Docker Project

CAB432 CLOUD COMPUTING ASSIGNMENT 1

Ka Long Lee

N9845097

Introduction

The main services of the proposed application are to deliver the latest information and the discussion about the movie to its fans. There are two primary information will be provided to the user, it includes movie news and tweets. Users can check the latest news of the movie even if hasn't released yet. They also can find out the most widely enjoyed movies at the moment by the ranking so that they can see how people talk or feel about the movie by checking related tweets. It is also interesting to note that they might also find out some spoiler about the movies via tweets.

Services

Provider	Usage	URL
The Movie Database (TMDB)	Retrieving movies information (title, Image, ranking, overview)	https://developers.themoviedb.org/3/getting-started/introduction
NEWSAPI	Retrieving news related to the specific movie	https://newsapi.org/
TwitterAPI	Retrieving tweets related to the specific movie so that user can join the discussion	https://developer.twitter.com/en/docs.html

Mashup Use Cases

User Story	Description	Service Used
Getting Movie News	As a movie fan I want to check the news of the movie so that I can get the latest information related to this movie even if hasn't released yet.	• The Movie Database (TMDB) • NEWSAPI
Getting Movie Tweets	As a movie fan I want to check the tweets of the movie so that I can know how people discuss and find some spoiler about the movie.	• The Movie Database (TMDB) • TweeterAPI

Getting Movie News

Appendix A-1 shows the ranking page of the web application. The top half of the picture shows the contents of the ranking page, while the bottom half displays the console of the browser recording the network traffic. User can find out the upcoming movies by selecting the type of ranking to upcoming. The application will then fire a **GET** method HTTP request automatically to the cloud server for getting upcoming movies information. After the server application receives the HTTP request, it will then send another **GET** method HTTP request to fetch movie data from The Movie Database (TMDB) API. Once the response data is received by the server, it will transfer the data to the client. Finally, the web browser receives the movies data and updates the contents on the screen. Users can click on the title of the movie entering to the movie description page.

Appendix A-2 reveals the initialisation stage of the movie description page. In this example, the application was trying to get the information related to a movie called "Dora and the Lost City of Gold" which will be on showing one week later. **Appendix A-2** shows that the website URL has been changed to end with a serial of numbers which is the movieID. In our case, the movieID is 499701. The web application will always send a **GET** method HTTP request to the server when initializing the movie description page. The HTTP request contains a parameter with the movieID to fetching the specific movie data. The following procedure is similar to the procedure of fetching the movie list which explained above. Once the web browser receives the movies data, it shows the fully detailed information about the movie on the screen.

Appendix A-3 shows the movie description page fetched the movie data successfully. However, it is not complete yet. The web application will extract the movie name from the HTTP response in the previous procedure. It then fires another **GET** method HTTP request to the server with a keyword query contains the movie name to get the news related to the movies. The server application will then fetch the news related to the keyword by sending an HTTP request to NEWSAPI server. After, the server returns the news data to the client when the response came back from NEWSAPI server. Finally, it shows the movie news on web content as **Appendix A-4** shown.

Getting Movie Tweets

Appendix B-1 shows the searching page of the web application. In this example, it shows an instruction trying to get the spoiler and discussion related to a movie called “X-Men Apocalypse”. Users can search for movies by entering the relevant keyword or the name of the movie. After users inserted the keyword and clicked on the search button, the application will then send a **GET** method HTTP request to the cloud server for getting relevant movies. Then, the server application receives the HTTP request and send another **GET** method HTTP request to fetch movie data from The Movie Database (TMDB) API. Once the response data from TMDB API is received by the server, it will transfer the data back to the client. Finally, the web browser receives the movies data and updates the contents on the screen. Users can click on the title of the movie entering to the movie description page.

Appendix B-2 reveals the movie description page loaded successfully. As the previous section explained that the web application will always send a **GET** method HTTP request to the server when initializing the movie description page. The HTTP request contains a parameter with the movieID to fetch the specific movie information. Once the web browser receives the data, it shows the fully detailed information about the movie on the screen. Then, it extracts the movie name sending two **GET** method HTTP requests to getting the news and tweets related to the movies. The server application will then fetch the tweets related to the movie name by sending an HTTP request to the Tweeter API server. Finally, the server returns the tweets data to the client when the response came back from Tweeter API server.

The tweets related to the movie displayed in **Appendix B-3**. It is evident that we can see how people discuss this movie. It is also interesting to note that there is one spoiler provided by a user which denoted by a hashtag. However, it is not always able to find some spoiler tweets.

Technical Description

Client Side

The user interface of the project is built with React. React is a JavaScript library helping developers to create interactives UI Components. It is also one of the widely used tools to build web application at the moment. Because it provides a lightweight and manageable method for developers to make web applications. Modern web applications are heavily relying on communicating with APIs or servers. It is difficult to display the correct UI components according to different circumstances. React provides a library called Context API for managing the state of the application. We can easily manage the state of the application to render the correct UI component effectively. Another difficulty of building interactive user interfaces is to determine the accurate time of sending HTTP request. React Component Lifecycle allows developers to handle the changes at different phases of a UI component's life easily. It makes the development process more effectively and robustness.

The proposed application has integration of two distinct data source, determining the accurate time to fetch data from the server becomes a challenge of building the application. React component Lifecycle makes it very easy to address this challenge. Two common lifecycle methods are being used in this project, it includes ComponentDidMount and ComponentWillUnmount.

ComponentDidMount method is heavily used in this project, it allows the application execute code before the UI component shows up. In this case, it is always suitable to send HTTP requests to fetch data from the server before the UI components initialize. Without knowing the movie name, it is impossible to fetch the news or tweets related to the movie. When the movie description page initializes, the application automatically sends a GET method HTTP request for fetching movie data before the UI shows up. Once, the movie data is received, the application extracts the movie name from the data, and pass it to the tweets and news section component to fetch the news and tweets data. In the end, all the data received, and it will be rendered to the page.

