

27 August 2021

# Project Name: Capstone Project

E-Commerce Online Purchasing Website

Made By Karabo Pheko  
8-27-2021

1. **Name:** Karabo Pheko
2. **Date:** 27 August 2021
3. **Project Name:** Capstone-Project (E-Commerce Website)
4. **Description:** A dynamic online shopping of groceries platform/website.
5. **Technologies:**
  - VS Code
  - Notepad++
  - Amazon Web Services, EC2 (Ubuntu) Instance (Host mean stack website for free)
  - Docker (Containerizing the angular application)
  - Node JS
  - Angular (Frontend)
  - MongoDB (Database)
  - Express JS (Backend)
  - Nginx Server (reverse proxy, load balancing)
  - Putty and PuttyGen (Connecting to EC2 Ubuntu Instance via SSH)
  - Git and Github (Version control and project management)
  - Postman and RestEasy (Making API requests)
6. **GitHub Links:**
  - Front-end Angular Application: <https://github.com/PhekoK/capstone-project.git>
  - Backend Express Server: <https://github.com/PhekoK/server.git>
7. **Public IPv4 DNS:**

<http://ec2-18-119-139-156.us-east-2.compute.amazonaws.com:80>

## STEPS CREATING PROJECT – FROM DEVELOPMENT TO PRODUCTION

### - Development

#### 1. Backend - Express server.

- Created Express Server and Mongoose models.
- User and Product Model

The screenshot shows the Visual Studio Code interface with two code files open side-by-side:

- userModel.js:** Contains the following code:

```
1 var mongoose = require('mongoose');
2
3 var userSchema = mongoose.Schema({
4   firstName: String,
5   lastName: String,
6   dob: String,
7   phoneNumber: String,
8   email: {type: String, unique: true },
9   password: String,
10  confirmPassword: String
11 }, { versionKey: false });
12
13 module.exports = mongoose.model('User', userSchema);
```

- productModel.js:** Contains the following code:

```
1 var mongoose = require('mongoose');
2
3 var productSchema = mongoose.Schema({
4   name: String,
5   price: Number,
6   availability: String, //In Stock or Out-of-Stock
7   quantity: Number,
8   image: String,
9   brand: String,
10  size: String,
11  category: String,
12  SKU: String,
13  date_added: {
14    type: Date,
15    default: Date.now
16  }
17 }, { versionKey: false });
18
19 /* productSchema.method("toJSON", function() {
20  const { _v, _id, ...object } = this.toObject();
21  object.id = _id;
22  return object;
23}); */
24
25 module.exports = mongoose.model('Product', productSchema);
```

- Added API endpoints(routes) to see if my routing is working.
- Also made use of httpError handler to catch http errors to prevent server from crashing and restarting manually.
- Product route

```

products.js
routes > JS products.js > ...
1 var express = require('express');
2 var router = express.Router();
3 var Product = require('../models/productModel');
4 /* GET items listing. */
5 router.get('/', function(req, res, next) {
6   Product.find((err, data) => {
7     if(err) throw err;
8     res.send(data);
9   });
10 });
11 */
12 });
13 */
14 //Request URL by Id: http://localhost:3000/products/:id
15 router.get('/:id', (req, res) => {
16   Product.findById(req.params.id, (err, data) => {
17     if(err) throw err;
18     if(!data)
19       return res.status(404).send('Product Not found with given ID');
20     res.send(data);
21   });
22 });
23 */
24 // PUT request to edit
25 router.put('/:id', (req, res) => {
26   Product.findByIdAndUpdate(req.params.id, (err, data) => {
27     if (err) throw err;
28     if(!data)
29       return res.status(404).send("Product with given ID does not exist!!");
30     Product.findByIdAndUpdate(req.params.id, req.body, (err, data) => {
31       if (err) throw err;
32       res.send(data);
33     });
34   });
35 });
36 */
37 //Request URL http://localhost:3000/products
38 // Header : Content Type: application/json
39 //Request Body: { "name": "Face Cloth", "price": 10, "availability": "In Stock", "quantity": 40, "image": "https://norwoodhome.co.za/wp-content/uploads/2020/10/1-ply-toilet-paper-twinsaver", "size": "48 rolls", "category": "Toiletries , Household", "SKU": "NRU12110" }
40 */
41 */
42 */
43 */
44 */
45 */
46 */
47 */
48 */
49 */
50 */
51 */
52 */
53 */
54 */
55 */
56 */
57 */
58 */
59 */
60 */
61 */
62 */

users.js
routes > JS users.js > ...
1 var express = require('express');
2 var router = express.Router();
3 var User = require('../models/userModel');
4 /*
5  * GET users listing. */
6 router.get('/', function(req, res, next) {
7   User.find((err, data) => {
8     if (err) throw err;
9     res.send(data);
10   });
11 });
12 */
13 */
14 /* GET Users by Id http://localhost:3000/users/:id */
15 router.get('/:id', (req, res) => {
16   User.findById(req.params.id, (err, data) => {
17     if(err) throw err;
18     if(!data)
19       return res.status(404).send('User Not found with given ID');
20     res.send(data);
21   });
22 });
23 */
24 */
25 */
26 */
27 */
28 */
29 */
30 */
31 */
32 */
33 */
34 */
35 */
36 */
37 */
38 */
39 */
40 */
41 */
42 */
43 */
44 */
45 */
46 */
47 */
48 */
49 */
50 */
51 */
52 */
53 */
54 */
55 */
56 */
57 */
58 */
59 */
60 */
61 */
62 */
63 */
64 */
65 */
66 */
67 */
68 */
69 */
70 */
71 */
72 */
73 */
74 */

```

### - User route

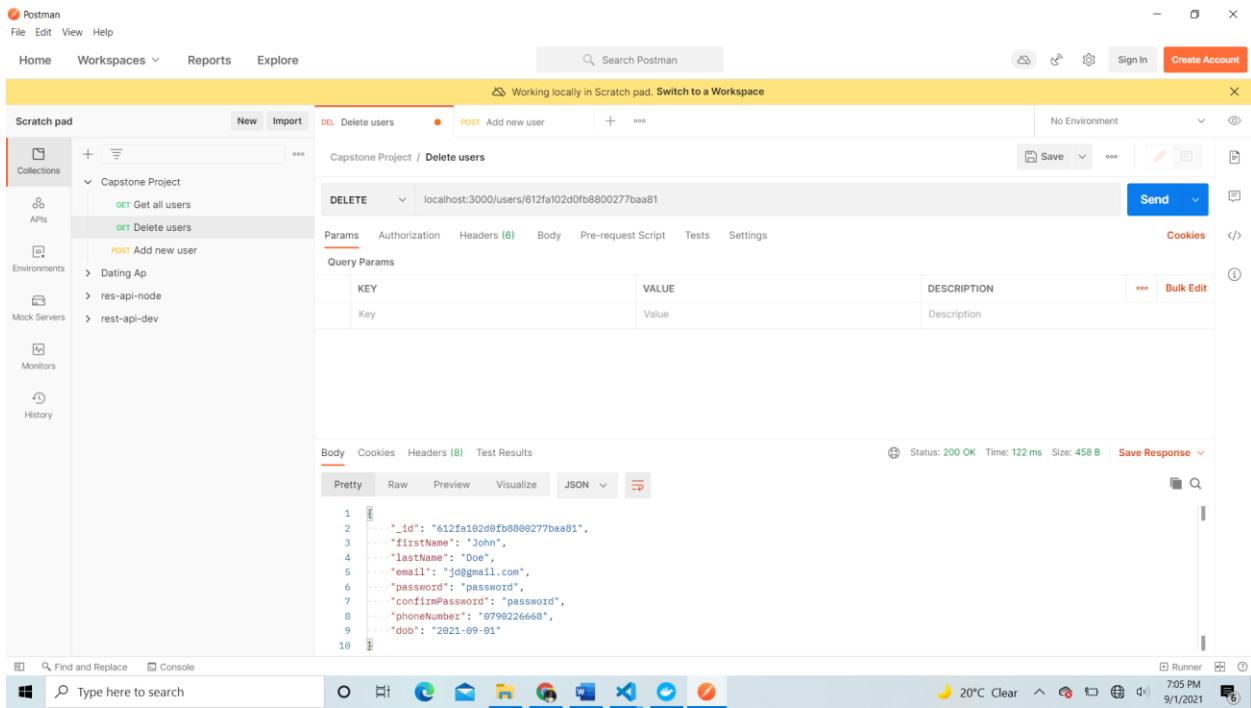
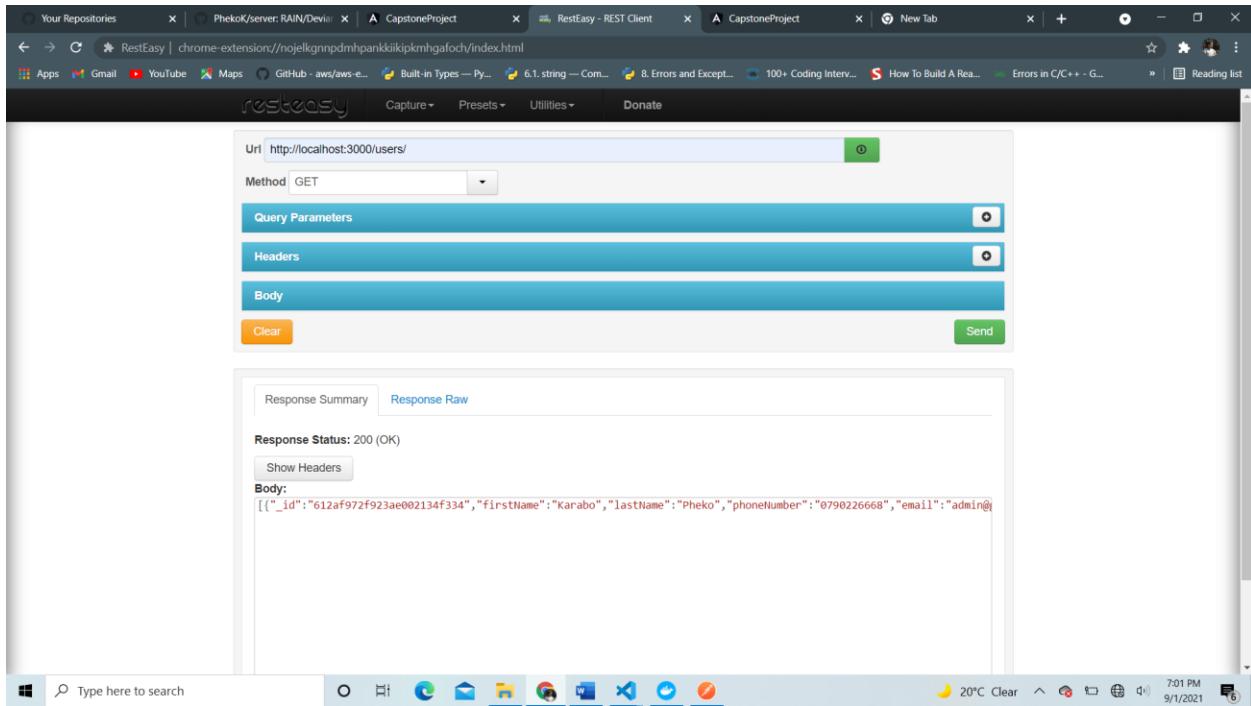
```

products.js
routes > JS products.js > ...
1 var express = require('express');
2 var router = express.Router();
3 var Product = require('../models/productModel');
4 /* GET items listing. */
5 router.get('/', function(req, res, next) {
6   Product.find((err, data) => {
7     if(err) throw err;
8     res.send(data);
9   });
10 });
11 */
12 });
13 */
14 //Request URL by Id: http://localhost:3000/products/:id
15 router.get('/:id', (req, res) => {
16   Product.findById(req.params.id, (err, data) => {
17     if(err) throw err;
18     if(!data)
19       return res.status(404).send('Product Not found with given ID');
20     res.send(data);
21   });
22 });
23 */
24 // PUT request to edit
25 router.put('/:id', (req, res) => {
26   Product.findByIdAndUpdate(req.params.id, (err, data) => {
27     if (err) throw err;
28     if(!data)
29       return res.status(404).send("Product with given ID does not exist!!");
30     Product.findByIdAndUpdate(req.params.id, req.body, (err, data) => {
31       if (err) throw err;
32       res.send(data);
33     });
34   });
35 });
36 */
37 //Request URL http://localhost:3000/products
38 // Header : Content Type: application/json
39 //Request Body: { "name": "Face Cloth", "price": 10, "availability": "In Stock", "quantity": 40, "image": "https://norwoodhome.co.za/wp-content/uploads/2020/10/1-ply-toilet-paper-twinsaver", "size": "48 rolls", "category": "Toiletries , Household", "SKU": "NRU12110" }
40 */
41 */
42 */
43 */
44 */
45 */
46 */
47 */
48 */
49 */
50 */
51 */
52 */
53 */
54 */
55 */
56 */
57 */
58 */
59 */
60 */
61 */
62 */

users.js
routes > JS users.js > ...
1 var express = require('express');
2 var router = express.Router();
3 var User = require('../models/userModel');
4 /*
5  * GET users listing. */
6 router.get('/', function(req, res, next) {
7   User.find((err, data) => {
8     if (err) throw err;
9     res.send(data);
10   });
11 });
12 */
13 */
14 /* GET Users by Id http://localhost:3000/users/:id */
15 router.get('/:id', (req, res) => {
16   User.findById(req.params.id, (err, data) => {
17     if(err) throw err;
18     if(!data)
19       return res.status(404).send('User Not found with given ID');
20     res.send(data);
21   });
22 });
23 */
24 */
25 */
26 */
27 */
28 */
29 */
30 */
31 */
32 */
33 */
34 */
35 */
36 */
37 */
38 */
39 */
40 */
41 */
42 */
43 */
44 */
45 */
46 */
47 */
48 */
49 */
50 */
51 */
52 */
53 */
54 */
55 */
56 */
57 */
58 */
59 */
60 */
61 */
62 */
63 */
64 */
65 */
66 */
67 */
68 */
69 */
70 */
71 */
72 */
73 */
74 */

```

- In Postman/RestEasy. Tested if API requests work before working on the client.



