# My Movie Bot

**J Component Project**
**for**
**ITE3007**
**CLOUD COMPUTING AND VIRTUALIZATION**

***Submitted by***

Aradhya Mathur(17BIT0146)

Kashish Gupta(17BIT0022)

**To**

Prof. Priya V

**In**

**D1+TD1 SLOT**

**S.I.T.E.**
**VIT University, Vellore**
**Tamil Nadu - 632 014**

April 2019

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

# SITE

## J Component Project

## Cloud Computing and Virtualization (ITE3007)

It is certified that the project entitled "*My Movie Bot* " is the bonafide work for J component of Cloud Computing and Virtualization by the following students

**Aradhya Mathur(17BIT0146)**

**Kashish Gupta(17BIT0022)**

of Information Technology under my supervision in D1+TD1 slot during the Winter Semester 2018-19 at V.I.T. University, Vellore-632 014.

Faculty Signature:

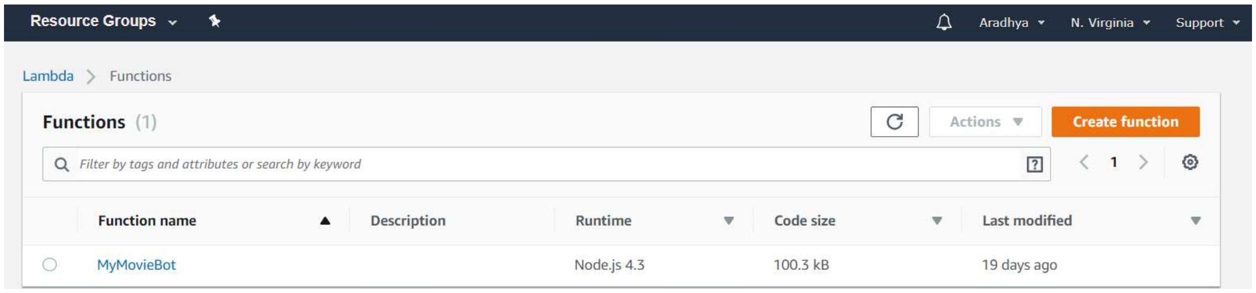
## Abstract

Amazon Lex bot named "My Movie Bot" helps to gather and display information about a movie of our choice. Amazon Lambda function is required and a sample code from AWS documentation is required for creating a function in Node 4.3. Bot is written in NodeJS and it utilizes TMBD which is 'The Movie Data Base" for gathering and returning desired information about a movie. We create a role in IAM (IAM role) with inline policies as given in AWS documentation and edit Trust Relationship.

After changing IAM of the bot with the new IAM role we created, we create slots and Intent of bot. After that we link the Amazon lambda function to our bot. We than build the bot and at last publish the bot.

## Working

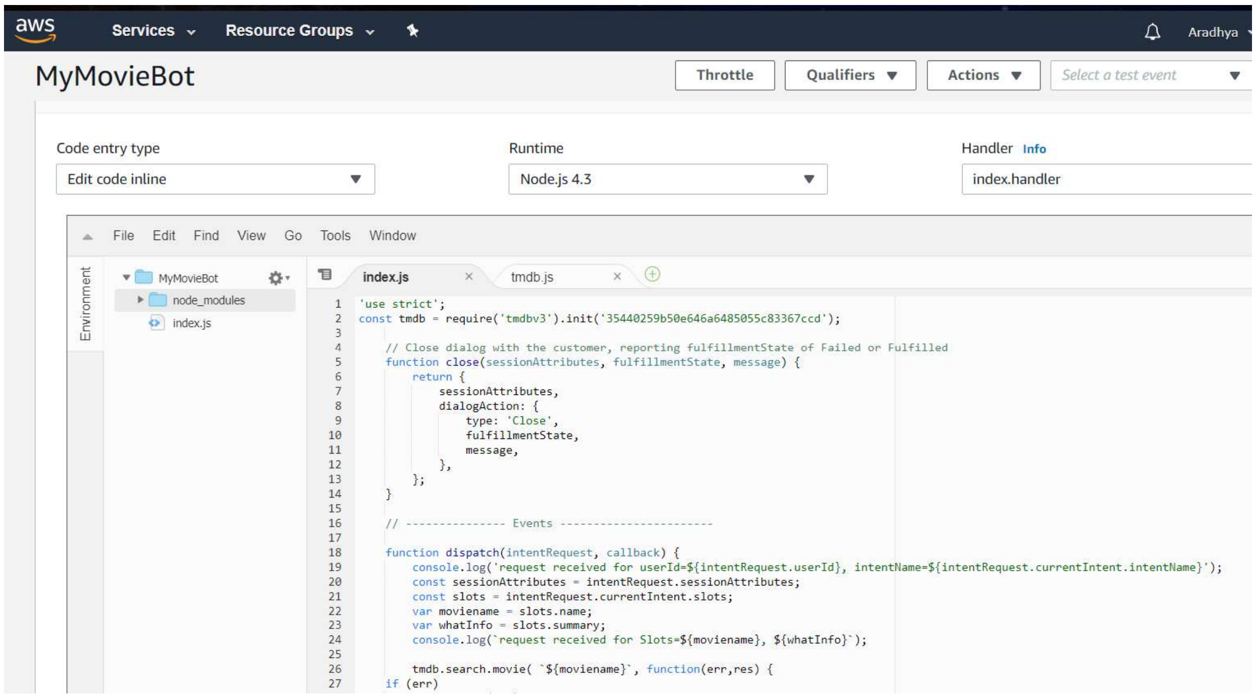
### 1) Amazon Lambda Function



| Function name | Description | Runtime | Code size | Last modified |
|---|---|---|---|---|
| MyMovieBot | | Node.js 4.3 | 100.3 kB | 19 days ago |

