Introduction

In this test you are going to code along to a simple users REST API. Afterwards you are going to fix some errors/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.

Goal / Requirements

Read the instructions carefully, in the end you should have a video explaining the following things:

- Is this a REST Application? If yes, can you name the REST resource?
- Explain what operations are permitted on that resource
- Demo the application, explain all the routes and what they do.
- Explain the changes you made to the code to fix the issues. Be sure to show the code and explain the different code paths

Instructions

- Make the code along: https://www.youtube.com/watch?v=I8WPWK9mS5M
- Fix the issues 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 (probably you've to try it a few times before doing it right).
- Submit your video, by sending the link to @Rob van Kruijsdijk on slack

Issues to fix

This code-along only follows the happy path as developers like to call it. This means that it does not handle user errors well. For example:

- If you send a PATCH request to your server for a user id that does not exist in the list, there will be a system error thrown from the server. This should send a readable error message back for the user of our API! Something like 'The user with id 123 not found in the database.'
- If you send a PUT request with a JSON object that does not have the firstName property set, there will be a system error thrown from the server. This should send a readable error message back for the user of our API! Something like 'The user needs a firstName property set.'

There are also some usability issues with the API. For example:

- If you try to delete a user with an id that is not in the list, you still get the message that the user was deleted. It would be much nicer if the person using our API gets the

message that no user with that id exists. Something like ‘The user with id 123 not found in the database.’
- If you try to get a user with an id that is not in the list, the API just returns an empty body. It would be nicer if it would send a message that no user with that id exists. Something like ‘The user with id 123 not found in the database.’

Please fix these things so that the API becomes a solid API that is easier to use and does not crash. Remember to keep the status codes in mind!

Video requirements

- Duration: 10 minutes **maximum**. (After 10 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!):
    - Explain the kind of resource(s) the application has
    - Explain what operations are permitted on those resource(s) and what HTTP Method they use
    - Explain the application code, explain all the routes and how they work in the code. Explain what code does what part
    - Explain the changes you made to the code to fix the issues in section 3
- After watching your video, the project must be clear to anyone who understands node.js.
- Keep in mind that we will only look at your video. Make sure that the code is clearly visible and you show you understand the topics.
- Check Malcolm’s example test here to have an idea what your end video should look like.

Grading

You will be graded on three main points:

- The fixes you made (up to 6 points)
- 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 10 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 10 minutes of the video, anything after that does not count. So make sure you do it within the time limit.