GURU NANAK DEV UNIVERSITY

Project Report

B.Tech (Computer Science and Engineering)

Session 2017-2021

Title :- EasyShop

(Live URL - https://easyshop-04k3.onrender.com/)

Submitted by :-

Ayush Kaushik

2017CSA1042

Section – A

Supervised by :-

Dr. Prabhsimran Singh

Table of Content

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations I would like to extend sincere thanks to all of them.

I thank my God for providing me everything I required in completing this project.

I am highly indebted to my guide Dr. Prabhsimran Singh for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

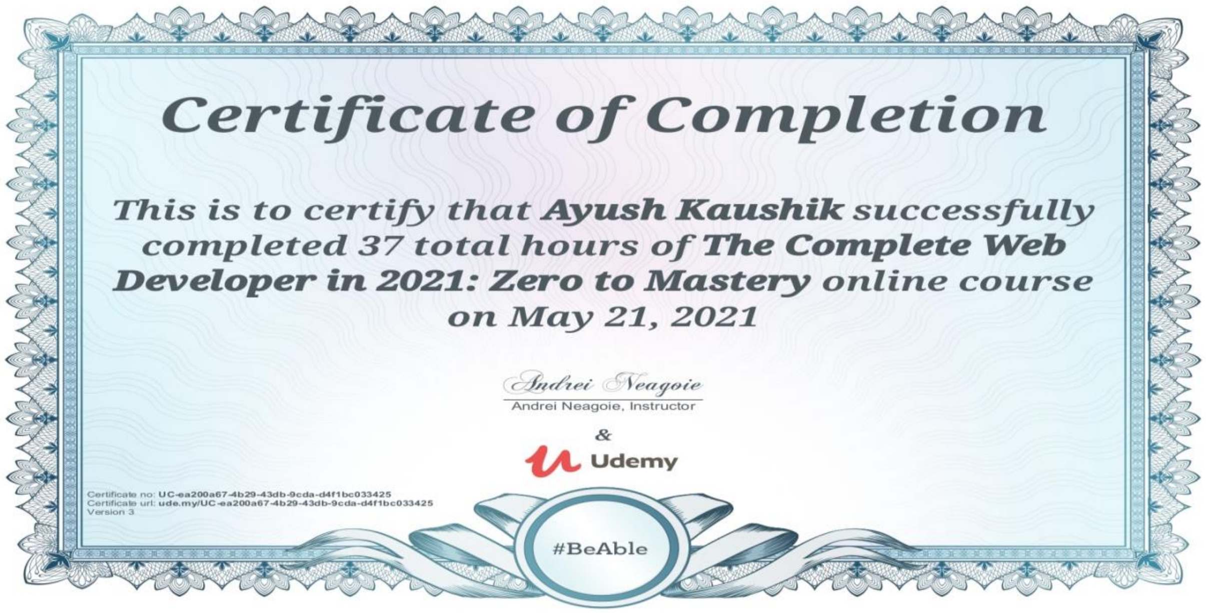
I would like to express gratitude towards my parent for their kind cooperation and encouragement which helped me in completing of this project.

I would like to express my special gratitude towards Coding blocks and Udemy for giving me such learning material and time.

My thanks and appreciation also goes to our Head of department Dr. Sandeep Sharma who helped us gain this opportunity to work on projects and gain some real working industrial skills.

Ayush Kaushik

Certificates



Certificate of Completion

This is to certify that **Ayush Kaushik** successfully completed 37 total hours of **The Complete Web Developer in 2021: Zero to Mastery** online course on May 21, 2021

Andrei Neagoie

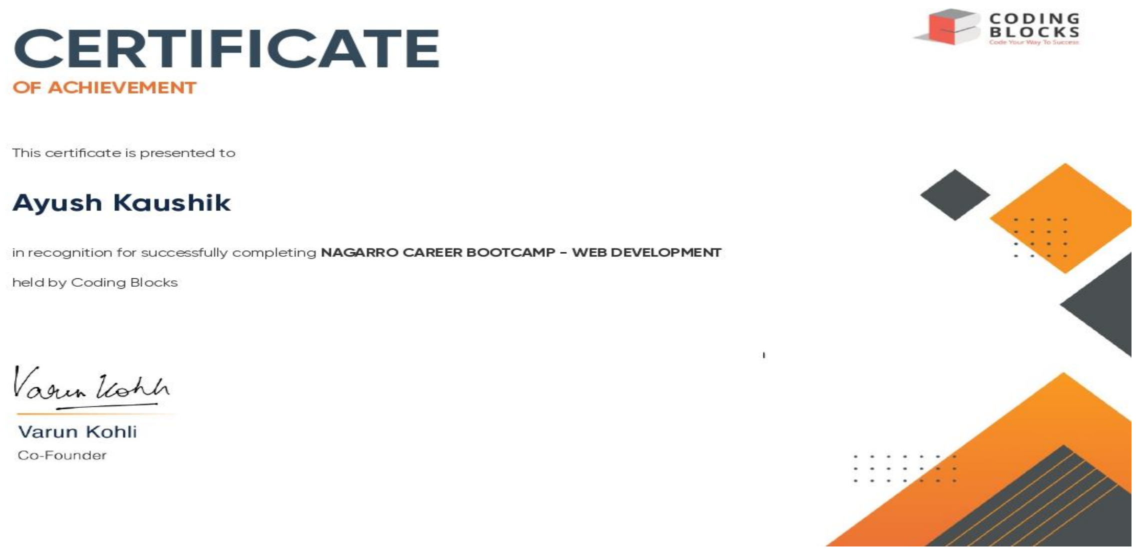
Andrei Neagoie, Instructor

&

Udemy

Certificate no: UC-ea200a67-4b29-43db-9cda-d4f1bc033425
Certificate url: ude.my/UC-ea200a67-4b29-43db-9cda-d4f1bc033425
Version 3

