

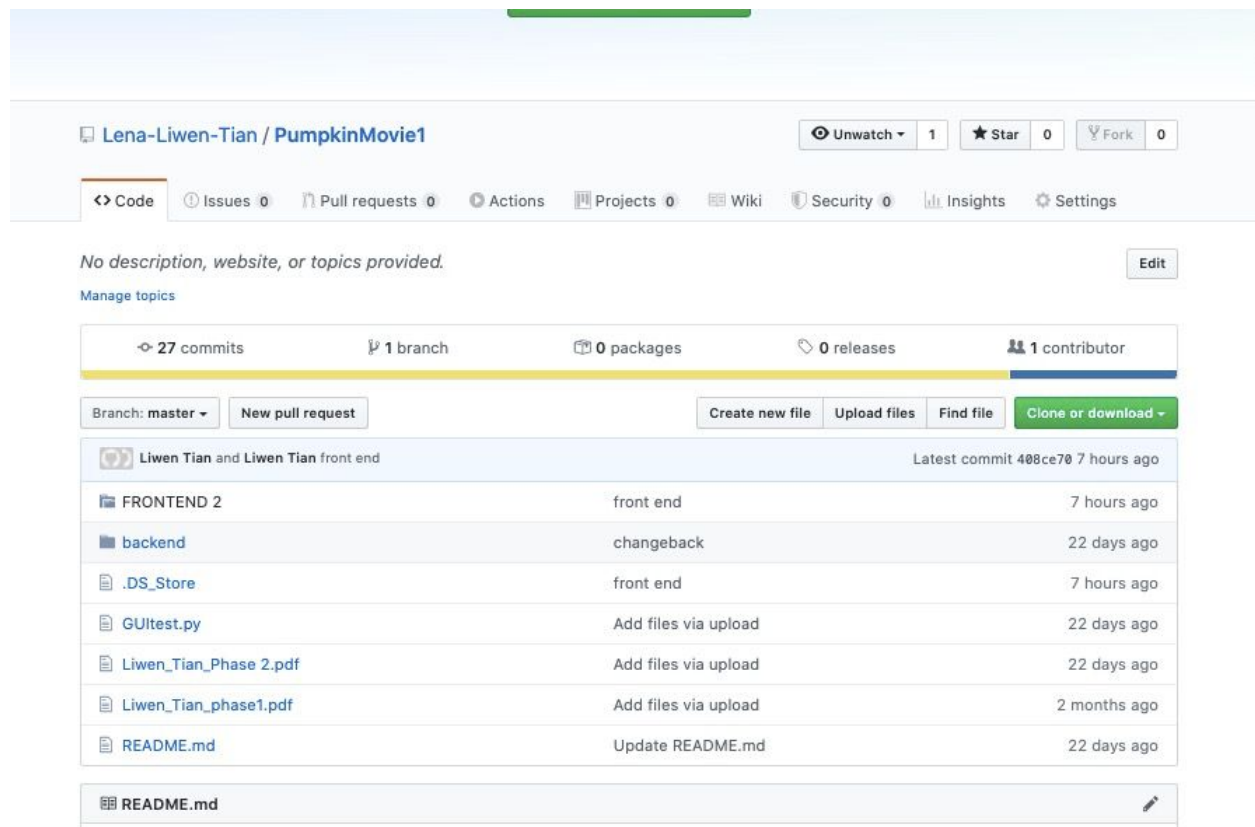
Technical Report for Phase Three

Website link

<https://master.d1n0ec8g2eialu.amplifyapp.com/>

Github link

<https://github.com/Lena-Liwen-Tian/PumpkinMovie1>



Team Information

Name: Liwen Tian

Email: liwentian1103@gmail.com

Github username: Lena-Liwen-Tian

Canvas group: group 8

Project name: Pumpkin Movie

Project Description

My project name is Pumpkin Movie, it provides users a better way to buy their ideal movie tickets online. On my website, users can find movie information in details, eg. movie time, movie genres, movie ratings. Users can also get the relevant information about all theatres in the United States, eg. location, phone numbers. Besides, the show time of different movies at

different theatres are also listed on the website for users to choose from. Pumpkin Movie allows users to combine information about movies, theatres and show time ahead of time, so that they can buy an ideal movie ticket in the end. The target users are young people who are looking for an easier way to buy an ideal ticket in order to get a better viewing experience in the theatre.

