ITCS212 Web Programming

Class Project Report

By

Section 2

6388139 Raweebhas Paiboonwong

6388143 Krisanakorn Rugbumroong

6388144 Krissanaphol Rugbumroong

6388182 Panithi Runggeratigul

Present

Prof. Jidapa Kraisangka

Prof. Wudhichart Sawangphol

2/2022

Overview

Our domain:

Our group choose **IMDb** as our domain. IMDb stand for “Internet Movie Database”. As the name suggest, IMDb is basically is a public movie database website.

Our website:

this is the look of our website

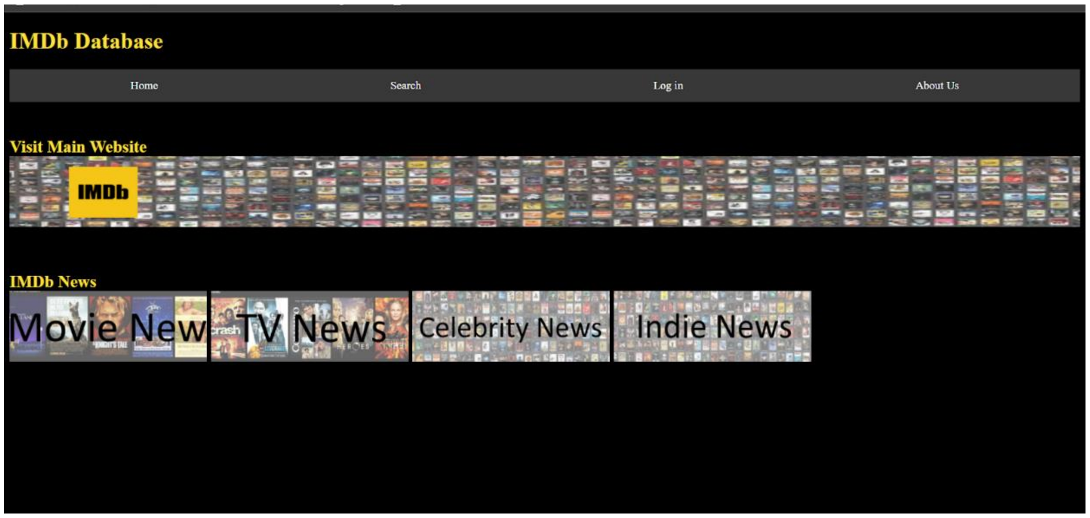


Figure 1 Our homepage

our website compose of 4 website include **homepage**, **login page**, **Search page** and **about us page**. User can navigate between this 4 pages by using a navigation at the top.

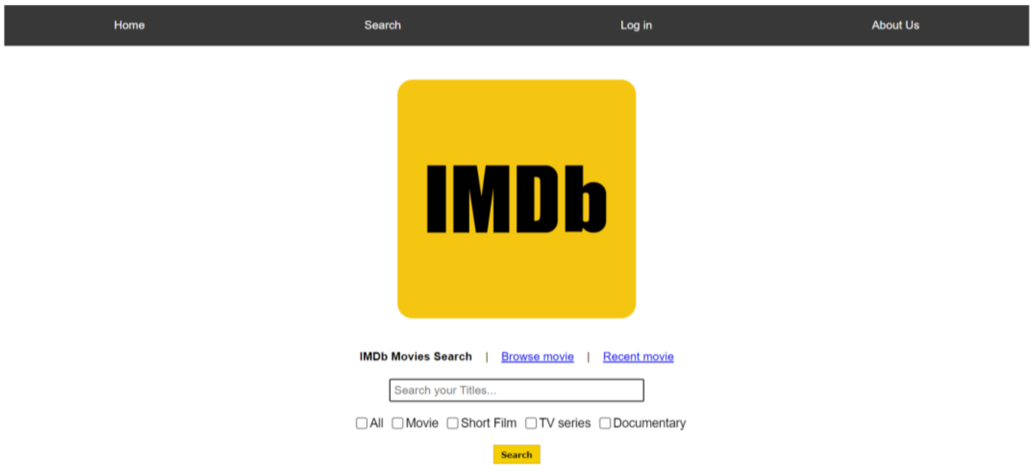


Figure 2 Phase I search page

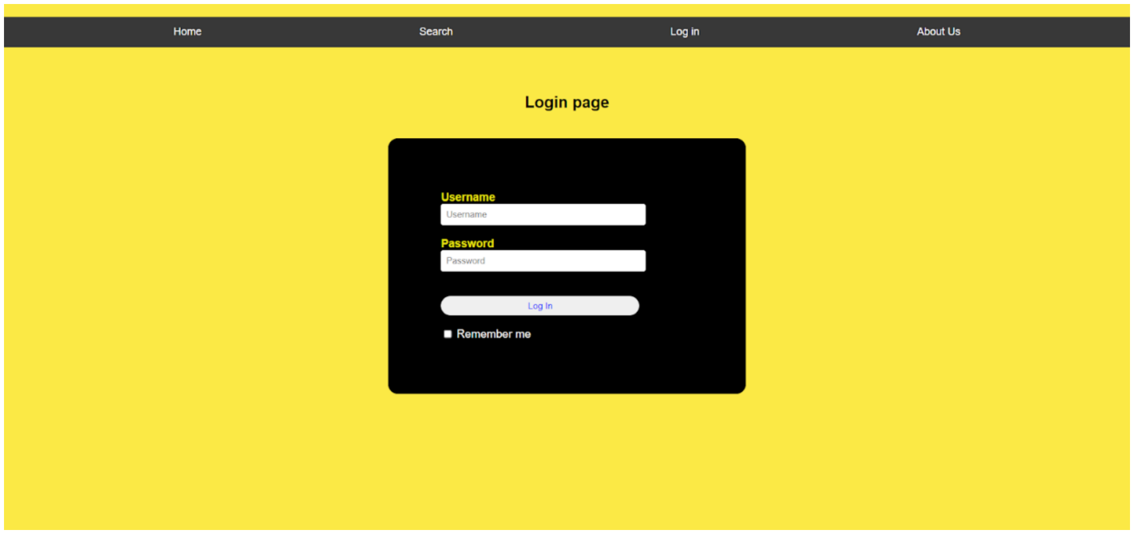
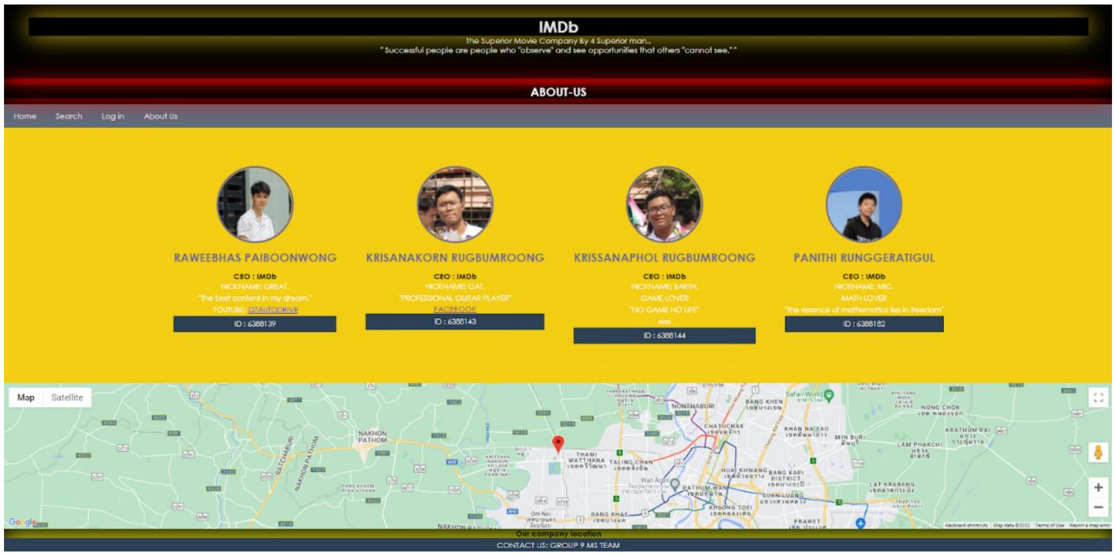


Figure 3 Phase I login page



IMDb

The Superior Movie Company By 4 Superior man..

" Successful people are people who "observe" and see opportunities that others "cannot see." "

ABOUT-US

RAWEEBHAS PAIBOONWONG

CEO : IMDb

NICKNAME: GREAT.

"The best content in my dream."

YOUTUBE: GTAUTODRIVE

ID : 6388139

KRISANAKORN RUGBUMROONG

CEO : IMDb

NICKNAME: OAT.

"PROFESSIONAL GUITAR PLAYER"

FACEBOOK

ID : 6388143

KRISSANAPHOL RUGBUMROONG

CEO : IMDb

NICKNAME: EARTH.

GAME LOVER

"NO GAME NO LIFE"

ezez

ID : 6388144

PANITHI RUNGGERATIGUL

CEO : IMDb

NICKNAME: MIC.

MATH LOVER

"The essence of mathematics lies in freedom"

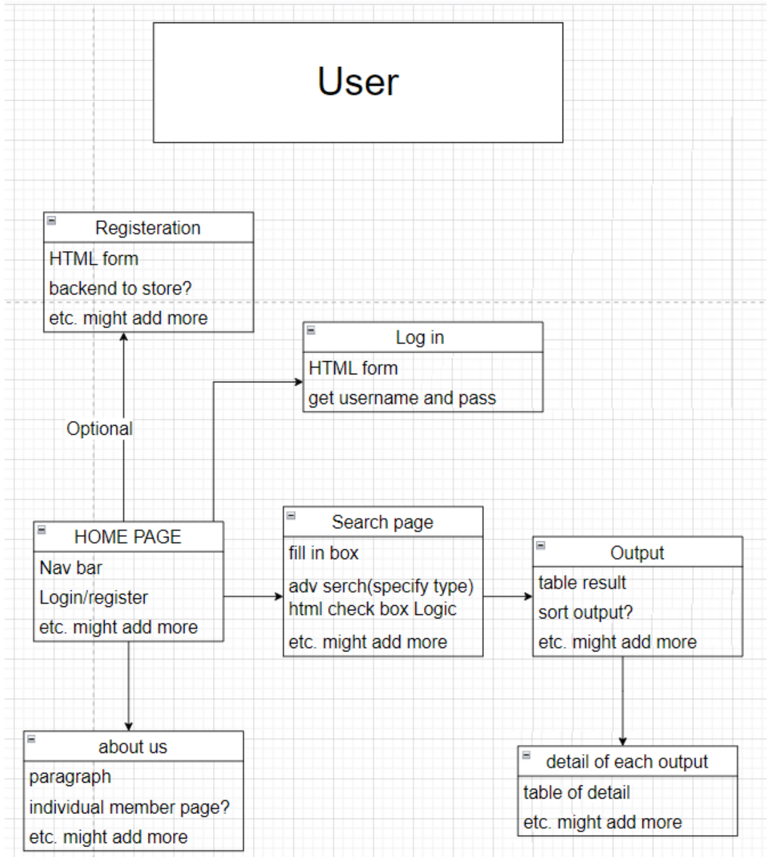
ID : 6388182

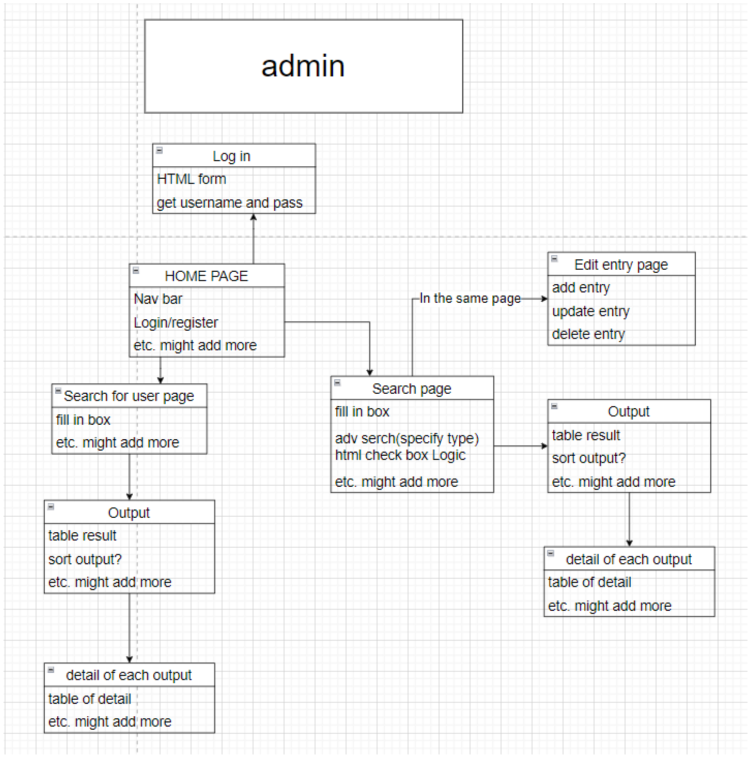
Our company location

CONTACT US: GROUP 9 MS TEAM

Figure 4 Phase II about us page

Navigation Diagram





Normal User Pages with explanation for each page

Home page

This page doesn't have anything fancy. It's mostly html and CSS. The only interesting part would be a navigation bar which we can use to navigate to different page.

