LAB 5

GREENS ARE BONUS

Implement an inventory management app where users can login and do simple CRUD operations on their products.

- A Product must have name, quantity, category and the creation and update time.
- A User must have username, password, firstName and date of birth (optional).
- A User can sign up, login, and add, edit, view or delete his own products
- Products status should depend on quantity
  Quantity > 2 ===> status = “available”
  0 < Quantity <= 2 ===> status = “low stock”
  Quantity = 0 ===> status = “out of stock”

Note:
Listing products should return status of each product

1- Create two models (user, product)
User : {
username: String, required, unique, min 8
firstName: String,required, min length 3, max length 15
lastName: String,required, min length 3, max length 15
password : String, required,
dob: Date, optional
createdAt: Date, timeStamp,
updatedAt: Date, timeStamp
}
Product {
owner: the ObjectId of the user,
name: String,unique per user, required, min 5, max 20,
categories:[String], optional, default is [“General”]
quantity: Number, required, min: 0
createdAt: Date, timeStamp,
updatedAt: Date, timeStamp
}
BONUS: Use autoIncremental id instead of mongo id
3 - Implement the following end points.

HTTP Method	route	Description
post	/users	- Register a user with the following **required** attributes Username,password , firstName, lastName Notes: - Return registered user with token if success - Handle validation errors returned from mongo
Post	/users/login	Return (user with token }) Don’t return password If the the authentication failed

		Return error with 401 status code
GET	/users	Return the first name of registered users
DELETE	/users/:id	Delete the user with selected id
PATCH	/users/:id	- Edit the user with the selected id - Return ({message:"user was edited successfully", user: theUserAfterEdit"}) if success - Handle validation errors returned from mongo
POST	/products	Create new product (name, category, quantity) You must save it with userId of the logged in user Return the new product to the user
PATCH	/products/:id	Edit product
PATCH	/products/:id/stock	Restock or destock product using {operation: "restock" or "destock", quantity: number}
DELETE	/products/:id	Delete product
GET	/users/:userId/products	Return the products of specific user
GET	/products?limit=10&skip=0&status=$value	Return the products with specific required filters (status, defaults are limit 10 skip 0) (don't reinvent the wheel)

4- Protect all endpoints with a proper authentication layer. (except for registration and login of course!)

ACL (Authorization)
1- Each user can only get/edit/delete his products.
2- Each user can only get/edit himself.
3- Add an admin role who can get/edit/delete any product.