**FULLSTACK END OF MODULE PROJECT**

| **Point-Of-Sale System** |
| --- |

**Description**: The aim of the FULLSTACK Final Project is to create a fully dynamic back end application. The application you are expected to build is a Point-Of-Sale system that will be linked to the JavaScript Front end End of Module Project. You can go to the *google play store* and download **Yoco Point Of Sale** so that you can see a working example of a Point-Of-Sale system.

**Expectations**: You are expected to produce a fully working full stack web application consisting of both the backend(db & api) using Express and MySQL OR MongoDB. The front end section will be done during using html, css, Vue & javascript. The backend is done using Express.JS /MySQL OR MongoDB. Your application must be deployed on Heroku.

For the application, a user is expected to log in before using the application. A user will then be able to sort products by category, see a list of products, create/update/delete products, add products to a customer for payments, calculate total cost for customers, and update products.

**Hosting & Deployment**

Your project code should be hosted on Github, and deployed on Heroku(Backend) and Netlify/Firebase(Front-end). A submission is only valid when both the Github, Heroku and Netlify OR Firebase links have been sent to the LMS and are fully functional.

## **Functionality**

* Login/Registration/Account
* Product CRUD (Create, Select, Update, Delete SQL functionalities), system
* Price Calculation
* Form Handling / Validation
* Event Handling
* Image hosting

**User Stories:**

**(API)**

* Users ( admin ) should be able to register an account.
* Users ( all users ) should be able to login.
* Users ( all users ) should be able to view their profile.
* Users (only admins ) should be able to add new products
* Users (all users) should be able to view available products
* Users (only the admins) should be able to delete products
* Users should be able to calculate total prices

NB: (For this, you need to provide a functionality that will display the total amount)

NB: Idea since your cart will be stored as an array of objects, then you can make use of the Array.reduce() to return the sum of the product's

**(Front-end)**

* Users (only the admin) should be able to use the register page(register form).
* Users (all the users) should be able to see the login page(A form with username and password).
* Users should be able to login to the system.
* Users should be able to view products.
* Users should be able to create and edit products.

**Bonus Features**

* Users should be able to filter their own products after login.
* Users should be able to filter products by type.
* The frontend design should conform to the highest standards.

The technical stack for the project **should be** the one listed below only.

**Tech Stack**:

**Backend**

* Express Web Framework(, SQL database( (MySQL) or NoSQL (MongoDB)

**Frontend**

* HTML, CSS, JavaScript, Vue

All of the following api endpoints **should be fully** implemented.

**Root/Home**:

| GET / |
| --- |

**Users:**

| GET /users  GET /users/:id  POST /users  PATCH /users  PUT /users/:id  DELETE /users/:id  GET /users/:id/cart  POST /users/:id/cart  PUT /users/:id/cart  DELETE /users/:id/cart |
| --- |

**Products:**

| GET /products  GET /products/:id  POST /products  PUT /products/:id  DELETE /products/:id |
| --- |

**Resources**

* [API Documentation Express](https://expressjs.com/en/5x/api.html)

## 

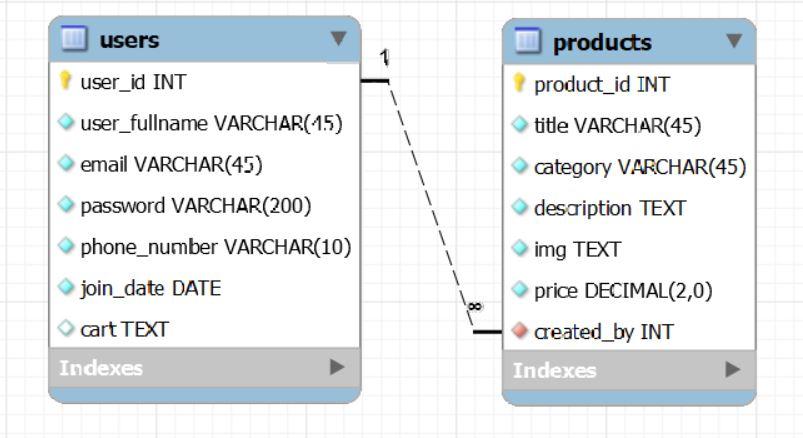
## 

## 

## How to structure data

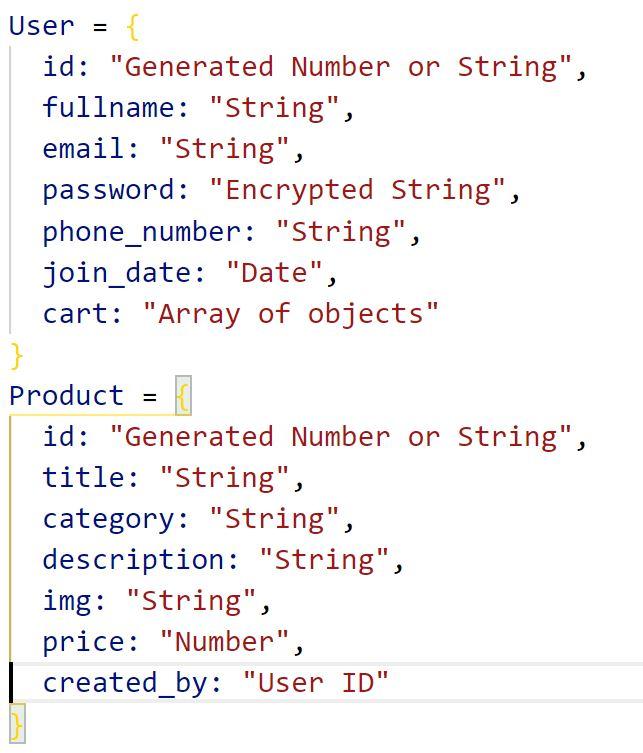
### SQL Database

If you are using MySQL or any other SQL database, please set up the data as follows:



### No SQL Database

If you are using MongoDB, please structure your Mongoose Schemas as such:



## Object Examples

User



Product