In contrast, ComponentWillUnmount is used to clean the state of the application when the component will be destroyed to maintain information accuracy. All the dynamic content showing in the application is read from the global state of the application. If the state did not clean after the users leave the page, it might lead to information inconsistency when they enter to a new page.

Rendering the correct UI components depends on different circumstances is also important for the application. React allows us to write JavaScript code for checking the state of the application whether the movie data received or not. It checks the state stored in the Context API. It then renders the spinner to indicate loading data when movie list data is undefined at the initialisation state. After, it renders the data to the content when data is received from the server.

Server Side

API routes	Method	Params	Query	Usage
api/movies/:id	GET	Id	-	Getting specific movie information by the movieID (tmdb-MovieID)
api/movies/?	GET	-	lang, ranking_type	Getting a list of movies by ranking type and languages lang: ['en', 'zh', 'fr', 'de', 'it', 'jp'] ranking_type: [“now_playing”, “popular”, “top_rated”, “upcoming”]
api/news/	GET	-	keyword	Getting news which related to the keyword value
api/tweets/	GET	-	query	Getting tweets which related to the query value
api/search/movie/	GET	-	name	Searching movies by name

Table1. Server API Routes

The Server-side application is built with NodeJS. ExpressJS is also used to establish the routes of the RESTful API for the proposed application. The server-side application is configured to accept incoming JSON data. The application will listen on the port 5000. But the port number will be changed if the server set the port of the environment variable to another number.

Four different kinds of API services are provided by the server application. It includes movies, news, tweets and search services. All the route of the API services starts with “api/” and followed by the services name. All the services only provide GET method routes to the user at the moment. Because the application does not have any functionality that related to update, insert and delete operation. The usage of all the routes is listed above.

Express validator is assembled to the application as the middleware. It can ensure that the required parameters or query of the route are provided by checking the incoming data whether to match the pre-defined requirement or not. When the client request does not fulfil the pre-defined requirement, it will send the error message back to the client and stop the server handling the service further. The error message is a JSON format data which contains the error status code and the error message in String. It can dramatically reduce service failures. The server will also return an error message in JSON format to the client if there is any problem when executing the code inside of the try-catch block. For example, errors in fetching data from the third-party API. If users try to fetch news data when NEWSAPI server is in maintenance, the server will send an error message to the client telling them why the request is failed.

Finally, the others unused routes will be served to return the website.

Docker

In this project, Docker is adopted to build and deploy the application. Docker makes the node application easier to create, deploy, and run it by using containers. It is significant to note that hosting an application and running on a different machine is never a simple task. It might occur some unexpectable errors when runs or install the application on another operating system besides the developer's machine. Docker is one of the solutions to this problem. It allows developers to run one or more containerized applications within one or more virtualized Linux instances. It means that we only need to ensure the application running successfully on Linux, then we can imageries the applications and run it as a container on another operation system by Docker.

Docker also simplified the deployment configuration of the node application. In the traditional approach, it required to take too many procedures for deploying the application. Firstly, we need to pull all the files back from the repository. Then, running the NPM command to download the thirty party modules manually. After, running the pre-written script in package.json to build the production version of the website. Finally, turn on the server-side application to serve the services. One of the significant features of Docker is the creation of Dockerfiles. Dockerfiles allows developers declaratively specify what instructions need to be finished for deploying the application and Docker will ensure new container gets created according to the specifications. It not only brings rapid application deployment to software development workflow but also makes sharing of the application easily. The owners of the application are allowed to push their docker images on Dockerhub. The other people can pull the image down and run it directly on their machine via Docker.

Appendix C-1 shows the docker file of the project. The docker file is used to create the docker image for the cloud server runs later. Node is used as the base images of the application because the application is heavily built with node. Then, docker set the working directory as a folder named as app. All the docker instruction will only work on this folder later on. After that docker will copy all the files in the root folder into the working directory. Then, it downloads all the required modules on both side application. It builds the production version of the website when the required module download completed. Docker then exposes the port 5000 to make the container becomes accessible later on from outside. Finally, docker starts the server-side application to host the API service and website. The image will be uploaded to the Dockerhub for public downloading to use the application on another machine after building it successfully.

Appendix C-2 shows the security setting of the amazon web service instance. Due to the security reason, AWS EC2 requires developers to configure the external port by themselves. Therefore, 5000 port of the server is configured to allow accessing by any public IP address.

Appendix C-3 reveals the docker configuration on the cloud server. Firstly, it downloads the docker image built in the previous section from Dockerhub. Then, docker runs the images in the background and connects the exposed port of the images to the server port. Finally, Users can access the API service and website by accessing the server domain name ended with the configured port number which is 5000.

Testing

In the server-side application, it is implemented unit testing to ensure that the server application will return the expected outcome on each route. There are five-unit testing files in the server-side project. Each file is responsible for checking each API routes which listed in Technical Description section. There are two third-party libraries used to implement unit tests for the project. It includes Jest and Supertest. Jest is a delightful Testing Framework for testing JavaScript code, while Supertest is a library to simulating node HTTP servers for testing purpose. The main idea for the unit testing is that test a single route a time by simulating the node application. It then sends HTTP requests to the simulated instance to check whether response the expected data or not.

Appendix D shows the result of the unit testing. Clearly, the server application can response the expected data from each API routes successfully. It is also able to send the front-end website to the client by accessing any unused routes. However, unit testing is only able to show successful use cases. It did not show what will happens if the user does not provide the required parameters or queries within the HTTP request package. As mentioned in the server-side section of the Technical Description, Express validator is implemented to the application as the middleware to prevent these cases happen. It will send an error message to users in JSON format with 400 status code notifying the user to provide the required parameters or queries for further actions. Both protections can dramatically increase the robustness of the application.