#BeAble



CODING BLOCKS

CERTIFICATE

OF ACHIEVEMENT

This certificate is presented to

Ayush Kaushik

in recognition for successfully completing **NAGARRO CAREER BOOTCAMP - WEB DEVELOPMENT**

held by Coding Blocks

Varun Kohli

Varun Kohli

Co-Founder

I am deeply honored by the knowledge and support I gained from Nagarro Coding Blocks Web Developer boot camp and Udemy which helped my boost my skills and leverage resources to work on this project.

Ayush Kaushik

3. Introduction To Project

As we know that today with the fussy messy busy schedule of ours we in general are neither getting enough time to buy things nor do we have lots and lots of time to choose the best product. So basically we are in a trap where time is directly proportional to the quality of product we receive.

Thus to solve this problem and to avoid wandering around those busy busting streets, I have decided to make an E-commercial website which would save efforts choosing right products within no time just through some scrolls and some clicks, a user can easily have the best product out in the market into their home. I know the major problem is the security, No need to worry, there will be a secure payment gateway to ensure there are no vulnerabilities with user's "most important information".

The user would be provided a sign-in page so that it would be easier to maintain records of their cart history.

They would be given an option to save products in a wish list. There would also be a rating option available for all products so that user could get to know about other person's buying experiences. So in a summary there would be all features one user can be comfortable with, along with :- Authentication, Payment gateways, Add or remove a product to or from the CART.

Please use following IDs for logging and testing as Retailer and Customer and Product published by retail to check CRUD operation.

Role	Login ID	Password	Product published
Retailer	xyz	xyz1234	Puma Shoe
Customer	test1	test1234	X

4. Key features

- A user can easily reach to their products within no span of time .
- A user would have an option to save the product in their cart for future purchase.
- A user would have the knowledge about products just by watching the reviews.
- Proper Authentication would be provided to the user so that connection between us and them is maintained.
- A secure Payment Gateway would be provided to the users to ensure their data privacy.
- A beautiful representation of products so that there is a clarity in vision of the user.

5. Hardware And Software Requirements

5.1. Hardware Requirements

- Disk Storage - A minimum of 500 GB of disk space would work fine
- Processor - Intel Core i5 (8th Gen) Processor
- RAM - 8 GB DDR4 RAM

5.2. Software Requirements

Technologies Used

1. Front-End

1. HTML5 (Hyper Text Markup Language)

   HTML5 is a markup language used for structuring and presenting content on the World Wide Web.

This is the latest version of HTML. HTML as well is not a programming language, it is a markup language. The audience for this is any software programmers or any persons around the world **HTML5** stands for Hypertext Markup Language version 5. World Wide Web Consortium (W3C) published it in October 2014. It is the most recent version of the language or code that explains web pages. HTML5 was created so as to enable various characteristics that the present day websites need. It is simple to adopt as there are no major alterations to the programming version of HTML.

The following are the very popular browsers that are compatible.

- Google Chrome 10.0 and higher
- Firefox 4.0 and higher
- Internet Explorer 8.0 and higher

2. CSS3 (Cascading Style Sheets)

It is a powerful tool for web designers to change the design and control over web pages that how it should be displayed. It is supported by all browsers and is designed primarily to separate the document content from document presentation.

CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each webpage.

CSS is easy to learn and understood but it provides powerful control over the presentation of an HTML document.

The following are the very popular browsers that are compatible.

- Google Chrome 10.0 and higher
- Firefox 4.0 and higher
- Internet Explorer 8.0 and higher

3. Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Here in my project I have used Bootstrap 5.

Bootstrap 5 alpha was officially released on June 16, 2020 after several months of refining. With all of the major changes in this version, the Boostrap 5 development team informed the users that the current version is still in alpha version thus, breaking changes will continue to occur until the first beta is released so it’s better to always check the open issues and pull requests on their official GitHub repository for open questions and feedback.

4. JavaScript

JavaScript is a very powerful **client-side scripting language**. JavaScript is used mainly for enhancing the interaction of a user with the webpage. In other words, you can make your webpage more lively and interactive, with the help of JavaScript. JavaScript is also being used widely in game development and Mobile application development.

Being a scripting language, JavaScript cannot run on its own. In fact, the browser is responsible for running JavaScript code**.** When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it.

The following are the very popular browsers that are compatible.

- Google Chrome 10.0 and higher
- Firefox 4.0 and higher
- Internet Explorer 8.0 and higher

2. Back End

1. Node.js

Node.js is an application runtime environment that allows you to write server-side applications in JavaScript.

Thanks to its unique I/O model, it excels at the sort of scalable and real-time situations we are increasingly demanding of our servers. It’s also lightweight, efficient, and its ability to use JavaScript on both frontend and backend opens new avenues for development.

It comes as no surprise that so many big companies have leveraged Node.js in production.

2. Express.js

Express.js is a free and open-source web application framework for Node.js. It is used for designing and building web applications quickly and easily. Web applications are web apps that you can run on a web browser. Since Express.js only requires JavaScript, it becomes easier for programmers and developers to build web applications and API without any effort.

Also Whenever we use MERN stack Express is best preferred for making and establishing servers.

