Movies.js

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

module.exports = Movie;
```

Penjelasan:

File ini berisi definisi kelas Movie yang menjadi blueprint/model struktur data film, sehingga setiap objek film yang dibuat atau diproses di aplikasi memiliki atribut Title, Director, Stars, dan Description secara konsisten.

moviesApp.js

```
const express = require("express");
const bodyParser = require("body-parser");
const fs = require("fs");
const path = require("path");
const Movie = require("./Movie");
const swaggerUi = require("swagger-ui-express");
const swaggerJsdoc = require("swagger-jsdoc");

const app = express();
const port = 3000; // You can change this if needed

// Helper to read and write JSON
const moviesFile = path.join(__dirname, "movies.json");

function readMovies() {
  const data = fs.readFileSync(moviesFile, "utf-8");
  return JSON.parse(data);
}

function writeMovies(movies) {
  fs.writeFileSync(moviesFile, JSON.stringify(movies, null, 2), "utf-8");
}

app.use(bodyParser.json());

// Swagger setup
const options = {
  definition: {
    openapi: "3.0.0",
    info: {
```

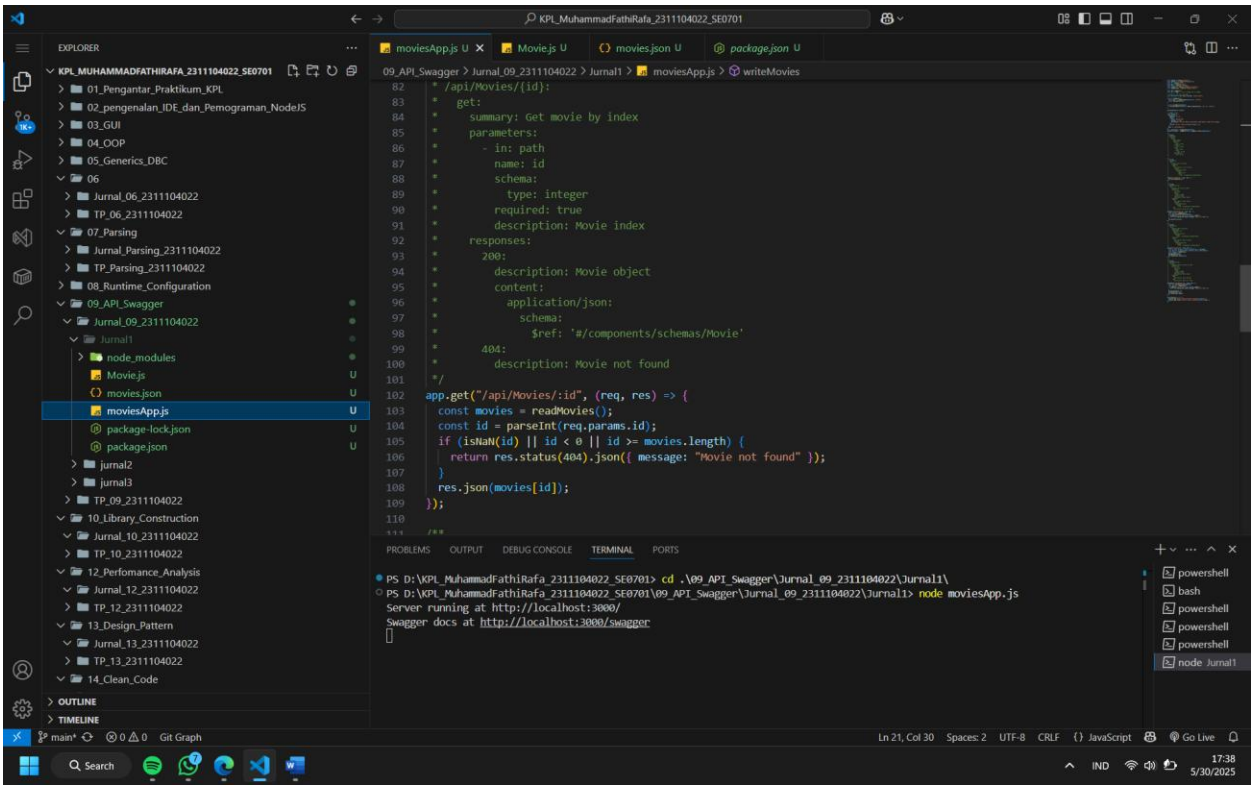
EMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
    info: {
      description: "API for Movies using Node.js and Express (JSON file storage)",
    },
    servers: [{ url: `http://localhost:${port}` }],
  },
  apis: ["./moviesApp.js"],
};
const swaggerSpec = swaggerJsdoc(options);
app.use("/swagger", swaggerUi.serve, swaggerUi.setup(swaggerSpec));

/**
 * @swagger
 * components:
 *   schemas:
 *     Movie:
 *       type: object
 *       properties:
 *         Title:
 *           type: string
 *         Director:
 *           type: string
 *         Stars:
 *           type: array
 *           items:
 *             type: string
 *         Description:
 *           type: string
```

```
/**
 * @swagger
 * /api/Movies:
 *   get:
 *     summary: Get all movies
 *     responses:
 *       200:
 *         description: List of movies
 *         content:
 *           application/json:
 *             schema:
 *               type: array
 *               items:
 *                 $ref: '#/components/schemas/Movie'
 */
app.get("/api/Movies", (req, res) => {
  res.json(readMovies());
});

/**
 * @swagger
 * /api/Movies/{id}:
 *   get:
 *     summary: Get movie by index
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
```


