```
// Pulkit Agarwal (19323939)
// CSU44000- INTERNET APPLICATIONS 2020-21
// Assignment-2 (Cloud Application Database)

const AWS = require("aws-sdk");
const express = require("express")
const path = require("path")

const app = express()
// const publicKey = "AKIAXSRFFSH5EKGZQ5LE"
// const privateKey = "RMICI4FXFLbQCEVrQVrlVB14llz7m4C5oq7otqzU"

let publicPath = path.resolve(__dirname, "public")
app.use(express.static(publicPath))

app.get("/", function (req, res) {
    res.sendFile(path.join(__dirname + "/index.html"))
})

const PORT = 3000;
app.listen(PORT, () => console.log(`Server started on PORT: ${PORT} \n`));

AWS.config.update({
    // accessKeyId: publicKey,
    // secretAccessKey: privateKey,
    // region: "eu-west-1",
    region: "us-east-1",
});

var dynamodb = new AWS.DynamoDB();
var docClient = new AWS.DynamoDB.DocumentClient();

app.post('/createDatabase', (req, res) => {
    console.log("Creating Table...")
    var params = {
        TableName: "Movies",
        KeySchema: [
            //Partition key
            { AttributeName: "year", KeyType: "HASH" },
            //Sort key
            { AttributeName: "title", KeyType: "RANGE" }
        ],
        AttributeDefinitions: [
            { AttributeName: "year", AttributeType: "N" },
            { AttributeName: "title", AttributeType: "S" }
        ],
        ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
        }
    };
    dynamodb.createTable(params, function (err, data) {
        if (err) {
            console.error("Unable to create table. Error JSON:", JSON.stringify(err,
null, 2));
        } else {
            console.log("Table Created. Table description JSON:",
JSON.stringify(data, null, 2));
```

```
        }
    });

    var s3BucketParams = {
        Bucket: 'csu44000assign2useast20',
        Key: 'moviedata.json'
    }

let promise = new Promise((res) =>{
    setTimeout(() => {
    var s3 = new AWS.S3();
    s3.getObject(s3BucketParams, function (err, data) {
        if (err) {
            console.log(err, err.stack);
        } else {
            var allMovies = JSON.parse(data.Body.toString());
             allMovies.forEach(function (movie) {
                //console.log(allMovies);
                var tableParams = {
                    TableName: "Movies",
                    Item: {
                        "title": movie.title,
                        "year": movie.year,
                        "release": new Date(movie.info.release_date).toDateString(),
                        "director":  movie.info.directors,
                        "rating": movie.info.rating,
                        "rank": movie.info.rank,
                        "cast": movie.info.actors,
                        "plot": movie.info.plot,
                        "runtime": movie.info.running_time_secs,
                        "poster": movie.info.image_url,
                    }

                };


                docClient.put(tableParams, function (err) {
                    if (err) {
                        console.error("\t\t\t\tError. Can't add Movie: ",
movie.title);
                    } else {
                        console.log("Added Movie:", movie.title);
                    }

                });

            });
        }
    })
},5000);
});
});
app.post('/queryDatabase/:title/:year', (req, res) => {
    var queryArray = {
        movieList :[]
    }
    var year = parseInt(req.params.year)
    var title = req.params.title
    var params = {
```

```
        TableName : "Movies",
        ProjectionExpression:"#yr, title, director, rating, #r, #release, #rt, #cast, plot",
        KeyConditionExpression: "#yr = :yyyy and begins_with (title, :letter1)",
        ExpressionAttributeNames:{
            "#yr": "year",
            "#r":"rank",
            "#release":"release",
            "#rt": "runtime",
            "#cast": "cast"
        },
        ExpressionAttributeValues: {
            ":yyyy": year,
            ":letter1": title
        }
    };

    docClient.query(params, function(err, data) {
        if (err) {
            console.log("Unable to query. Error:", JSON.stringify(err, null, 2));
        } else {
            data.Items.forEach(function(item) {
                console.log("Query Request Data: " + '\n' + item.title +'\n'+ item.year+'\n' + item.director+'\n' + item.rating + '\n' + item.plot + '\n' + item.runtime + '\n' + item.cast);
                var hours = (Math.floor(item.runtime/3600) %24)
                var minutes = Math.floor(item.runtime/60) %60;
                var queryYear = item.year
                var queryTitle = item.title
                var queryDirector = item.director
                var queryRating = item.rating
                var queryRank = item.rank
                var queryRelease = item.release
                var queryRuntime = hours + "hr " + minutes+ "min"
                var queryCast = item.cast
                var queryPlot = item.plot
                var queryPoster = item.poster

                queryArray.movieList.push(
                    {
                        Title: queryTitle,
                        Year : queryYear,
                        Director: queryDirector,
                        Rating: queryRating,
                        Rank: queryRank,
                        Release: queryRelease,
                        Runtime: queryRuntime,
                        Cast: queryCast,
                        Plot: queryPlot,
                        Poster: queryPoster,
                    }
                )
            });
            res.json(queryArray)
        }
    });
});

app.post('/deleteDatabase', (req, res) => {
```

```
    console.log("Deleting table...");
    var params = {
        TableName : "Movies",
    };
    dynamodb.deleteTable(params, function(err, data) {
        if (err) {
            console.error("Unable to delete table. Error JSON:", JSON.stringify(err,
null, 2));
        } else {
            console.log("Successfully deleted table 'Movies'");
        }

    });
});
```