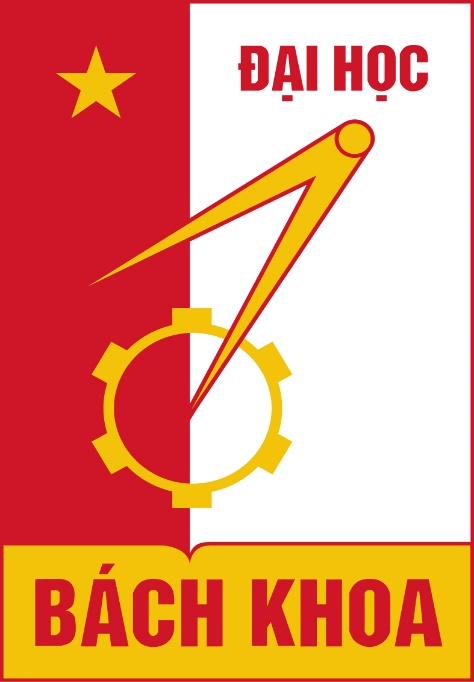
**Trường đại học Bách Khoa Hà Nội**

**Viện Công Nghệ Thông Tin và Truyền Thông**

----------o 0 o----------



**Báo cáo số 1**

Đề tài: Website đánh giá và review phim

**Giáo viên hướng dẫn**: Thầy Nguyễn Nhất Hải

**Sinh viên thực hiện**: Nguyễn Việt Tiến

Hà Nội, 18 tháng 5 năm 2019

# **Mục lục**

Mục lục

[**Mục lục** 1](#_Toc9071982)

[**LỜI MỞ ĐẦU** 2](#_Toc9071983)

[**Phần I: Tìm hiểu về NODE.JS và một số công nghệ liên quan** 3](#_Toc9071984)

[1. Giới thiệu chung và cài đặt môi trường 3](#_Toc9071985)

[1.1 Tổng quan Node.js 3](#_Toc9071986)

[1.2 Cài đặt môi trường 5](#_Toc9071987)

[2. Tìm hiểu về Node.js 5](#_Toc9071988)

[2.1 Node module, module.exports và require 5](#_Toc9071989)

[2.2 Sử dụng NPM (Node Package Manager) để quản lý package và module 6](#_Toc9071990)

[2.3 Cách dùng promise trong Node.js 7](#_Toc9071991)

[3. Tìm hiểu ExpressJS framework 9](#_Toc9071992)

[3.1 Giới thiệu ExpressJS framework 9](#_Toc9071993)

[3.2 Cài đặt và cách xây dựng cấu trúc thư mục 9](#_Toc9071994)

[3.3 Express router 10](#_Toc9071995)

[3.4 Template engine EJS 13](#_Toc9071996)

[3.5 Cài đặt và kết nối cơ sở dữ liệu trong Node.js 13](#_Toc9071997)

[3.6 Thực hiện tạo connection đến database (*Hình 12*) 14](#_Toc9071998)

[**Phần 2: Khảo sát và phân tích hệ thống** 14](#_Toc9071999)

[1.Phân tích yêu cầu đề 14](#_Toc9072000)

[2.Phân tích và thiết kế hệ thống 14](#_Toc9072001)

[2.1 Biểu đồ UseCase tổng quan 15](#_Toc9072002)

[2.2 Phân rã UseCase “User” 16](#_Toc9072003)

[2.3 Phân rã UseCase “Admin” 17](#_Toc9072004)

[2.4 Database 18](#_Toc9072005)

[**Phần 3: Demo sản phẩm** 19](#_Toc9072006)

[1.Cơ sở dữ liệu 19](#_Toc9072007)

[2.Giao diện sản phẩm 19](#_Toc9072008)

[**Phần 4: Kết luận** 22](#_Toc9072009)

[2.Chưa làm được 22](#_Toc9072010)

[3.Hướng phát triển 23](#_Toc9072011)

[**TÀI LIỆU THAM KHẢO** 23](#_Toc9072012)

# **LỜI MỞ ĐẦU**

Với sự phát triển của những công nghệ, hiện nay có rất nhiều các ngôn ngữ lập trình được ưa chuộng trong đó có Javascript. Node.js là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng. Rất nhiều những ông lớn trong lĩnh vực công nghệ đang sử dụng Nodejs như: *Amazon, Ebay, Linkedin, Microsoft, Paypal, Trello, Uber* và còn nhiều cái tên nổi tiếng khác nữa.

Câu hỏi đặt ra là tại sao những ông lớn lại lựa chọn sử dụng Node.js? Để trả lời câu hỏi trên và học thêm công nghệ mới em quyết định lựa chọn chủ đề tìm hiểu về Nodejs. Hiện thực của việc tìm hiểu là sản phẩm “Website đánh giá và review phim”.

Tuy đã cố gắng nhưng trong project không thể tránh được nhiều thiếu sót, em mong được sự phản hồi và góp ý từ thầy để hoàn thiện hơn hiểu biết và sản phẩm này. Em xin chân thành cảm ơn!

Nguyễn Việt Tiến

Hà Nội, ngày 18 tháng 5 năm 2019

# **Phần I: Tìm hiểu về NODE.JS và một số công nghệ liên quan**

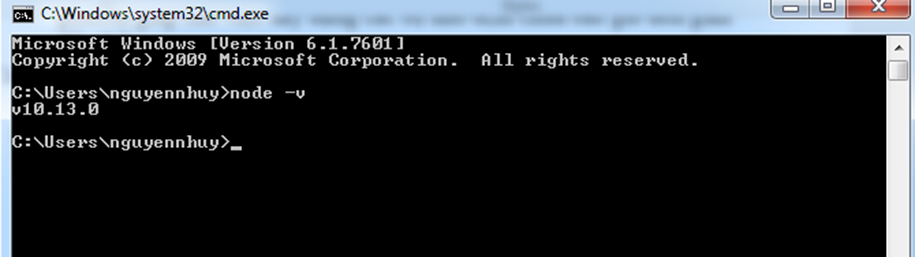
## Giới thiệu chung và cài đặt môi trường

