MOVIES

API

Student
Mohammed Riyaan

Index

Page No	Title
3	About Movies API
5	Tech Stack
7	Proposed Solution
10	Demo and Installation of Movies API

About Project Movies API

Application Programming Interface (API)

API stands for Application Programming Interface. In simple words, an API is like a messenger that allows different software applications to communicate and interact with each other. It defines a set of rules and protocols that determine how one software application can access and use the functionalities or data of another application.

An API acts as an intermediary that enables different software applications to exchange information, request services, or perform actions. It provides a simplified and standardized way for developers to access the functionalities or data of another application without needing to understand the internal workings of that application.

Movies-API

The Movies API is an application programming interface (API) that provides access to a database of movies, allowing developers to retrieve movie information, search for movies, and perform various operations related to movies programmatically.

This API is typically used by developers to build movie-related applications, websites, or services. It provides a convenient way to access movie data, including details such as movie titles, release dates, genres, cast and crew information, ratings, and more.

By utilizing the Movies API, developers can create applications that offer features like movie search, movie recommendations, movie reviews, movie ratings, and movie-related information retrieval.

To interact with the Movies API, developers make HTTP requests to specific endpoints provided by the API. These endpoints typically have different routes or URLs to perform various operations, such as retrieving a list of movies, searching for movies by specific criteria, or retrieving details about a particular movie.

The Movies API responses are typically returned in a structured format, such as JSON (JavaScript Object Notation), which makes it easy for developers to parse and use the data within their applications.

Tech stack used in Movies API

Backend - Server

1. Programming Languages - JavaScript
2. Runtime Environment - Node JS
3. BackEnd Framework – ExpressJS

Client Tools

PostMan - Postman is a popular client tool used for testing and interacting with web APIs (Application Programming Interfaces). It provides a user-friendly interface that allows developers to make requests to APIs, inspect responses, and automate API testing and documentation.

NPM Packages

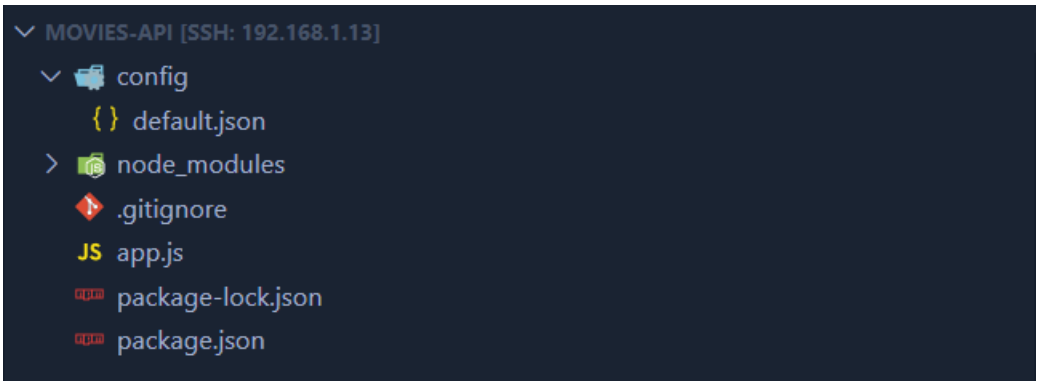
1. express:- The Express npm package is a fast, unopinionated, and minimalist web application framework for Node.js. It provides a robust set of features for building web applications and APIs. Express is one of the most popular and widely used frameworks in the Node.js ecosystem due to its simplicity and flexibility.

2. config:- The "config" npm package is a popular configuration management library for Node.js applications. It provides a convenient way to define and access configuration values for different environments (e.g., development, production, staging)

3. axios:- The "axios" npm package is a popular JavaScript library used for making HTTP requests from both browsers and Node.js applications. It provides a simple and intuitive API for handling asynchronous operations and interacting with APIs.

4. Proposed Solution

Folder Structure



1) config :- We are creating a config directory in your project's root directory. This is where you'll store your configuration files . Inside the config directory, create a configuration file for each environment or configuration profile you want to define. For example, you can create default.json for common settings. Here's an ex of default.json:-

```
{
    "PORT":5000,
    "MOVIES_API":"eef5fbbefe20ab8806b36d9974568649"
}
```

2) node_modules :- This folder contains the installed npm packages and their dependencies.

3) .gitignore :- The purpose of gitignore files is to ensure that certain files not tracked by Git remain untracked.

