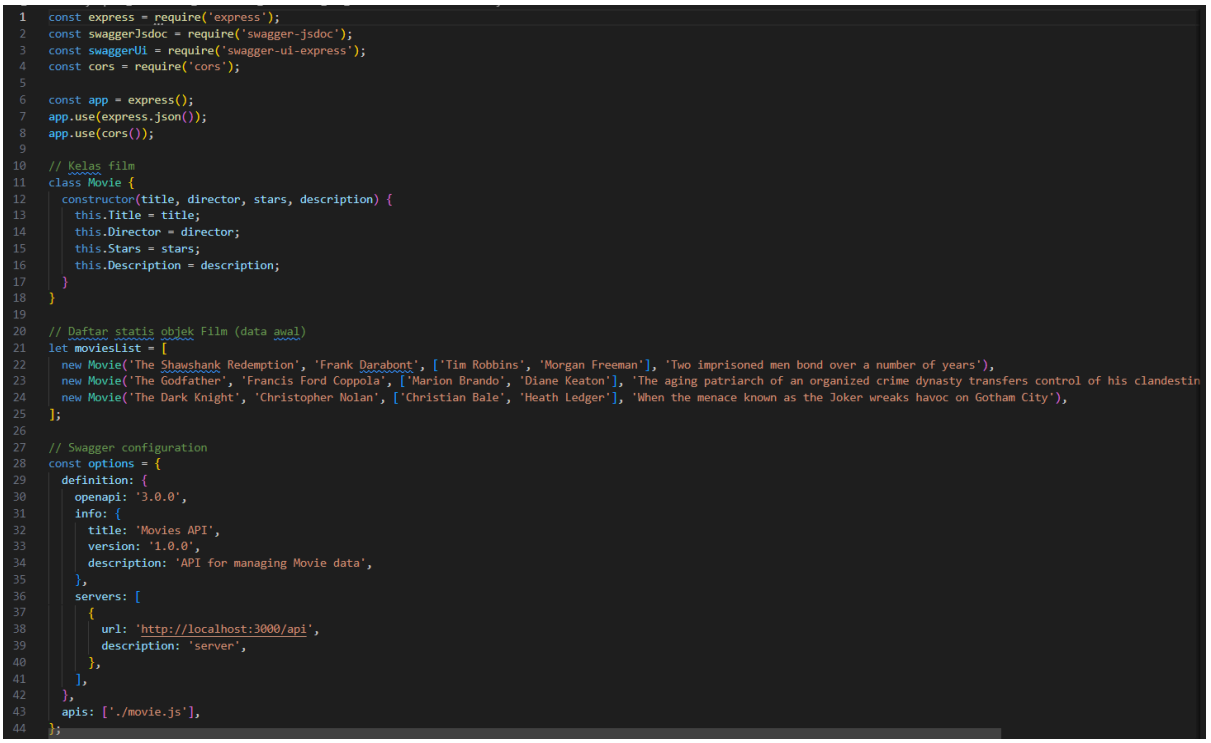
Jurnal Modul 09

2311104008
Viona Aziz Syahputri

Link Gitbut
https://github.com/viona123/KPL_Viona-Aziz-Syahputri_2311104008_SE07-01/tree/main/09_API/Jurnal_GUI_2311104008

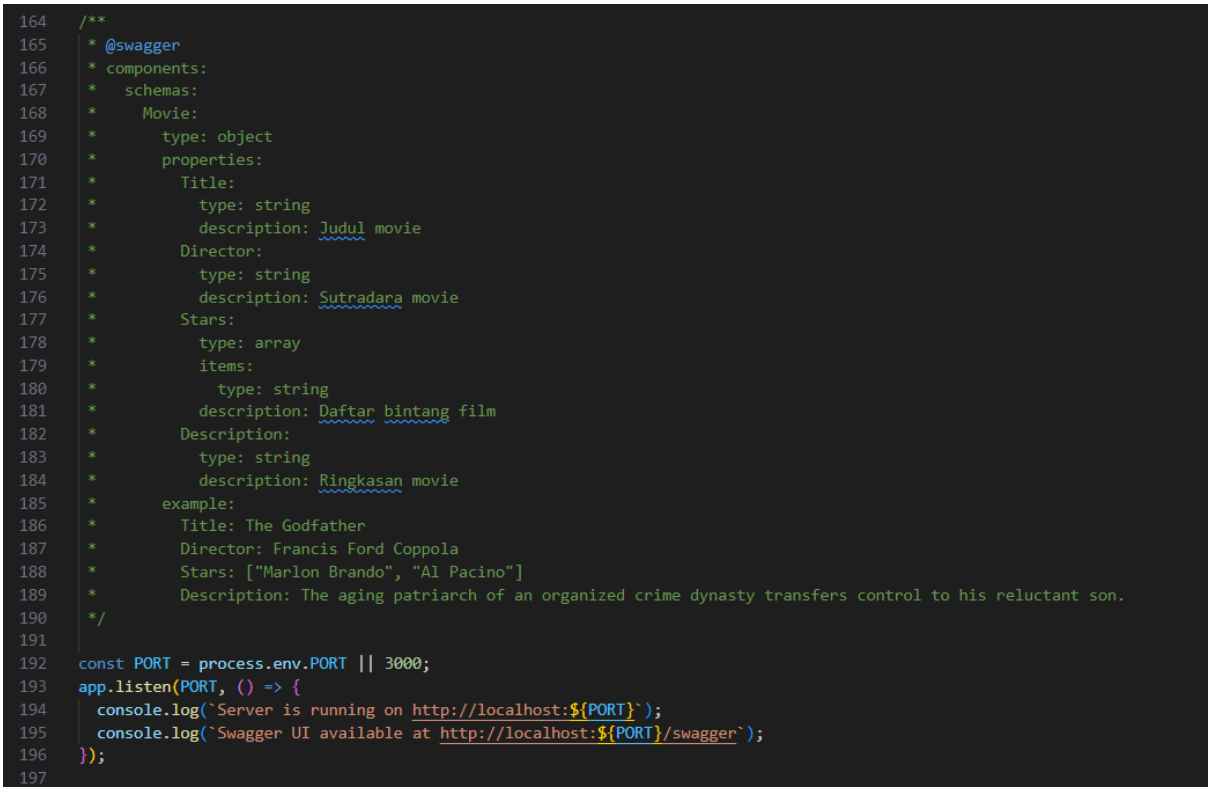


```
const express = require('express');
const swaggerJsdoc = require('swagger-jsdoc');
const swaggerUi = require('swagger-ui-express');
const cors = require('cors');

const app = express();
app.use(express.json());
app.use(cors());

// Kelas film
class Movie {
  constructor(title, director, stars, description) {
    this.Title = title;
    this.Director = director;
    this.Stars = stars;
    this.Description = description;
  }
}

// Daftar statis objek Film (data awal)
let moviesList = [
  new Movie('The Shawshank Redemption', 'Frank Darabont', ['Tim Robbins', 'Morgan Freeman'], 'Two imprisoned men bond over a number of years'),
  new Movie('The Godfather', 'Francis Ford Coppola', ['Marion Brando', 'Diane Keaton'], 'The aging patriarch of an organized crime dynasty transfers control of his clandestin
  new Movie('The Dark Knight', 'Christopher Nolan', ['Christian Bale', 'Heath Ledger'], 'When the menace known as the Joker wreaks havoc on Gotham City'),
];

// Swagger configuration
const options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'Movies API',
      version: '1.0.0',
      description: 'API for managing Movie data',
    },
    servers: [
      {
        url: 'http://localhost:3000/api',
        description: 'server',
      },
    ],
  },
  apis: ['./movie.js'],
};
```

```
const swaggerSpec = swaggerJsdoc(options);
app.use('/swagger', swaggerUi.serve, swaggerUi.setup(swaggerSpec));

/**
 * @swagger
 * tags:
 *   name: Movies
 *   description: Movies management API
 */

/**
 * @swagger
 * /Movies:
 *   get:
 *     summary: Mendapatkan daftar semua film
 *     tags: [Movies]
 *     responses:
 *       200:
 *         description: Sukses mendapatkan data film
 *         content:
 *           application/json:
 *             schema:
 *               type: array
 *               items:
 *                 $ref: '#/components/schemas/Movie'