User Stories

In Phase 3, I have 12 new user stories. The first 5 are created by the client team. The rest of them are created by myself. They are shown as follows.

1. As a user, I want to buy tickets online, it makes me feel convenient.

I estimated that I would spend 5 hours doing it since I did not complete similar things before. But I found good tutorials online and got it done within 2 hours. I used react - stripe to do it. The assumption for this user story is young users prefer to do a lot of things online including buying movie tickets.

2. As a user, I want to search movies by name, so that I can find the movie I want quickly.

I estimated that I would spend one day on it. Actually, I spent two days doing it. I used Python for creating search box before and this is my first time to use React. The assumption for this is that users usually use movie names to find the movie they like.

3. As a user, I want to filter movies by countries, so I can see foreign movies

I estimated that I would spend about 4 hours on it and I spent about 2 hours. I found similar frame work on React and did it in my own way. The assumption for this is that users from different countries want to find the movie spoken in their own language.

4. As a user, I want to search movies by actor name so that I can find the movie played by my favorite actors.

I estimated that I would spend about 1 hour on this, and I did spend about one hour on it. I created a filter component in React which accepts different parameters. The assumption is that when picking up a movie to watch, users care about the actors who play it.

5.As a user, I want to search the cinema by name, so I can find the movie in a specific cinema quickly.

I estimated that I would spend about 2 hours on this. I spent about half an hour. Like what I did for the filter, I also created a search component in React and it makes everything easier. The assumption is that users sometimes memorize the name of the cinema giving them good watching experience.

6.As a user, I want to filter the most nearby cinema, so that I do not spend a long time on the way to see the movie.

I estimated that I would spend about 10 minutes on it. Actually,I spent 20 minutes. The assumption is that users sometimes do not want to spend too much time on the way. They want to save time for watching the movie.

7. As a user, I want to filter the showtime by time, so I can find the movie shown at night.

I estimated that I would spend about 10 minutes one it. I spent 30 minutes on it in total. Since the format of time I stored in the database is string, I have to convert them into integers before sorting them. The assumption is that some users have their time preference. Some users may want to watch movies at night while some may not want to go out too late at night.

8.As a user, I want to search the cinema by zip code because I am more familiar with zip code rather than theatre name.

I thought I would spend 20 minutes on it, but I actually spent 1 hour on it. Searching by zip code and search by theatre name are two different methods. I split them rather than combine them together. The assumption is that zip code is easier for users to memorize compared to theatre names. So users may need the choice to search by zip code.

9.As a user, I want to sort the movies by ratings so that I can find the most popular movies.

I thought I would spend 20 minutes on it and I think it did take me about 20 minutes. I did it together with other sorting methods. The assumption is that users prefer to see popular movies so that they need a convenient way to find it.

10.As a user, I want to sort the movie by year, so that I can find the most recent movies.

I estimated that I need 20 minutes on it and I spent 10 minutes on it in total. This is easy, since years are all integers so that they are easy to be sorted. The assumption is that movie fans frequently check the latest movies in order to catch the movie trend.

11.As a user, I want to sort the showtime by time so that I can find the earliest movie shown today quickly.

