Microservice Gateway: https://moviesearch.azurewebsites.net/

When creating this movie search microservice, we used Node.js, Microsoft Azure App Services, and Docker for seamless CI/CD integration. The movie information was procured courtesy of this Kaggle dataset. To make the movie accessible from the microservice, we created a MongoDB database, which the microservice connects to serve the query. The app only handles GET requests for the movie search and POST requests for the /predict function, which fetches the movie prediction created by the Vietnam team.

Accessing the prediction model created by the Vietnam team had to be done through a proxy microservice in this manner, because the prediction API does not have CORS enabled. An additional function was also needed to send the request to the prediction API, as the entire output from the movie search was not needed. This, along with other major parts of the microservice, can be found below.

MongoDB connection

```
const { MongoClient } = require('mongodb');
const mongoUri = "mongodb+srv://----------------------";// connection url
const client = new MongoClient(mongoUri);
let db; // Database reference
// Connect to MongoDB
(async()=>{
    try{
        await client.connect();
        db = client.db("coil");
        console.log("Connected to mongoDB cluster - coil database");
    } catch (error){
        console.error("MongoDB connection error:", error);
        process.exit(1);
    }
})()
```

Movie Search Path:

```
app.get('/:movie', async (req, res) => {
    try {
        console.log('/:movie, movie=', req.params.movie); //Debug info
        let movieRegEx = new RegExp(req.params.movie, 'i');
        const results = await db.collection("movies")
            .find({ title: movieRegEx })
            .toArray();
        console.log('Movie search results:', results);
        res.json(results);
    } catch (error){
        console.error('Movie seach error:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

Prediction Fetch Path:

```
app.post('/predict', async (req, res) => {
    try {
        const tempInput = buildInput(req.body);
        const predictionInput = Array.isArray(tempInput) ? tempInput[0] :
tempInput;
        const predictionRes = await
fetch('https://data-mining-t89i.onrender.com/movie-rating-prediction/api/v
1/movies/predict-rating', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(predictionInput)
        });

        if (!predictionRes.ok) {
            console.error('Prediction API error:', await
predictionRes.text());
            return res.status(500).json({ error: 'Prediction service
failed' });
        }

        const predictionData = await predictionRes.json();
        res.json(predictionData);
    } catch (error) {
        console.error('Prediction route error:', error);
        res.status(500).json({ error: 'Internal Server Error' });
    }
});
```

Build Input (used in /predict):

```
function buildInput(input){
    let result = [
        {
            "id": input.id,
            "voteCount": input.vote_count,
            "revenue": input.revenue,
            "runtime": input.runtime,
            "budget": input.budget,
            "popularity": input.popularity,
            "numVotes": input.numVotes,
            "title": input.title,
            "status": input.status,
            "releaseDate": input.release_date,
            "adult": input.adult,
            "tconst": input.tconst,
            "originalLanguage": input.original_language,
            "originalTitle": input.original_title,
            "genres": input.genres,
            "keywords": input.keywords,
            "directors": input.directors,
            "writers": input.writers,
            "cast": input.cast,
            "backdropPath": input.backdrop_path,
            "homepage": input.homepage,
            "overview": input.overview,
            "posterPath": input.poster_path,
            "tagline": input.tagline,
            "productionCompanies": input.production_companies,
            "productionCountries": input.production_countries,
            "spokenLanguages": input.spoken_languages,
            "voteAverage": input.averageRating
        }
    ]
    return result;
}
```