2021

CAB230 REST API – Server Side

CAB230 Happiness API – The Server Side Application

Eunyoung(Jasmine) Hur

N10622012

5/3/2021

Contents

This template is adapted from one created for a more elaborate application. The original author spends most of his professional life talking to clients and producing architecture and services reports. You may find this a bit more elaborate than you are used to, but it is there to help you get a better mark

This report will probably be around 5 pages or so including screenshots

Introduction

Purpose & description

Setting up server-side enables communication with the database and error handling. The server side is built as an API, which uses HTTP requests to access and use data. The API consists of PUT, DELETE, POST, and GET, mainly using POST and GET in this assignment. In this task, six endpoints were used, which are located at one end of the communication channel where the API can access the resources needed to interact with other systems. The endpoints are covered in more detail below.

Completeness and Limitations

In this task, almost all endpoints pass the test and are easy to use. However, there is a problem that the API in the profile part does not work smoothly.

/rankings

Fully functional

This endpoint contains id, rank, country, score, economy, family, health, freedom, generosity, trust, and year information. This rankings endpoint can be found with only the desired information by searching years and countries. Here, only GET was used because the information was simply requested.

```
app.use('/rankings', rankingsRouter);
```

```
router.get("/", function(req, res, next)
```

/countries

Fully functional

```
router.get("/", function(req, res, next) {
  req.db.from('rankings').distinct('country').orderBy('country').pluck('country')
```

This information is listed by importing only the country from the Rankings table. The data is made because users want to search for a country. Similarly to rankings, GET was used simply because it asked for information.

/factors/{year}

Fully functional

It is an endpoint that can be used when the user is logged in and authenticated. Note that year is not a query because it is the default value. The GET was also used because Factor does not modify information.

```
router.get('/:year', function(req, res, next) {
```

/user/register

Fully functional

```
router.post('/user/register', function(req, res, next) {
```

It is the endpoint for registration information. It communicates to POST because it needs to be modified and be added information. raw data should not be included when information is registered, for that reason, the information is inserted after a hash operation.

/user/login

Fully functional

```
router.post('/user/login', function(req, res, next) {
```

This is the endpoint for login. Likewise, POST was used the same as a register. If login is successful, return the expiration date, token.

/user/{email}/profile

```
router.get('/user/:id/profile', function(req, res, next) {
```

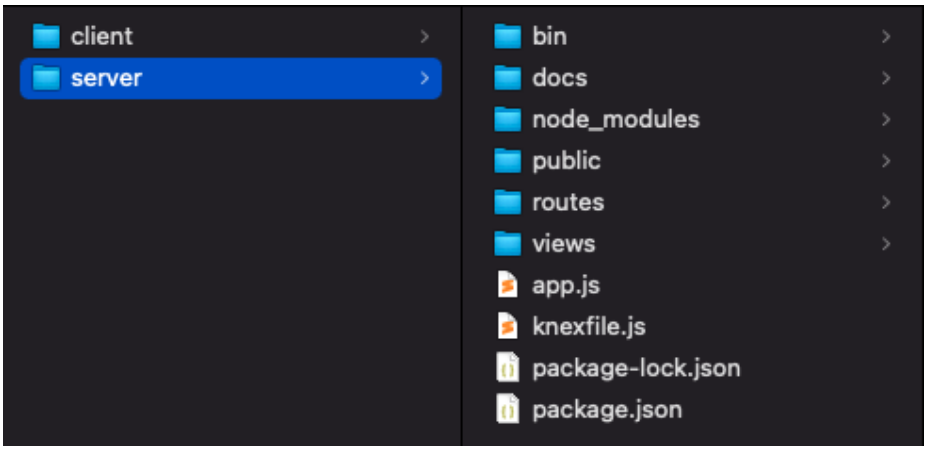
An endpoint that can make an inquiry into a user's profile. Like Factor, it is a service that requires login. Notice that id does is not entered as the query string.