#### index.js

```
 'use strict';


const tmdb = require('tmdbv3').init('35440259b50e646a6485055c83367ccd');

   function close(sessionAttributes, fulfillmentState, message) {

      return {

         sessionAttributes,

         dialogAction: {

            type: 'Close',
                fulfillmentState,

                message,

            },

        };

    }

    function dispatch(intentRequest, callback) {

        console.log('request received for userId=${intentRequest.userId},
intentName=${intentRequest.currentIntent.intentName}');

        const sessionAttributes = intentRequest.sessionAttributes;

        const slots = intentRequest.currentIntent.slots;

        var moviename = slots.name;

        var whatInfo = slots.summary;

        console.log(`request received for Slots=${moviename}, ${whatInfo}`);

        tmdb.search.movie( `${moviename}`, function(err,res) {

    if (err)

        console.log(err);

    else

        tmdb.movie.info(res.results[0].id, function(err,res){

          var resPlot = res.overview;

          var resDate = res.release_date;

          console.log(res);

if  (whatInfo === 'Year' || whatInfo === 'Release Date' || whatInfo === 'release date'|| whatInfo === 'date'||
whatInfo === 'date'|| whatInfo === 'Release Year' || whatInfo === 'release year' || whatInfo === 'year'){

        callback(close(sessionAttributes, 'Fulfilled',

        {'contentType': 'PlainText', 'content': ` ${moviename} released in: ${resDate}`}));
    }

else if (whatInfo === 'Plot' || whatInfo === 'Story' || whatInfo === 'plot' || whatInfo === 'story'){

        callback(close(sessionAttributes, 'Fulfilled',

        {'contentType': 'PlainText', 'content': `Plot of ${moviename} is: ${resPlot}`}));

    }

else

        callback(close(sessionAttributes, 'Fulfilled',

        {'contentType': 'PlainText', 'content': `MovieName: ${moviename}, Year: ${resDate}, APlot:
${resPlot}`}));

});});}

    exports.handler = (event, context, callback) => {

        try {

            dispatch(event,

                (response) => {

                    callback(null, response);

                });

        } catch (err) {

            callback(err);

        }

    };
```



Code entry type: Edit code inline

Runtime: Node.js 4.3

Handler: index.handler

```
'use strict';
const tmdb = require('tmdbv3').init('35440259b50e646a6485055c83367ccd');

    // Close dialog with the customer, reporting fulfillmentState of Failed or Fulfilled
    function close(sessionAttributes, fulfillmentState, message) {
        return {
            sessionAttributes,
            dialogAction: {
                type: 'Close',
                fulfillmentState,
                message,
            },
        };
    }

    // --------------- Events -----------------------

    function dispatch(intentRequest, callback) {
        console.log('request received for userId=${intentRequest.userId}, intentName=${intentRequest.currentIntent.intentName}');
        const sessionAttributes = intentRequest.sessionAttributes;
        const slots = intentRequest.currentIntent.slots;
        var moviename = slots.name;
        var whatInfo = slots.summary;
        console.log(`request received for Slots=${moviename}, ${whatInfo}`);

        tmdb.search.movie( `${moviename}`, function(err,res) {
    if (err)
```

