Backend Node.js + Express + Mongoose

1. Setup and Connect to MongoDB

```javascript
// server.js
const express = require('express');
const mongoose = require('mongoose');
const app = express();

app.use(express.json());

// Connect to MongoDB
mongoose.connect('mongodb://localhost:27017/cinecritique', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', () => {
  console.log('Connected to MongoDB');
});

// Start server
app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

2. Define Schemas & Models

```javascript
// models/User.js
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true }
});

module.exports = mongoose.model('User', userSchema);
```

```javascript
// models/Movie.js
const mongoose = require('mongoose');

const movieSchema = new mongoose.Schema({
  title: { type: String, required: true },
  genre: { type: [String], required: true },
  releaseYear: Number,
  averageRating: { type: Number, default: 0 }
}, { timestamps: true });

module.exports = mongoose.model('Movie', movieSchema);
```

```javascript
// models/Review.js
const mongoose = require('mongoose');

const reviewSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  movieId: { type: mongoose.Schema.Types.ObjectId, ref: 'Movie', required: true },
  rating: { type: Number, required: true, min:1, max:5 },
  reviewText: String
}, { timestamps: true });

module.exports = mongoose.model('Review', reviewSchema);
```

3. User Registration API

```javascript
// routes/auth.js
const express = require('express');
const bcrypt = require('bcrypt');
const User = require('../models/User');
const router = express.Router();

router.post('/register', async (req, res) => {
  try {
    const { username, email, password } = req.body;
    const userExists = await User.findOne({ email });
    if (userExists) return res.status(409).json({ message: 'Email already registered' });

    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new User({ username, email, password: hashedPassword });
    await user.save();
    res.status(201).json({ message: 'User registered successfully' });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

4. Submit Review API with Movie Average Rating Update

```javascript
// routes/reviews.js
const express = require('express');
const Review = require('../models/Review');
const Movie = require('../models/Movie');
const router = express.Router();

router.post('/movies/:movieId/reviews', async (req, res) => {
  try {
    const { movieId } = req.params;
    const { userId, rating, reviewText } = req.body; // Assuming userId is passed in body for simplicity

    const review = new Review({ userId, movieId, rating, reviewText });
    await review.save();

    const reviews = await Review.find({ movieId });
    const avgRating = reviews.reduce((acc, r) => acc + r.rating, 0) / reviews.length;
    await Movie.findByIdAndUpdate(movieId, { averageRating: avgRating });

    res.status(201).json({ message: 'Review submitted successfully', review });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

module.exports = router;
```

5. Sample Running Server Code with Routes

```javascript
// server.js continued
const authRoutes = require('./routes/auth');
const reviewRoutes = require('./routes/reviews');

app.use('/api/auth', authRoutes);
app.use('/api', reviewRoutes);

// Example endpoint to get all movies
const Movie = require('./models/Movie');
app.get('/api/movies', async (req, res) => {
  const movies = await Movie.find();
  res.json(movies);
});
```

Frontend React Components

1. MovieList Component (Output Movie Titles and Ratings)

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

function MovieList() {
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    fetch('/api/movies')
      .then(res => res.json())
      .then(setMovies)
      .catch(console.error);
  }, []);

  return (
    <div>
      <h2>Movie List</h2>
      <ul>
        {movies.map(movie => (
          <li key={movie._id}>
            {movie.title} - Rating: {movie.averageRating ? movie.averageRating.toFixed(1) : 'No ratings yet'}
          </li>
        ))}
      </ul>
    </div>
  );
}

export default MovieList;
```

2. ReviewForm Component

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

function ReviewForm({ movieId, userId, onReviewSubmitted }) {
  const [rating, setRating] = useState(1);
  const [reviewText, setReviewText] = useState('');

  async function handleSubmit(e) {
    e.preventDefault();

    try {
      const resp = await fetch(`/api/movies/${movieId}/reviews`, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ userId, rating, reviewText }),
      });
      if (resp.ok) {
        onReviewSubmitted();
        setRating(1);
        setReviewText('');
      } else {
        alert('Review submission failed');
      }
    } catch {
      alert('Error submitting review');
    }
  }

  return (
    <form onSubmit={handleSubmit}>
      <label>
        Rating (1-5):
        <input type="number" min="1" max="5" value={rating} onChange={e =>
setRating(Number(e.target.value))} required />
      </label>
      <br/>
      <label>
        Review:
        <textarea value={reviewText} onChange={e => setReviewText(e.target.value)} required />
      </label>
      <br/>
      <button type="submit">Submit Review</button>
```

Sample Outputs

Backend Console Output

```
Connected to MongoDB
Server running on port 3000
```

Postman / API Request Examples

- POST /api/auth/register

Request JSON:

```json
{
  "username": "john",
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```json
{
  "message": "User registered successfully"
}
```

- POST /api/movies/605c72f7f1d2a913a04f9b88/reviews

Request JSON:

```json
{
  "userId": "605c72a5f1d2a913a04f9b87",
  "rating": 5,
  "reviewText": "Awesome movie!"
}
```

Response:

```json
{
  "message": "Review submitted successfully",
  "review": {
    "_id": "605c731bf1d2a913a04f9b89",
    "userId": "605c72a5f1d2a913a04f9b87",
    "movieId": "605c72f7f1d2a913a04f9b88",
    "rating": 5,
    "reviewText": "Awesome movie!",
    "createdAt": "2025-11-22T00:00:00.000Z"
  }
}
```

React UI Output

- Movie List:
  - Inception - Rating: 4.8
  - Interstellar - Rating: 4.7
  - No ratings yet (for movies without reviews)
- Review Form:
  - A simple form to add rating (1-5) and text review and submit.

This example covers end-to-end flow with backend connection, data models, APIs, and basic React UI with sample output expectation.