### 1.1 Tổng quan Node.js

* ***Khái niệm Node.js***:
* Node.js là một nền tảng (Platform) phát triển độc lập được xây dựng ở trên Javascript Runtime của Chrome mà chúng ta có thể xây dựng được các ứng dụng mạng một cách nhanh chóng và dễ dàng mở rộng.
* Node.js cung cấp kiến trúc hướng sự kiện (event-driven) và non-blocking I/O API, tối ưu hóa thông lượng của ứng dụng và có khả năng mở rộng cao
* Mọi hàm trong Node.js là không đồng bộ (asynchronous). Do đó, các tác vụ đều được xử lý và thực thi ở chế độ nền (background processing)
* Node.js tạo ra được các ứng dụng có tốc độ xử lý nhanh, realtime thời gian thực. Node.js áp dụng cho các sản phẩm có lượng truy cập lớn, cần mở rộng nhanh, cần đổi mới công nghệ, hoặc tạo ra các dự án Startup nhanh nhất có thể.
* ***Ứng dụng của Node.js***
* **Websocket server**: Các máy chủ web socket như là Online Chat, Game Server…
* **Fast File Upload Client**: Là các chương trình upload file tốc độ cao.
* **Ad Server**: Các máy chủ quảng cáo.
* **Cloud Services**: Các dịch vụ đám mây.
* **RESTful API**: Đây là những ứng dụng mà được sử dụng cho các ứng dụng khác thông qua API.
* **Any Real-time Data Application**: Bất kỳ một ứng dụng nào có yêu cầu về tốc độ thời gian thực. Micro Services: Ý tưởng của micro services là chia nhỏ một ứng dụng lớn thành các dịch vụ nhỏ và kết nối chúng lại với nhau. Node.js có thể làm tốt điều này.
* ***Nhược điểm của Node.js***
* Giống như hầu hết các công nghệ mới, việc triển khai Node.js trên host không phải là điều dễ dàng.
* Nếu bạn cần xử lý các ứng dụng tốn tài nguyên CPU như encoding video, convert file, decoding encryption… hoặc các ứng dụng tương tự như vậy thì không nên dùng NodeJS (Lý do: Node.js được viết bằng C++ & Javascript, nên phải thông qua thêm 1 trình biên dịch của NodeJS sẽ lâu hơn 1 chút ). Trường hợp này bạn hãy viết 1 Addon C++ để tích hợp với Node.js để tăng hiệu suất tối đa.
* ***Ưu điểm của Nodejs***
* Đặc điểm nổi bật của Node.js là nó *nhận và xử lý nhiều kết nối chỉ với một single-thread.* Điều này giúp hệ thống tốn ít RAM nhất và chạy nhanh nhất khi không phải tạo thread mới cho mỗi truy vấn giống PHP. Ngoài ra, tận dụng ưu điểm non-blocking I/O của Javascript mà Node.js tận dụng tối đa tài nguyên của server mà không tạo ra độ trễ như PHP.
* JSON APIs Với cơ chế *event-driven, non-blocking I/O(Input/Output)* và mô hình kết hợp với Javascript là sự lựa chọn tuyệt vời cho các dịch vụ Webs làm bằng JSON.
* Khả năng xử lý nhiều Request/s đồng thời thời gian phản hồi nhanh.
* Shelling tools unix Node.js sẽ tận dụng tối đa Unix để hoạt động. Tức là Node.js có thể *xử lý hàng nghìn Process và trả ra 1 luồng* khiến cho hiệu xuất hoạt động đạt mức tối đa nhất và tuyệt vời nhất.
* Với sự ra đời của các ứng dụng di động & HTML 5 nên Node.js rất hiệu quả khi xây dựng những *ứng dụng thời gian thực (real-time applications)* như ứng dụng chat, các dịch vụ mạng xã hội như Facebook, Twitter,…
* ***Khi nào nên dùng Node.js***
* Node.js thực sự tỏa sáng trong việc xây dựng RESTful API (json). Gần như không có ngôn ngữ nào xử lý JSON dễ dàng hơn Javascript, chưa kể các API server thường không phải thực hiện những xử lý nặng nề nhưng lượng concurrent request thì rất cao. Mà Node.js thì xử lý non-blocking. Chẳng còn gì thích hợp hơn Node.js trong trường hợp này!
* Những website stateful. Node.js xử lý mọi request trên cùng một process giúp cho việc xây dựng các bộ nhớ đệm chưa bao giờ đơn giản đến thế

### 1.2 Cài đặt môi trường

* Đầu tiên, bạn truy cập vào liên kết [*https://nodejs.org/en/download/*](https://nodejs.org/en/download/)để tải về phiên bản Node.js tương thích với hệ điều hành mà bạn đang sử dụng.
* Sau khi tải về bạn click đúp vào file cài đặt và tiến hành tiến trình cài đặt. Trong khi cài đặt bạn nên chọn các thông số mặc định sẵn có mà trình cài đặt đưa ra cho bạn.
* Sau khi cài xong kiểm tra bằng cách vào cửa sổ cmd thực hiện lệnh *node –v.* Nếu thành công thì ta sẽ được ảnh như sau (*Hình 1*):