KPL_MuhammadFathiRafa_2311104022_SE0701
EXPLORER
KPL_MUHAMMADFATHIRAFA_2311104022_SE0701
01_Pengantar_Praktikum_KPL
02_pengenalan_IDE_dan_Pemograman_NodeJS
03_GUI
04_OOP
05_Generics_DBC
06
Jurnal_06_2311104022
TP_06_2311104022
07_Parsing
Jurnal_Parsing_2311104022
TP_Parsing_2311104022
08_Runtime_Configuration
09_API_Swagger
Jurnal_09_2311104022
Jurnal1
node_modules
Movie.js
movies.json
moviesApp.js
package-lock.json
package.json
jurnal2
jurnal3
TP_09_2311104022
10_Library_Construction
Jurnal_10_2311104022
TP_10_2311104022
12_Perfomance_Analysis
Jurnal_12_2311104022
TP_12_2311104022
13_Design_Pattern
Jurnal_13_2311104022
TP_13_2311104022
14_Clean_Code
OUTLINE
TIMELINE
moviesApp.js U
Movie.js U
movies.json U
package.json U
09_API_Swagger > Jurnal_09_2311104022 > Jurnal1 > moviesApp.js > writeMovies
82 * /api/Movies/{id}:
83 * get:
84 * summary: Get movie by index
85 * parameters:
86 * - in: path
87 * name: id
88 * schema:
89 * type: integer
90 * required: true
91 * description: Movie index
92 * responses:
93 * 200:
94 * description: Movie object
95 * content:
96 * application/json:
97 * schema:
98 * $ref: '#/components/schemas/Movie'
99 * 404:
100 * description: Movie not found
101 */
102 app.get("/api/Movies/:id", (req, res) => {
103 const movies = readMovies();
104 const id = parseInt(req.params.id);
105 if (isNaN(id) || id < 0 || id >= movies.length) {
106 return res.status(404).json({ message: "Movie not found" });
107 }
108 res.json(movies[id]);
109 });
110
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\KPL_MuhammadFathiRafa_2311104022_SE0701> cd .\09_API_Swagger\Jurnal_09_2311104022\Jurnal1\
PS D:\KPL_MuhammadFathiRafa_2311104022_SE0701\09_API_Swagger\Jurnal_09_2311104022\Jurnal1> node moviesApp.js
Server running at http://localhost:3000/
Swagger docs at http://localhost:3000/swagger
powershell
bash
powershell
powershell
powershell
node Jurnal1
main*
Git Graph
Ln 21, Col 30
Spaces: 2
UTF-8
CRLF
JavaScript
Go Live
Search
IND
17:38
5/30/2025

```
/**
 * @swagger
 * /api/Movies:
 *   post:
 *     summary: Add a new movie
 *     requestBody:
 *       required: true
 *       content:
 *         application/json:
 *           schema:
 *             $ref: '#/components/schemas/Movie'
 *     responses:
 *       201:
 *         description: Movie created
 *         content:
 *           application/json:
 *             schema:
 *               $ref: '#/components/schemas/Movie'
 */
app.post("/api/Movies", (req, res) => {
  const { Title, Director, Stars, Description } = req.body;
  const movie = new Movie(Title, Director, Stars, Description);
  const movies = readMovies();
  movies.push(movie);
  writeMovies(movies);
  res.status(201).json(movie);
});
```

