**Increment 4 – Code**

**EBS**

**Team 6**

**Server Code:**

Server.js

//ebs application start file - server  
  
  
// required packages  
const ***express*** = require('express'); //express package to render HTML pages using JS  
const morgan = require('morgan'); //Morgan is used for logging request details  
const ***bodyParser*** = require('body-parser'); //body-parser to parse the JSON Data   
const ***mongoose*** = require('mongoose'); //Mongoose package to connect to back-end mongoDB  
const cors = require('cors'); //Package to connect middle-ware or cross-platform applications  
const ***config*** = require('./config');  
  
const app = ***express***(); //wrapping the new express application in app variable   
  
// connect to MongoDB  
***mongoose***.connect(***config***.database, { useNewUrlParser: true }, err => {  
 if (err) {  
 ***console***.log(err);  
 } else {  
 ***console***.log('Connected to the database');  
 }  
});  
  
//express application using required packages   
app.use(***bodyParser***.json());  
app.use(***bodyParser***.urlencoded({ extended: false }));  
app.use(morgan('dev'));  
app.use(cors());  
  
const ***userRoutes*** = require('./routes/account');  
const ***mainRoutes*** = require('./routes/main');  
const ***sellerRoutes*** = require('./routes/seller');  
const ***productSearchRoutes*** = require('./routes/product-search');  
  
//express application using Routes from this application  
app.use('/api', ***mainRoutes***);  
app.use('/api/accounts', ***userRoutes***);  
app.use('/api/seller', ***sellerRoutes***);  
app.use('/api/search', ***productSearchRoutes***);  
  
  
//Setting up the port for server to run on   
app.listen(***config***.port, err => {  
 ***console***.log('Server connected at port: ' + ***config***.port);  
});

Config.js

//configurations required  
module.exports = {  
 database: "mongodb+srv://EBSAdmin:YF$fZtcFNf5G\_cm@ebs-cluster.fdbo9.mongodb.net/EBS?retryWrites=true&w=majority",  
 port: 3030,  
 secret: 'V6hWnHCSaHOY9unEjufSndna25G5OcZu',  
 access\_key: "AKIATGOLB5WUHZKUDL55",  
 access\_secret: "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",  
 algolia\_api\_key: "7c30920f765d1bdeb20a9dbb938e7975",  
 algolia\_app\_id: "IW0O07R0KJ",  
 algolia\_index: "ebs"  
 };

**Routes:**

Account.js

// Account.JS file to maintain every users account details(SignUp,Login,Orders) and handle routes   
  
//Including the required packages and assigning it to Local Variables  
const ***router*** = require('express').Router();  
const ***jwt*** = require('jsonwebtoken');  
const ***User*** = require('../models/user');  
const ***Order*** = require('../models/order');  
const ***config*** = require('../config');  
const checkJWT = require('../middlewares/check-jwt');  
  
  
//Function to facilitate Sign Up feature   
***router***.post('/signup', (req, res, next) => {  
 let user = new ***User***();  
 user.name = req.body.name;  
 user.email = req.body.email;  
 user.password = req.body.password;  
 user.picture = user.gravatar();  
 user.isSeller = req.body.isSeller;  
  
 ***User***.findOne({ email: req.body.email }, (err, existingUser) => {  
 if (existingUser) {  
 res.json({  
 success: false,  
 message: 'Account with that email is already exist'  
 });  
  
 } else {  
 user.save();  
  
 var token = ***jwt***.sign({  
 user: user  
 }, ***config***.secret, {  
 expiresIn: '7d'  
 });  
  
 res.json({  
 success: true,  
 message: 'Enjoy your token',  
 token: token  
 });  
 }  
  
 });  
});  
  
//Function to facilitate login feature  
***router***.post('/login', (req, res, next) => {  
  