Modules used

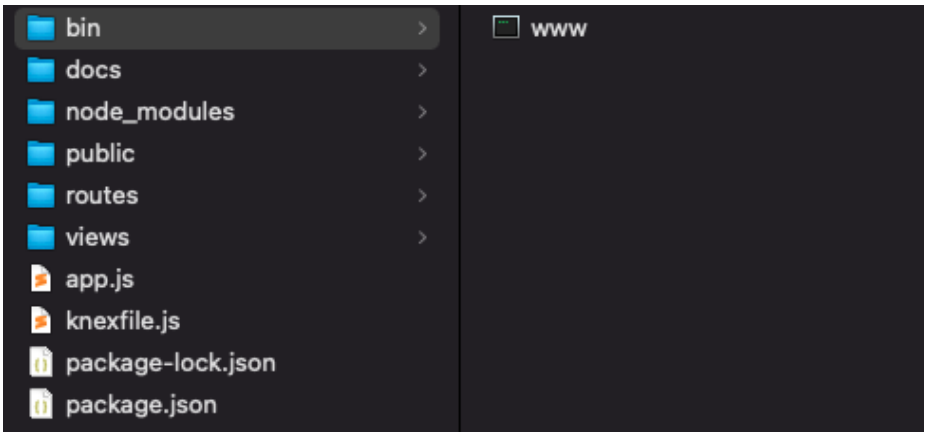
No additional modules used

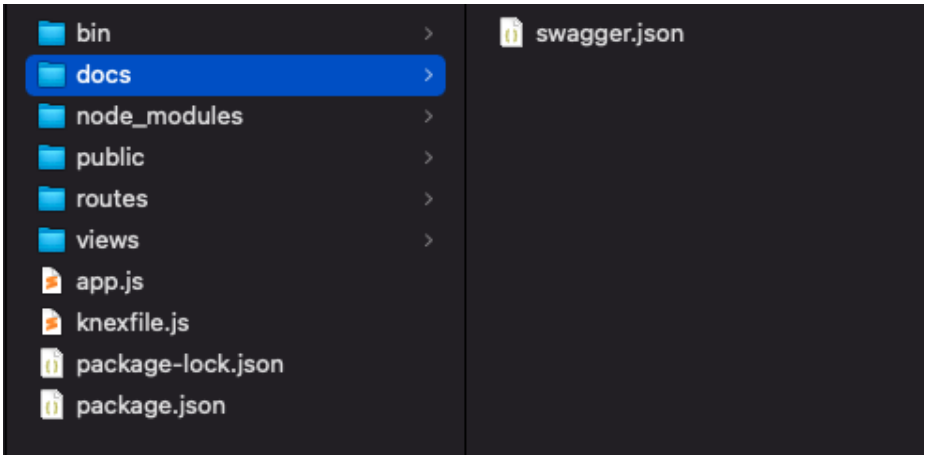
Technical Description

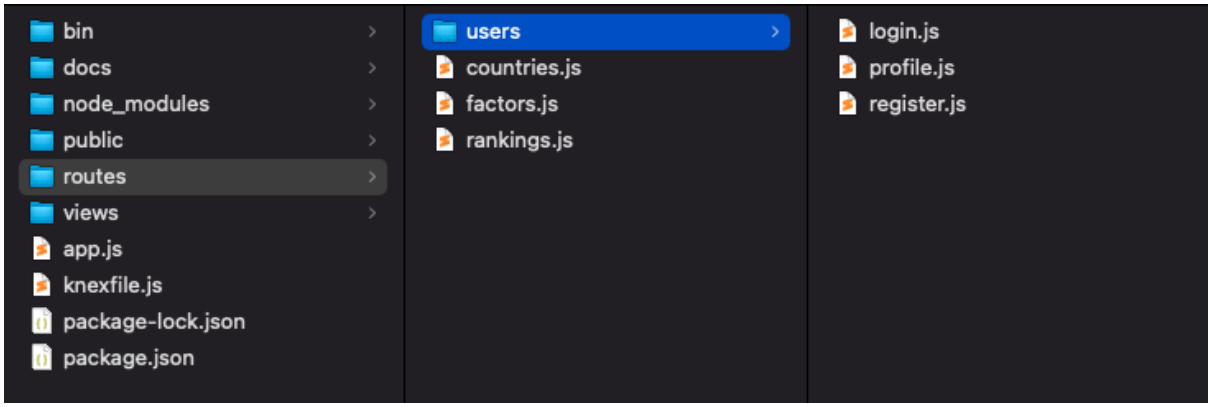
Architecture



When you download the Express module, the folders are created like the upper right side on the screenshot. If the server-side is configured, the client-side can be cleaned up by creating 2 folders as shown on the left side.