S OUTPUT DEBUG CONSOLE TERMINAL PORTS

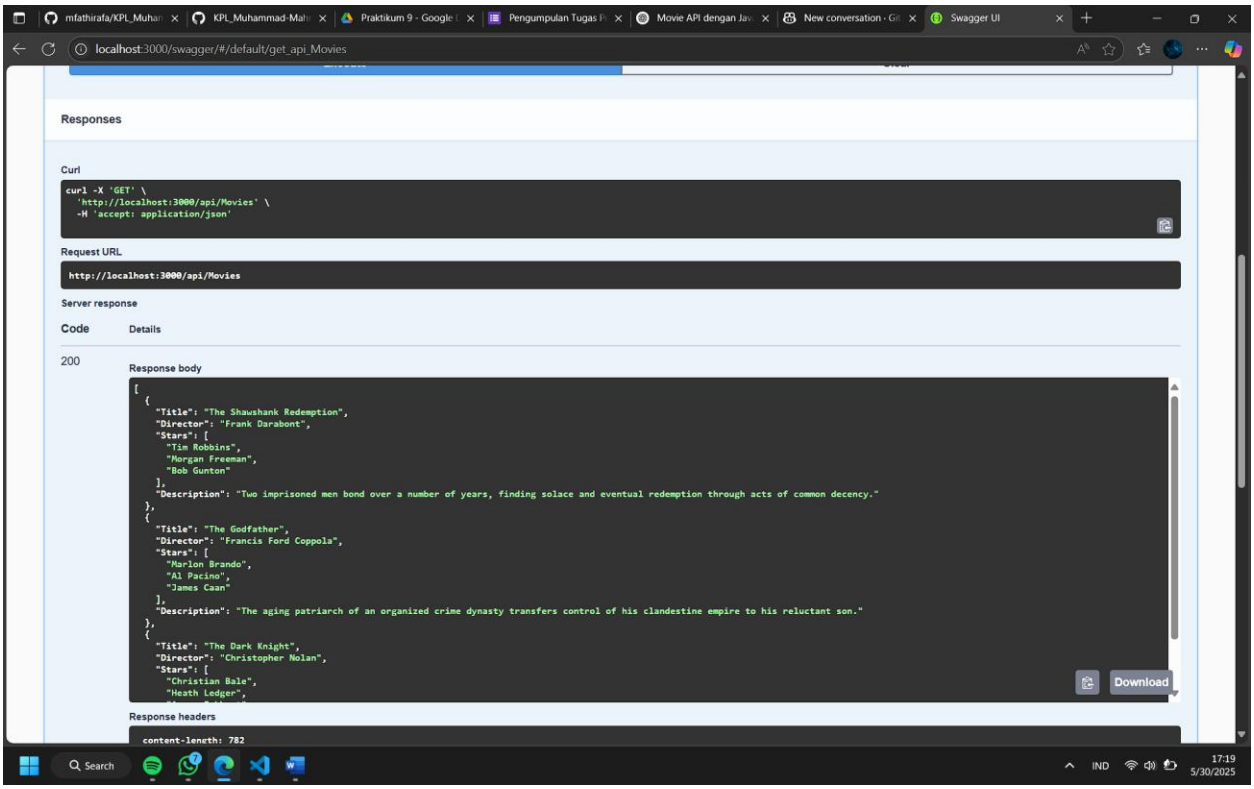
```
 *     parameters:
 *       - in: path
 *         name: id
 *         schema:
 *           type: integer
 *         required: true
 *         description: Movie index
 *     responses:
 *       204:
 *         description: Movie deleted
 *       404:
 *         description: Movie not found
 */
app.delete("/api/Movies/:id", (req, res) => {
  const id = parseInt(req.params.id);
  const movies = readMovies();
  if (isNaN(id) || id < 0 || id >= movies.length) {
    return res.status(404).json({ message: "Movie not found" });
  }
  movies.splice(id, 1);
  writeMovies(movies);
  res.status(204).send();
});

app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}/`);
  console.log(`Swagger docs at http://localhost:${port}/swagger`);
});
```