Possible extensions

The Tweets Search function is not work as perfect as I expected. The API and the third-party library do not compatible with hashtags search. TweeterAPI only allows developer search tweets by the content. It is not an effective way to help movie fans to find out the spoiler related to the movie.

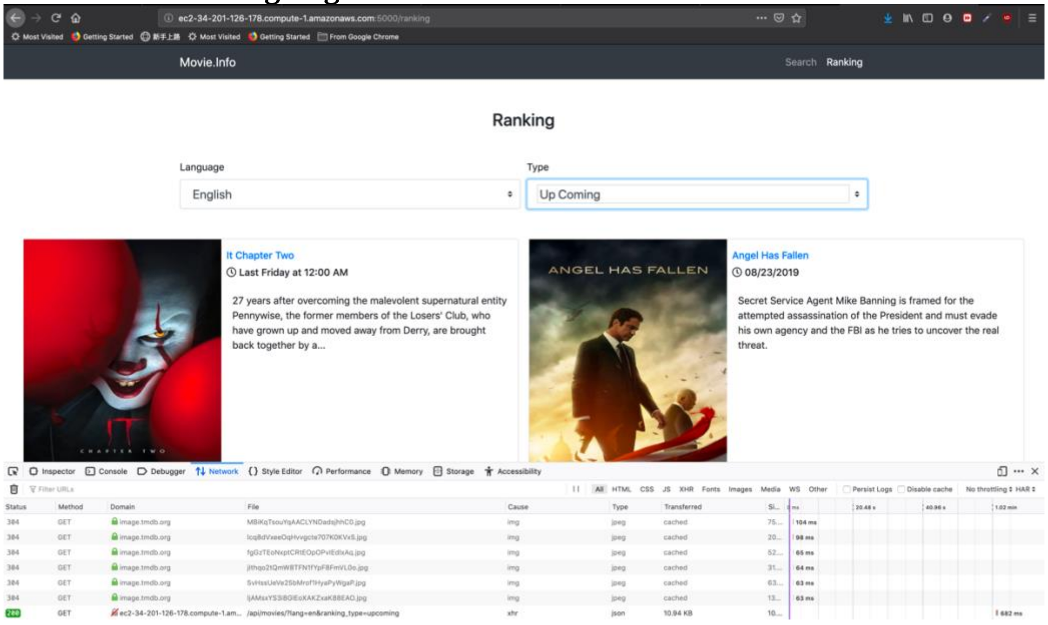
Adding pagination for news, movies and tweets search API will be the next phase of the development. Users are limited to search a few numbers of this information at the moment. It is not difficult to make the API service compatible with pagination. However, it takes sufficient time to design an algorithm to calculate the pagination which allowed for user select.

Joining the movie discussion in Tweeter is also a feature that can be developed in the next phase. Currently, the application only allows user to check the tweets. However, this feature also takes a sufficient time for developing login system.

Natural Language Processing is also required to be adopted to the project. Some of the movie names are not outstanding to be found by NEWSAPI and TweeterAPI. It might lead to display unrelated information to the user. Natural Language Processing might able to improve the accuracy of searching for information related to a specific movie.

Appendix

Appendix A - 1 – Ranking Page



ec2-34-201-126-178.compute-1.amazonaws.com:5000/ranking

Most Visited · Getting Started · 新手上路 · Most Visited · Getting Started · From Google Chrome

Movie.Info — Search · Ranking

Ranking

Language: English

Type: Up Coming

It Chapter Two
Last Friday at 12:00 AM
27 years after overcoming the malevolent supernatural entity Pennywise, the former members of the Losers' Club, who have grown up and moved away from Derry, are brought back together by a...

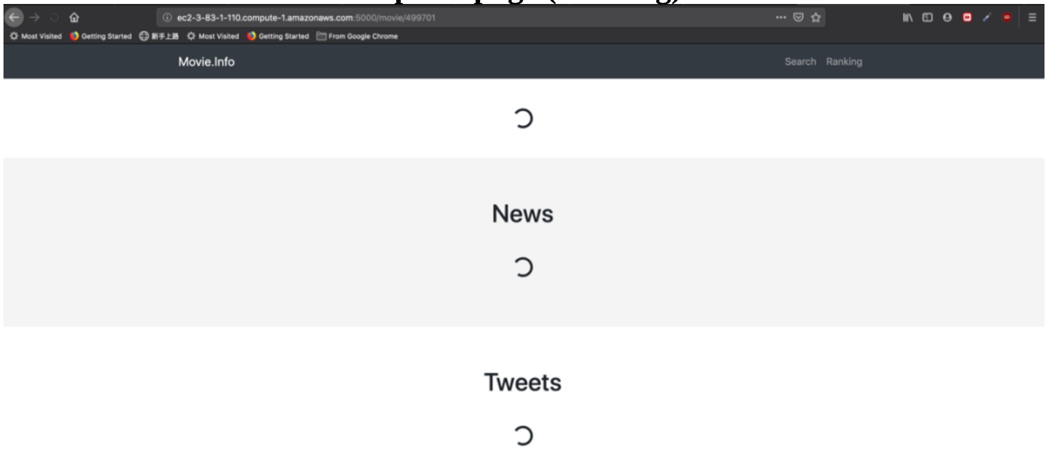
Angel Has Fallen
08/23/2019
Secret Service Agent Mike Banning is framed for the attempted assassination of the President and must evade his own agency and the FBI as he tries to uncover the real threat.