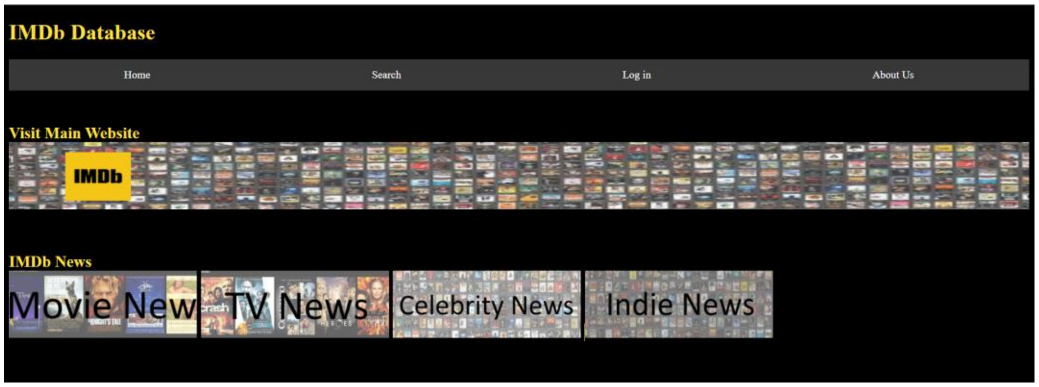


Figure 5 Homepage

Login page:

This page requires user username and password to log in to the website. It will fetch the data from database once then let the user to login if there is their data in database. If not it will show that " Your username or password are incorrect" or if users insert nothing then it will show "Please insert your username and password" instead. Once the users have logged in then it will redirect the next page. For normal user, it will redirect to the search page but for the admin user, it will redirect to the homepage.

Phase I:

The login page using the html and CSS to create the object within this page. This page consists of 2 main parts which are the navigation bar part and the login part. The navigation bars are just like the other pages which have an home, search, login and about us page. For the log in part, there are the box that will let the user to fill out their username and password and let the user to log in by click at the login button,

Phase II:

Fetch the parameter that the user inserted once the user clicked the submit button. The parameter will be the username and password. Once the web service gets the input, the webservice will use the SELECT method to find the valid user in the database.

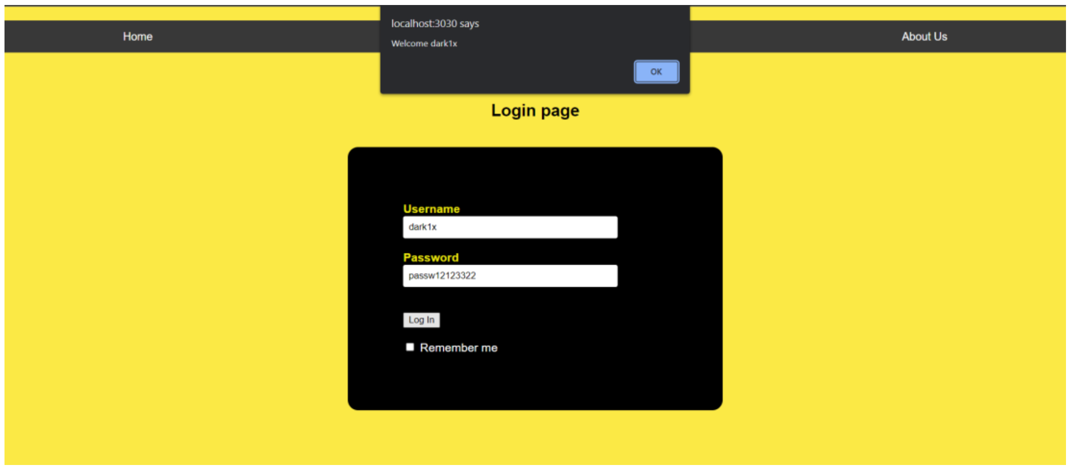


Figure 6 Phase II login successfully

Search page:

This page serves as a search product page. In other words, It's a page that will let users search movies in our database. Users have to type in movie name that he/she want to search. select movie categories that he/she want to search. Once he/she finished, the page will redirected to the result page.

Phase I:

Search page are implemented as a html file. In phase 1, this page doesn't have any search functionality yet. This is more like a prototype of how the search would look like.

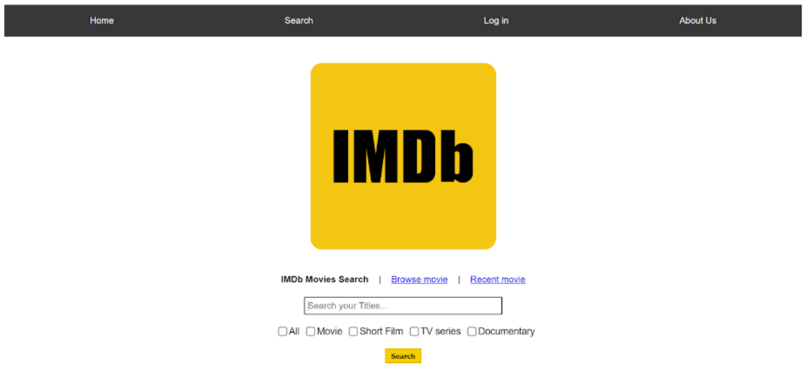


Figure 7 Search page in Phase I

Phase II:

In phase 2, search page now has search functionality as well as criteria search thanks to JavaScript. Once user enter query and select the criteria, Client JavaScript will fetch the matched result from server. After that, user will be redirect to the result page.

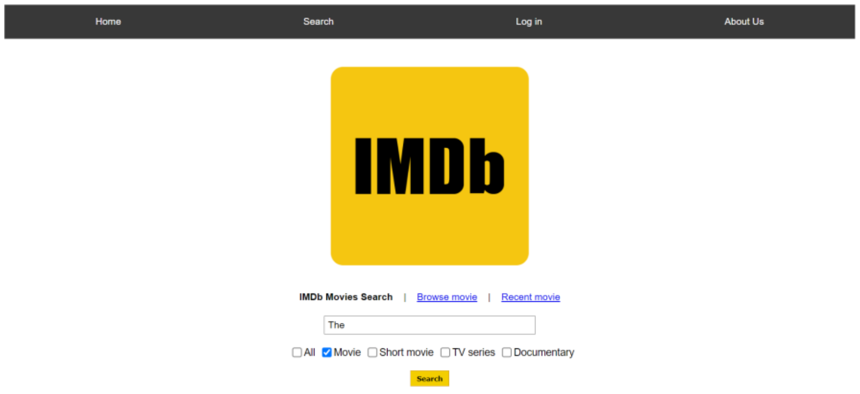
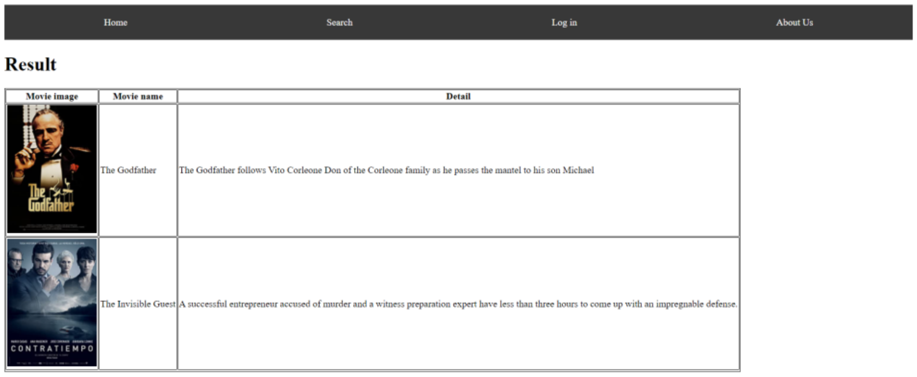

Home
Search
Log in
About Us
IMDb
IMDb Movies Search | Browse movie | Recent movie
The
☐ All ☑ Movie ☐ Short movie ☐ TV series ☐ Documentary
Search

Figure 8 Search page: search query "The" in "movie" categories



Result

Movie image	Movie name	Detail
	The Godfather	The Godfather follows Vito Corleone Don of the Corleone family as he passes the mantel to his son Michael
	The Invisible Guest	A successful entrepreneur accused of murder and a witness preparation expert have less than three hours to come up with an impregnable defense.

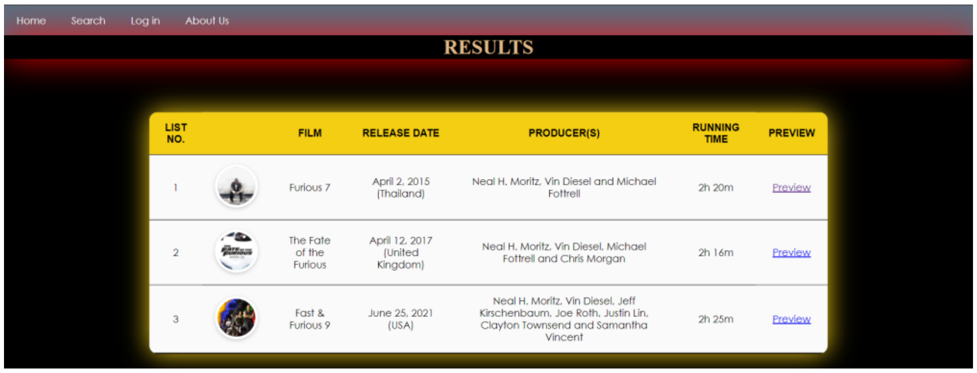
Figure 9 Result page after search

Result page:

This is the page that contain fetched result from search services.

Phase I:

In this phase, result page are more like a prototype of how our result from search page would look like



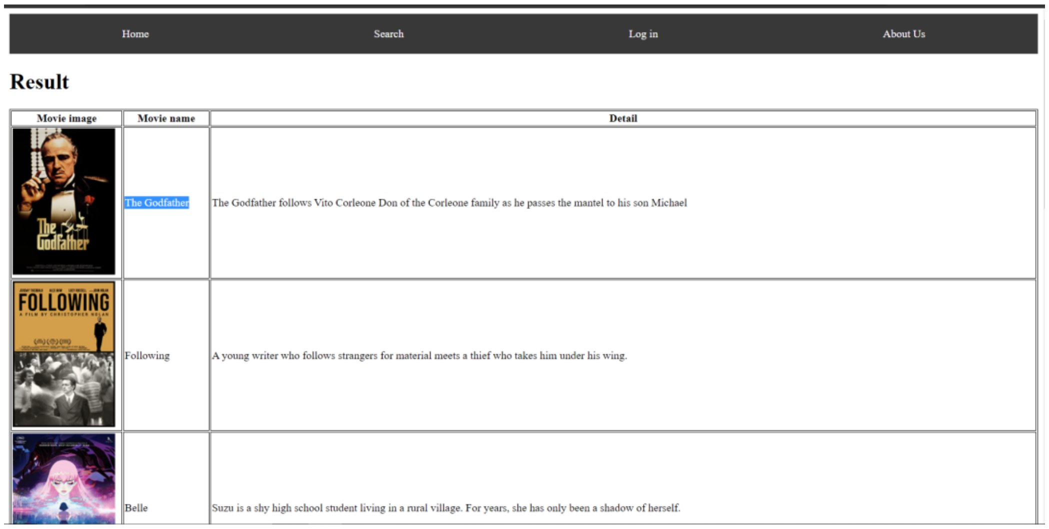
RESULTS

LIST NO.		FILM	RELEASE DATE	PRODUCER(S)	RUNNING TIME	PREVIEW
1		Furious 7	April 2, 2015 (Thailand)	Neal H. Moritz, Vin Diesel and Michael Fottrell	2h 20m	Preview
2		The Fate of the Furious	April 12, 2017 (United Kingdom)	Neal H. Moritz, Vin Diesel, Michael Fottrell and Chris Morgan	2h 16m	Preview
3		Fast & Furious 9	June 25, 2021 (USA)	Neal H. Moritz, Vin Diesel, Jeff Kirschenbaum, Joe Roth, Justin Lin, Clayton Townsend and Samantha Vincent	2h 25m	Preview

Figure 10 Phase I result page

Phase II:

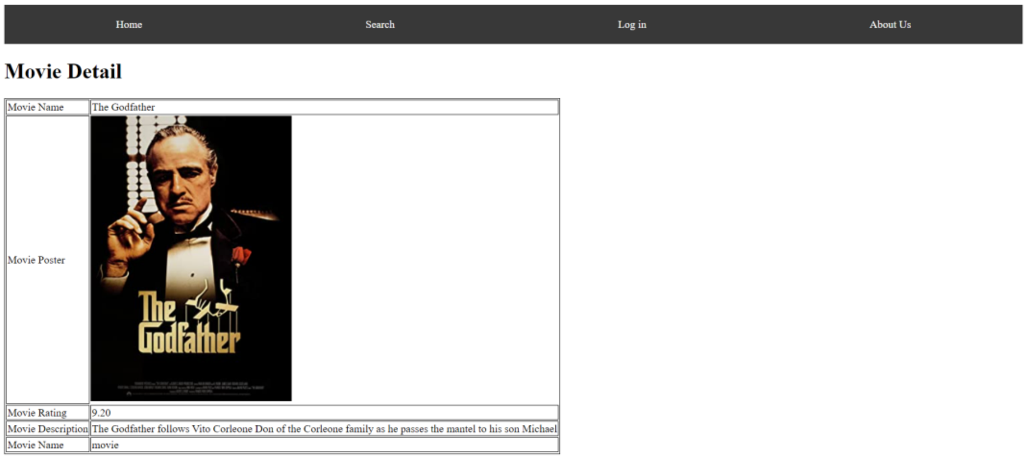
In this phrase, due to the complexity of phase I CSS, we decide to use simpler CSS style. However, this time, result page are actually contain a fetched data from the server. User can also view detail movie information by clicking at the movie name;



Result

Movie image	Movie name	Detail
	The Godfather	The Godfather follows Vito Corleone Don of the Corleone family as he passes the mantel to his son Michael
	Following	A young writer who follows strangers for material meets a thief who takes him under his wing.
	Belle	Suzu is a shy high school student living in a rural village. For years, she has only been a shadow of herself.

