PUSL 3120

Full Stack Development Module

Referral Coursework

Palle Prasad

10749129

Full Stack Development Module Report

Video Link: https://youtu.be/2AOvP5eBW7w

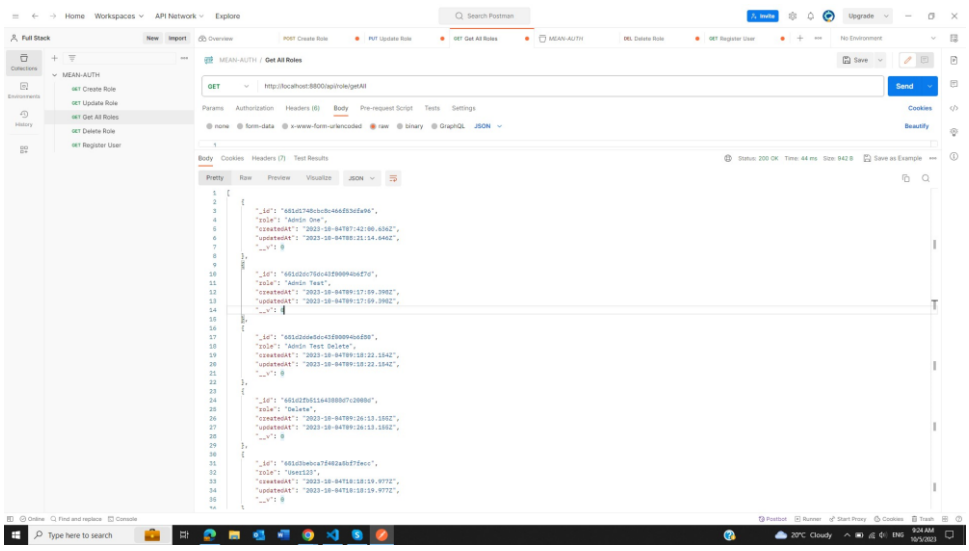
User Authentications and Authorization

This part of the project has been implemented with the aim of developing user authentications for the application.

Two different schemas have been created here.

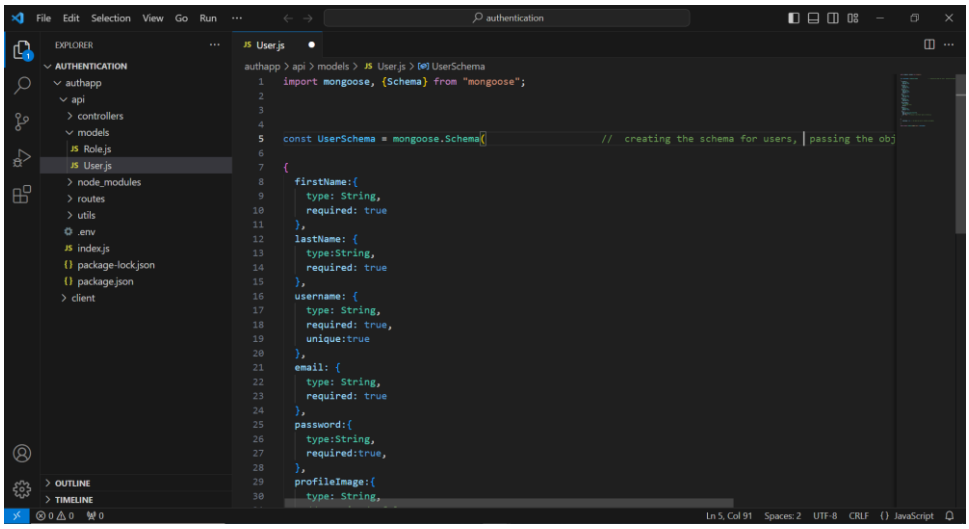
1. Users
2. Role

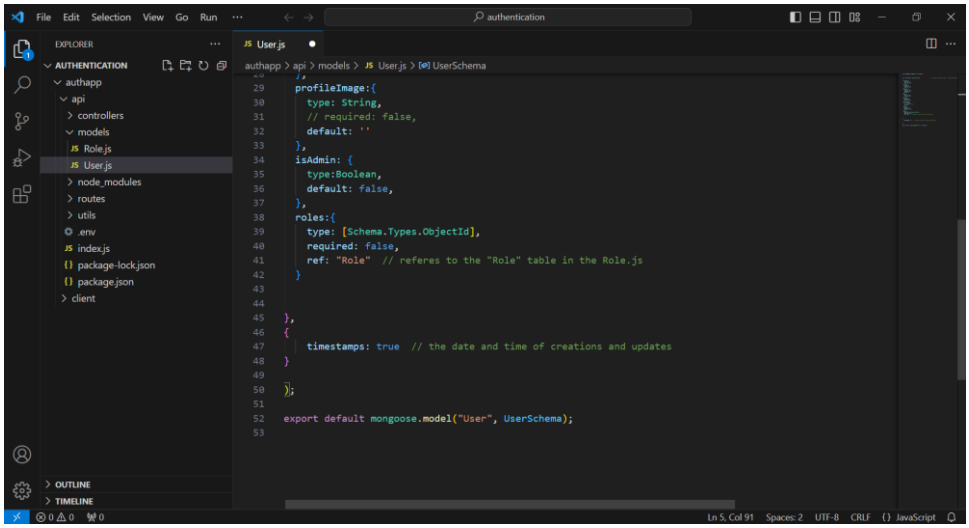
Role schema will be used to store all the roles with an ID.



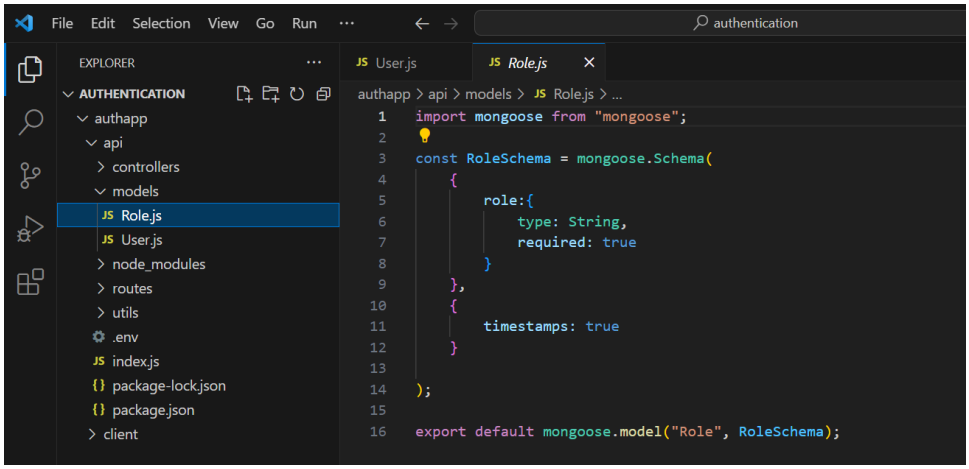
Some examples of roles can be described as Admin, and User.

User.js





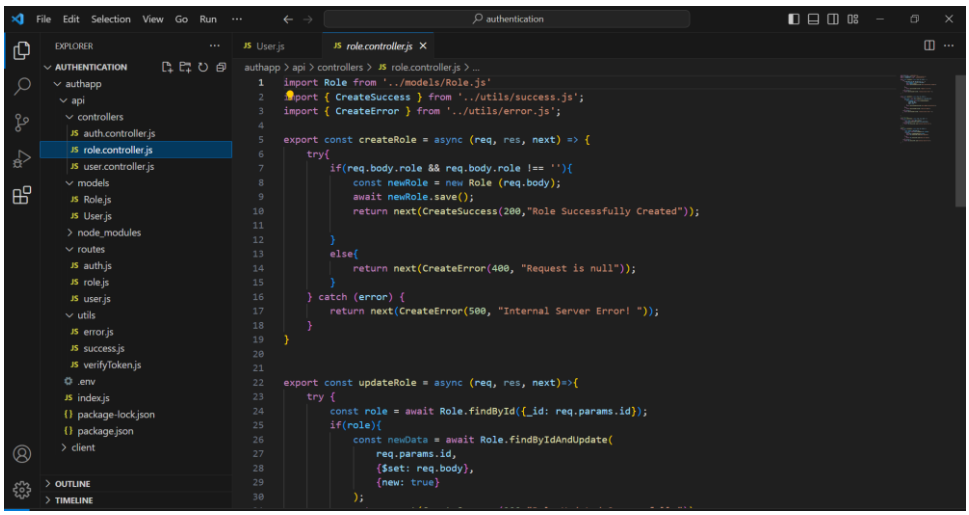
Role.js file



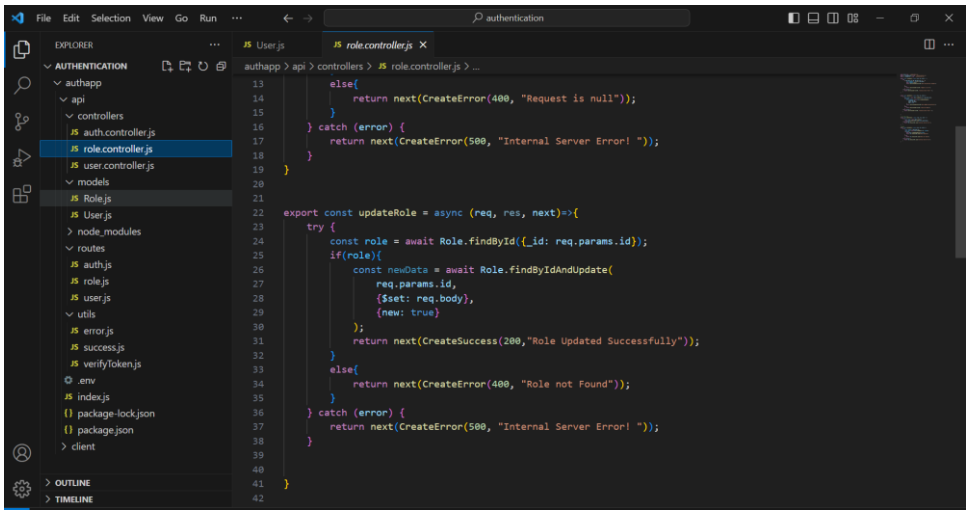
Creating APIs for Creating, Reading, Updating, Deleting roles