Inspector · Console · Debugger · Network · Style Editor · Performance · Memory · Storage · Accessibility

Filter URLs · All · HTML · CSS · JS · XHR · Fonts · Images · Media · WS · Other · Persist Logs · Disable cache · No throttling · HAR

Status	Method	Domain	File	Cause	Type	Transferred	Si...	
304	GET	image.tmdb.org	MBiKqTsouYqAACLYNDadsjhhC0.jpg	img	jpeg	cached	75...	104 ms
304	GET	image.tmdb.org	lcq8dVxeeOqHvvgcte707K0KVxS.jpg	img	jpeg	cached	20...	98 ms
304	GET	image.tmdb.org	fgGzTEoNxptCRtEOpOPvlEdlxAq.jpg	img	jpeg	cached	52...	65 ms
304	GET	image.tmdb.org	jlthqo2tQmW8TFN1fYpF8FmVL0o.jpg	img	jpeg	cached	31...	64 ms
304	GET	image.tmdb.org	SvHssUeVe25bMrof1HyaPyWgaP.jpg	img	jpeg	cached	63...	63 ms
304	GET	image.tmdb.org	ljAMsxYSSiBGIEoXAKZxaK88EAO.jpg	img	jpeg	cached	13...	63 ms
200	GET	ec2-34-201-126-178.compute-1.am...	/api/movies/?lang=en&ranking_type=upcoming	xhr	json	10.94 KB	10...	682 ms

Appendix A - 2 – The movie description page (Loading)



ec2-3-83-1-110.compute-1.amazonaws.com:5000/movie/499701

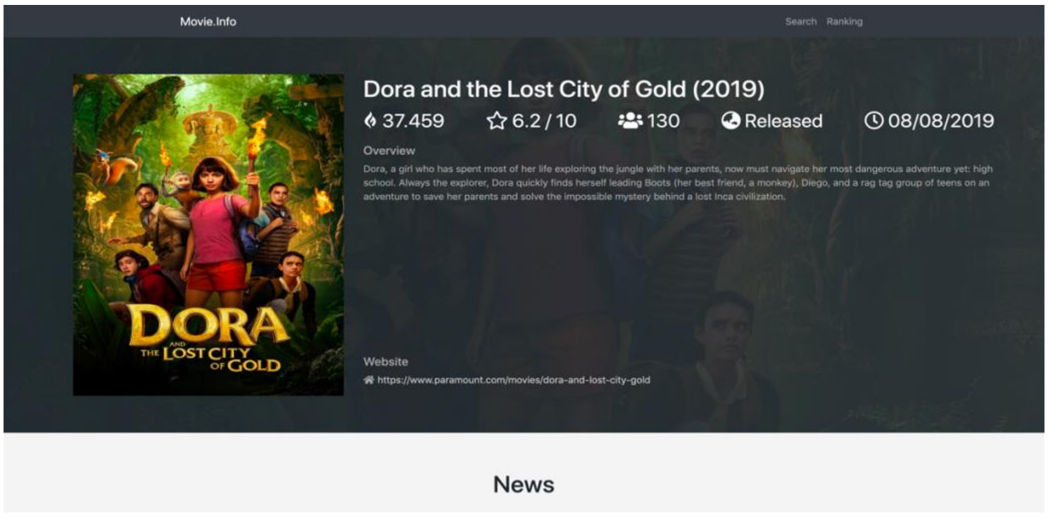
Most Visited · Getting Started · 新手上路 · Most Visited · Getting Started · From Google Chrome

Movie.Info — Search · Ranking

News

Tweets

Appendix A - 3 – The movie description page (Loaded Successfully)



Movie.Info — Search · Ranking

Dora and the Lost City of Gold (2019)

37.459 · 6.2 / 10 · 130 · Released · 08/08/2019

Overview

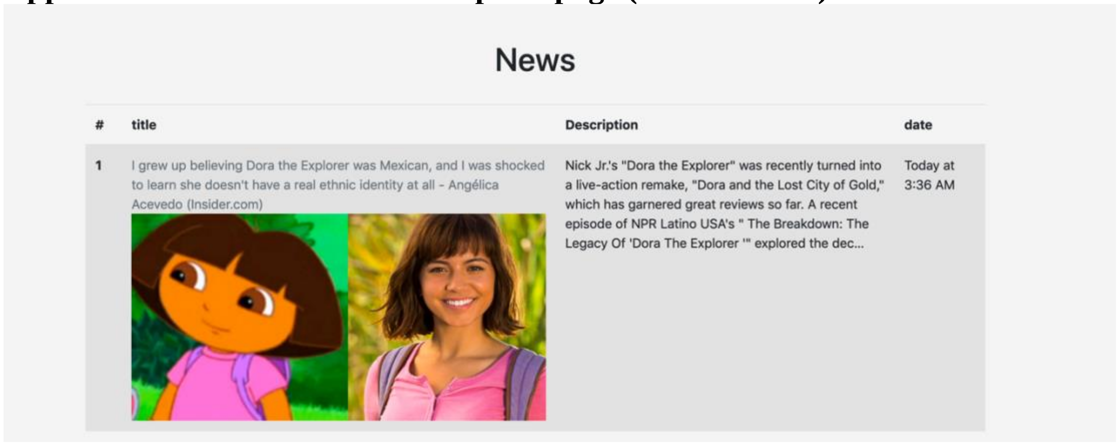
Dora, a girl who has spent most of her life exploring the jungle with her parents, now must navigate her most dangerous adventure yet: high school. Always the explorer, Dora quickly finds herself leading Boots (her best friend, a monkey), Diego, and a rag tag group of teens on an adventure to save her parents and solve the impossible mystery behind a lost Inca civilization.