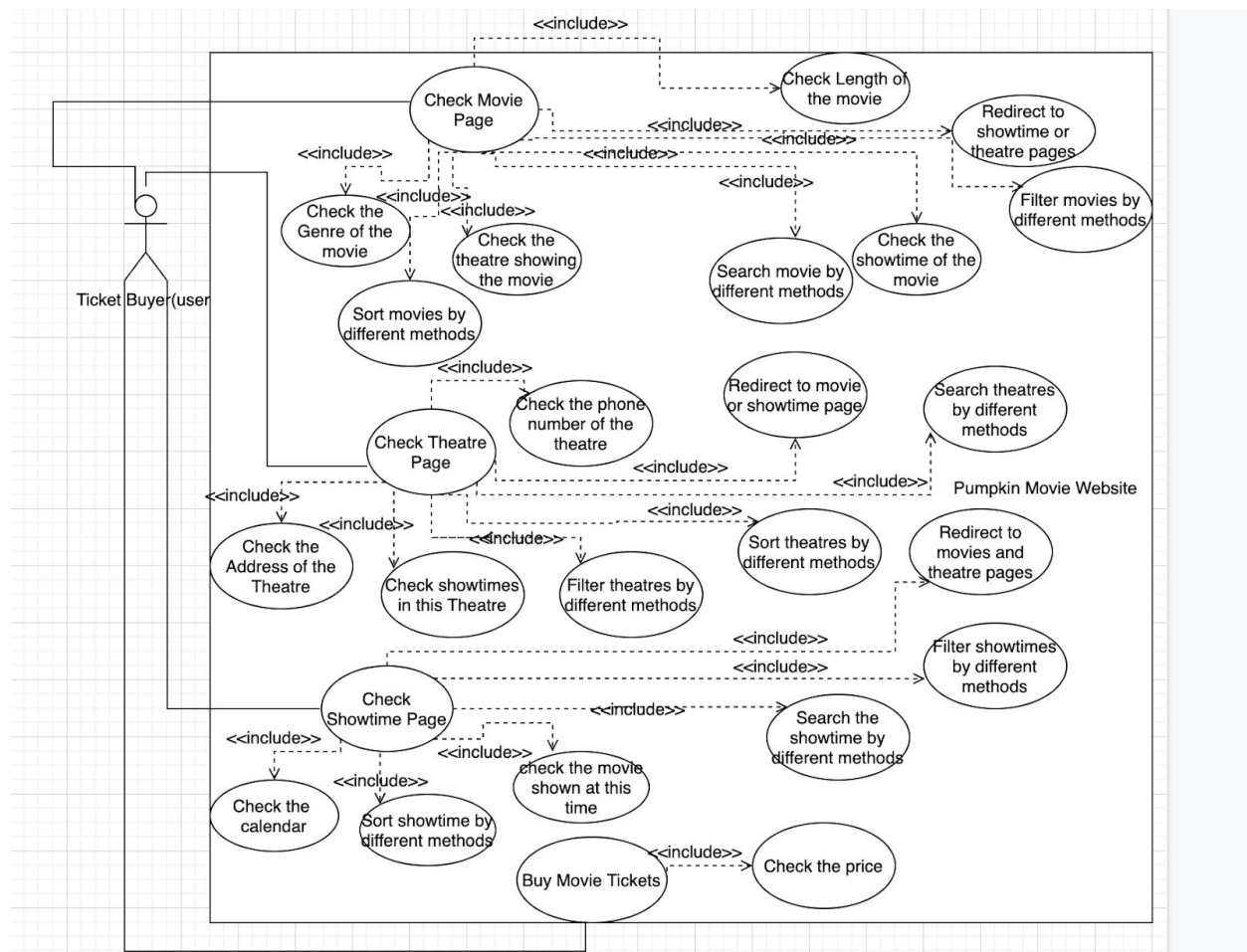
I estimated that I would spend 30 minutes on it. This is the last sort method I did in this Phase, so it is the easiest one. I spent 10 minutes one it. The assumption is that users have the need to see the time plan of movies to pick up their favorite one.

12.As a user, I want to use my credit card to buy the ticket online because I am used to doing that.

This is an extra user story I created for myself. I estimated I spent one hour on it and actually 30 minutes. The assumption is that users paid with their credit card the most of time when buying something online, so they are used to it.

