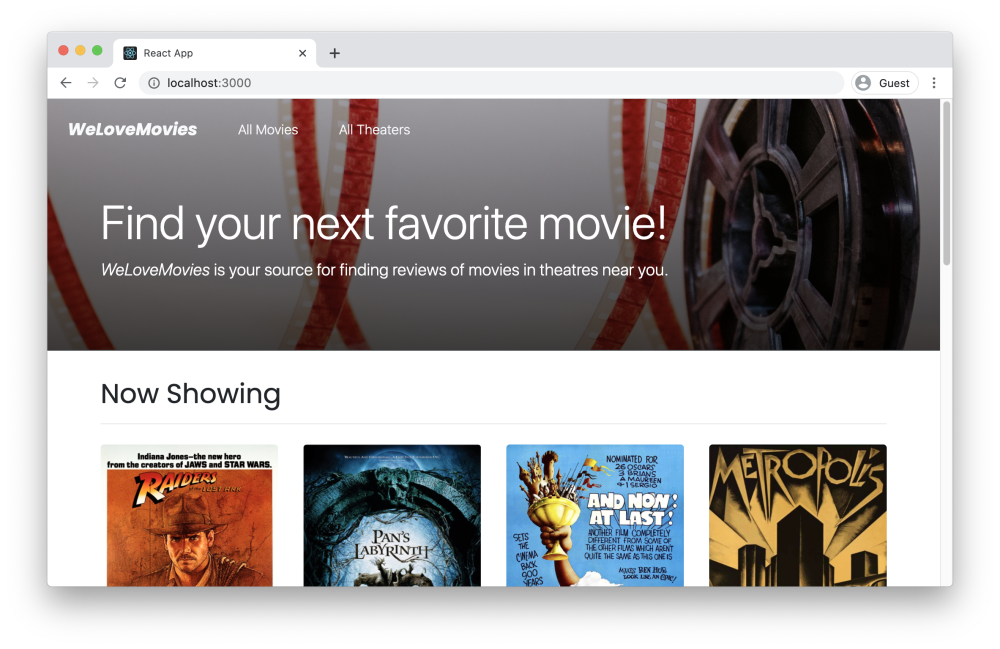
# Project: WeLoveMovies

You've been hired on as a back end developer for a new startup called WeLoveMovies! As another developer works on the design and front end experience, you have been tasked with setting up a database and building out specific routes so that users can gain access to data about movies, theaters, and reviews.

1:1



This project is designed to test your ability to both build complex servers and access data through a database. To succeed at this project, you'll need to demonstrate you can do the following:

* Install and use common middleware packages.
* Receive requests through routes.
* Running tests from the command line.
* Access relevant information through route and query parameters.
* Create an error handler for the case where a route does not exist.
* Build an API following RESTful design principles.
* Create and customize a knexfile.js file.
* Create a connection to your database with knex.
* Write database queries to complete CRUD routes in an Express server.
* Return joined and nested data with Knex.
* Write database migrations using Knex's migration tools.
* Deploy your back end server to a cloud service.

You will not need to make any edits to HTML or CSS for this project.

## Project setup

Follow the instructions below to get this project up and running on your own machine:

### Note: This project will not deploy properly without additional configuration. You do NOT need to deploy it to finish this assignment.

* Sync this Qualified assignment with your computer.
* If you choose to work on this project disconnected from Qualified, run npm install.

Some helper scripts have been added to the package.json file. You are welcome to modify these as you wish.

Visit the following repository if you would like to see your project connected to a Front End Application:

* [Starter Code: WeLoveMovies](https://github.com/Thinkful-Ed/starter-movie-front-end)

Instructions are included there on how to get the project up and running.

## Running tests

To run the tests, you can run the following command:

npm test

Note that the tests make use of an in-memory SQLite database.

**NOTE** When updating a record in an in-memory SQLite database, the server does not automatically respond with an array of updated records like PostgreSQL does. As a result, when updating a record you will need to query the database again to return updated record.

## Instructions

You are tasked with both setting up the database and building a number of routes that will be used by the front end application. For this project, you will start by making the necessary changes to the data tier and then proceed to make changes to the application tier following an [inside-out](https://courses.thinkful.com/connecting-it-all-v1/checkpoint/4) development workflow. Each table is detailed below as is each route.

### General tasks

You will also need to make sure the following tasks are complete.

* Your app.js file and server.js file are correctly configured, with your app.js file exporting the application created from Express.
* You make use of the cors package so that requests from the front end can correctly reach the back end.
* If a request is made to a route that does not exist, the server returns a 404 error.
* If a request is made to a route that exists, but the HTTP method is wrong, a 405 error is returned.
* All of your routes should respond with the appropriate status code and should use a data key in the response.

### Database Tables

You will create five tables for this project. View the docs/tables/ folder in this project to get more detailed information on each table.

You will need to create migrations for each of these tables and run those migrations.

Seed data is included in this project in the [./src/db/seeds](https://www.qualified.io/assess/src/db/seeds) folder. The seeds will run correctly if and only if the tables are setup as described in the previous documents.

### Routes

You will create five routes for this project. View the docs/routes/ folder in this project to get more detailed information on each table.

Note that some routes return data dependent on query parameters.

**Note:** In addition to needing to pass the tests and requirements in the instructions here, please review the Rubric Requirements for the human-graded part of this project in your Thinkful curriculum page.