3. Database(MongoDB)

In my project I have used Mongo DB as a Database. Mongo DB is an open-source document database built on a horizontal scale-out architecture. Founded in 2007, Mongo DB has a worldwide following in the developer community. Instead of storing data in tables of rows or columns like SQL databases, each row in a Mongo DB database is a document described in JSON, a formatting language.

4. Passport.js

Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more.

In my project I have used Passport.js for the Authorization purpose and in Passport.js I have used **passport.local** for connecting my database with authorization.Passport.js provides username and password attributes itself therefore I have not mentioned it in my database Schema.

6. Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multilevel DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO. That's why DFDs remain so popular after all these years.

Data Flow Diagram Notations

The two main types of notation used for data flow diagrams are Yourdon-Coad and Gane Sarson, both named after their creators, all experts who helped develop DFD methodology:

Ed Yourdon, Peter Coad, Chris Gane and Trish Sarson.

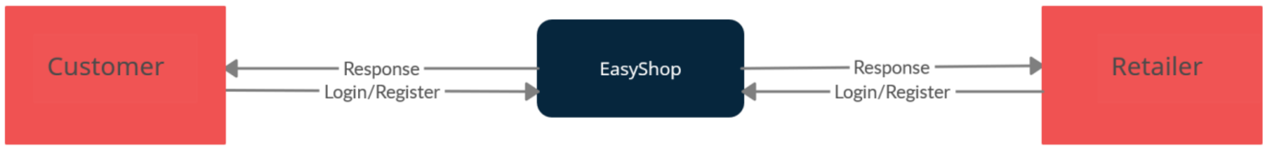
1) A square or rectangle defines the source or destination of system data.
2) An arrow identifies data flow - data in motion. It is a pipeline through which data flows.
3) A circle or a bubble represents a process that transforms incoming data flow(s) into outgoing data flow(s).
4) A rectangle marked with (D) is a data store – data at rest, or temporary repository of data.All data flow diagrams include four main elements: entity, process, data store and data flow.

External Entity – Also known as actors, sources or sinks, and terminators, external entities produce and consume data that flows between the entity and the system being diagrammed. These data flows are the inputs and outputs of the DFD. Since they are external to the system being analyzed, these entities are typically placed at the boundaries of the diagram. They can represent another system or indicate a subsystem.

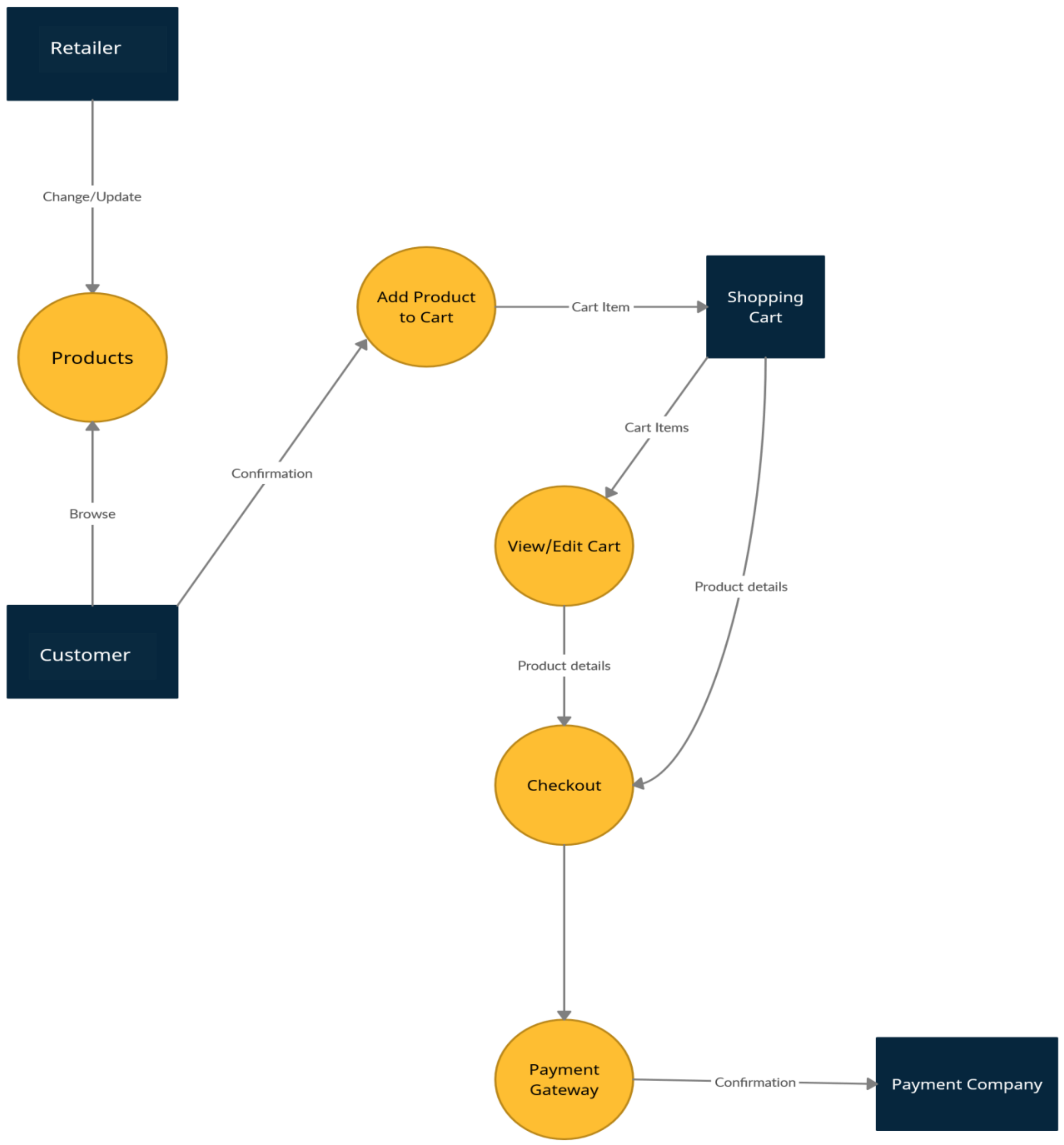
Process – An activity that changes or transforms data flows. Since they transform incoming data to outgoing data, all processes must have inputs and outputs on a DFD. In GaneSarson notation, a rectangular box is used and may be labeled with a reference number ,location of where in the system the process occurs and a short title that describes its function. Processes are typically oriented from top to bottom and left to right on a data flow diagram.