Use Case Diagram

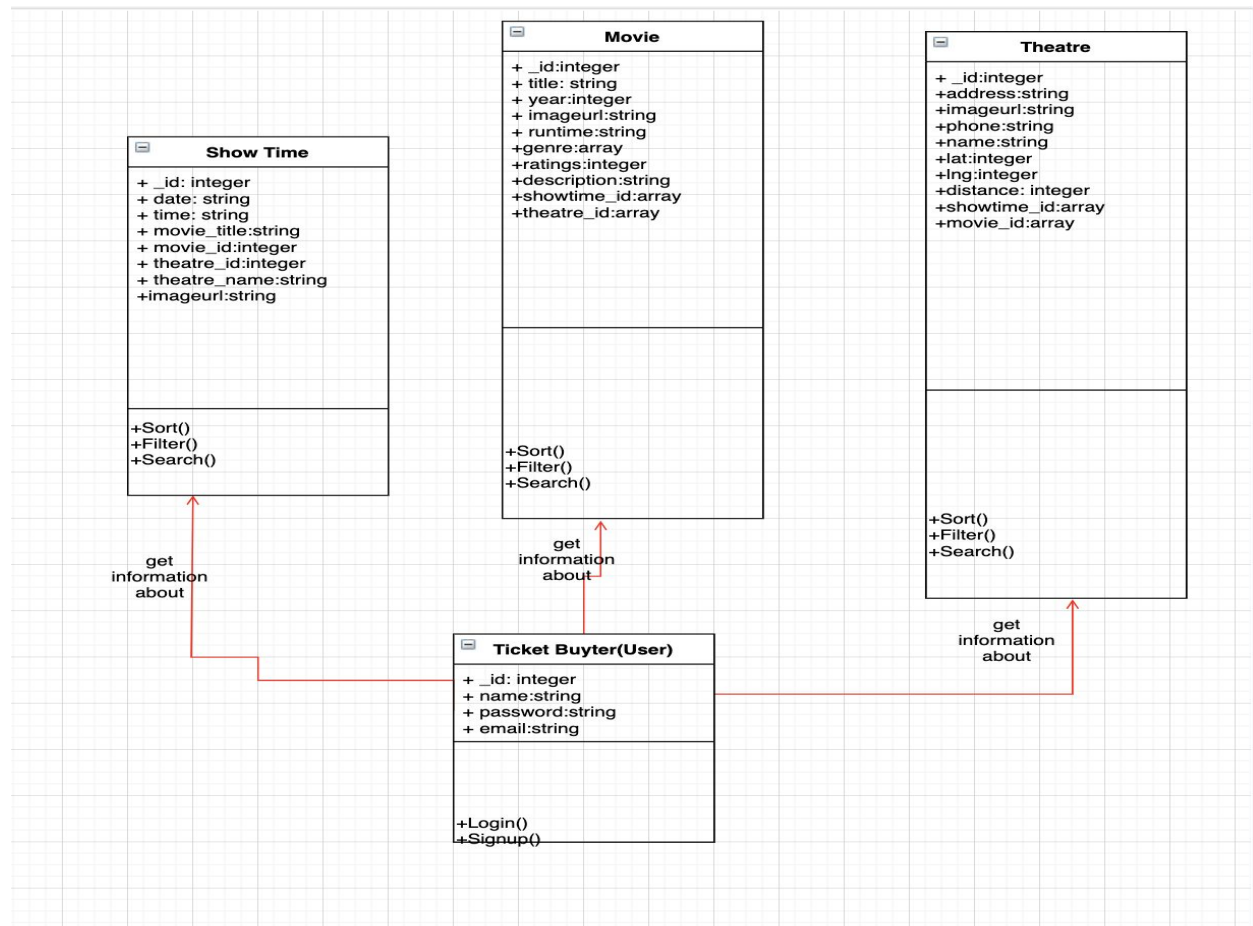
My use case diagram is shown below. Based on the user story above, the User(can also be called TicketBuyer) is an actor, he/she has associations with the Pumpkin Movie Website. He/she has about 12 new use cases in this Phase. I combined them with the use cases in Phase 1 and Phase 2.



Design and UML Diagrams

The general idea of Pumpkin Movie is to let users get the information about movies, theatres and showtimes ahead of time, then they can buy their ideal tickets in the end. The first UML diagram shows three models I created. I added some features of different models in this Phase.

The Movies model has 10 fields: id, title, year, image, runtime, genre, ratings, description, showtime_id and theatre_id. Genre is an array because one movie may have two genres. The Showtime model has 9 fields: id, date, time, movie_title, theatre_name, movie_id, theatre_id and imageurl. Movie and Theatre are related to each other (many to many relationships) through the Show Time. The Theatre model has 10 fields: id, address, imageurl, phone, name, latitude, longitude, distance, showtime_id and movie_id. Now three models are related to each other. All these three models can be filtered, sorted and Searched. The Ticket Buyer(User) shown at the bottom can get the information from all three models above. User has four fields: _id, name, password and email. They can login and signup into the system.