 ***User***.findOne({ email: req.body.email }, (err, user) => {  
 if (err) throw err;  
  
 if (!user) {  
 res.json({  
 success: false,  
 message: 'Authenticated failed, User not found'  
 });  
 } else if (user) {  
  
 var validPassword = user.comparePassword(req.body.password);  
 if (!validPassword) {  
 res.json({  
 success: false,  
 message: 'Authentication failed. Wrong password'  
 });  
 } else {  
 var token = ***jwt***.sign({  
 user: user  
 }, ***config***.secret, {  
 expiresIn: '7d'  
 });  
  
 res.json({  
 success: true,  
 mesage: "Enjoy your token",  
 token: token  
 });  
 }  
 }  
  
 });  
});  
  
  
//Function to handle Profile API (GET,POST) functionality for authenticated users   
***router***.route('/profile')  
 .get(checkJWT, (req, res, next) => {  
 ***User***.findOne({ \_id: req.decoded.user.\_id }, (err, user) => {  
 res.json({  
 success: true,  
 user: user,  
 message: "Successful"  
 });  
 });  
 })  
 .post(checkJWT, (req, res, next) => {  
 ***User***.findOne({ \_id: req.decoded.user.\_id }, (err, user) => {  
 if (err) return next(err);  
  
 if (req.body.name) user.name = req.body.name;  
 if (req.body.email) user.email = req.body.email;  
 if (req.body.password) user.password = req.body.password;  
  
 user.isSeller = req.body.isSeller;  
  
 user.save();  
 res.json({  
 success: true,  
 message: 'Successfully edited your profile'  
 });  
 });  
 });  
  
 ***router***.route('/address')  
 .get(checkJWT, (req, res, next) => {  
 ***User***.findOne({ \_id: req.decoded.user.\_id }, (err, user) => {  
 res.json({  
 success: true,  
 address: user.address,  
 message: "Successful"  
 });  
 });  
 })  
 .post(checkJWT, (req, res, next) => {  
 ***User***.findOne({ \_id: req.decoded.user.\_id }, (err, user) => {  
 if (err) return next(err);  
  
 if (req.body.addr1) user.address.addr1 = req.body.addr1;  
 if (req.body.addr2) user.address.addr2 = req.body.addr2;  
 if (req.body.city) user.address.city = req.body.city;  
 if (req.body.state) user.address.state = req.body.state;  
 if (req.body.country) user.address.country = req.body.country;  
 if (req.body.postalCode) user.address.postalCode = req.body.postalCode;  
   
 user.save();  
 res.json({  
 success: true,  
 message: 'Successfully edited your address'  
 });  
 });  
 });  
  
  
  
   
   
 //Function to handle Orders functionality for authenticated users   
***router***.get('/orders', checkJWT, (req, res, next) => {  
 ***Order***.find({ owner: req.decoded.user.\_id })  
 .populate('products.product')  
 .populate('owner')  
 .exec((err, orders) => {  
 if (err) {  
 res.json({  
 success: false,  
 message: "Couldn't find your order"  
 });  
 } else {  
 res.json({  
 success: true,  
 message: 'Found your order',  
 orders: orders  
 });  
 }  
 });  
 });  
  
 //Function to handle specific order functionality   
 ***router***.get('/orders/:id', checkJWT, (req, res, next) => {  
 ***Order***.findOne({ \_id: req.params.id })  
 .deepPopulate('products.product.owner')  
 .populate('owner')  
 .exec((err, order) => {  
 if (err) {  
 res.json({  
 success: false,  
 message: "Couldn't find your order"  
 });  
 } else {  
 res.json({  
 success: true,  
 message: 'Found your order',  
 order: order  
 });  
 }  
 });  
 });  
  
  
//Exporting the module   
module.exports = ***router***;

Main.js

//main.JS file to facilitate REST services for Products,Categories, Review and payment functionality   
  
//Including the required packages and assigning it to Local Variables  
const ***router*** = require('express').Router();  
const async = require('async');  
const stripe = require('stripe')('sk\_test\_51JpKC0CRoz77xUcIoMH80m5oUEbp3i2wv7JYA6BB7XVEqbnX14ldn1SLXNjO6VrnVaaKVc51hITHHdxnTNsJVCNT00cXAhwAc9');  
  
const ***Category*** = require('../models/category');  
const ***Product*** = require('../models/product');  
const ***Review*** = require('../models/review');  
const ***Order*** = require('../models/order');  
  
const checkJWT = require('../middlewares/check-jwt');  
  