Website

https://www.paramount.com/movies/dora-and-lost-city-gold

News

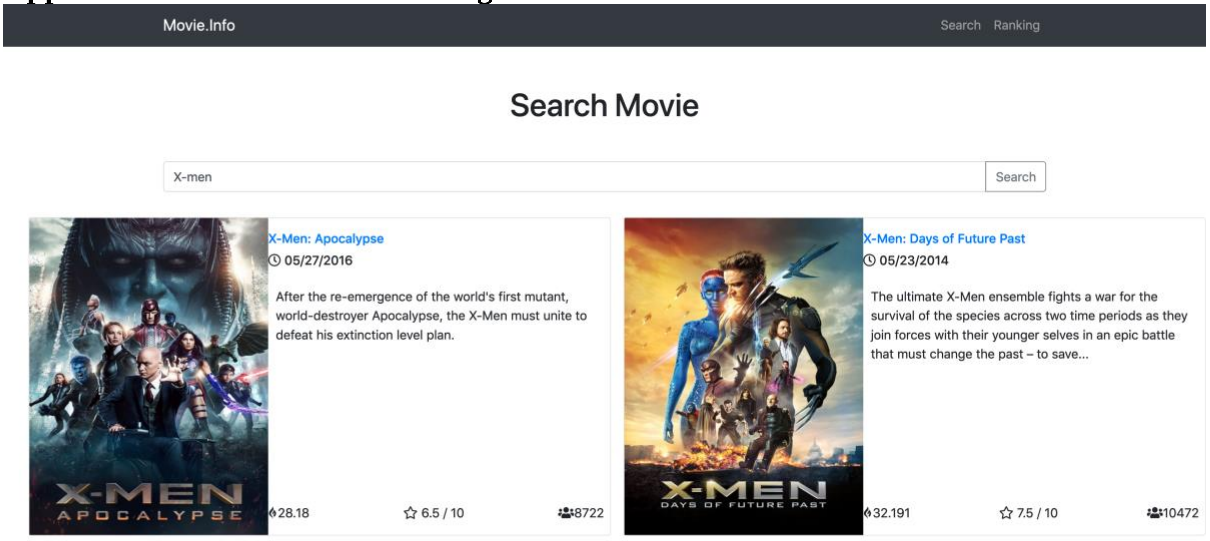
Appendix A - 4 – The movie description page (News Section)



News

#	title	Description	date
1	I grew up believing Dora the Explorer was Mexican, and I was shocked to learn she doesn't have a real ethnic identity at all - Angélica Acevedo (Insider.com)	Nick Jr.'s "Dora the Explorer" was recently turned into a live-action remake, "Dora and the Lost City of Gold," which has garnered great reviews so far. A recent episode of NPR Latino USA's " The Breakdown: The Legacy Of 'Dora The Explorer' " explored the dec...	Today at 3:36 AM

Appendix B – 1 – Search Movie Page



Movie.Info Search Ranking

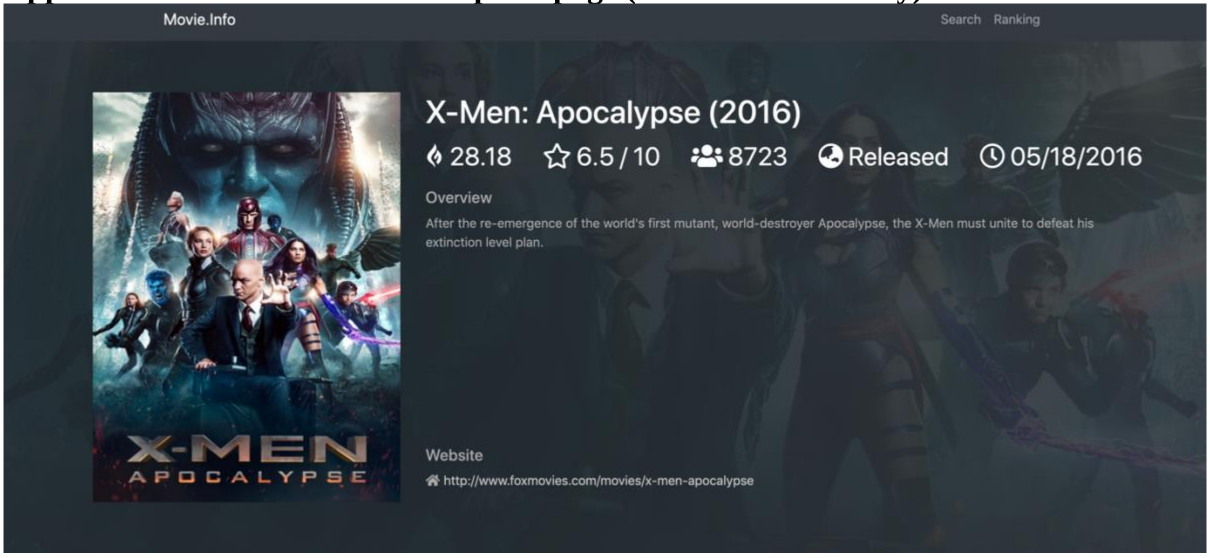
Search Movie

X-men [Search]

X-Men: Apocalypse
05/27/2016
After the re-emergence of the world's first mutant, world-destroyer Apocalypse, the X-Men must unite to defeat his extinction level plan.
28.18 6.5 / 10 8722

X-Men: Days of Future Past
05/23/2014
The ultimate X-Men ensemble fights a war for the survival of the species across two time periods as they join forces with their younger selves in an epic battle that must change the past – to save...
32.191 7.5 / 10 10472

Appendix B – 2 - The movie description page (Loaded Successfully)



Movie.Info Search Ranking

X-Men: Apocalypse (2016)