* role.controller.js

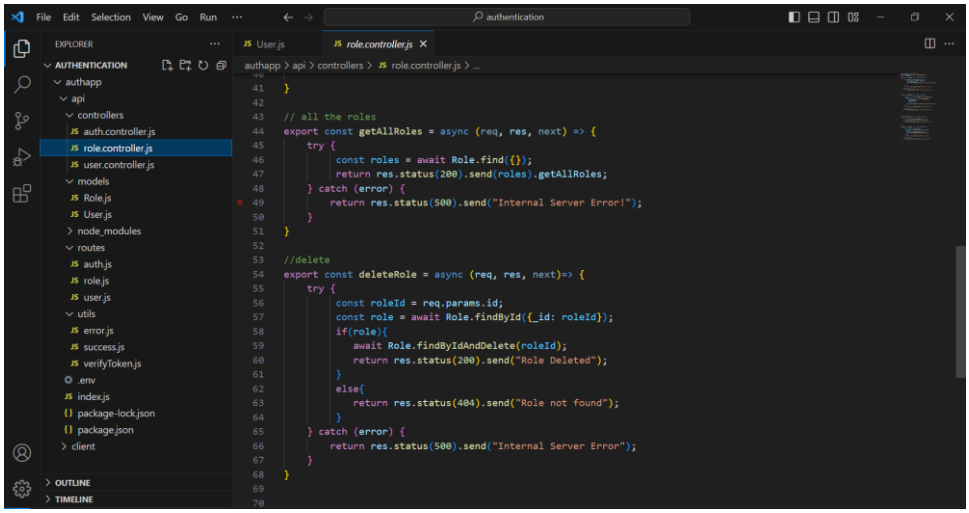
Create Role



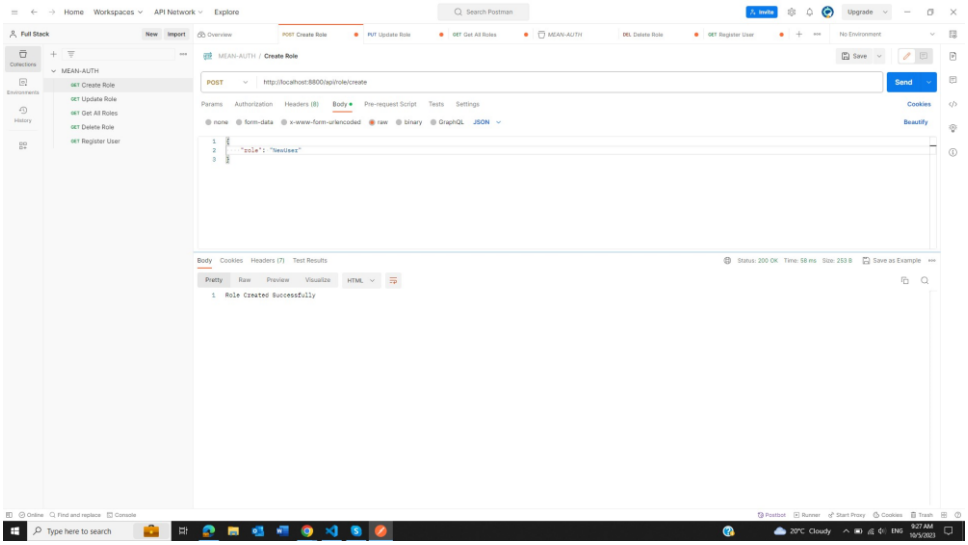
Update Role

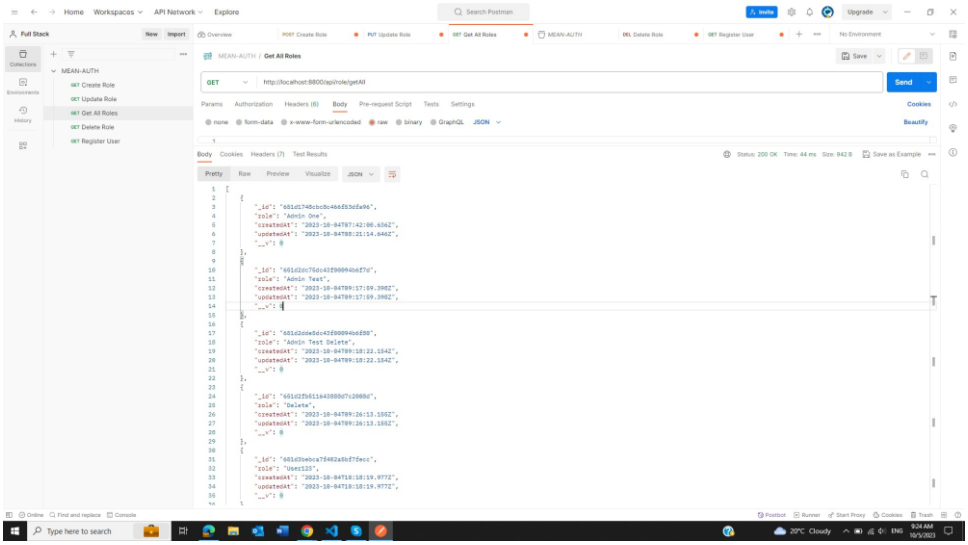


Get all Roles & Delete Role



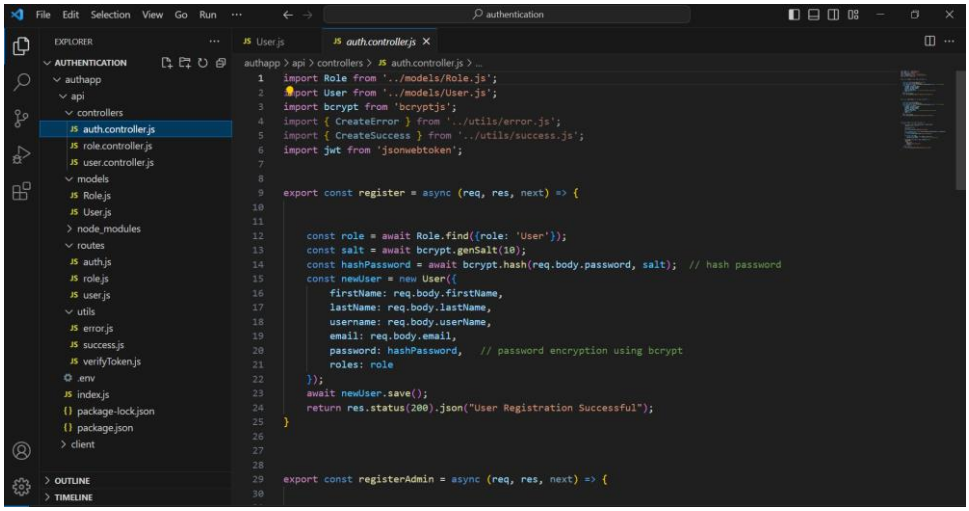
Testing APIs with the use of postman

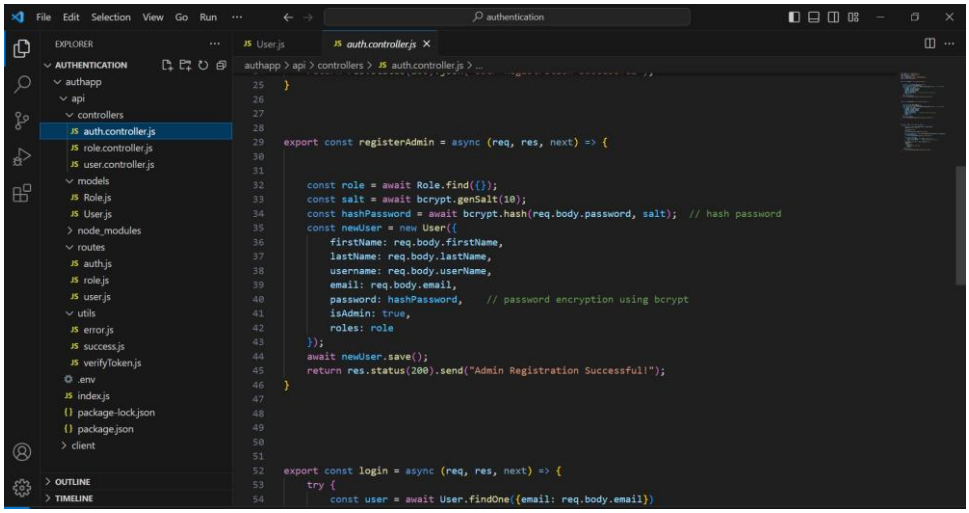


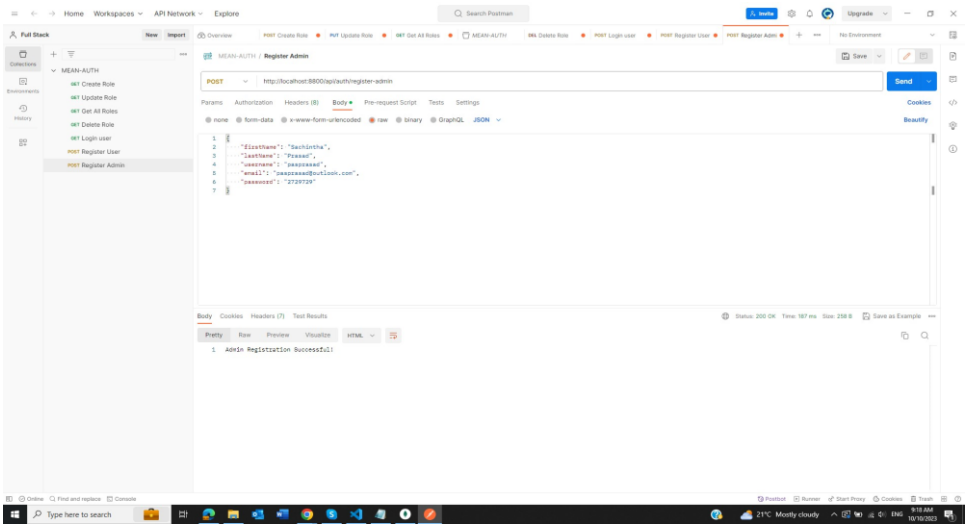


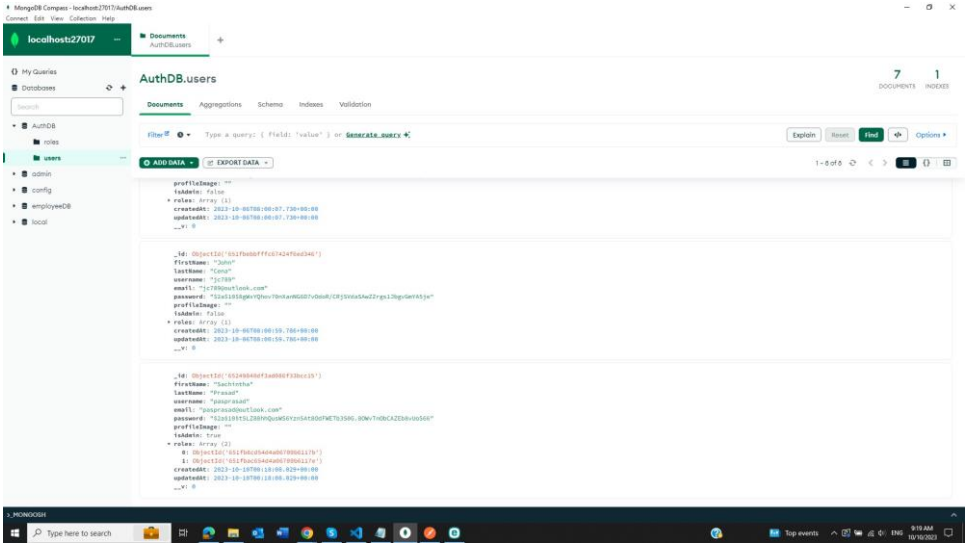
Register, Register as an Admin and Login Functionalities

Theses functionalities have been implemented inside of the auth.controller.js file which is inside of the controllers folder.

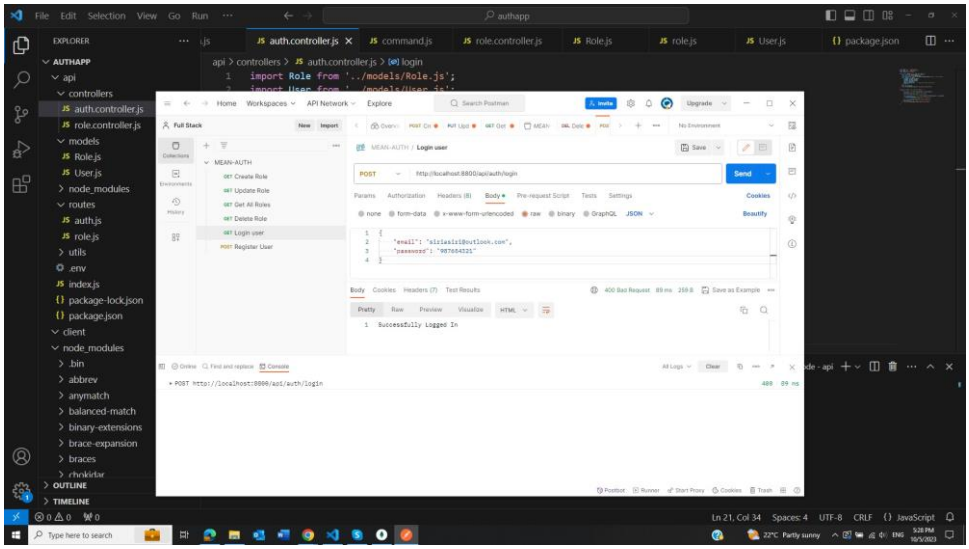


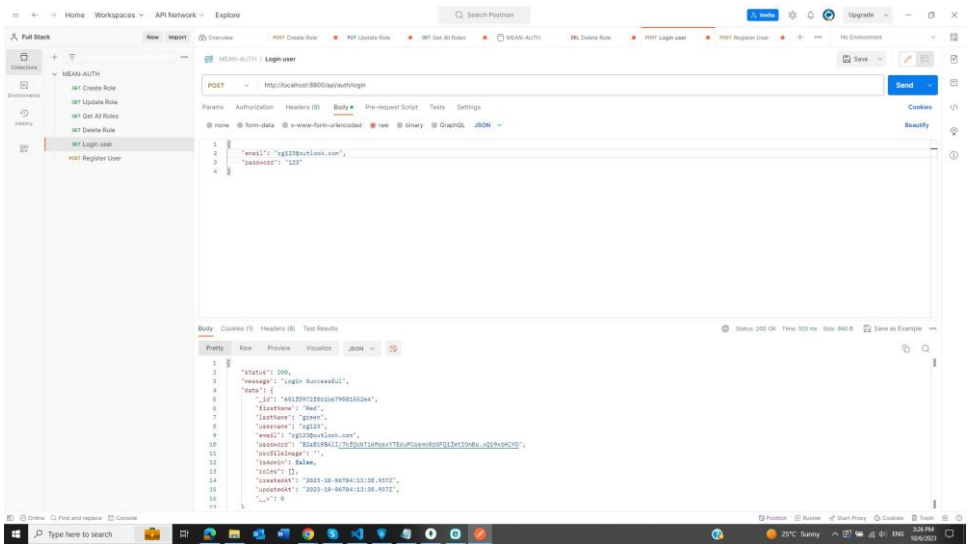



Home
Workspaces
API Network
Explore
Full Stack
MEAN-AUTH
Create Role
Update Role
Get All Roles
Delete Role
Login user
Register User
Register Admin
MEAN-AUTH / Register Admin
POST
http://localhost:8800/api/auth/register-admin
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Beautify
{
"firstName": "Sachintha",
"lastName": "Prasad",
"username": "psaprasad",
"email": "psaprasad@outlook.com",
"password": "2729729"
}
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
Visualize
HTML
1 Admin Registration Successful!
Type here to search
21°C Mostly cloudy
ENG
9:18 AM
10/10/2023