I planned to let users make transactions in a process movie -> theatre->showtimes->make transactions. But now I feel that users check movies, theatres and showtimes not in a specific order. So I do not think the sequential diagram I used before is appropriate for this system now.

Testing

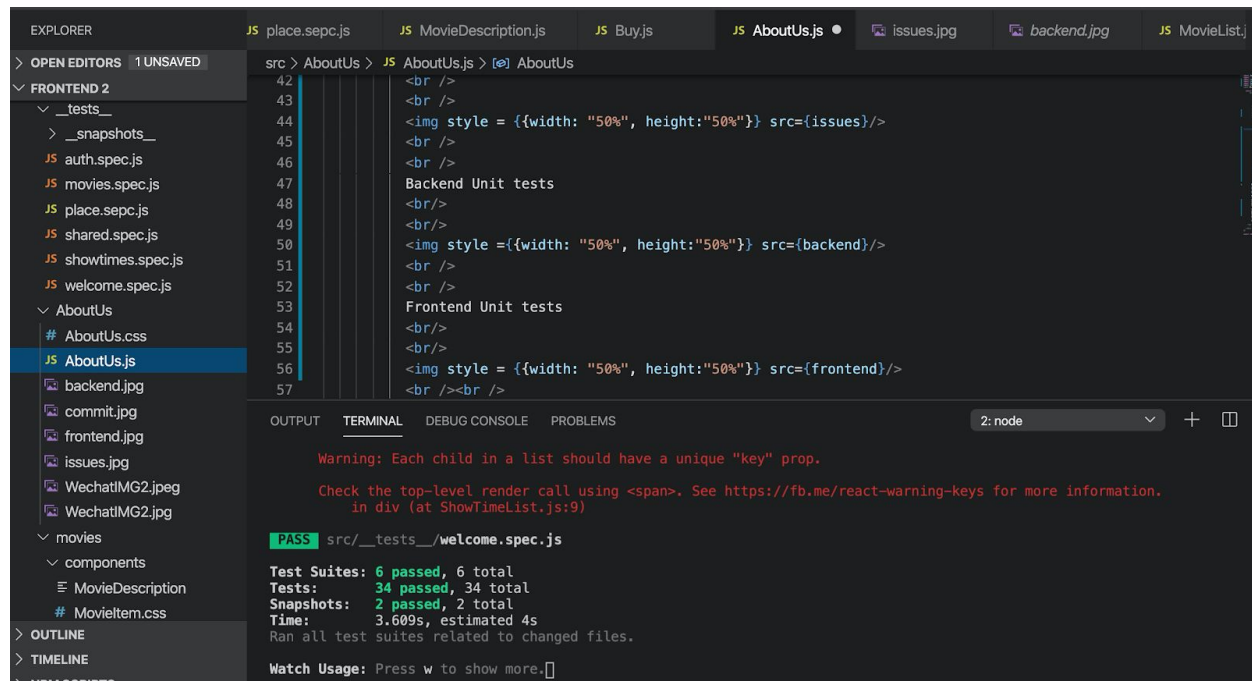
I spent half of my time testing the project in this Phase. I used Mocha and Chai to test the Nodejs codes I wrote for the backend. I mostly tested all routes to see if I can take different actions in a right way and get the ideal responses on different routes. I also tested the models (I added one user model in this phase) I created. I used the fake database in MongoDB to see if I can connect to it and got the right information I want through the models I wrote in the backend. For the backend, I did 10 tests and all of them were successful in the end.

```
will be removed in a future version. To use the new Server Discover and Monitoring engine, pass
option { useUnifiedTopology: true } to the MongoClient constructor.
Connected to test database
  GET Cinemas
(node:1608) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
connected!
    ✓ should respond with an object of cinema array (48ms)
  GET Movies
    ✓ should respond with an object of movie array (42ms)
  GET Showtimes
    ✓ should respond with an object of array of showtime (43ms)
  GET Movie By Id
    ✓ should respond with a movie with specific id (84ms)
  GET Movie By Id error test
    ✓ should return error message
  GET cinema by Id
    ✓ should respond with a movie with specific id (84ms)
  GET Users
    ✓ should respond with an object of user array (44ms)

Database Tests
(node:1608) DeprecationWarning: current URL string parser is deprecated, and will be removed in a
future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.conn
ect.
We are connected to test database!
  Test Database
    ✓ Should retrieve cinema data from test database (41ms)
    ✓ Should retrieve showtimes data from test database (40ms)
    ✓ Should retrieve movies data from test database (41ms)

10 passing (1s)
```

