# CS 387 - Lab 5 - IITB Online store in Node.js and Postgres

This lab is meant to expose you to technology many of you will use in your projects. Node.js applications connecting to a backend database. Doing this assignment will help you develop reusable code modules that you can employ directly in your projects as well should you choose to go the route of Node.js.

Before starting this assignment, go through the following:

**Node.js:**
You can learn more about Node.js here.

**Node.js Setup:**
For Node.js development it is recommended to use Virtual Studio Code or Intellij Idea Ultimate. You don't HAVE to use these, you can do it just as well with a text editor such as VIM or Emacs but these IDEs make life simpler.

**Install Instructions:**

```
> sudo apt update
> sudo apt install nodejs
> nodejs -v (Confirm that the installation was successful )
> sudo apt install npm
> mkdir ebay && cd ebay  // this is for the specific application you will build for this assignment.
> npm init
> npm -v         // (Verify the installed version)
```

**Node.js Modules Installations**: **here**
**Node.js Hello World:** here
**Node.js Callbacks**: here
**Node.js Express**: here
**EJS**: here

In this assignment you will be developing a Node.js application similar to this(Click the link and use this application first). Finally you should be able to create an application which will support the following user interaction.

User: Open **`localhost::3000/admin/add-product`** method type: 'GET'(already implemented)
App: Returns a form
User: Fills the form, clicks **AddProduct** button, method type: 'POST' to /add-product (already implemented for you as a example of how to code a Node.js action)
App: Adds this new product to the inventory and stays on the same /add-products page.
User: Open **`localhost:3000/prods`** method type: 'GET'
App: Returns list of products available in the inventory, along with details of each product such as image, title, quantity, price.
User: Clicks "Add to cart" adds the item into the cart if that product has quantity available in the inventory
App: Adds the item to the cart and redirects the user to /cart
User: Open **`localhost:3000/cart`** method type: 'GET'
App: List all products added to cart by the user, show credits remaining and buy buttons to purchase the cart.
User: Click on the **buy** button in carts page, method 'POST' redirect to **`localhost:3000/orders`** on completion.
App: If not sufficient credit stays on the same page, else cart and orders are updated.
User: Open **`localhost:3000/orders`** method type: 'GET'
App: Lists the order history of the user.

**1: Project Setup**

For node.js installation refer here, now clone the sample project folder from here, which already implements adding a product to cart and in this assignment you will build on this code, install all the dependencies using the command below.

**git clone https://github.com/sanarah/LAB5.git**
**cd path_to_LAB5 && npm install**

**2: Understanding the Code**

The code **app.js** basically builds an express server which listens on port 3000.It also routes the "/admin/*" http requests to corresponding controller functions.

The sample code that you have right now implements an architecture known as MVC, if you consider the request http://localhost:3000/admin/add-product, it is routed to the function **get_test**(controller) which fetches data using Class Prod(model) and finally shows the page add_product.ejs which is the view users of the application see.

As you can see that extension of html files here is .ejs which means we are using ejs engine to dynamically render html files.

From an application perspective we don't want users to see our private files, so a folder public is created in which we keep files that can be accessed.

The code database.js in the folder **utils** connects to the database which stores the products, for this you have to configure the files with the corresponding values of postgres on your computer. Add product button adds product to the inventory

After reading all this you should be able to understand the complete code i.e how to route requests, how to setup servers, how to setup controllers, how to connect to databases, how to render html dynamically, how to fetch/update data.

For this app we have to store cart, products, orders, user information in the database. The following file contains the database schema required for this app. For this app we hardcoded a default user in the database. Any action on the cart, orders, credit will be done to this user.

## 3: The Products Page

This page has to display all the products from the inventory,and attributes like title, price, quantity available, image for each of the products and each product should have an **Add to Cart Button** from where we can place the item into the cart. This page has to be served at path /prods. Follow the mvc model used by add-product which essentially means creating a route and a controller function and model.

## 4: The Cart Page

This page will display all the items of the cart which are added by the user. And this page also shows the current credit that the user has.This page has to be served at path /cart. When the user clicks on the buy button all the items available in the cart will be executed/bought and will show in the orders page. Essentially you should have a table in the database which can store the cart details of all the users. Follow the mvc model used by add-product which essentially means creating a route and a controller function and model.

**Constraints:**

- **The order will only be executed if the user has available credits more than or equal to the cart price.**
- **Order won't be executed if the particular item quantity is more than the available quantity in the inventory.**
- **User credits have to be decremented after the purchase order and cart and order tables have to be updated accordingly.**

## 5: The Orders Page

This page will display all your executed orders done by the particular user.This page has to be served at path /orders, for each type of product you have to show its title, quantity purchased, image, and price.Essentially you should have a table in database which can store the order details of all the users.

## Submission Instructions:

- Make a folder named **<rollnumber1>-<rollnumber2>-a5**
- Your submission folder should contain your application files, refer to the file tree here.

```
-<rollnumber1>-<rollnumber2>-a5
        -LAB5
                -controllers
                        -put all your controller files(.js) here
                -models
                        -put all your models files(.js) here
                -public
                        -put all your public files such as css/others here
                -routes
                        -put all your routing logic files(.js) here
                -utils
                        -put all your util files here(.js)
                -views
                        -put all your ejs files here.(.ejs)
                -app.js
                -package-lock.json and package.json
```

- **Zip** the folder and upload it to the **moodle**
- Before submitting, check all the files and run your code one more time to ensure that everything is working fine.

## Grading Rubric

| Exercise | Marks | |
|---|---|---|
| 1 | Products page | 8 |
| 2 | Add to cart button | 6 |
| 3 | Cart page | 8 |
| 4 | Buy button | 18 |
| 5 | Orders Page | 8 |
| | Total | 50 |