**# BE Northcoders NC Games Portfolio Check List**

​

**## Readme - Remove the one that was provided and write your own:**

​

- [ ✅] Link to hosted version

- [ ✅] Write a summary of what the project is

- [ ✅] Provide clear instructions of how to clone, install dependencies, seed local database, and run tests

- [ ✅] Include information about how to create `.env.test` and `.env.development` files

- [ ] Specify minimum versions of `Node.js` and `Postgres` needed to run the project

​

**## General**

​

- [ ✅] Remove any unnecessary `console.logs` and comments

- [ ] Remove all unnecessary files (e.g. old `README.md`, `error-handling.md`, `hosting.md`, `./db/utils/README.md` etc.)

- [ ✅] Functions and variables have descriptive names

​

**## Creating tables**

​

- ✅] [ ] Use `NOT NULL` on required fields

- [ ✅] Default `created\_at` in reviews and comments tables to the current date:`TIMESTAMP DEFAULT NOW()`

​

- *\_you have used CURRENT\_TIMESTAMP and that is fine :) \_*

​

**## Inserting data**

​

- [ ✅] Drop tables and create tables in seed function in correct order

- [ ✅] Make use of pg-format to insert data in the correct order

​

**## Tests**

​

- [ ✅] Seeding before each test

- [ ✅] Descriptive `it`/`test` block descriptions

- ✅] [ ] If asserting inside a `forEach`, also has an assertion to check length is at least > 0

- [ ✅] Evidence of building up complex query endpoints using TDD

- [ ✅] Ensure all tests are passing

- [ ✅] Cover all endpoints and errors

​

- `GET /api/categories`

​

- [ ✅] Status 200, array of category objects

​

- `GET /api/reviews/:review\_id`

​

- [ ✅] Status 200, single review object (including `comment\_count`)

- [ ✅] Status 400, invalid ID, e.g. string of "not-an-id"

- [ ✅] Status 404, non existent ID, e.g. 0 or 9999

​

- `PATCH /api/reviews/:review\_id`

​

- [ ✅] Status 200, updated single review object

- [ ✅] Status 400, invalid ID, e.g. string of "not-an-id"

- [ ✅] Status 400, invalid inc\_votes type, e.g. property is not a number

- [ ✅] Status 404, non existent ID, e.g. 0 or 9999

- [ ] Status 200, missing `inc\_votes` key. No effect to article.

​

- `GET /api/reviews`

​

- [ ✅] Status 200, array of review objects (including `comment\_count`, excluding `body`)

- [ ✅] Status 200, default sort & order: `created\_at`, `desc`

- [ ✅] Status 200, accepts `sort\_by` query, e.g. `?sort\_by=votes`

- [ ✅] Status 200, accepts `order` query, e.g. `?order=desc`

- [ ✅] Status 200, accepts `category` query, e.g. `?category=dexterity`

- [ ✅] Status 400. invalid `sort\_by` query, e.g. `?sort\_by=bananas`

- [ ✅] Status 400. invalid `order` query, e.g. `?order=bananas`

- [ ] Status 404. non-existent `category` query, e.g. `?category=bananas`

- [ ] Status 200. valid `category` query, but has no reviews responds with an empty array of reviews, e.g. `?category=children's games`

​

- `GET /api/reviews/:review\_id/comments`

​

- [ ✅] Status 200, array of comment objects for the specified review

- [ ✅] Status 400, invalid ID, e.g. string of "not-an-id"

- [ ] Status 404, non existent ID, e.g. 0 or 9999

- [ ] Status 200, valid ID, but has no comments responds with an empty array of comments

​

- `POST /api/reviews/:review\_id/comments`

​

- [ ✅]] Status 201, created comment object

- [ ✅]] Status 400, invalid ID, e.g. string of "not-an-id"

- [ ✅]] Status 404, non existent ID, e.g. 0 or 9999

- [ ] Status 400, missing required field(s), e.g. no username or body properties

- [ ✅]] Status 404, username does not exist

- [ ✅] Status 201, ignores unnecessary properties

​

- `DELETE /api/comments/:comment\_id`

​

- [ ✅] Status 204, deletes comment from database

- [ ✅] Status 404, non existent ID, e.g 999

- [ ✅] Status 400, invalid ID, e.g "not-an-id"

​

- `GET /api`

​

- [ ✅] Status 200, JSON describing all the available endpoints

​

**## Routing**

​

- [ ✅] Split into api, categories, users, comments and reviews routers

- [ ] Use `.route` for endpoints that share the same path

​

**## Controllers**

​

- [ ✅] Name functions and variables well

- [ ✅] Add catch blocks to all model invocations (and don't mix use of`.catch(next);` and `.catch(err => next(err))`)