## tmdb.js

```
var request = require('request');

String.prototype.format = function() {

    var content = this;

    for (var i=0; i < arguments.length; i++)

    {

        var replacement = '{' + i + '}';

        content = content.replace(replacement, arguments[i]);

    }

    return content;

};



var me;
var tmdb = function(api_key) {

        me = this;

        this.api_key = api_key;

        this.config = null;

        this.base = 'http://api.themoviedb.org/3';

        this.api_urls =

                {

                        configuration:          this.base+'/configuration?api_key='+this.api_key,

                        misc_latest:            this.base+'/latest/movie?api_key='+this.api_key,

        misc_upcoming:          this.base+'/movie/upcoming?page={0}&api_key='+this.api_key,

                        misc_now_playing:       this.base+'/movie/now-
playing?page={0}&api_key='+this.api_key,

                        misc_popular:
this.base+'/movie/popular?page={0}&api_key='+this.api_key,

                        misc_top_rated:         this.base+'/movie/top-
rated?page={0}&api_key='+this.api_key,

                        movie_info:             this.base+'/movie/{0}?api_key='+this.api_key,

                        movie_alternative_titles:
this.base+'/movie/{0}/alternative_titles?api_key='+this.api_key,

                        movie_casts:            this.base+'/movie/{0}/casts?api_key='+this.api_key,

                        movie_images:           this.base+'/movie/{0}/images?api_key='+this.api_key,

                        movie_keywords:
this.base+'/movie/{0}/keywords?api_key='+this.api_key,

                        movie_releases:         this.base+'/movie/{0}/releases?api_key='+this.api_key,

                        movie_trailers:         this.base+'/movie/{0}/trailers?api_key='+this.api_key,

                        movie_translations:
this.base+'/movie/{0}/translations?api_key='+this.api_key,
        movie_similar:          this.base+'/movie/{0}/similar_movies?page={1}&api_key='+this.api_key,

                        person_info:            this.base+'/person/{0}?api_key='+this.api_key,

                        person_credits:          this.base+'/person/{0}/credits?api_key='+this.api_key,

                        person_images:            this.base+'/person/{0}/images?api_key='+this.api_key,

                        collection_info:         this.base+'/collection/{0}?api_key='+this.api_key,

                        search_movie:
    this.base+'/search/movie?query={0}&page={1}&api_key='+this.api_key,

                        search_person:
    this.base+'/search/person?query={0}&page={1}&api_key='+this.api_key,

        search_companies:
    this.base+'/search/company?query={0}&page={1}&api_key='+this.api_key,

                        auth_request_token:
         this.base+'/authentication/token/new?api_key='+this.api_key,

                        auth_session_id:
         this.base+'/authentication/session/new?request_token={0}&api_key='+this.api_key,

                        write_rate_movie:
         this.base+'/movie/{0}/rating?session_id={1}&api_key='+this.api_key,

        company_info:           this.base+'/company/{0}?api_key='+this.api_key,

        company_movies:           this.base+'/company/{0}/movies?api_key='+this.api_key,

        account_info:           this.base+'/account?session_id={0}&api_key='+this.api_key,

        account_add_favorite:
    this.base+'/account/{0}/favorite?session_id={1}&api_key='+this.api_key,

        account_favorite_movies:
    this.base+'/account/{0}/favorite_movies?session_id={1}&api_key='+this.api_key,

        account_add_movie_watchlist:
    this.base+'/account/{0}/movie_watchlist?session_id={1}&api_key='+this.api_key,

        account_movie_watchlist:
    this.base+'/account/{0}/movie_watchlist?session_id={1}&api_key='+this.api_key,

        account_rated_movies:
    this.base+'/account/{0}/rated_movies?session_id={1}&api_key='+this.api_key,
		genre_list:			this.base+'/genre/list?api_key='+this.api_key,

		genre_movies:
this.base+'/genre/{0}/movies?page={0}&page={1}&api_key='+this.api_key

			};



		this.configuration(function(err,res) {

			if(!err) {

				me.config = res;

			} else {

				console.error('Error loading configuration: '+err);

			}

		});

};


/**

 * factory function

 **/

module.exports.init = function(apikey) {

		return new tmdb(apikey);

};



/**

 * misc methods

 * all but the 'latest'-function can be supplied with a page argument,

 * if page is left out the first page is returned
 **/

tmdb.prototype.misc = {

   upcoming: function(p, callback) {

                    if(arguments.length === 1) {

                              callback = p;

                              var page = 1;

                    } else {

                              var page = ((typeof p !== 'number') ? page = 1 : page = p);

                    }

                    var url = me.api_urls.misc_upcoming.format(page);

                    executeQuery({url: url}, callback);

          },

          latest: function(callback) {

                    var url = me.api_urls.misc_latest;

                    executeQuery({url: url}, callback);

          },

          nowPlaying: function(p, callback) {

                    if(arguments.length === 1) {

                              callback = p;

                              var page = 1;

                    } else {

                              var page = ((typeof p !== 'number') ? page = 1 : page = p);

                    }

                    var url = me.api_urls.misc_now_playing.format(page);
```

```
        executeQuery({url: url}, callback);
    },
    popular: function(p, callback) {
        if(arguments.length === 1) {
            callback = p;
            var page = 1;
        } else {
            var page = ((typeof p !== 'number') ? page = 1 : page = p);
        }
        var url = me.api_urls.misc_popular.format(page);
        executeQuery({url: url}, callback);
    },
    topRated: function(p, callback) {
        if(arguments.length === 1) {
            callback = p;
            var page = 1;
        } else {
            var page = ((typeof p !== 'number') ? page = 1 : page = p);
        }
        var url = me.api_urls.misc_top_rated.format(page);
        executeQuery({url: url}, callback);
    },
};
```

```
/**
 * genre methods
 **/
tmdb.prototype.genre = {
    list: function(callback) {
        var url = me.api_urls.genre_list;
        executeQuery({url:url}, callback);
    },

    movies: function(q, p, callback) {
        if(arguments.length === 2) {
            callback = p;
            p = 1;
        } else {
            if (typeof p !== 'number') { p = 1; }
        }
        var url = me.api_urls.genre_movies.format(q,p);
        executeQuery({url:url}, callback);
    }
};

/**
 * Get current configuration for constructing complete image urls
 **/
```

```
tmdb.prototype.configuration = function(callback) {
            var url = me.api_urls.configuration;
            executeQuery({url: url}, callback);
};


/**
 * movie methods
 * q = id (can be either tmdb id or imdb id)
 **/
tmdb.prototype.movie = {
    info: function(q, callback) {
        var url = me.api_urls.movie_info.format(q);
        executeQuery({url:url}, callback);
    },
    alternativeTitles: function(q, callback) {
        var url = me.api_urls.movie_alternative_titles.format(q);
        executeQuery({url:url}, callback);
    },
    casts: function(q, callback) {
        var url = me.api_urls.movie_casts.format(q);
        executeQuery({url:url}, callback);
    },
    images: function(q, callback) {
        var url = me.api_urls.movie_images.format(q);
        executeQuery({url:url}, callback);
    },
    keywords: function(q, callback) {
        var url = me.api_urls.movie_keywords.format(q);
        executeQuery({url:url}, callback);
    },
    releases: function(q, callback) {
        var url = me.api_urls.movie_releases.format(q);
        executeQuery({url:url}, callback);
    },
    trailers: function(q, callback) {
        var url = me.api_urls.movie_trailers.format(q);
        executeQuery({url:url}, callback);
    },
    translations: function(q, callback) {
        var url = me.api_urls.movie_translations.format(q);
        executeQuery({url:url}, callback);
    },
    similar: function(q, p, callback) {
        if(arguments.length === 2) {
            callback = p;
            p = 1;
        } else {
            if (typeof p !== 'number') { p = 1; }
        }
        var url = me.api_urls.movie_similar.format(q,p);
        executeQuery({url:url}, callback);
    }
};

/**
 * search methods
 * q = searchterm, p = page
 * page defaults to 1 if not specified or invalid
 **/
tmdb.prototype.search = {
    movie: function(q, p, callback) {
                        if(arguments.length === 2) {
                                    callback = p;
                                    p = 1;
                        } else {
                                    if (typeof p !== 'number') { p = 1; }
                        }
        var url = me.api_urls.search_movie.format(q, p);
        executeQuery({url:url}, callback);
    },
    person: function(q, p, callback) {
                        if(arguments.length === 2) {
```