Figure 11 Phase II result page, movie name is also clickable

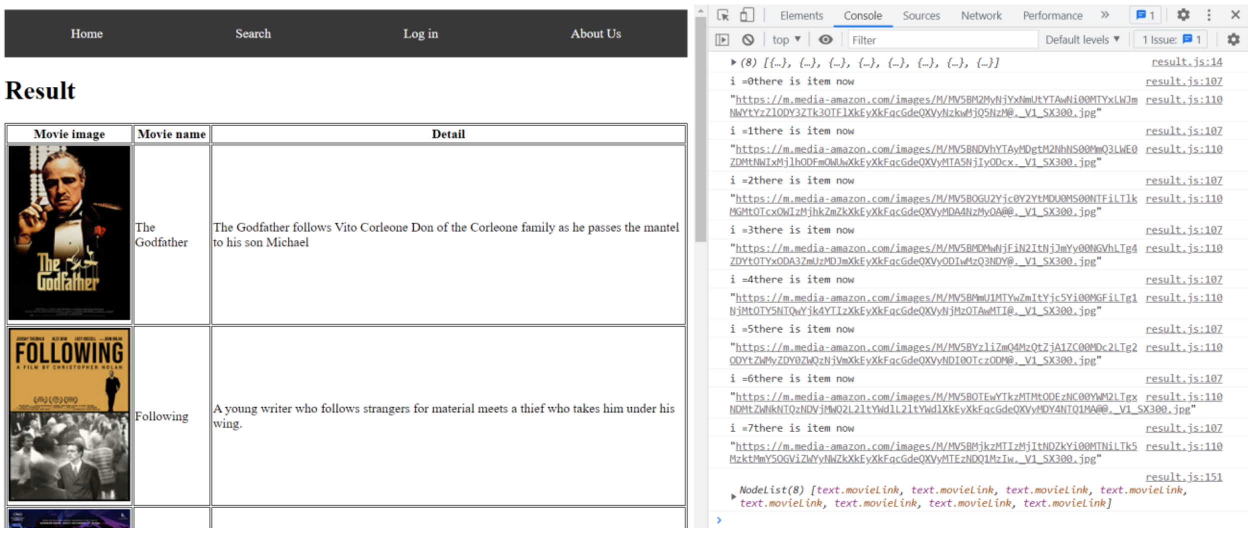


Movie Detail

Movie Name	The Godfather
Movie Poster	
Movie Rating	9.20
Movie Description	The Godfather follows Vito Corleone Don of the Corleone family as he passes the mantel to his son Michael
Movie Name	movie

Figure 12 After click at The Godfather. user will redirect to this movie detail page

In this phase, result page also connected to one public API which is OMDb. OMDb is a free API that can fetch movie data from IMDb website. We use this API to fetch the movie cover.
https://www.omdbapi.com/



Result

Movie image	Movie name	Detail
	The Godfather	The Godfather follows Vito Corleone Don of the Corleone family as he passes the mantel to his son Michael
	Following	A young writer who follows strangers for material meets a thief who takes him under his wing.

```
▸ (8) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]                               result.js:14
i =0there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BM2MyNjYxNmUtYTAwNi00MTYxLWJm        result.js:110
NWYtYzZlODY3ZTk3OTFlXkEyXkFqcGdeQXVyNzkwMjQ5NzM@._V1_SX300.jpg"
i =1there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BNDVhYTAyMDgtM2NhNS00MmQ3LWE0        result.js:110
ZDMtNWIxMjlhODFmOWUwXkEyXkFqcGdeQXVyMTA5NjIyODcx._V1_SX300.jpg"
i =2there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BOGU2Yjc0Y2YtMDU0MS00NTFiLTlk        result.js:110
MGMtOTcxOWIzMjhkZmZkXkEyXkFqcGdeQXVyMDA4NzMyOA@@._V1_SX300.jpg"
i =3there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BMDMwNjFiN2ItNjJmYy00NGVhLTg4        result.js:110
ZDYtOTYxODA3ZmUzMDJmXkEyXkFqcGdeQXVyODIwMzQ3NDY@._V1_SX300.jpg"
i =4there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BMmU1MTYwZmItYjc5Yi00MGFiLTg1        result.js:110
NjMtOTY5NTQwYjk4YTIzXkEyXkFqcGdeQXVyNjMzOTAwMTI@._V1_SX300.jpg"
i =5there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BYzliZmQ4MzQtZjA1ZC00MDc2LTg2        result.js:110
ODYtZWMyZDY0ZWQzNjVmXkEyXkFqcGdeQXVyNDI0OTczODM@._V1_SX300.jpg"
i =6there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BOTEwYTkzMTMtODEzNC00YWM2LTgx        result.js:110
NDMtZWNkNTQzNDVjMWQ2L2ltYWdlL2ltYWdlXkEyXkFqcGdeQXVyMDY4NTQ1MA@@._V1_SX300.jpg"
i =7there is item now                                                         result.js:107
"https://m.media-amazon.com/images/M/MV5BMjkzMTIzMjItNDZkYi00MTNiLTk5        result.js:110
MzktMmY5OGViZWYyNWZkXkEyXkFqcGdeQXVyMTEzNDQ1MzIw._V1_SX300.jpg"
                                                                              result.js:151
▸ NodeList(8) [text.movieLink, text.movieLink, text.movieLink, text.movieLink,
  text.movieLink, text.movieLink, text.movieLink, text.movieLink]
›
```

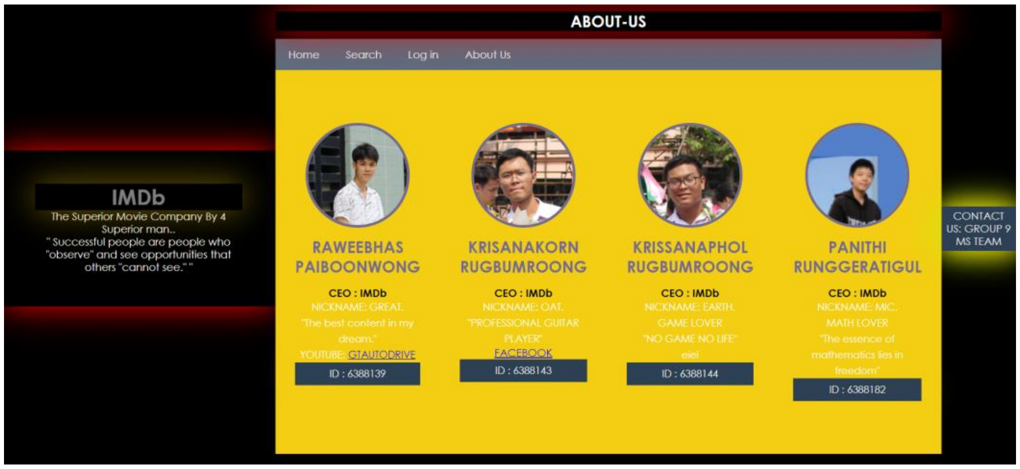
Figure 13 fetched URL of movie cover in console

About us page:

On this page, Normal Users can see the information of our members and contact, and also users can use the navigation bar to another page too. In phase 1 but in phase 2 our group provides a google Maps free trial API for our website to let Normal Users use to see our location clearly.

Phase I:

On this phase I in About us page users can see member information and work navigation bar (prototype).



The Superior Movie Company By 4 Superior man..
" Successful people are people who "observe" and see opportunities that others "cannot see." "

RAWEEBHAS PAIBOONWONG
CEO : IMDb
NICKNAME: GREAT.
"The best content in my dream."
YOUTUBE: GTAUTODRIVE
ID : 6388139

KRISANAKORN RUGBUMROONG
CEO : IMDb
NICKNAME: OAT.
"PROFESSIONAL GUITAR PLAYER"
FACEBOOK
ID : 6388143

KRISSANAPHOL RUGBUMROONG
CEO : IMDb
NICKNAME: EARTH.
GAME LOVER
"NO GAME NO LIFE"
eiei
ID : 6388144

PANITHI RUNGGERATIGUL
CEO : IMDb
NICKNAME: MIC.
MATH LOVER
"The essence of mathematics lies in freedom"
ID : 6388182

Figure 14 Phase I about us page

Phase II:

We add the .js file and the google map API to the page.

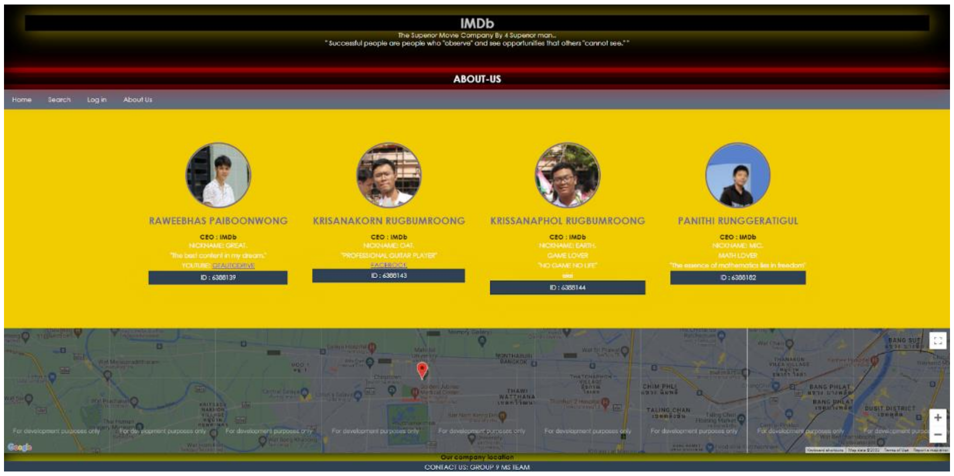
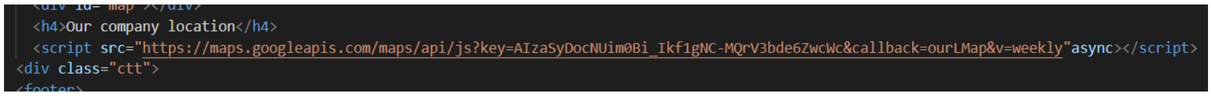
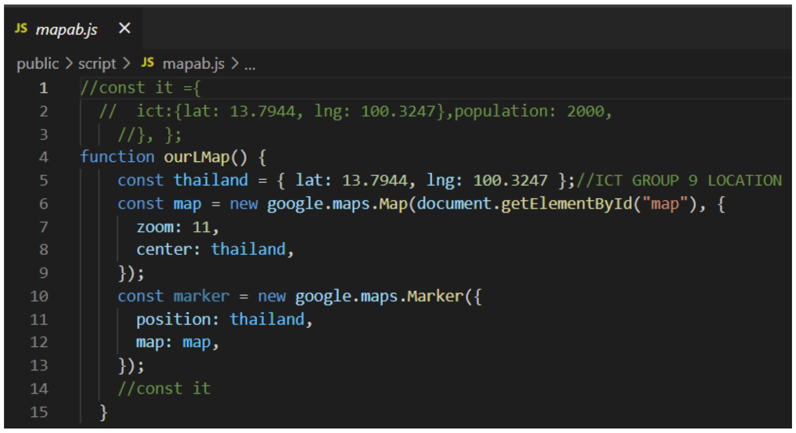


Figure 15 Phase II about us page with google map API



```
  <div id="map"></div>
  <h4>Our company location</h4>
  <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyDocNUim0Bi_Ikf1gNC-MQrV3bde6ZwcWc&callback=ourLMap&v=weekly"async></script>
<div class="ctt">
<footer>
```



```javascript
//const it ={
  //  ict:{lat: 13.7944, lng: 100.3247},population: 2000,
    //}, };
function ourLMap() {
    const thailand = { lat: 13.7944, lng: 100.3247 };//ICT GROUP 9 LOCATION
    const map = new google.maps.Map(document.getElementById("map"), {
      zoom: 11,
      center: thailand,
    });
    const marker = new google.maps.Marker({
      position: thailand,
      map: map,
    });
    //const it
  }
```

Figure 16 Google map API with ICT building as a location

Administrators Pages with explanation for each page

Home page

This page is very much the same as user's homepage except that it's implemented as Reactjs.

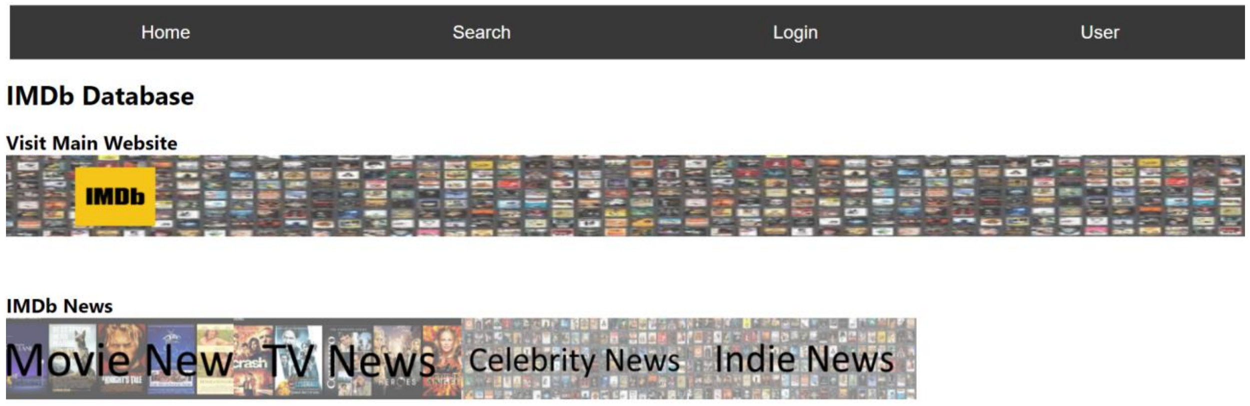


Figure 17 Phase III home page

Login page

The react of the login page for administrator is similar to the normal user from phase II except that this page is implemented as Reactjs. It will fetch the same parameters and the web service will SELECT from the database and find if the parameters are valid or not.