- App.js – contains the middleware and routes of the angular project

```

File Edit Selection View Go Run Terminal Help
OPEN EDITORS JS app.js ...
EXPLORER SERVER ...
OPEN EDITORS JS app.js ...
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
var mongoose = require('mongoose');
var cors = require('cors');
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
var productsRouter = require('./routes/products');

mongoose.connect("mongodb://localhost:27017/capstoneDB",
  { useNewUrlParser: true, useUnifiedTopology: true })
  .then(() => { console.log('Connected to Database!!!') })
  .catch((error) => { console.log(error) });

var app = express();

app.use(cors());
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/products', productsRouter);

module.exports = app;

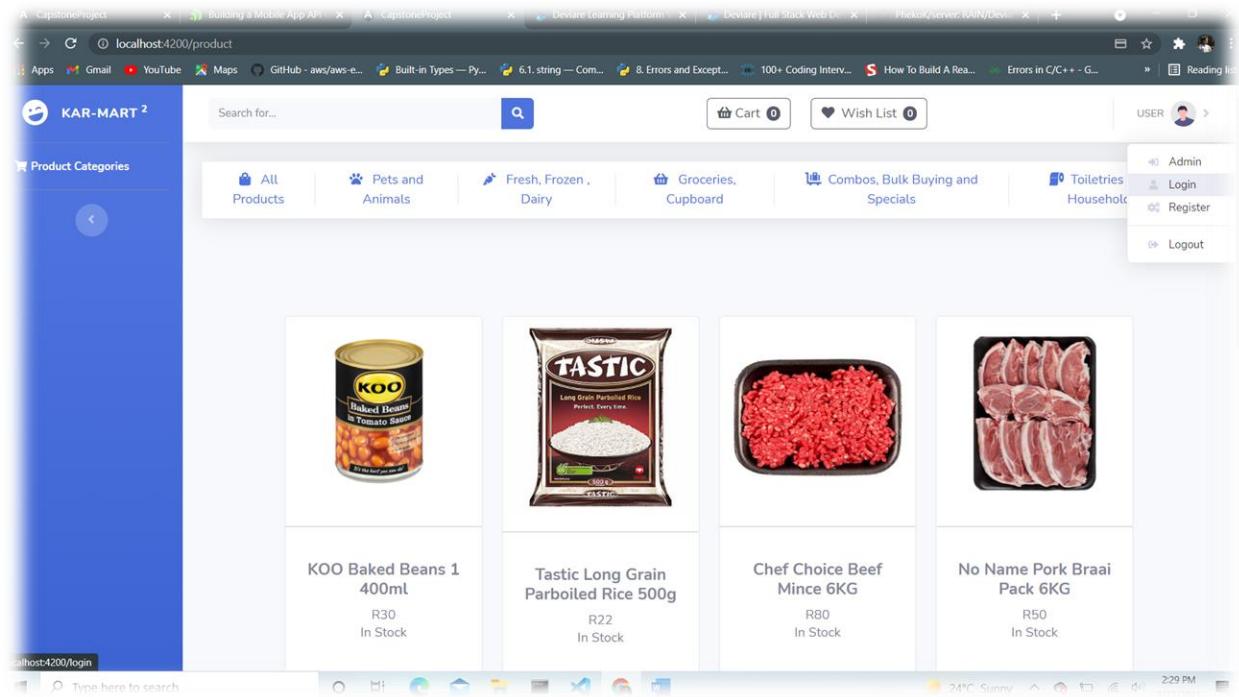
```

## 2. Frontend – Angular Application

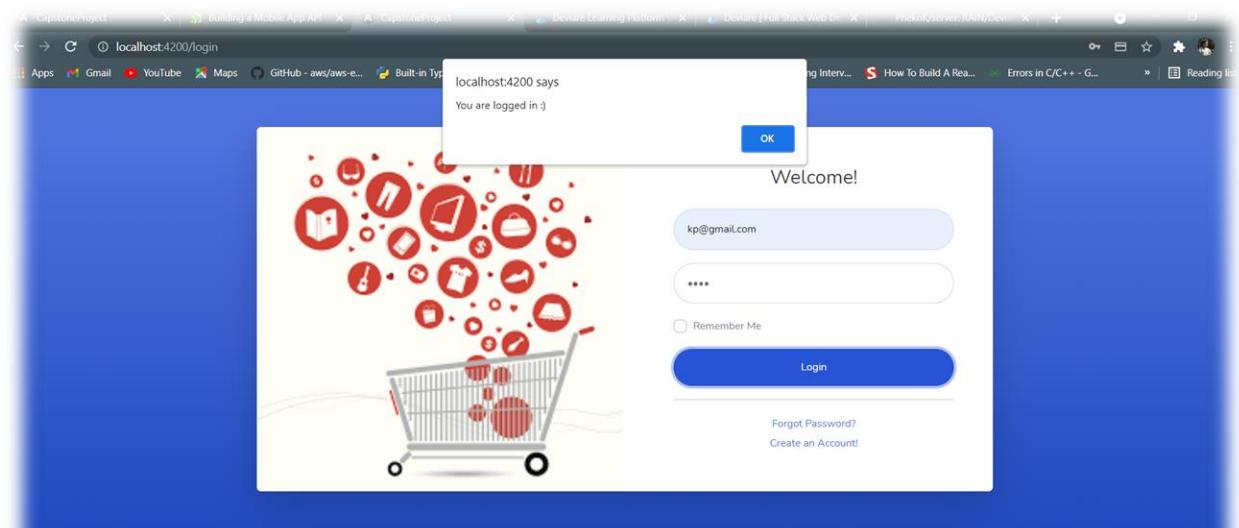
- Used bootstrap to enhance UI for both user and administrator portals.
- And used font-awesome icons.

### A. User Portal

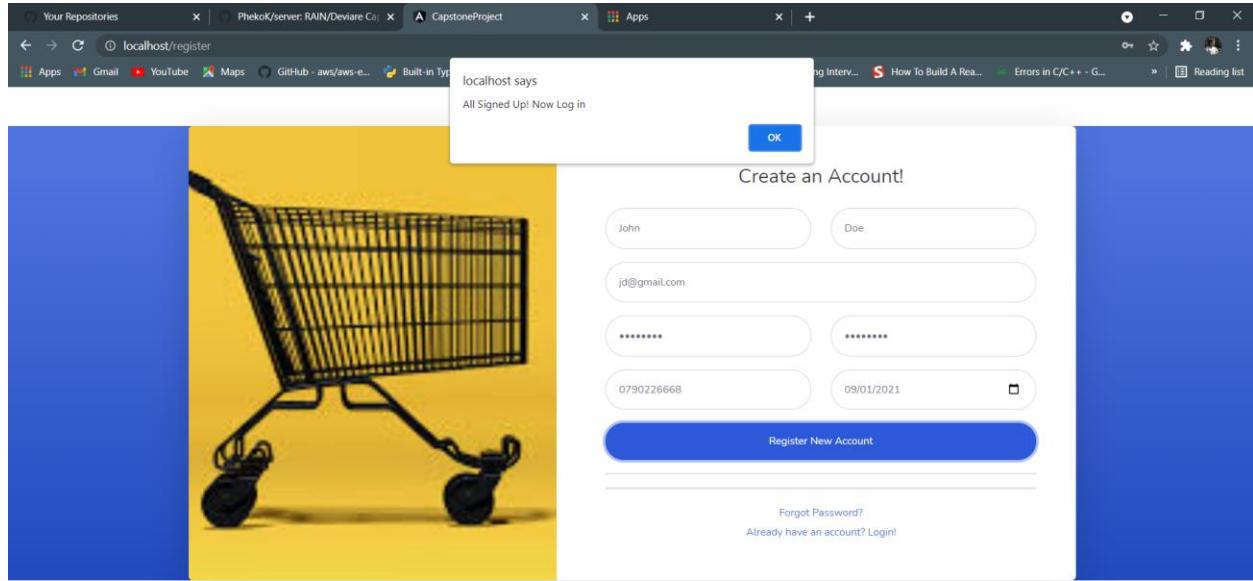
- The user can register and login with their credentials.
- View shopping items in product catalogue page to select which items to purchase.
- Add items to shopping cart, view cart items and delete them.
- If logged in, purchase and proceed to checkout page.
- Review invoice of purchased products.
- User may click the user button to either login/ register with their credentials.
- Product page – with products to choose from



- Login page
- User with existing account may log in. Once logged in, user will navigate to the product catalogue page.

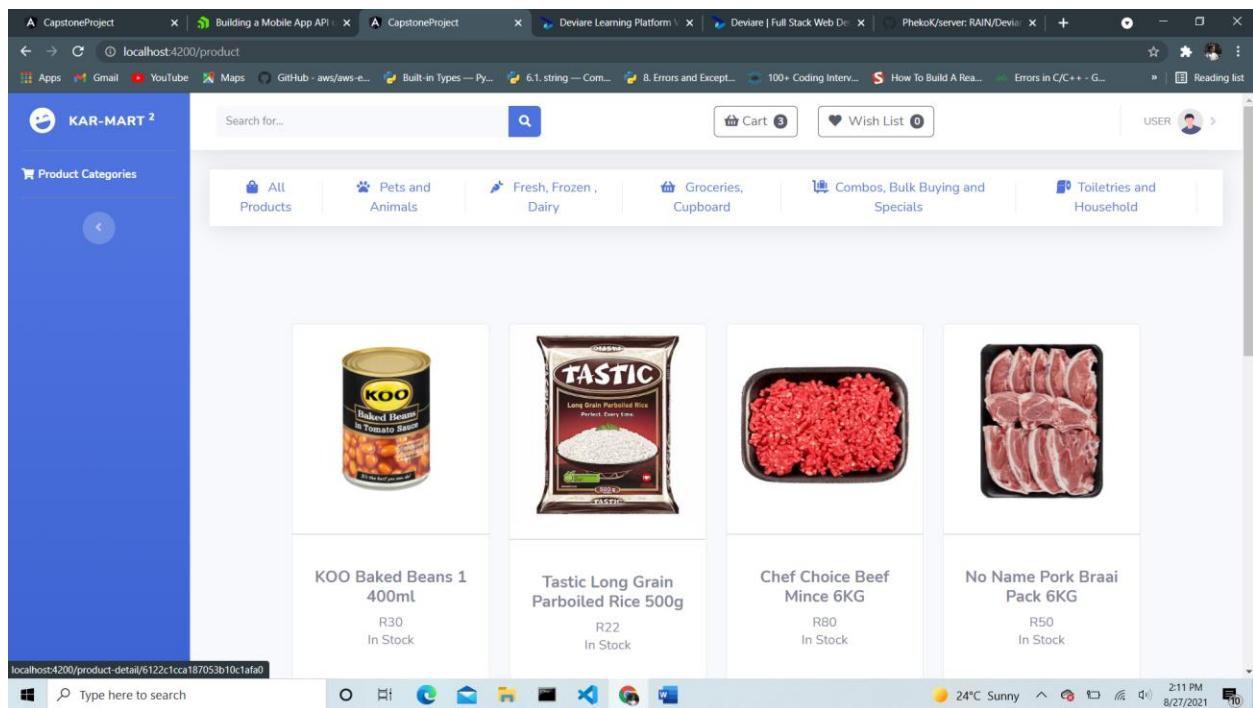


- Register page
- A new user may register their credentials and navigate to login page to sign in.



Product catalogue page.

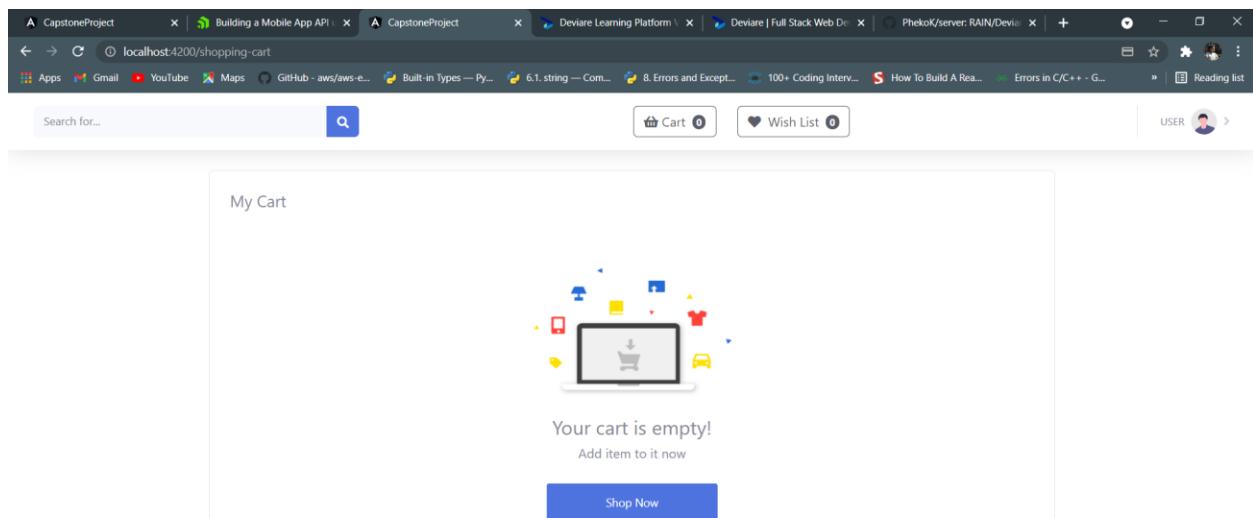
- The page is accessible for viewing items to all users (registered and unregistered) for the user to browse items to shop.
- Once user adds items to shopping cart and they are logged in, they can proceed to checkout page.



- When user added products in the cart, cart items are show as demonstrated below (if cart is not empty).

Sr.No	Product Image	Product Name	Price	Quantity	Total	Action
1		Long Grain Parboiled Rice	R22	1	R22	
2		Beef Mince	R80	1	R80	
3		Face Cloth	R10	1	R10	

- If cart is empty, user is prompted to product page and browse items, as shown below.



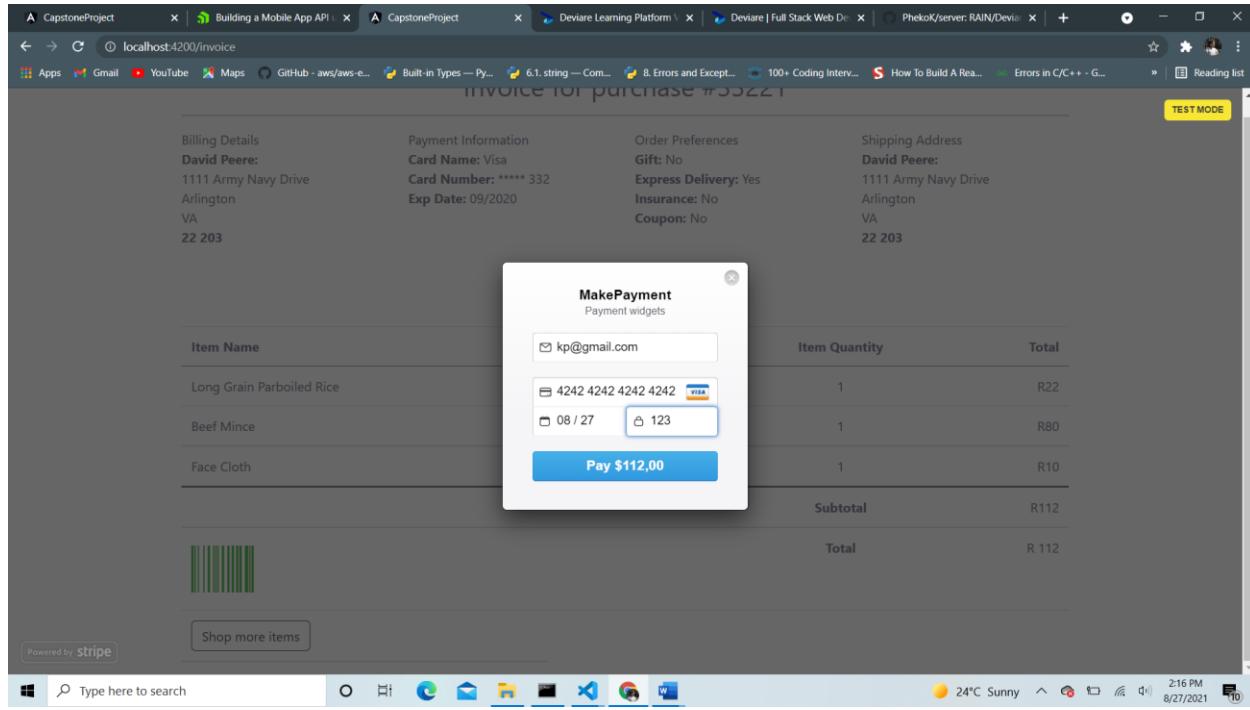
- The checkout page shows billing information that user needs to fill in for shipping/delivery.

