Documentation for custom tasks - Senior Node js Developer , Legal Doctrine.

- I changed the url provided in the custom task , from PUT /text/:textId to PUT /**texts**/:textId to follow best practices of Rest api design , “texts” is a resources, so when we request a specific resource, we ask resources to give it for us , an example :
  Get /resources/:id, Get /resources/:id/votes, when we want to store a resource we push to the resources ex: Post /resources .

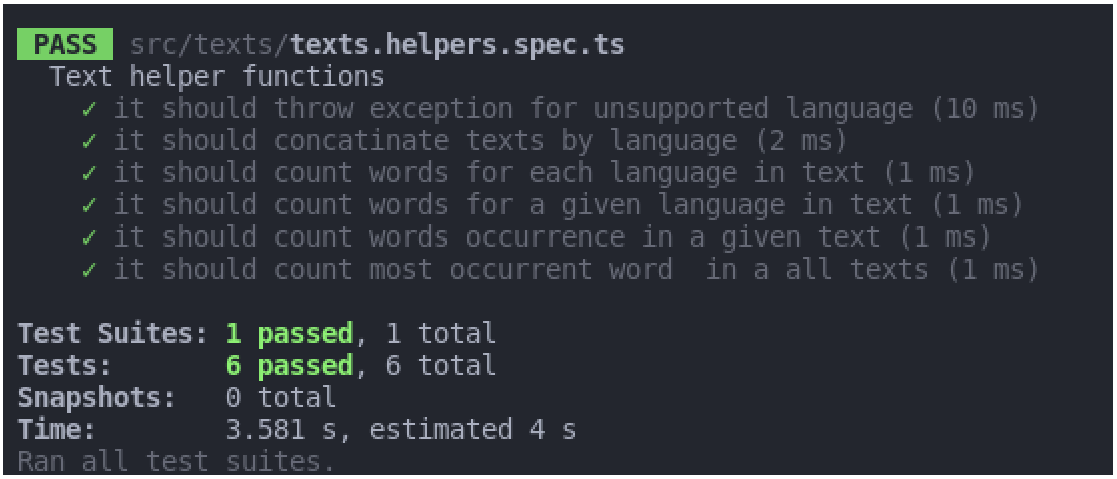
- Because the test says, use any Node js framework , i choose Nest Js which good framework for enterprise applications, due to its architecture, Nestjs uses Express.js by default, you don’t need to manage typescript versions and other packages yourself, Nest js takes care of all the thing, you only need to focus on business logic and deliver a value.

- the test says “In order to be sure that you have a strong understanding of JavaScript and Node.js”, for mostOccurrent route , i used javascript to achieve the desired behavior, but we should be aware that this is an expensive operation to do in the server side (Node js is not very efficient in heavy computations, due to its design, and depends on the implementation of the feature) , a solution maybe delegate the task to the database, most databases are optimized for those things, but we should decide only when we have some benchmarks or noticing some blocking (Node js).

- I used Mongodb Atlas free tier, so the file **.development.env** is not included in the github repository.

- I added unit tests for each helper method , found in **src/texts.helpers.spec.ts**
  Run the command : **npm run test**