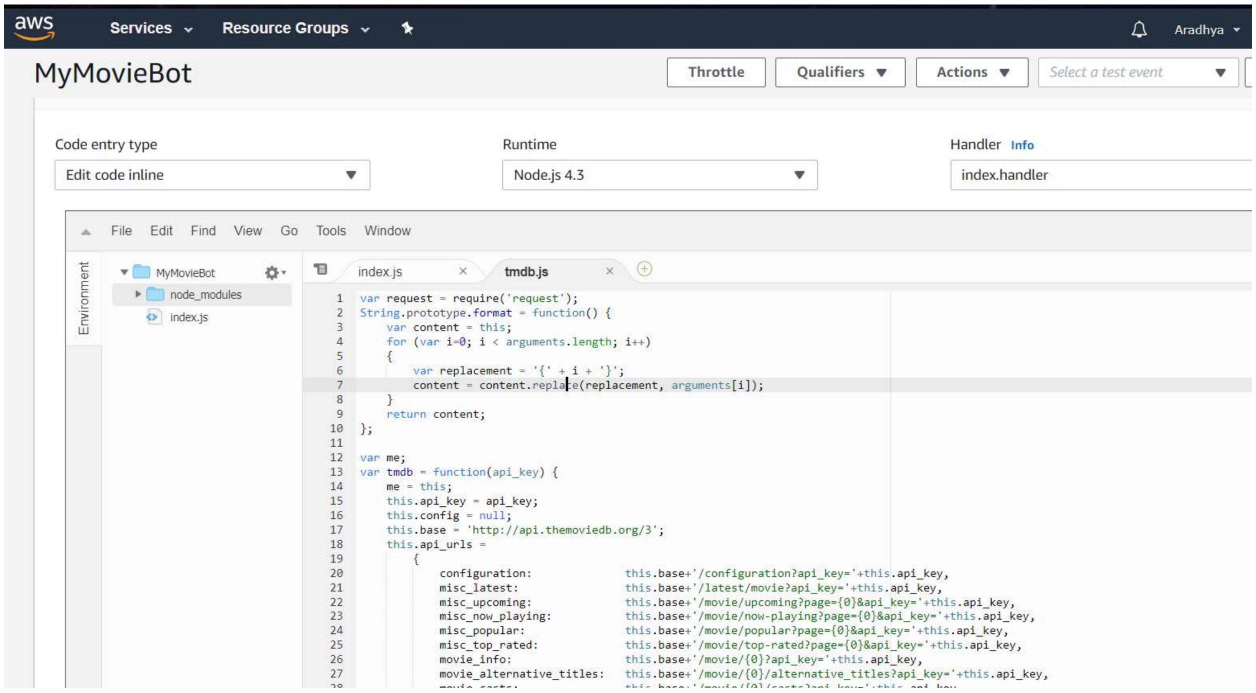
```
                        callback = p;

                        p = 1;

                } else {

                        if (typeof p !== 'number') { p = 1; }

                }

        var url = me.api_urls.search_person.format(q, p);

        executeQuery({url:url}, callback);

          },

    companies: function(q, p, callback) {

                if(arguments.length === 2) {

                        callback = p;

                        p = 1;

                } else {

                        if (typeof p !== 'number') { p = 1; }

                }

        var url = me.api_urls.search_companies.format(q, p);

        executeQuery({url:url}, callback);

    },

};


/**

 * person methods

 * id = person id

 **/
```

```
tmdb.prototype.person = {

    info: function(id, callback) {

        var url = me.api_urls.person_info.format(id);

        executeQuery({url:url}, callback);

    },

    credits: function(id, callback) {

        var url = me.api_urls.person_credits.format(id);

        executeQuery({url:url}, callback);

    },

    images: function(id, callback) {

        var url = me.api_urls.person_images.format(id);

        executeQuery({url:url}, callback);

    },

};


/**

 * collection methods

 * id = collection id

 **/

tmdb.prototype.collection = {

    info: function(id, callback) {

        var url = me.api_urls.collection_info.format(id);

        executeQuery({url: url}, callback);

    },
};


/**
 * authentication methods
 **/
tmdb.prototype.authentication = {
		requestToken: function(callback) {
				var url = me.api_urls.auth_request_token;
				executeQuery({url:url}, callback);
		},
		sessionId: function(token, callback) {
				var url = me.api_urls.auth_session_id.format(token);
				executeQuery({url:url}, callback);
		}
};


/**
 * company methods
 **/
tmdb.prototype.company = {
    info: function(id, callback) {
        var url = me.api_urls.company_info.format(id);
        executeQuery({url: url}, callback);
    },
    movies: function(id, callback) {
        var url = me.api_urls.company_movies.format(id);
        executeQuery({url: url}, callback);
    },
};

/**
 * account methods
 **/
tmdb.prototype.account = {
    info: function(sid, callback) {
        var url = me.api_urls.account_info.format(sid);
        executeQuery({url: url}, callback);
    },
    favorite_movies: function(id, sid, callback) {
        var url = me.api_urls.account_favorite_movies.format(id,sid);
        executeQuery({url: url}, callback);
    },
    rated_movies: function(id, sid, callback) {
        var url = me.api_urls.account_rated_movies.format(id,sid);
        executeQuery({url: url}, callback);
    },
    add_favorite: function(aid, mid, sid, isfavorite, callback) {
        var url = me.api_urls.account_add_favorite.format(aid,sid);
        executePost({url: url},{movie_id: mid, favorite: isfavorite}, callback);

    },

    movie_watchlist: function(id, sid, callback) {

        var url = me.api_urls.account_movie_watchlist.format(id,sid);

        executeQuery({url: url}, callback);

    },

    add_movie_watchlist: function(aid, mid, sid, isinwatchlist, callback) {

        var url = me.api_urls.account_add_movie_watchlist.format(aid,sid);

        executePost({url: url},{movie_id: mid, movie_watchlist: isinwatchlist}, callback);

    },

};




/**

 * write methods

 * id = item id (movie id etc.)

 * sid = session id

 **/

tmdb.prototype.write = {

            rateMovie: function(id, sid, rating, callback) {

                        var url = me.api_urls.write_rate_movie.format(id,sid);

                        executePost({url:url}, {value:rating}, callback);

            },

};
```

```
/**
 * Sends the query to tmdb and ships the response of to be processed
 **/
var executeQuery = function(url,callback) {
    request(
        {
            uri:encodeURI(url.url),
            headers: {"Accept": 'application/json'}
        },
        function(err,res,body) {
            processQuery(url,err,res,body,callback);
        }
    );
}

/**
 * Processes the query response from TMDb
 **/
var processQuery = function(url, error, response, body, callback) {
    var res = null;
    try {
        res = JSON.parse(body);
    } catch(e) {
}

if(!error && response.statusCode === 200 && !res.status_code) {
  callback(undefined,res);
  return;
}

     if(res.status_code) {
              switch(res.status_code) {
                       case 6: // Invalid id
                                callback(res,undefined);
                                break;
                       case 7: // Invalid API key
                                callback(res,undefined);
                                break;
                       case 10: // API key suspended, not good
                                callback(res,undefined);
                                break;
                       case 12: // The item/record was updated successfully
                                callback(res,undefined);
                                break;
                       case 17: // Session denied
                                callback(res,undefined);
                                break;
```

```
                    }
            } else {
                    callback(error,res);
            }
    }

    /**
     * posts the data to TMDb and ships the response of to be processed
     **/
    var executePost = function(url,data,callback) {
      request(
        {
                            method: 'POST',
          uri:encodeURI(url.url),
                            json: data,
          headers: {"Content-Type": 'application/json', "Accept": 'application/json'},
        },
        function(err,res,body) {
          processPost(url,err,res,body,callback);
        }
      );
    }
    var processPost = function(url, error, response, body, callback) {
      var res = null;
```

```
try {
   res = body;
} catch(e) {
}
if(!error && response.statusCode === 200 && !res.status_code) {
   callback(undefined,res);
   return;
}
       if(res.status_code) {
                 switch(res.status_code) {
      case 5: // not valid json supplied
         callback(res,undefined);
         break;
                              case 6: // Invalid id
                                        callback(res,undefined);
                                        break;
                              case 7: // Invalid API key
                                        callback(res,undefined);
                                        break;
                              case 10: // API key suspended, not good
                                        callback(res,undefined);
                                        break;
                              case 12: // The item/record was updated successfully
                                        callback(undefined,res);
                                        break;

                case 13: // The item/record was deleted successfully

                    callback(undefined,res);

                    break;

                                case 17: // Session denied

                                        callback(res,undefined);

                                        break;

                        }

            } else {

                        callback(error,res);

            }

}
```