A screenshot of a web browser window showing a checkout page. The URL in the address bar is 'localhost:4200/checkout'. The page has two main sections: 'Billing address' on the left and 'Your cart' on the right. In the 'Billing address' section, there are input fields for First name ('Karabo'), Last name ('Pheko'), Email ('kp@gmail.com'), Address ('7128 Czarevich Street, Braamfischerville Phase 2'), and Address 2 ('Apartment or suite'). Below these fields are two checkboxes: 'Shipping address is the same as my billing address' and 'Save this information for next time'. At the bottom of this section are two buttons: 'Shop more' and 'Pay'. In the 'Your cart' section, there is a summary of three items:

Item	Description	Price
Long Grain Parboiled Rice 500g	Groceries, Cupboard	R22
Beef Mince 6KG	Fresh, Frozen, Dairy	R80
Face Cloth 1s	Toiletries, Household	R10

A 'Total (Rands)' of 'R112' is shown. Below the cart summary is a 'Promo code' input field and a 'Redeem' button. The Windows taskbar at the bottom of the screen shows the date and time (2:15 PM, 8/27/2021) and various pinned icons.

- The 'pay' method invokes a method that navigates to the invoice page about the shipping information and an alert generated by stripe for 'card' payments.



- Once verified, a stripe token will be generated, and user will navigate to invoice page.

The screenshot shows a web browser window with multiple tabs open. The active tab displays an invoice for purchase #55221. The invoice details include:

- Billing Details:** David Peere, 1111 Army Navy Drive, Arlington, VA 22203
- Payment Information:** Card Name: Visa, Card Number: \*\*\*\* 332, Exp Date: 09/2020
- Order Preferences:** Gift: No, Express Delivery: Yes, Insurance: No, Coupon: No
- Shipping Address:** David Peere, 1111 Army Navy Drive, Arlington, VA 22203

A modal window titled "MakePayment" is overlaid on the page, prompting for payment information. The fields shown are:

- Email: kp@gmail.com
- Card Number: 4242 4242 4242 4242
- Expiration Date: 08 / 27
- CVV: 123

The main invoice table shows the following items:

Item Name	Item Quantity	Total
Long Grain Parboiled Rice	1	R22
Beef Mince	1	R80
Face Cloth	1	R10
<b>Subtotal</b>		R112
<b>Total</b>		R 112

At the bottom left is a barcode icon, and at the bottom right is a "Shop more items" button. The status bar at the bottom indicates "Powered by stripe".

- Invoice page is shown, displaying details of users order summary and shipping details

The screenshot shows a web browser window with multiple tabs open. The active tab displays an invoice for purchase #55221. A modal window is visible with the message "localhost:4200 says Stripe token generated!" and an "OK" button.

The invoice details are identical to the previous screenshot:

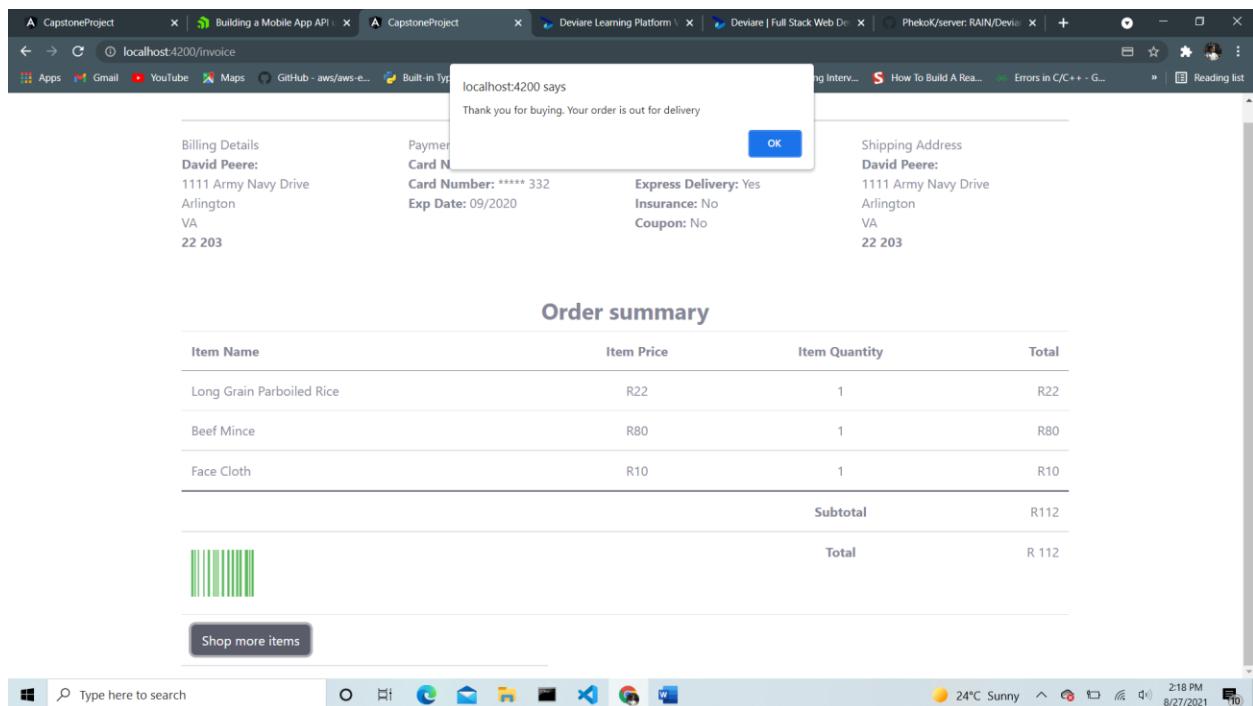
- Billing Details:** David Peere, 1111 Army Navy Drive, Arlington, VA 22203
- Payment Information:** Card Name: Visa, Card Number: \*\*\*\* 332, Exp Date: 09/2020
- Order Preferences:** Gift: No, Express Delivery: Yes, Insurance: No, Coupon: No
- Shipping Address:** David Peere, 1111 Army Navy Drive, Arlington, VA 22203

The main invoice table shows the following items:

Item Name	Item Price	Item Quantity	Total
Long Grain Parboiled Rice	R22	1	R22
Beef Mince	R80	1	R80
Face Cloth	R10	1	R10
<b>Subtotal</b>			R112
<b>Total</b>			R 112

At the bottom left is a barcode icon, and at the bottom right is a "Shop more items" button. The status bar at the bottom indicates "Type here to search" and "24°C Sunny 8/27/2021".

- When done. User may click “shop more” button to navigate to the product portal page. Or sign out.



## B. Administrator Portal

- An Admin may login and register their credentials.
- Uses email: [admin@gmail.com](mailto:admin@gmail.com) and password : admin to access the admin portal.
- Once logged in. They oversee users and products.
- Add new users, update user credentials, and delete them.
- Add new products, update, and delete them.
  
- users – admin can add new user, update current user and delete existing user.

Dashboard

User Management

#	First Name	Last Name	Date of Birth	Phone Number	Email	Password	Action
1	John	Doe	15-11-1997	1234567890	jd@user.com	pass	
2	Karabo	Pheko	2021-08-03	0780112568	kp@gmail.com	pass	
3	Sandy	Sun	1999-06-24	0121451789	ss@gmail.com	password	
4	admin	admin	2021-08-11	0119887605	admin@gmail.com	admin	
5	New User	Nu	2000-05-14	0112568792	nu@user.com	pass	

Copyright © Made By Karabo Pheko @ 2021

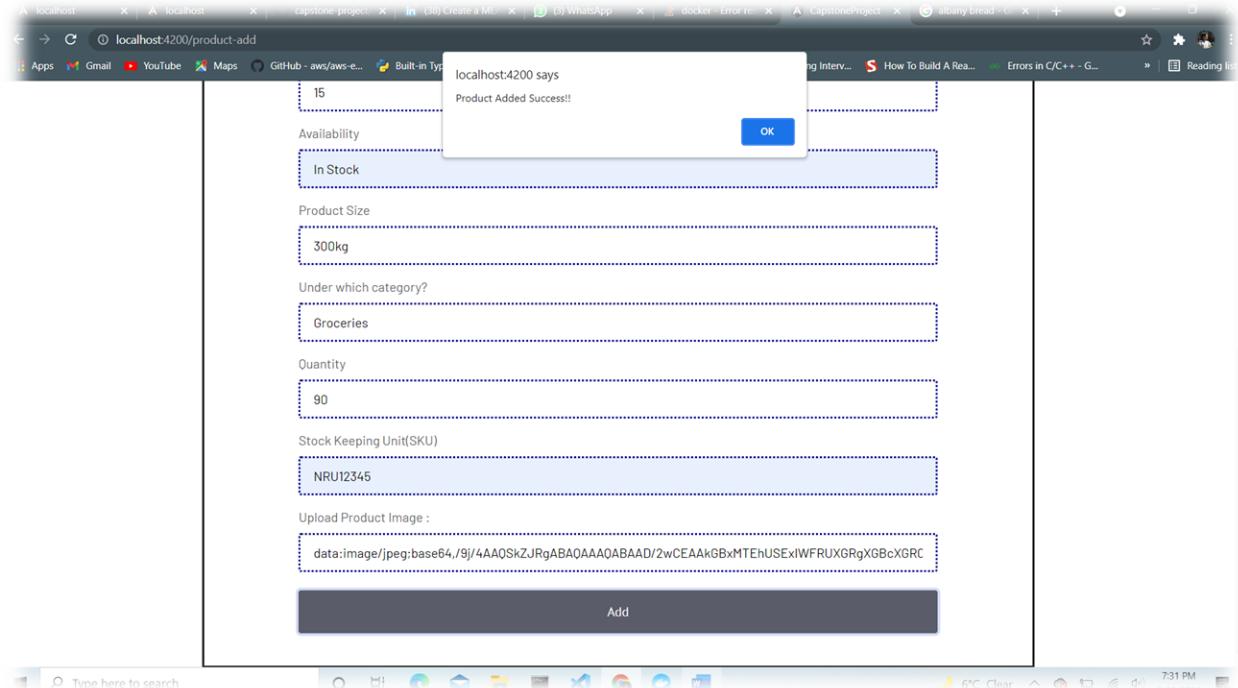
Products: - admin can add, delete, and update products.

Dashboard

Stock Management

#	Image	Product Name	Price	Availability	Quantity	Brand	Size	Category	Stock Keeping Unit (SKU)	Action
1		Baked Beans 1	30	In Stock	50	KOO	400ml	Groceries, Cupboard	NRU11111	
2		Long Grain Parboiled Rice	22	In Stock	50	Tastic	500g	Groceries, Cupboard	NRU20010	
3		Beef Mince	80	In Stock	20	Chef Choice	6KG	Fresh, Frozen, Dairy	NRU10050	
4		Pork Braai Pack	50	In Stock	30	No Name	6KG	Fresh, Frozen, Dairy	NRU12550	

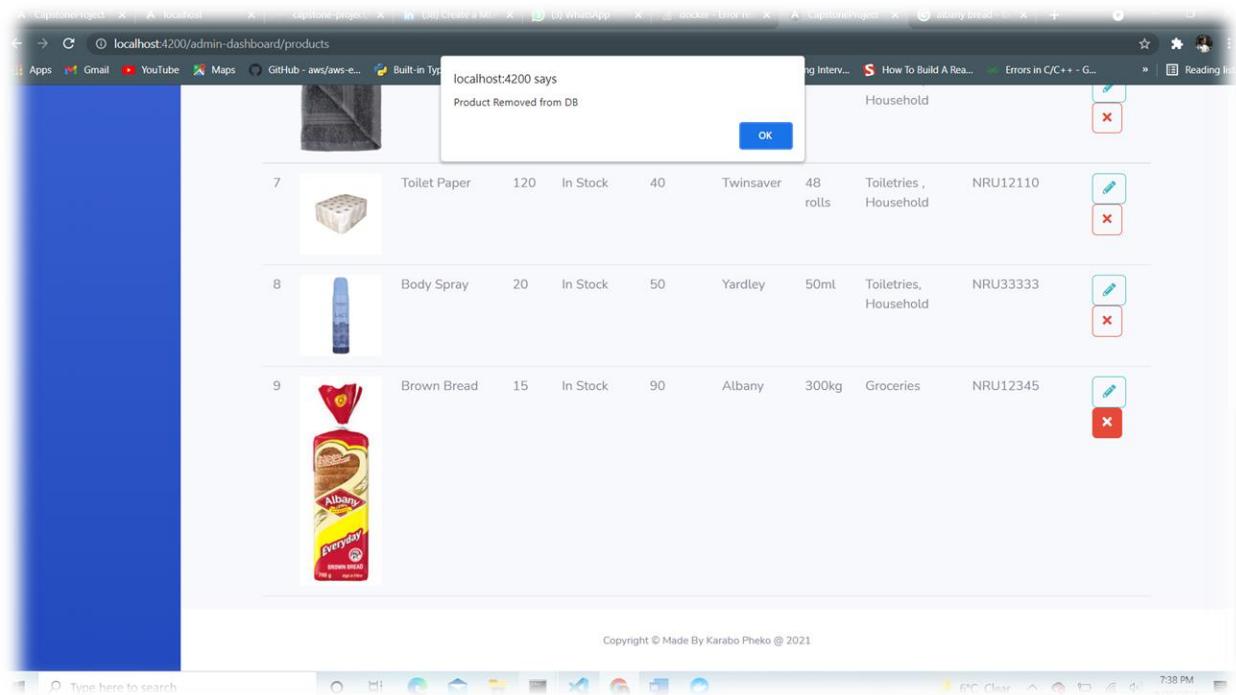
- Product Add – Admin can add a new product item.



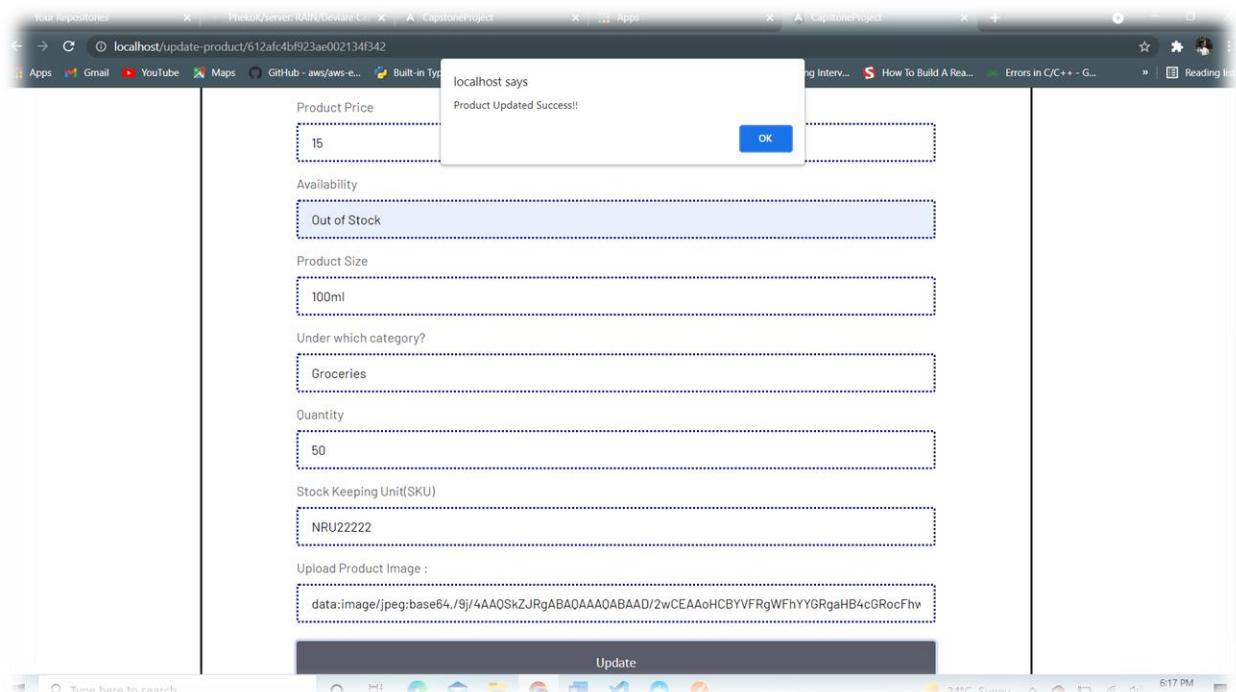
- product added will appear on the product overview page at the bottom of the page for the admin to edit or remove item.

