|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name : | Nguyen Quoc Thang |  | Date : | 30/09/2022 |

|  |
| --- |
| **General** |

1. What are some of the Node.js frameworks/technologies that you have worked with?

|  |  |
| --- | --- |
| Name | Version/Release, Skill Level (B/I/A) or Years |
| Node.js | 16.15.1 - B |
| TypeScript | none |
| jQuery | none |
| Express.js | 4.18.1 - B |
| Sequelize.js | none |
|  |  |
|  |  |
|  |  |
|  |  |

2. What are some of the IDEs (that you have used) and your choice of IDE/code editor when developing Node.js applications?

I currently use VS Code as the editor when developing Node.js apps because it allows me to install many extensions that further improve my productivity

|  |
| --- |
| Skill Level   * **N** = Novice level; does not know much * **B** = Beginner level; know how to write/code/use * **I** = Intermediate level; have few years of experience coding/using and is comfortable * **A** = Advance level; expert in using/coding in (almost) all the aspects and knows the nuances |

3. What are some of the revision source control systems that you have used?

|  |  |
| --- | --- |
| Name | Version/Release, Skill Level (N/B/I/A) or Years |
| Git | I mainly use Git Desktop and have a basic level of Git CLI |
| SVN | none |
| Mercurial | none |
|  |  |

4. What are some of the unit test frameworks/tools that you have used?

|  |  |
| --- | --- |
| Name | Version/Release, Skill Level (N/B/I/A) or Years |
| Jasmine | none |
| Jest | B |
| Mocha | none |
|  |  |
|  |  |

5. What are some of the viewing/presentation technologies/frameworks that you have used?

|  |  |
| --- | --- |
| Name (and Version) | Version/Release, Skill Level (N/B/I/A) or Years |
| ReactJS | >=18, basic level |
| React Native | >=0.68 basic level |
| Bootstrap | basic |
| Pug.js | none |
| Mustache.js | none |
| Ember.js | none |
|  |  |
|  |  |

6. What kind of database systems that you have worked with?

|  |  |
| --- | --- |
| Name (and Version) | Version/Release, Skill Level (N/B/I/A) or Years |
| Postgres | 1 year |
| Oracle RDBMS | none |
| MongoDB | 1 year |
|  |  |

7. What kind of CI/CD systems that you have worked with?

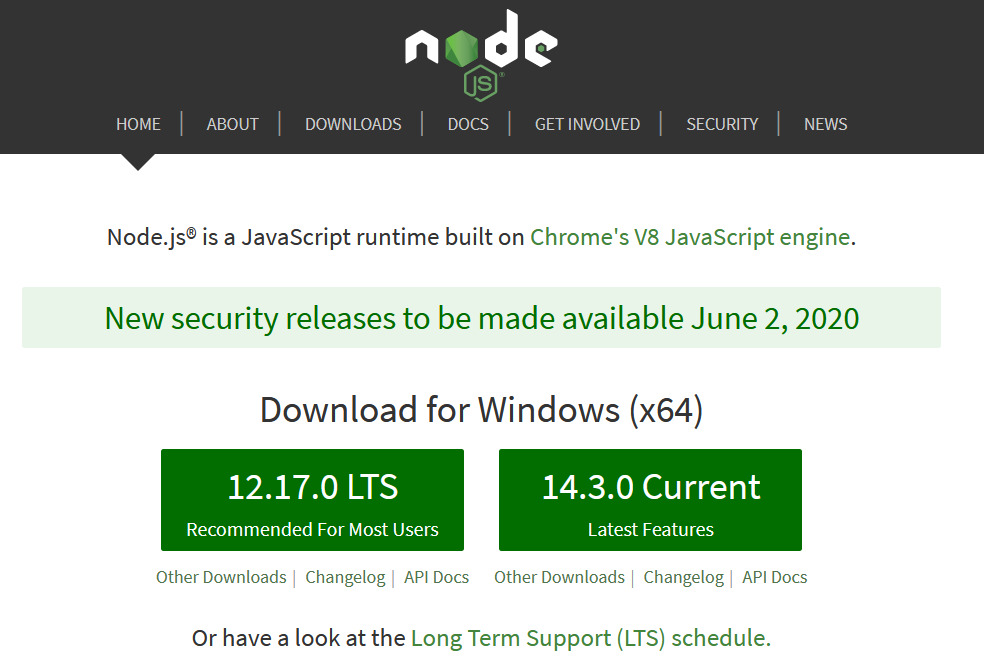
|  |  |
| --- | --- |
| Name (and Version) | Version/Release, Skill Level (N/B/I/A) or Years |
| Jenkins | None |
| SonarQube | None |
| Docker | None |
| Kubernetes | None |
|  |  |
|  |  |