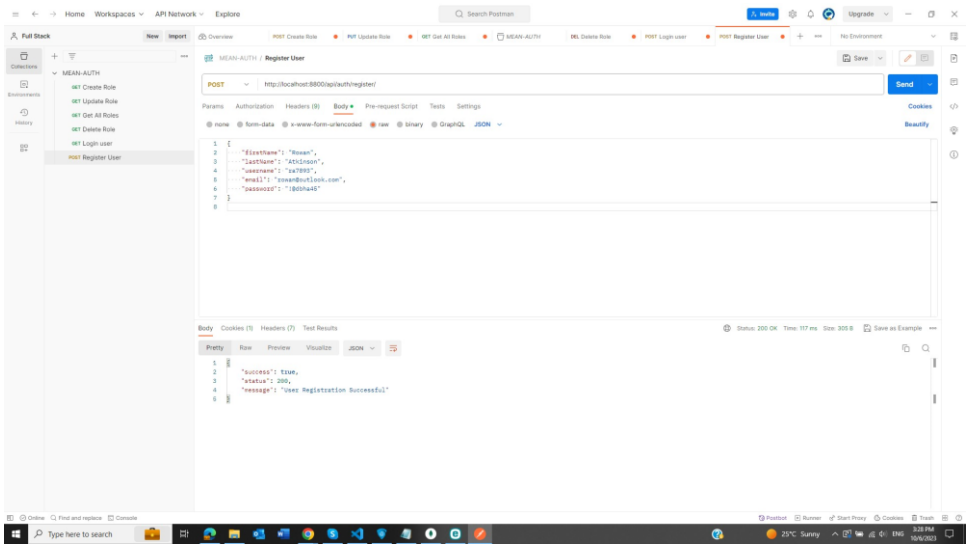

MongoDB Compass - localhost:27017/AuthDB.users
Connect
Edit
View
Collection
Help
localhost:27017
Documents
AuthDB.users
My Queries
Databases
AuthDB
roles
users
admin
config
employeeDB
local
AuthDB.users
7
DOCUMENTS
1
INDEXES
Documents
Aggregations
Schema
Indexes
Validation
Filter
Type a query: { field: 'value' } or Generate query
Explain
Reset
Find
Options
ADD DATA
EXPORT DATA
1 - 8 of 8
profileImage: ""
isAdmin: false
roles: Array (1)
createdAt: 2023-10-06T08:00:07.730+00:00
updatedAt: 2023-10-06T08:00:07.730+00:00
__v: 0
_id: ObjectId('651fb0bbfffc674247bed346')
firstName: "John"
lastName: "Cena"
username: "jc789"
email: "jc789@outlook.com"
password: "$2a$10$gKnYQhnv70nXaxNGGD7vOdoR/CMjSVdaSAwZZrgs1JbgvGmYA5je"
profileImage: ""
isAdmin: false
roles: Array (1)
createdAt: 2023-10-06T08:00:59.786+00:00
updatedAt: 2023-10-06T08:00:59.786+00:00
__v: 0
_id: ObjectId('6524bb4dff3a086ff33bcc15')
firstName: "Sachintha"
lastName: "Prasad"
username: "psaprasad"
email: "psaprasad@outlook.com"
password: "$2a$10$TSLZ8RhPQuoWS6Yzn54t8OdTWETb3S0G.8OWv7nObCAZEb8vUo56G"
profileImage: ""
isAdmin: true
roles: Array (2)
0: ObjectId('651fb0c054d4a0d7f00d117b')
1: ObjectId('651fbac054d4a0d7f00d117e')
createdAt: 2023-10-10T09:18:08.829+00:00
updatedAt: 2023-10-10T09:18:08.829+00:00
__v: 0
3_MONGOOSH
Type here to search
Top events
ENG
9:19 AM
10/10/2023

Testing on Postman

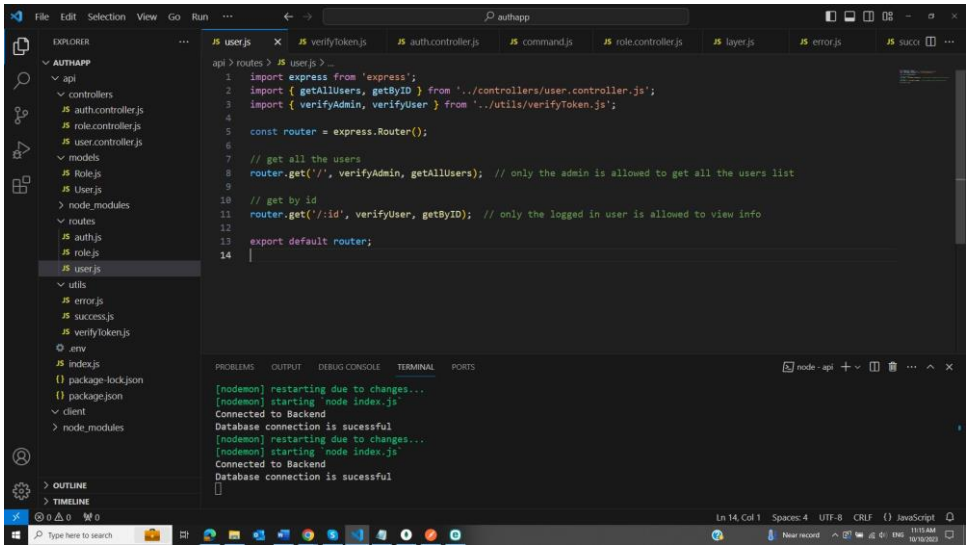


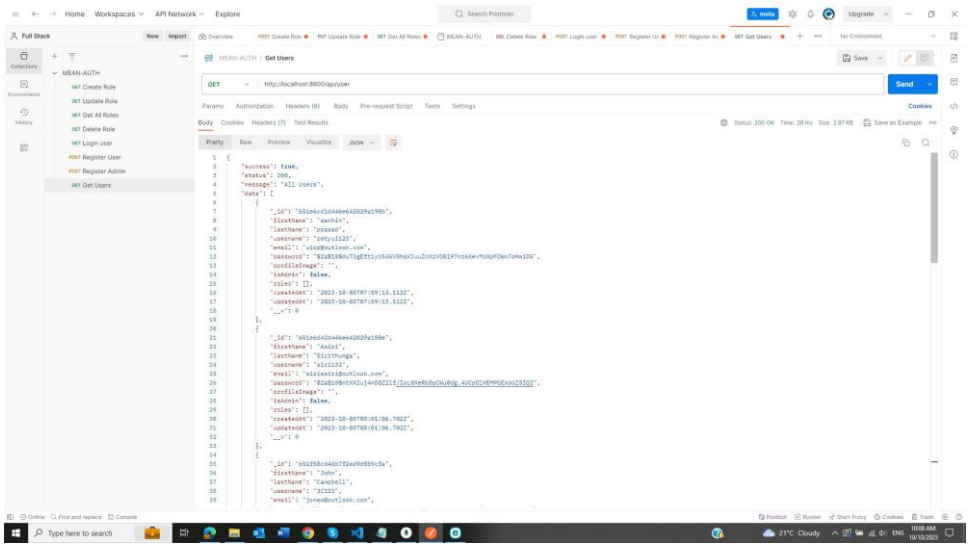


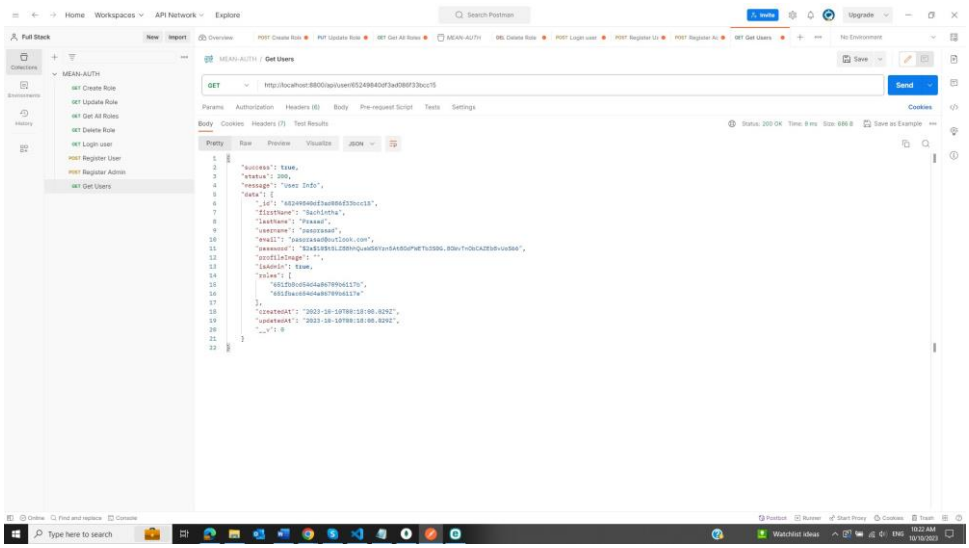
User Registration Testing on Postman



getAllUser and getById





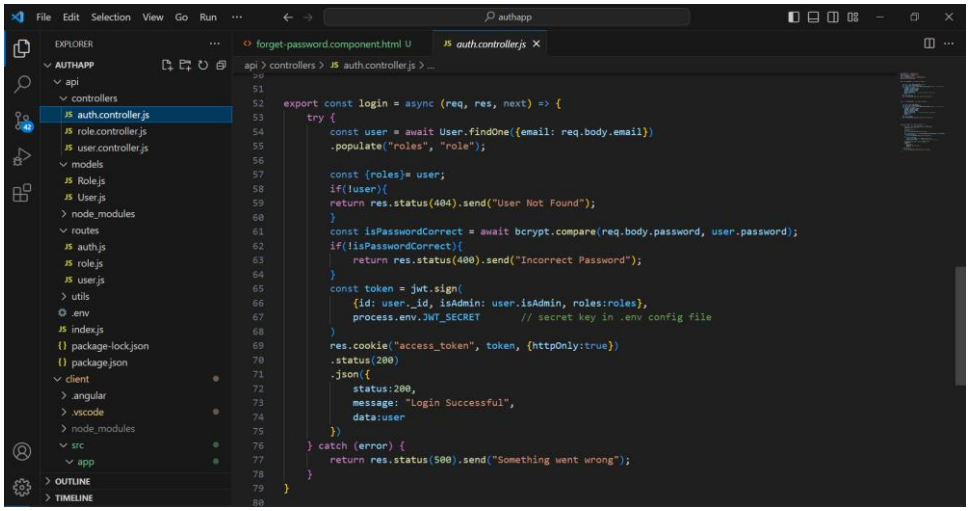


JSON WEB TOKENS

A token will be created only after the login is successful.

JWT Secret key is stored in the .env file

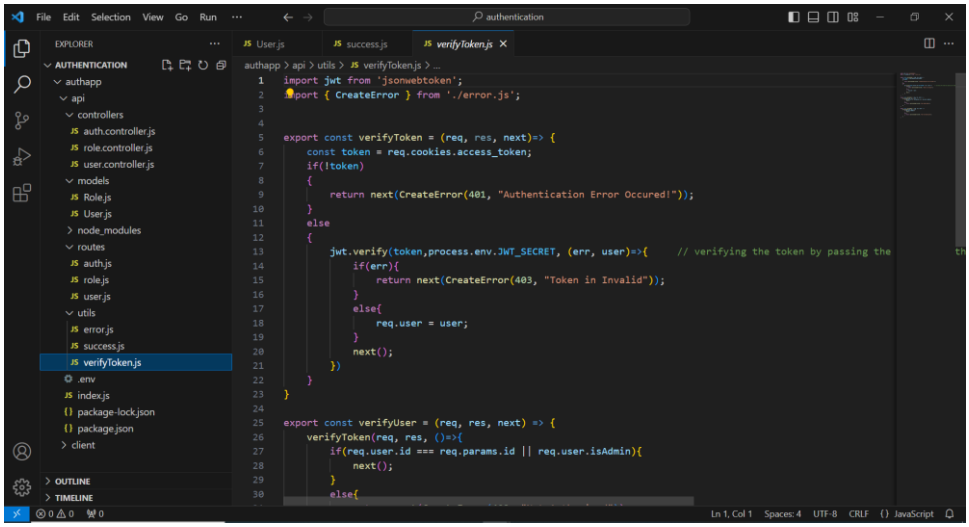
This will create the jwt token for the user:

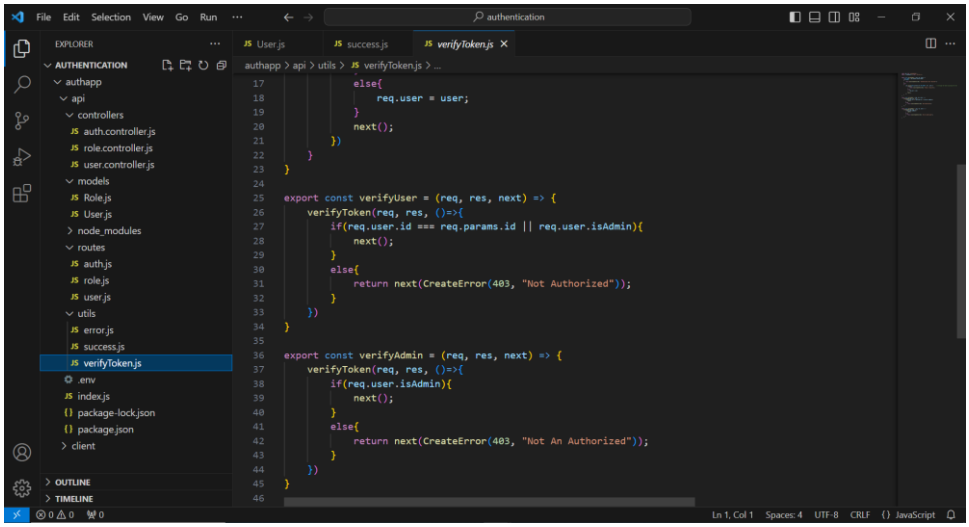


This token will be sent directly to the users using a cookie.