Similar to what I did in phase2, I incorporated Jest and Enzyme to test the front end which is based on React. I tested different stateful components to see if they worked as I expected. I used Shallow and Mount a lot of times. I completed 34 front end tests.



```
src > AboutUs > JS AboutUs.js > [e] AboutUs
42 |
43 | <br />
44 | <br />
45 | <img style = {{width: "50%", height:"50%"}} src={issues}/>
46 | <br />
47 | Backend Unit tests
48 | <br/>
49 | <br/>
50 | <img style={{width:"50%", height:"50%"}} src={backend}/>
51 | <br />
52 | <br />
53 | Frontend Unit tests
54 | <br/>
55 | <br/>
56 | <img style = {{width: "50%", height:"50%"}} src={frontend}/>
57 | <br /><br />

Warning: Each child in a list should have a unique "key" prop.
Check the top-level render call using <span>. See https://fb.me/react-warning-keys for more information.
    in div (at ShowTimeList.js:9)

PASS src/_tests_/welcome.spec.js
Test Suites: 6 passed, 6 total
Tests: 34 passed, 34 total
Snapshots: 2 passed, 2 total
Time: 3.609s, estimated 4s
Ran all test suites related to changed files.
Watch Usage: Press w to show more.
```

I used Selenium to do the GUI test. I need to control the time in order to get the ideal result. I used Selenium to test if different buttons work in the way they supposed to automatically. I tested all pages to make sure that all components exist and work well. This is easier than the unit tests I did.

Models

I have four models from three sources. I have already fetched all information I wanted from the different databases and related them in Phase2. In this phase, I collected 130 new movie data, 30 new theatres data near Austin. I think I have enough showtime data after Phase2 so I did not collect more showtimes. I added one user model in this phase.

My screenshots of different models in MongoDB are shown below.

This is the Cinema Model, it has id, cinema_id, cinema_name, address, address2, city, county, postcode, latitude, longitude, distance, logo-url, image, phone. I collected 85 documents for it.

pumpkinmovie.cinemas

COLLECTION SIZE: 20.79KB

TOTAL DOCUMENTS: 50

INDEXES TOTAL SIZE: 36KB

Find

Indexes

Aggregation

Search^{BETA}

INSERT DOCUMENT

FILTER {"filter":"example"}

Find

Reset

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("5e76be6c5b3aa9c6fb3d60e8")
cinema_id: 23962
cinema_name: "iPic Austin"
address: "3225 Amy Donovan Plaza"
address2: ""
city: "Austin"
county: "Travis"
postcode: 78758
lat: 30.398809
lng: -97.72644
distance: 19.358387225115
logo_url: "https://assets.movieglu.com/chain_logos/us/UK-915-sq.jpg"
ima... : "https://cdn.vox-cdn.com/thumbor/smxJktrHapC6NfuZle52ZQtRvi4=/280x0:3
58..."
phone: "5125683400"
```

I got these data from MovieGlueDatabase. I used Python to Fetch data through their API.