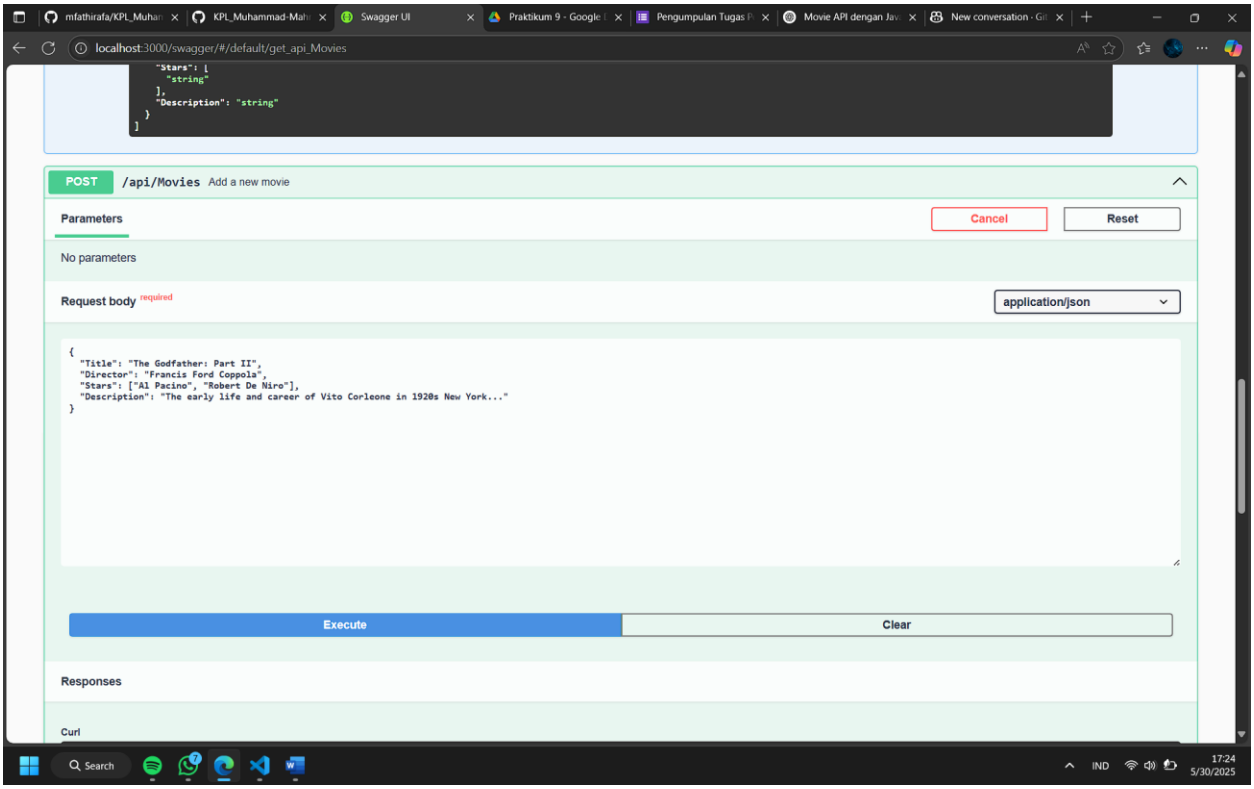
Penjelasan:

File ini merupakan inti aplikasi yang menjalankan server Express, mengatur endpoint RESTful untuk operasi CRUD pada data film (GET, POST, DELETE), melakukan pembacaan dan penulisan data ke file movies.json, serta menyediakan dokumentasi dan antarmuka uji API menggunakan Swagger UI di endpoint /swagger.