Data Store – A data store does not generate any operations but simply holds data for later access. Data stores could consist of files held long term or a batch of documents stored briefly while they wait to be processed. Input flows to a data store include information or operations that change the stored data. Output flows would be data retrieved from the Store. **Data Flow** – Movement of data between external entities, processes and data stores is represented with an arrow symbol, which indicates the direction of flow. This data could be electronic, written or verbal. Input and output data flows are labeled based on the type of data or its associated process or data store, and this name is written alongside the arrow

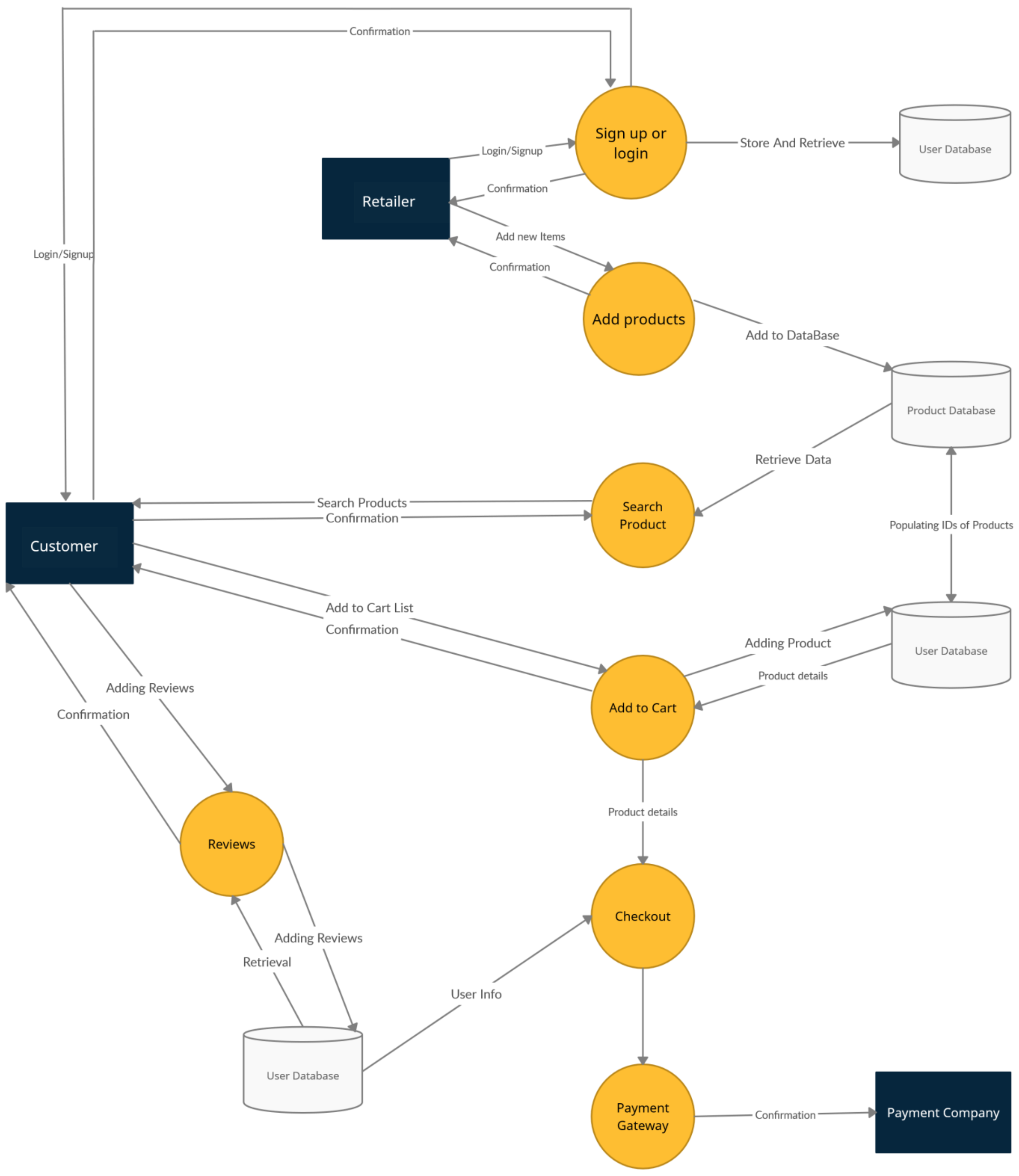
DFD Level 0



DFD Level 1