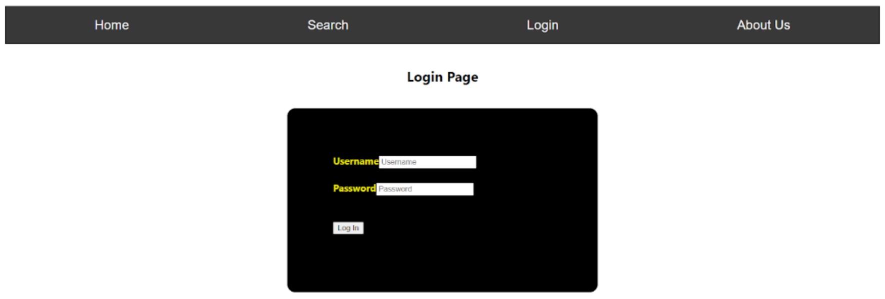


Figure 18 Phase III login page

```
import React from 'react'
import styled from "styled-components";
import './css/login.css'

const H2 = styled.h2`
    text-align: center;
    color: #000000;
    padding: 20px;
`;

class Login extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            Uname: "",
            Pass: ""
        };
        this.handleChange = this.handleChange.bind(this);
        this.handleSubmit = this.handleSubmit.bind(this);
    }
    handleChange(e) {
        const target = e.target;
        const value = target.value;
        const elementname = target.name;
        this.setState({
            [elementname]: value,
        });
    }
    handleSubmit(e) {
        e.preventDefault();
        CheckUser()
    }
    render() {
    return (
        <>
        <H2>Login Page</H2>
        <div className="login" id="login">
            <form onSubmit={this.handleSubmit}>
                <label className="myLabel"><b>Username</b></label>
                <input type="text" name="Uname" id="Uname" place-
holder="Username"  value={this.state.Uname} on-
Change={this.handleChange}/><br/><br/>
                <label className="myLabel"><b>Password</b></label>
                <input type="Password" name="Pass" id="Pass" place-
holder="Password"  value={this.state.Pass} on-
Change={this.handleChange}/><br/><br/><br/>
                <input type="button" onClick={(e)=>this.handleSubmit(e)}
value="Log In"/><br/><br/>
            </form>
        </div>
        </>
    );
    }
}
```

Constructor with handle the change and submit.

Call CheckUser() method once the submit

Render HTML page in the react, each text input will call handleChange() and button will call handleSubmit().

The inputs are username and password (both inputs are required).

```
function CheckUser() {
    //console.log("log in")
    let username = document.getElementById("Uname").value;
    let password = document.getElementById("Pass").value;
    let rooturl = "http://localhost:3030/login/"+username+"/"+password;

    var log = document.getElementById("login");
    var error_log = document.createElement("div");
    error_log.setAttribute("id", "err_log");

    if (username === "" || password === "") {
        error_log.innerText = "Please insert username and password";
        error_log.style.color = "red";
        try {
            var first_search = document.getElementById("err_log");
            log.removeChild(first_search);
            log.appendChild(error_log);
        }
        catch(error) {
            log.appendChild(error_log);
        }
    }
    else {
        fetch(rooturl).then((res) => res.json()).then((data) => {
        try {
            alert("Welcome " + data.result.username);
            window.location.replace("http://localhost:3000");
        }
        catch(error) {
            error_log.innerText = "Your username or password maybe
incorrect";
            error_log.style.color = "red";
            log.appendChild(error_log);
            try {
                var first_search = document.getElementById("err_log");
                log.removeChild(first_search);
                log.appendChild(error_log);
            }
            catch(error) {
                log.appendChild(error_log);
            }
        }
        })
    }
}

export default Login;
```

CheckUser() will receive the input.

First, it will check that, is the user missed any input or not. It will attach the error text; “Please insert username and password”, if any input is not provided.

Try and catch will remove the old error text to avoid the repeat error message when the user inserts multiple invalid inputs.

After all the inputs are received, it will fetch the URL with the parameter username and password. In case of error (the username or password is not match in the database, it will attach the error text; “Your username or password maybe incorrect.”

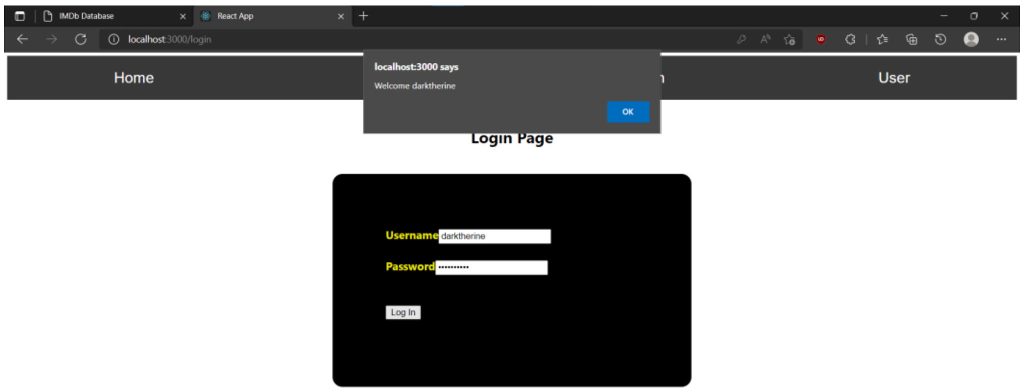


Figure 19 Phase III login successfully

Search page

This search has have some similarity to user’s search page. which is the layout and the functionality of movie search. However, we also add insert, update and delete forms which will connected to our service and server.

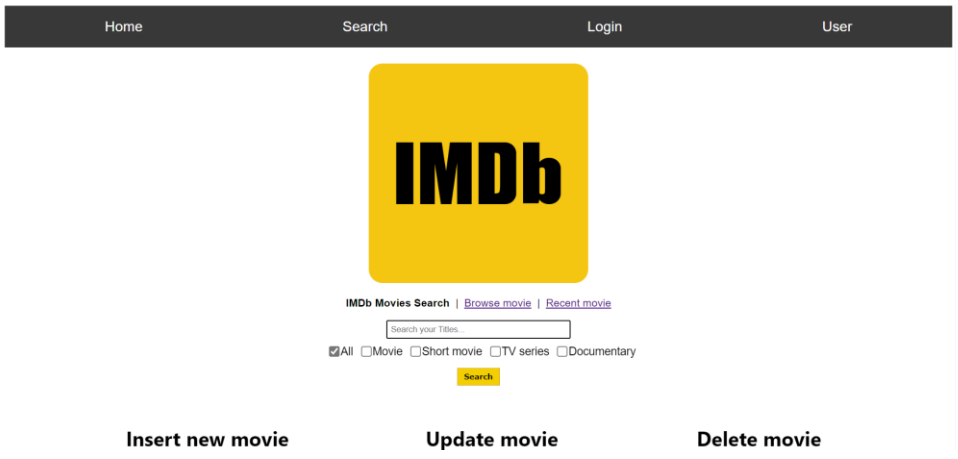


Figure 20 Phase III search page

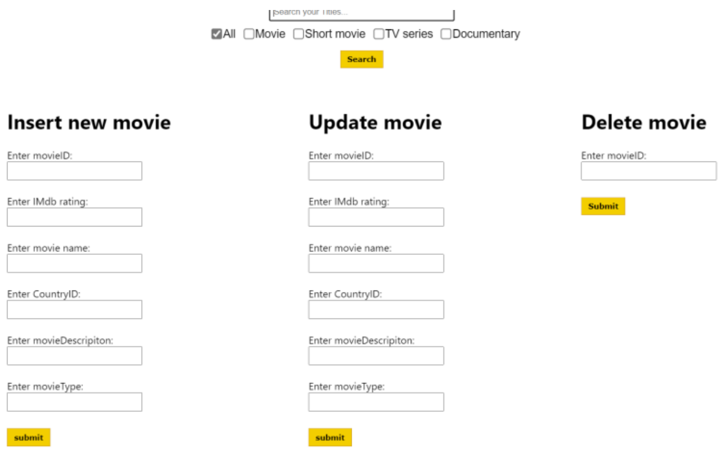


Figure 21 Phase III search page: insert, update and delete form.

User management page

This page is quite similar to search movie page. Except that instead of search, insert, update and delete movie, it’s do all of those action to user instead.

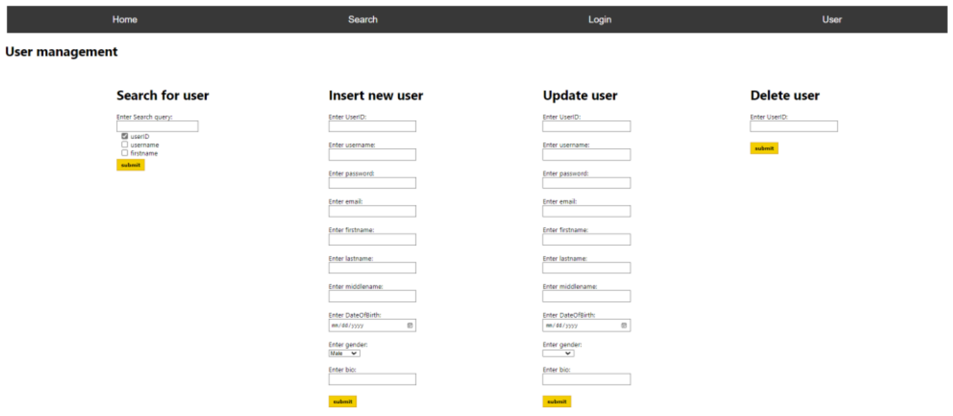


Figure 22 Phase III User management page

All explanations for web services with code

Initialize services

At first, we would just import module and initialize express before we doing any routing.

```
const mysql = require('mysql2');
const dotenv = require('dotenv');
const express = require('express');
const app = express();
const router = express.Router();
const path = require('path');
const bodyParser = require('body-parser');
var cors = require('cors');


// set cors option otherwise it would be blocked by CORS policy in browser
var corsOption = {
    'Access-Control-Allow-Origin': 'http://localhost:3000/',
    'Access-Control-Allow-Methods' : 'GET, POST, OPTIONS, PUT, PATCH, DELETE',
    'preflightContinue': true
}

// set up env
dotenv.config({ path: "./process.env"  });

// route express to static file (CSS, Javascript) in folder public
```

```
app.use(express.static(__dirname + '/public'));
app.use(express.urlencoded({ extended: true}));
// express parse request body
app.use(bodyParser.text());
app.use(bodyParser.json());
```

Initialize express and create connection to database

After that, we start to create express router as well as a connection to the database

```
app.use("/", router);
router.use(cors(corsOption));
// complex http method (PUT,DELETE) sent OPTION first before apply CORS header
// this is just to route every OPTION method to have CORS header
router.options('*',cors(corsOption));


app.listen(3030, function(){
    console.log("listening at Port " + 3030);
});


var dbconn = mysql.createConnection({
    host : process.env.host,
    user : process.env.user,
    password : process.env.pwd,
    database : process.env.db
});


dbconn.connect(function(err){
    if(err) throw err;
    console.log("Connected to " + process.env.db);
});
```

URL routing

Next, we set express router to return the page correspond to the request (In Phase II, Phase III we use react to route instead).

```
//home
router.get('/',function(req,res){
    res.statusCode = 200; //status 200: OK
    console.log("Accessed Home page");
    res.sendFile(path.join(__dirname+'/public/homepage.html'));
   });
//login
   router.get('/login',function(req,res){
    res.statusCode = 200; //status 200: OK
    console.log("Accessed login page");
    res.sendFile(path.join(__dirname+'/public/login.html'));
   });

   //search
router.get('/search',function(req,res){
    //if(err) throw err;
    res.statusCode = 200;
    console.log("get search page");
    res.sendFile(path.join(__dirname + '/public/search.html'))
});

//aboutus
router.get('/aboutus',function(req,res){
    res.statusCode = 200; //status 200: OK
    console.log("Accessed login page");
    res.sendFile(path.join(__dirname+'/public/aboutuspGT.html'));
   });
```

Login service

Next, we implement login web service that will authenticate username and password of user. We also use this service at login page

```
//login services
router.get('/login/:user/:pass', function (req, res) {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT,
PATCH, DELETE'); // If needed
    res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-
type');
    res.setHeader('Access-Control-Allow-Credentials', true);
    var username = req.params.user;
    var password = req.params.pass;
    var sql = "SELECT * FROM user_ where username=" + mysql.escape(username)+ "
AND u_password=" + mysql.escape(password);
    if (!username && !password) {
        return res.status(400).send({ error: true, message: 'Please insert
username and password' });
    }
    dbconn.query(sql, function (error, results) {
    if (error) throw error;
        return res.send({ error: false, result: results[0], message: 'User
retrieved' });
    });
});
```