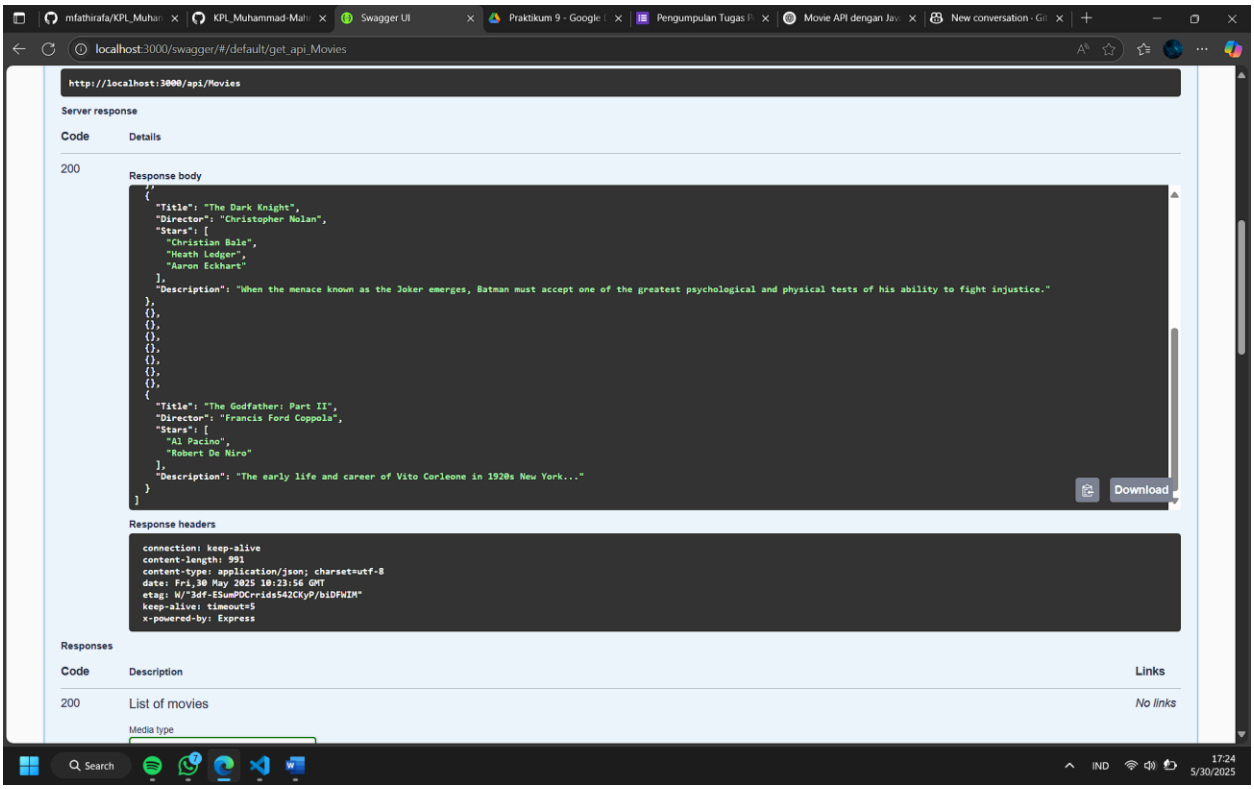
Get- all film



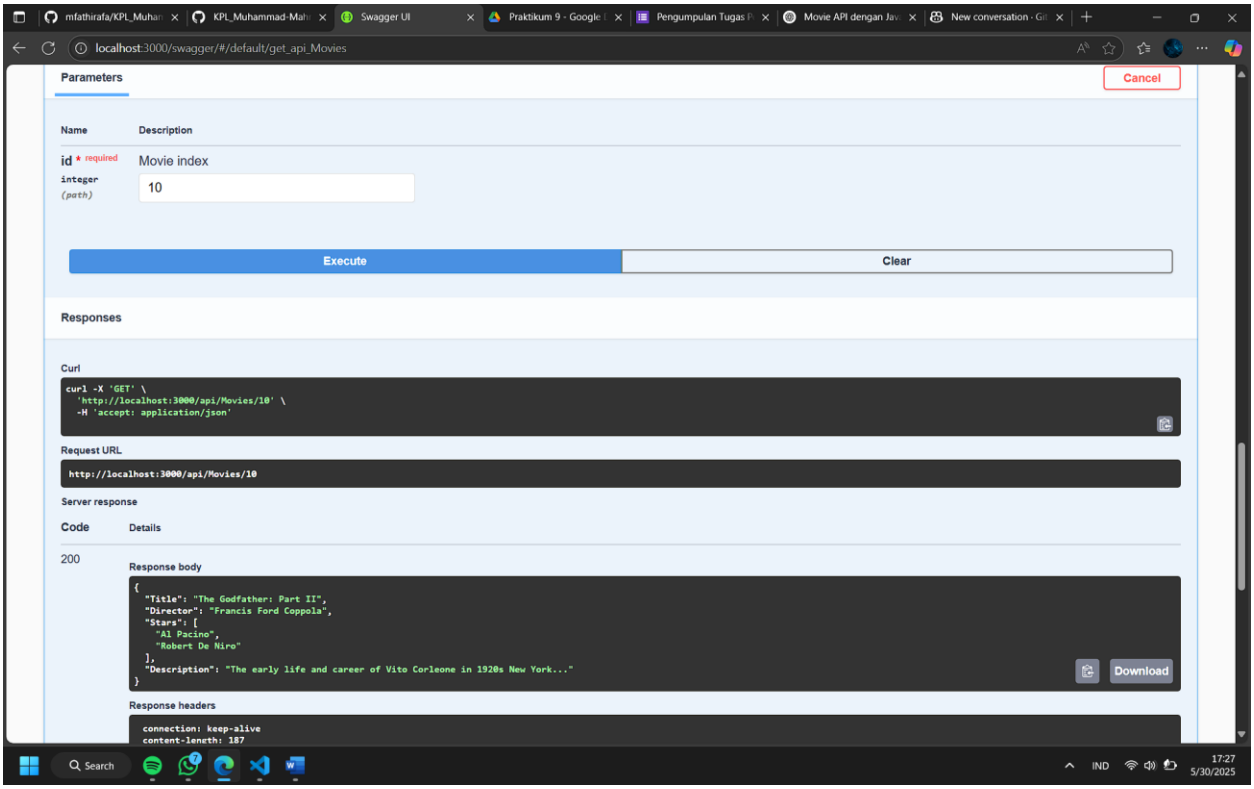
Post-menambahkan data film baru



