newDocs

Page • 1 backlink • Tag

Frontend API Documentation

All endpoints require authentication. Include an Authorization header with your JWT token (e.g., `Authorization: Token <your_token_here>`). Replace IDs with actual values from your application.

---

0. App Launch:

- To launch the app install daphne:
- `pip3 install daphne`
- and then :

`daphne myproject.asgi:application`

---

1. Authentication

a. User Registration

- **Endpoint:**

  `POST /api/user/register/`

- **Request Body:**

```
JSON
{
  "username": "student1",
  "password": "yourpassword",
  "email": "student1@example.com",
  "role": "student"
}
```

- **Response:**

  Returns created user information.

b. User Login (Obtain JWT Token)

- **Endpoint:**

  `POST /api/token/`

- **Request Body:**

```
JSON
{
  "username": "student1",
  "password": "yourpassword"
}
```

- **Response Example:**

```
JSON
{
  "access": "eyJ0eXAiOiJKV1QiLCJh...",
  "refresh": "eyJ0eXAiOiJKV1QiLCJh..."
}
```

c. Refresh Token

- **Endpoint:**

  `POST /api/token/refresh/`

- **Request Body:**

```
JSON
{
  "refresh": "<your_refresh_token>"
}
```

---

2. Classes and Enrollments

a. List & Create Classes

- **Endpoint:**

  `GET /class/classes/`

  `POST /class/classes/` (Educators only)

- **POST Request Example:**

```JSON
{
  "title": "Mathematics 101",
  "tags": ["math", "basics"],
  "schedule": "Mon 9am-11am",
  "description": "A basic math class",
  "syllabus": "Algebra, Geometry, etc."
}
```

- **GET Response Example (List):**

```JSON
[
  {
    "id": 1,
    "title": "Mathematics 101",
    "author": "educator1",
    "tags": ["math", "basics"],
    "schedule": "Mon 9am-11am",
    "description": "A basic math class",
    "syllabus": "Algebra, Geometry, etc.",
    "enrolled_students": ["student1", "student2"]
  }
]
```

b. Retrieve, Update, Delete a Class (Educators only)

- **Endpoint:**

  `GET/PUT/PATCH/DELETE /class/classes/<int:pk>/`

- **Usage:**

  Only the educator (author) can update or delete their classes.

c. Create Enrollment Request (Students)

- **Endpoint:**

`POST /class/enroll/`

- **Request Body Example:**

```
JSON
{
  "class_obj": 1
}
```

- **Response:**

  Enrollment request with status defaulted to "pending".

d. Manage Enrollment Status (Educators)

- **List All Enrollment Requests for Educator's Classes:**

  Endpoint:

  `GET /class/enrollments/`

- **Update Enrollment Status:**

  Endpoint:

  `PATCH /class/enrollments/<int:pk>/`

  Request Body Example:

```
JSON
{
  "status": "accepted"
}
```

- **Cancel Enrollment (Students):**

  Endpoint:

  `DELETE /class/enrollments/<int:pk>/`

e. List Enrollments for Students

- **Pending Enrollment Requests:**

  Endpoint:

  `GET /class/student/enrollments/pending/`

- **Accepted Enrollment Requests:**

  Endpoint:

`GET /class/student/enrollments/accepted/`

f. List Available and Accepted Classes for Students

- **Available Classes (not enrolled):**

  Endpoint:

  `GET /class/student/classes/available/`

- **Accepted Classes (enrolled):**

  Endpoint:

  `GET /class/student/classes/accepted/`

---

3. Docs (Documents)

a. Upload Document (Educators only)

- **Endpoint:**

  `POST /docs/<int:class_id>/upload/`

- **Request:** Use multipart/form-data with these fields:
  - **title:** (String)
  - **file_type:** (e.g., "pdf")
  - **file_size:** (e.g., "1.2MB")
  - **file:** (The file to upload)

- **Response:**

  JSON representing the uploaded document, including a URL to download it.

b. List Documents for a Class

- **Endpoint:**

  `GET /docs/<int:class_id>/list/`

- **Response Example:**

```
JSON
[
  {
    "id": 5,
    "title": "Lecture 1 Notes",
    "file_type": "pdf",
    "file_size": "1.2MB",
    "author": "educator1",
    "class_obj": 1,
```

```
    "file": "http://127.0.0.1:8000/documents/lecture1_notes.pdf"
  }
]
```

c. Download Document

- **Endpoint:**

  `GET /documents/<str:filename>/`

- **Usage:**

  This endpoint streams the file as a download.