 */
app.get('/api/Movies', (req, res) => {
  res.json(moviesList);
});

/**
 * @swagger
 * /Movies/{id}:
 *   get:
 *     summary: Mendapatkan data film berdasarkan id
 *     tags: [Movies]
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
 *           type: integer
 *         required: true
 *         description: Id Movie (dimulai dari 0)
```

```
 *     responses:
 *       200:
 *         description: Data Movie
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Movie'
 *       404:
 *         description: Movie tidak ditemukan
 */
app.get('/api/Movies/:id', (req, res) => {
  const id = parseInt(req.params.id);
  if (id >= 0 && id < moviesList.length) {
    res.json(moviesList[id]);
  } else {
    res.status(404).json({ message: 'Movie not found' });
  }
});

/**
 * @swagger
 * /Movies:
 *   post:
 *     summary: Menambahkan film baru
 *     tags: [Movies]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Movie'
 *     responses:
 *       201:
 *         description: Movie berhasil ditambahkan
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Movie'
 */
app.post('/api/Movies', (req, res) => {
  const { Title, Director, Stars, Description } = req.body;
  const newMovie = new Movie(Title, Director, Stars, Description);
  moviesList.push(newMovie);
```

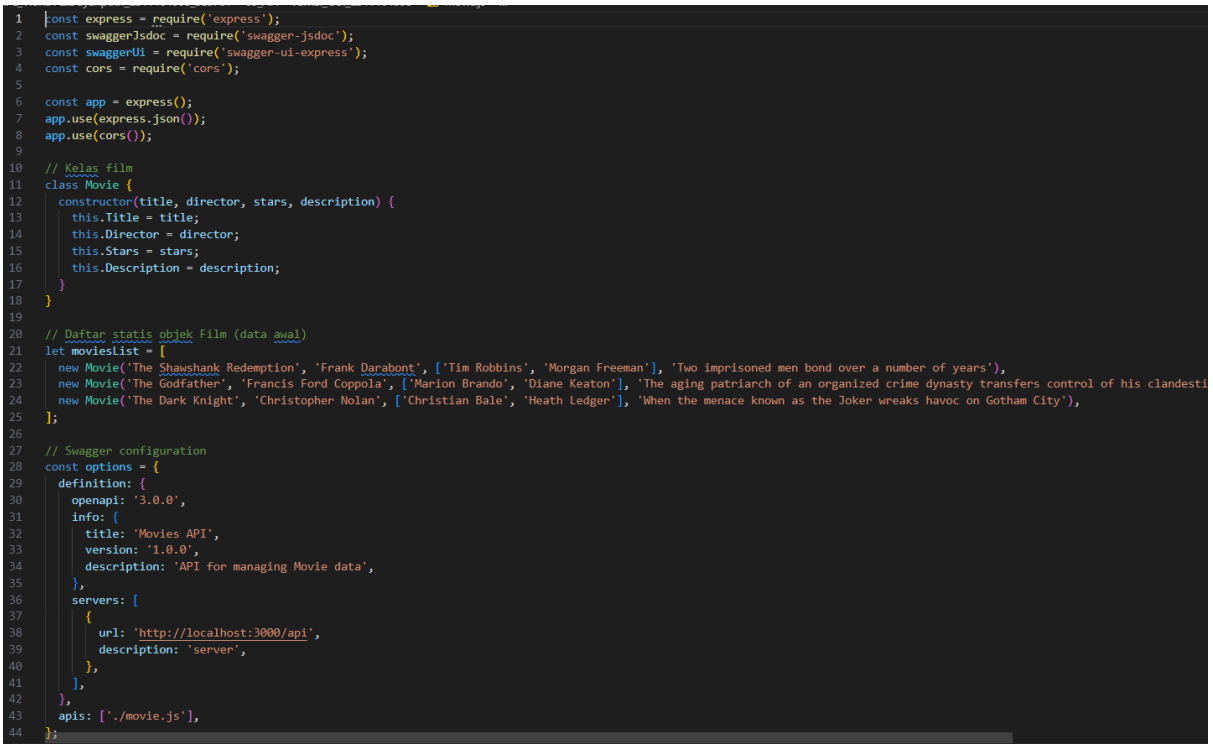
```
  res.status(201).json(newMovie);
});

/**
 * @swagger
 * /Movies/{id}:
 *   delete:
 *     summary: Menghapus film berdasarkan id
 *     tags: [Movies]
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
 *           type: integer
 *         required: true
 *         description: Id Movie (dimulai dari 0)
 *     responses:
 *       200:
 *         description: Movie berhasil dihapus
 *       404:
 *         description: Movie tidak ditemukan
 */
app.delete('/api/Movies/:id', (req, res) => {
  const id = parseInt(req.params.id);
  if (id >= 0 && id < moviesList.length) {
    moviesList.splice(id, 1);
    res.json({ message: 'Movie deleted successfully' });
  } else {
    res.status(404).json({ message: 'Movie not found' });
  }
});

/**
 * @swagger
 * components:
 *   schemas:
 *     Movie:
 *       type: object
 *       properties:
 *         Title:
 *           type: string
 *           description: Judul movie
 *         Director:
```



```
/**
 * @swagger
 * components:
 *   schemas:
 *     Movie:
 *       type: object
 *       properties:
 *         Title:
 *           type: string
 *           description: Judul movie
 *         Director:
 *           type: string
 *           description: Sutradara movie
 *         Stars:
 *           type: array
 *           items:
 *             type: string
 *           description: Daftar bintang film
 *         Description:
 *           type: string
 *           description: Ringkasan movie
 *       example:
 *         Title: The Godfather
 *         Director: Francis Ford Coppola
 *         Stars: ["Marlon Brando", "Al Pacino"]
 *         Description: The aging patriarch of an organized crime dynasty transfers control to his reluctant son.
 */

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
  console.log(`Swagger UI available at http://localhost:${PORT}/swagger`);
});

```

Kode ini adalah program Node.js sederhana yang bikin API buat ngatur data film pakai framework Express.js. Di bagian awal, dipakai beberapa library penting: express buat bikin server dan nangani request, cors supaya server bisa diakses dari mana aja (misalnya dari tampilan website), dan swagger-jsdoc sama swagger-ui-express buat bikin dokumentasi otomatis dari API-nya. Data filmnya disimpan sementara di array, isinya beberapa film lengkap sama judul, sutradara, aktor, dan deskripsinya. Ada juga class Movie biar setiap data film punya format yang sama dan rapi.

API ini punya fitur-fitur dasar, kayak ngeliat semua film (GET /api/Movies), ngeliat detail film berdasarkan urutan data (GET /api/Movies/:id), nambah film baru (POST /api/Movies), dan hapus film (DELETE /api/Movies/:id). Semua fitur itu udah otomatis dijelasin lewat Swagger, jadi kalau buka http://localhost:3000/swagger, kita bisa lihat dokumentasinya langsung di browser dan bisa coba-coba juga tanpa harus bikin program tambahan. Intinya, program ini cocok banget buat belajar bikin REST API yang udah lengkap dan ada dokumentasinya.