The folder in Routes contains all the endpoints. User-related information files like login.js, register.js, profile.js are managed by putting them together in a user folder.

These files are managed by app.js. App.js is central to managing files in the kenexfile.js and routes folders at once and linking them together.

Security

- Use of `knex` or other query builder without raw SQL
- Use of `helmet` with at least the default settings enabled
- Use of `morgan` with a logging level similar to that used in the pracs.
- Appropriate handling of user passwords as described in the JWT Server-Side worksheet.
- Deployment using TLS/HTTPS

- The nature of the vulnerability

Usually, it can be described as the weakest security part of a website, it is definitely the login and membership part. Because if the application uses raw data directly when users log in and join a member, it can be exposed the member's information.

It is also exposed as it is when sending information to the header, which can also be a major security weakness.

- Whether or not your Application deals with it

Therefore, the application was supplemented with JWT. By encrypting passwords, passwords can be stored more securely in a database.

About header security, a Helmet can be used to protect headers.

Testing

Test Report

Started: 2021-06-18 05:40:13

Suites (1)	Tests (301)
0 passed	206 passed
1 failed	95 failed
0 pending	0 pending

/Users/jasmine/Desktop/QUT/2021-2/CAB230/Assignment3/happinessapi-tests-master/integration.test.js 2.858s

profile > retrieval with default profile values > with unauthenticated user default profile values	*should return status code 200*	failed	0.001s
profile > retrieval with default profile values > with unauthenticated user default profile values	*should return status text - OK*	failed	0.001s
profile > retrieval with default profile values > with unauthenticated user default profile values	*should return user email property*	failed	0s
profile > retrieval with default profile values > with unauthenticated user default profile values	*should return null for unset firstName*	failed	0s
profile > retrieval with default profile values > with unauthenticated user default profile values	*should return null for unset lastName*	failed	0s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return status code 200*	failed	0.001s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return status text - OK*	failed	0s

profile > retrieval with default profile values > with authenticated matching user default profile values	*should return user email property*	failed	0.001s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return null for unset firstName*	failed	0s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return null for unset lastName*	failed	0s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return null for unset dob*	failed	0s
profile > retrieval with default profile values > with authenticated matching user default profile values	*should return null for unset address*	failed	0s
profile > retrieval with default profile values > with authenticated non-matching user default profile values	*should return status code 200*	failed	0.001s
profile > retrieval with default profile values > with authenticated non-matching user default profile values	*should return status text - OK*	failed	0s

profile > retrieval with default profile values > with authenticated non-matching user default profile values	*should return user email property*	failed	0s
profile > retrieval with default profile values > with authenticated non-matching user default profile values	*should return null for unset firstName*	failed	0s
profile > retrieval with default profile values > with authenticated non-matching user default profile values	*should return null for unset lastName*	failed	0s

profile > update of user profile > with unauthenticated user	*should return status code 401*	failed	0s
profile > update of user profile > with unauthenticated user	*should return status text - Unauthorized*	failed	0s
profile > update of user profile > with unauthenticated user	*should return error with boolean of true*	failed	0.001s
profile > update of user profile > with unauthenticated user	*should contain message property*	failed	0s
profile > update of user profile > with authenticated non-matching user	*should return status code 403*	failed	0s
profile > update of user profile > with authenticated non-matching user	*should return status text - Forbidden*	failed	0s
profile > update of user profile > with authenticated non-matching user	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated non-matching user	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with missing body keys	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with missing body keys	*should return status text - Bad Request*	failed	0s
profile > update of user profile > with authenticated matching user > with missing body keys	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated matching user > with missing body keys	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with missing body keys	*should return specific message for 'Request body incomplete: firstName, lastName, dob and address are required.'*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid firstName	*should return status code 400*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid firstName	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid firstName	*should return error with boolean of true*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid firstName	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid firstName	*should return a specific message for 'Request body invalid, firstName, lastName and address must be strings only.'*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid lastName	*should return status code 400*	failed	0s