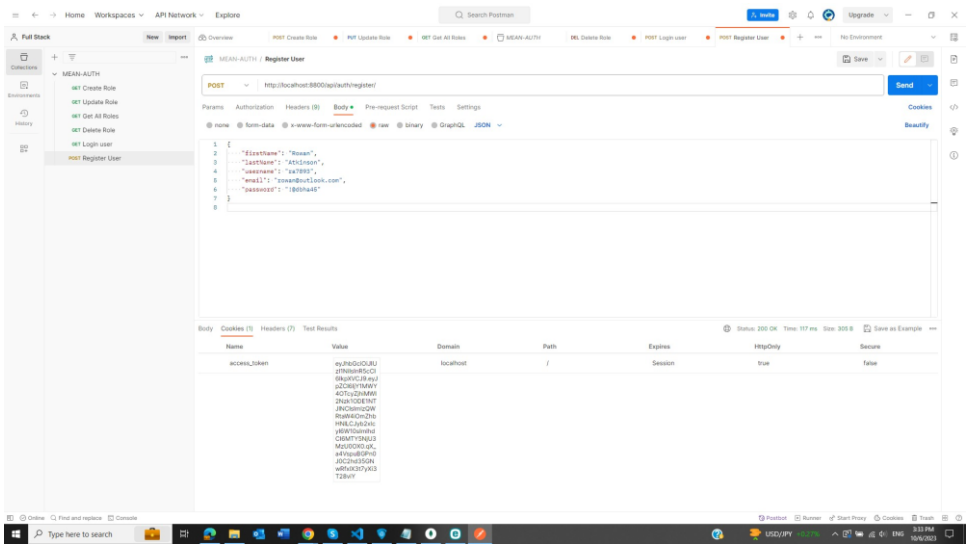
Protecting the API end point using JWT authentication.

Verifying Token(verifyToken.js)

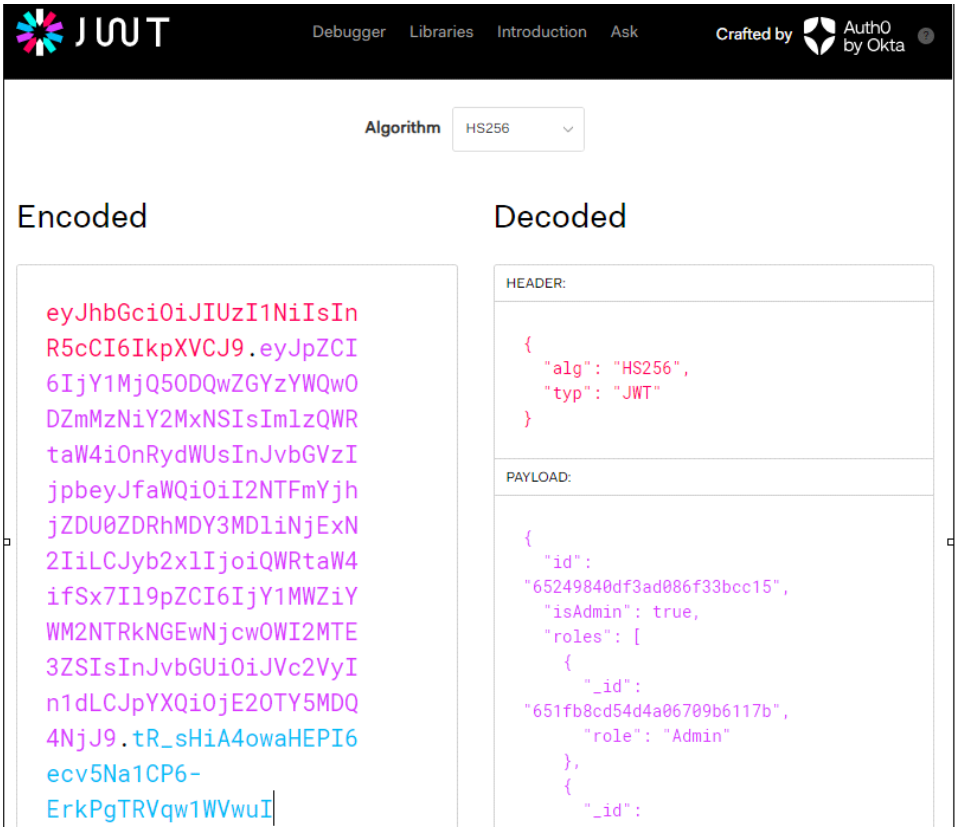




Access token cookie



Using jwt.io to decode the encoded tokens

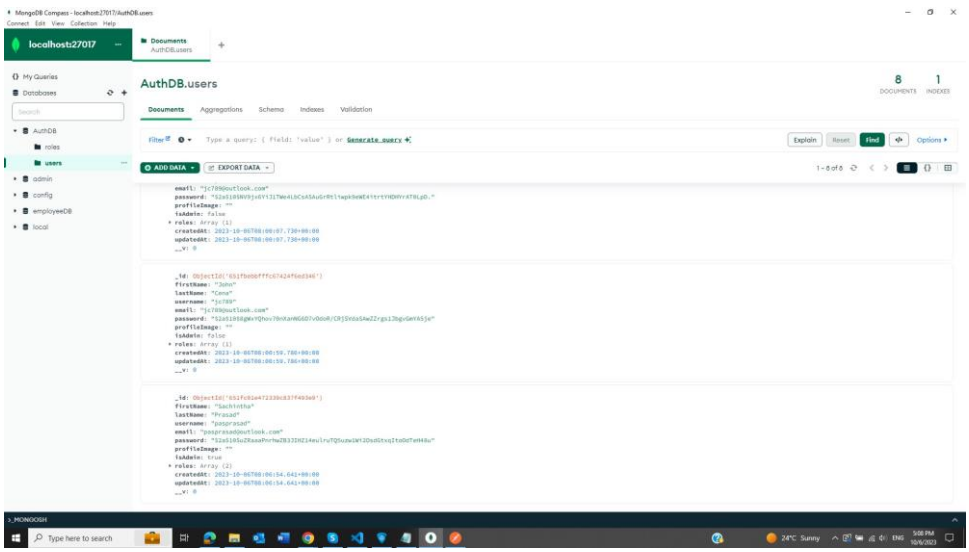


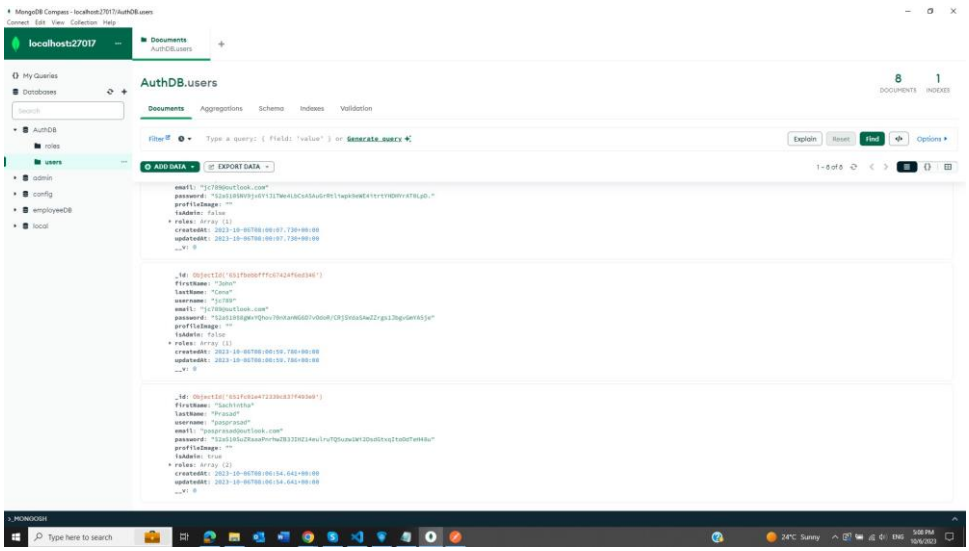
Encoded
Decoded
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY1MjQ5ODQwZGYzYWQwODZmMzNiY2MxNSIsImlzQWRtaW4iOnRydWUsInJvbGVzIjpbeyJfaWQiOiI2NTFmYjhjZDU0ZDRhMDY3MDliNjExN2IiLCJyb2xlIjoiQWRtaW4ifSx7Il9pZCI6IjY1MWZiYWM2NTRkNGEwNjcwOWI2MTE3ZSIsInJvbGUiOiJVc2VyIn1dLCJpYXQiOjE2OTY5MDQ4NjJ9.tR_sHiA4owaHEPI6ecv5Na1CP6-ErkPgTRVqw1WVwuI
HEADER:
{
"alg": "HS256",
"typ": "JWT"
}
PAYLOAD:
{
"id":
"65249840df3ad086f33bcc15",
"isAdmin": true,
"roles": [
{
"_id":
"651fb8cd54d4a06709b6117b",
"role": "Admin"
},
{
"_id":
"651fbac654d4a06709b6117e",
"role": "User"
}
],
"iat": 1696904862
}

The users' passwords have also been encrypted before storing on MongoDB.

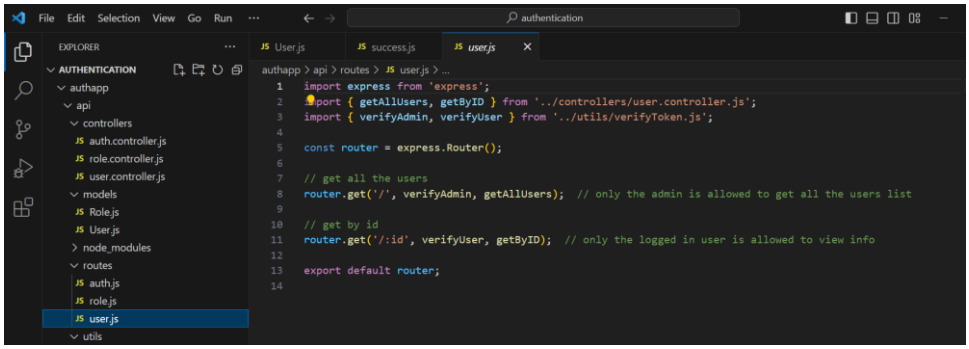
```
export const register = async (req, res, next) => {


    const role = await Role.find({role: 'User'});
    const salt = await bcrypt.genSalt(10);
    const hashPassword = await bcrypt.hash(req.body.password, salt);  // hash password
    const newUser = new User({
        firstName: req.body.firstName,
        lastName: req.body.lastName,
        username: req.body.userName,
        email: req.body.email,
        password: hashPassword,   // password encryption using bcrypt
        roles: role
    });
    await newUser.save();
    return res.status(200).json("User Registration Successful");
}
```

User Registration Details Stored on MongoDB.

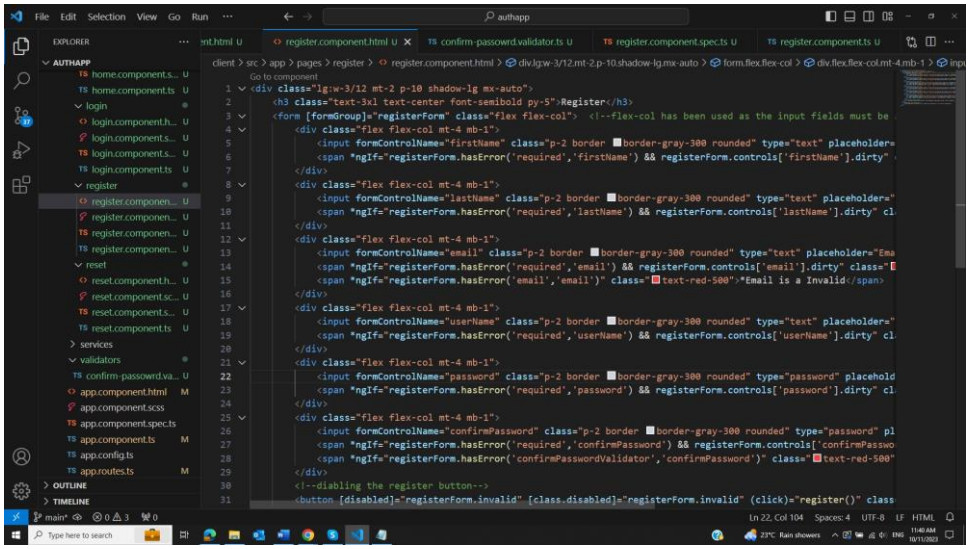


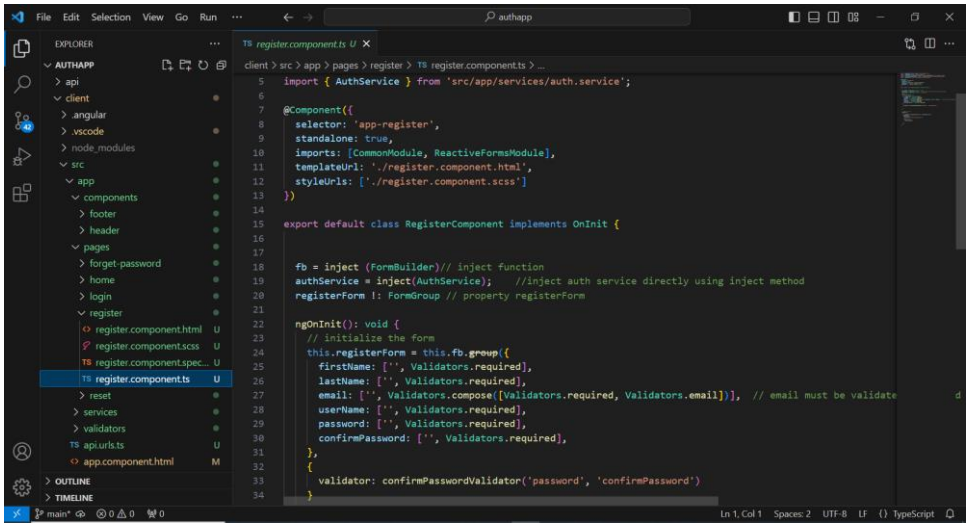


Only the logged in Admins are allowed to view user info. And only the logged in user can view profile info filtered by ID.