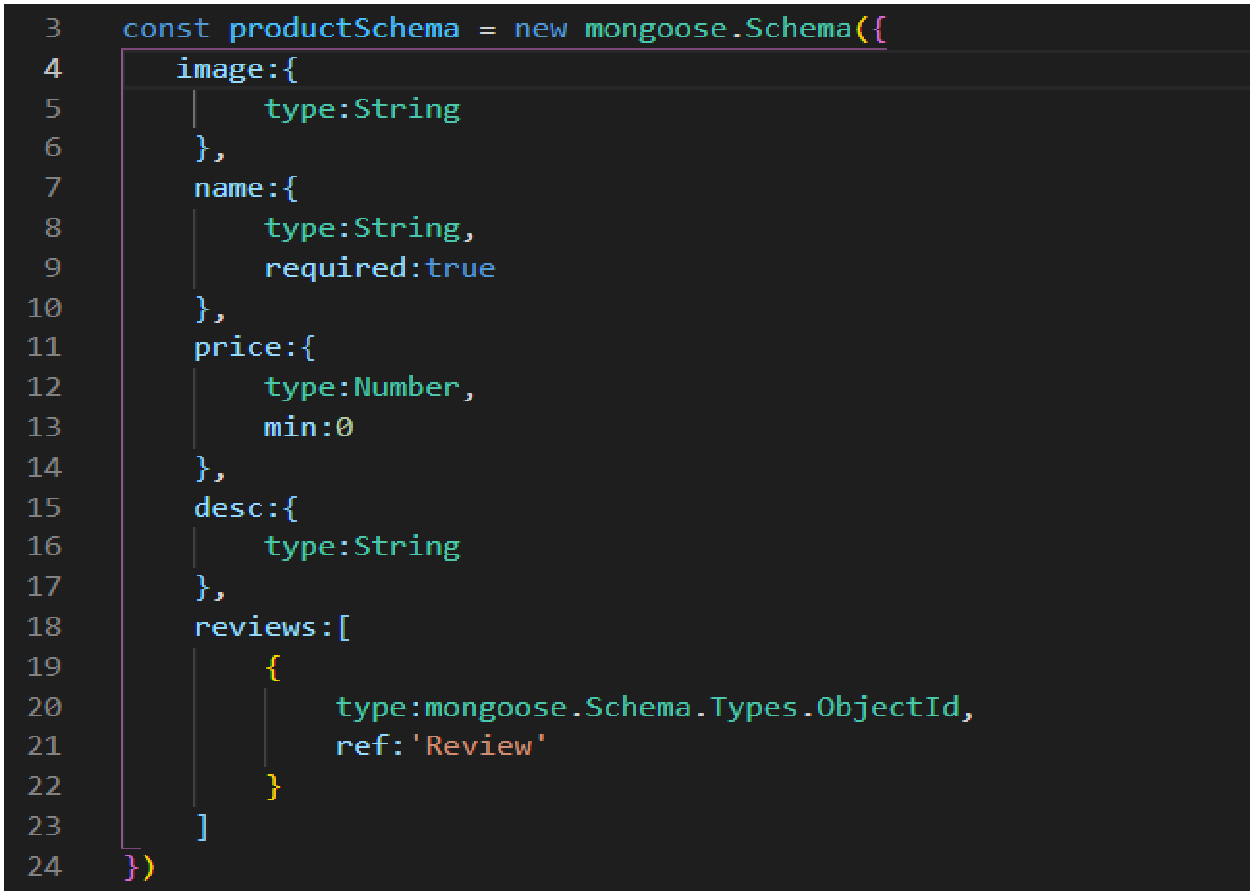

Retailer
Change/Update
Products
Browse
Customer
Confirmation
Add Product to Cart
Cart Item
Shopping Cart
Cart Items
View/Edit Cart
Product details
Product details
Checkout
Payment Gateway
Confirmation
Payment Company

DFD Level 2


Confirmation
Login/Signup
Sign up or login
Store And Retrieve
User Database
Retailer
Login/Signup
Confirmation
Add new Items
Confirmation
Add products
Add to DataBase
Product Database
Retrieve Data
Populating IDs of Products
Customer
Search Products
Confirmation
Search Product
Add to Cart List
Confirmation
Adding Product
Product details
User Database
Add to Cart
Adding Reviews
Confirmation
Reviews
Product details
Checkout
Adding Reviews
Retrieval
User Info
User Database
Payment Gateway
Confirmation
Payment Company