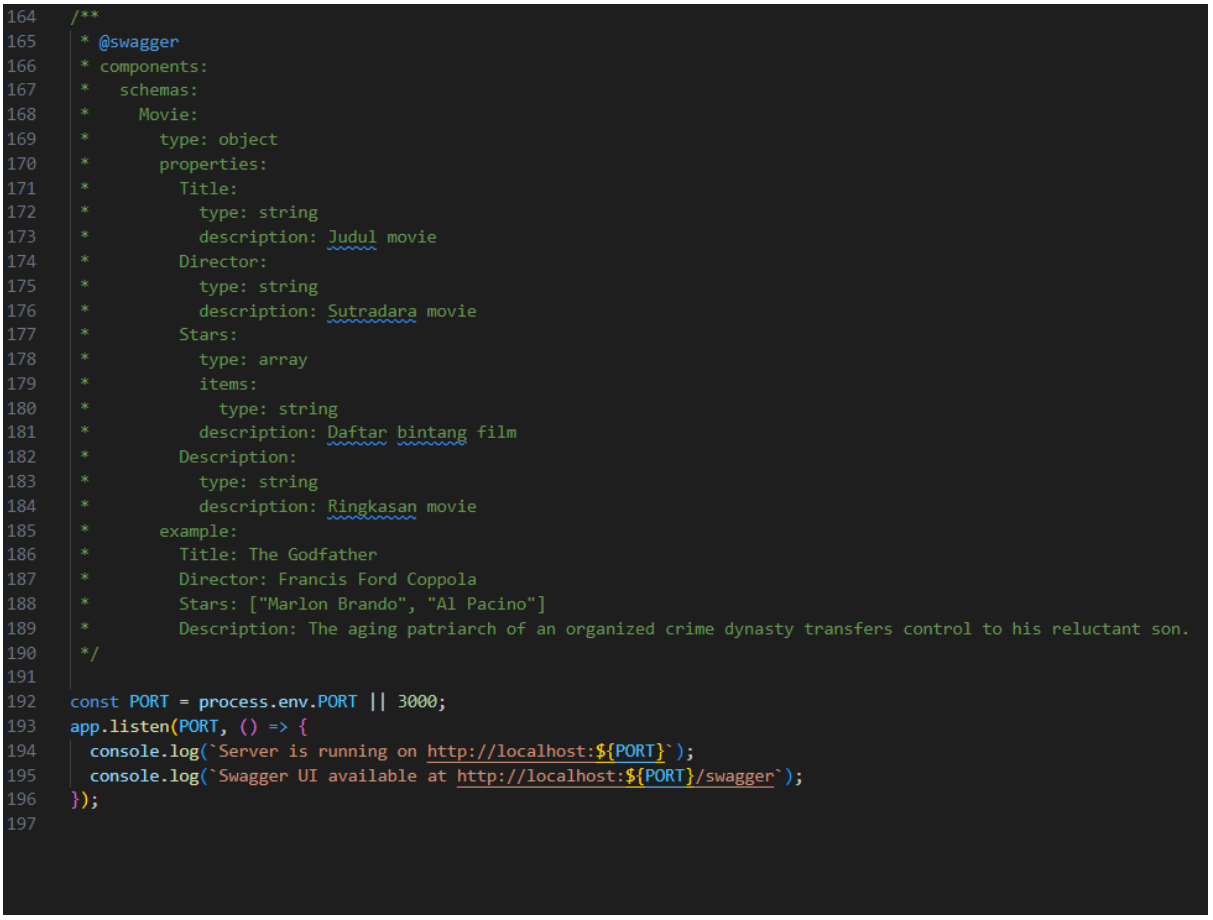


```
const express = require('express');
const swaggerJsdoc = require('swagger-jsdoc');
const swaggerUi = require('swagger-ui-express');
const cors = require('cors');

const app = express();
app.use(express.json());
app.use(cors());

// Kelas film
class Movie {
  constructor(title, director, stars, description) {
    this.Title = title;
    this.Director = director;
    this.Stars = stars;
    this.Description = description;
  }
}

// Daftar statis objek Film (data awal)
let moviesList = [
  new Movie('The Shawshank Redemption', 'Frank Darabont', ['Tim Robbins', 'Morgan Freeman'], 'Two imprisoned men bond over a number of years'),
  new Movie('The Godfather', 'Francis Ford Coppola', ['Marion Brando', 'Diane Keaton'], 'The aging patriarch of an organized crime dynasty transfers control of his clandesti
  new Movie('The Dark Knight', 'Christopher Nolan', ['Christian Bale', 'Heath Ledger'], 'When the menace known as the Joker wreaks havoc on Gotham City'),
];

// Swagger configuration
const options = {
  definition: {
    openapi: '3.0.0',
    info: {
      title: 'Movies API',
      version: '1.0.0',
      description: 'API for managing Movie data',
    },
    servers: [
      {
        url: 'http://localhost:3000/api',
        description: 'server',
      },
    ],
  },
  apis: ['./movie.js'],
};
```

```
const swaggerSpec = swaggerJsdoc(options);
app.use('/swagger', swaggerUi.serve, swaggerUi.setup(swaggerSpec));

/**
 * @swagger
 * tags:
 *   name: Movies
 *   description: Movies management API
 */

/**
 * @swagger
 * /Movies:
 *   get:
 *     summary: Mendapatkan daftar semua film
 *     tags: [Movies]
 *     responses:
 *       200:
 *         description: Sukses mendapatkan data film
 *         content:
 *           application/json:
 *             schema:
 *               type: array
 *               items:
 *                 $ref: '#/components/schemas/Movie'
 */
app.get('/api/Movies', (req, res) => {
  res.json(moviesList);
});

/**
 * @swagger
 * /Movies/{id}:
 *   get:
 *     summary: Mendapatkan data film berdasarkan id