8. What kind of Cloud based systems that you have worked with?

|  |  |
| --- | --- |
| Name (and Version) | Version/Release, Skill Level (N/B/I/A) or Years |
| Amazon Web Services (AWS) | Basic: Lambda, API Gateway, Cognito, S3 |
| Google Cloud Platform (GCP) | Google Map API - Basic |
| Microsoft Azure | None |
|  |  |
|  |  |

|  |
| --- |
| **Preliminary - Initial Setup** |

Install the latest LTS version of Node.js (<https://nodejs.org/en/>) runtime of the correct platform.



As of the point of writing, and for Windows platform, it is "Windows (x64) version 12.17.0 LTS".

Install "npm". It should be bundled with Node.js runtime.

Install "npx". It should be bundled with "npm".

Optionally install "nvm" (<https://github.com/nvm-sh/nvm>) or "nodist" (<https://github.com/nullivex/nodist>).

Attached is a starter Node.js project, using these dependencies:

* Express (<https://expressjs.com/>)
* Sequelize (<https://sequelize.org/>)
* EJS (<https://ejs.co/>)
* Jest (<https://jestjs.io/>)
* Supertest (<https://github.com/visionmedia/supertest>)



<https://www.dropbox.com/s/cmlo0c7dp5f1qp8/tony-stark-nodejs-starter.zip?dl=0>

The project is using embedded SQLite3 database. Feel free to change it to use RDBMS (MySQL, PostgreSQL) of your choice.

Modify the content/configuration of these files:

* .env
* src/config/database.js

Run these commands:

* npm install
* npm run db\_migrate
* npm run db\_seed\_all
* npm run dev
* npm run test

The starter project is a combination of REST API and Web UI application.

The REST API is defined as below.



<https://www.dropbox.com/s/0dzgy7wr4can2bw/product_rest.zip?dl=0>

When using the Postman collection, remember to edit/change the collection variable "url\_api", to use the correct path.

This project is using EJS (<https://ejs.co/>) as the UI presentation framework. You are allowed to remove it and replace with a JavaScript framework of your choice, that you are familiar with, e.g.

* Angular (<https://angular.io/>)
* Vue.js (<https://vuejs.org/>)
* React (<https://reactjs.org/>)
* others

Screenshots provided below, are served as guideline and display the minimum requirements. Feel free to express your creativity and enhance accordingly.

Resources from internet

* <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/development_environment>
* <https://swagger.io/specification/>
* <https://en.wikipedia.org/wiki/OpenAPI_Specification>
* <https://en.wikipedia.org/wiki/Web_Application_Description_Language>
* <https://www.postman.com/downloads/>

|  |
| --- |
| **Coding Tasks** |

**Guidelines**

1. Complete as many tasks as possible, within the stipulated time frame (~ 1 mandays).
   * If you are of junior level, complete at least one task.
2. Provide the project source code, minus theses: "node\_modules", "dist", "\*.log".
3. Code only in TypeScript (<https://www.typescriptlang.org/>).
4. Grading criteria:
   * Correctness (functioning, bug free, performant)
   * Style (code formatting, naming convention, readability)
   * Comprehensiveness (flexible, reusable, validation, unit tests)