Household										
									 	
7		Toilet Paper	120	In Stock	40	Twinsaver	48 rolls	Toiletries , Household	NRU12110	 
8		Body Spray	20	In Stock	50	Yardley	50ml	Toiletries, Household	NRU33333	 
9		Brown Bread	15	In Stock	90	Albany	300kg	Groceries	NRU12345	 

- Removed Item will call an alert prompt informing the admin that a product has been removed from Database.

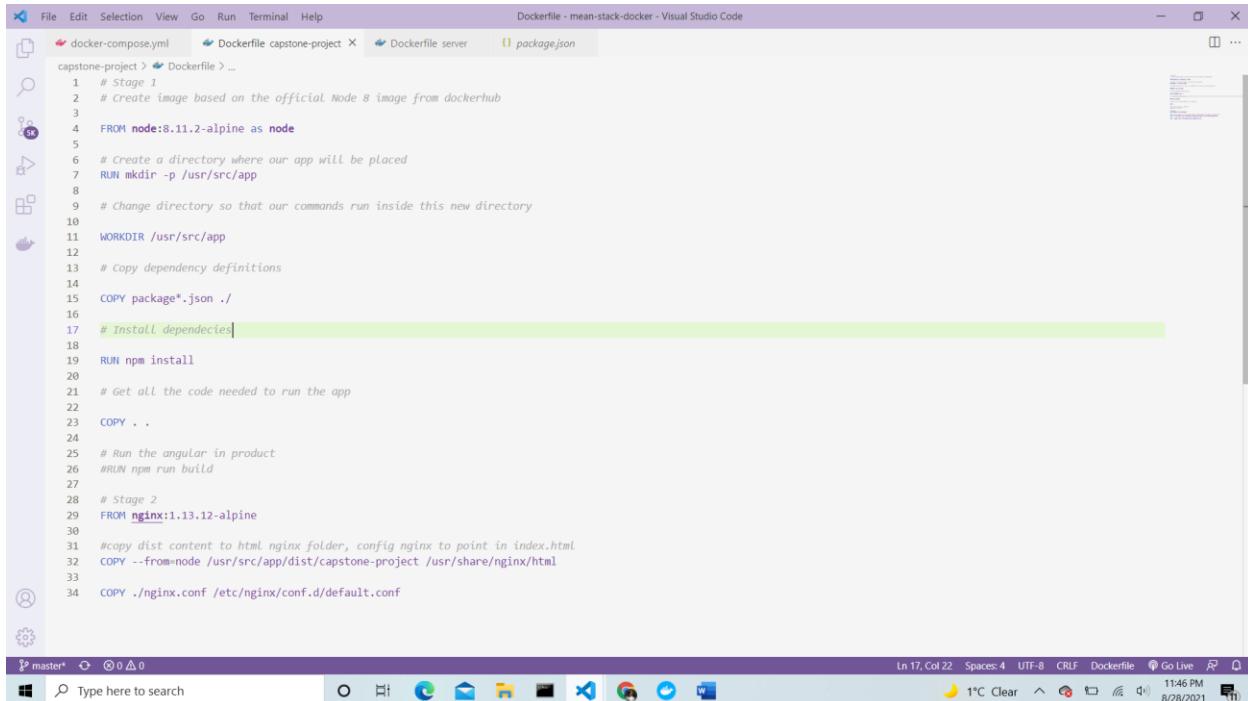


- Admin can edit an existing product.



### **3. Containerizing Frontend (Angular) & Backend(express) Application using Docker and Dockerhub.**

- Steps taken to containerize angular application
  1. Dockerized Angular capstone-project app deployed in nginx server.
  2. Dockerized the express server API– named ‘server’
  3. Use MongoDB container
  4. Docker compose
  5. Connecting the 3 Docker containers (angular, server and mongodb database)
  
- Created a mean-stack-docker directory and copied capstone client project and express server to the folder.
- For my angular application to run, I needed the image with NodeJS installed on it and add angular client project into the image and install all the dependencies.
- Ran the angular project in production mode and copied the folder ‘/dist’ content from the WORKDIR to the nginx html
- Now the project work in production mode deployed in nginx server (localhost:80)
- The above instructions are to be written in our Dockerfile from the client side.
  
- Dockerfile: mean-stack-docker/capstone/Dockerfile (frontend)

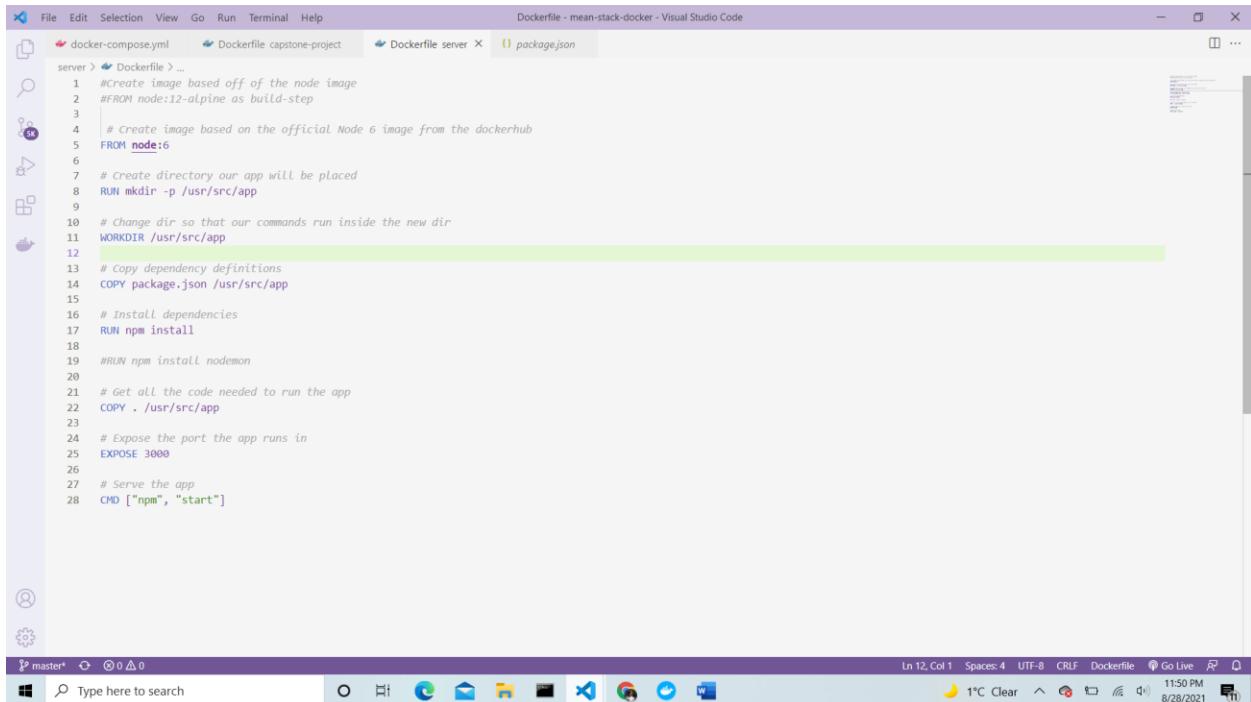


The screenshot shows a Visual Studio Code window with the title "Dockerfile - mean-stack-docker - Visual Studio Code". The code editor displays a Dockerfile with the following content:

```
FROM node:8.11.2-alpine as node
# Create a directory where our app will be placed
RUN mkdir -p /usr/src/app
# Change directory so that our commands run inside this new directory
WORKDIR /usr/src/app
# Copy dependency definitions
COPY package*.json .
# Install dependencies
RUN npm install
# Get all the code needed to run the app
COPY . .
# Run the angular in product
#RUN npm run build
# Stage 2
FROM nginx:1.13.12-alpine
#copy dist content to html nginx folder, config nginx to point in index.html
COPY --from=node /usr/src/app/dist/capstone-project /usr/share/nginx/html
COPY ./nginx.conf /etc/nginx/conf.d/default.conf
```

The Dockerfile defines two stages: "node" and "nginx". The "node" stage uses the official Node.js 8.11.2 Alpine image. It creates a directory for the application, changes the working directory to /usr/src/app, copies the package.json files, installs dependencies using npm, and copies the entire project directory. The "nginx" stage uses the official Nginx 1.13.12 Alpine image. It copies the built Angular application from the "node" stage's /usr/src/app/dist directory to the /usr/share/nginx/html directory and copies the Nginx configuration file.

- Created server directory under the root directory of mean-stack-docker.
- Inside the mean-stack-docker/server/Dockerfile (backend).



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files: docker-compose.yml, Dockerfile capstone-project, Dockerfile server, and package.json.
- Dockerfile:** Content (lines 1-28):

```
1 #Create image based off of the node image
2 #FROM node:12-alpine as build-step
3
4 # Create image based on the official Node 6 image from the dockerhub
5 FROM node:6
6
7 # Create directory our app will be placed
8 RUN mkdir -p /usr/src/app
9
10 # Change dir so that our commands run inside the new dir
11 WORKDIR /usr/src/app
12
13 # Copy dependency definitions
14 COPY package.json /usr/src/app
15
16 # Install dependencies
17 RUN npm install
18
19 #RUN npm install nodemon
20
21 # Get all the code needed to run the app
22 COPY . /usr/src/app
23
24 # Expose the port the app runs in
25 EXPOSE 3000
26
27 # Serve the app
28 CMD ["npm", "start"]
```

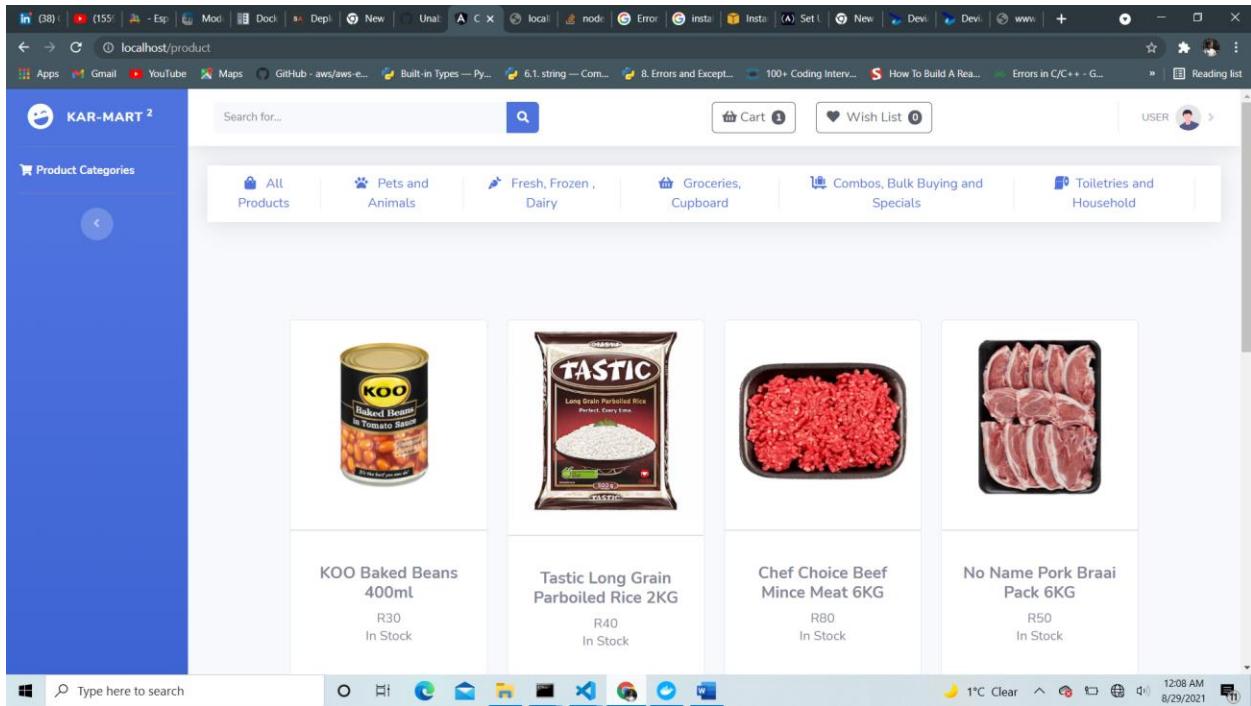
**Taskbar:** Shows master\*, Type here to search, and various pinned icons.

**Status Bar:** Shows Ln 12, Col 1, Spaces: 4, UTF-8, CR/LF, Dockerfile, Go Live, 11:50 PM, 1°C Clear, 8/28/2021.

- Ran
- Docker build -t server – to check test if server image is working under port 80.
- Docker run -d --name server -p 3000:3000 server
- Then ran localhost:3000 on browser to test
- MongoDB container
- Assuming I have mongodb image already, I ran docker run -d --name mongodb -p 27017:27017 mongo

## Docker Compose

- Created docker-compose.yml file in the mean-stack root directory
- Ran *docker-compose build --cache*
- Ran *docker-compose up*
- Ran localhost:80 in browser and display the following page below.



- Docker-compose.yml file

```

version: '3.7'
services:
  mongo-db:
    image: mongo
    ports:
      - '27017:27017'
  server:
    build: server
    environment:
      - MONGO_URI=mongodb://mongo-db/capstonedb
    ports:
      - '3000:3000'
    links:
      - mongo-db
  client:
    hostname: localhost
    build: capstone-project
    ports:
      - '80:80'

```

- Ran docker-compose up --build

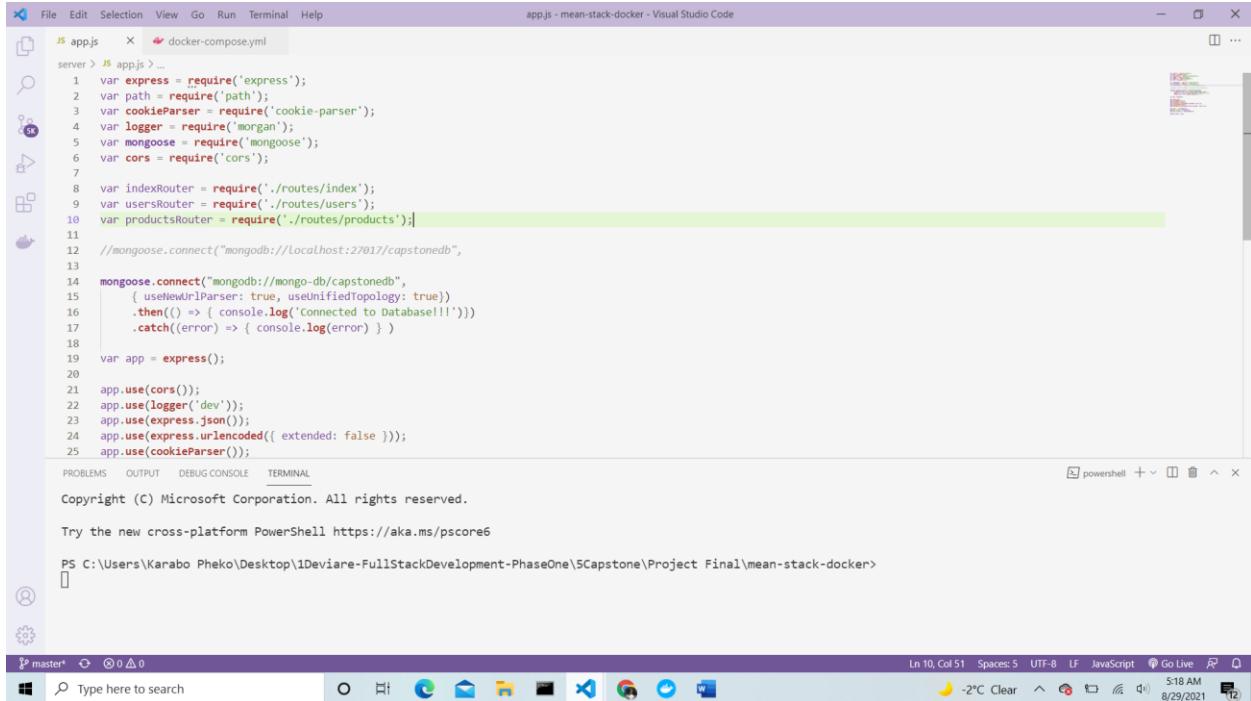
```

C:\Windows\System32\cmd.exe - docker-compose up --build
[...]

