Got you. Let's build a **plain React JS frontend** from scratch for your NestJS Movies & Reviews backend.

✅ All APIs integrated
✅ No CSS at all
✅ Uses `fetch` (no extra libraries)
✅ Works with backend at `http://localhost:3000`

---

1. Create React App (Vite + React)

```bash
# 1. Create project
npm create vite@latest movie-frontend -- --template react

cd movie-frontend

# 2. Install deps
npm install

# 3. Start dev server
npm run dev
```

Your backend should already be running on **http://localhost:3000**.

---

2. (Backend) Enable CORS (important)

In your NestJS `main.ts` (backend), make sure you have:

```ts
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';
import { ValidationPipe } from '@nestjs/common';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  app.enableCors({
    origin: 'http://localhost:5173', // Vite frontend
```

```
  });

  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      forbidNonWhitelisted: true,
      transform: true,
    }),
  );

  await app.listen(3000);
}
bootstrap();
```

---

3. Frontend Code

We'll only touch **two files**:

* `src/main.jsx`
* `src/App.jsx`

No CSS files, no imports of `.css`.

3.1 `src/main.jsx`

Replace everything with:

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App.jsx';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

3.2 `src/App.jsx`

Replace everything with this (full code, all APIs wired):

```jsx
import React, { useEffect, useState } from 'react';

const API_BASE = 'http://localhost:3000';

function App() {
  // Movies
  const [movies, setMovies] = useState([]);
  const [loadingMovies, setLoadingMovies] = useState(false);
  const [moviesError, setMoviesError] = useState('');

  // Selected movie details
  const [selectedMovie, setSelectedMovie] = useState(null);
  const [reviews, setReviews] = useState([]);
  const [loadingReviews, setLoadingReviews] = useState(false);
  const [reviewsError, setReviewsError] = useState('');

  // Create movie form
  const [newMovieTitle, setNewMovieTitle] = useState('');
  const [newMovieGenre, setNewMovieGenre] = useState('');
  const [newMovieReleaseYear, setNewMovieReleaseYear] = useState('');

  // Filters & sorting
  const [filterGenre, setFilterGenre] = useState('');
  const [filterReleaseYear, setFilterReleaseYear] = useState('');
  const [sortBy, setSortBy] = useState('');
  const [sortOrder, setSortOrder] = useState('ASC');

  // Create review form
  const [reviewReviewer, setReviewReviewer] = useState('');
  const [reviewRating, setReviewRating] = useState('');
  const [reviewComment, setReviewComment] = useState('');

  // ---------- API functions ----------

  async function fetchMovies() {
    try {
      setLoadingMovies(true);
```

```
    setMoviesError('');

    const params = new URLSearchParams();

    if (filterGenre) params.append('genre', filterGenre);
    if (filterReleaseYear) params.append('releaseYear', filterReleaseYear);
    if (sortBy) params.append('sortBy', sortBy);
    if (sortOrder) params.append('sortOrder', sortOrder);

    const url = `${API_BASE}/movies${params.toString() ? `?${params.toString()}` : ''}`;

    const res = await fetch(url);
    if (!res.ok) {
      throw new Error('Failed to fetch movies');
    }
    const data = await res.json();
    setMovies(data);
  } catch (err) {
    setMoviesError(err.message || 'Error fetching movies');
  } finally {
    setLoadingMovies(false);
  }
}

async function fetchMovieById(id) {
  try {
    const res = await fetch(`${API_BASE}/movies/${id}`);
    if (!res.ok) {
      throw new Error('Failed to fetch movie');
    }
    const data = await res.json();
    setSelectedMovie(data);
  } catch (err) {
    console.error(err);
  }
}

async function fetchReviewsByMovieId(movieId) {
  try {
    setLoadingReviews(true);
    setReviewsError('');
    const res = await fetch(`${API_BASE}/reviews/${movieId}`);
    if (!res.ok) {