```

{
  "cinema_id": 8893,
  "cinema_name": "Cineworld Leicester Square",
  "address": "Leicester Square",
  "city": "London",
  "county": "Greater London",
  "postcode": "WC2H 7NA",
  "lat": 51.51078,
  "lng": -0.1305,
  "distance": 0.031255415281335,
  "logo_url": "https://d2z9fe5yu2p0av.cloudfront.net/chainlogos/uk/UK-868-sq.jpg"
},
{
  "cinema_id": 8923,
  "cinema_name": "Vue Cinemas - West End",
  "address": "3 Cranbourn Street",
  "city": "London",
  "county": "Greater London",
  "postcode": "WC2H 7AL",
  "lat": 51.511181,
  "lng": -0.12934,
  "distance": 0.064288623403495,
  "logo_url": "https://d2z9fe5yu2p0av.cloudfront.net/chainlogos/uk/UK-712-sq.jpg"
}

```

The second Model is Movie. I collected about 200 documents for it after clearning. It has id, Title, Year, Rated, Released, Runtime, Genre, Director, Writer, Actors, Plot, Language, Country, Awards, Poster, Ratings, metascore, imdbRating, imdbVotes, imdbID, Type, DVD, BoxOffice, Production, Website. I got these from IMDB.

pumpkinmovie.moviesnew

COLLECTION SIZE: 102.82KB TOTAL DOCUMENTS: 67 INDEXES TOTAL SIZE: 36KB

Find Indexes Aggregation Search^{BETA}

INSERT DOCUMENT

FILTER {"filter":"example"}

Find

Reset

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("5e7a3b484eedf7a695398b76")
Title: "Bridesmaids"
Year: "2011"
Rated: "R"
Released: "13 May 2011"
Runtime: "125 min"
Genre: "Comedy, Romance"
Director: "Paul Feig"
Writer: "Kristen Wiig, Annie Mumolo"
Actors: "Kristen Wiig, Terry Crews, Maya Rudolph, Tom Yi"
Plot: "Annie (Kristen Wiig), is a maid of honor whose life unravels as she le..."
Language: "English, Thai, Spanish"
Country: "USA"
Awards: "Nominated for 2 Oscars. Another 25 wins & 70 nominations."
Poster: "https://m.media-amazon.com/images/M/MV5BMjAyOTMyMzUxN15BMl5BanBnXkFtZT..."
> Ratings: Array
  Metascore: "75"
  imdbRating: "6.8"
  imdbVotes: "257,942"
  imdbID: "tt1478338"
  Type: "movie"
  DVD: "20 Sep 2011"
  BoxOffice: "$166,500,000"
  Production: "Universal Studios"
  Website: "N/A"
```

Examples

By Title

Title: Year: Plot: Response:

Request:

<http://www.omdbapi.com/?t=onward>

Response:

```
{
  "Title": "Onward",
  "Year": "2020",
  "Rated": "PG",
  "Released": "06 Mar 2020",
  "Runtime": "N/A",
  "Genre": "Animation, Adventure, Comedy, Family, Fantasy",
  "Director": "Dan Scanlon",
  "Writer": "Dan Scanlon (screenplay by), Jason Headley (screenplay by), Keith Bunin (screenplay by)",
  "Actors": "Tom Holland, Chris Pratt, Julia Louis-Dreyfus, Octavia Spencer",
  "Plot": "Set in a suburban fantasy world, two teenage elf brothers embark on a quest to discover if there is still magic out there.",
  "Language": "English",
  "Country": "USA",
  "Awards": "N/A",
  "Poster": "https://m.media-amazon.com/images/M/MV5BMjZlZjZkNzQzMmVhYjY0YVZmLTk5ZTEtNWU0NGVhMzYyWU1xkEykFqcGdeQXVyNDg4NjY5OTQ@_V1_SX300.jpg",
  "Ratings": [
    {
      "Source": "Rotten Tomatoes",
      "Value": "82%"
    }
  ],
  "Metascore": "N/A",
  "imdbRating": "N/A",
  "imdbVotes": "N/A",
  "imdbID": "tt7146812",
  "Type": "movie",
  "DVD": "N/A",
  "BoxOffice": "N/A",
  "Production": "Disney/Pixar",
  "Website": "N/A",
  "Response": "True"
}
```

The third Model is Show Time. I did not have a lot of updates about it since it seems perfect after Phase2. Movies and theatres are related to each other through showtimes. I got showtimes in local theatres(Texas) in the recent 90 days. However, a lot of theatres closed due to Cronovirus. I did not get as much data as I expected. I collected about 200 documents for it. I got these data from GraceNote Developer I used their sample php codes to fetch the data.

pumpkinmovie.showtimesnew

COLLECTION SIZE: 1014.04KB TOTAL DOCUMENTS: 509 INDEXES TOTAL SIZE: 36KB

[Find](#) [Indexes](#) [Aggregation](#) [Search^{BETA}](#)