```

The screenshot shows a command-line window running on Windows. It displays the output of a `docker-compose up --build` command. The logs from multiple containers are shown, including mongo-db\_1, mongo-d\_1, client\_1, server\_1, and mongo-db\_1 again. The logs include timestamps, connection IDs, and various system messages related to MongoDB operations like indexing and checkpoints.

## - Minor change in app.js file in server



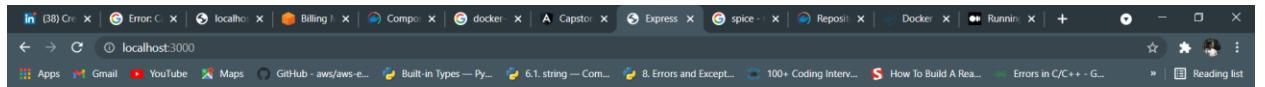
The screenshot shows the Visual Studio Code interface with the `app.js` file open. The file contains code for a Node.js application using Express.js and Mongoose. It includes routes for index, users, and products, and sets up middleware like cors, logger, and json parsing. A connection to a MongoDB database is established at the bottom.

```

app.js
1 var express = require('express');
2 var path = require('path');
3 var cookieParser = require('cookie-parser');
4 var logger = require('morgan');
5 var mongoose = require('mongoose');
6 var cors = require('cors');
7
8 var indexRouter = require('./routes/index');
9 var usersRouter = require('./routes/users');
10 var productsRouter = require('./routes/products');
11
12 //mongoose.connect("mongodb://localhost:27017/capstone");
13
14 mongoose.connect("mongodb://mongo-db/capstone",
15   { useNewUrlParser: true, useUnifiedTopology: true })
16   .then(() => { console.log("Connected to Database!!!") })
17   .catch((error) => { console.log(error) })
18
19 var app = express();
20
21 app.use(cors());
22 app.use(logger('dev'));
23 app.use(express.json());
24 app.use(express.urlencoded({ extended: false }));
25 app.use(cookieParser());

```

## - Test that server is working

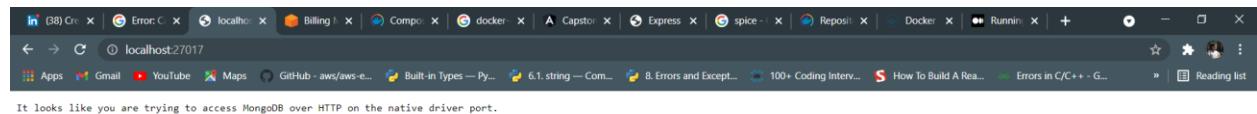


## Express

Welcome to Express



### - Test mongo image is running properly (localhost:27017)



It looks like you are trying to access MongoDB over HTTP on the native driver port.



### - Finally test your application (localhost:80)

KAR-MART

Stock Management

#	Image	Product Name	Price	Availability	Quantity	Brand	Size	Category	Stock Keeping Unit (SKU)	Action
1		Spice	15	In Stock	50	No Name	100ml	Groceries	NRU22222	

Copyright © Made By Karabo Pheko @ 2021

#### 4. AMAZON WEB SERVICES

##### A. Setting up EC2.

- Since application is containerized, we set an EC2 instance for the container to run on.
- Navigated to AWS and logged in with my credentials.

#### CREATE A NEW UBUNTU INSTANCE ON AWS EC2

1. Clicked 'Services' on the dropdown menu and selected 'EC2'. Clicked 'Launch Instance' button

Instances | EC2 Management Con x + us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:instanceState=running

Services ▾

New EC2 Experience Tell us what you think

EC2 Dashboard

Events

Tags

Limits

Instances

Instances New

Instance Types

Launch Templates

Spot Requests

Savings Plans

Reserved Instances New

Dedicated Hosts

Capacity Reservations

Images

AMIs

Elastic Block Store

Volumes

Snapshots

Feedback English (US) ▾

Type here to search

Search for services, features, marketplace products, and docs [Alt+S]

Instances Info

Filter instances

Instance state: running X Clear filters

Name Instance ID Instance state Instance type Status check Alarm status Availability Zone Public IPv4 DNS

No matching instances found

Select an instance above

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

12°C Mostly cloudy 9:00 PM 8/15/2021

2. Choose AMI you want to work with, in my case, I chose free tier Ubuntu Server 18.04 Lifetime Support.

Deploying Node.js Apps to AWS: x Launch instance wizard | EC2 Management Con x docker start container, Teoma.co x +

us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard

Services ▾

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Windows Microsoft Windows 2012 R2 Standard edition with 64-bit architecture. [English]

Free tier eligible

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

**SUSE Linux Enterprise Server 12 SP5 (HVM), SSD Volume Type** - ami-04aa88aeabb9fefdb83

SUSE Linux Enterprise Server 12 Service Pack 5 (HVM), EBS General Purpose (SSD) Volume Type. Public Cloud, Advanced Systems Management, Web and Scripting, and Legacy modules enabled.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

**Ubuntu Server 18.04 LTS (HVM), SSD Volume Type** - ami-0b9064170e32bde34 (64-bit x86) / ami-026141f3d5c6d2d0c (64-bit Arm)

Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

**Amazon Linux 2 with .Net Core, PowerShell, Mono, and MATE Desktop Environment** - ami-0eeb760abccf51451

.NET Core 5.0, Mono 6.12, PowerShell 7.1, and MATE DE pre-installed to run your .NET applications on Amazon Linux 2 with Long Term Support (LTS).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

**Ubuntu Server 16.04 LTS (HVM), SSD Volume Type** - ami-0d563aeddd4be7ff (64-bit x86) / ami-0bf25b43a4479334 (64-bit Arm)

Ubuntu Server 16.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

Cancel and Exit

64-bit (x86) Select 64-bit (x86)

Feedback English (US) ▾

Type here to search

Search for services, features, marketplace products, and docs [Alt+S]

Instances Info

Filter instances

Instance state: running X Clear filters

Name Instance ID Instance state Instance type Status check Alarm status Availability Zone Public IPv4 DNS

© 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

9°C Clear 10:05 AM 8/16/2021

### 3. Choose Instance type – t2-micro free tier.

**Step 2: Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
t2	t2.micro	1	1	EBS only	-	Low to Moderate	Yes
t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

**Currently selected:** t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, ~1 GiB memory, EBS only)

**Filter by:** All instance families ▾ Current generation ▾ Show/Hide Columns

**Next:** Configure Instance Details

### 4. Add tags( Optional), in my case I added a Key=Angular and Value=Angular Nodejs

**Step 5: Add Tags**

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all Instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	(128 characters maximum)	Value	(256 characters maximum)	Instances	Volumes	Network Interfaces
Angular		Angular Node JS		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Add another tag** (Up to 50 tags maximum)

**Next:** Configure Security Group

- Added a new security group, added rule for type HTTP and exposed port number 80 so we can accept connections from the web.

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:  Create a new security group  
 Select an existing security group

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Custom 0.0.0.0/0,::/0	e.g. SSH for Admin Desktop

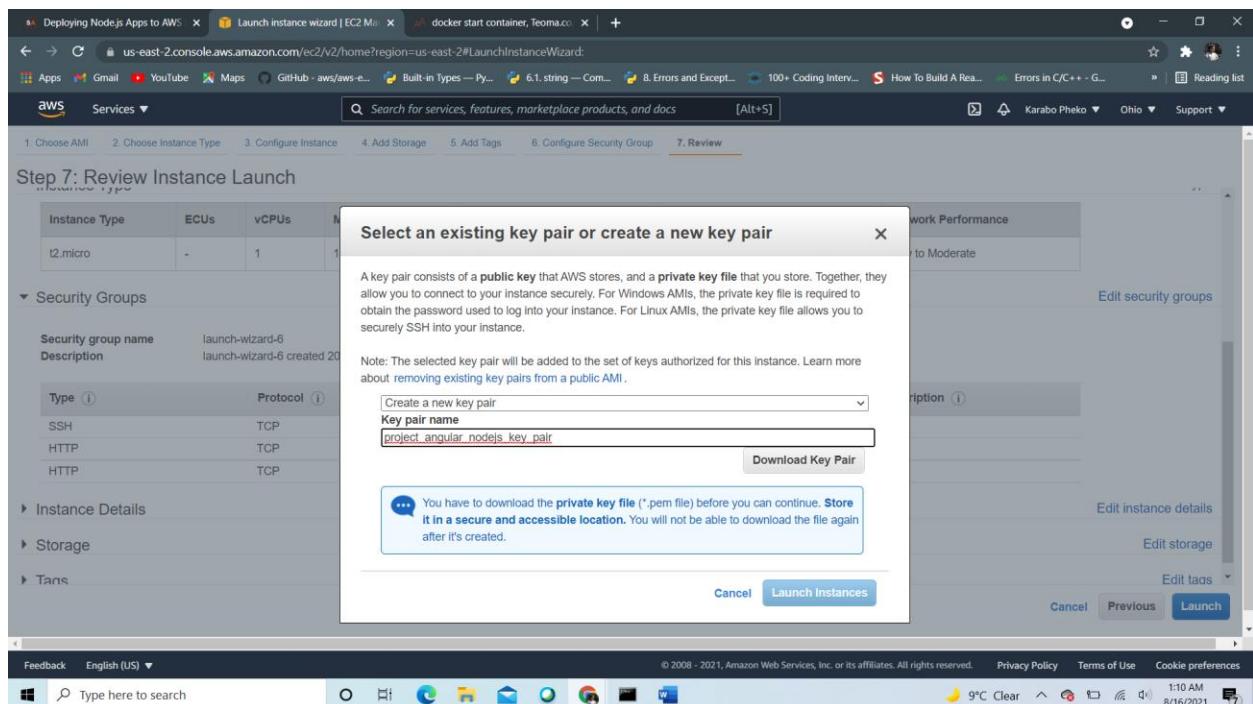
[Add Rule](#)

**Warning**  
 Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

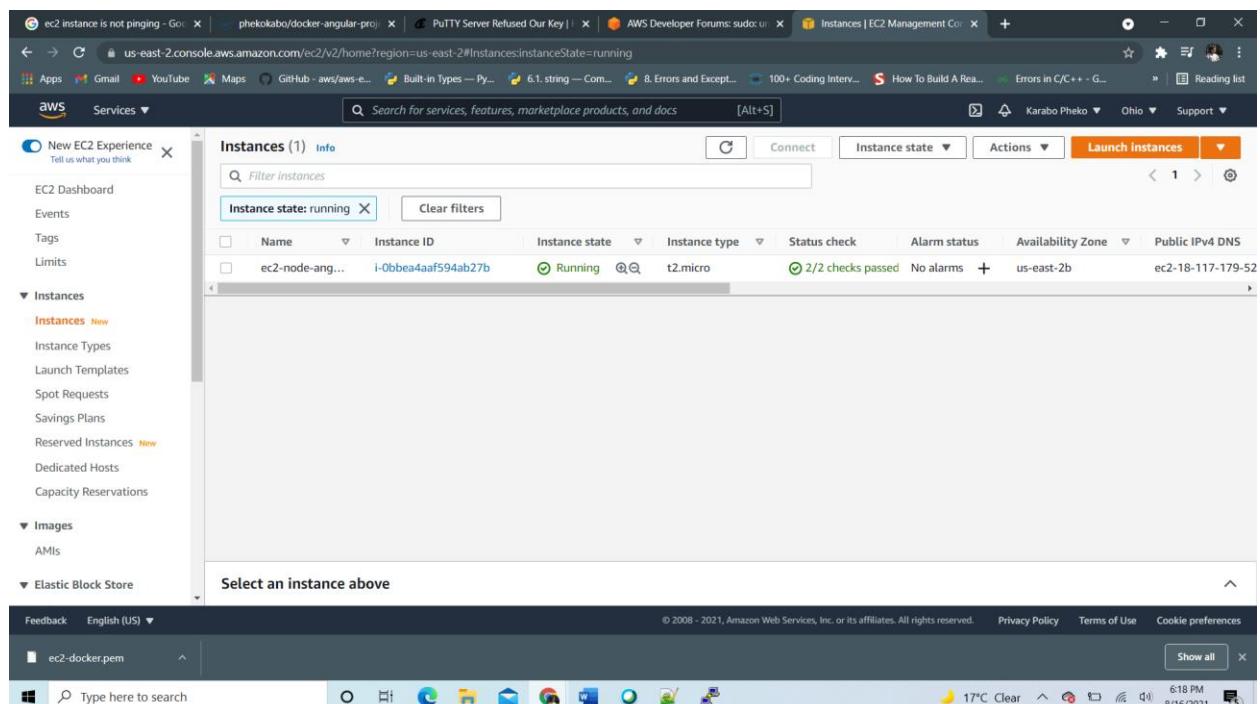
Feedback English (US) ▾ Type here to search © 2008 - 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences 9°C Clear 1:08 AM 8/16/2021

[Cancel](#) [Previous](#) [Review and Launch](#)

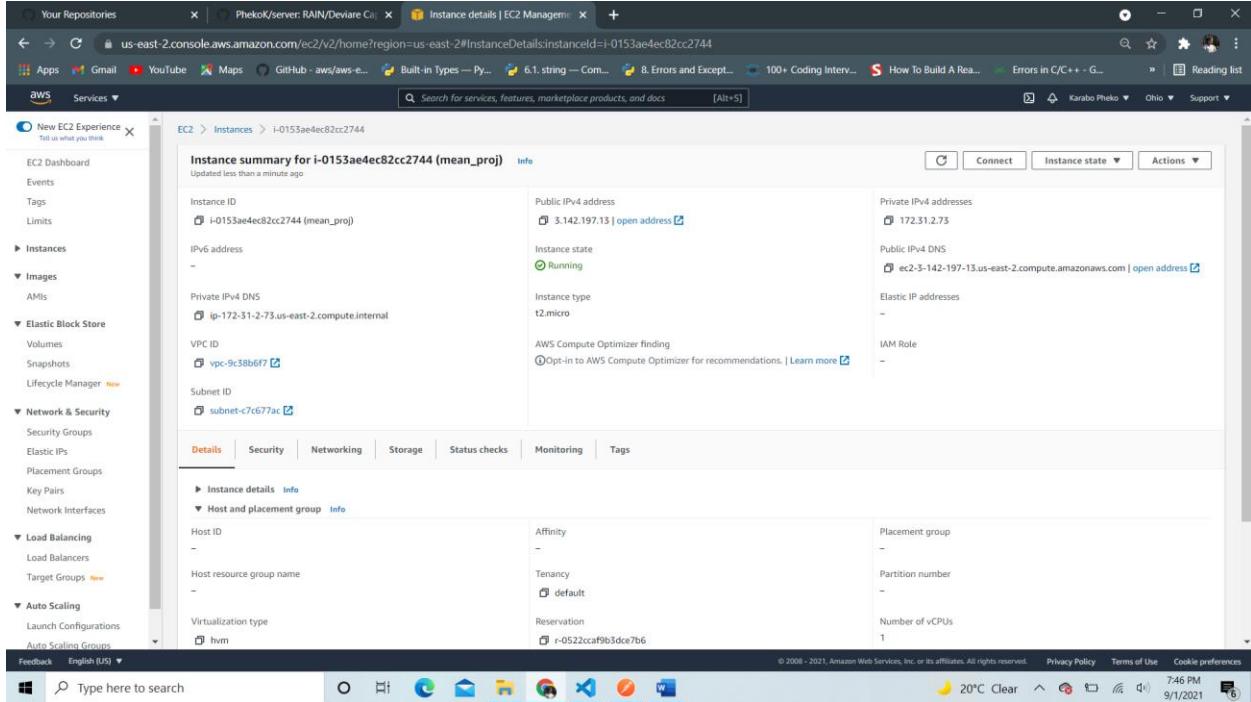
- After that, clicked Review and Launch button and created a new key\_pair to connect to my instance.