- **Example URL:**

  `http://127.0.0.1:8000/documents/lecture1_notes.pdf`

d. Delete Document (Educators only)

- **Endpoint:**

  `DELETE /docs/delete/<int:pk>/`

---

4. Chat & Chatbot (Realtime via WebSockets)

a. Group Chat for a Class

- **WebSocket URL:**

  `ws://127.0.0.1:8000/ws/chat/<class_id>/`

- **Usage (JavaScript Example):**

```javascript
// Connect to the group chat for class with ID 1:
const socket = new WebSocket("ws://127.0.0.1:8000/ws/chat/1/?
token=YOUR_JWT_TOKEN");
socket.onopen = () => console.log("Connected to group chat");
socket.onmessage = (event) => {
  const data = JSON.parse(event.data);
  console.log("Message from group chat:", data);
};
// To send a message:
socket.send(JSON.stringify({ message: "Hello everyone!" }));
```

b. Chatbot

- **WebSocket URL:**

  `ws://127.0.0.1:8000/ws/chatbot/`

- **Usage (JavaScript Example):**

```javascript
// Connect to the chatbot:
const chatbotSocket = new
WebSocket("ws://127.0.0.1:8000/ws/chatbot/?
token=YOUR_JWT_TOKEN");
chatbotSocket.onopen = () => console.log("Connected to
chatbot");
chatbotSocket.onmessage = (event) => {
  const data = JSON.parse(event.data);
  console.log("Chatbot says:", data.message);
};
// To send a message:
chatbotSocket.send(JSON.stringify({ message: "What is the
meaning of life?" }));
```

---

5. Stats (Ratings & Feedback)

a. Ratings

i. Create Rating

- **Endpoint:**

  `POST /stats/rate/`

- **Request Body:**

```json
{
  "class_obj": 1,
  "rating": 4
}
```

- **Response Example:**

```json
{
  "id": 15,
```

```
  "student": "student1",
  "class_obj": 1,
  "rating": 4,
  "created_at": "2025-03-16T20:45:00Z"
}
```

ii. Average Rating for a Class

- **Endpoint:**

  `GET /stats/class/1/average/`

- **Response:**

```
JSON
{ "average": 3.8 }
```

iii. Average Rating for an Educator

- **Endpoint:**

  `GET /stats/educator/2/average/`

- **Response:**

```
JSON
{ "average": 4.2 }
```

b. Feedback (with Sentiment Analysis using Hugging Face)

i. Submit Feedback

- **Endpoint:**

  `POST /stats/feedback/`

- **Request Body:**

```
JSON
{
  "class_obj": 1,
  "text": "The course was engaging and informative."
}
```

- **Response Example:**

```JSON
{
  "id": 8,
  "student": "student1",
  "class_obj": 1,
  "text": "The course was engaging and informative.",
  "sentiment_score": 0.92,
  "sentiment_label": "POSITIVE",
  "created_at": "2025-03-16T21:00:00Z"
}
```

ii. Average Sentiment for a Class

- **Endpoint:**

  `GET /stats/class/1/sentiment/`

- **Response:**

```JSON
{ "average": 0.85 }
```

iii. Average Sentiment for an Educator

- **Endpoint:**

  `GET /stats/educator/2/sentiment/`

- **Response:**

```JSON
{ "average": 0.80 }
```

---

6. Tests (Online Examinations)

a. Create Test (Educators only)

- **Endpoint:**

  `POST /tests/create/`

- **Request Body Example:**

```JSON
{
  "title": "Midterm Exam",
  "description": "Exam covering chapters 1-5",
  "class_obj": 1,
  "questions": [
    {
      "text": "What is 2+2?",
      "order": 1,
      "answer_options": [
        {"text": "3", "is_correct": false},
        {"text": "4", "is_correct": true},
        {"text": "5", "is_correct": false},
        {"text": "22", "is_correct": false}
      ]
    },
    {
      "text": "What is the derivative of x^2?",
      "order": 2,
      "answer_options": [
        {"text": "x", "is_correct": false},
        {"text": "2x", "is_correct": true},
        {"text": "x^2", "is_correct": false},
        {"text": "2", "is_correct": false}
      ]
    }
  ]
}
```

b. List Tests for a Class

- **Endpoint:**

  `GET /tests/list/?class_id=1`

- **Response Example:**

```JSON
[
  {
    "id": 3,
    "title": "Midterm Exam",
    "description": "Exam covering chapters 1-5",
```

```
    "class_obj": 1,
    "created_by": "educator1",
    "created_at": "2025-03-16T20:30:00Z",
    "questions": [
      {
        "id": 7,
        "text": "What is 2+2?",
        "order": 1,
        "answer_options": [
          {"id": 15, "text": "3"},
          {"id": 16, "text": "4"},
          {"id": 17, "text": "5"},
          {"id": 18, "text": "22"}
        ]
      },
      { ... }
    ]
  }
]
```


c. Delete Test (Educators only)

- **Endpoint:**

  `DELETE /tests/delete/<int:pk>/`

WebSocket Endpoints Recap

- **Group Chat:**

  URL: `ws://127.0.0.1:8000/ws/chat/<class_id>/?token=YOUR_JWT_TOKEN`

  Example: For class with ID 1:

  `ws://127.0.0.1:8000/ws/chat/1/?token=YOUR_JWT_TOKEN`

- **Chatbot:**

  URL: `ws://127.0.0.1:8000/ws/chatbot/?token=YOUR_JWT_TOKEN`

  Connect and exchange messages with the Studysphere helper.

Final Remarks

- **Authentication:** All endpoints require a valid JWT token.

- **Request/Response Format:** JSON is used for REST endpoints; WebSockets use JSON messages.
- **Testing Tools:** Use curl, Postman for REST endpoints, and a WebSocket tester (or your frontend) for chat/websocket endpoints.
- **Deployment:** Replace `127.0.0.1:8000` with your actual backend URL in production.