**Comments**

-Watch your semicolons. It’s something your linter won’t catch, so be consistent with what lines end with a ;

-Really glad to see you using *this* correctly between instance and class methods!

-We can meet and go over routing and error handling more in depth sometime.

-Some of your code could be simplified using Sequelize methods, such as .findById() and .update()

-think about what’s on your req.body, as you can often just pass it in instead of moving each prop off of it to a new object

**Language / Library Fluency (10)**

(0) Needs Work (1) Satisfactory (2) Good (3) Excellent (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Free of syntax errors and runtime errors generated by tests |  |  |  |  |  |
| Properly manages control flow (conditionals, iteration, etc.) |  |  |  |  |  |
| Uses full range of API as intended rather than hacking together solutions with only a few parts |  |  |  | Use .findById()  And .update() |  |
| Promise chains properly compose asynchronous actions |  |  |  | One promise nested instead of chained. Move .catch to end of .then()s |  |
| Incorporates some ES6 (let, const, arrows, destructuring, etc.) -- GRADED ON 2-4 scale here |  |  | Great time to start using let, const, and arrow functions! |  |  |

Total: 16 / 20 = **8 / 10**

**Economy and Performance (5)**

(0) Needs Work (1) Satisfactory (2) Good (3) Excellent (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Avoids repetition in favor of reusable variables and functions (DRY) |  |  |  |  |  |
| Chooses the appropriate data structures and methods from JS for the job |  |  |  |  |  |
| Doesn’t perform unnecessary operations or declare unused variables or functions |  |  |  |  |  |

Total: 12 / 12 = **5 / 5**

**Readability and Maintainability (5)**

(0) Needs Work (1) Satisfactory (2) Good (3) Excellent (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variable and function names are meaningful and self-documenting |  |  |  | Good names overall.  “Instance” is probably too general a name |  |
| Formatting is consistent (including whitespace, indentation, semicolons, bracket placement, etc.) |  |  | Watch your semicolons! |  |  |
| Comments help clarify complex or unconventional code (2-4 scale, clean code doesn’t necessarily have to be commented) |  |  |  |  |  |

Total: 9 / 12 = **4 / 5**

**Cleanliness (5)**

(0) Needs Work (1) Satisfactory (2) Good (3) Excellent (4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Handles errors properly and consistently |  | Watch your .catchs, Missed the custom error |  |  |  |
| Doesn’t contain commented-out code (except to show incomplete thoughts) or console.log statements |  |  |  |  |  |
| Conforms to best practices and avoids use of deprecated features or anything known to introduce bugs |  |  |  |  |  |

Total: 8 / 12 = **3 / 5**

**Overall**

(75% of score) Specs Passing: 22 / 25 = 66 / 75

(25% of score) Rubric Total: 20 / 25

(up to 10% added) Extra Credit Passing: = 0 / 2 = 0 / 10

**Final Grade:** 86 / 100