Search movie service

Next, we implement search movie service which require to return data from user's query and criteria. This web service is used in search page

```
/**
 * Search for movie
 * test: localhost:3030/search-movie
 * method: POST
 * put this in the body RAW JSON
 *
{
    "que" : "The",
    "crit": "short"
}
 */

router.post('/search-movie/',function(req,res){

    console.log("search service started");
    console.log(req.body);
    let query = req.body.que;
    let checkbox=req.body.crit;
    let crit="";

    if(checkbox==null);

    else if(checkbox[0]=="all" ||
checkbox=="all"){crit+="movie|short|TVSeries|documentary";}
    else if(checkbox[0].length>1){
        for(let i=0;i<checkbox.length;i++){
            console.log("crit :"+checkbox[i]);

            if(i<checkbox.length-1){crit+=checkbox[i]+"|";}
            else if(i==checkbox.length-1){crit+=checkbox[i]}
        }
    }
    else{
        console.log("crit: "+checkbox);
        crit+=checkbox;
    }
    console.log("criteria :"+crit);

    query = '%'+query+'%'
    var sql = "SELECT * FROM movie WHERE movieName LIKE "+mysql.escape(query)+" AND movieType REGEXP" + mysql.escape(crit);


    dbconn.query(sql,function(error,results){
        if (error) throw error;
        //console.log({data: results})
        return res.send({error: false, data: results, message: "Movie retrieved"})
    })
});
```

Search user service

Next, we implement search user service which is quite similar to search movie except that it’s a search for user instead.

```
/**
 * Search for user
 * test: localhost:3030/search-user
 * method: POST
 * put this in the body RAW JSON
 *
{
    "que" : "dark",
    "crit": "userID"
}
 */


router.post('/search-user/',function(req,res){

    console.log("search service started");
    console.log(req.body);
    let query = req.body.query;
    console.log(query);
    query = '%'+query+'%'
    let checkbox=req.body.crit;
    let crit="";
    if(checkbox==null);
    else if(checkbox[0].length>1){
        for(let i=0;i<checkbox.length;i++){
            //console.log("crit :"+checkbox[i]);
            if(i==0){
                crit+=" "+checkbox[i]+" LIKE "+mysql.escape(query)+" ";
            }
            else{
                crit+="OR "+checkbox[i]+" LIKE "+mysql.escape(query)+" ";
            }
        }
    }
    else{
        console.log("crit: "+checkbox);
        crit+=checkbox;
    }
    // console.log("criteria :"+crit);
    var sql = 'SELECT * FROM user_ WHERE'+crit;
    //console.log(sql);
    dbconn.query(sql,function(error,results){
        if (error) throw error;
        return res.send({error: false, data: results, message: "User retrieved"})
    })
});
```

View all movie, user service

Next, we implement view service for both movie and user. This will return all of movie and user entries in the database

```
/**
 * view movie
 * method: GET
 * test: localhost:3030/movies
 */
router.get('/movies', function (req, res) {
    dbconn.query('SELECT * FROM movie', function (error, results) {
    if (error) throw error;
    return res.send({ error: false, data: results, message: 'Movie list.' });});
});
/**
 * view user
 * method: GET
 * test: localhost:3030/users
 */
router.get('/users', function (req, res) {
    dbconn.query('SELECT * FROM user_', function (error, results) {
    if (error) throw error;
    return res.send({ error: false, data: results, message: 'User list.' });});
});
```

View movie, user with matching movieID and userID
Next, we also implement view service for user and movie. But instead of view all of entries, We only return matching userID and movieID that match in URL params

```
/**
 * get movie by movieID
 * test: localhost:3030/13651628
 * method: GET
 */
router.get('/title/:movieID', function(req,res){
    let movieID = req.params.movieID;
    //console.log(req.body);
    dbconn.query("SELECT * FROM movie WHERE movieID
LIKE?",[movieID],function(error,results){
        if (error) throw error;
        console.log({data: results});
        return res.send({error: false,data: results, messege: "Movie retrieved"})
    })
});
/**
 * get user by userID
 * test: localhost:3030/search-user/421233
 */
router.get('/search-user/:id',function(req,res){
    let userID = req.params.id;
    //console.log(req.body);
    //res.send(`query = ${query}`);
    console.log(`${userID}`);
    userID = '%'+userID+'%'
    //console.log(query);
    dbconn.query("SELECT * FROM user_ WHERE userID LIKE
?",[userID],function(error,results){
        if (error) throw error;
        console.log({data: results})
        return res.send({error: false, data: results, message: "User retrieved"})
    })
});
```

Insert movie and user services

We implement this service by taking the JSON body of the request and add them to database

```
/**
 * insert user
 * test: localhost:3030/ins-user
 * method: POST
 * put this in body: raw JSON
 {
    "user":{
        "userID" : 104142,
        "username" : "nnloat",
        "u_password" : "123456789",
        "email" : "nno1212@gmail.com",
        "firstname" : "Oatty",
        "lastname": "Oat",
        "middlename" : null,
        "DateOfBirth" : "1990-2-1",
        "gender" : "M",
        "bio": null

    }
}
 *
 */
router.post('/ins-user', function (req, res) {
    let user = req.body.user;
    //console.log(user);
    if (!user) {
    return res.status(400).send({ error: true, message: 'Please provide user
information' });
    }
    dbconn.query("INSERT INTO user_ SET ? ", user, function (error, results) {
    if (error) throw error;
    return res.send({error: false, data: results.affectedRows, message: 'New user
has been added successfully.'});
    });
});
/**
 * insert movie
 * test: localhost:3030/ins-mov
 * method : POST
 * put this in body: raw JSON
 *
```

```
{
    "movie":{
        "movieID" :"tt2625030",
        "IMDB_rating" : 7.6,
        "movieName" : "New World",
        "countryID" : 101256,
        "movieDescripiton" : "An undercover cop finds it difficult to play both a
cop and a goon.",
        "movieType": "movie"
    }
}
 */
router.post('/ins-mov', function (req, res) {
    let movie = req.body.movie;
    //console.log(movie);
    if (!movie) {
    return res.status(400).send({ error: true, message: 'Please provide movie
information' });
    }
    dbconn.query("INSERT INTO movie SET ? ", movie, function (error, results) {
    if (error) throw error;
    return res.send({error: false, data: results.affectedRows, message: 'New
movie has been added successfully.'});
    });
});
```

Update movie and user services

We implement this service by taking JSON body of the request then update the database according to it.

```
/**
 * update movie
 * test: localhost:3030/upd-movie
 * method: PUT
 * update movie name from "New World" to "New World updated"
 * put this in body: raw JSON
{
    "movie":{
        "movieID" :"tt2625030",
        "IMDB_rating" : 7.6,
        "movieName" : "New World updated",
        "countryID" : 101256,
        "movieDescripiton" : "An undercover cop finds it difficult to play both a
cop and a goon.",
        "movieType": "movie"
    }
}
 *
 */
router.put('/upd-movie', function (req, res) {
    console.log("updating movie");
    let movieID = req.body.movie.movieID;
    let movie = req.body.movie;
    if (!movieID || !movie) {
        return res.status(400).send({ error: movie, message: 'Please provide
movie information!!' });
    }
    dbconn.query("UPDATE movie SET ? WHERE movieID = ?", [movie, movieID],
function (error,results) {
    if (error) throw error;
    return res.send({error: false, data: results.affectedRows, message: 'Movie
has been updated successfully.'})
    });
});

/**
 * update user
 * test: localhost:3030/upd-user
 * method: PUT
 * update username from "nnloat" to 'nnloatty'
 * put this in body: raw JSON
{

"user":{
        "userID" : 104142,
        "username" : "nnloatty",
        "u_password" : "123456789",
        "email" : "nno1212@gmail.com",
        "firstname" : "Oatty",
        "lastname": "Oat",
        "middlename" : null,
        "DateOfBirth" : "1990-2-1",
        "gender" : "M",
        "bio": null

    }
}
 */
router.put('/upd-user', function (req, res) {
    let userID = req.body.user.userID;
    let user_ = req.body.user;
    if (!userID || !user_) {
        return res.status(400).send({ error: user_, message: 'Please provide user
information!!' });
    }
    dbconn.query("UPDATE user_ SET ? WHERE userID = ?", [user_, userID], function
(error,results) {
    if (error) throw error;
    return res.send({error: false, data: results.affectedRows, message: 'User has
been updated successfully.'})
    });
});
```

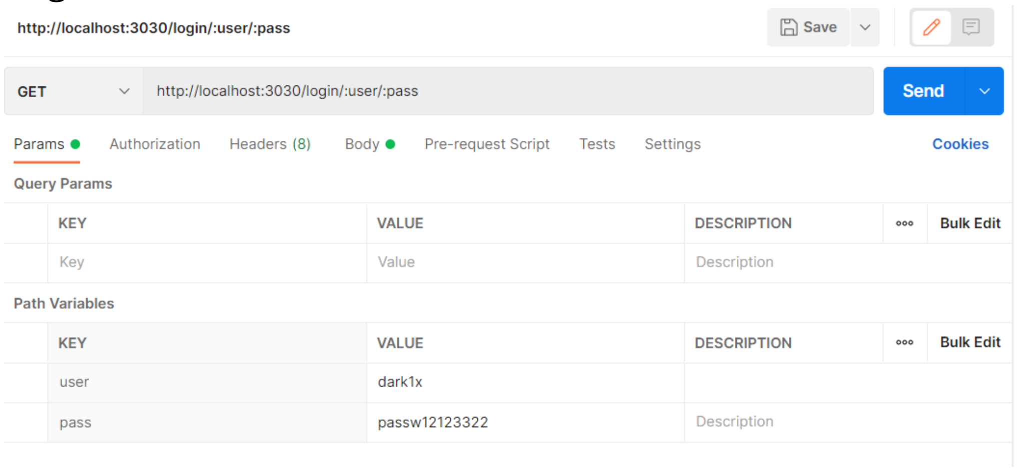
Delete movie and user service
This service take Text body of the request. Although we say that it's take Text body, the body of request still have to be in JSON format in order to parse it. We discovered later that sending DELETE method information by body is discouraged, that's why it just send as raw text instead of JSON.

```
/**
 * delete user by userID
 * test: localhost:3030/user
 * method: DELETE
 * put this in body: raw Text
 *
{
    "userID": 104142
}
 */
router.delete('/del-user', function (req, res) {
    let jsonBody = JSON.parse(req.body);
    let userID = jsonBody.userID;
    if (!userID) {
        return res.status(400).send({ error: true, message: 'Please provide
userID' });
    }
    dbconn.query('DELETE FROM user_ WHERE userID = ?', [userID], function (error,
results)
    {
    if (error) throw error;
        return res.send({ error: false, data: results.affectedRows, message:
'User has been deleted successfully.' });
    });
});

/**
 * delete movie by movieID
 * test: localhost:3030//movie
 * method: DELETE
 * put this in body: raw Text
 *
{
    "movieID": "tt2625030"
}
 */
router.delete('/del-movie', function (req, res) {
    console.log("body is "+req.body);
    let jsonBody = JSON.parse(req.body);
    console.log(jsonBody);
    let movieID = jsonBody.movieID;
    // let movieID = req.body.movieID;
    if (!movieID) {
        return res.status(400).send({ error: true, message: 'Please provide
movieID' });
    }
    dbconn.query('DELETE FROM movie WHERE movieID = ?', [movieID], function
(error, results)
    {
    if (error) throw error;
        return res.send({ error: false, data: results.affectedRows, message:
'Movie has been deleted successfully.' });
    });
});
```

All web services testing results using Postman

Login



http://localhost:3030/login/:user/:pass

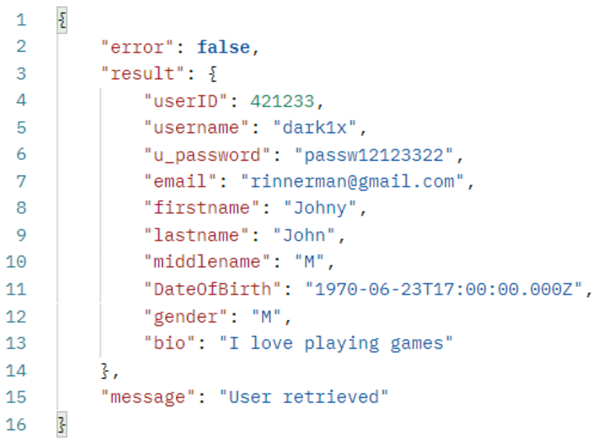
GET http://localhost:3030/login/:user/:pass