  
//Function to facilitate obtaining the product information   
***router***.get('/products', (req, res, next) => {  
 const perPage = 10;  
 const page = req.query.page;  
 async.parallel([  
 function(callback) {  
 ***Product***.count({}, (err, count) => {  
 var totalProducts = count;  
 callback(err, totalProducts);  
 });  
 },  
 function(callback) {  
 ***Product***.find({})  
 .skip(perPage \* page)  
 .limit(perPage)  
 .populate('category')  
 .populate('owner')  
 .exec((err, products) => {  
 if(err) return next(err);  
 callback(err, products);  
 });  
 }  
 ], function(err, results) {  
 var totalProducts = results[0];  
 var products = results[1];  
   
 res.json({  
 success: true,  
 message: 'category',  
 products: products,  
 totalProducts: totalProducts,  
 pages: ***Math***.ceil(totalProducts / perPage)  
 });  
 });  
   
});  
  
//Function to facilitate categories GET and POST requests   
***router***.route('/categories')  
 .get((req, res, next) => {  
 ***Category***.find({}, (err, categories) => {  
 res.json({  
 success: true,  
 message: "Success",  
 categories: categories  
 })  
 })  
 })  
 .post((req, res, next) => {  
 let category = new ***Category***();  
 category.name = req.body.category;  
 category.save();  
 res.json({  
 success: true,  
 message: "Successful"  
 });  
 });  
  
  
 //Function to facilitate get request of specific categories  
 ***router***.get('/categories/:id', (req, res, next) => {  
 const perPage = 10;  
 const page = req.query.page;  
 async.parallel([  
 function(callback) {  
 ***Product***.count({ category: req.params.id }, (err, count) => {  
 var totalProducts = count;  
 callback(err, totalProducts);  
 });  
 },  
 function(callback) {  
 ***Product***.find({ category: req.params.id })  
 .skip(perPage \* page)  
 .limit(perPage)  
 .populate('category')  
 .populate('owner')  
 .populate('reviews')  
 .exec((err, products) => {  
 if(err) return next(err);  
 callback(err, products);  
 });  
 },  
 function(callback) {  
 ***Category***.findOne({ \_id: req.params.id }, (err, category) => {  
 callback(err, category)  
 });  
 }  
 ], function(err, results) {  
 var totalProducts = results[0];  
 var products = results[1];  
 var category = results[2];  
 res.json({  
 success: true,  
 message: 'category',  
 products: products,  
 categoryName: category.name,  
 totalProducts: totalProducts,  
 pages: ***Math***.ceil(totalProducts / perPage)  
 });  
 });  
   
 });  
  
 //Function to facilitate get request of specific product   
 ***router***.get('/product/:id', (req, res, next) => {  
 ***Product***.findById({ \_id: req.params.id })  
 .populate('category')  
 .populate('owner')  
 .deepPopulate('reviews.owner')  
 .exec((err, product) => {  
 if (err) {  
 res.json({  
 success: false,  
 message: 'Product is not found'  
 });  
 } else {  
 if (product) {  
 res.json({  
 success: true,  
 product: product  
 });  
 }  
 }  
 });  
 });  
  
  
 //Function to facilitate review functionality   
 ***router***.post('/review', checkJWT, (req, res, next) => {  
 async.waterfall([  
 function(callback) {  
 ***Product***.findOne({ \_id: req.body.productId}, (err, product) => {  
 if (product) {  
 callback(err, product);  
 }  
 });  
 },  
 function(product) {  
 let review = new ***Review***();  
 review.owner = req.decoded.user.\_id;  
  
 if (req.body.title) review.title = req.body.title;  
 if (req.body.description) review.description = req.body.description  
 review.rating = req.body.rating;  
  
 product.reviews.push(review.\_id);  
 product.save();  
 review.save();  
 res.json({  
 success: true,  
 message: "Successfully added the review"  
 });  
 }  
 ]);  
 });  
  