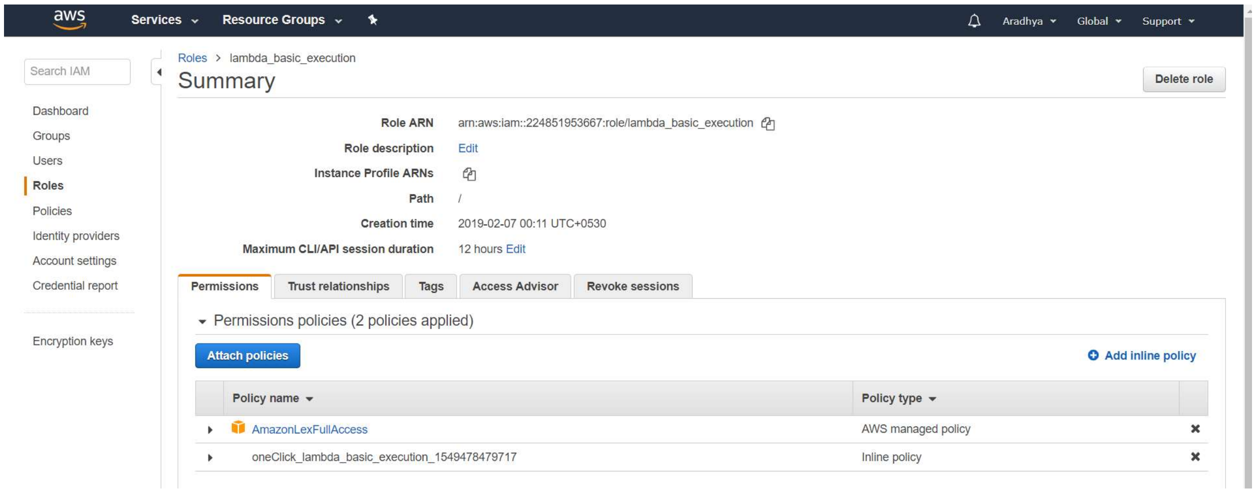


Code entry type: Edit code inline
Runtime: Node.js 4.3
Handler Info: index.handler

```
var request = require('request');
String.prototype.format = function() {
    var content = this;
    for (var i=0; i < arguments.length; i++)
    {
        var replacement = '{' + i + '}';
        content = content.replace(replacement, arguments[i]);
    }
    return content;
};

var me;
var tmdb = function(api_key) {
    me = this;
    this.api_key = api_key;
    this.config = null;
    this.base = 'http://api.themoviedb.org/3';
    this.api_urls =
        {
            configuration:              this.base+'/configuration?api_key='+this.api_key,
            misc_latest:                this.base+'/latest/movie?api_key='+this.api_key,
            misc_upcoming:              this.base+'/movie/upcoming?page={0}&api_key='+this.api_key,
            misc_now_playing:           this.base+'/movie/now-playing?page={0}&api_key='+this.api_key,
            misc_popular:               this.base+'/movie/popular?page={0}&api_key='+this.api_key,
            misc_top_rated:             this.base+'/movie/top-rated?page={0}&api_key='+this.api_key,
            movie_info:                 this.base+'/movie/{0}?api_key='+this.api_key,
            movie_alternative_titles:   this.base+'/movie/{0}/alternative_titles?api_key='+this.api_key,
```

## 2) IAM ROLES

Changed inline policies (permission) and edited Trust relationships



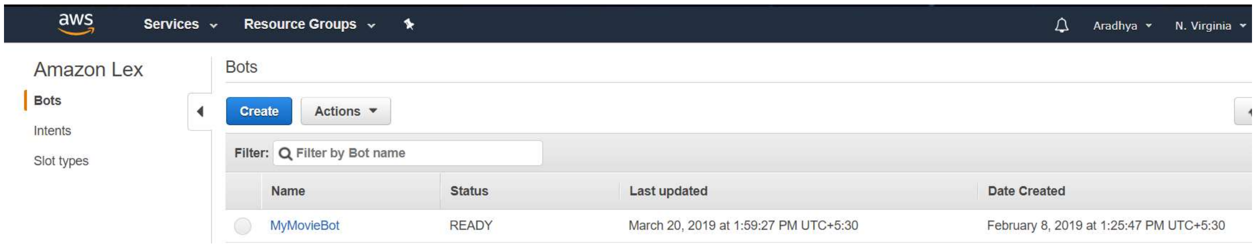
Summary

| | |
|---|---|
| Role ARN | arn:aws:iam::224851953667:role/lambda_basic_execution |
| Role description | Edit |
| Instance Profile ARNs | |
| Path | / |
| Creation time | 2019-02-07 00:11 UTC+0530 |
| Maximum CLI/API session duration | 12 hours Edit |