7.DataBase Schema

productSchema : Contains information about products

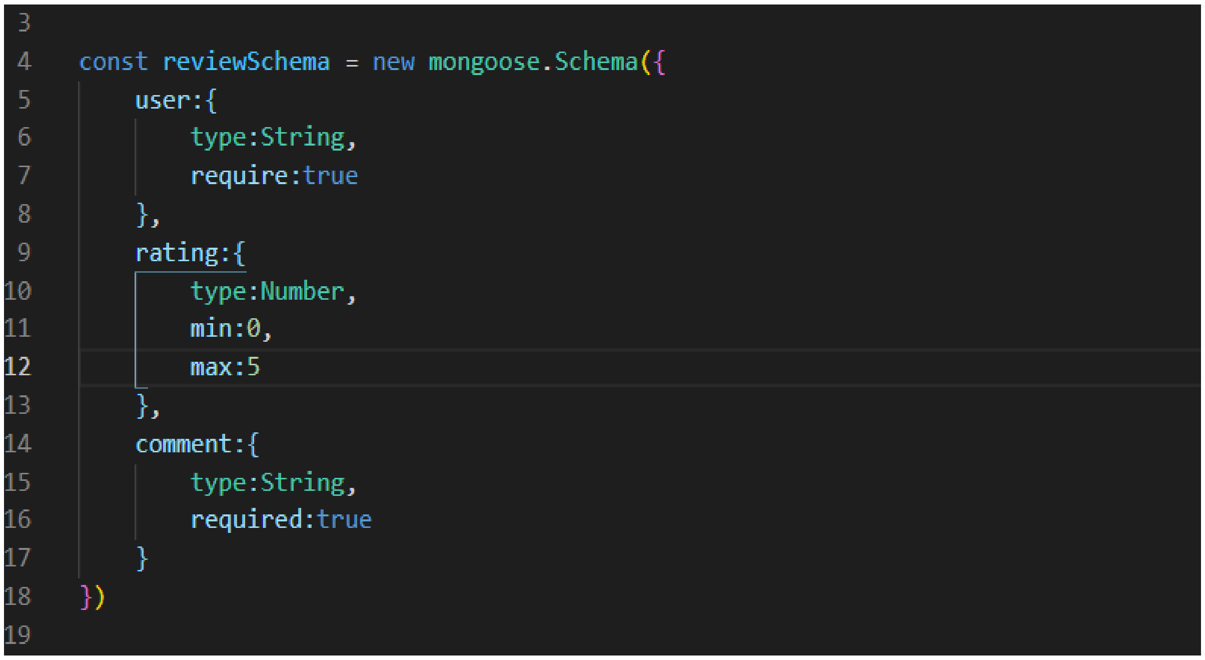


```
const productSchema = new mongoose.Schema({
    image:{
        type:String
    },
    name:{
        type:String,
        required:true
    },
    price:{
        type:Number,
        min:0
    },
    desc:{
        type:String
    },
    reviews:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Review'
        }
    ]
})
```

The productSchema has all the information about products along with a reviews array that stores objectId to store rating and reviews in it.

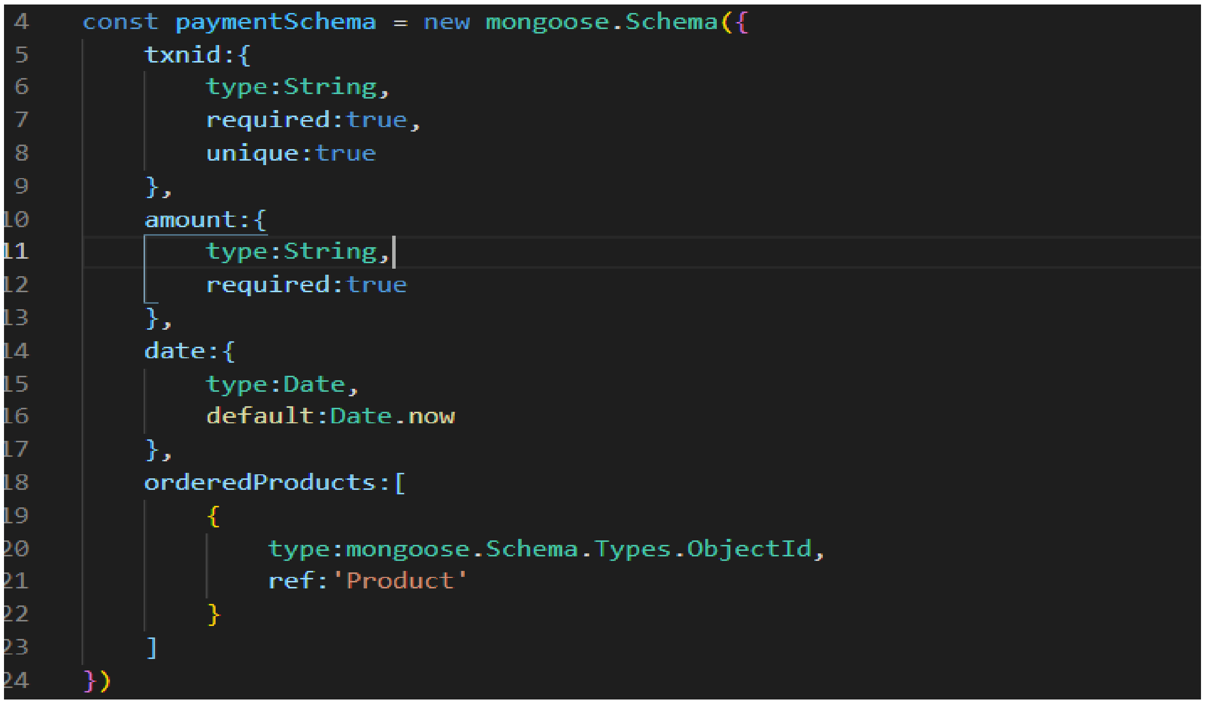
reviewSchema to store rating and reviews on each product by different customers

The reviewSchema stores rating and comments on each product which later on gets transferred to array of reviews in productSchema via ObjectId.