//Function to facilitate payment functionality using STRIPE API   
***router***.post('/payment', checkJWT, (req, res, next) => {  
 const stripeToken = req.body.stripeToken;  
 const currentCharges = ***Math***.round(req.body.totalPrice \* 100);  
  
 stripe.customers  
 .create({  
 source: stripeToken.id  
 })  
 .then(function(customer) {  
 return stripe.charges.create({  
 amount: currentCharges,  
 currency: 'usd',  
 customer: customer.id  
 });  
 })  
 .then(function(charge) {  
 const products = req.body.products;  
  
 let order = new ***Order***();  
 order.owner = req.decoded.user.\_id;  
 order.totalPrice = currentCharges / 100;  
   
 products.map(product => {  
 order.products.push({  
 product: product.product,  
 quantity: product.quantity  
 });  
 });  
  
 order.save();  
 res.json({  
 success: true,  
 message: "Successfully made a payment"  
 });  
 });  
});  
  
   
//Exporting the module   
module.exports = ***router***;

Product-search.js

//Using Algolia API in product searching  
  
//Including the required packages and assigning it to Local Variables  
const ***router*** = require('express').Router();  
const ***config*** = require('../config');  
  
  
const algoliasearch = require('algoliasearch');  
const client = algoliasearch(***config***.algolia\_app\_id, ***config***.algolia\_api\_key);  
const index = client.initIndex(***config***.algolia\_index);  
  
//Function providing product search functionality   
***router***.get('/', (req, res, next) => {  
 if (req.query.query) {  
 index.search({  
 query: req.query.query,  
 page: req.query.page,  
 }, (err, content) => {  
 res.json({  
 success: true,  
 message: "Here is your search",  
 status: 200,  
 content: content,  
 search\_result: req.query.query  
 });  
 });  
 }  
});  
  
//Exporting the module   
module.exports = ***router***;

Seller.js

// Seller.JS file to maintain every sellers details and storing the resources on AWS   
  
//Including the required packages and assigning it to Local Variables  
const ***router*** = require('express').Router();  
const ***Product*** = require('../models/product');  
const ***config*** = require('../config');  
  
  
const aws = require('aws-sdk');  
const multer = require('multer');  
const multerS3 = require('multer-s3');  
const s3 = new aws.S3({ accessKeyId: ***config***.access\_key, secretAccessKey: ***config***.access\_secret });  
  
const ***faker*** = require('faker');  
  
const checkJWT = require('../middlewares/check-jwt');  
  
//function to upload resources to AWS using multer service   
var upload = multer({  
 storage: multerS3({  
 s3: s3,  
 bucket: 'ebsproducts',  
 metadata: function (req, file, cb) {  
 cb(null, {fieldName: file.fieldname});  
 },  
 key: function (req, file, cb) {  
 cb(null, ***Date***.now().toString())  
 }  
 })  
});  
  
  
//Function to handle the product's GET and POST requests by seller   
***router***.route('/products')  
 .get(checkJWT, (req, res, next) => {  
 ***Product***.find({ owner: req.decoded.user.\_id })  
 .populate('owner')  
 .populate('category')  
 .exec((err, products) => {  
 if (products) {  
 res.json({  
 success: true,  
 message: "Products",  
 products: products  
 });  
 }  
 });  
 })  
 .post([checkJWT, upload.single('product\_picture')], (req, res, next) => {  
 ***console***.log("upload"+ upload);  
 ***console***.log(req.file);  
 let product = new ***Product***();  
 product.owner = req.decoded.user.\_id;  
 product.category = req.body.categoryId;  
 product.title = req.body.title;  
 product.price = req.body.price;  
 product.description = req.body.description;  
 product.image = req.file.location;  
 product.save();  
 res.json({  
 success: true,  
 message: 'Successfully Added the product'  
 });  
 });  
  