Permissions policies (2 policies applied)

| Policy name | Policy type |
|---|---|
| AmazonLexFullAccess | AWS managed policy |
| oneClick_lambda_basic_execution_1549478479717 | Inline policy |

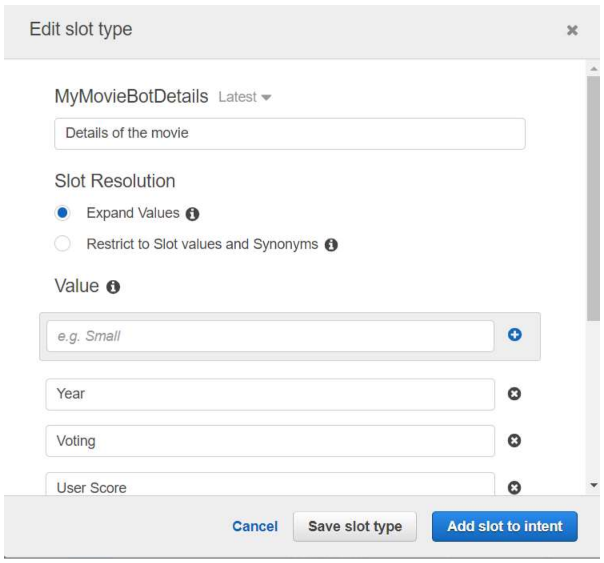
## 3) AWS Lex Bot



Bots

| Name | Status | Last updated | Date Created |
|---|---|---|---|
| MyMovieBot | READY | March 20, 2019 at 1:59:27 PM UTC+5:30 | February 8, 2019 at 1:25:47 PM UTC+5:30 |

## Slots



MyMovieBotDetails

Details of the movie

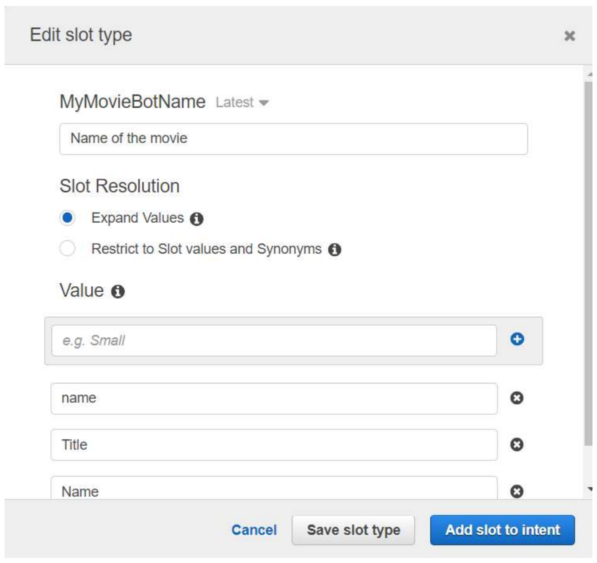
Slot Resolution: [✓ Expand Values] Restrict to Slot values and Synonyms

Value

Year

Voting

User Score



MyMovieBotName

Name of the movie

Slot Resolution: [✓ Expand Values] Restrict to Slot values and Synonyms

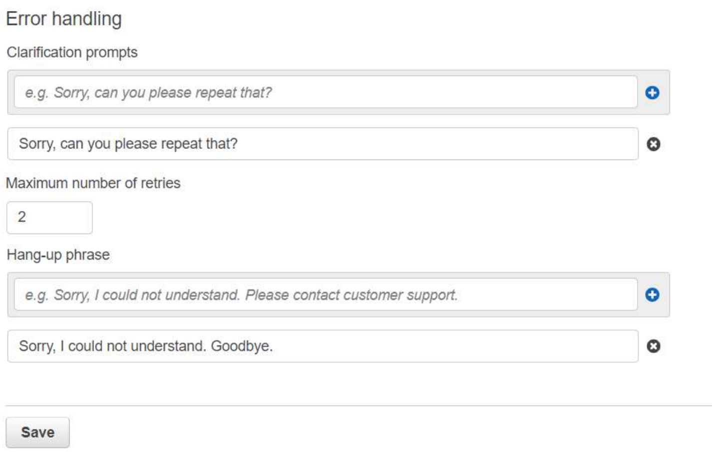
Value

name

Title

Name

## Error Handling



Error handling

Clarification prompts

e.g. Sorry, can you please repeat that?

Sorry, can you please repeat that?

Maximum number of retries

2

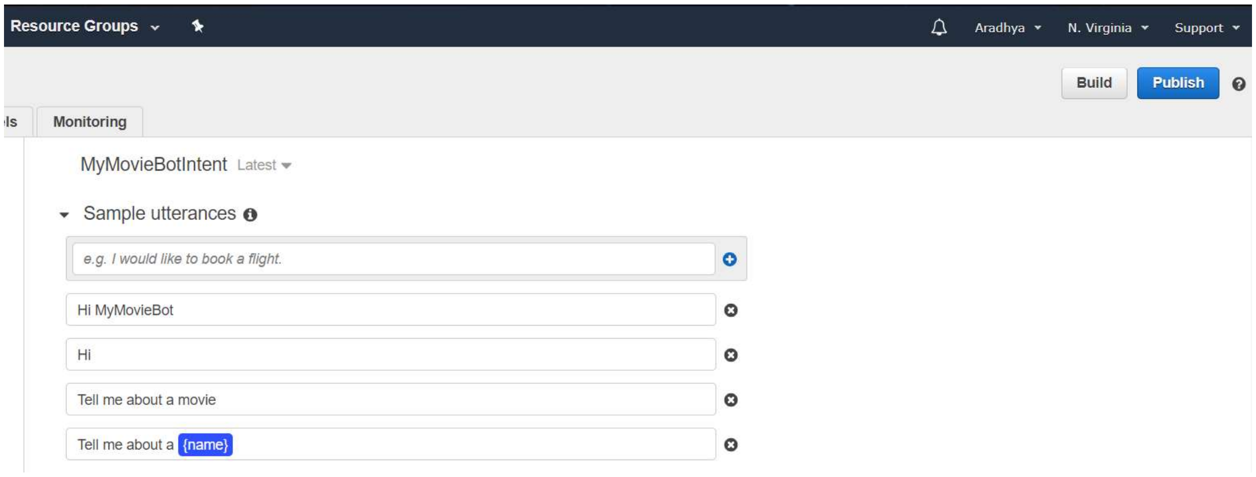
Hang-up phrase

e.g. Sorry, I could not understand. Please contact customer support.