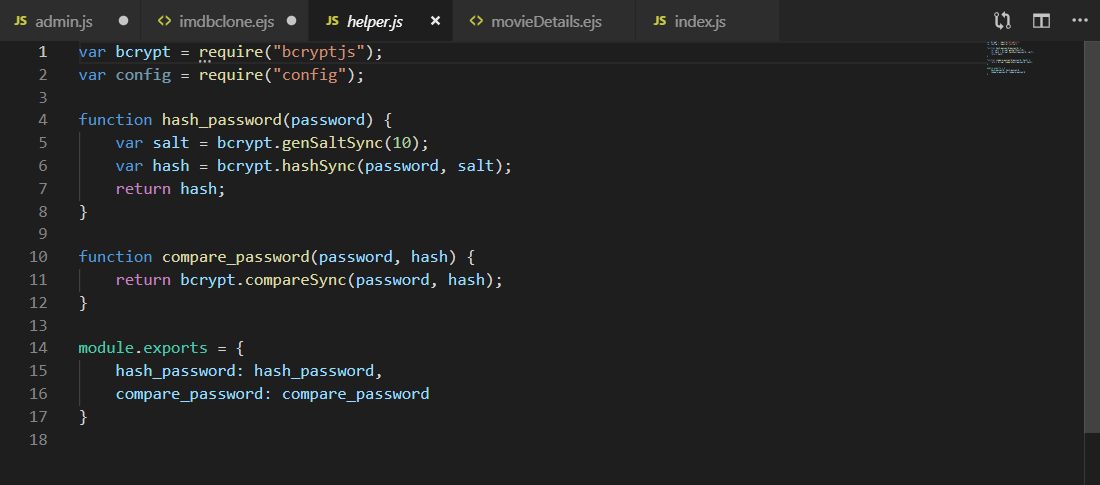


Hình 1: Kiểm tra cấu hình thành công

## Tìm hiểu về Node.js

### 2.1 Node module, module.exports và require

* Module
* Node.js sử dụng module để đơn giản hóa việc tạo ra các ứng dụng phức tạp. Module giống như các thư viện trong C, Java,…
* Mỗi module chứa một tập các hàm chức năng có liên quan đến một đối tượng của module qua đó giúp việc quản lý mã lệnh của chương trình dễ dàng hơn.
* Một module có thể đơn giản là một hàm hay một đối tượng. Mỗi module thường được khai bảo ở một tập tin riêng rẽ.
* Ví dụ:
* Module có sẵn: *“http”* là module chứa các hàm cụ thể liên quan đến thiết lập HTTP. Node.js cung cấp một vài các module core kèm theo để hỗ trợ chúng ta truy cập file trên hệ thống, tạo các máy chủ HTTP, TCP/UDP, và các hàm tiện ích nhỏ hữu dụng khác.
* Module tự viết (*Hình 2: Module giúp mã hóa và giải mã mật khẩu*)



Hình 2: Module giúp mã hóa và giải mã mật khẩu

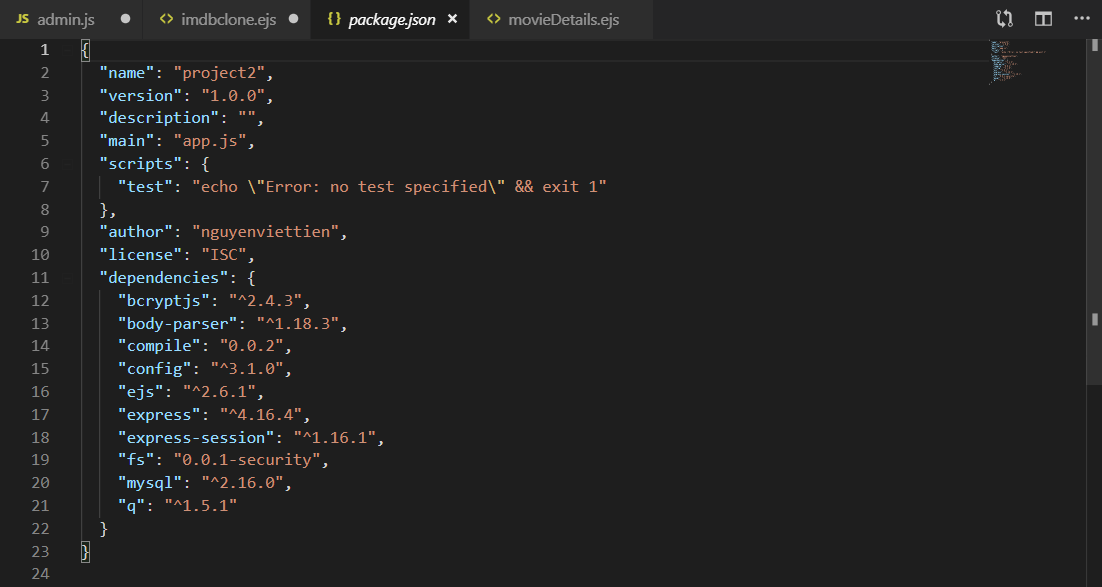
* Module.exports
* Module là các đoạn code được đóng gói lại với nhau,... Code trong một Module thường là private – nghĩa là các hàm, biến được định nghĩa và truy cập bởi bên trong của Module. Để "chìa ra" các hàm hoặc biến để sử dụng bên ngoài module bạn cần sử dụng module.exports.
* Ví dụ: *Hình 2* là module gồm 2 hàm *hash\_password()* và *compare\_password(),* để sử dụng được bên ngoài module ta cần gọi module.exports hai hàm đó.
* Require
* Để sử dụng 1 module nào đó, ta đơn giản chỉ cần khai báo với hàm *require(“ten\_module”);*
* Ví dụ: *Hình 2* sử dụng module *bcryptjs* có sẵn của Node.js để thực hiện mã hóa và giải mã mật khẩu. Ta khai báo bằng cách khởi tạo biến *bcrypt*

*var bcrypt = require(“bcryptjs”);*

Qua đó ta có thể sử dụng: *bcrypt.genSaltSync()* và *bcrypt.hashSync()*

### 2.2 Sử dụng NPM (Node Package Manager) để quản lý package và module

* Khái niệm NPM
* NPM viết tắt của Node Package Manager là một công cụ (chương trình) quản lý các thư viện lập trình Javascript cho Node.js.
* NPM về cơ bản khá giống với Maven, sự khác biệt ở đây là NPM quản lý các thư viện Javascript trong khi đó Maven quản lý các thư viện Java.
* Nó sẽ giúp ta thực hiện việc include những thư viện một cách tự động thay vì ta phải tải source code của thư viện đó rồi đặt nó và trong project của mình. Bạn chỉ cần chạy một dòng lệnh để tải về và dễ dàng include chúng.
* Cài đặt NPM
* Khi bạn có 1 project trắng và chưa có gì, để bắt đầu ta gõ lệnh: *npm init.* Khi đó, terminal sẽ hiện ra một số câu hỏi về thông tin project, ví dụ tên project, version, git repository,.. Ta có thể nhấn Enter để bỏ qua cái nào mình không cần thiết nhập.
* Sau khi hoàn thành, bạn sẽ thấy file *package.json* được tạo ra trong thư mục project như hình vẽ (*Hình 3*):