/\* Just for testing if products are added\*/  
***router***.get('/faker/test',(req, res, next) => {  
 for (i = 0; i < 15; i++) {  
 let product = new ***Product***();  
 product.category = "5acc1902580ba509c6622bd7";  
 product.owner = "5acbfed6571913c9a9e98135";  
 product.image = ***faker***.image.cats();  
 product.title = ***faker***.commerce.productName();  
 product.description = ***faker***.lorem.words();  
 product.price = ***faker***.commerce.price();  
 product.save();  
 }  
  
 res.json({  
 message: "Successfully added 20 pictures"  
 });  
  
});  
  
  
//Exporting the module   
module.exports = ***router***;

**Models:**

Product.js

//Product.JS to create Product Schema in the application   
  
//Including the required packages and assigning it to Local Variables  
const ***mongoose*** = require('mongoose');  
const Schema = ***mongoose***.Schema;  
const deepPopulate = require('mongoose-deep-populate')(***mongoose***);  
const mongooseAlgolia = require('mongoose-algolia');  
const ***config*** = require('../config');  
  
  
  
//Creating a new Product Schema  
const ProductSchema = new Schema({  
   
 category: { type: Schema.***Types***.ObjectId, ref: 'Category'},  
 owner: { type: Schema.***Types***.ObjectId, ref: 'User'},  
 reviews: [{ type: Schema.***Types***.ObjectId, ref: 'Review'}],  
 image: ***String***,  
 title: ***String***,  
 description: ***String***,  
 price: ***Number***,  
 crated: { type: ***Date***, default: ***Date***.now }  
}, {  
 toObject: { virtuals: true },  
 toJSON: { virtuals: true }  
});  
  
ProductSchema  
 .virtual('averageRating')  
 .get(function() {  
 let rating = 0;  
 if (this.reviews.length === 0) {  
 rating = 0;  
 } else {  
 this.reviews.map((review) => {  
 rating += review.rating;  
 });  
 rating = rating / this.reviews.length;  
 }  
 return rating;  
 });  
  
 //Adding Plug-ins to ProductSchema like Algolia to facilitate searching of products   
ProductSchema.plugin(deepPopulate); //Facilitate rating of the product   
ProductSchema.plugin(mongooseAlgolia, {  
 appId: ***config***.algolia\_app\_id,  
 apiKey: ***config***.algolia\_api\_key,  
 indexName: ***config***.algolia\_index,  
 selector: '\_id title image reviews description price owner created averageRating',  
 populate: {  
 path: 'owner reviews',  
 select: 'name rating'  
 },  
 defaults: {  
 author: 'unknown'  
 },  
 mappings: {  
 title: function(value) {  
 return `${value}`  
 }  
 },  
 debug: true  
})  
   
//Wrapping product schema to Model and synchronizing Algolia API   
let ***Model*** = ***mongoose***.model('Product', ProductSchema);  
***Model***.SyncToAlgolia();  
***Model***.SetAlgoliaSettings({  
 searchableAttributes: ['title']  
});  
  
//Exporting the wrapped Model(Algolia API + ProductSchema)  
module.exports = ***Model***

**Client Code:**

Index.html

<!doctype html>  
<html lang="en">  
<head>  
 <meta charset="utf-8">  
 <title>EBS</title>  
 <base href="/">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">  
 <script src="https://checkout.stripe.com/checkout.js"></script>  
 <link rel="icon" type="image/png" href="assets/img/logo.png">  
 <script>  
 (function(w, d)  
 { w.CollectId = "618d8efd11c7462f21dec6ef"; var h = d.head || d.getElementsByTagName("head")[0];  
 var s = d.createElement("script"); s.setAttribute("type", "text/javascript");  
 s.async=true; s.setAttribute("src", "https://collectcdn.com/launcher.js");  
 h.appendChild(s);  
 })(***window***, ***document***);  
 </script>  
</head>  
<body>  
 <app-root></app-root>  
</body>  
</html>

**Services:**

Rest\_api.service.ts

// rest-api.service.ts - Type script file to provide REST(GET,POST) Services in the e learning application  
  