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

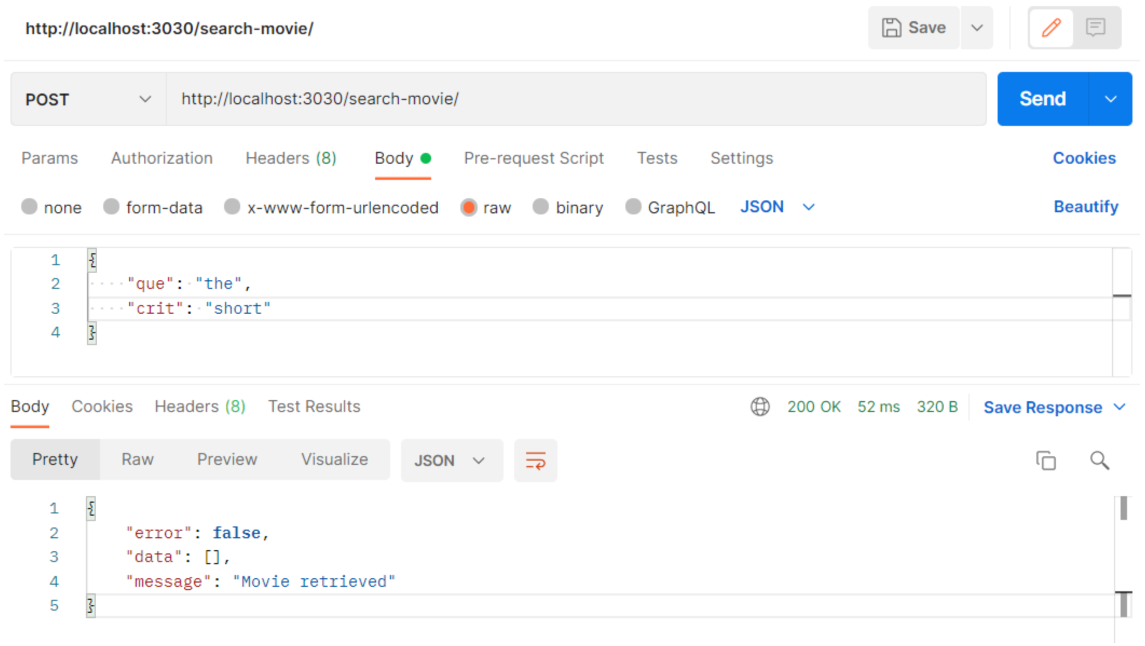
Path Variables

KEY	VALUE	DESCRIPTION
user	dark1x	
pass	passw12123322	Description



```
{
    "error": false,
    "result": {
        "userID": 421233,
        "username": "dark1x",
        "u_password": "passw12123322",
        "email": "rinnerman@gmail.com",
        "firstname": "Johny",
        "lastname": "John",
        "middlename": "M",
        "DateOfBirth": "1970-06-23T17:00:00.000Z",
        "gender": "M",
        "bio": "I love playing games"
    },
    "message": "User retrieved"
}
```

Search-movie

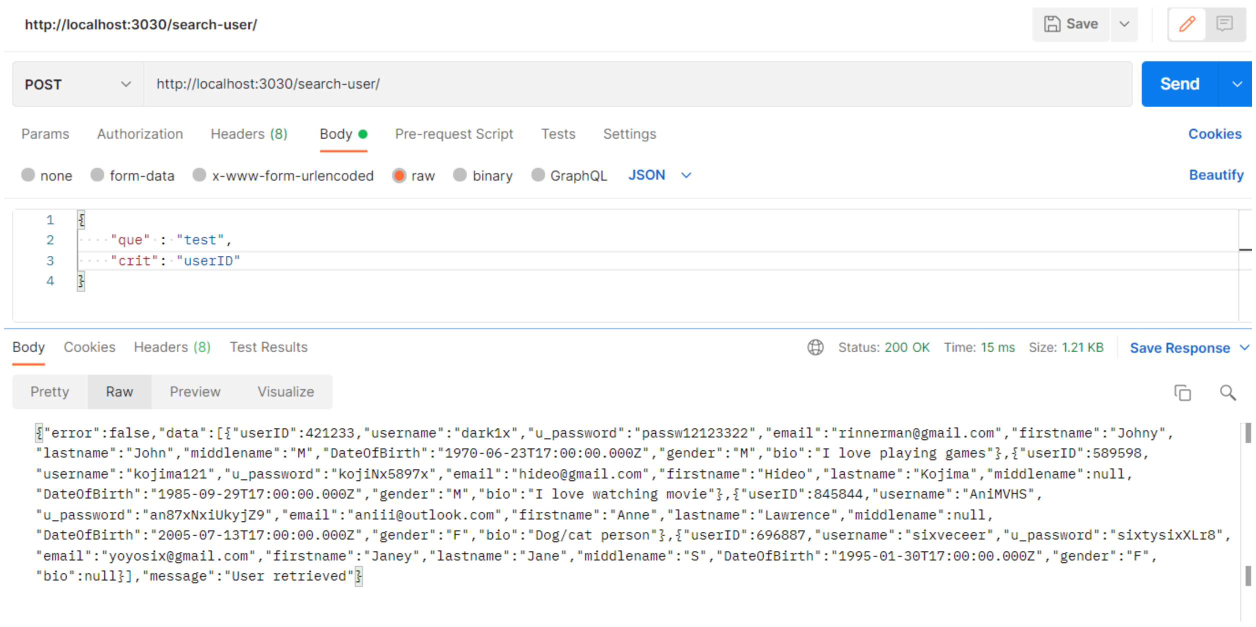


POST http://localhost:3030/search-movie/

```
{
    "que": "the",
    "crit": "short"
}
```

```
{
    "error": false,
    "data": [],
    "message": "Movie retrieved"
}
```

Search-user

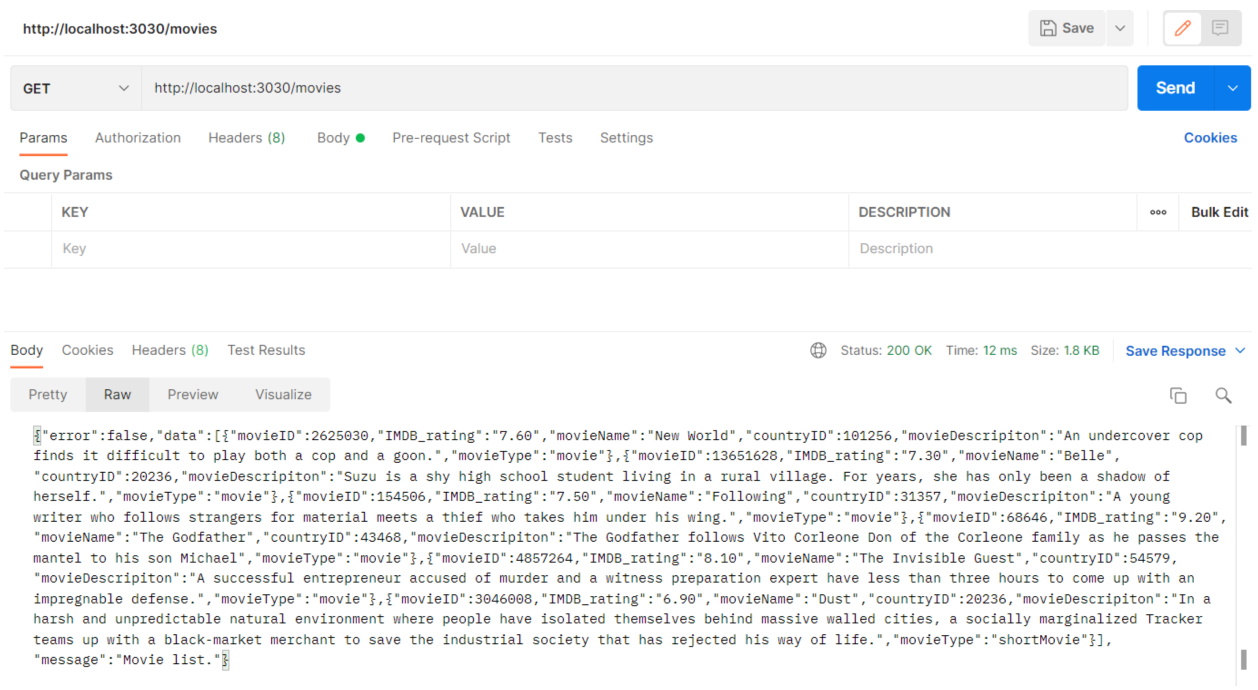


POST http://localhost:3030/search-user/

```
{
    "que" : "test",
    "crit": "userID"
}
```

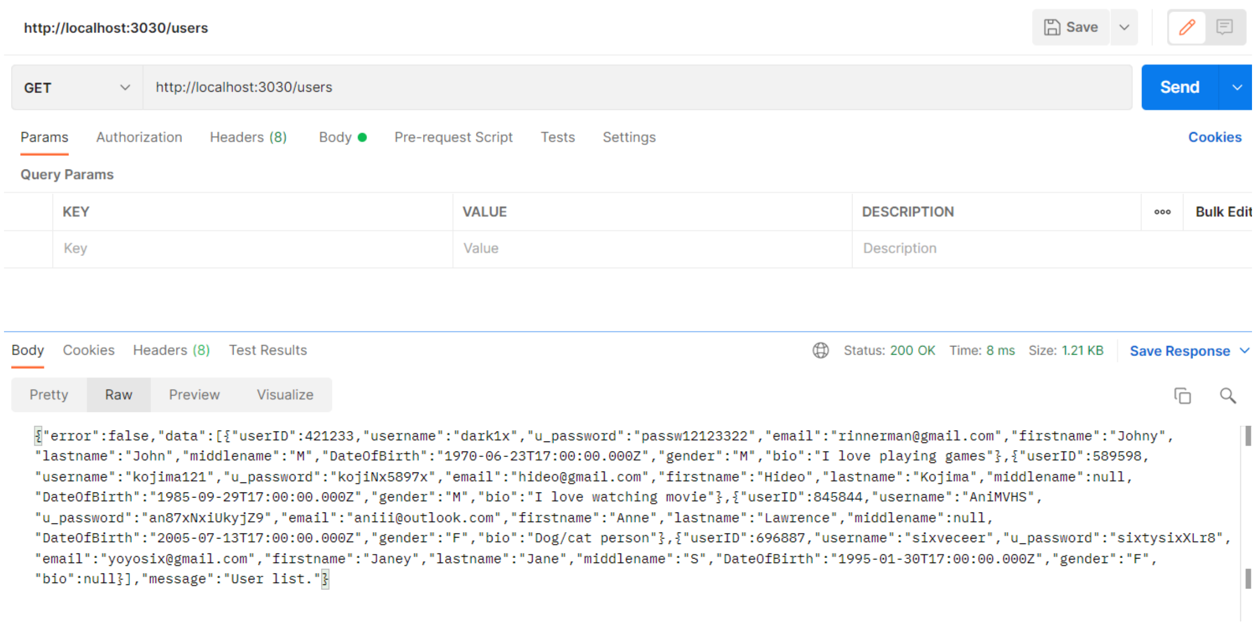
```
{"error":false,"data":[{"userID":421233,"username":"dark1x","u_password":"passw12123322","email":"rinnerman@gmail.com","firstname":"Johny",
"lastname":"John","middlename":"M","DateOfBirth":"1970-06-23T17:00:00.000Z","gender":"M","bio":"I love playing games"},{"userID":589598,
"username":"kojima121","u_password":"kojiNx5897x","email":"hideo@gmail.com","firstname":"Hideo","lastname":"Kojima","middlename":null,
"DateOfBirth":"1985-09-29T17:00:00.000Z","gender":"M","bio":"I love watching movie"},{"userID":845844,"username":"AniMVHS",
"u_password":"an87xNxiUkyjZ9","email":"aniii@outlook.com","firstname":"Anne","lastname":"Lawrence","middlename":null,
"DateOfBirth":"2005-07-13T17:00:00.000Z","gender":"F","bio":"Dog/cat person"},{"userID":696887,"username":"sixveceer","u_password":"sixtysixXLr8",
"email":"yoyosix@gmail.com","firstname":"Janey","lastname":"Jane","middlename":"S","DateOfBirth":"1995-01-30T17:00:00.000Z","gender":"F",
"bio":null}],"message":"User retrieved"}
```