profile > update of user profile > with authenticated matching user > with invalid lastName	*should return status text - Bad Request*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid lastName	*should return error with boolean of true*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid lastName	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid lastName	*should return a specific message for 'Request body invalid, firstName, lastName and address must be strings only.'*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid address	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid address	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid address	*should return error with boolean of true*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid address	*should contain message property*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid address	*should return a specific message for 'Request body invalid, firstName, lastName and address must be strings only.'*	failed	0.001s

profile > update of user profile > with authenticated matching user > with invalid date format	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid date format	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with invalid date format	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid date format	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with invalid date format	*should return a specific message for 'Invalid input: dob must be a real date in format YYYY-MM-DD.'*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (out of bounds check)	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (out of bounds check)	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (out of bounds check)	*should return error with boolean of true*	failed	0s

profile > update of user profile > with authenticated matching user > with valid formatted non-real date (out of bounds check)	*should contain message property*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (out of bounds check)	*should return a specific message for 'Invalid input: dob must be a real date in format YYYY-MM-DD.'*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (Javascript date rollover check)	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (Javascript date rollover check)	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (Javascript date rollover check)	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (Javascript date rollover check)	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (Javascript date rollover check)	*should return a specific message for 'Invalid input: dob must be a real date in format YYYY-MM-DD.'*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (non leap-year check)	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (non leap-year check)	*should return status text - Bad Request*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (non leap-year check)	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated matching user > with valid formatted non-real date (non leap-year check)	*should contain message property*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the future	*should return status code 400*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the future	*should return status text - Bad Request*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid date in the future	*should return error with boolean of true*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the future	*should contain message property*	failed	0.001s

profile > update of user profile > with authenticated matching user > with valid date in the future	*should return a specific message for 'Invalid input, dob must be a date in the past.'*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return status code 200*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return status text - OK*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should be an object result*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return user email property*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return updated firstName*	failed	0.001s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return updated lastName*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return updated dob*	failed	0s
profile > update of user profile > with authenticated matching user > with valid date in the past	*should return updated address*	failed	0.001s
profile > retrieval after update of user profile > with unauthenticated user updated profile values	*should return status code 200*	failed	0s
profile > retrieval after update of user profile > with unauthenticated user updated profile values	*should return status text - OK*	failed	0s
profile > retrieval after update of user profile > with unauthenticated user updated profile values	*should return user email property*	failed	0s
profile > retrieval after update of user profile > with unauthenticated user updated profile values	*should return updated firstName*	failed	0s
profile > retrieval after update of user profile > with unauthenticated user updated profile values	*should return updated lastName*	failed	0s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return status code 200*	failed	0.001s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return status text - OK*	failed	0s

profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return user email property*	failed	0s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return updated firstName*	failed	0s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return updated lastName*	failed	0s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return updated dob*	failed	0.001s
profile > retrieval after update of user profile > with authenticated matching user updated profile values	*should return updated address*	failed	0s
profile > retrieval after update of user profile > with authenticated non-matching user updated profile values	*should return status code 200*	failed	0s
profile > retrieval after update of user profile > with authenticated non-matching user updated profile values	*should return status text - OK*	failed	0s
profile > retrieval after update of user profile > with authenticated non-matching user updated profile values	*should return user email property*	failed	0s
profile > retrieval after update of user profile > with authenticated non-matching user updated profile values	*should return updated firstName*	failed	0s
profile > retrieval after update of user profile > with authenticated non-matching user updated profile values	*should return updated lastName*	failed	0s
Miscellaneous > with swagger docs route	*should return Swagger UI*	failed	0s

Difficulties / Exclusions / unresolved & persistent errors /

The most difficult part was connecting to the VM and allowing the application to run.

The restriction of access to ssl prevented it from running, so all permissions were enabled using chmod, but it was not possible to run.