Sorry, I could not understand. Goodbye.

Save

## Intent



Resource Groups | Aradhya | N. Virginia | Support

Build | Publish

ls | Monitoring

MyMovieBotIntent Latest

Sample utterances

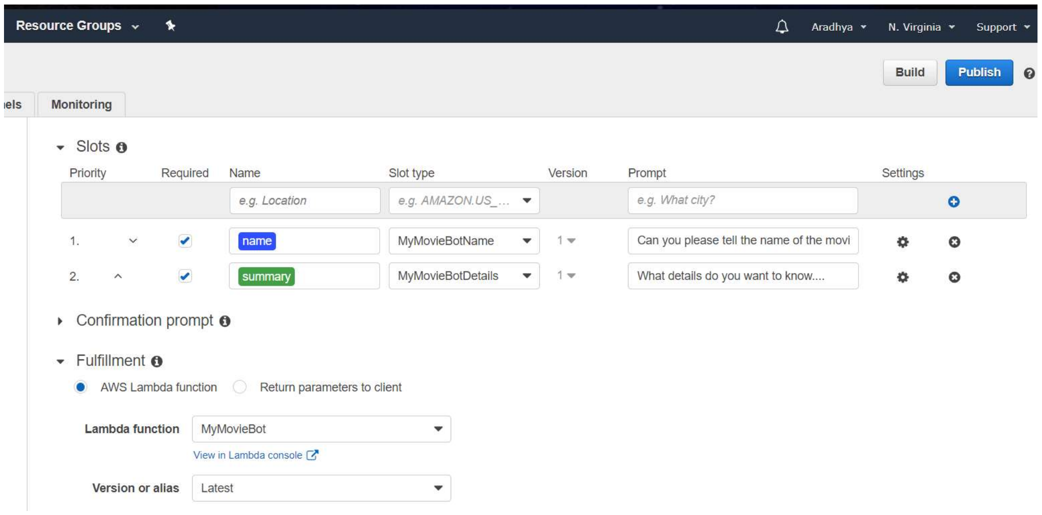
e.g. I would like to book a flight.

Hi MyMovieBot

Hi

Tell me about a movie

Tell me about a {name}



Resource Groups | Aradhya | N. Virginia | Support

Build | Publish

els | Monitoring

Slots

| Priority | Required | Name | Slot type | Version | Prompt | Settings |
|---|---|---|---|---|---|---|
| | | e.g. Location | e.g. AMAZON.US_... | | e.g. What city? | |
| 1. | ✓ | name | MyMovieBotName | 1 | Can you please tell the name of the movi | |
| 2. | ✓ | summary | MyMovieBotDetails | 1 | What details do you want to know.... | |

Confirmation prompt

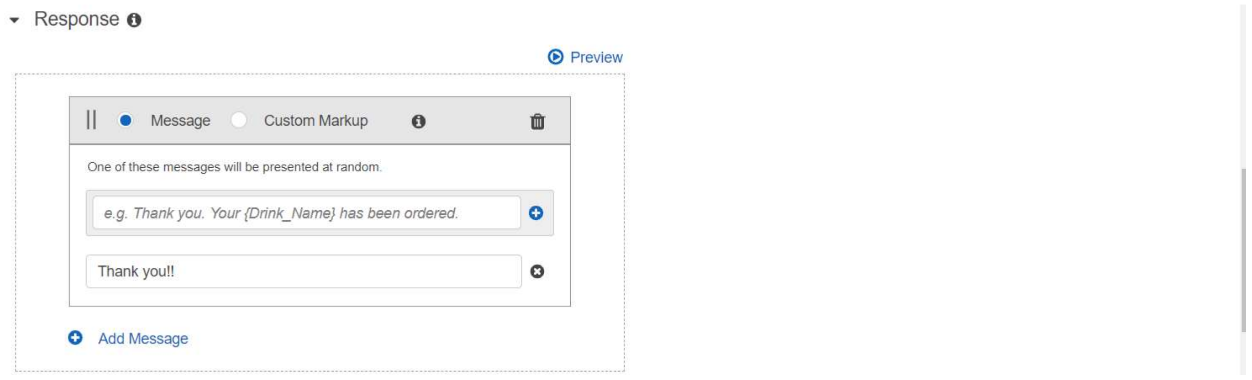
Fulfillment

● AWS Lambda function ○ Return parameters to client

Lambda function: MyMovieBot

View in Lambda console

Version or alias: Latest



Response

Preview

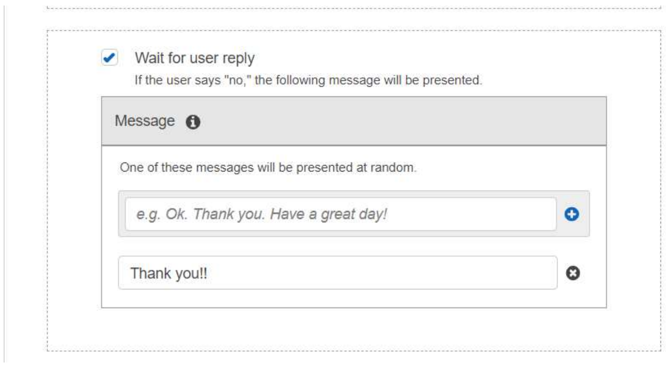
Message [✓] Custom Markup

One of these messages will be presented at random.

e.g. Thank you. Your {Drink_Name} has been ordered.