Hình 3: File package.json được tạo ra

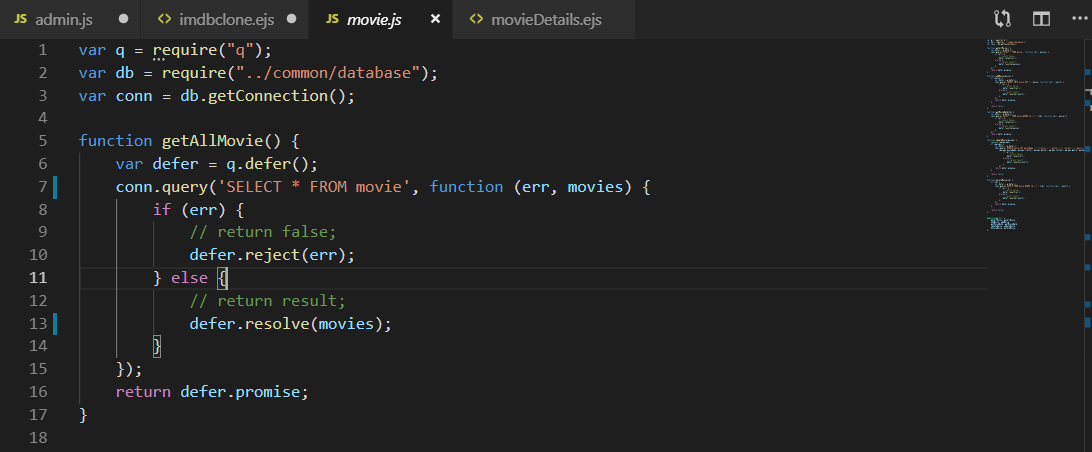
* Cài đặt các package trong NPM
* Thực hiện câu lệnh *npm install (-g) (--save) “ten\_module”* thêm tùy chọn *-g* nếu muốn cài global, *--save* nếu muốn hiển thị module đó ở file *package.json.*
* *Hình 3:* Một số module được dùng trong ứng dụng đã được cài như: *bcryptjs, body-parser, config, ejs, express, express-session, mysql,.*

### 2.3 Cách dùng promise trong Node.js

* **Vì sao nên dùng Promise thay cho callback**
* Promise là một giải pháp giúp bạn xử lý các hàm asynchronous một cách tốt hơn. Promise được sinh ra để thể hiện cho kết quả sau cùng của một tác vụ bất đồng bộ.
* Nếu so sánh với các phướng thức callback truyền thống thì promise cung cấp một biện pháp thay thế đơn giản hơn trong việc thực thi, biên soạn và quản lý các tác vụ bất đồng bộ.
* Ngoài ra promise còn cho phép chúng ta handle các lỗi xảy ra khi thực thi tác vụ bất đồng bằng các sử dụng các phương thức tiếp cận tương tự với sử dụng try/catch trong các tác vụ đồng bộ (synchronous) thông thường.
* **Cách dùng Promise trong Node.js**
* Để sử dụng Promise trong Nodejs, thường thì các bạn cần cài thêm một module như “q” để viết Promise.
* Để cài đặt module “q” vào project của mình ta thực hiện câu lệnh:

*npm install q --save*

* Để sử dụng module “q”, ta thực hiện *require()* module này vào nơi mà mình sử dụng. Em sử dụng *q.defer()* trong project như sau*:*



Hình 4: Xây dựng hàm getAllMovie() để trả về một promise nếu thành công

* Sau đó trong *router.get("/signup", function(req, res) {…};* em gọi đến hàm getAllMovie() để lấy các movie từ database nếu như đăng kí chính xác.



Hình 5: Gọi hàm getAllMovie()

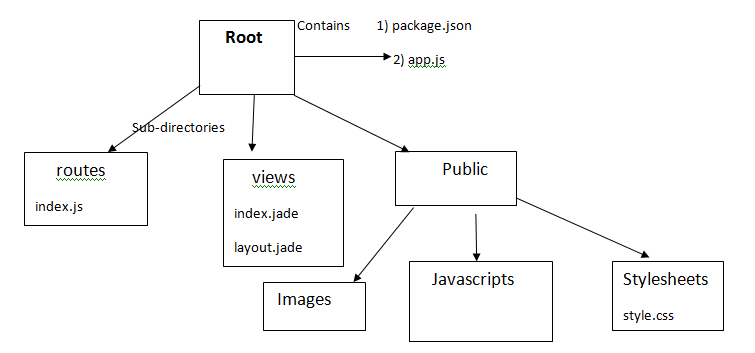
## 3. Tìm hiểu ExpressJS framework

### 3.1 Giới thiệu ExpressJS framework

* ExpressJS là một Framework nhỏ, nhưng linh hoạt được xây dựng trên nền tảng của Nodejs. Nó cung cấp các tính năng mạnh mẽ để phát triển web hoặc mobile.
* Về các package hỗ trợ: ExpressJS có vô số các package hỗ trợ nên các bạn không phải lo lắng khi làm việc với Framework này.
* Về performance: ExpressJS cung cấp thêm về các tính năng (feature) để dev lập trình tốt hơn. Chứ không làm giảm tốc độ của Node.js.

### 3.2 Cài đặt và cách xây dựng cấu trúc thư mục

* **Cấu trúc của ExpressJS** (*Hình 6*):