User Registration Form with Validations





In the register form the ‘Register’ button has been disabled if the fields are empty.

All the required fields must be filled before the user can click on the register button.

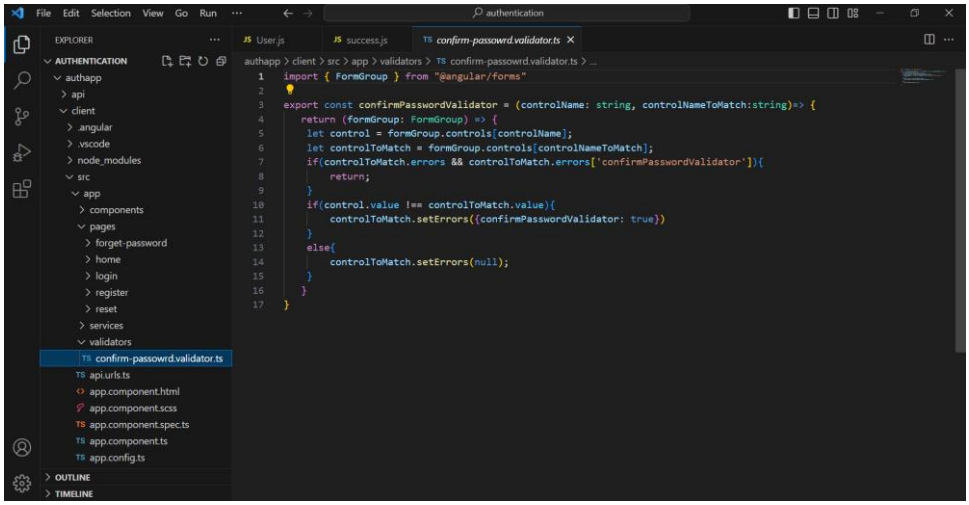
*ngIf="registerForm.hasError('required','firstName') &&
registerForm.controls['firstName'].dirty"

If the form fields become dirty, the required error will be displayed to the user. And only after all the required fields are filled with values, the submit button will be enabled and allowed to submit the registration details.

Creating custom validators for confirming password.

Validators folder of the application contains all the custom validators.

Ex: confirm-password.validator.ts file includes the custom validation for password confirmation.

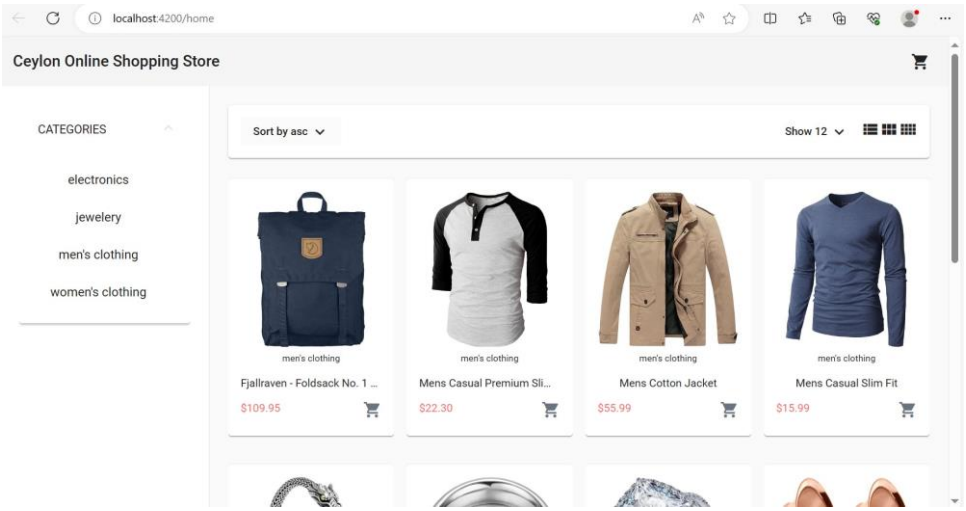


An e commerce store

This application has been developed with the aim of providing customers with an exclusive online shopping experience. The application allows users to select a range of products also with the ability to categorize the products based on four different product categories. The product categories include electronics, jewellery, men’s clothing and women’s clothing.

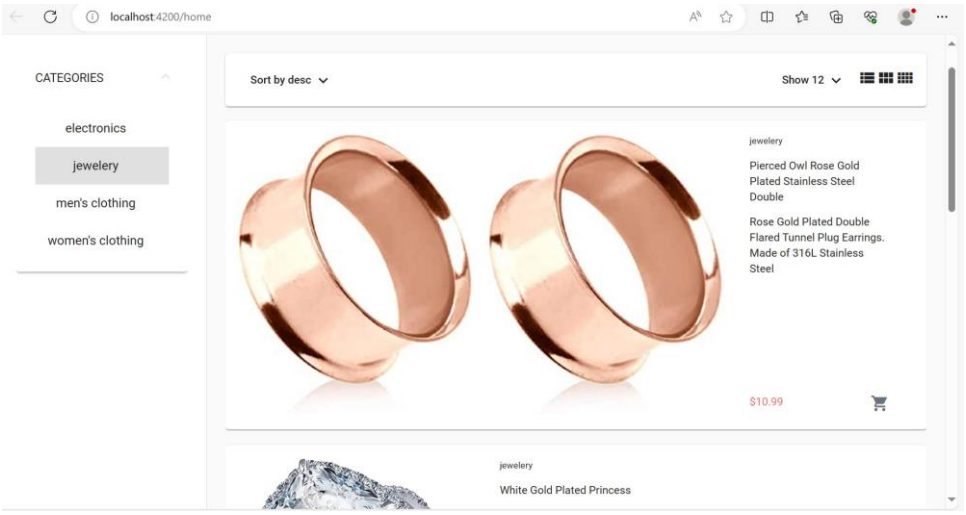
The categories section on the left side panel allows users to sort the products displayed on the shopping store. The header section on the top of the web store allows users to sort the displayed products according to the ascending order or descending order based on the user's preference.

On the right corner of this header section, the users can change the layout view of the products. Users can select how many items they wish to view along with the layout view preference.



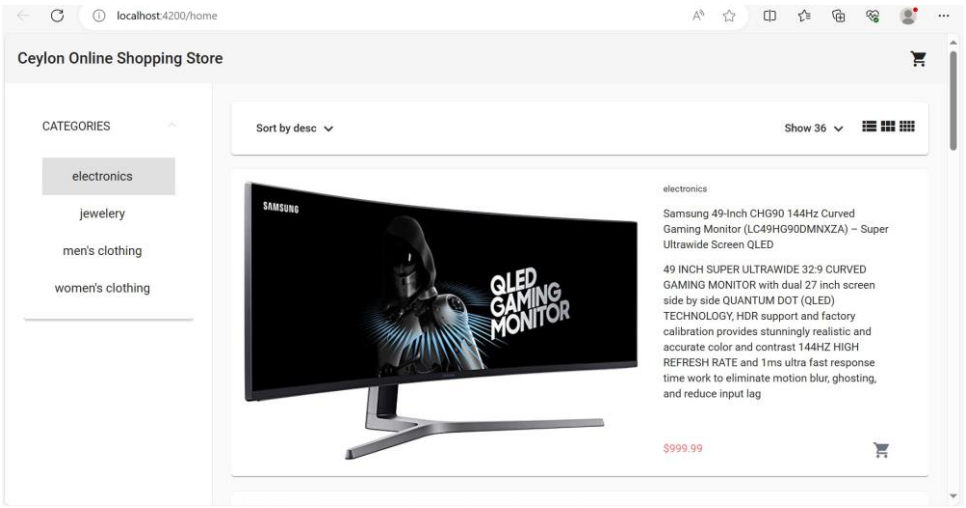
Fake store API has been used here for displaying products and Stripe has been used for the payment integration.

Product Layout View 1

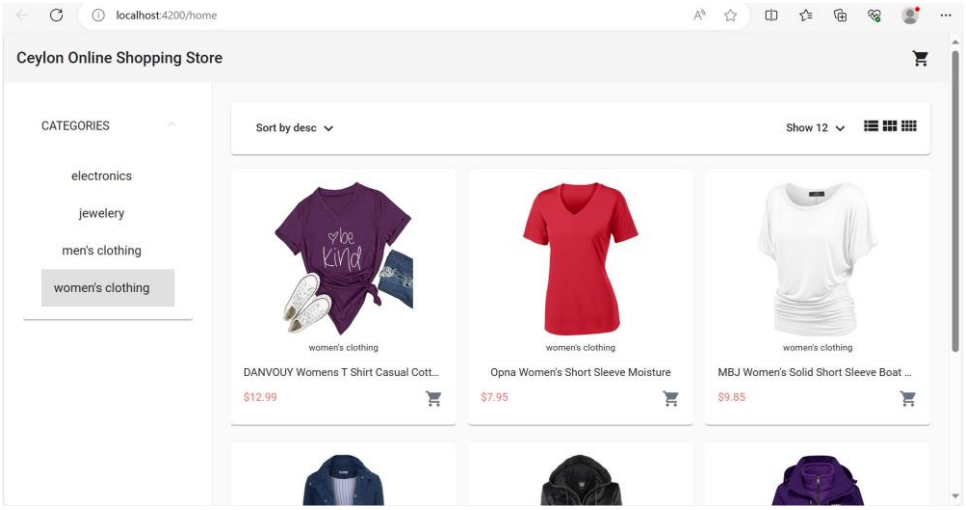


This allows users to view a single product on the web page at a time.

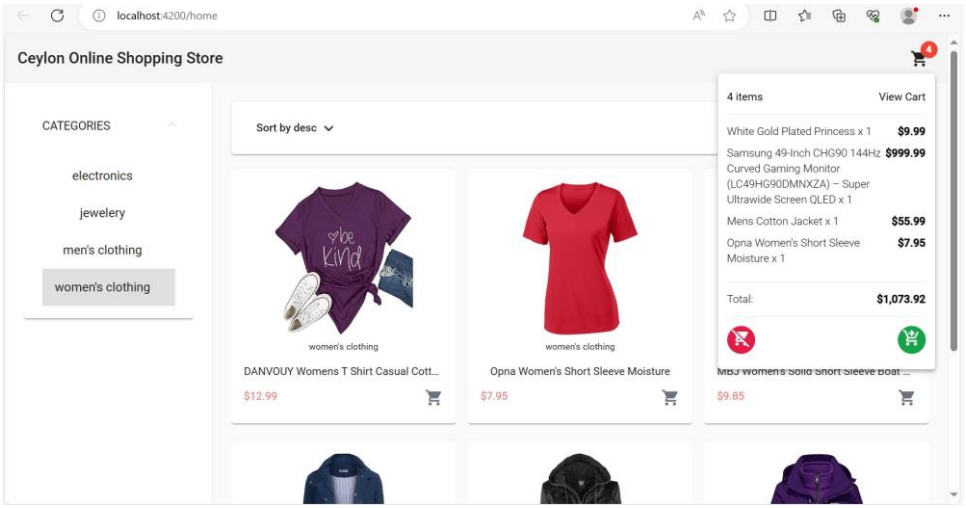
Product Layout View 2



Product Layout View 3



Cart View 1

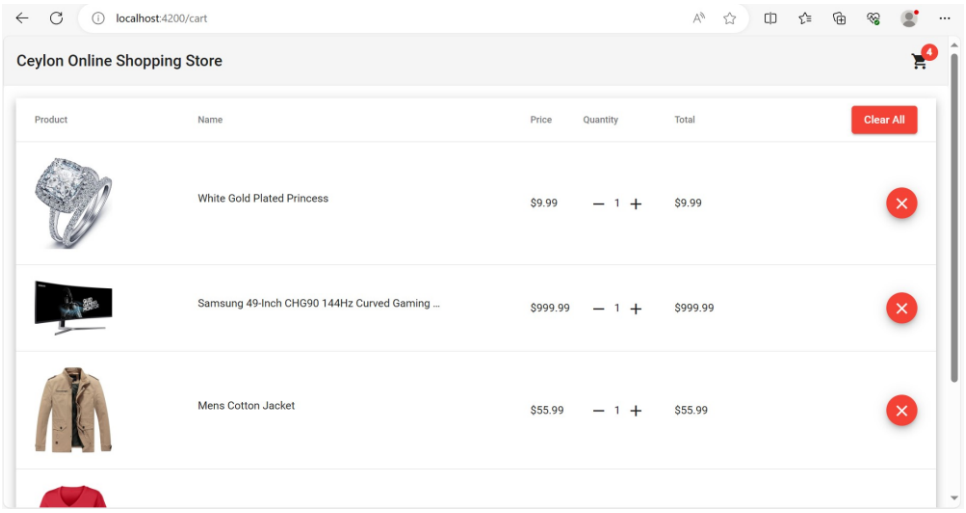


Products that have been added to the cart can be displayed by clicking on the cart item on the home page without navigating to the cart page.