      throw new Error('Failed to fetch reviews');
```

```
    }
    const data = await res.json();
    setReviews(data);
  } catch (err) {
    setReviewsError(err.message || 'Error fetching reviews');
  } finally {
    setLoadingReviews(false);
  }
}

async function createMovie(event) {
  event.preventDefault();

  try {
    setMoviesError('');

    const body = {
      title: newMovieTitle,
    };

    if (newMovieGenre.trim() !== '') {
      body.genre = newMovieGenre;
    }

    if (newMovieReleaseYear.trim() !== '') {
      body.releaseYear = Number(newMovieReleaseYear);
    }

    const res = await fetch(`${API_BASE}/movies`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(body),
    });

    if (!res.ok) {
      const errorText = await res.text();
      throw new Error(errorText || 'Failed to create movie');
    }

    const created = await res.json();
    console.log('Movie created:', created);
```

```
    // Reset form
    setNewMovieTitle('');
    setNewMovieGenre('');
    setNewMovieReleaseYear('');

    // Refresh movie list
    await fetchMovies();
  } catch (err) {
    setMoviesError(err.message || 'Error creating movie');
  }
}

async function createReview(event) {
  event.preventDefault();

  if (!selectedMovie) {
    alert('Select a movie first');
    return;
  }

  try {
    setReviewsError('');

    const body = {
      movieId: selectedMovie.id,
      reviewer: reviewReviewer,
      rating: Number(reviewRating),
    };

    if (reviewComment.trim() !== '') {
      body.comment = reviewComment;
    }

    const res = await fetch(`${API_BASE}/reviews`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(body),
    });

    if (!res.ok) {
      const errorText = await res.text();
      throw new Error(errorText || 'Failed to create review');
```

```
    }

    const created = await res.json();
    console.log('Review created:', created);

    // Reset form
    setReviewReviewer('');
    setReviewRating('');
    setReviewComment('');

    // Refresh reviews
    await fetchReviewsByMovieId(selectedMovie.id);
    // Optional: refresh movies to reflect rating sorting if needed
    await fetchMovies();
  } catch (err) {
    setReviewsError(err.message || 'Error creating review');
  }
}

// ---------- Effects ----------

useEffect(() => {
  fetchMovies();
}, []); // initial load

// ---------- Handlers ----------

function handleApplyFilters(event) {
  event.preventDefault();
  fetchMovies();
}

function handleSelectMovie(movie) {
  setSelectedMovie(movie);
  fetchMovieById(movie.id); // to use /movies/:id API
  fetchReviewsByMovieId(movie.id); // to use /reviews/:movieId API
}

// ---------- Render ----------

return (
  <div>
    <h1>Movies & Reviews</h1>
```

```
{/* Create Movie */}
<section>
  <h2>Add New Movie</h2>
  <form onSubmit={createMovie}>
    <div>
      <label>
        Title:{' '}
        <input
          type="text"
          value={newMovieTitle}
          onChange={(e) => setNewMovieTitle(e.target.value)}
          required
        />
      </label>
    </div>

    <div>
      <label>
        Genre:{' '}
        <input
          type="text"
          value={newMovieGenre}
          onChange={(e) => setNewMovieGenre(e.target.value)}
          placeholder="Action, Drama, etc."
        />
      </label>
    </div>

    <div>
      <label>
        Release Year:{' '}
        <input
          type="number"
          value={newMovieReleaseYear}
          onChange={(e) => setNewMovieReleaseYear(e.target.value)}
          placeholder="e.g. 2023"
        />
      </label>
    </div>

    <button type="submit">Create Movie</button>
  </form>
  {moviesError && <p>Error: {moviesError}</p>}
</section>
```

```
{/* Filters & Sorting */}
<section>
  <h2>Filter & Sort Movies</h2>
  <form onSubmit={handleApplyFilters}>
    <div>
      <label>
        Genre:{' '}
        <input
          type="text"
          value={filterGenre}