// including required files and services  
import { ***Injectable*** } from '@angular/core';  
import { HttpClient, HttpHeaders } from '@angular/common/http';  
  
// exporting the RestAPi Service  
@Injectable()  
export class RestApiService {  
  
 constructor(private http: HttpClient) { }  
  
 getHeaders() {  
 const token = ***localStorage***.getItem('token');  
 return token ? new HttpHeaders().set('Authorization', token) : null;  
 }  
  
 get(link: string) {  
 return this.http.get(link, { headers: this.getHeaders() }).toPromise();  
 }  
  
 post(link: string, body: any) {  
 return this.http.post(link, body, { headers: this.getHeaders() }).toPromise();  
 }  
}

Data.service.ts

// DataService.ts - Type Script file to facilitate DataService to know type of message,handle cart functionality  
  
  
// including required modules and services  
import { ***Injectable*** } from '@angular/core';  
import { NavigationStart, Router } from '@angular/router';  
import { RestApiService } from './rest-api.service';  
  
  
// Exporting the DataService  
@Injectable()  
export class DataService {  
 message = '';  
 messageType = 'danger';  
  
 user: any;  
 cartItems = 0;  
  
 constructor(private router: Router, private rest: RestApiService) {  
 this.router.events.subscribe(event => {  
 if (event instanceof NavigationStart) {  
 this.message = '';  
 }  
 });  
 }  
  
 error(message) {  
 this.messageType = 'danger';  
 this.message = message;  
 }  
  
 success(message) {  
 this.messageType = 'success';  
 this.message = message;  
 }  
  
 warning(message) {  
 this.messageType = 'warning';  
 this.message = message;  
 }  
  
 async getProfile() {  
 try {  
 if (***localStorage***.getItem('token')) {  
 const data = await this.rest.get(  
 'http://localhost:3030/api/accounts/profile',  
 );  
 this.user = data['user'];  
 ***console***.log(this.user);  
 }  
 } catch (e) {  
 this.error(e);  
 }  
 }  
  
 getCart() {  
 const cart = ***localStorage***.getItem('cart');  
 return cart ? ***JSON***.parse(cart) : [];  
 }  
  
 addToCart(item: string) {  
 const cart: any = this.getCart();  
 if (cart.find(data => ***JSON***.stringify(data) === ***JSON***.stringify(item))) {  
 return false;  
 } else {  
 cart.push(item);  
 this.cartItems++;  
 ***localStorage***.setItem('cart', ***JSON***.stringify(cart));  
 return true;  
 }  
 }  
  
 removeFromCart(item: string) {  
 let cart: any = this.getCart();  
 if (cart.find(data => ***JSON***.stringify(data) === ***JSON***.stringify(item))) {  
 cart = cart.filter(data => ***JSON***.stringify(data) !== ***JSON***.stringify(item));  
 this.cartItems--;  
 ***localStorage***.setItem('cart', ***JSON***.stringify(cart));  
 }  
 }  
  
 clearCart() {  
 this.cartItems = 0;  
 ***localStorage***.setItem('cart', '[]');  
 }  
}

App.module.ts

// app.module.ts- TypeScript file which acts as a entry point to e learning client application ///  
// It contains app-routing module and several components //  
/////////////////////////////////////////////////////////////////////////////////////////////////  
  
// Including required modules and Services  
import { BrowserModule } from '@angular/platform-browser';  
import { ***NgModule*** } from '@angular/core';  
  
import { AppRoutingModule } from './app-routing.module';  
  
import { AppComponent } from './app.component';  
  
import { FormsModule } from '@angular/forms';  
import { HttpClientModule } from '@angular/common/http';  
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';  
  
import { RestApiService } from './rest-api.service';  
import { DataService } from './data.service';  
import { AuthGuardService } from './auth-guard.service';  
  
