Java script:

```
// server.js - Backend API
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const morgan = require('morgan');
require('dotenv').config();

const app = express();

// Middleware
app.use(cors());
app.use(express.json());
app.use(morgan('dev'));

// Database connection
mongoose.connect(process.env.MONGODB_URI, {
  useNewUrlParser: true,
  useUnifiedTopology: true
})
.then(() => console.log('Connected to MongoDB'))
.catch(err => console.error('MongoDB connection error:', err));

// Movie Schema
const movieSchema = new mongoose.Schema({
  title: { type: String, required: true },
  genre: { type: [String], required: true },
  year: { type: Number, required: true },
  rating: { type: Number, min: 0, max: 10 },
  description: { type: String, required: true },
  imageUrl: { type: String, default: '' },
  createdAt: { type: Date, default: Date.now }
});

const Movie = mongoose.model('Movie', movieSchema);

// API Routes
app.get('/api/movies', async (req, res) => {
  try {
    const movies = await Movie.find();
    res.json(movies);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

app.get('/api/movies/search', async (req, res) => {
  try {
    const query = req.query.q || '';
    const movies = await Movie.find({
      $or: [
        { title: { $regex: query, $options: 'i' } },
        { description: { $regex: query, $options: 'i' } }
      ]
    });
    res.json(movies);
```

```
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

app.get('/api/movies/filter', async (req, res) => {
  try {
    const genre = req.query.genre;
    const query = genre === 'all' ? {} : { genre: genre };
    const movies = await Movie.find(query);
    res.json(movies);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

app.post('/api/movies', async (req, res) => {
  try {
    const movie = new Movie(req.body);
    const newMovie = await movie.save();
    res.status(201).json(newMovie);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Error handling middleware
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ message: 'Something went wrong!' });
});

// Start server
const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```