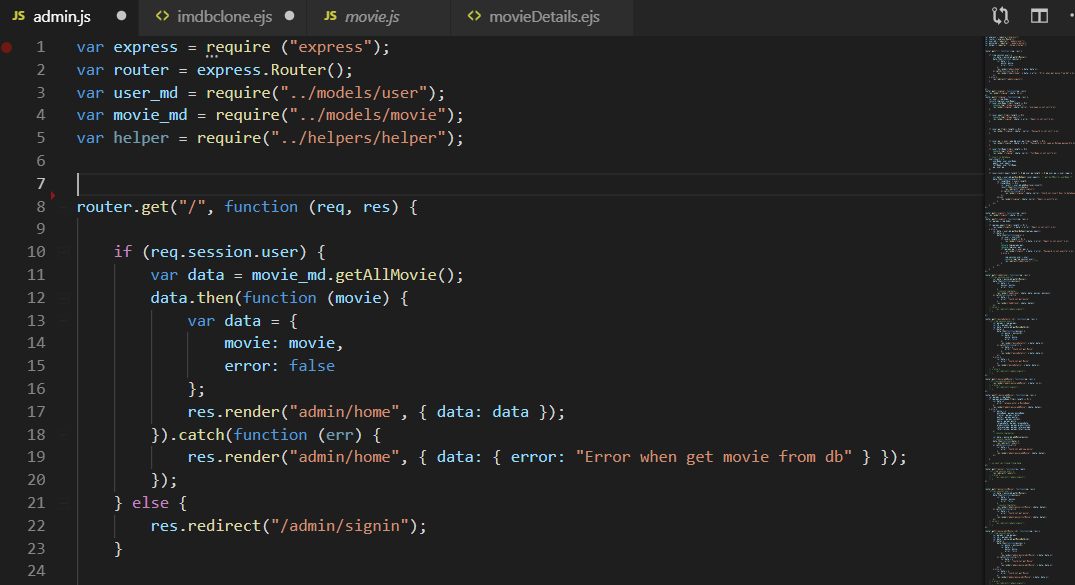


*Hình 6: Cấu trúc của ExpressJS*

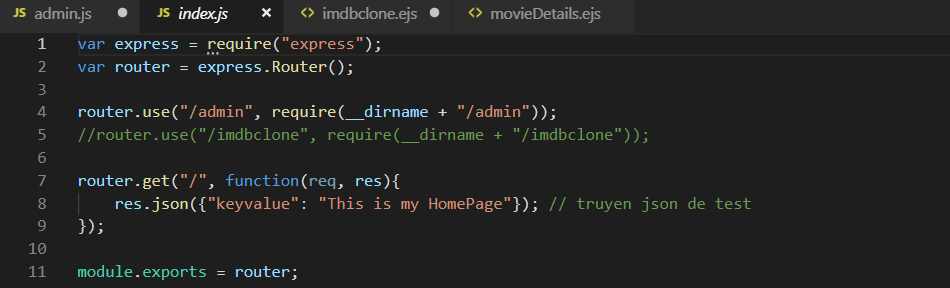
* app.js chứa các thông tin về cấu hình, khai báo, các định nghĩa,... để ứng dụng của chúng ta chạy được.
* package.json chứa các package đã cấu hình trong ứng dụng.
* Folder routes: chứa các route có trong ứng dụng.
* Folder view: chứa view/template cho ứng dụng.
* Folder public chứa các file css, js, images,... cho ứng dụng
* Ngoài ra chúng ta có thể tự thêm những folder khác phục vụ cho mục đích riêng.
* **Cài đặt ExpressJS**
* Cũng như những module khác chúng ta cài đặt ExpressJS bằng cách sử dụng câu lệnh sau: *npm install express --save*

### 3.3 Express router

* **Khái niệm**
* Router là một Object, là một instance riêng của middleware và route. Chính vì nó là một instance của middleware và route nên nó có các chức năng của cả hai. Chúng ta có thể gọi nó là một mini-application.
* Các Application dùng ExpressJS làm core đều có phần Router được tích hợp sẵn trong đó.
* Router hoạt động như một middleware nên chúng ta có thể dùng nó như một arguments (*Hình 7*). Hoặc dùng nó như một arguments cho route khác (*Hình 8*).

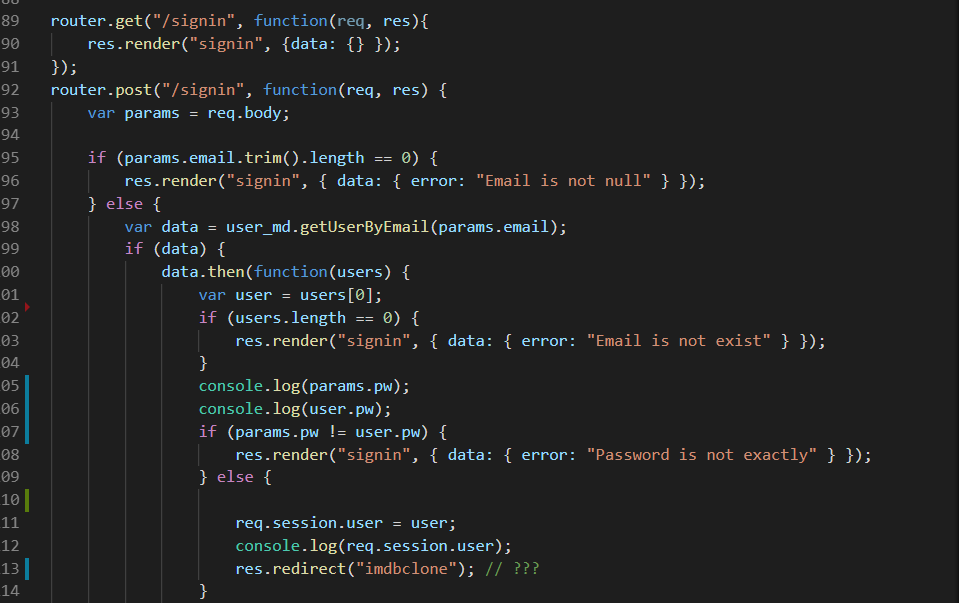


*Hình 7: Router trang chủ của ứng dụng*



*Hình 8: Router include router admin trong ứng dụng demo*

* **Tìm hiểu router.method()**
* Router.method() cung cấp cho chúng ta chức năng Routing trong ExpressJS. Cụ thể method ở đây là các HTTP method mà chúng ta thường xuyên sử dụng. Chẳng hạn get, post, put,...



*Hình 9: Phương thức get và post với đường dẫn signin*



*Hình 10: Phương thức put và delete*

### 3.4 Template engine EJS

* Sử dụng template engine ejs là cách nhanh và dễ dàng để quản lý phần views của ứng dụng.
* Để sử dụng được trước tiên ta phải cài module ejs vào project bằng cách thực hiện câu lệnh: *npm install ejs –save*
* Sau đó ta cấu hình lại đường dẫn của tầng views và template engine trong file app.js như sau:



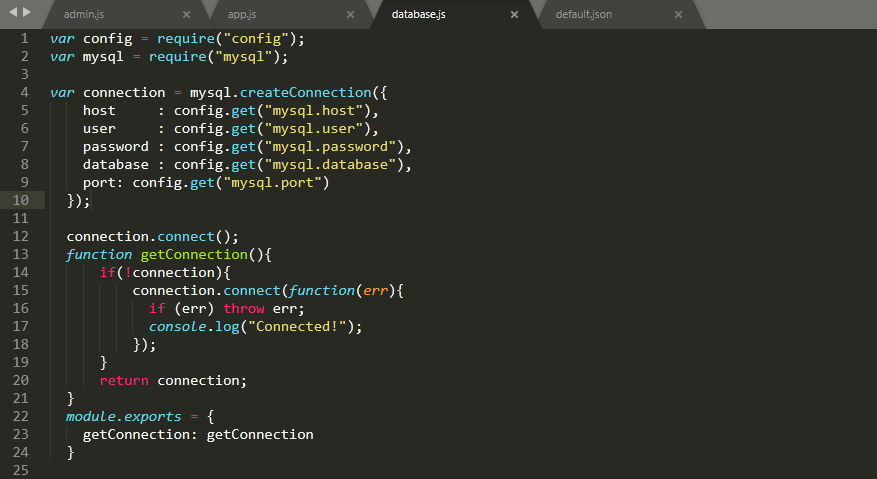
### 3.5 Cài đặt và kết nối cơ sở dữ liệu trong Node.js