 *     tags: [Movies]
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
 *           type: integer
 *         required: true
 *         description: Id Movie (dimulai dari 0)
 *     responses:
```

```
 *       200:
 *         description: Data Movie
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Movie'
 *       404:
 *         description: Movie tidak ditemukan
 */
app.get('/api/Movies/:id', (req, res) => {
  const id = parseInt(req.params.id);
  if (id >= 0 && id < moviesList.length) {
    res.json(moviesList[id]);
  } else {
    res.status(404).json({ message: 'Movie not found' });
  }
});

/**
 * @swagger
 * /Movies:
 *   post:
 *     summary: Menambahkan film baru
 *     tags: [Movies]
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Movie'
 *     responses:
 *       201:
 *         description: Movie berhasil ditambahkan
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Movie'
 */
app.post('/api/Movies', (req, res) => {
  const { Title, Director, Stars, Description } = req.body;
  const newMovie = new Movie(Title, Director, Stars, Description);
  moviesList.push(newMovie);
  res.status(201).json(newMovie);
});
```

```
/**
 * @swagger
 * /Movies/{id}:
 *   delete:
 *     summary: Menghapus film berdasarkan id
 *     tags: [Movies]
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
 *           type: integer
 *         required: true
 *         description: Id Movie (dimulai dari 0)
 *     responses:
 *       200:
 *         description: Movie berhasil dihapus
 *       404:
 *         description: Movie tidak ditemukan
 */
app.delete('/api/Movies/:id', (req, res) => {
  const id = parseInt(req.params.id);
  if (id >= 0 && id < moviesList.length) {
    moviesList.splice(id, 1);
    res.json({ message: 'Movie deleted successfully' });
  } else {
    res.status(404).json({ message: 'Movie not found' });
  }
});

/**
 * @swagger
 * components:
 *   schemas:
 *     Movie:
 *       type: object
 *       properties:
 *         Title:
 *           type: string
 *           description: Judul movie
 *         Director:
 *           type: string