4) app.js :- This file serves as the entry point of the application where you set up the Express server, configure middleware, and define routes.

5) package-lock.json :- package-lock. json is a file that is automatically generated by npm when a package is installed. It records the exact version of every installed dependency, including its sub-dependencies and their versions.

6) package.json :- The package.json file contains information about the project, including its dependencies, scripts, and metadata.

7) README.md :- This file typically includes documentation about the project, providing information on how to set it up, run it, and any other relevant details.

Procedure

Installing express , config and axios packages

Importing packages

```
import express from "express"
import config from "config"
import axios from "axios"
```

We are declaring PORT and moviesAPI in which the values used in config file are stored When you execute const app = express(), you are creating a new Express application by invoking the express() function. By assigning the returned application object to the variable app (using the const keyword), you can use the app variable to configure and define routes, middleware, and other aspects of your Express application.

```
const PORT = config.get("PORT")
const moviesAPI = config.get("MOVIES_API")
```

We are defining the first route for the root URL (/) using the app.get() method. The callback function (req, res) => {...} is executed when an HTTP GET request is made to the root URL. In this case, it sends the response in json format {message:"Home Page"} back to the client with a status code (200).

```
app.get("/",(req,res)=>{
    res.status(200).json({message:"Home Page"})
})
```

The second route sets up a route with the path /movies/upcoming . Inside the route handler, an asynchronous function is defined using async and await. The variable moviesData uses Axios to make an HTTP GET request to the Movies API. We also define the base URL of the Movies API as moviesAPI. Once the response is received from the Movies API, the movie data is extracted from the response and to represent the results only , we are storing

it as moviesData.data.results in data variable .

Then to reperesent the Upcoming Movies we are using map method to the original_title variable where the title of the movies are stored.

Finally the results is stored in the variable moviesName

Then the response is sent to the client in a json format using the value of moviesName with status code (200).

If there is an error during the API request, an error message (“Internal Server Error”) along with 500 HTTP status code is sent as the response in json format.

```
app.get("/movies/upcoming",async(req,res)=>{
  try {
    const moviesData = await axios.get(`https://api.themoviedb.org/3/movie/upcoming?api_key=${moviesAPI}`)
    let data = moviesData.data.results
    let moviesName = data.map((x)=>x.original_title)
    res.status(200).json({"Upcoming Movies": moviesName})
  } catch (error) {
    res.status(500).json({message:"Internal Server Error"})
  }
})
```

If we enter the invalid route then the response should be “Invalid Route” in json format along with status code (404).

```
app.use((req,res)=>{
    res.status(404).json({message:"Invalid Route"})
})
```

The below code starts the server and listens on port . Once the server is up and running, the callback function (req, res) => {...} is executed, which logs the message `Server Listening At Port ${PORT}` to the console.

```
app.listen(PORT,()=>{
    console.log(`Server Listening At Port ${PORT}`)
})
```

Installation of Movies API

GitHub Repository: https://github.com/mohdriyaan/MoviesAPI

To clone and Install packages:

```
riyaan@riyaan:~$ git clone git@github.com:mohdriyaan/Movies-API.git
Cloning into 'Movies-API'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

To Run Project

cd Movies-API

npm i

npm start

Output :- After Executing The Server Starts Running At the Designated Port.

```
riyaan@riyaan:~/Movies-API$ npm start

> mohdriyaan@1.0.0 start
> node app.js

Server is Running at PORT Number 5000
```

After starting the server , you can access your application in a web browser or client tool such as PostMan by navigating to https://YOUR-WIFI-IP:5000. In this case we are using PostMan.

The Output Of First Route http://192.168.1.13:5000 Will Be :-

The Output Of Second Route http://192.168.1.13:5000/in Will Be :-

```
"Upcoming Movies": [
    "Guy Ritchie's The Covenant",
    "Renfield",
    "Astérix & Obélix : L'Empire du Milieu",
    "Knights of the Zodiac",
    "M3GAN",
    "To Catch a Killer",
    "Spider-Man: Across the Spider-Verse",
    "The Flash",
    "Transformers: Rise of the Beasts",
    "Marcel the Shell with Shoes On",
    "Transfusion",
    "Viejos",
    "Terrifier 2",
    "Extraction 2",
    "Kandahar",
    "Love Again",
    "Simulant",
    "Elemental",
    "About My Father",
    "Asteroid City"
]
```