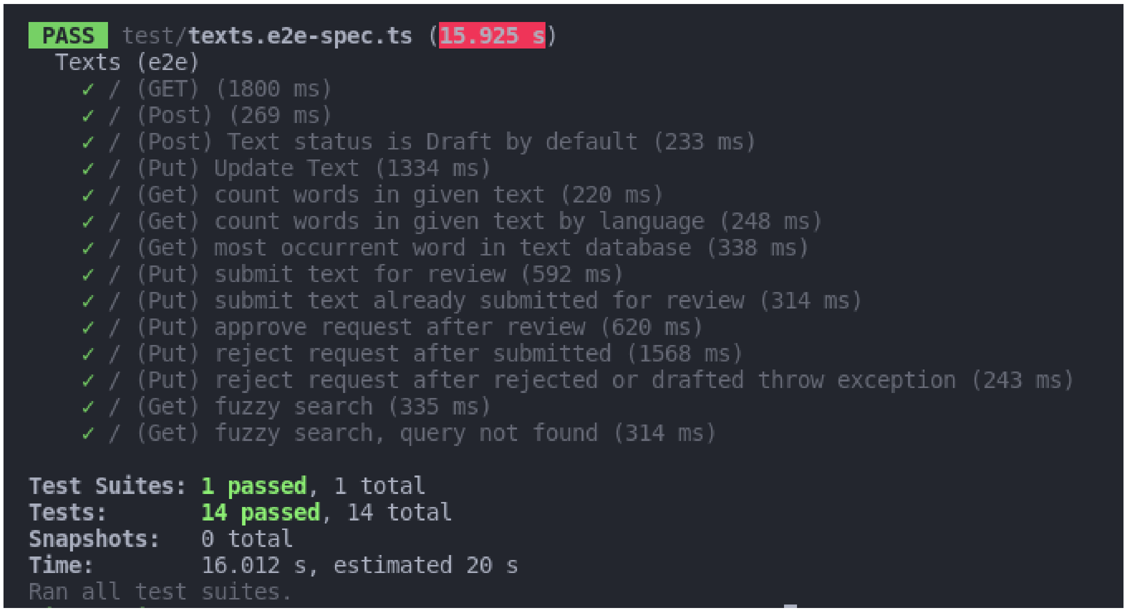
  here screenshot of tests



```
 PASS  src/texts/texts.helpers.spec.ts
  Text helper functions
    ✓ it should throw exception for unsupported language (10 ms)
    ✓ it should concatinate texts by language (2 ms)
    ✓ it should count words for each language in text (1 ms)
    ✓ it should count words for a given language in text (1 ms)
    ✓ it should count words occurrence in a given text (1 ms)
    ✓ it should count most occurrent word  in a all texts (1 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.581 s, estimated 4 s
Ran all test suites.
```

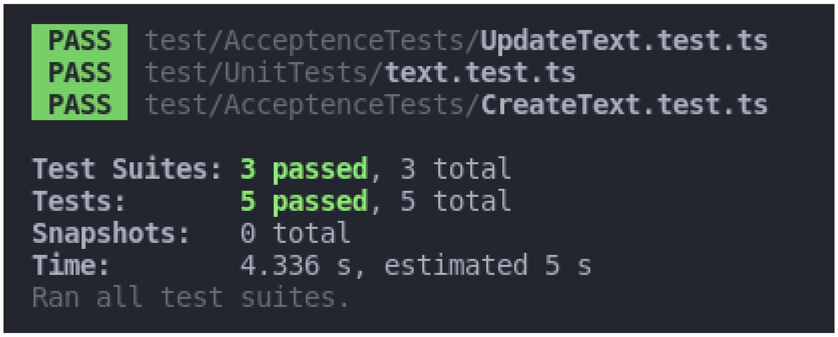
- I added e2e tests , found in **tests/texts.e2e-spec.ts**
  Run the command : **npm run test:e2e**



```
 PASS  test/texts.e2e-spec.ts (15.925 s)
  Texts (e2e)
    ✓ / (GET) (1800 ms)
    ✓ / (Post) (269 ms)
    ✓ / (Post) Text status is Draft by default (233 ms)
    ✓ / (Put) Update Text (1334 ms)
    ✓ / (Get) count words in given text (220 ms)
    ✓ / (Get) count words in given text by language (248 ms)
    ✓ / (Get) most occurrent word in text database (338 ms)
    ✓ / (Put) submit text for review (592 ms)
    ✓ / (Put) submit text already submitted for review (314 ms)
    ✓ / (Put) approve request after review (620 ms)
    ✓ / (Put) reject request after submitted (1568 ms)
    ✓ / (Put) reject request after rejected or drafted throw exception (243 ms)
    ✓ / (Get) fuzzy search (335 ms)
    ✓ / (Get) fuzzy search, query not found (314 ms)

Test Suites: 1 passed, 1 total
Tests:       14 passed, 14 total
Snapshots:   0 total
Time:        16.012 s, estimated 20 s
Ran all test suites.
```

- I have added a decoupled version of the homework with hexagonal architecture in Nodejs, typescript, jest. You can find the project in the repos in the folder named **“LegalDoctrine-Hexagonal-architecture”.** Here the tests :



```
 PASS  test/AcceptenceTests/UpdateText.test.ts
 PASS  test/UnitTests/text.test.ts
 PASS  test/AcceptenceTests/CreateText.test.ts

Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        4.336 s, estimated 5 s
Ran all test suites.
```

- In the end, I enjoyed doing this homework, and the non restriction of the homework like choosing any Nodejs framework, and i believe if you can do it in one framework you can do it in others (Nest Js has some learning curve more than other frameworks).

  I’m waiting for your feedback.

  Project’s repos :
  https://github.com/BenaliDjamel/LegalDoctrine-Senior-Nodejs-Challenge

Djamel Benali.