 *           description: Sutradara movie
 *         Stars:
 *           type: array
```



```
/**
 * @swagger
 * components:
 *   schemas:
 *     Movie:
 *       type: object
 *       properties:
 *         Title:
 *           type: string
 *           description: Judul movie
 *         Director:
 *           type: string
 *           description: Sutradara movie
 *         Stars:
 *           type: array
 *           items:
 *             type: string
 *           description: Daftar bintang film
 *         Description:
 *           type: string
 *           description: Ringkasan movie
 *       example:
 *         Title: The Godfather
 *         Director: Francis Ford Coppola
 *         Stars: ["Marlon Brando", "Al Pacino"]
 *         Description: The aging patriarch of an organized crime dynasty transfers control to his reluctant son.
 */

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
  console.log(`Swagger UI available at http://localhost:${PORT}/swagger`);
});
```

Kode ini bikin semacam server sederhana pakai Node.js dan Express.js, yang fungsinya buat ngatur data film. Di awal, ada beberapa modul tambahan yang dipakai, kayak cors supaya server bisa diakses dari aplikasi lain (misalnya frontend), terus ada swagger-jsdoc dan swagger-ui-express buat ngebikin dokumentasi API otomatis. Filmnya disimpan sementara di sebuah array moviesList, jadi nggak pake database. Ada juga class Movie yang dipakai buat nyusun format data film biar rapi, isinya kayak judul film, sutradaranya, siapa aja pemainnya, dan deskripsi singkatnya.

Aplikasi ini punya beberapa fitur utama, kayak nampilin semua film lewat GET /api/Movies, ngeliat detail film tertentu lewat GET /api/Movies/:id, nambahin film baru lewat POST /api/Movies, dan hapus film pakai DELETE /api/Movies/:id. Semua fitur tadi bisa dicoba dan dilihat cara kerjanya lewat dokumentasi Swagger yang bisa diakses di http://localhost:3000/swagger. Jadi, intinya kode ini cocok banget buat latihan bikin API sederhana dengan fitur lengkap dan dokumentasi yang bisa langsung dicoba lewat browser. Mau nambah, hapus, atau lihat data film bisa langsung dari sana.