import { HomeComponent } from './home/home.component';  
import { MessageComponent } from './message/message.component';  
import { RegistrationComponent } from './registration/registration.component';  
import { LoginComponent } from './login/login.component';  
import { ProfileComponent } from './profile/profile.component';  
import { SettingsComponent } from './settings/settings.component';  
import { AddressComponent } from './address/address.component';  
import { CategoriesComponent } from './categories/categories.component';  
import { PostProductComponent } from './post-product/post-product.component';  
import { MyProductsComponent } from './my-products/my-products.component';  
import { CategoryComponent } from './category/category.component';  
import { ProductComponent } from './product/product.component';  
import { SearchComponent } from './search/search.component';  
import { CartComponent } from './cart/cart.component';  
import { MyordersComponent } from './myorders/myorders.component';  
import { OrderdetailsComponent } from './orderdetails/orderdetails.component';  
  
  
// Module decorator specifying all the components used in the application  
@NgModule({  
 declarations: [  
 AppComponent,  
 HomeComponent,  
 MessageComponent,  
 RegistrationComponent,  
 LoginComponent,  
 ProfileComponent,  
 SettingsComponent,  
 AddressComponent,  
 CategoriesComponent,  
 PostProductComponent,  
 MyProductsComponent,  
 CategoryComponent,  
 ProductComponent,  
 SearchComponent,  
 CartComponent,  
 MyordersComponent,  
 OrderdetailsComponent,  
 ],  
 imports: [  
 BrowserModule,  
 AppRoutingModule,  
 NgbModule.*forRoot*(),  
 FormsModule,  
 HttpClientModule,  
 ],  
 providers: [RestApiService, DataService, AuthGuardService],  
 bootstrap: [AppComponent],  
})  
// Exporting the AppModule  
export class AppModule {}

App-routing.module.ts

// app-routing.module.ts- Module which handles various routes in EBS client application ///  
  
// including required services and modules  
import { ***NgModule*** } from '@angular/core';  
import { Routes, RouterModule } from '@angular/router';  
  
import { HomeComponent } from './home/home.component';  
import { RegistrationComponent } from './registration/registration.component';  
import { LoginComponent } from './login/login.component';  
import { ProfileComponent } from './profile/profile.component';  
import { SettingsComponent } from './settings/settings.component';  
import { AddressComponent } from './address/address.component';  
import { CategoriesComponent } from './categories/categories.component';  
import { PostProductComponent } from './post-product/post-product.component';  
import { MyProductsComponent } from './my-products/my-products.component';  
import { CategoryComponent } from './category/category.component';  
import { ProductComponent } from './product/product.component';  
import { SearchComponent } from './search/search.component';  
import { CartComponent } from './cart/cart.component';  
import { MyordersComponent} from './myorders/myorders.component';  
import { OrderdetailsComponent} from './orderdetails/orderdetails.component';  
  
import { AuthGuardService } from './auth-guard.service';  
  
  
// assigning all possible routes to variable  
const routes: Routes = [  
 {  
 path: '',  
 component: HomeComponent,  
 },  
 {  
 path: 'search',  
 component: SearchComponent,  
 },  
 {  
 path: 'cart',  
 component: CartComponent,  
 },  
 {  
 path: 'categories',  
 component: CategoriesComponent,  
 },  
 {  
 path: 'categories/:id',  
 component: CategoryComponent,  
 },  
 {  
 path: 'product/:id',  
 component: ProductComponent,  
 },  
 {  
 path: 'orders/:id',  
 component: OrderdetailsComponent,  
 },  
 {  
 path: 'register',  
 component: RegistrationComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'login',  
 component: LoginComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile',  
 component: ProfileComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile/settings',  
 component: SettingsComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile/address',  
 component: AddressComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile/postproduct',  
 component: PostProductComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile/myproducts',  
 component: MyProductsComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: 'profile/orders',  
 component: MyordersComponent,  
 canActivate: [AuthGuardService],  
 },  
 {  
 path: '\*\*',  
 redirectTo: '',  
 },  
];  
  
// decorator to import and export routing Module in the application  
@NgModule({  
 imports: [RouterModule.*forRoot*(routes)],  
 exports: [RouterModule],  
})  
export class AppRoutingModule {}