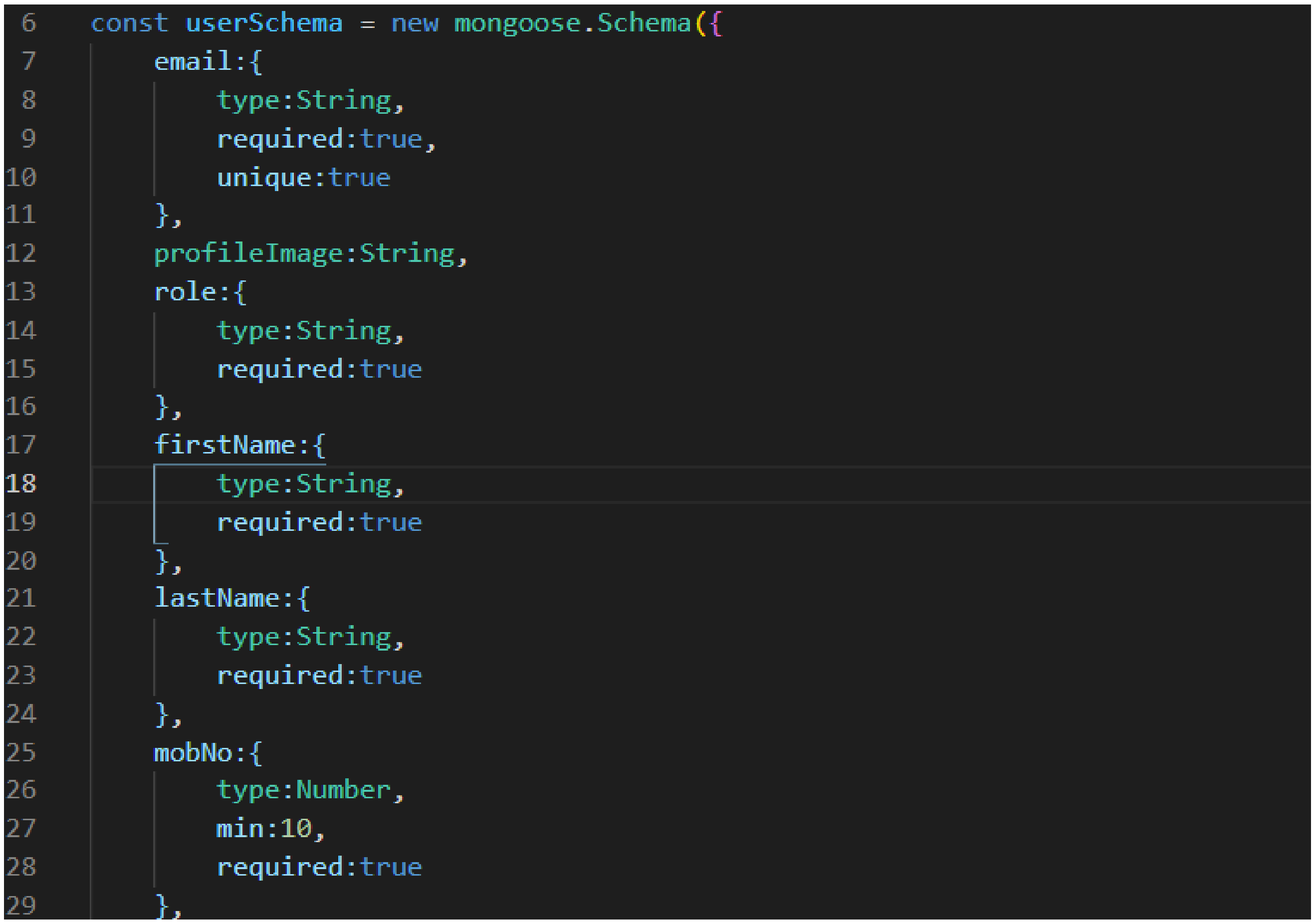
```
const reviewSchema = new mongoose.Schema({
    user:{
        type:String,
        require:true
    },
    rating:{
        type:Number,
        min:0,
        max:5
    },
    comment:{
        type:String,
        required:true
    }
})
```

paymentSchema: The paymentSchema holds the ObjectId of the products from product database along with transaction Id , amount, date of order.



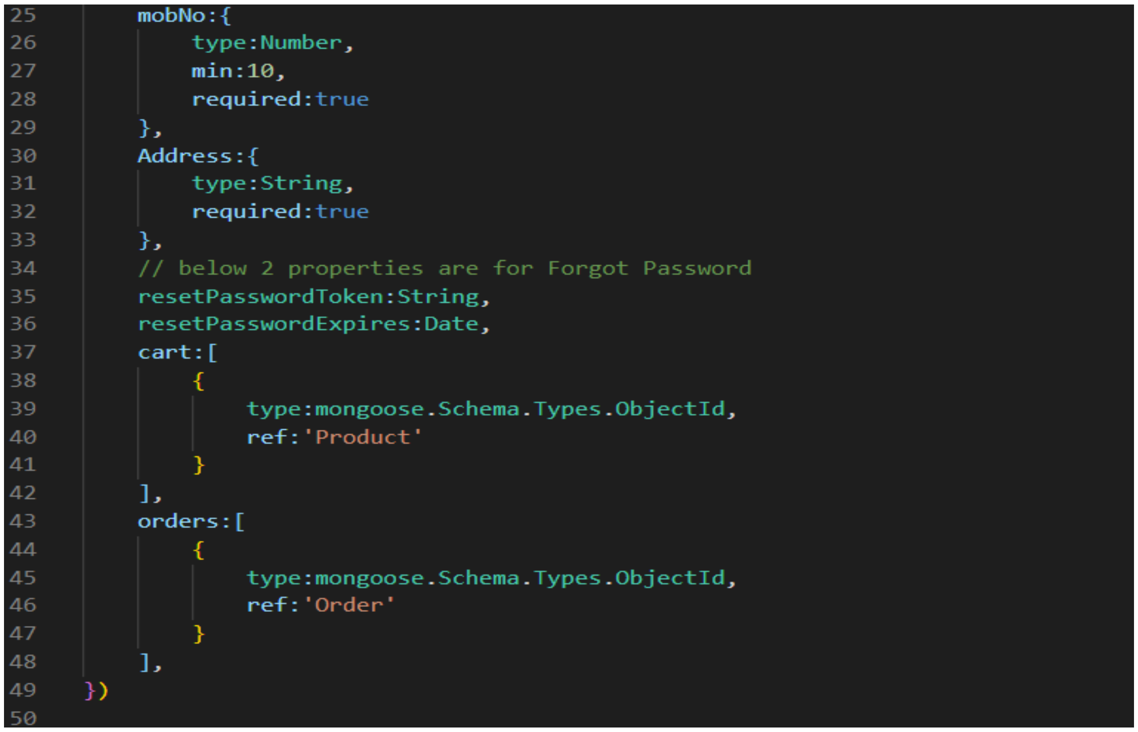
```
const paymentSchema = new mongoose.Schema({
    txnid:{
        type:String,
        required:true,
        unique:true
    },
    amount:{
        type:String,
        required:true
    },
    date:{
        type:Date,
        default:Date.now
    },
    orderedProducts:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Product'
        }
    ]
})
```