View-movie



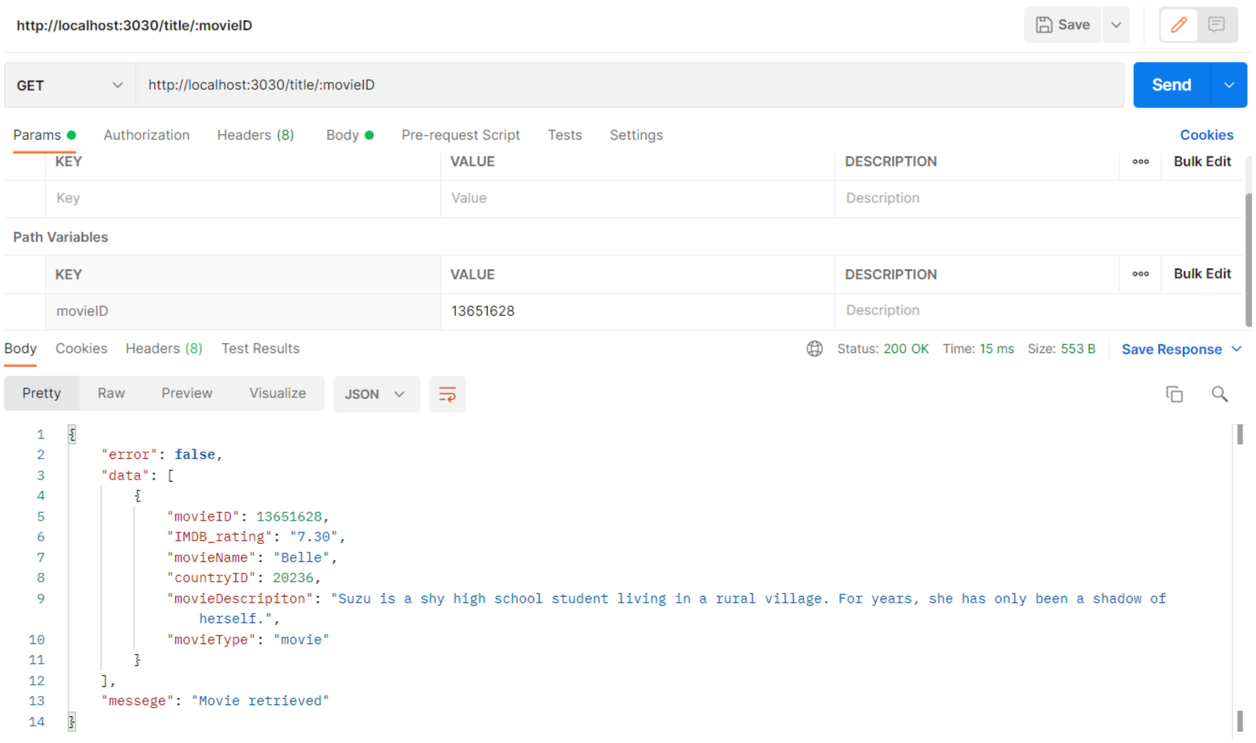
```
{"error":false,"data":[{"movieID":2625030,"IMDB_rating":"7.60","movieName":"New World","countryID":101256,"movieDescripiton":"An undercover cop
finds it difficult to play both a cop and a goon.","movieType":"movie"},{"movieID":13651628,"IMDB_rating":"7.30","movieName":"Belle",
"countryID":20236,"movieDescripiton":"Suzu is a shy high school student living in a rural village. For years, she has only been a shadow of
herself.","movieType":"movie"},{"movieID":154506,"IMDB_rating":"7.50","movieName":"Following","countryID":31357,"movieDescripiton":"A young
writer who follows strangers for material meets a thief who takes him under his wing.","movieType":"movie"},{"movieID":68646,"IMDB_rating":"9.20",
"movieName":"The Godfather","countryID":43468,"movieDescripiton":"The Godfather follows Vito Corleone Don of the Corleone family as he passes the
mantel to his son Michael","movieType":"movie"},{"movieID":4857264,"IMDB_rating":"8.10","movieName":"The Invisible Guest","countryID":54579,
"movieDescripiton":"A successful entrepreneur accused of murder and a witness preparation expert have less than three hours to come up with an
impregnable defense.","movieType":"movie"},{"movieID":3046008,"IMDB_rating":"6.90","movieName":"Dust","countryID":20236,"movieDescripiton":"In a
harsh and unpredictable natural environment where people have isolated themselves behind massive walled cities, a socially marginalized Tracker
teams up with a black-market merchant to save the industrial society that has rejected his way of life.","movieType":"shortMovie"}],
"message":"Movie list."}
```

View-user



```
{"error":false,"data":[{"userID":421233,"username":"dark1x","u_password":"passw12123322","email":"rinnerman@gmail.com","firstname":"Johny",
"lastname":"John","middlename":"M","DateOfBirth":"1970-06-23T17:00:00.000Z","gender":"M","bio":"I love playing games"},{"userID":589598,
"username":"kojima121","u_password":"kojiNx5897x","email":"hideo@gmail.com","firstname":"Hideo","lastname":"Kojima","middlename":null,
"DateOfBirth":"1985-09-29T17:00:00.000Z","gender":"M","bio":"I love watching movie"},{"userID":845844,"username":"AniMVHS",
"u_password":"an87xNxiUkyjZ9","email":"aniii@outlook.com","firstname":"Anne","lastname":"Lawrence","middlename":null,
"DateOfBirth":"2005-07-13T17:00:00.000Z","gender":"F","bio":"Dog/cat person"},{"userID":696887,"username":"sixveceer","u_password":"sixtysixXLr8",
"email":"yoyosix@gmail.com","firstname":"Janey","lastname":"Jane","middlename":"S","DateOfBirth":"1995-01-30T17:00:00.000Z","gender":"F",
"bio":null}],"message":"User list."}
```

Search-movie (by id)



GET http://localhost:3030/title/:movieID

KEY	VALUE	DESCRIPTION
Key	Value	Description

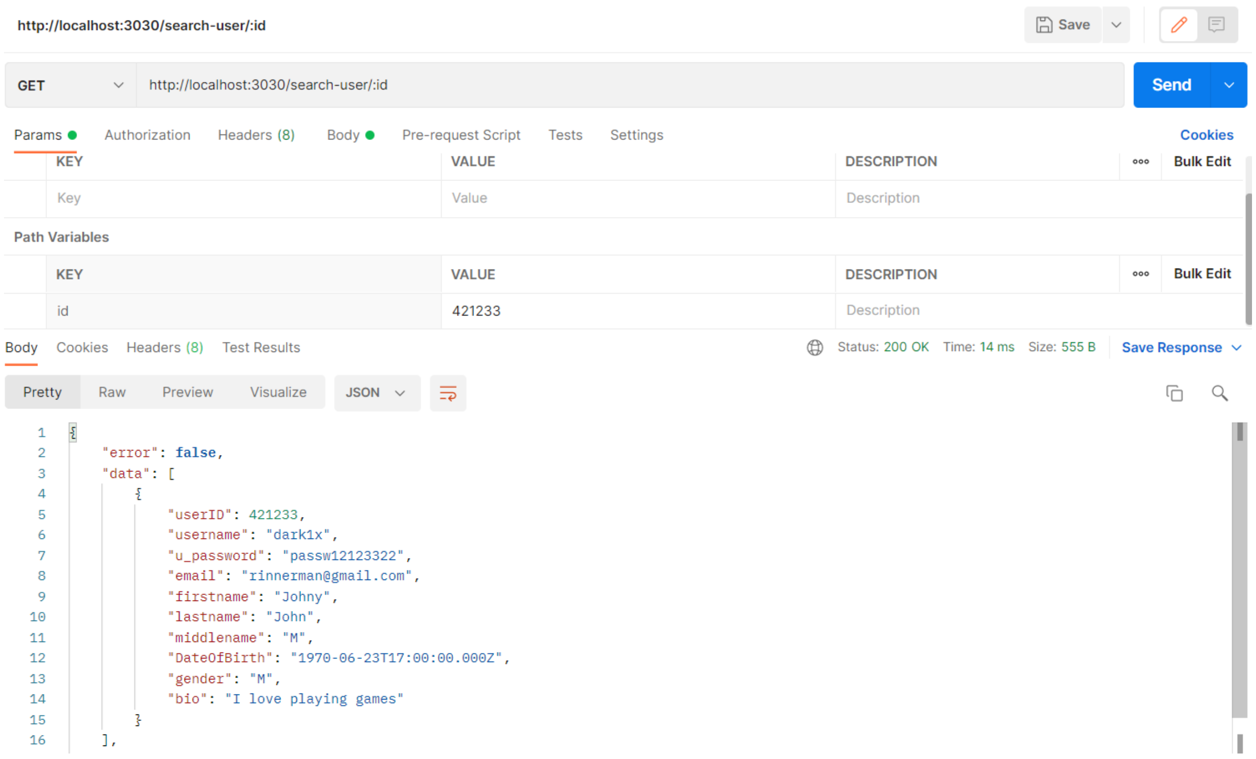
Path Variables

KEY	VALUE	DESCRIPTION
movieID	13651628	Description

Status: 200 OK Time: 15 ms Size: 553 B

```
{
    "error": false,
    "data": [
        {
            "movieID": 13651628,
            "IMDB_rating": "7.30",
            "movieName": "Belle",
            "countryID": 20236,
            "movieDescripiton": "Suzu is a shy high school student living in a rural village. For years, she has only been a shadow of
                herself.",
            "movieType": "movie"
        }
    ],
    "messege": "Movie retrieved"
}
```

Search-user (by id)



GET http://localhost:3030/search-user/:id

KEY	VALUE	DESCRIPTION
Key	Value	Description

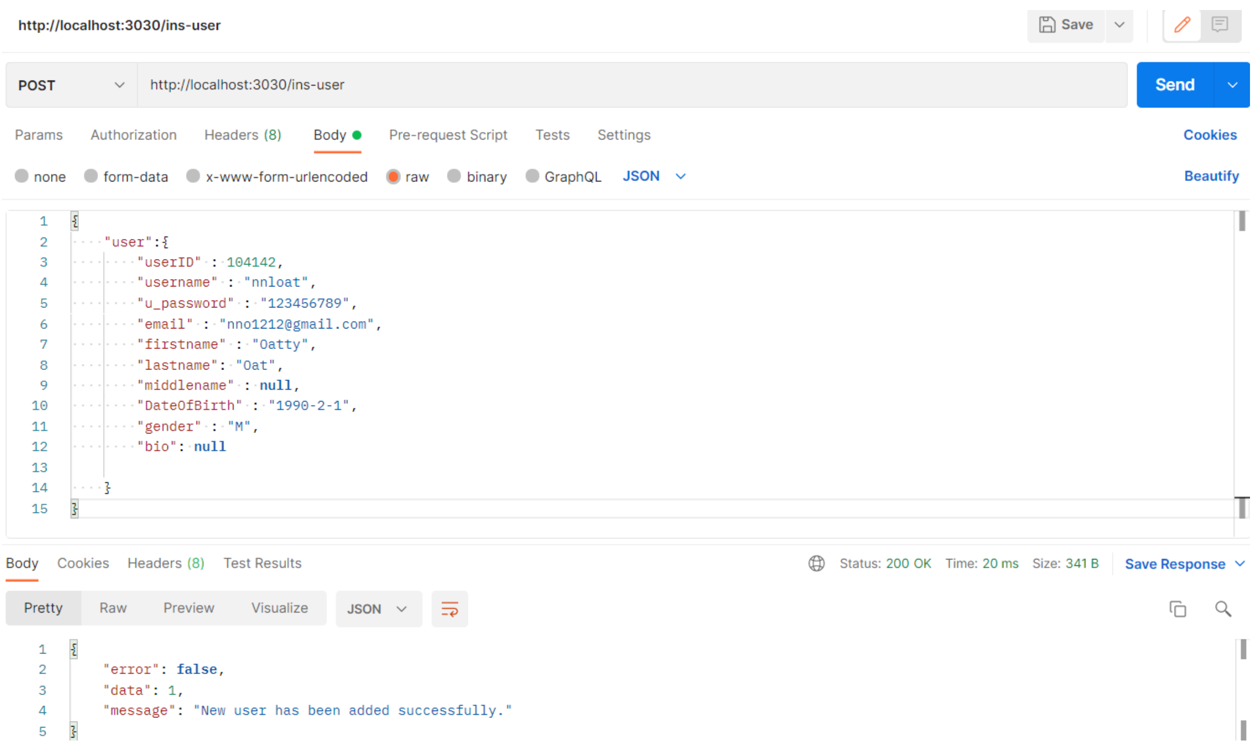
Path Variables

KEY	VALUE	DESCRIPTION
id	421233	Description

Status: 200 OK Time: 14 ms Size: 555 B

```
{
    "error": false,
    "data": [
        {
            "userID": 421233,
            "username": "dark1x",
            "u_password": "passw12123322",
            "email": "rinnerman@gmail.com",
            "firstname": "Johny",
            "lastname": "John",
            "middlename": "M",
            "DateOfBirth": "1970-06-23T17:00:00.000Z",
            "gender": "M",
            "bio": "I love playing games"
        }
    ],
```

Insert-user



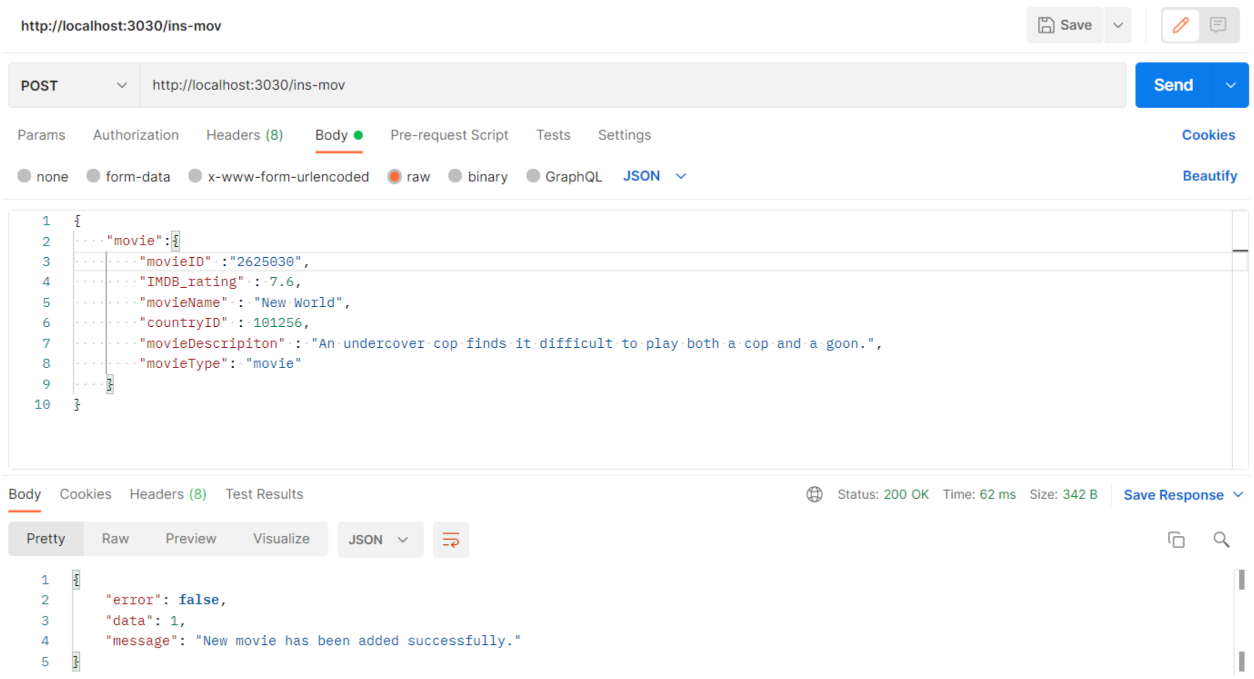
http://localhost:3030/ins-user

```
{
    "user":{
        "userID" : 104142,
        "username" : "nnloat",
        "u_password" : "123456789",
        "email" : "nno1212@gmail.com",
        "firstname" : "Oatty",
        "lastname": "Oat",
        "middlename" : null,
        "DateOfBirth" : "1990-2-1",
        "gender" : "M",
        "bio": null

    }
}
```

```
{
    "error": false,
    "data": 1,
    "message": "New user has been added successfully."
}
```

