Video API Integration: Django + React (Beginner Friendly)

This guide explains how your Django backend exposes videos and how your React app fetches and displays them on CourseVideoPage. It also shows how to add more fields (like description and tutor) and render them.

What You Built (Backend)

- **Model:** A `Video` with `title`, `url`, and `added`.
- **Serializer:** Converts `Video` instances to JSON (`id`, `title`, `url`, `added`).
- **Viewset:** A DRF `ModelViewSet` that supports list/detail and CRUD.
- **URLs:** A DRF router registered under `/api/video/`.

Files involved:

- Backend/video/models.py
- Backend/video/serializers.py
- Backend/video/views.py
- Backend/video/urls.py
- Backend/main/urls.py

The Endpoint

- List all videos: `GET http://localhost:8000/api/video/videos/`
- Single video: `GET http://localhost:8000/api/video/videos/:id/`
- Writes (POST/PUT/DELETE) require JWT; reads are open by default.

How React Fetches Videos

We use Axios in `frontend/src/api.js` with an environment-based base URL, and an API helper function:

```
// frontend/src/api.js
import axios from "axios";
import { ACCESS_TOKEN } from "./constants";

const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL,
});

api.interceptors.request.use((config) => {
  const token = localStorage.getItem(ACCESS_TOKEN);
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});

export async function fetchVideos() {
  const { data } = await api.get('/api/video/videos/');
  return data; // [{ id, title, url, added }, ...]
}
```

On `CourseVideoPage.jsx`, we load videos on mount, derive “lessons” from them, and embed YouTube when possible:

```
// frontend/src/pages/CourseVideoPage.jsx
import { useEffect, useState } from 'react';
import { fetchVideos } from '../api';

export default function CourseVideoPage() {
  const [videos, setVideos] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [currentLessonIndex, setCurrentLessonIndex] = useState(null);

  useEffect(() => {
    (async () => {
      try {
        const data = await fetchVideos();
        setVideos(data);
      } catch (e) {
        setError(e.message);
      } finally {
        setLoading(false);
      }
    })();
  }, []);

  const lessons = videos.map((v) => ({
    id: v.id,
    title: v.title,
    url: v.url,
    time: new Date(v.added).toLocaleString(),
  }));

  const toEmbedUrl = (url) => {
    try {
      const u = new URL(url);
      if (u.hostname.includes('youtube.com')) {
        const vid = u.searchParams.get('v');
        if (vid) return `https://www.youtube.com/embed/${vid}`;
      }
      if (u.hostname === 'youtu.be') {
        const vid = u.pathname.replace('/', '');
        if (vid) return `https://www.youtube.com/embed/${vid}`;
      }
    } catch {}
    return null;
  };
  // render: loading/error states, list of lessons, and iframe when selected
}
```

Two Ways to Configure the API Base

Pick one of the following patterns to avoid 404s:

- **Option A (Base = host root):**
  - `.env` → `VITE_API_URL=http://127.0.0.1:8000`
  - Helper path → `api.get('/api/video/videos/')`
- **Option B (Base = host + /api):**
  - `.env` → `VITE_API_URL=http://127.0.0.1:8000/api`
  - Helper path → `api.get('/video/videos/')`

Choose one style and keep it consistent across your helpers.

Add More Fields (description, tutor)

You want to show more info (e.g., description, course tutor) on the page. Here's a simple approach:

Backend Changes

1. **Update the model**

```
# Backend/video/models.py
class Video(models.Model):
    title = models.CharField(max_length=255)
    url = models.URLField()
    added = models.DateTimeField(auto_now_add=True)
    description = models.TextField(blank=True)        # new
    tutor = models.CharField(max_length=255, blank=True)  # new
```

2. **Create and apply migrations**

```
cd D:\Scopio\Backend
benv\Scripts\activate
python manage.py makemigrations
python manage.py migrate
```

3. **Expose fields via serializer**

```
# Backend/video/serializers.py
class VideoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Video
        fields = ['id', 'title', 'url', 'added', 'description', 'tutor']
```

That's it—your API will now return `description` and `tutor` alongside each video.

Frontend Changes

1. **Use the new fields**

```
// In CourseVideoPage.jsx, when deriving lessons:
const lessons = videos.map((v) => ({
  id: v.id,
  title: v.title,
  url: v.url,
  time: new Date(v.added).toLocaleString(),
  description: v.description,
  tutor: v.tutor,
}));
```

2. **Render description and tutor**

```
{currentLessonIndex !== null && (
  <div className="lesson-details">
    <h2>{lessons[currentLessonIndex].title}</h2>
    {lessons[currentLessonIndex].tutor && (
      <p className="lesson-tutor">Tutor: {lessons[currentLessonIndex].tutor}
</p>
    )}
    {lessons[currentLessonIndex].description && (
      <p className="lesson-description">
{lessons[currentLessonIndex].description}</p>
    )}
  </div>
)}
```

If you need richer tutor data (avatar, bio, social links), consider a separate `Tutor` model and a foreign key from `Video` to `Tutor`, then nest/expand the serializer accordingly.

Embedding Other Providers

- **YouTube:** handled by `toEmbedUrl()` above.
- **Vimeo (basic example):**

```
if (u.hostname.includes('vimeo.com')) {
  const vid = u.pathname.split('/').filter(Boolean).pop();
  if (vid) return `https://player.vimeo.com/video/${vid}`;
}
```

Add other providers similarly.

Troubleshooting

- **404 Not Found:**
  - Ensure the helper path matches your base URL style (see Two Ways to Configure above).
  - Endpoint should be `/api/video/videos/` (or `/video/videos/` if base includes `/api`).
- **Server Not Running:**

  ```
  cd D:\Scopio\Backend
  benv\Scripts\activate
  python manage.py runserver
  ```

- **Database Connection Issues:**
  - Make sure Postgres is running (if you're not using SQLite locally).
  - If needed, temporarily switch to SQLite in `settings.py` for local dev.

Quick Test Commands

- **Check API directly (PowerShell):**

  ```
  $url = "http://localhost:8000/api/video/videos/"
  (Invoke-WebRequest -Uri $url -UseBasicParsing).Content
  ```

- **Build frontend:**

  ```
  cd D:\Scopio\frontend
  npm run build
  ```

Reference Map

- Backend endpoints: `/api/video/videos/`
- Frontend base: `.env` → `VITE_API_URL`
- Fetch helper: `frontend/src/api.js`
- Page integration: `frontend/src/pages/CourseVideoPage.jsx`

If you want, I can scaffold a small JWT-protected “Add Video” form in React and a simple admin UI to create/update videos from the browser.