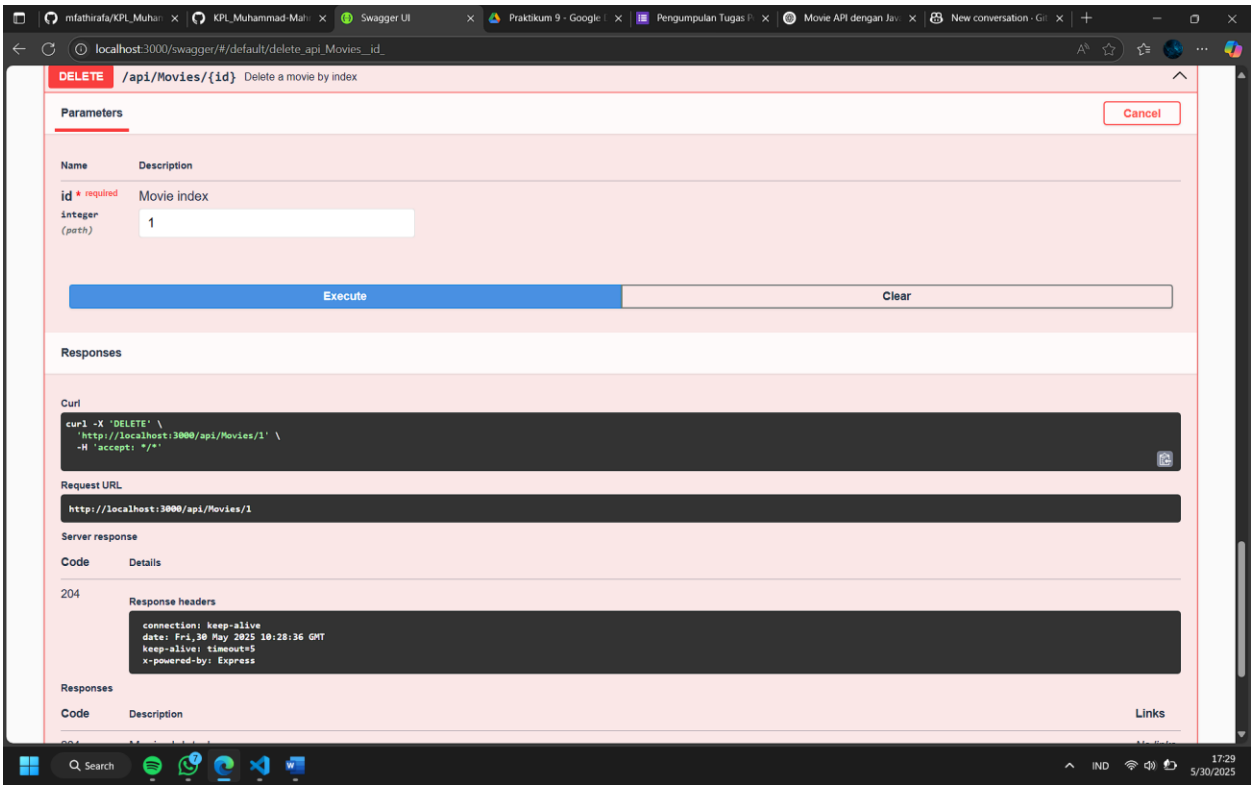
Get-melihat film yang ditambahkan



Get-berdasarkan id



Delete-id 1



Get-menngecek film yang telah didelete