28.18 6.5 / 10 8723 Released 05/18/2016

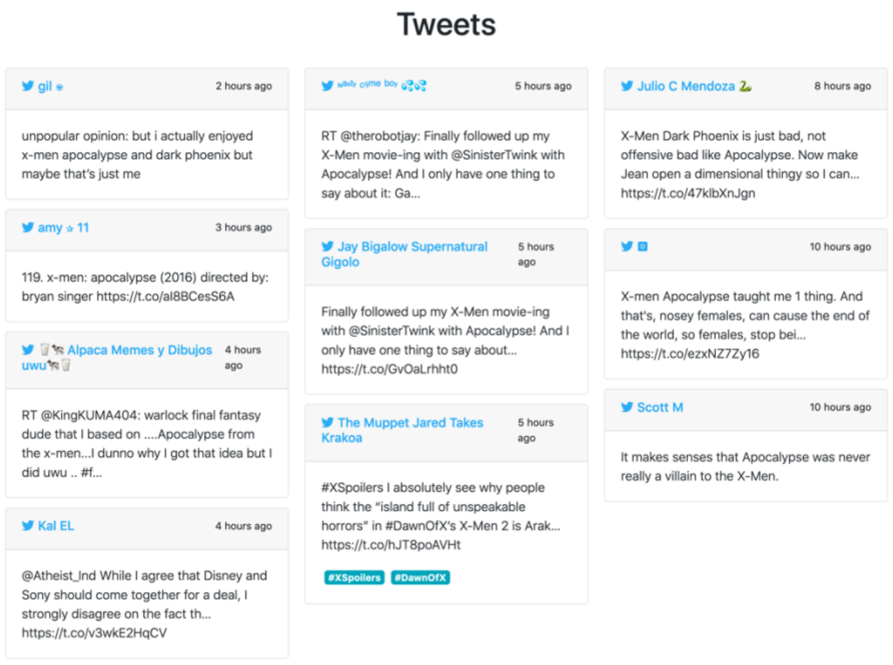
Overview

After the re-emergence of the world's first mutant, world-destroyer Apocalypse, the X-Men must unite to defeat his extinction level plan.

Website

http://www.foxmovies.com/movies/x-men-apocalypse

Appendix B – 3 – Tweets Section of the movie description page



Tweets

gil 2 hours ago
unpopular opinion: but i actually enjoyed x-men apocalypse and dark phoenix but maybe that's just me

amy ⭑ 11 3 hours ago
119. x-men: apocalypse (2016) directed by: bryan singer https://t.co/al8BCesS6A

Alpaca Memes y Dibujos uwu 4 hours ago
RT @KingKUMA404: warlock final fantasy dude that I based on ....Apocalypse from the x-men...I dunno why I got that idea but I did uwu .. #f...

Kal EL 4 hours ago
@Atheist_Ind While I agree that Disney and Sony should come together for a deal, I strongly disagree on the fact th... https://t.co/v3wkE2HqCV

nasty crime boy 5 hours ago
RT @therobotjay: Finally followed up my X-Men movie-ing with @SinisterTwink with Apocalypse! And I only have one thing to say about it: Ga...

Jay Bigalow Supernatural Gigolo 5 hours ago
Finally followed up my X-Men movie-ing with @SinisterTwink with Apocalypse! And I only have one thing to say about... https://t.co/GvOaLrhht0

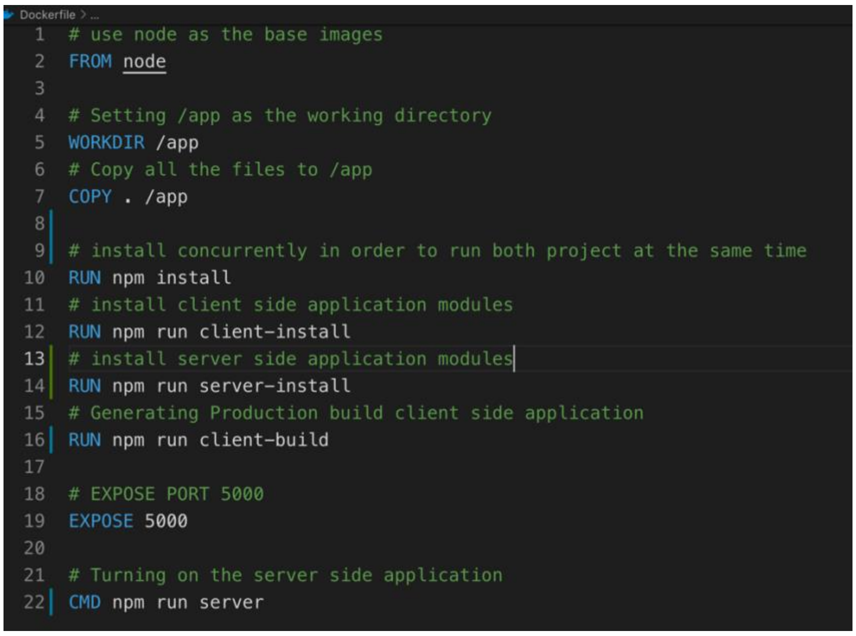
The Muppet Jared Takes Krakoa 5 hours ago
#XSpoilers I absolutely see why people think the "island full of unspeakable horrors" in #DawnOfX's X-Men 2 is Arak... https://t.co/hJT8poAVHt
#XSpoilers #DawnOfX

Julio C Mendoza 8 hours ago
X-Men Dark Phoenix is just bad, not offensive bad like Apocalypse. Now make Jean open a dimensional thingy so I can... https://t.co/47klbXnJgn

10 hours ago
X-men Apocalypse taught me 1 thing. And that's, nosey females, can cause the end of the world, so females, stop bei... https://t.co/ezxNZ7Zy16