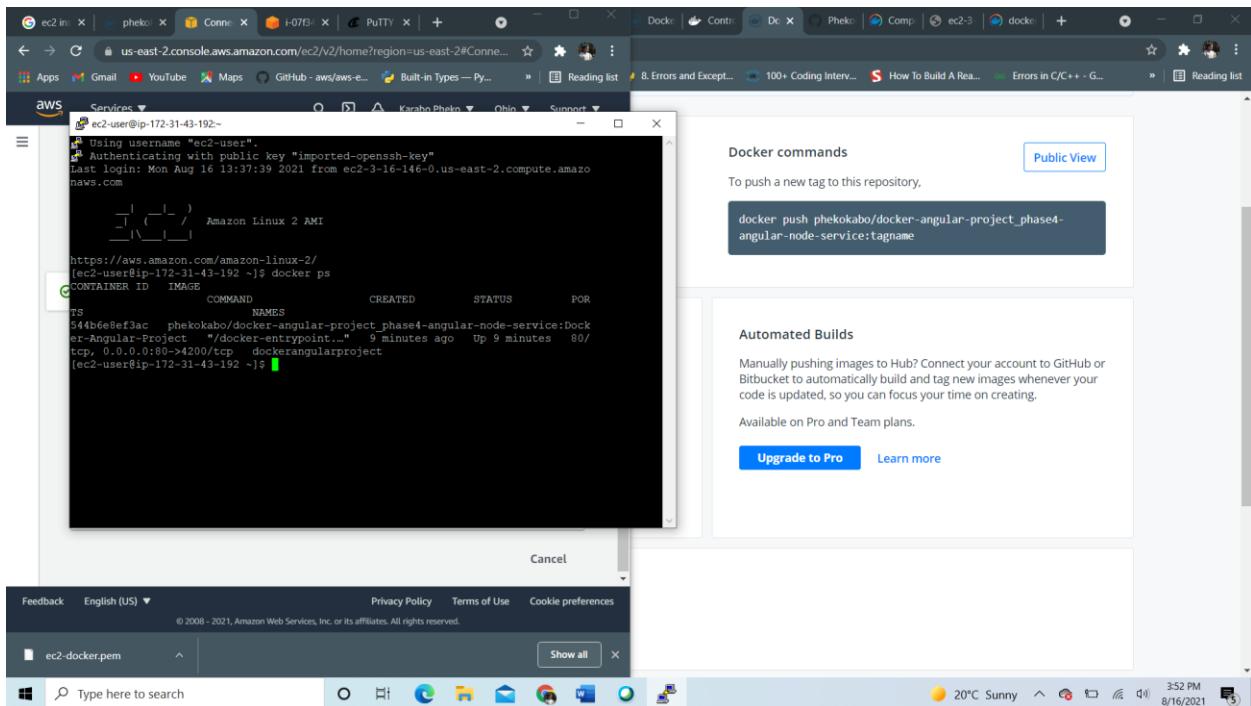
- Checked launched instances after reviewing



7. Launch EC2 instance using putty or virtual machine provided by AWS and SSH into the Amazon virtual machine by using the public IPv4 DNS detailed after I launched the instance.

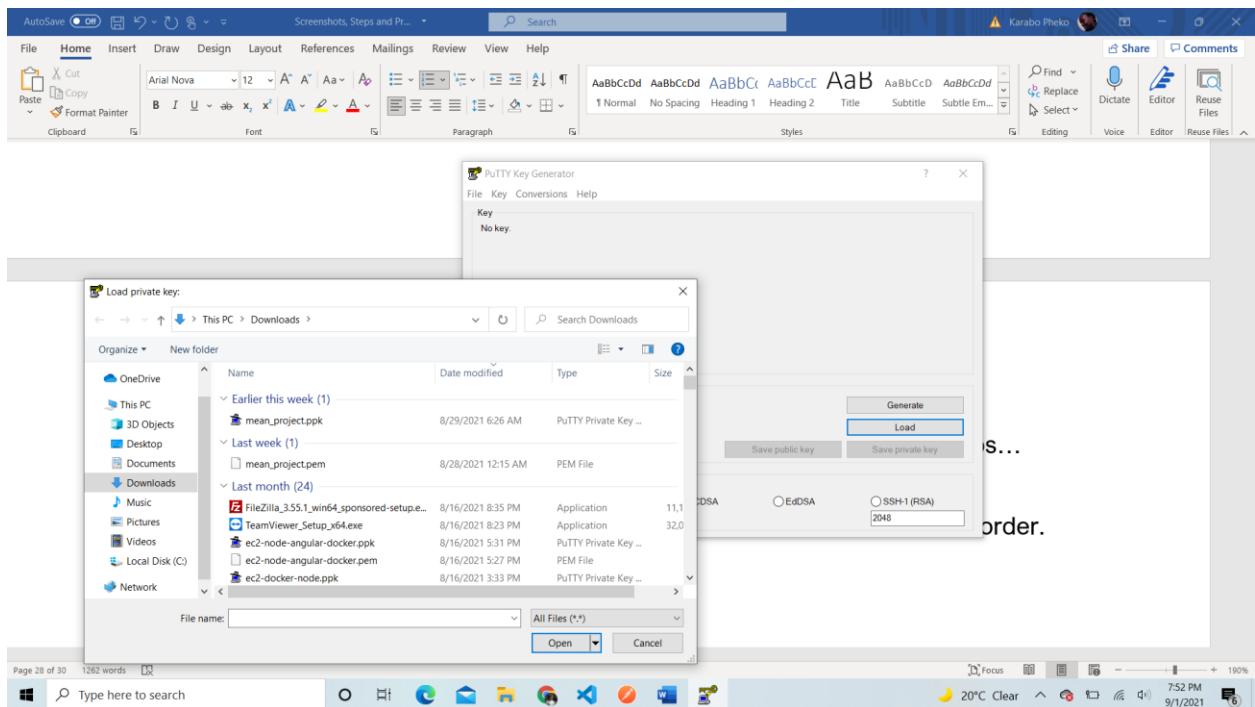


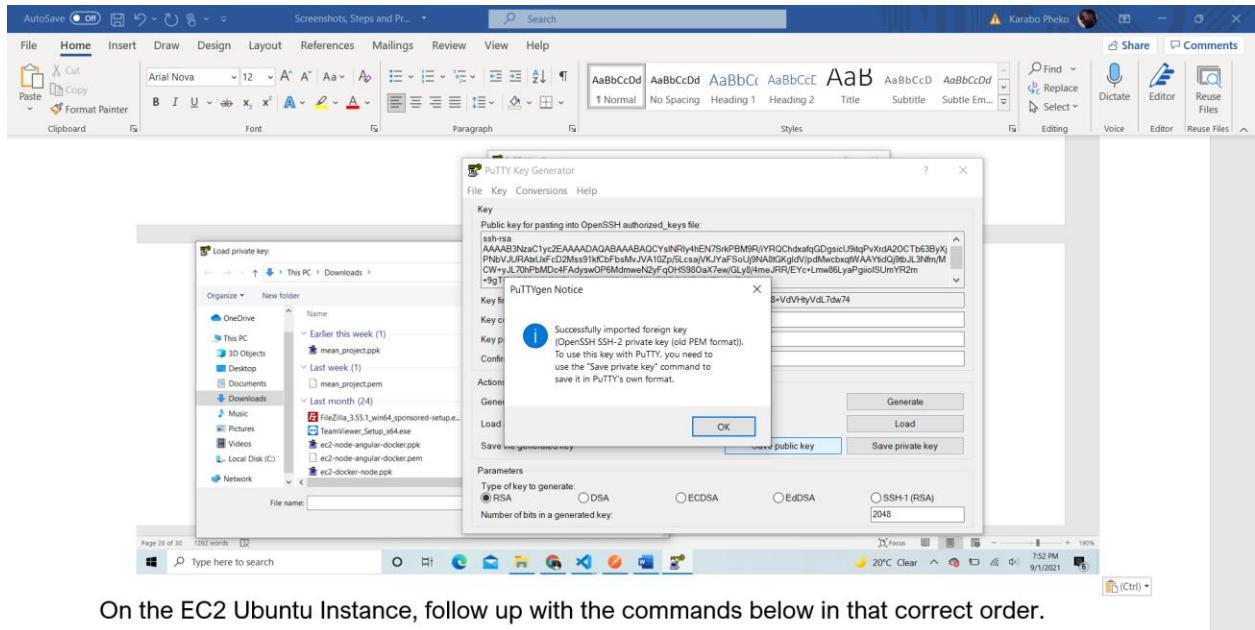
- Then the following instance is launched.



## CONNECT TO UBUNTU INSTANCE EC2 VIA SSH

- Alternatively you could use the one provided by putty by the following steps...
- 1. Open puTTygen and generate a private key, click load and add key with a .pem extension (Privacy Enhanced Mail) – used to sign me in in my SSH terminal to the virtual machine
- 2. Once selected, click ‘Save private key’ and your key is generated.
- 3. Once saved, use the key to connect to putty and open the virtual machine provided putty, by providing the IPv4 address as your address, the login username, i.e ubuntu, and the private key file generated by puTTygen for authentication with the .ppk extension and your instance is launched!





On the EC2 Ubuntu Instance, follow up with the commands below in that correct order.



- Once generated, head to putty to launch your instance and should look like something like this

```

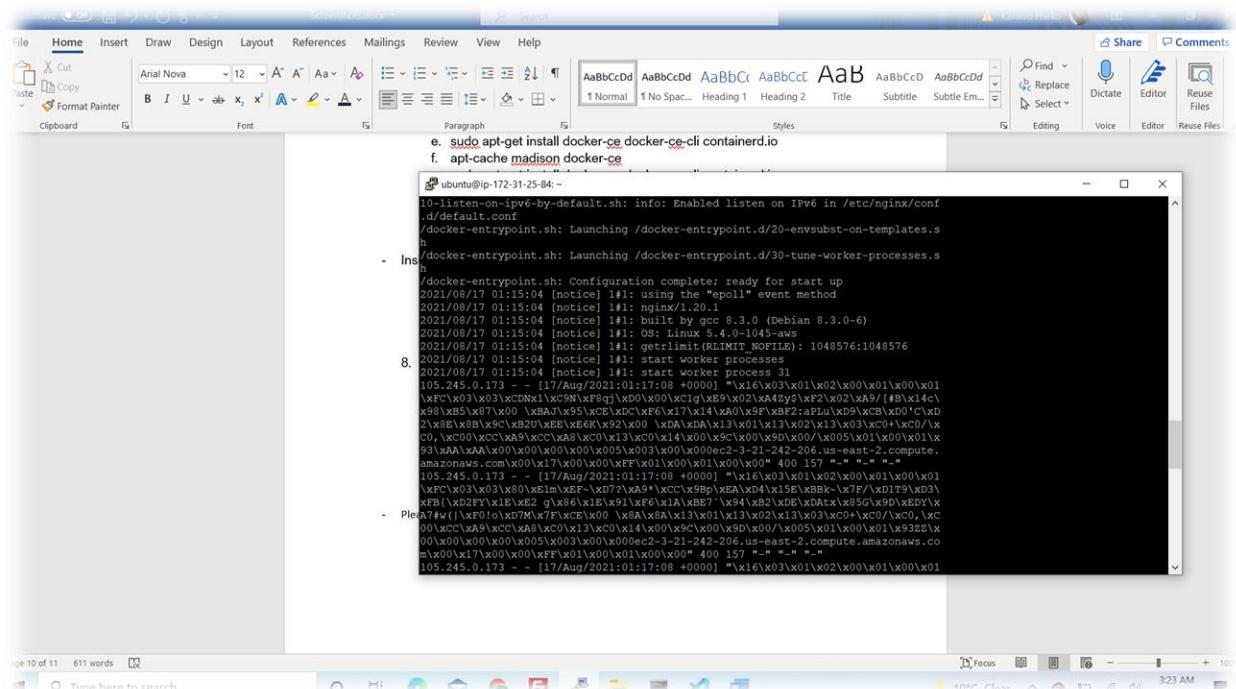
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Last login: Sun Aug 29 09:06:18 2021 from 41.13.22.226
ubuntu@ip-172-31-2-73:~$ 

```

# Setup Web Server with Node.js + MongoDB + NGINX

On the EC2 Ubuntu Instance, follow up with the commands below in that correct order.

- a. sudo apt-get update
  - b. sudo apt-get upgrade
  - c. sudo apt-get install \  
    apt-transport-https \  
    ca-certificates \  
    curl \  
    gnupg-agent \  
    software-properties-common
  - d. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
  - e. sudo apt-get install docker-ce docker-ce-cli containerd.io
  - f. apt-cache madison docker-ce
  - g. sudo apt-get install docker-ce docker-ce-cli containerd.io
  - h. sudo apt install docker.io
  - i. sudo apt install docker-compose



- git clone your backend and fronted application to your instance like so:

```
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1055-aws x86_64)

Documentation: https://help.ubuntu.com
Management: https://landscape.canonical.com
Support: https://ubuntu.com/advantage

System information as of Thu Sep 2 10:23:42 UTC 2021

System load: 0.01      Users logged in:          0
Usage of /: 67% of 7.69GB   IP address for eth0:    172.31.2.73
Memory usage: 26%        IP address for br-471122b33f49: 172.20.0.1
Swap usage: 0%           IP address for docker0: 172.17.0.1
Processes: 103

Ubuntu Pro delivers the most comprehensive open source security and
compliance features.

https://ubuntu.com/aws/pro

Canonical Livepatch is available for installation.
- Reduce system reboots and improve kernel security. Activate at:
  https://ubuntu.com/livepatch

packages can be updated.
of these updates is a security update.
see these additional updates run: apt list --upgradable

root@ubuntu:~# apt login: Sun Aug 29 09:06:18 2021 from 41.13.22.226
root@ubuntu:~# cd mean-stack-docker
root@ubuntu:~# ls
mean-stack-docker
root@ubuntu:~# cd mean-stack-docker
root@ubuntu:~# ls
capstone-project docker-compose.yml server
root@ubuntu:~# ls
root@ubuntu:~#
```

