": "string",

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

]

}

]

}

**Add Course Section**

POST /courses/sections/{course\_id}

Add a new section to an existing course.

**Request Body:**

{

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

**Response:**

{

"id": "string",

"course\_id": "string",

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED",

"created\_at": "string"

}

**Get Teacher Course With Sections**

GET /courses/teacher/{course\_id}

Get a specific course with all its sections for the current teacher. Only the teacher who created the course can access this endpoint.

**Response:**

{

"id": "string",

"title": "string",

"code": "string",

"description": "string" | null,

"level": "BEGINNER" | "INTERMEDIATE" | "ADVANCED",

"start\_date": "YYYY-MM-DD" | null,

"end\_date": "YYYY-MM-DD" | null,

"created\_by": "string",

"created\_at": "string",

"sections": [

{

"id": "string",

"section\_name": "string",

"status": "ACTIVE" | "INACTIVE" | "ARCHIVED"

}

]

}

**Configure Course Enrollment**

POST /courses/enrollment-config/{course\_id}

Toggle enrollment status for a course. Only the teacher who created the course or users with admin roles can do this.

**Request Body:**

{

"status": true | false,

"expire\_date": "YYYY-MM-DD" | null // Required when status=true

}

**Response:**

{

"course\_id": "string",

"enrollment\_active": true | false,

"expire\_date": "YYYY-MM-DD" | null,

"updated\_at": "string"

}

**Get Course Enrollment Config**

GET /courses/enrollment-config/{course\_id}

Get enrollment configuration for a course.

**Response:**

{

"course\_id": "string",

"enrollment\_active": true | false,

"expire\_date": "YYYY-MM-DD" | null,

"created\_at": "string",

"updated\_at": "string"

}

**Update Enrollment Expiration Date**

PUT /courses/enrollment-config/expire-date/{course\_id}

Update the expiration date of a course enrollment configuration.