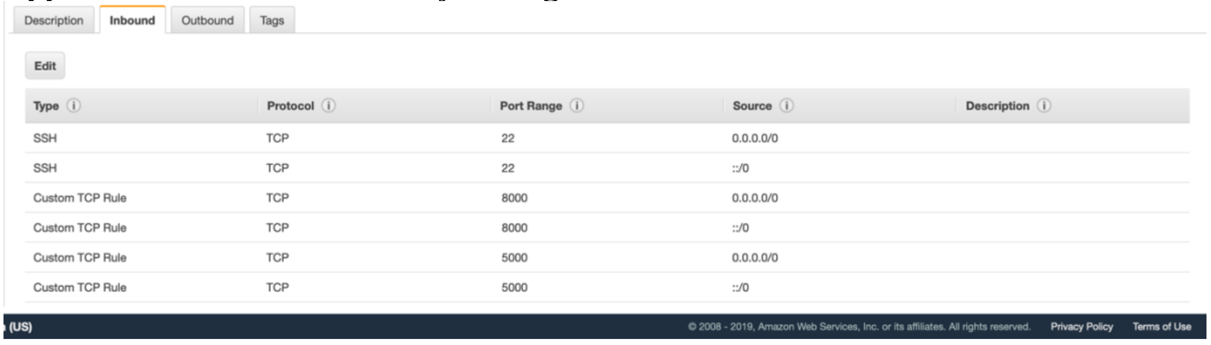
Scott M 10 hours ago
It makes senses that Apocalypse was never really a villain to the X-Men.

Appendix C - 1 – Dockerfile Content



```
# use node as the base images
FROM node

# Setting /app as the working directory
WORKDIR /app
# Copy all the files to /app
COPY . /app

# install concurrently in order to run both project at the same time
RUN npm install
# install client side application modules
RUN npm run client-install
# install server side application modules
RUN npm run server-install
# Generating Production build client side application
RUN npm run client-build

# EXPOSE PORT 5000
EXPOSE 5000

# Turning on the server side application
CMD npm run server
```

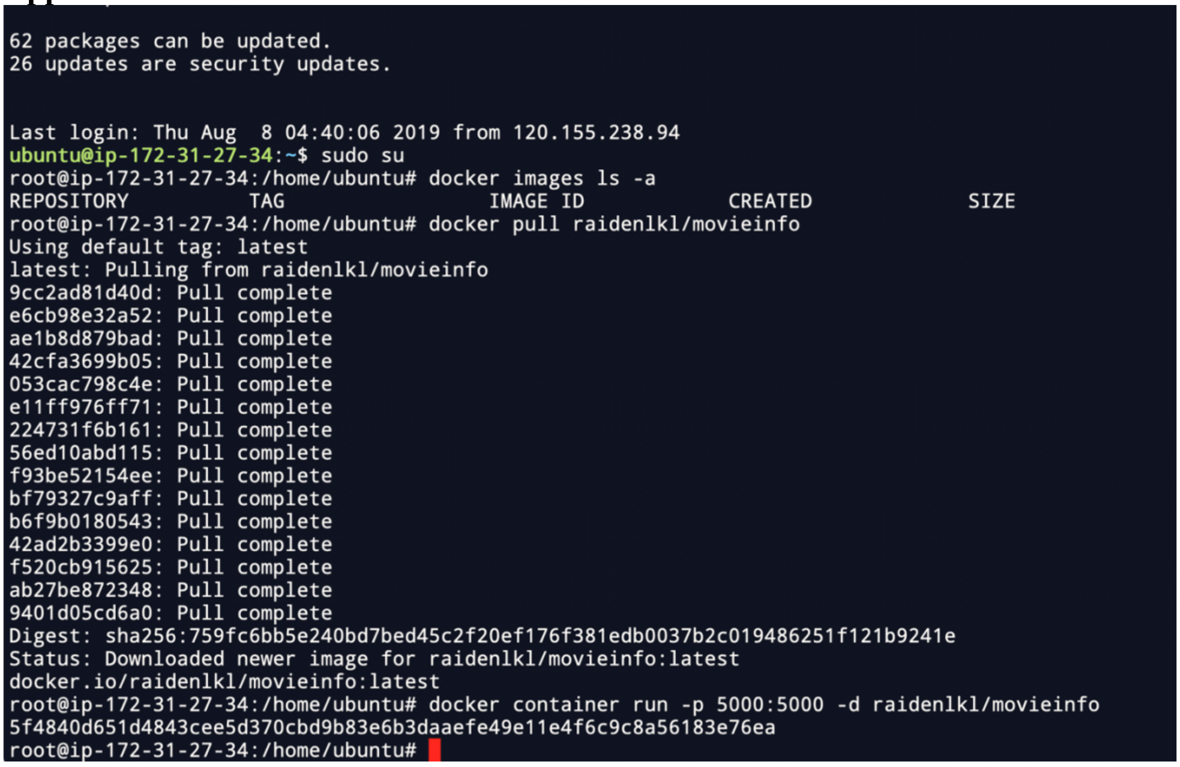
Appendix C - 2 – Server Security Setting



Type	Protocol	Port Range	Source	Description
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	::/0	
Custom TCP Rule	TCP	8000	0.0.0.0/0	
Custom TCP Rule	TCP	8000	::/0	
Custom TCP Rule	TCP	5000	0.0.0.0/0	
Custom TCP Rule	TCP	5000	::/0	