The button displayed on the bottom left corner can be used to clear the products that have been already added to the cart.

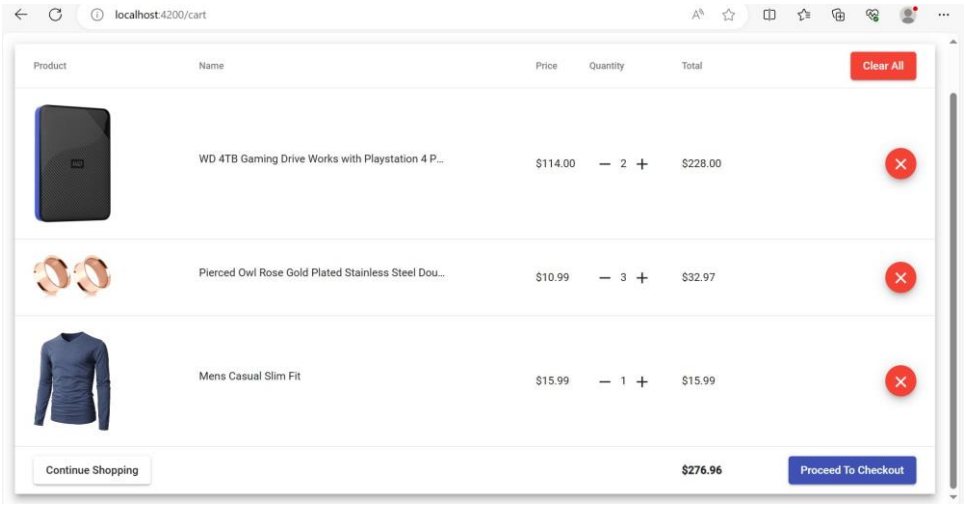
The button on the bottom right corner here can be used to proceed to the cart page as displayed below.

Cart View 2

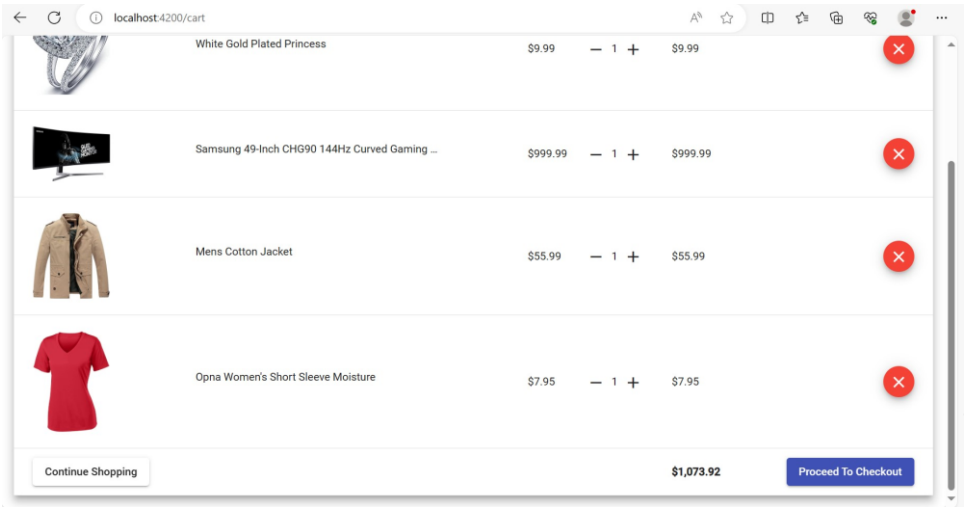


The “clear all” button displayed on the top right corner can be used to empty the cart by removing all the products added to the cart.

The quantity of the products can also be changed by clicking on the – and + signs.



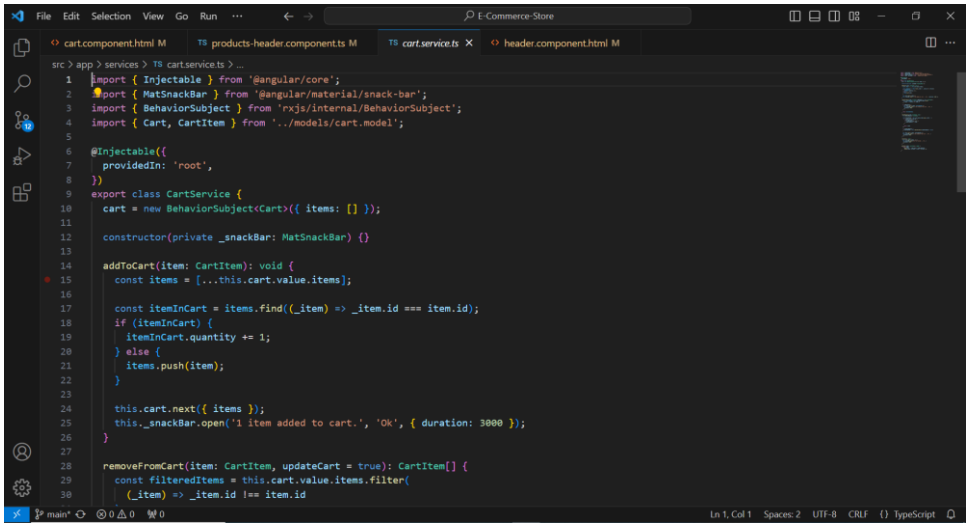
Cart View 3



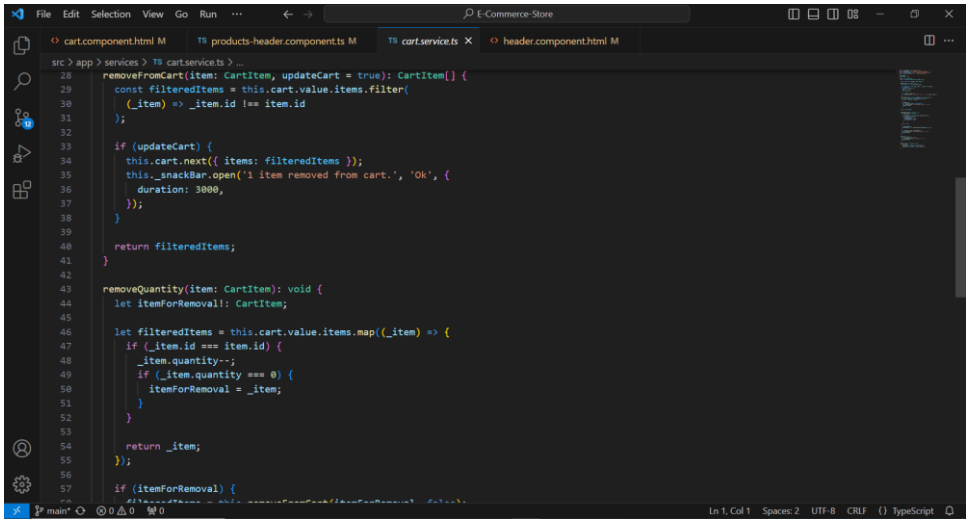
“Continue Shopping” button displayed on the bottom left corner can be used to navigate back to the Home page and add more products to the cart.

The total amount of all the products added to the cart can also be seen here on the bottom right corner of the cart page.

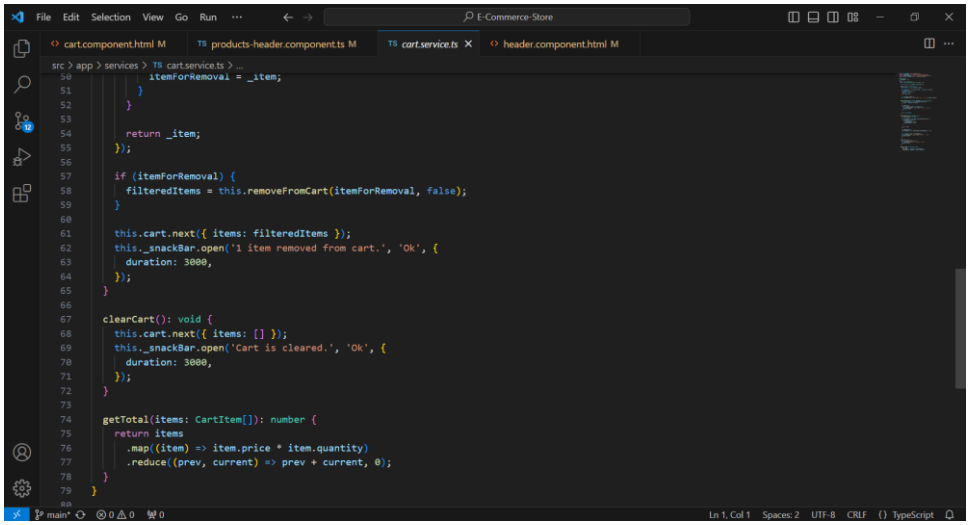
Cart Service



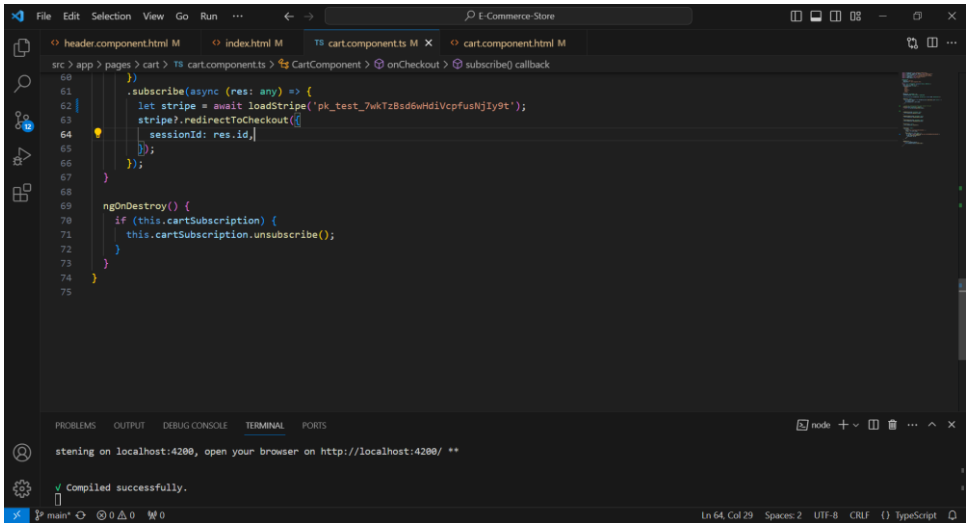
```
import { Injectable } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';
import { BehaviorSubject } from 'rxjs/internal/BehaviorSubject';
import { Cart, CartItem } from '../models/cart.model';

@Injectable({
  providedIn: 'root',
})
export class CartService {
  cart = new BehaviorSubject<Cart>({ items: [] });

  constructor(private _snackBar: MatSnackBar) {}

  addToCart(item: CartItem): void {
    const items = [...this.cart.value.items];

    const itemInCart = items.find((_item) => _item.id === item.id);
    if (itemInCart) {
      itemInCart.quantity += 1;
    } else {
      items.push(item);
    }

    this.cart.next({ items });
    this._snackBar.open('1 item added to cart.', 'Ok', { duration: 3000 });
  }

  removeFromCart(item: CartItem, updateCart = true): CartItem[] {
    const filteredItems = this.cart.value.items.filter(
      (_item) => _item.id !== item.id
```