Thank you!!

Add Message



[✓] Wait for user reply

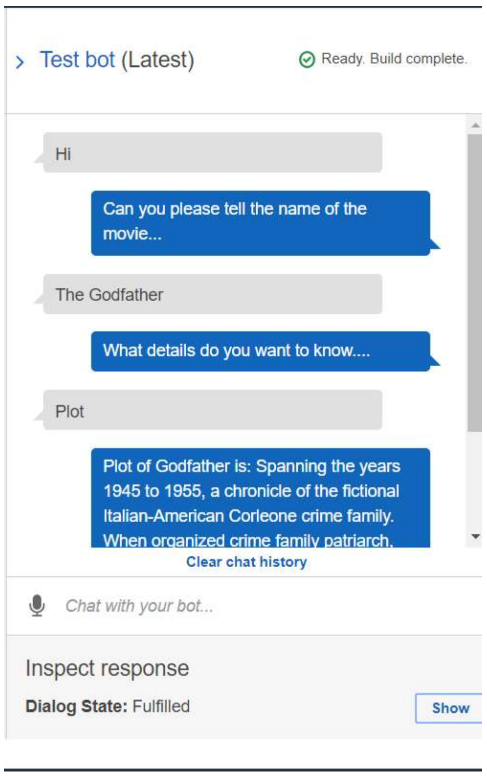
If the user says "no," the following message will be presented.

Message

One of these messages will be presented at random.

e.g. Ok. Thank you. Have a great day!

Thank you!!



Test bot (Latest) Ready. Build complete.

Hi

Can you please tell the name of the movie...

The Godfather

What details do you want to know....

Plot

Plot of Godfather is: Spanning the years 1945 to 1955, a chronicle of the fictional Italian-American Corleone crime family. When organized crime family patriarch,

Clear chat history

Chat with your bot...

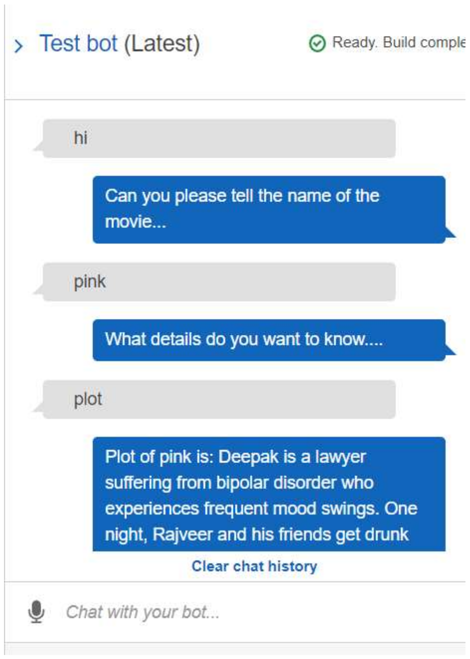
Inspect response

**Dialog State:** Fulfilled

Show

## Result and Conclusion

The MyMovieBot works and provides plot and year of the movies. In includes all the movies currently in the database of TMDB website. Bot works in text as well as speech mode.



hi

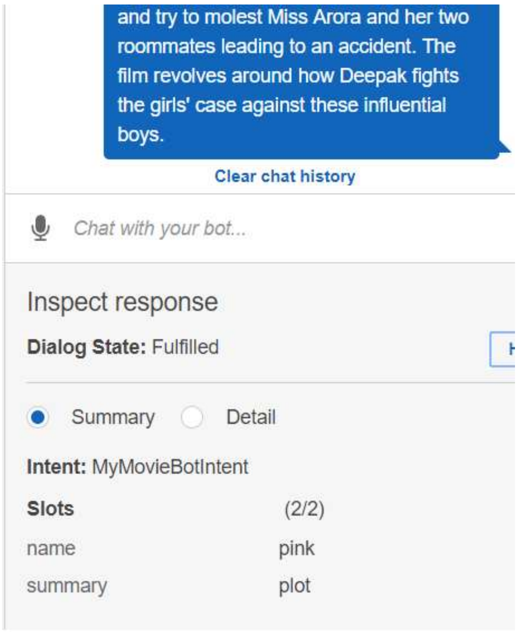
Can you please tell the name of the movie...

pink

What details do you want to know....

plot

Plot of pink is: Deepak is a lawyer suffering from bipolar disorder who experiences frequent mood swings. One night, Rajveer and his friends get drunk



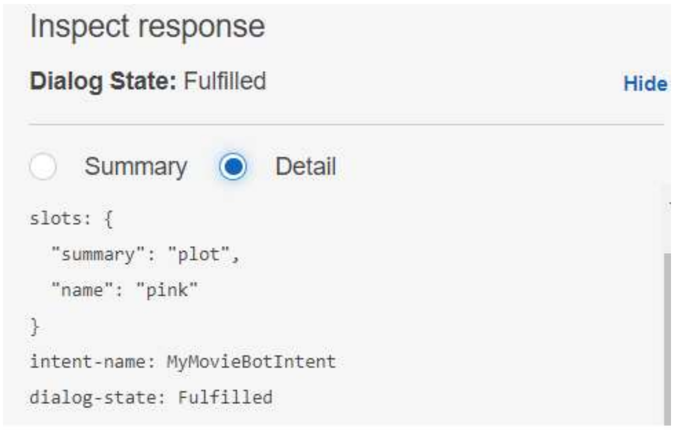
and try to molest Miss Arora and her two roommates leading to an accident. The film revolves around how Deepak fights the girls' case against these influential boys.

**Dialog State:** Fulfilled

[✓ Summary] Detail

**Intent:** MyMovieBotIntent

| **Slots** | (2/2) |
|---|---|
| name | pink |
| summary | plot |



**Dialog State:** Fulfilled

Summary [✓ Detail]

```
slots: {
  "summary": "plot",
  "name": "pink"
}
intent-name: MyMovieBotIntent
dialog-state: Fulfilled
```

## References

https://docs.aws.amazon.com/