* Cài đặt cơ sở dữ liệu mysql cho project bằng cách thêm module mysql vào project. Chạy lệnh : *npm install mysql –save*.
* Tạo folder config để cấu hình cho server và cơ sở dữ liệu (*Hình 11*)



*Hình 11 : File cấu hình hệ thống*

### 3.6 Thực hiện tạo connection đến database (*Hình 12*)



*Hình 12: Xây dựng module trả về kết nối với database*

# **Phần 2: Khảo sát và phân tích hệ thống**

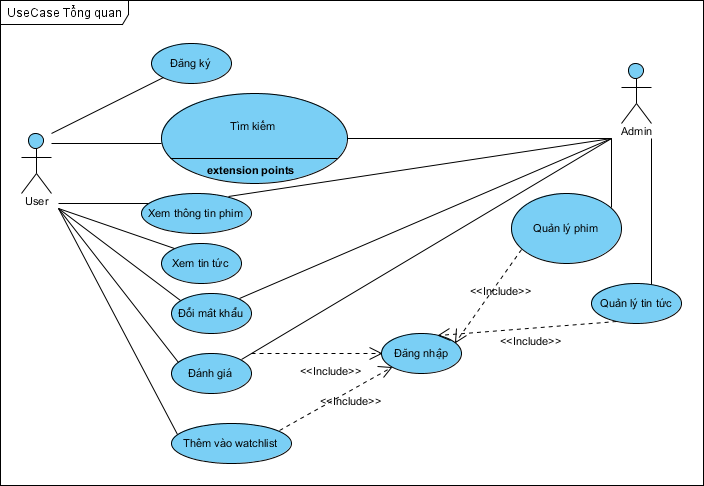
## 1.Phân tích yêu cầu đề

Tên đề tài: “Website đánh giá và review phim”

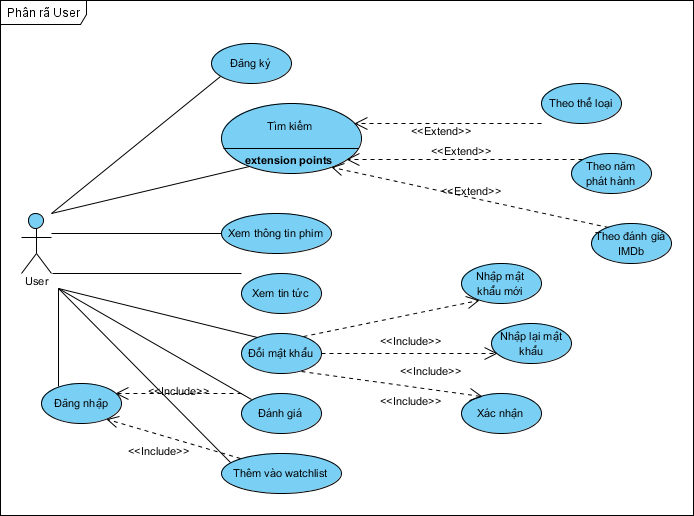
* Chức năng: đây là một website cho phép tra cứu thông tin của các bộ phim điện ảnh đã được đánh giá.
* Đăng ký tài khoản user
* Cho phép thêm, sửa, xóa thông tin các bộ phim ở cơ sở dữ liệu.
* Đánh giá phim (chỉ khi là user mới được đánh giá)
* Các phim đã xem
* Các phim đã đánh giá
* Hiển thị danh sách các bộ phim theo từng thể loại
* Hiển thị danh sách các bộ phim theo từng năm sản xuất
* Hiển thị danh sách các bộ phim theo điểm đánh giá IMDB
* Tìm kiếm các bộ phim theo thể loại
* Tìm kiếm phim theo tên nhà sản xuất
* Tìm kiếm phim theo tên diễn viên có xuất hiện