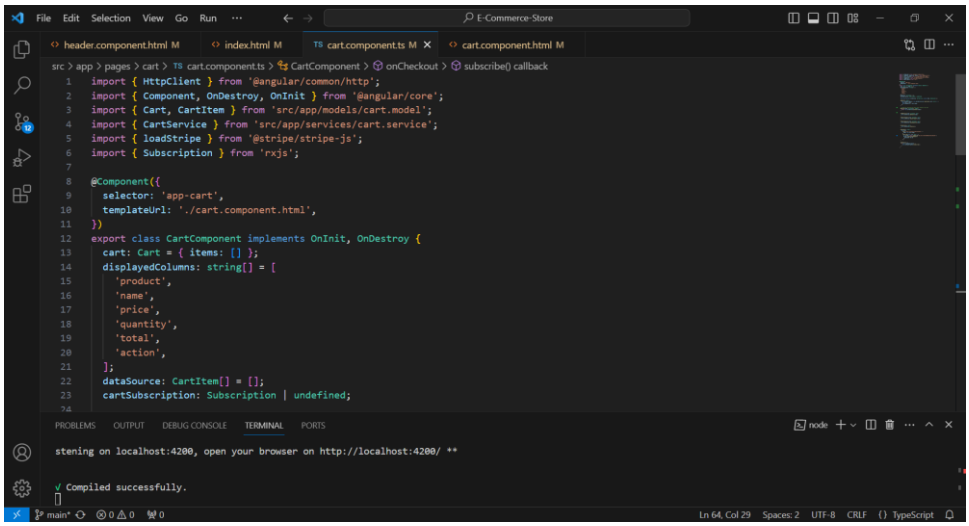
```
  removeFromCart(item: CartItem, updateCart = true): CartItem[] {
    const filteredItems = this.cart.value.items.filter(
      (_item) => _item.id !== item.id
    );

    if (updateCart) {
      this.cart.next({ items: filteredItems });
      this._snackBar.open('1 item removed from cart.', 'Ok', {
        duration: 3000,
      });
    }

    return filteredItems;
  }

  removeQuantity(item: CartItem): void {
    let itemForRemoval!: CartItem;

    let filteredItems = this.cart.value.items.map((_item) => {
      if (_item.id === item.id) {
        _item.quantity--;
        if (_item.quantity === 0) {
          itemForRemoval = _item;
        }
      }

      return _item;
    });

    if (itemForRemoval) {
```



Adding API key for Stripe Payment Integration



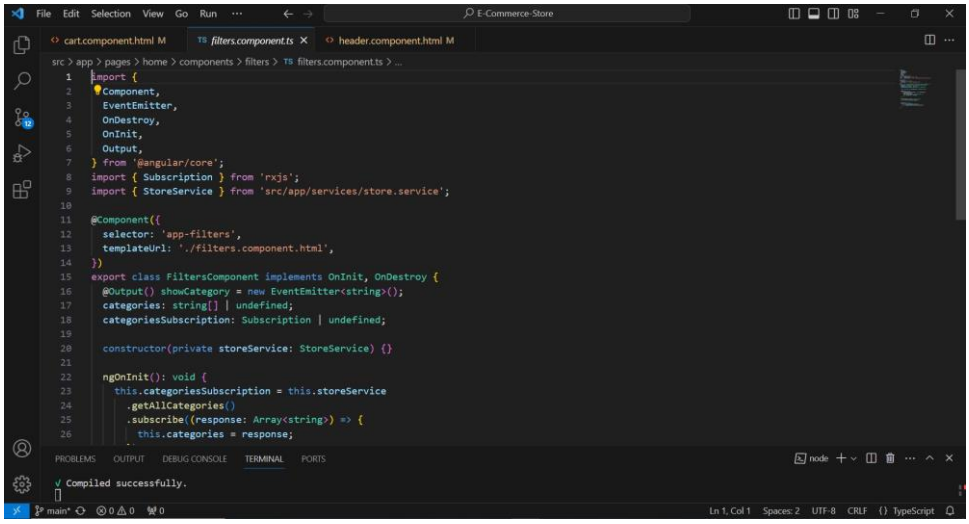
Cart Component Code



```
import { HttpClient } from '@angular/common/http';
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Cart, CartItem } from 'src/app/models/cart.model';
import { CartService } from 'src/app/services/cart.service';
import { loadStripe } from '@stripe/stripe-js';
import { Subscription } from 'rxjs';

@Component({
  selector: 'app-cart',
  templateUrl: './cart.component.html',
})
export class CartComponent implements OnInit, OnDestroy {
  cart: Cart = { items: [] };
  displayedColumns: string[] = [
    'product',
    'name',
    'price',
    'quantity',
    'total',
    'action',
  ];
  dataSource: CartItem[] = [];
  cartSubscription: Subscription | undefined;
```

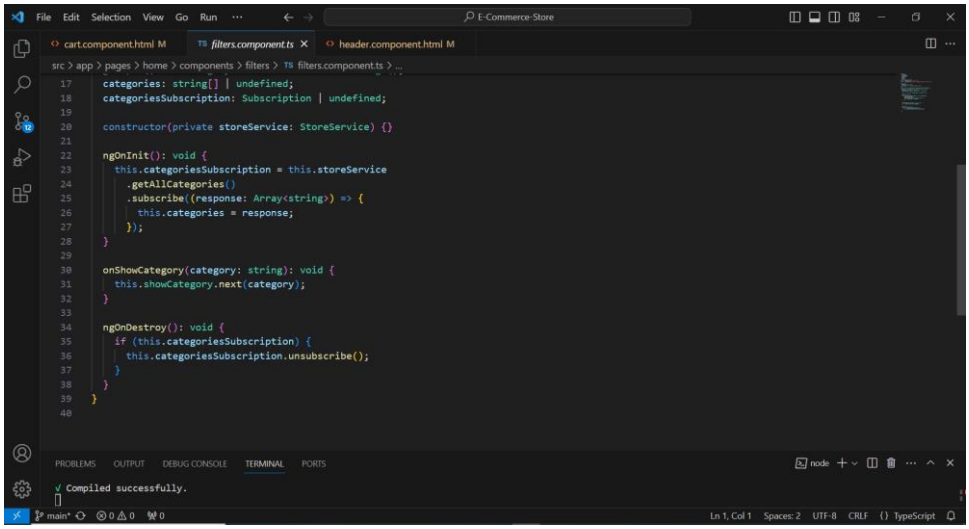
```
stening on localhost:4200, open your browser on http://localhost:4200/ **

√ Compiled successfully.
```

Filters Component

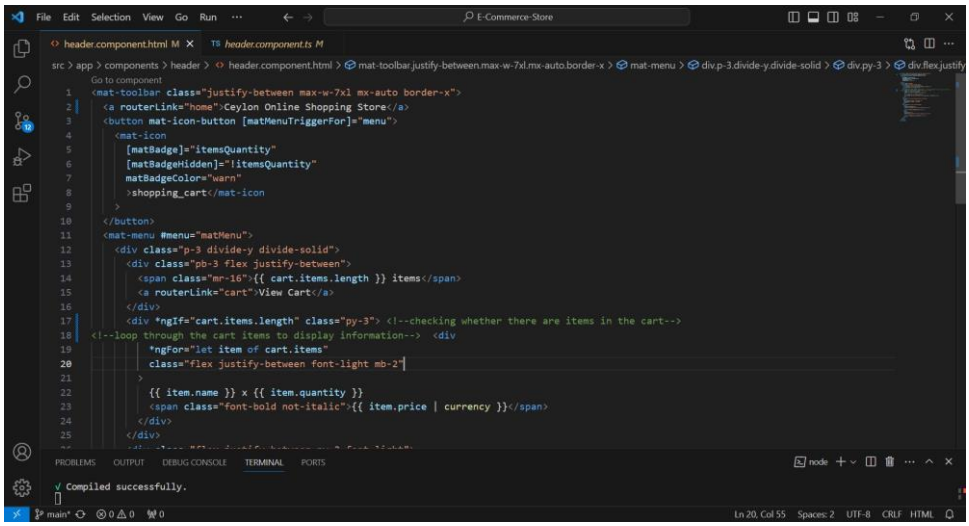


```
import {
  Component,
  EventEmitter,
  OnDestroy,
  OnInit,
  Output,
} from '@angular/core';
import { Subscription } from 'rxjs';
import { StoreService } from 'src/app/services/store.service';

@Component({
  selector: 'app-filters',
  templateUrl: './filters.component.html',
})
export class FiltersComponent implements OnInit, OnDestroy {
  @Output() showCategory = new EventEmitter<string>();
  categories: string[] | undefined;
  categoriesSubscription: Subscription | undefined;

  constructor(private storeService: StoreService) {}

  ngOnInit(): void {
    this.categoriesSubscription = this.storeService
      .getAllCategories()
      .subscribe((response: Array<string>) => {
        this.categories = response;
```

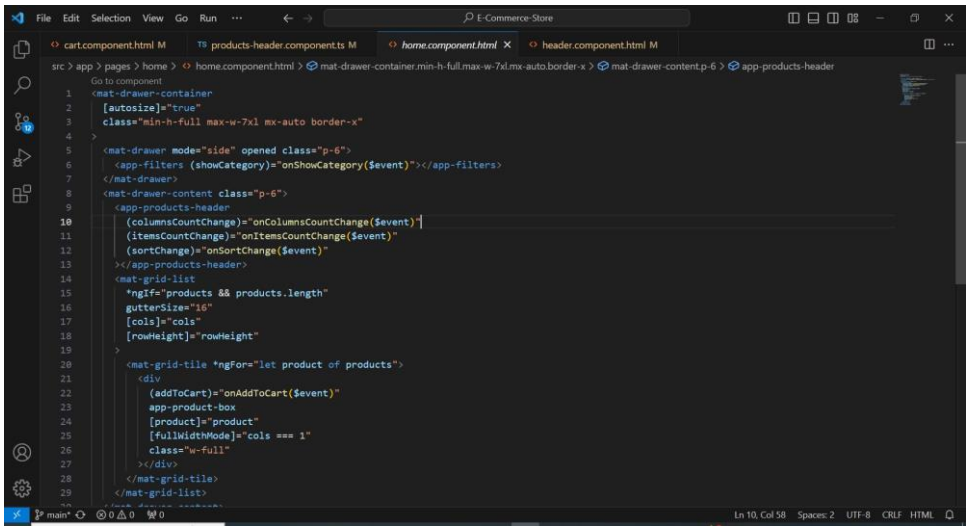
```
√ Compiled successfully.
```



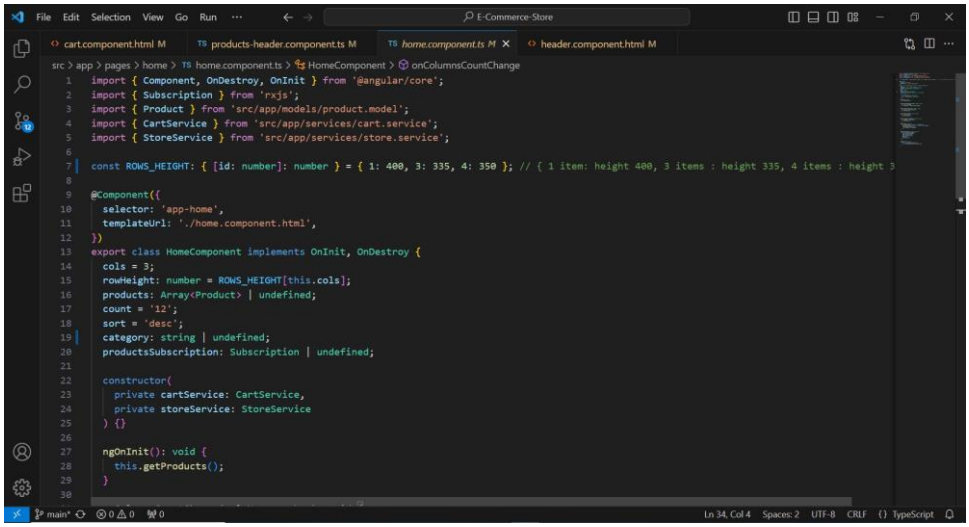
Header Component

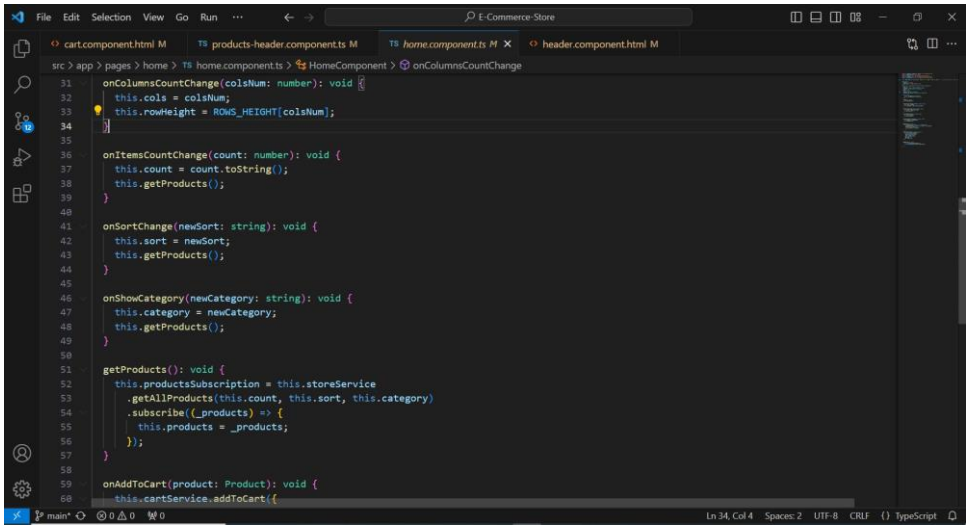


Home Component



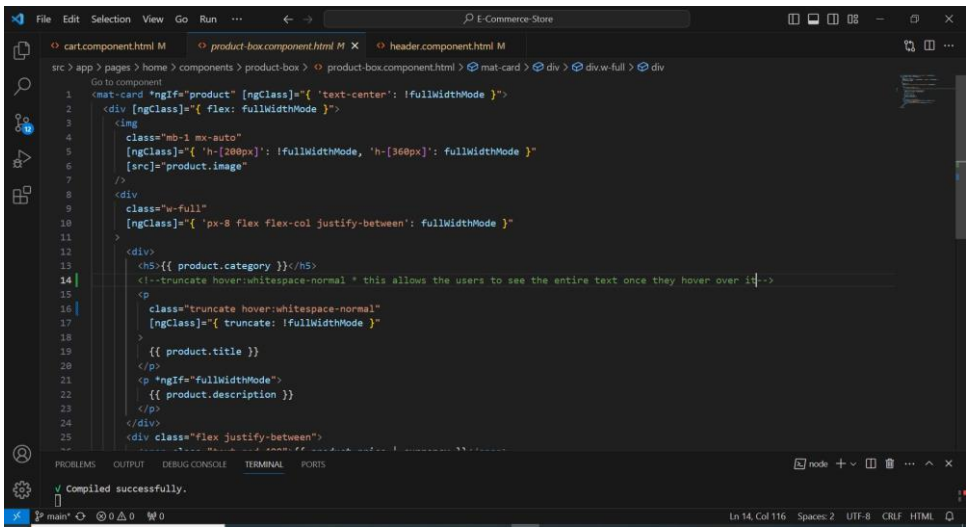
Here, mat-grid-list separates the child components into columns and rows.





```
onColumnsCountChange(colsNum: number): void {
  this.cols = colsNum;
  this.rowHeight = ROWS_HEIGHT[colsNum];
}

onItemsCountChange(count: number): void {
  this.count = count.toString();
  this.getProducts();
}

onSortChange(newSort: string): void {
  this.sort = newSort;
  this.getProducts();
}

onShowCategory(newCategory: string): void {
  this.category = newCategory;
  this.getProducts();
}

getProducts(): void {
  this.productsSubscription = this.storeService
    .getAllProducts(this.count, this.sort, this.category)
    .subscribe((_products) => {
      this.products = _products;
    });
}

onAddToCart(product: Product): void {
  this.cartService.addToCart({
```