- cd to capstone-project (frontend) and install nginx server

```
root@ubuntu:~# cd capstone-project
root@ubuntu:~/capstone-project# apt update
root@ubuntu:~/capstone-project# apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libjpeg-turbo0:amd64
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  nginx
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 22.1 MB/s of archives.
After this operation, 2461 kB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-http-xslt-filter amd64 1.14.0-0ubuntu1.9 [12.7 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-mail amd64 1.14.0-0ubuntu1.9 [41.6 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-stream amd64 1.14.0-0ubuntu1.9 [63.5 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx-core amd64 1.14.0-0ubuntu1.9 [413 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx all 1.14.0-0ubuntu1.9 [3596 B]
Fetched 2461 kB in 0s (22.1 MB/s)
(Reading database ... 87264 files and directories currently installed.)
Preconfiguring packages ...
Selecting previously unselected package libjpeg-turbo0:amd64.
(Reading database ... 87264 files and directories currently installed.)
Preparing to unpack .../00-libjpeg-turbo0 1.5.2-0ubuntu5.18.04.4_amd64.deb ...
Packing libjpeg-turbo0:amd64 (1.5.2-0ubuntu5.18.04.4) ...
Selecting previously unselected package fonts-dejavu-core.
Preparing to unpack .../01-fonts-dejavu-core_2.37-1_all.deb ...
Packing fonts-dejavu-core_2.37-1_all.deb ...
Selecting previously unselected package fontconfig-config.
Preparing to unpack .../02-fontconfig-config 2.12.6-0ubuntu2_all.deb ...
Packing fontconfig-config (2.12.6-0ubuntu2) ...
Selecting previously unselected package libfontconfig1:amd64.
Preparing to unpack .../03-libfontconfig1 2.12.6-0ubuntu2_amd64.deb ...
Packing libfontconfig1:amd64 (2.12.6-0ubuntu2) ...
Selecting previously unselected package libjpeg8:amd64.
Preparing to unpack .../04-libjpeg8 8c-2ubuntu8_amd64.deb ...
Packing libjpeg8:amd64 (8c-2ubuntu8) ...
Selecting previously unselected package libjbig0:amd64.
Preparing to unpack .../05-libjbig0 2.1-3.libuidl1_amd64.deb ...
Packing libjbig0:amd64 (2.1-3.libuidl1_amd64) ...
Selecting previously unselected package libriff5:amd64.
Preparing to unpack .../06-libriff5 4.0.9-5ubuntu0.4_amd64.deb ...
Packing libriff5:amd64 (4.0.9-5ubuntu0.4) ...
Selecting previously unselected package libwebp6:amd64.
Preparing to unpack .../07-libwebp6_0.6.1-2ubuntu0.18.04.1_amd64.deb ...
Packing libwebp6:amd64 (0.6.1-2ubuntu0.18.04.1) ...
Selecting previously unselected package libxpm4:amd64.
Preparing to unpack .../08-libxpm4 1.33a.5.12-1_amd64.deb ...
Packing libxpm4:amd64 (1.33a.5.12-1) ...
Selecting previously unselected package libgd3:amd64.
Preparing to unpack .../09-libgd3 2.2.5-4ubuntu0.4_amd64.deb ...
Packing libgd3:amd64 (2.2.5-4ubuntu0.4) ...
Selecting previously unselected package nginx-common.
Preparing to unpack .../10-nginx-common_1.14.0-0ubuntu1.9_all.deb ...
Packing nginx-common_1.14.0-0ubuntu1.9_all.deb ...
Selecting previously unselected package libnginx-mod-http-geosip.
Preparing to unpack .../11-libnginx-mod-http-geosip 1.14.0-0ubuntu1.9_amd64.deb ...
Packing libnginx-mod-http-geosip (1.14.0-0ubuntu1.9) ...
Selecting previously unselected package libnginx-mod-http-image-filter.
Preparing to unpack .../12-libnginx-mod-http-image-filter 1.14.0-0ubuntu1.9_amd64.deb ...
Packing libnginx-mod-http-image-filter (1.14.0-0ubuntu1.9) ...
Selecting previously unselected package libnginx-mod-http-xslt-filter.
Preparing to unpack .../13-libnginx-mod-http-xslt-filter 1.14.0-0ubuntu1.9_amd64.deb ...
Packing libnginx-mod-http-xslt-filter (1.14.0-0ubuntu1.9) ...
```

- Run 'sudo apt-get install build-essential libssl-dev' to install node

```

ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ sudo apt-get install nginx -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbigr2 libjpeg-turbo0 libjpeg8 libnginx-mod-http-geoip libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp libxpm4 nginx nginx-common nginx-core
Suggested packages:
libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbigr2 libjpeg-turbo0 libjpeg8 libnginx-mod-http-geoip libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter
libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp libxpm4 nginx nginx-common nginx-core
0 upgraded, 19 newly installed, 0 to remove and 29 not upgraded.
Need to get 2461 kB of archives.
After this operation, 8210 kB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libjpeg-turbo0 amd64 1.5.2-0ubuntu5.18.04.4 [110 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 fonts-dejavu-core all 2.37-1 [104 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 fontconfig-config all 2.11.6-0ubuntu2 [55.8 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libfontconfig1 amd64 2.11.6-0ubuntu2 [137 kB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libfontconfig1 libfontconfig1 amd64 2.11.6-0ubuntu2 [214 kB]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libjbigr2 libjbigr2 amd64 2.1-3.1build1 [26.7 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtiff5 amd64 4.0.9-Subuntu0.4 [153 kB]
Get:8 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libwebp libwebp amd64 0.6.1-Subuntu0.4.1 [186 kB]
Get:9 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic/main amd64 libxpm4 libxpm4 amd64 1:3.5.12-1 [34.0 kB]
Get:10 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libgd3 libgd3 amd64 2.2.5-0ubuntu0.4 [119 kB]
Get:11 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx-common all 1.14.0-Subuntu1.9 [37.2 kB]
Get:12 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-http-geoip amd64 1.14.0-0ubuntu1.9 [11.0 kB]
Get:13 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-http-image-filter amd64 1.14.0-0ubuntu1.9 [14.3 kB]
Get:14 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-http-xslt-filter amd64 1.14.0-0ubuntu1.9 [12.7 kB]
Get:15 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-mail amd64 1.14.0-0ubuntu1.9 [41.6 kB]
Get:16 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libnginx-mod-stream amd64 1.14.0-0ubuntu1.9 [63.5 kB]
Get:17 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx-core amd64 1.14.0-Subuntu1.9 [413 kB]
Get:18 http://us-east-2.ec2.archive.ubuntu.com/ubuntu bionic-updates/main amd64 nginx 1.14.0-Subuntu1.9 [3396 kB]
Fetched 2461 kB in 0s (22.1 MB/s)
Preconfiguring packages
Selecting previously unselected package libjpeg-turbo0:amd64.
(Reading database ... 87264 files and directories currently installed.)
Preparing to unpack .../00-libjpeg-turbo0_1.5.2-0ubuntu5.18.04.4_amd64.deb ...

```

- Run ‘curl -o- <https://raw.githubusercontent.com/nvm-sh/v0.34.0/install.sh> | bash’ to install node version manager.
- Ran ‘~/.nvm/nvm.sh’ to activate nvm
- Ran ‘nvm install node’ to install node where we will use node version 16.8.0

```

ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ curl -o- https://raw.githubusercontent.com/nvm-sh/v0.34.0/install.sh | bash
Total % Received % Xferd Average Speed Time Time Current
Total Dload Upload Total Spent Left Speed
0 20 100 20 0 0 363 0 ---:---:---:---:---:--- 363
sh: line 1: 400: command not found
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ curl -o- https://raw.githubusercontent.com/nvm-sh/v0.34.0/install.sh | bash
Total % Received % Xferd Average Speed Time Time Current
Total Dload Upload Total Spent Left Speed
0 20 100 20 0 0 344 0 ---:---:---:---:---:--- 344
sh: line 1: 400: command not found
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ curl -o- https://raw.githubusercontent.com/nvm-sh/v0.34.0/install.sh | bash
Total % Received % Xferd Average Speed Time Time Current
Total Dload Upload Total Spent Left Speed
0 13226 100 13226 0 0 230k 0 ---:---:---:---:---:--- 230k
-- Downloading nvm from git to '/home/ubuntu/.nvm'
Cloning into '/home/ubuntu/.nvm'...
note: Enumerating objects: 278, done.
note: Counting objects: 100% (278/278), done.
note: Compressing objects: 100% (245/245), done.
note: Total 278 (delta 31), reused 101 (delta 20), pack-reused 0
-- Receiving objects: 100% (278/278), 142.25 KiB | 5.69 MiB/s.
-- Resolving deltas: 100% (31/31), done.
-- Compressing and cleaning up git repository
-- Appending NVM source string to '/home/ubuntu/.bashrc'
-- Appending hash completion source string to '/home/ubuntu/.bashrc'
-- Close and reopen your terminal to start using nvm or run the following to use it now:
$ export NVM_DIR=$HOME/.nvm"
$ $NVM_DIR/nvm.sh" # This loads nvm
$ $NVM_DIR/bash_completion" ] && . "$NVM_DIR/bash_completion" # This loads nvm bash_completion
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ . ~/.nvm/nvm.sh
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ nvm install node
Unloading https://nodejs.org/dist/v16.8.0/node-v16.8.0-linux-x64.tar.xz...
Unpacking https://nodejs.org/dist/v16.8.0/node-v16.8.0-linux-x64.tar.xz... 100%
Checksums matched!
using node v16.8.0 (npm v7.21.0)
warning default alias: default -> node (> v16.8.0)
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ 

```

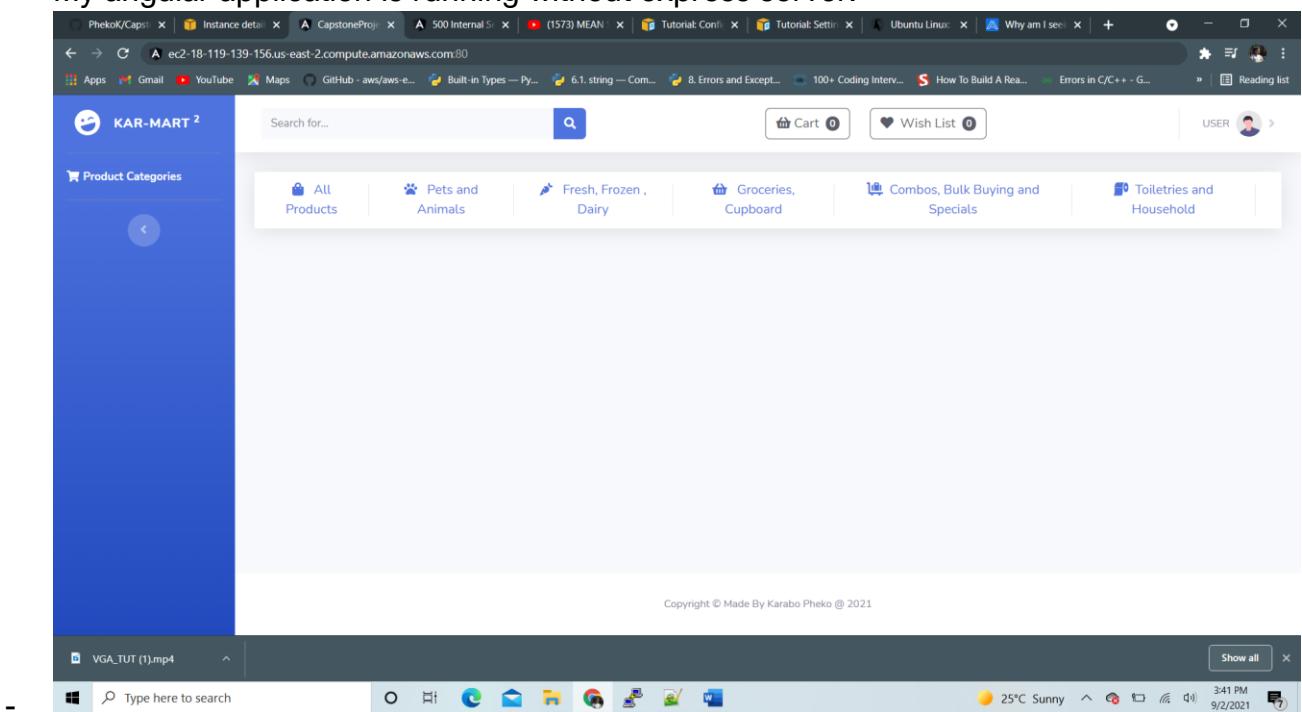
- Created /etc/nginx/sites-available/default file and modified it like so:

```
ADMEmd karma.conf.js package.json tsconfig.json
rular.json nginx.conf src tsconfig.spec.json
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ sudo nano /etc/nginx/sites-ava
ble/default
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ sudo nano /etc/nginx/sites-ava
ble/default
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ cat etc/nginx/sites-avail
le/default
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ cat /etc/nginx/sites-avail
le/default
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ curl http://ec2-18-119-139-156.us-east-2.compute.amazonaws.com;
index index.html;
root /home/mean-stack-docker/capstone-project;
location / {
    expires -1;
    add_header Cache-Control 'no-store, no-cache, must-revalidate, proxy-revali
te, max-age=0';
    try_files try_files $uri $uri/ /index.html;
}
location /product/ {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
    proxy_redirect off;
}

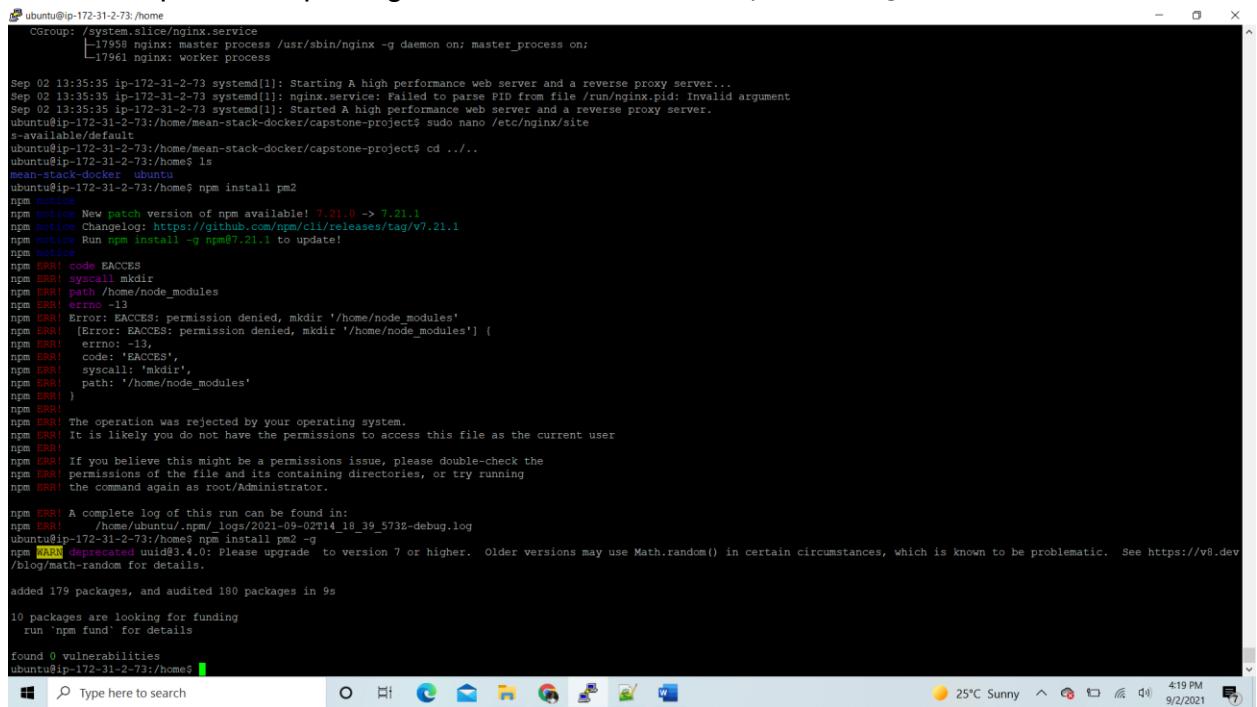
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ sudo systemctl status nginx
nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2021-09-02 13:35:35 UTC; 3min ago
       Docs: man:nginx(8)
  Process: 17483 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run
  Process: 17954 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, stat
  Process: 17941 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exi
  Main PID: 17958 (nginx)
     Tasks: 2 (limit: 1140)
    CGroup: /system.slice/nginx.service
           └─17958 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
              ├─17961 nginx: worker process

○ 02 13:35:35 ip-172-31-2-73 systemd[1]: Starting A high performance web server and a revers
○ 02 13:35:35 ip-172-31-2-73 systemd[1]: nginx.service: Failed to parse PID from file /run/n
○ 02 13:35:35 ip-172-31-2-73 systemd[1]: Started A high performance web server and a reverse
ntrip[172-31-2-73:/home/mean-stack-docker/capstone-project$ █
```

- Checked if nginx is running, and opened browser with the link:
  - My angular application is running without express server.