          onChange={(e) => setFilterGenre(e.target.value)}
          placeholder="Action"
        />
      </label>
    </div>

    <div>
      <label>
        Release Year:{' '}
        <input
          type="number"
          value={filterReleaseYear}
          onChange={(e) => setFilterReleaseYear(e.target.value)}
          placeholder="2023"
        />
      </label>
    </div>

    <div>
      <label>
        Sort By:{' '}
        <select
          value={sortBy}
          onChange={(e) => setSortBy(e.target.value)}
        >
          <option value="">Default (createdAt)</option>
          <option value="releaseYear">Release Year</option>
          <option value="rating">Average Rating</option>
        </select>
      </label>
    </div>

    <div>
```

```
      <label>
        Sort Order:{' '}
        <select
          value={sortOrder}
          onChange={(e) => setSortOrder(e.target.value)}
        >
          <option value="ASC">ASC</option>
          <option value="DESC">DESC</option>
        </select>
      </label>
    </div>

    <button type="submit">Apply Filters</button>
  </form>
</section>

{/* Movies List */}
<section>
  <h2>Movies List</h2>
  {loadingMovies && <p>Loading movies...</p>}
  {!loadingMovies && movies.length === 0 && <p>No movies found.</p>}
  <ul>
    {movies.map((movie) => (
      <li key={movie.id}>
        <div>
          <strong>{movie.title}</strong>
        </div>
        <div>Genre: {movie.genre || 'N/A'}</div>
        <div>Release Year: {movie.releaseYear || 'N/A'}</div>
        <button onClick={() => handleSelectMovie(movie)}>
          View Details & Reviews
        </button>
      </li>
    ))}
  </ul>
</section>

{/* Selected Movie Details & Reviews */}
{selectedMovie && (
  <section>
    <h2>Movie Details</h2>
    <p>ID: {selectedMovie.id}</p>
    <p>Title: {selectedMovie.title}</p>
    <p>Genre: {selectedMovie.genre || 'N/A'}</p>
```

```
<p>Release Year: {selectedMovie.releaseYear || 'N/A'}</p>

{/* Add Review */}
<h3>Add Review for this Movie</h3>
<form onSubmit={createReview}>
  <div>
    <label>
      Reviewer:{' '}
      <input
        type="text"
        value={reviewReviewer}
        onChange={(e) => setReviewReviewer(e.target.value)}
        required
      />
    </label>
  </div>

  <div>
    <label>
      Rating (1-5):{' '}
      <input
        type="number"
        min="1"
        max="5"
        value={reviewRating}
        onChange={(e) => setReviewRating(e.target.value)}
        required
      />
    </label>
  </div>

  <div>
    <label>
      Comment:{' '}
      <textarea
        value={reviewComment}
        onChange={(e) => setReviewComment(e.target.value)}
        placeholder="Optional comment"
      />
    </label>
  </div>

  <button type="submit">Submit Review</button>
</form>
```

```
        {reviewsError && <p>Error: {reviewsError}</p>}

        {/* Reviews List */}
        <h3>Reviews</h3>
        {loadingReviews && <p>Loading reviews...</p>}
        {!loadingReviews && reviews.length === 0 && (
          <p>No reviews for this movie yet.</p>
        )}
        <ul>
          {reviews.map((review) => (
            <li key={review.id}>
              <div>
                <strong>{review.reviewer}</strong> — Rating: {review.rating}
              </div>
              <div>Comment: {review.comment || 'No comment'}</div>
              <div>Created At: {review.createdAt}</div>
            </li>
          ))}
        </ul>
      </section>
    )}
  </div>
 );
}

export default App;
```

---

4. What's Integrated?

This React app calls **all your backend APIs**:

* `POST /movies` → Add movie (form at top)
* `GET /movies` → List movies with filters & sorting
* `GET /movies/:id` → Fetch single movie when "View Details & Reviews" clicked
* `POST /reviews` → Add review for selected movie
* `GET /reviews/:movieId` → Fetch reviews for selected movie

No CSS, only plain HTML tags and JS logic.
