Introduction

In this test you are going to code along to a simple movies REST API. Afterwards you are going to refactor it and add some features to make it a beautiful and robust API for end users. Finally you will be making a video to demonstrate the API, explain the code and answer a couple of questions. This mimics something developers do all the time where they are explaining their code to the rest of the team.

Instructions

- Make the code along: https://www.youtube.com/watch?v=coyUaGdGqp8
- Fix the things mentioned below in the project
- Read the video requirements.
- Answer the questions on paper and make a script for your video. The script should contain the things mentioned in the Goal / Requirements section!
- Record your video (you will probably have to try it a few times before doing it right).
- Submit your video, by sending the link to @Rob van Kruijsdijk on slack

Things to fix

The code-along does not follow a couple of the programming guidelines and seems to trust the user completely, which a backend developer should never do! At HackYourFuture, however, we only hand in code that is pristine so let's fix that!

Fix the following problems:

- Split up the code into logical files, controllers with controllers and routes with routes
- Make sure that the status codes are always correct, let's be explicit here and always set them even if the default is 200. Rather be extra careful with this.
- Our POST request just lets the user add anything to the database. Add the following checks (and make sure to inform the user with error messages what is going wrong):
    - Make sure that ONLY the fields title, director and release_date are given OR only put those 3 fields into the database
    - Have our API decide what the id will be so that it is unique in our list
    - If everything is successful, let our user know what id was given to the movie so they can access it easily again
    - If not successful then let the user know what is wrong. Remember that the user cannot see your code so the error message needs to have all the information the user of your API will need!
- If you try to delete a movie with an id that is not in the list, you still get the message that the movie was deleted. It would be much nicer if the person using our API gets the message that no movie with that id exists. Something like ‘The movie with id 123 not found in the database.’
- Add tests for all of your endpoints. You will need to add the jest and the supertest library and set it up. Remember to test both the happy path as well as the error paths you have created!

Please fix these things so that the API becomes a solid API that is easier to use and is ready for a production environment. Remember to keep the status codes in mind!

Video requirements

- Duration: 12 minutes maximum. (After 12 minutes, we'll stop watching the video)
- High quality (at least 720p)
- We need to see you and your screen.
- You need to include all of the following things in your video (this is what you are graded on!):
    - Is this a REST Application? If yes, can you name the REST resource?
    - Explain what operations are permitted on those resource(s) and what HTTP Method they use
    - Demo the application, explain all the routes and what they do.
    - Explain the changes you made to the code to make it more robust. Be sure to show the code and explain the different code paths
    - Explain the tests you made (i.e. what you are testing)
- After watching your video, the project must be clear to anyone who understands node.js. So remember to keep your code explanations high level, the viewer can read code!
- Keep in mind that we will only look at your video. Make sure that the code you are talking about is clearly visible!

Grading

You will be graded on three main points:

- The fixes you made (up to 12 points)
- The tests you wrote (up to 6 points)
    - Did you test both the happy and the error paths?
- The style of the code, make the code 'production ready' (up to 4 points)
    - no more 'console.log' test lines
    - no commented out code
    - comments for functions/lines of code that can be unclear for other programmers
    - check your variable/function naming
    - check that the code is split logically (repeated code should be in a function, no huge functions)
- The video (up to 8 points)
    - Make sure you cover everything mentioned in the video requirements, you get points for each part
    - NOTE: We will only listen to the first 12 minutes of the video, anything after that does not count. So make sure you do it within the time limit.