- Installed ‘npm install pm2 -g’ to make sure our node script is running



```

ubuntu@ip-172-31-2-73:/home
  CGroup: /system.slice/nginx.service
    ├─17958 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
    └─17961 nginx: worker process

Sep 02 13:35:35 ip-172-31-2-73 systemd[1]: Starting A high performance web server and a reverse proxy server...
Sep 02 13:35:35 ip-172-31-2-73 systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
Sep 02 13:35:35 ip-172-31-2-73 systemd[1]: Started A high performance web server and a reverse proxy server.
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ sudo nano /etc/nginx/sites-available/default
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/capstone-project$ cd ...
ubuntu@ip-172-31-2-73:/home$ ls
mean-stack-docker ubuntu
ubuntu@ip-172-31-2-73:/home$ npm install pm2
npm notice
npm notice New patch version of npm available! 7.21.0 -> 7.21.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.21.1
npm notice Run npm install -g npm@7.21.1 to update!
npm notice
npm ERR! code EACCES
npm ERR! syscall mkdir
npm ERR! path /home/node_modules
npm ERR! errno -13
npm ERR! Error: EACCES: permission denied, mkdir '/home/node_modules'
npm ERR! [Error: EACCES: permission denied, mkdir '/home/node_modules'] {
npm ERR!   errno: -13,
npm ERR!   code: 'EACCES',
npm ERR!   syscall: 'mkdir',
npm ERR!   path: '/home/node_modules'
npm ERR! }
npm ERR!
npm ERR! The operation was rejected by your operating system.
npm ERR! It is likely you do not have the permissions to access this file as the current user.
npm ERR! If you believe this might be a permissions issue, please double-check the
npm ERR! permissions of the file and its containing directories, or try running
npm ERR! the command again as root/Administrator.

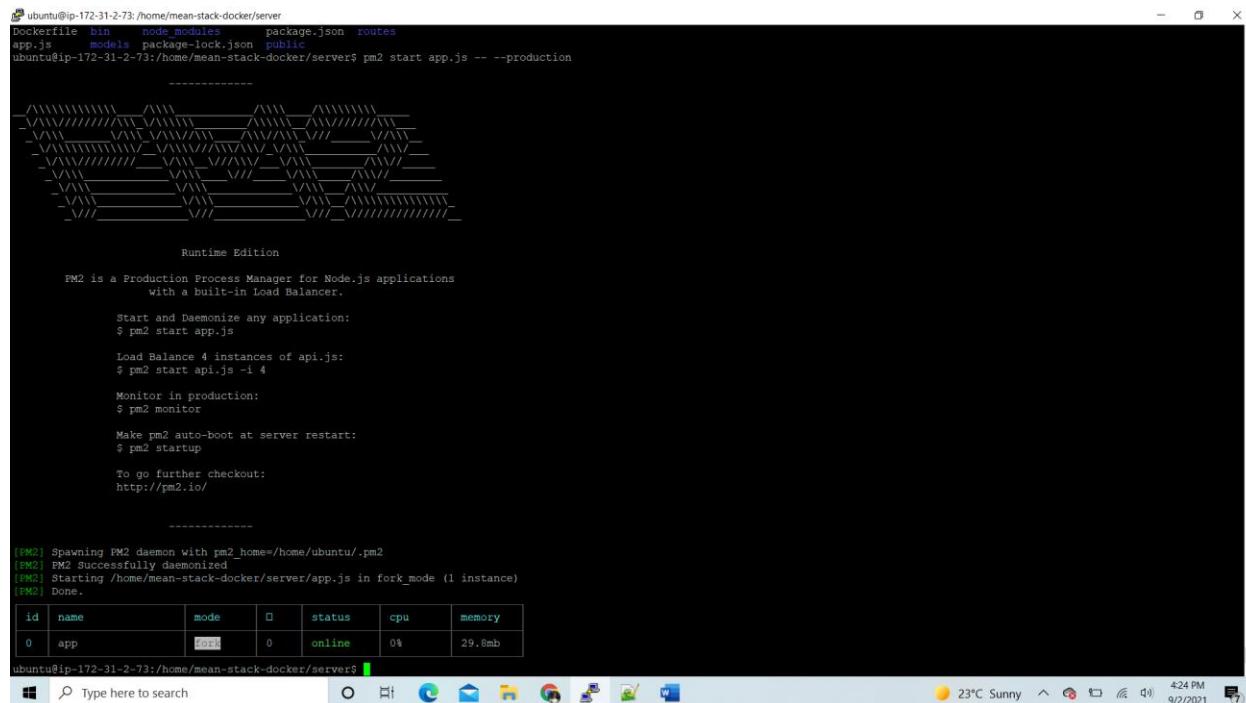
npm ERR! A complete log of this run can be found in:
npm ERR! /home/ubuntu/.npm/_logs/2021-09-02T14_18_39_573Z-debug.log
ubuntu@ip-172-31-2-73:/home$ npm install pm2 -g
npm WARN deprecated uidl@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstances, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 179 packages, and audited 180 packages in 9s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ubuntu@ip-172-31-2-73:/home$ 
```

- Run pm2 in production



```

ubuntu@ip-172-31-2-73:/home/mean-stack-docker/server
  Dockerfile bin node_modules package.json routes
app.js models package-lock.json public
ubuntu@ip-172-31-2-73:/home/mean-stack-docker/server$ pm2 start app.js -- --production
-----
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
- \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
- \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
- \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
- \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
- \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ 
-----  

Runtime Edition  

PM2 is a Production Process Manager for Node.js applications  

with a built-in Load Balancer.  

Start and Daemonize any application:  

$ pm2 start app.js  

Load Balance 4 instances of api.js:  

$ pm2 start api.js -i 4  

Monitor in production:  

$ pm2 monitor  

Make pm2 auto-boot at server restart:  

$ pm2 startup  

To go further checkout:  

http://pm2.io/  

-----  

[PM2] Spawning PM2 daemon with pm2_home=/home/ubuntu/.pm2  

[PM2] PM2 Successfully daemonized  

[PM2] Starting /home/mean-stack-docker/server/app.js in fork_mode (1 instance)  

[PM2] Done.  

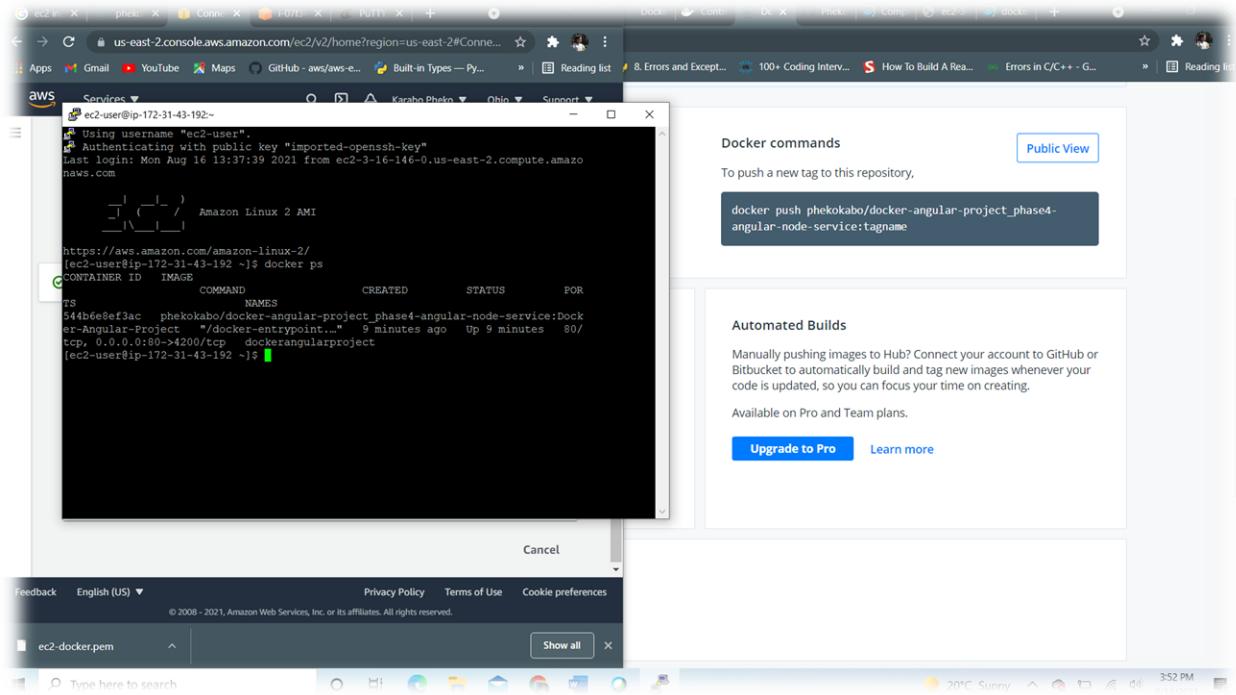

| id | name | mode | status | cpu    | memory |
|----|------|------|--------|--------|--------|
| 0  | app  | fork | 0      | online | 0%     |


ubuntu@ip-172-31-2-73:/home/mean-stack-docker/server$ 
```

- Run ‘sudo systemctl status nginx’ to check whether nginx server is running

- Run 'npm start' in vs code to start express server and you have the website running like the pictures below.

8. Once connected to ec2 instance and everything up and running, you should be able to access the website using the public ipv4 dns provided below:

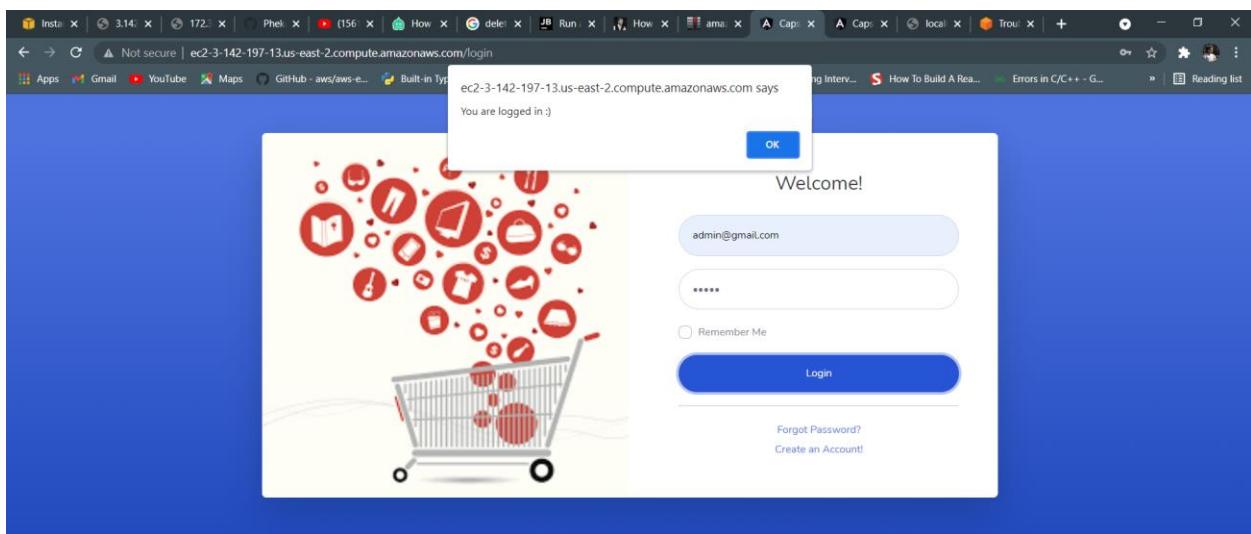
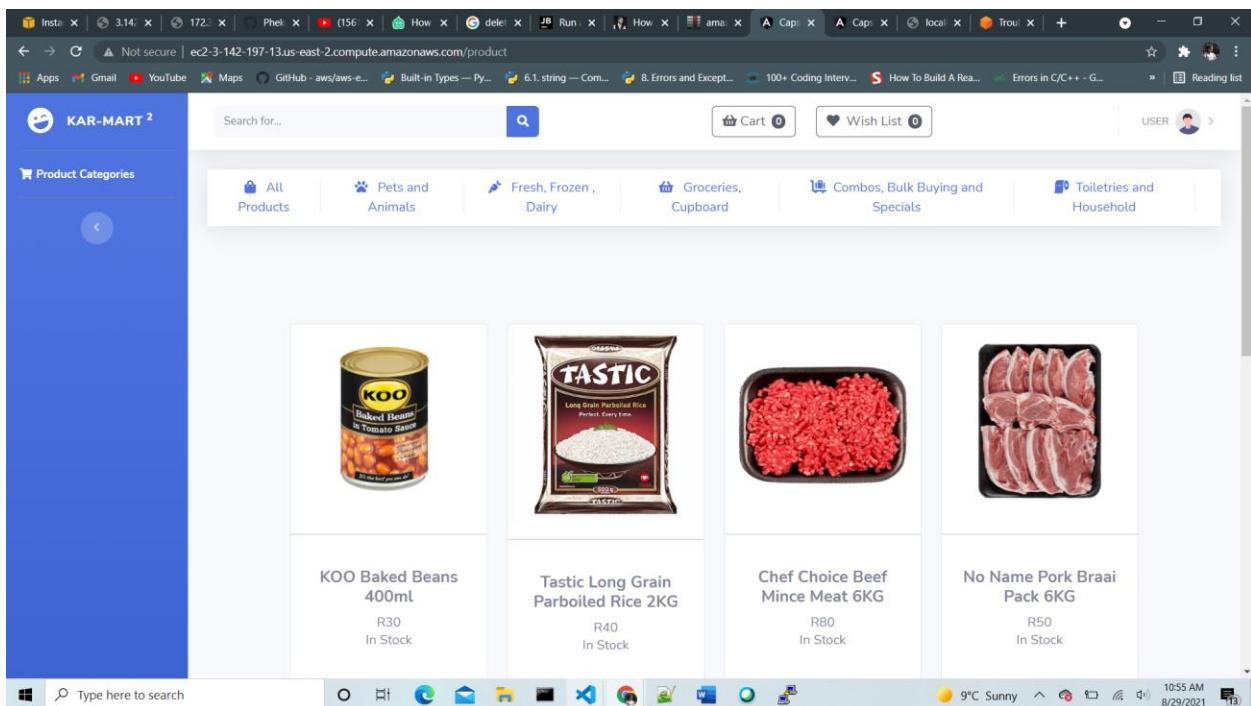


### Public IPv4 DNS:

- Use the link below to log into the application

<http://ec2-18-119-139-156.us-east-2.compute.amazonaws.com:80>, this lands you to the product page

- The website is now running under EC2 instance



Not secure | ec2-3-142-197-13.us-east-2.compute.amazonaws.com/invoice

ec2-3-142-197-13.us-east-2.compute.amazonaws.com says  
Thank you for buying. Your order is out for delivery

**Billing Details**

**David Peere:**  
1111 Army Navy Drive  
Arlington  
VA  
22 203

**Payment**

**Card Name:** Visa  
**Card Number:** \*\*\*\* 332  
**Exp Date:** 09/2020

**Gift:** No  
**Express Delivery:** Yes  
**Insurance:** No  
**Coupon:** No

**Shipping Address**

**David Peere:**  
1111 Army Navy Drive  
Arlington  
VA  
22 203

**Order summary**

Item Name	Item Price	Item Quantity	Total
Long Grain Parboiled Rice	R40	1	R40
Beef Mince Meat	R80	1	R80
	<b>Subtotal</b>		R120
		<b>Total</b>	R 120



[Shop more items](#)