Insert-movie

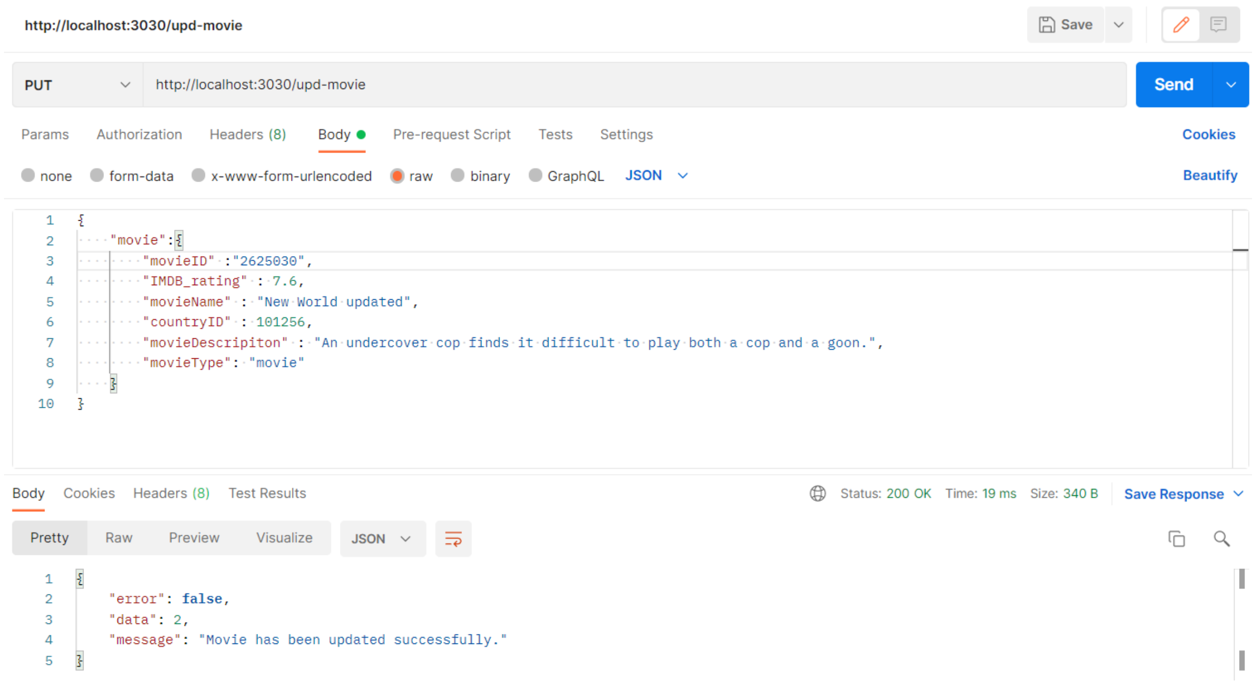


http://localhost:3030/ins-mov

```
{
    "movie":{
        "movieID" :"2625030",
        "IMDB_rating" : 7.6,
        "movieName" : "New World",
        "countryID" : 101256,
        "movieDescripiton" : "An undercover cop finds it difficult to play both a cop and a goon.",
        "movieType": "movie"
    }
}
```

```
{
    "error": false,
    "data": 1,
    "message": "New movie has been added successfully."
}
```

Update-movie



http://localhost:3030/upd-movie

PUT http://localhost:3030/upd-movie

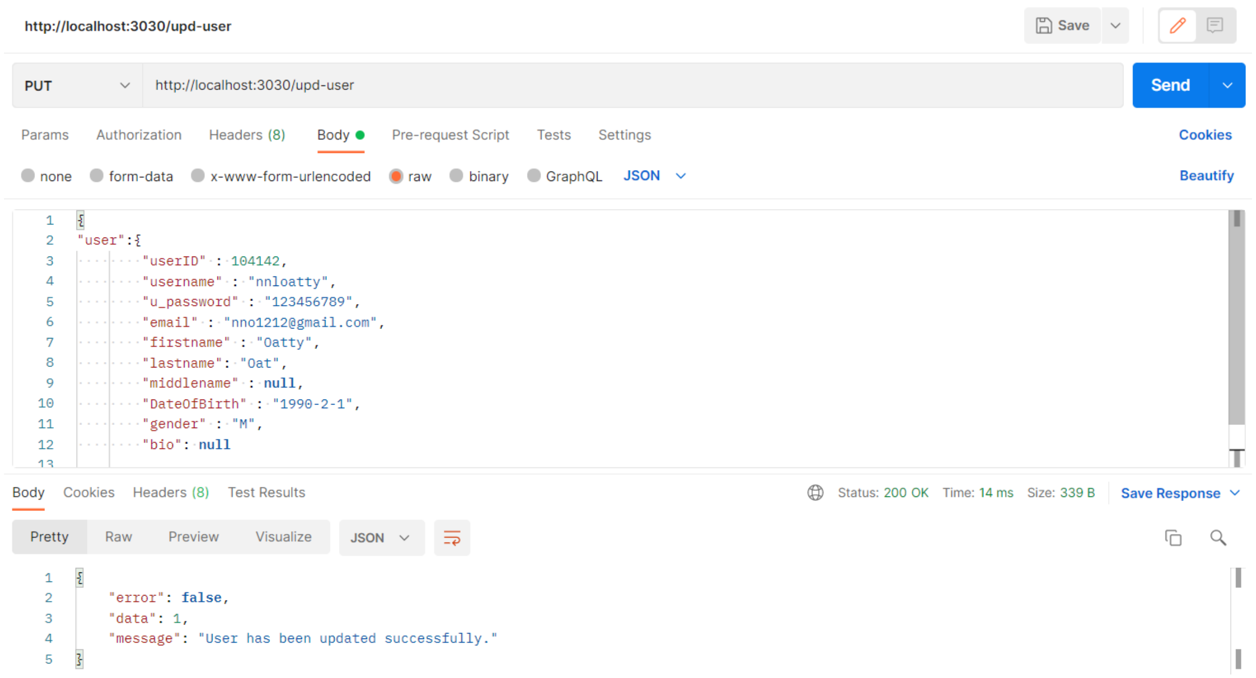
Body: none, form-data, x-www-form-urlencoded, [✓ raw], binary, GraphQL, JSON

```
{
    "movie":{
        "movieID" :"2625030",
        "IMDB_rating" : 7.6,
        "movieName" : "New World updated",
        "countryID" : 101256,
        "movieDescripiton" : "An undercover cop finds it difficult to play both a cop and a goon.",
        "movieType": "movie"
    }
}
```

Status: 200 OK Time: 19 ms Size: 340 B

```
{
    "error": false,
    "data": 2,
    "message": "Movie has been updated successfully."
}
```

Update-user



http://localhost:3030/upd-user

PUT http://localhost:3030/upd-user

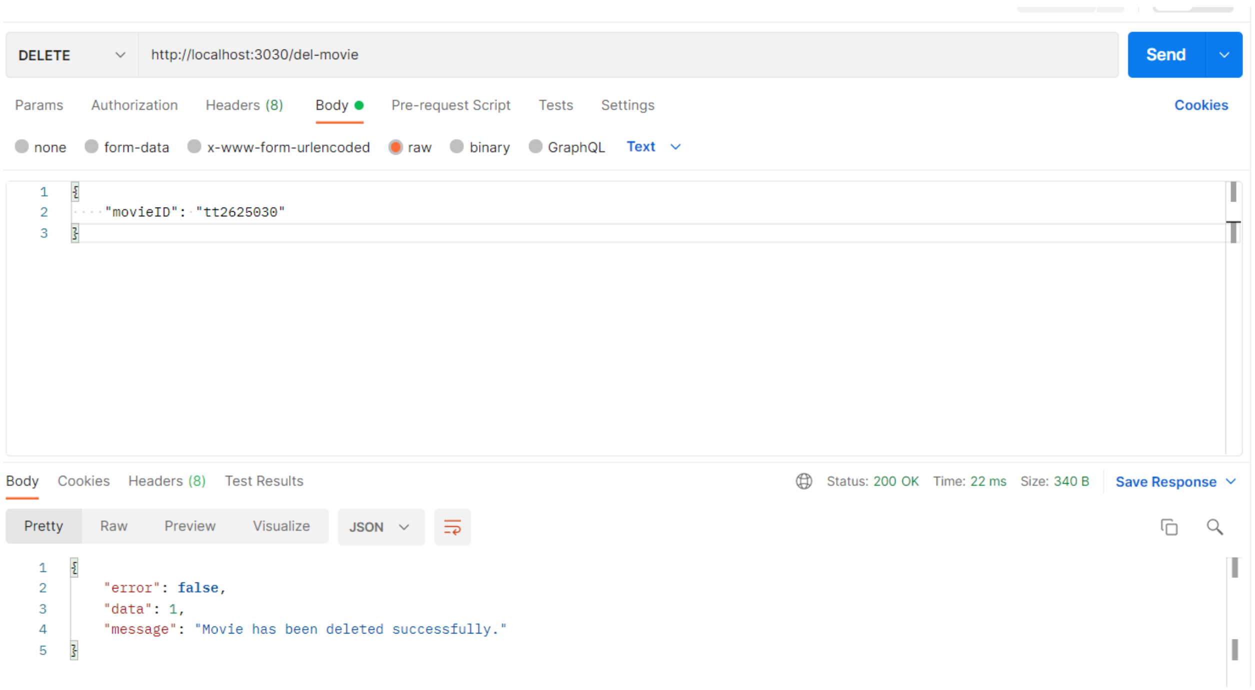
Body: none, form-data, x-www-form-urlencoded, [✓ raw], binary, GraphQL, JSON

```
{
"user":{
        "userID" : 104142,
        "username" : "nnloatty",
        "u_password" : "123456789",
        "email" : "nno1212@gmail.com",
        "firstname" : "Oatty",
        "lastname": "Oat",
        "middlename" : null,
        "DateOfBirth" : "1990-2-1",
        "gender" : "M",
        "bio": null
```

Status: 200 OK Time: 14 ms Size: 339 B

```
{
    "error": false,
    "data": 1,
    "message": "User has been updated successfully."
}
```

Delete movie



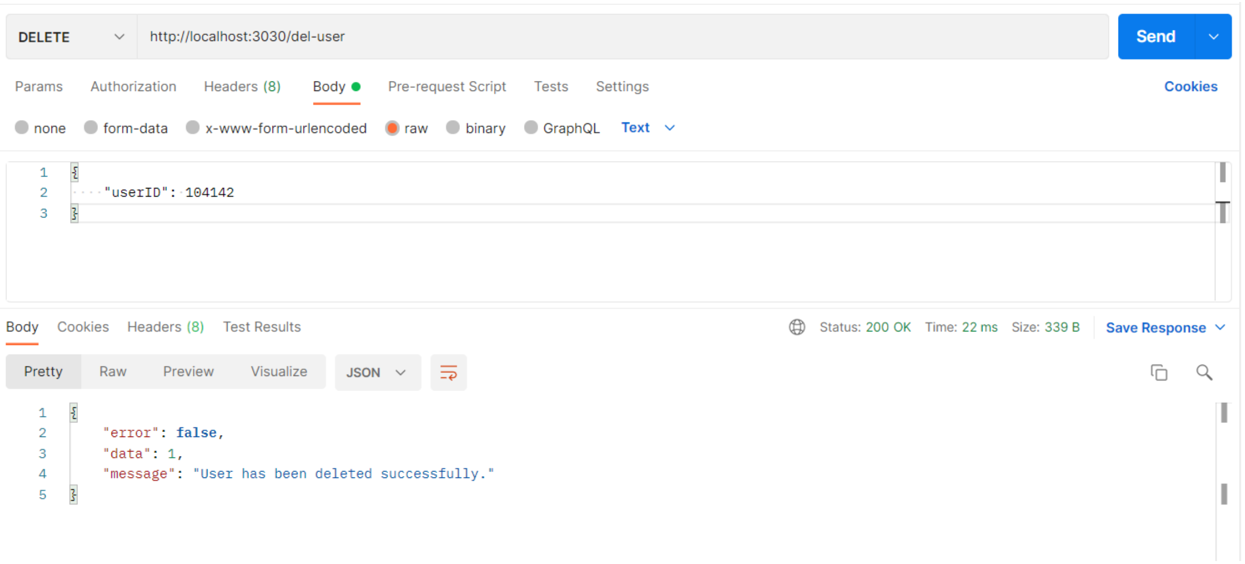
http://localhost:3030/del-movie

```
{
    "movieID": "tt2625030"
}
```

Status: 200 OK Time: 22 ms Size: 340 B

```
{
    "error": false,
    "data": 1,
    "message": "Movie has been deleted successfully."
}
```

Delete user



http://localhost:3030/del-user

```
{
    "userID": 104142
}
```

Status: 200 OK Time: 22 ms Size: 339 B

```
{
    "error": false,
    "data": 1,
    "message": "User has been deleted successfully."
}
```