​

**## Models**

​

- Protected from SQL injection

- [ ✅] Using parameterized queries for values in `db.query` e.g `$1` and array of variables

- [ ✅] Sanitizing any data for tables/columns, e.g. greenlisting when using template literals or pg-format's `%s`

- [ ✅] Consistently use either single object argument *\_****\*\*or\*\*****\_* multiple arguments in model functions

- [ ✅] Use `LEFT JOIN` for comment counts

​

**## Errors**

​

- [ ✅] Use error handling middleware functions in app and extracted to separate directory/file

- [ ✅] Consistently use `Promise.reject` in either models *\_****\*\*OR\*\*****\_* controllers

​

**## Extra Tasks - To be completed after hosting**

​

- `GET /api/users`

​

- [ ✅] Status 200, responds with array of user objects

​

- `GET /api/users/:username`

​

- [ ✅] Status 200, responds with single user object

- [ ✅] Status 404, non existent ID, e.g 999

- [ ] Status 400, invalid ID, e.g "not-an-id"

​

- `PATCH /api/comments/:comment\_id`

​

- [✅ ] Status 200, updated single comment object

- [ ✅] Status 400, invalid ID, e.g. string of "not-an-id"

- [ ✅] Status 400, invalid inc\_votes type, e.g. property is not a number

- [ ✅] Status 404, non existent ID, e.g. 0 or 9999

- [ ] Status 200, missing `inc\_votes` key. No effect to comment.

​

**## Extra Advanced Tasks**

​

**### Easier**

​

- [ ] Patch: Edit an review body

- [ ] Patch: Edit a comment body

- [ ] Patch: Edit a user's information

- [ ] Get: Search for an review by title

- [ ] Post: add a new user

​

**### Harder**

​

- [ ] Protect your endpoints with JWT authorization. We have notes on this that will help a bit, *\_but it will make building the front end of your site a little bit more difficult\_*

- [ ] Get: Add functionality to get reviews created in last 10 minutes

- [ ] Get: Get all reviews that have been liked by a user. This will require an additional junction table.

- [ ] Research and implement online image storage or random generation of images for categories

​

##SUMMARY

​

Some wonderful code and great looking work here Kirsty! I love your use of Async/Await and there is some really nice logic in your models for interacting with the database. This is a really strong effort. Don't be put off by the amount of feedback below, it's just me being nitpicky as usual but it needs to be as this is an important project.

​

Keep up the amazing work!

​

**## FEEDBACK**

​

✅- *\_NOT NULL CONSTRAINT - There are still a couple of columns in some of your tables that do not have a NOT NULL constraint - We could have a review with no designer for example! \_*

​

✅- *\_You have used a forEach to check what each review should look like which is great! However, we have not checked the length for the amount of reviews we are getting back! You have done this check for your categories and your comments which is great. It is important that we make this check as we could get the wrong number of reviews back, but still pass the test as they are structured in the correct way. \_*

​

- `PATCH /api/reviews/:review\_id` - *\_Your code for this looks great but I want to bring your attention as to what happens if I send in an empty object on a PATCH request to increment the votes...it changes the value of your votes in the Reviews table to NULL. We might want to put some kind of check in to avoid this issue \_*

​

- `GET /api/reviews` - *\_A category that does not exist gives back an empty array. We would probably want to do this as a 404 - Not Found. It would make more sense to have an empty array for reviews of categories that actually currently exist in our database\_*

​

- `GET /api/reviews/:review\_id/comments` - *\_Similar to above - if I try and search comments for review with an ID of 999 - I get back an empty array. This behavious makes more sense for a valid review\_Id with no comments. 404 error make more sense if the review does not exist\_*

​

- `POST /api/reviews/:review\_id/comments` - *\_If I send in a body of just a valid username property - your server throws a 500 error. This is due to line 91 in your reviews controller trying to check the lenght of the body variable. The issue is that body is undefined as nothing has been sent in on the request body. A 500 error is going to crash our server so this is something that will need to be remedied.\_*

​

- `PATCH /api/comments/:comment\_id` - *\_If I add an extra key value pair to my request body - then your server sends back a 400 bad request. If there is a correct inc\_votes in this object, could we possibly just ignore anything else? - If is this is the behaviour you would like then that is fine, but I just wanted to put it out there in the ether.\_*