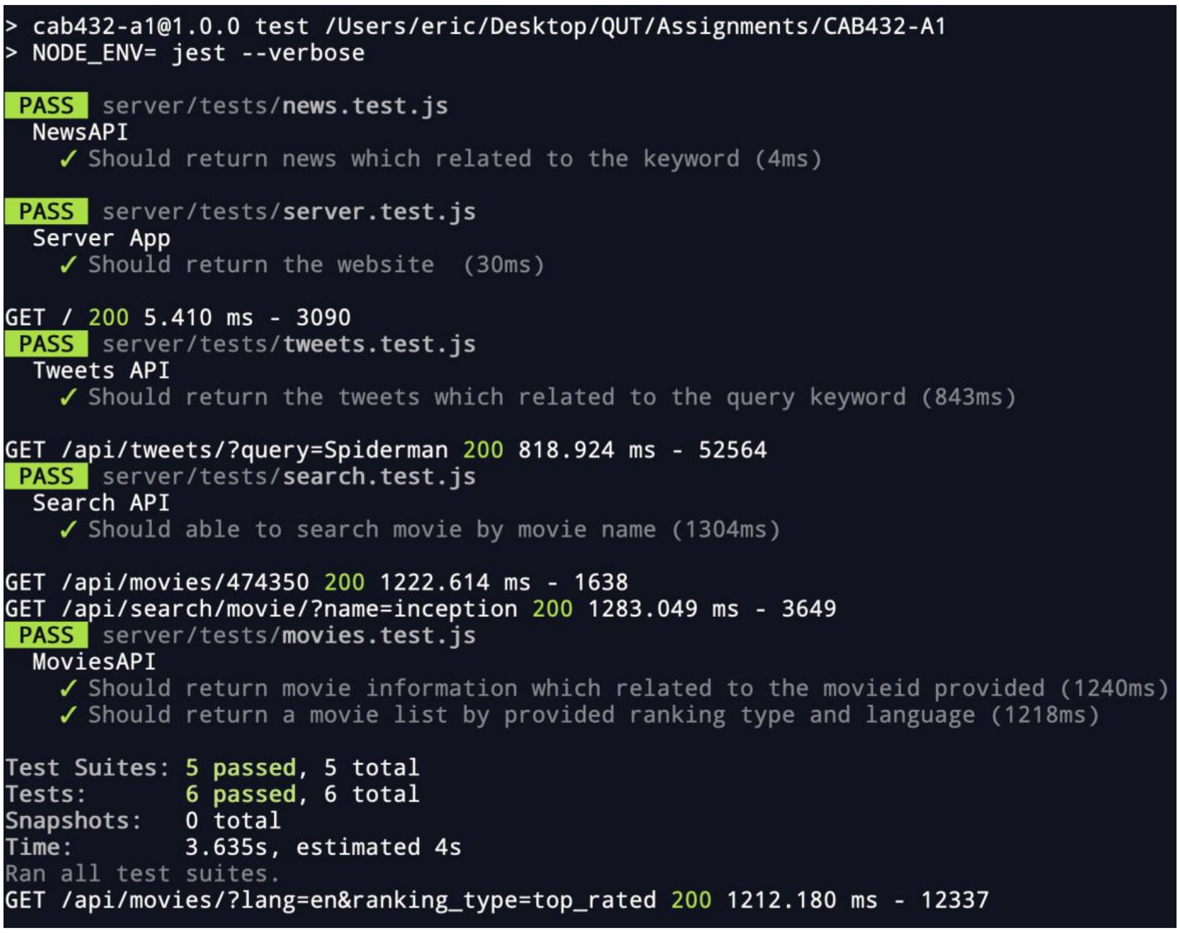
(US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Appendix C - 3 – Docker command on cloud server



```
62 packages can be updated.
26 updates are security updates.


Last login: Thu Aug  8 04:40:06 2019 from 120.155.238.94
ubuntu@ip-172-31-27-34:~$ sudo su
root@ip-172-31-27-34:/home/ubuntu# docker images ls -a
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root@ip-172-31-27-34:/home/ubuntu# docker pull raidenlkl/movieinfo
Using default tag: latest
latest: Pulling from raidenlkl/movieinfo
9cc2ad81d40d: Pull complete
e6cb98e32a52: Pull complete
ae1b8d879bad: Pull complete
42cfa3699b05: Pull complete
053cac798c4e: Pull complete
e11ff976ff71: Pull complete
224731f6b161: Pull complete
56ed10abd115: Pull complete
f93be52154ee: Pull complete
bf79327c9aff: Pull complete
b6f9b0180543: Pull complete
42ad2b3399e0: Pull complete
f520cb915625: Pull complete
ab27be872348: Pull complete
9401d05cd6a0: Pull complete
Digest: sha256:759fc6bb5e240bd7bed45c2f20ef176f381edb0037b2c019486251f121b9241e
Status: Downloaded newer image for raidenlkl/movieinfo:latest
docker.io/raidenlkl/movieinfo:latest
root@ip-172-31-27-34:/home/ubuntu# docker container run -p 5000:5000 -d raidenlkl/movieinfo
5f4840d651d4843cee5d370cbd9b83e6b3daaefe49e11e4f6c9c8a56183e76ea
root@ip-172-31-27-34:/home/ubuntu#
```

Appendix D – Unit Test Case (Server-side)



```
> cab432-a1@1.0.0 test /Users/eric/Desktop/QUT/Assignments/CAB432-A1
> NODE_ENV= jest --verbose

 PASS  server/tests/news.test.js
  NewsAPI
    ✓ Should return news which related to the keyword (4ms)

 PASS  server/tests/server.test.js
  Server App
    ✓ Should return the website  (30ms)

GET / 200 5.410 ms - 3090
 PASS  server/tests/tweets.test.js
  Tweets API
    ✓ Should return the tweets which related to the query keyword (843ms)

GET /api/tweets/?query=Spiderman 200 818.924 ms - 52564
 PASS  server/tests/search.test.js
  Search API
    ✓ Should able to search movie by movie name (1304ms)

GET /api/movies/474350 200 1222.614 ms - 1638
GET /api/search/movie/?name=inception 200 1283.049 ms - 3649
 PASS  server/tests/movies.test.js
  MoviesAPI
    ✓ Should return movie information which related to the movieid provided (1240ms)
    ✓ Should return a movie list by provided ranking type and language (1218ms)

Test Suites: 5 passed, 5 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.635s, estimated 4s
Ran all test suites.
GET /api/movies/?lang=en&ranking_type=top_rated 200 1212.180 ms - 12337
```