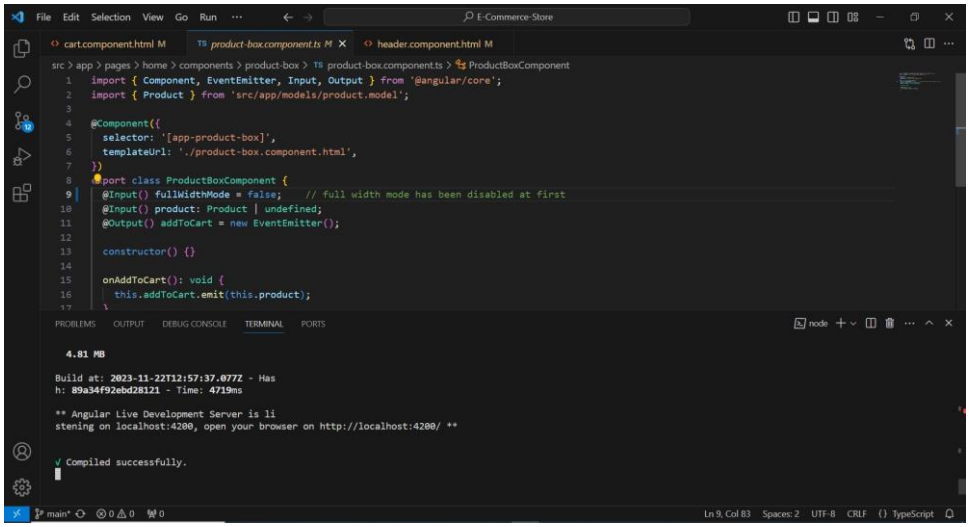
Product Box Component



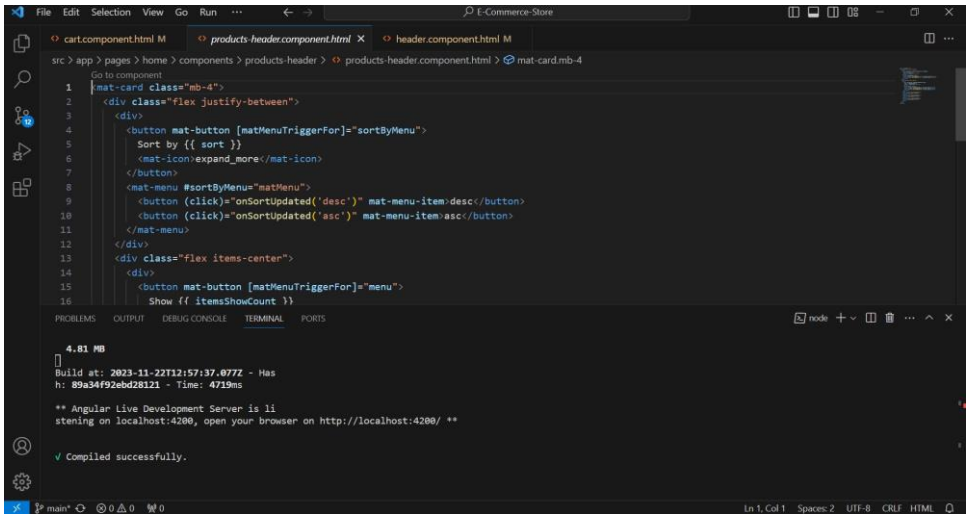
```
<mat-card *ngIf="product" [ngClass]="{ 'text-center': !fullWidthMode }">
  <div [ngClass]="{ flex: fullWidthMode }">
    <img
      class="mb-1 mx-auto"
      [ngClass]="{ 'h-[200px]': !fullWidthMode, 'h-[360px]': fullWidthMode }"
      [src]="product.image"
    />
    <div
      class="w-full"
      [ngClass]="{ 'px-8 flex flex-col justify-between': fullWidthMode }"
    >
      <div>
        <h5>{{ product.category }}</h5>
        <!--truncate hover:whitespace-normal * this allows the users to see the entire text once they hover over it-->
        <p
          class="truncate hover:whitespace-normal"
          [ngClass]="{ truncate: !fullWidthMode }"
        >
          {{ product.title }}
        </p>
        <p *ngIf="fullWidthMode">
          {{ product.description }}
        </p>
      </div>
      <div class="flex justify-between">
```

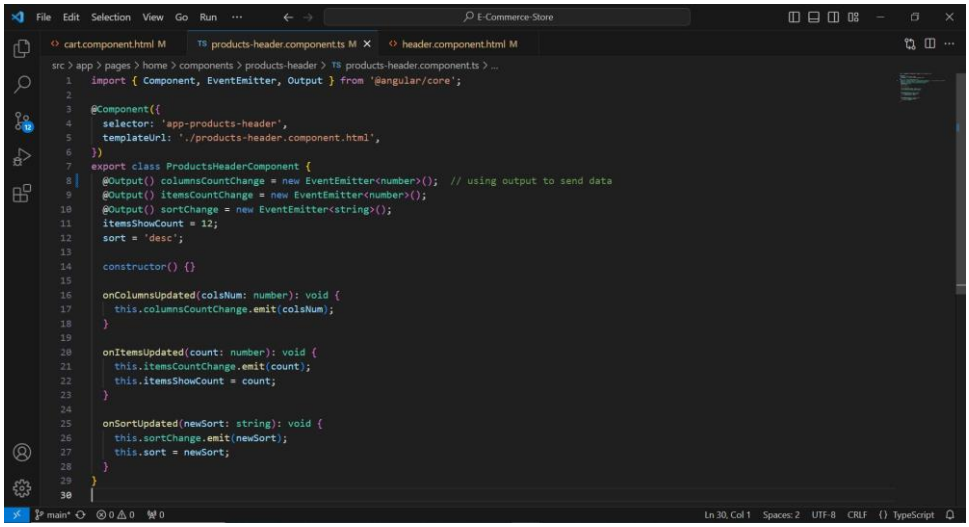
```
√ Compiled successfully.
```

Once the user hover over a specific item, the entire title description for that product can be viewed by the users.

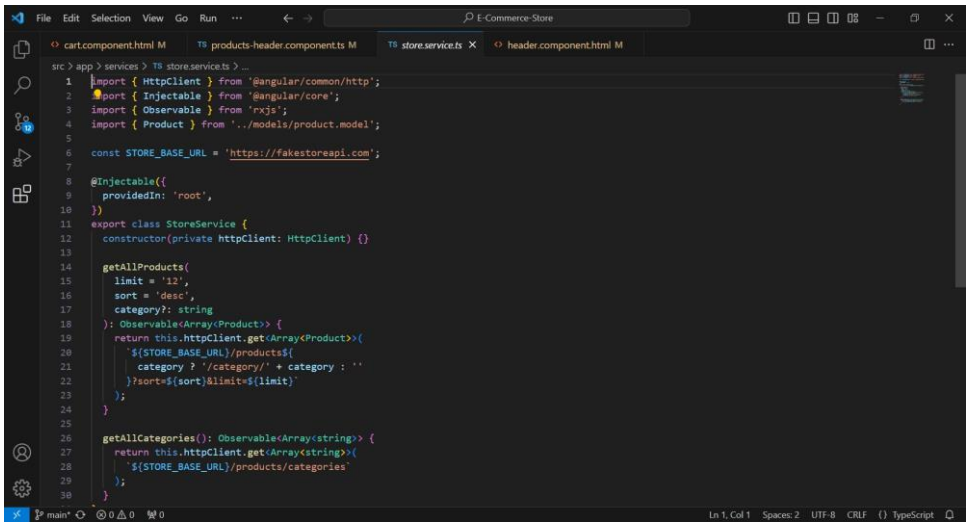


Product Header Component





Store Service code



Fake Store API has been used to display products on the web store.

Dear Sir/Madam,

I have not been able to fully complete this coursework as a result of my severe health condition which had been affecting my day-to-day activities in a negative manner over a long period of time. I have also been attending an internship in Japan at a Japanese IT company which also affected my ability to work on this.

I sincerely apologise for not being able to work my best on this coursework.

This is the only coursework that I had to complete as a repeat module during the last three years of my degree program. I have been able to pass all the exams and course works at the first attempt and I was keen on maintaining that perfect academic record until the end of my degree program. However, my health condition and some other factors have collectively affected my ability work longer periods of time and perform at my full capacity. The pain associated with my health condition sometimes last for couple of hours which leaves me incapable of working as I have to lay down after applying and taking the medications.

I kindly request you to allow me to grant a passing mark as I may not get another opportunity to complete this coursework before graduation.

I have also included some documents as evidence.

Thank You!

RUHIRU MEDI CARE CENTRE

Dr T I Weerasinghe
MBBS (Peradeniya)(SL)
Reg No: 20268
HWH

2/2, Galwana Junction,
Udumulla road,
Mulleriyawa, Sri Lanka
Tel: 0767628545

To: Dean, School of Engineering, Computing, Mathematics

Medical Certificate

I certify that Mr. P.A.S Prasad was under my care since January, receiving treatments for a chronic anal fissure and its associated symptoms of rectal bleeding, constipation, and severe abdominal pain during and after bowel movements. Mr. Prasad's condition has not been improved with medicine and other treatments and several recurrences of the condition has occurred during the last couple of months.

As a result, I have determined that he needs to go through a Lateral Internal Sphincterotomy surgery as the next immediate step of treatment. I have further instructed Mr. Prasad to consult Dr Sanjeewa Seneviratne (Consultant Surgeon) to receive further treatments including the surgery.

I hereby confirm that Mr. Prasad was not in good condition to carry out academic studies and other related tasks during the several recurrences of his condition which affected the overall quality of his day to day activities.

Additionally, it may take 1-2 weeks for him to recover after the surgery and get back to his normal routine and I request to provide him with the assistance he needs throughout this time.

Date: 20/05/2023

Medical Officer:

Dr. T.I. Weerasinghe
M.B.B.S. (Peradeniya)
SLMC Reg. No. 20268

社団法人
組込みシステム技術協会
Japan Embedded Systems Technology Association

29/03/2023

Internship Offer

Dear Palle Arachchilage Sachintha Prasad,

I am pleased to confirm your internship as "Sri Lanka Embedded Engineer Training Program" in BRYCEN Co. Ltd. Your duties and assignments for this position are as follows.

For the purpose of working as an advanced IT engineer in Japan in the future

(1) Acquisition of industrial equipment development in Japan

(2) Acquisition of teamwork in product development

(3) Accustomed to the living environment in Japan

Your first day of work will be 1st August 2023, and your internship is scheduled to end on 31st October 2023, in 30F St.Luke's Tower, 8-1 Akashi-cho, Chuo-ku, Tokyo 104-6591 Japan.

We provide flight tickets, company housing, living expenses, etc.

If you have any questions, please feel free to contact Yuki Kitamura (ykitamura@brycen.co.jp). We wish you will have a good work experience to improve your career opportunities during the internship.

Sincerely,

Yuki Kitamura

Human Resourses Department

Brycen Co., Ltd.

St.Luke's Tower 30th floor, 8-1 Akashicho Minato-Ku Tokyo

Phone: +81-(0)3- 6264-7221

JASA
本部：東京都 中央区 入船 1-5-11 弘報ビル 5 階　〒104-0042
1-5-11Irifune Chuo Tokyo Japan 104-0042
TEL:03-6372-0211　FAX:03-6372-0212　Email：jasainfo@jasa.or.jp
支部：北海道、東北、関東、中部、北陸、近畿、九州