**Task 1**

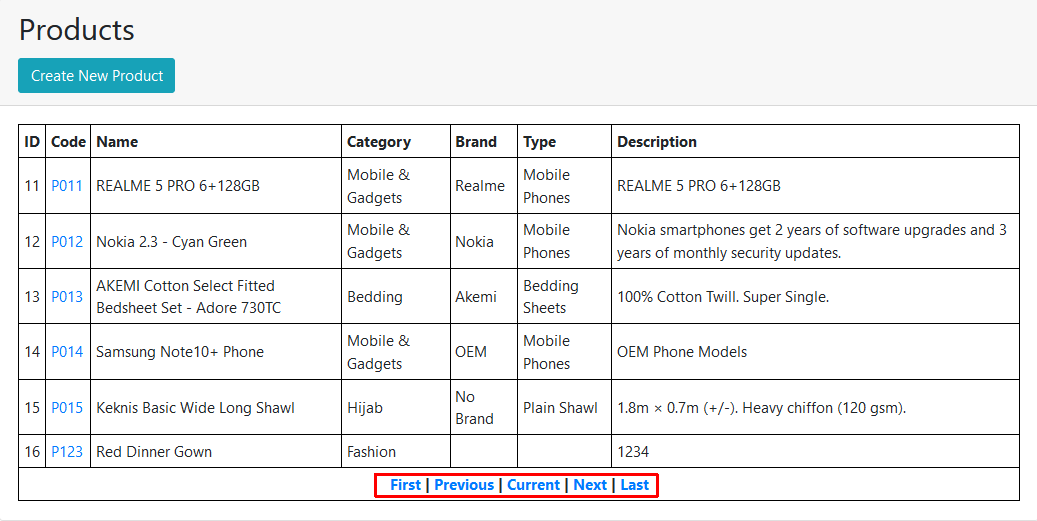
* Implement the REST API which provides endpoints for caller to interact with the "product" resource, according to the specification given (refer to the YAML files above).
* Implement the missing functionalities.
  + listing of products (with pagination support) => GET /api/products
  + retrieve single product => GET /api/products/{code}
  + create new product => POST /api/products { request body in JSON }
  + update product => PUT /api/products/{code} { request body in JSON }
  + delete product => DELETE /api/products/{code}
* For "listing of products", change it to accept 2 new parameters "sort" and "dir" at query string level.
  + "sort" will be the field name of the record, e.g. "code", "description", "name", etc
  + "dir" are these values: "asc", "desc"
  + The enhance functionality is to allow retrieving paginated listing and sort it by one of the fields, either ascending or descending.
  + GET /api/products?page=1&size=5&sort=name&dir=desc
  + "dir" will be "asc", "ascending" by default..
  + "sort" will be "id", by default.
  + If none available, then default to "id desc".
* Use the Postman collection to conduct testing.

**Task 2**

* Fix the current unit tests (npm run test), after finish the implementations above.
* Add in more unit tests to test different scenarios.
  + Missing parameter ("code") when doing "update", "delete", "retrieve single".
  + Missing record in database.
  + Fail to insert / update / delete record.

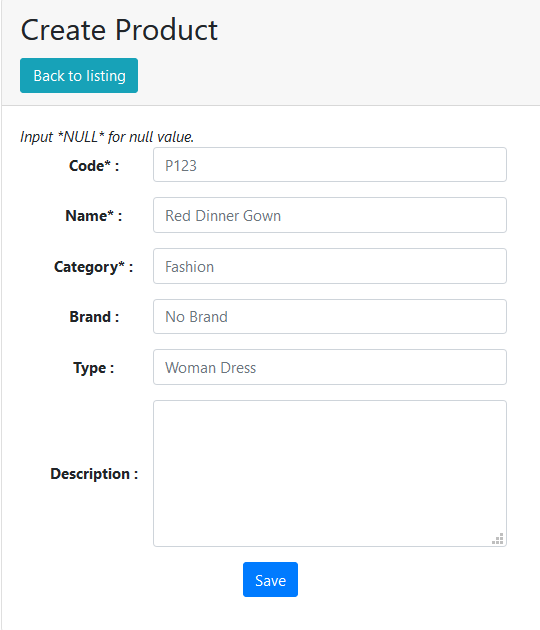
**Task 3**

* Build a HTML page to display the list of the products from the data store, in tabular format.
* Must show (at least) these product information (name, category, description; refer to model definition) as table columns.
* Must have pagination (<https://www.google.com/search?q=pagination+listing>)
  + "page": page number counting starts from 1
  + "size": default of 10 records per page and maximum 25 records per page



**Task 4**

* Build a "create product" (FORM POST) page/component to add new product item.
* Should have proper validation logic and display error/success messages.
* It can be modal popup dialog style or it can be shown on separate page.
* Implement the corresponding logic at the server side to allow adding new product item to the data store.



**Task 5**

* Build an "edit product" (FORM POST) page/component to update existing product item.
* Should have proper validation logic and display error/success messages.
* It can be modal popup dialog style or it can be shown on separate page.
* Implement the corresponding logic at the server side to allow update of product item from the data store.