## 2.Phân tích và thiết kế hệ thống

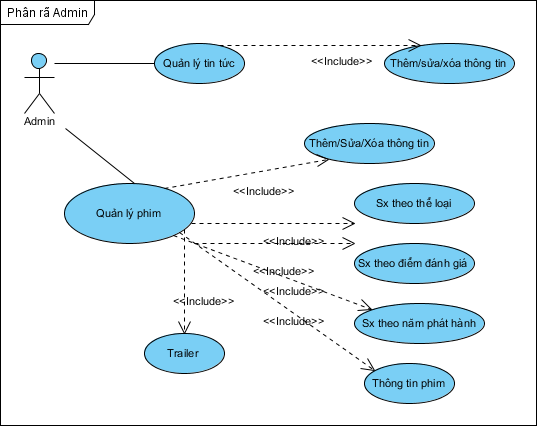
### 2.1 Biểu đồ UseCase tổng quan



### 2.2 Phân rã UseCase “User”



### 2.3 Phân rã UseCase “Admin”



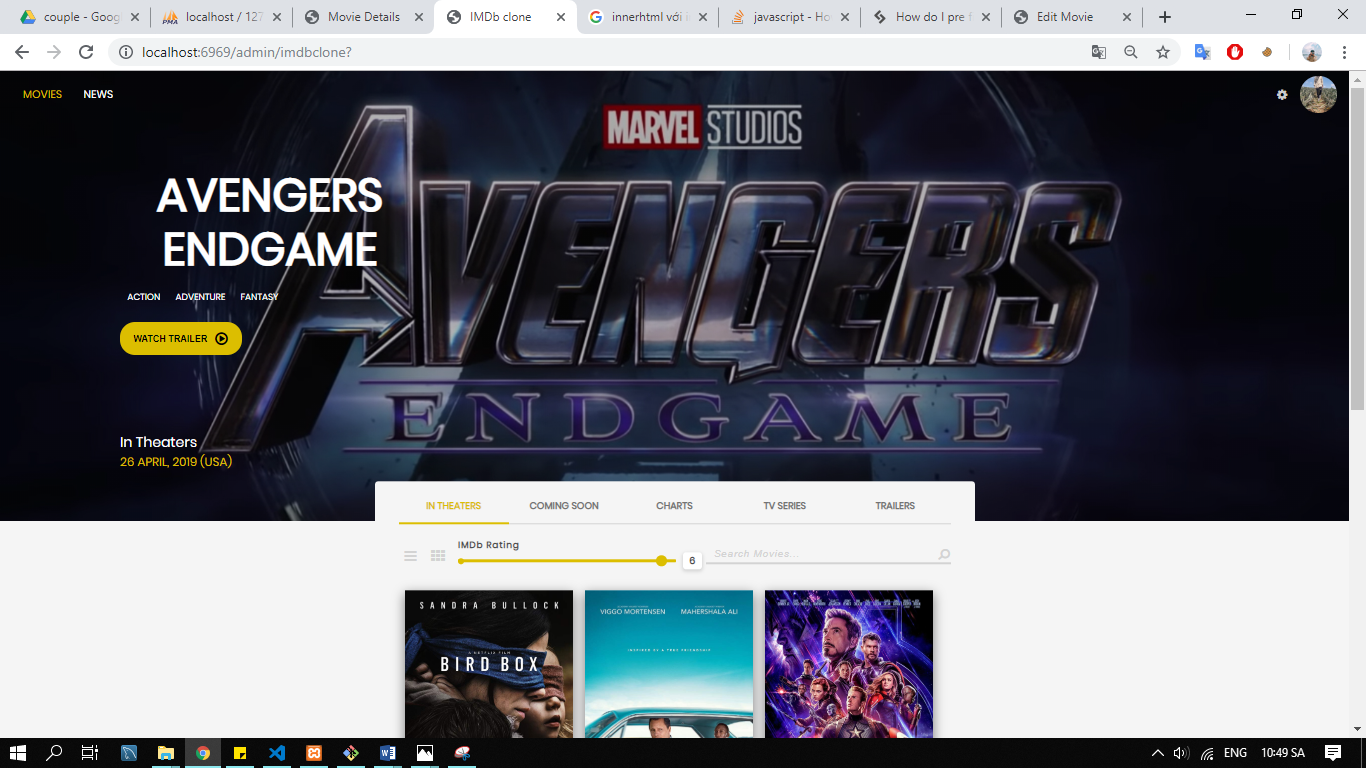
### 2.4 Database

# **Phần 3: Demo sản phẩm**

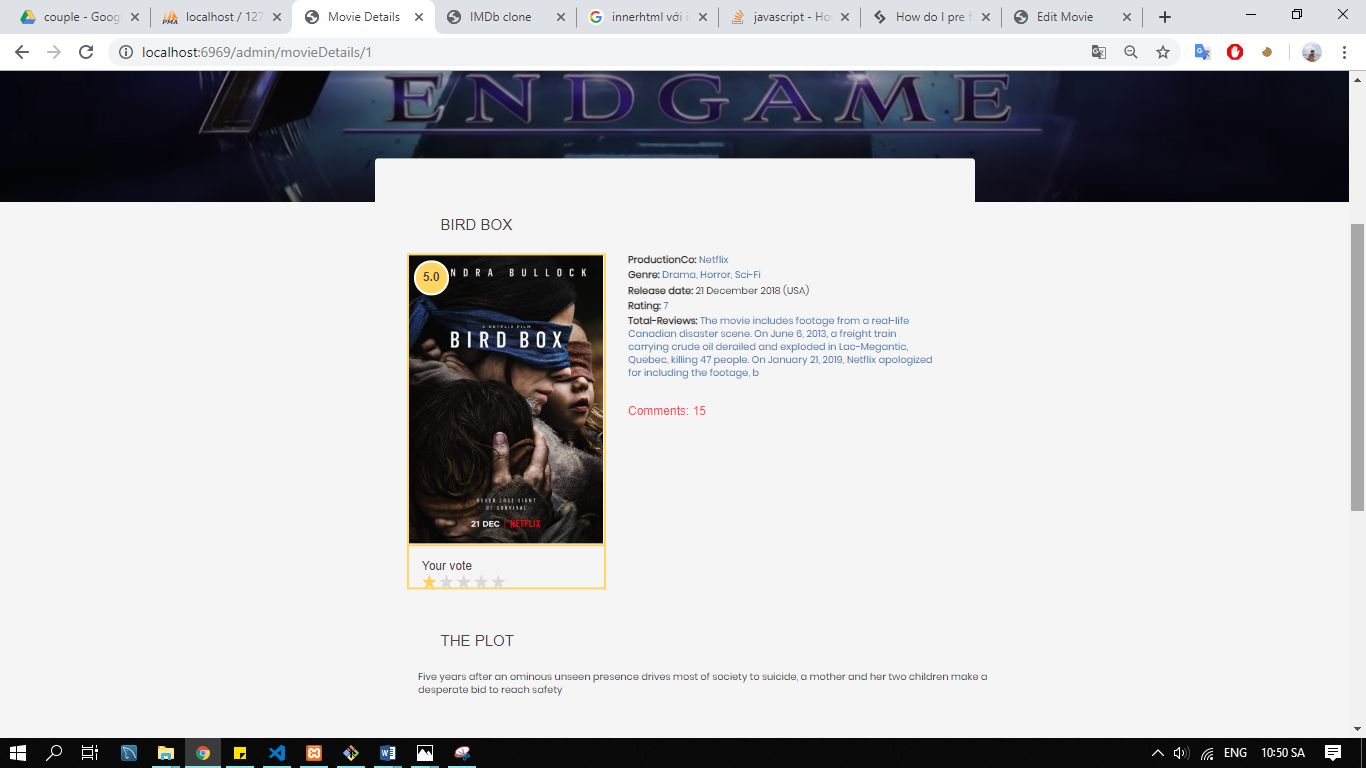
## 1.Cơ sở dữ liệu

## 2.Giao diện sản phẩm

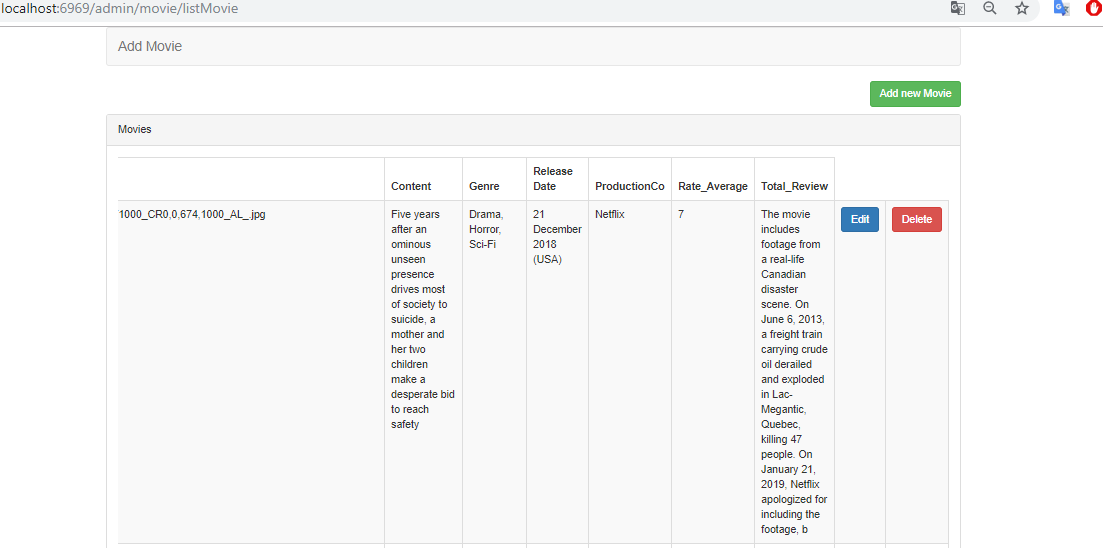
2.1 Home



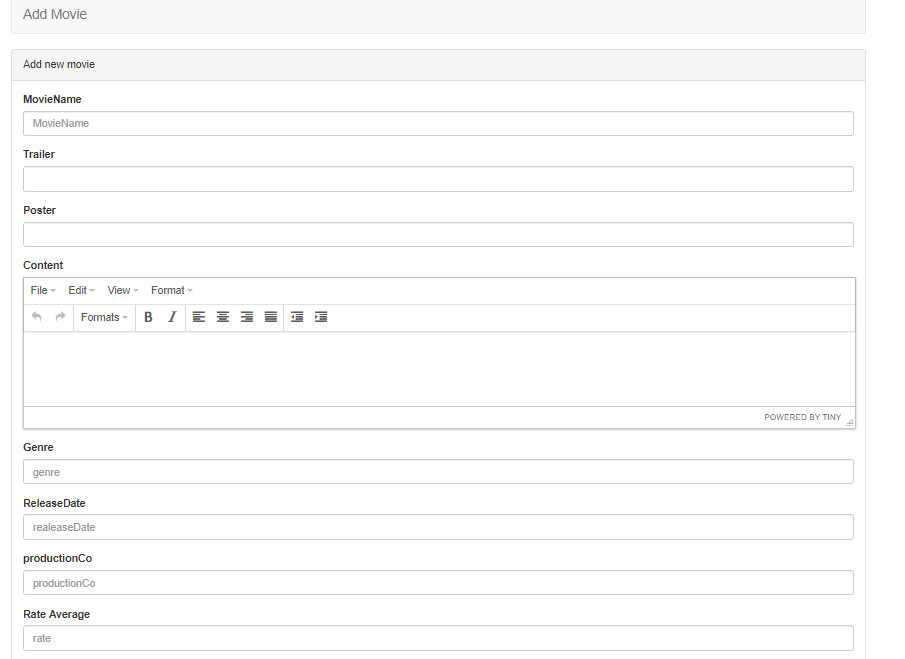
2.2 Thông tin chi tiết Movie



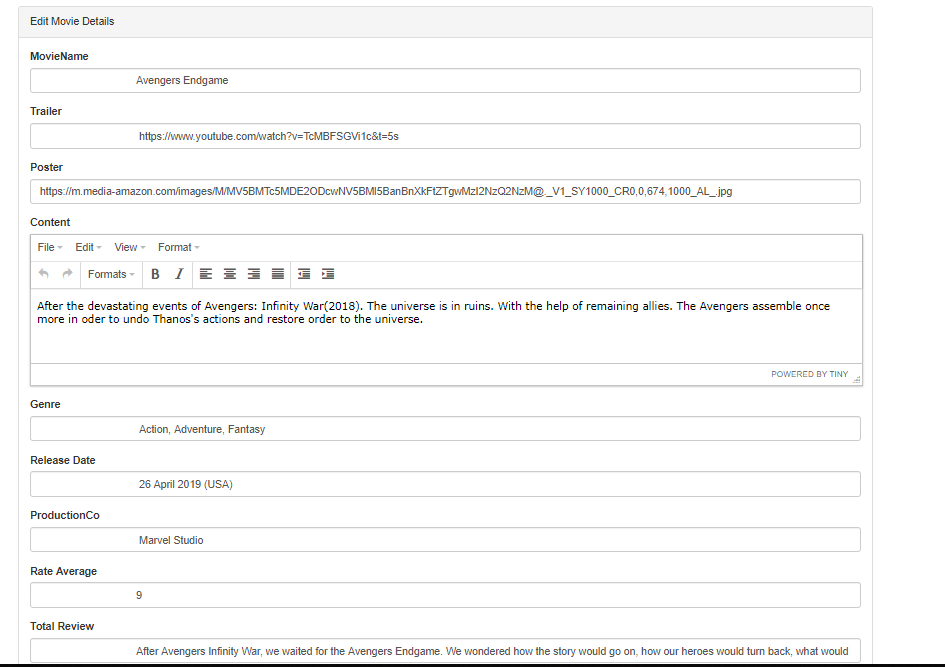
2.3 Danh sách movie



2.4 Thêm mới movie



2.5 Chỉnh sửa thông tin Movie



# **Phần 4: Kết luận**

**1. Đã làm được**

* Sau khi làm project về tìm hiểu về Node.js em đã có thể nắm cơ bản về Node.js và một số module và framework trong Node.js như: ExpressJS, http, …
* Hiểu về cách quản lý package trong Node.js bằng NPM

## 2.Chưa làm được

* Giao diện chưa được đẹp mắt
* Chưa xử lý được sự kiện comment reallife
* Chưa hoàn thiện được những chắc năng ban đầu đề ra khi phân tích nghiệp vụ: hiển thị phim theo từng thể loại, đánh giá phim, trang News….

## 3.Hướng phát triển

* Thiết kế giao diện đẹp mắt hơn
* Hoàn thiện các chức năng còn thiếu

# **TÀI LIỆU THAM KHẢO**

* <https://viblo.asia/>
* <https://techmaster.vn>
* <https://www.codehub.vn/>
* <https://www.tutorialspoint.com/nodejs/>
* Và một số tài liệu trên mạng khác