FILTER {"filter":"example"}

QUERY RESULTS 181-200 OF MANY

```

_id: ObjectId("5e7971787f41567c80174f81")
title: "Fast & Furious 9: The IMAX 2D Experience"
des: "Dominic Toretto and his crew join forces to battle his deadly brother."
id: ObjectId("5e76fbcf41e903c8d678be57")
theatre: Object
  id: "4604"
  name: "Bullock Museum Spirit Theater"
  date: "2020-05-27"
  time: "16:30"
  cinemaid: 4381

```

Sample Request

```
http://data.tmsapi.com/v1.1/movies/9128357/showings?startDate=2012-12-08&api_key=1234567890
```

Sample Response

```

[
  {
    "tmsId": "MV003903970000",
    "rootId": "9128357",
    "title": "Argo",
    "titleLang": "en",
    "shortDescription": "A CIA agent hatches a risky plan to get six Americans out of Tehran during the Iran hostage crisis.",
    "descriptionLang": "en",
    "ratings": [ { "body": "Motion Picture Association of America", "code": "R" } ],
    "genres": ["Historical drama", "Thriller"],
    "topCast": ["Ben Affleck", "Bryan Cranston", "Alan Arkin" ],
    "preferredImage": {
      "uri": "movieposters/v5/AllPhotos/9128357/p9128357_p_v5_aa.jpg",
      "height": "360",
      "width": "240",
      "primary": "true",
      "category": "Poster Art",
      "caption": { "content": "Argo (2012)", "lang": "en" }
    },
    "releaseYear": 2012,
    "qualityRating": { "ratingsBody": "TMS", "value": "3.5" },
    "officialUrl": "http://argothemovie.warnerbros.com/",
    "runTime": "PT02H00M",
    "showtimes": [
      {
        "theatre": { "id": "5936", "name": "Regal City North Stadium 14" },
        "barg": false,
        "dateTime": "2012-12-08T19:05",
      },
      {
        "theatre": { "id": "5936", "name": "Regal City North Stadium 14" },
        "barg": false,
        "dateTime": "2012-12-08T22:05",
      }
    ]
  }
]

```

I copied some of my Python Scripts for cleaning and organizing the database.

The new model I created in this phase is the User model. It has `_id`, `reviews`(extra fields not required so I will implement it later), `tickets`, `name`, `email`, `image` and `password`. The user can create the account and log into the system and create their own transaction. This is the extra model I am planning to do later. In Phase3, I completed user login, signup and make transactions.

```
  _id: ObjectId("5ea0ed9d4708540743df5d10")
  > reviews: Array
  > tickets: Array
  name: "lena"
  email: "mosquitotian@gmail.com"
  image: "uploads/images/c45c29d0-8500-11ea-abff-bde9ed5a5334.jpeg"
  password: "$2a$12$1lZEG7sioH1Zb0SxLElhreJYrcgcOp4Rzg/.IJI6ZDkuQP1ElbVEq"
  __v: 0
```

Tools

I use Reactjs to build the frontend and Nodejs for the backend. For the frontend, I also used Bootstrap to build some React Components like Modal. To fetch data and organize the database, I incorporated Python and php. I mostly used Python to clean data and build relationships among different databases. To test the App, I used Mocha, Jest, Chai, Enzyme and Selenium in both phase 2 and phase 3. Mocha is for backend testing. Jest, Chai and Enzyme are for back end testing. I used Selenium for GUI testing. To deploy the App I used AWS Amplify together with Heroku. AWS for front end and Heroku for back end, which is the same as what I did in Phase 2.

Reflections

I feel that I almost completed all main functions in phase 2. So in this phase, I did some extra things like authentication, online transaction by myself. I also created one more model and collected more data. I think it is too hard to make an App perfect. I still feel there are a lot of things I can add to the App though they are not required by the course. I think I will continue to implement more features on this App. But I have to admit, I am still not good at testing, it is too tedious for me. Creating an App is interesting and creative, but testing is not. I hope I could gradually become used to it in the future since I know testing becomes more and more important now. But I really learned a lot from this Project, both front end and back end. If I work in a team, I would not be so tired but I could not learn so many things either.