userSchema has all information about product although username and password are being provided in it by **Passport.js** the userSchema has an array of cart and payments which stores object Id of products and payement information in it.



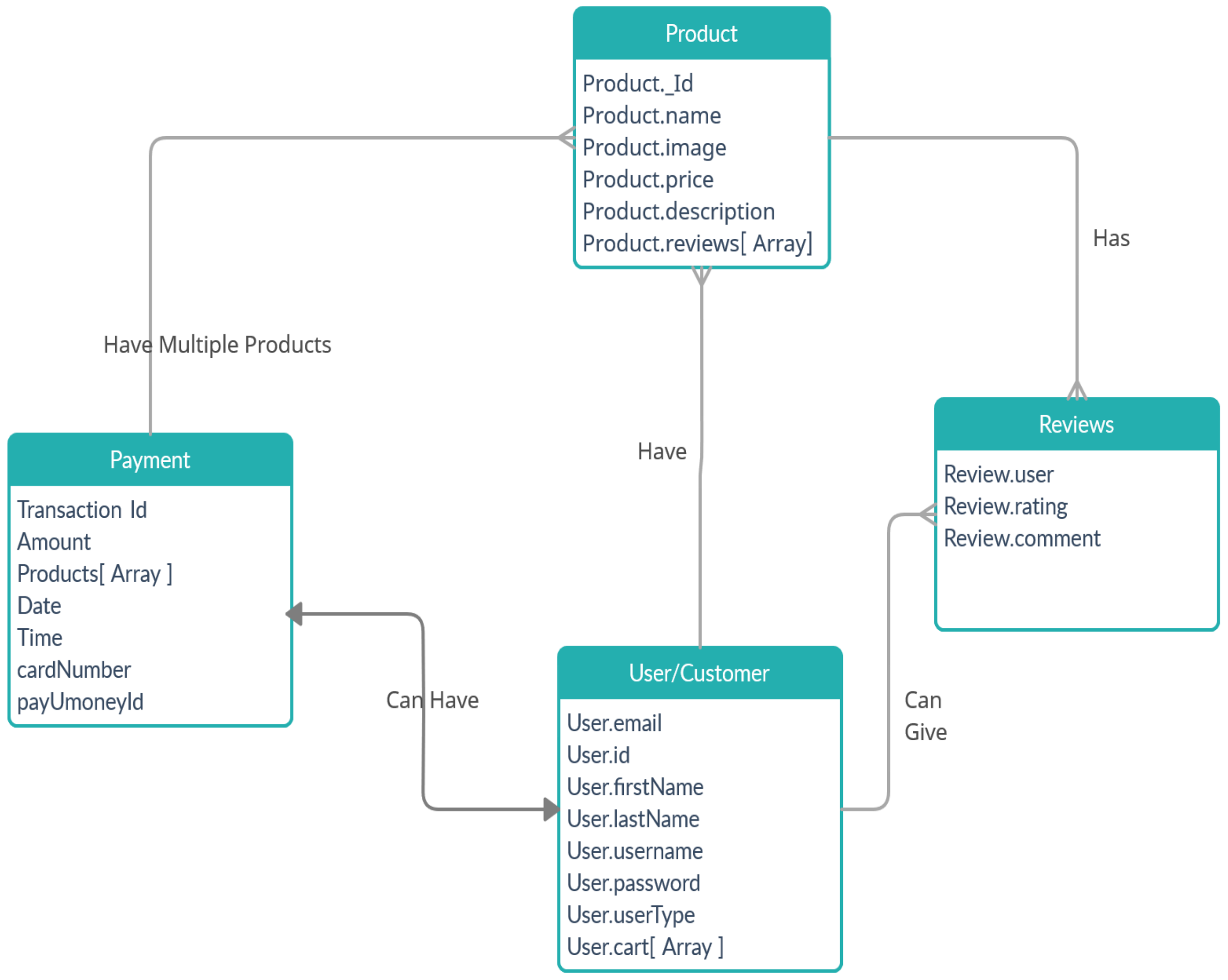
```
const userSchema = new mongoose.Schema({
    email:{
        type:String,
        required:true,
        unique:true
    },
    profileImage:String,
    role:{
        type:String,
        required:true
    },
    firstName:{
        type:String,
        required:true
    },
    lastName:{
        type:String,
        required:true
    },
    mobNo:{
        type:Number,
        min:10,
        required:true
    },
```

userSchema continued ….



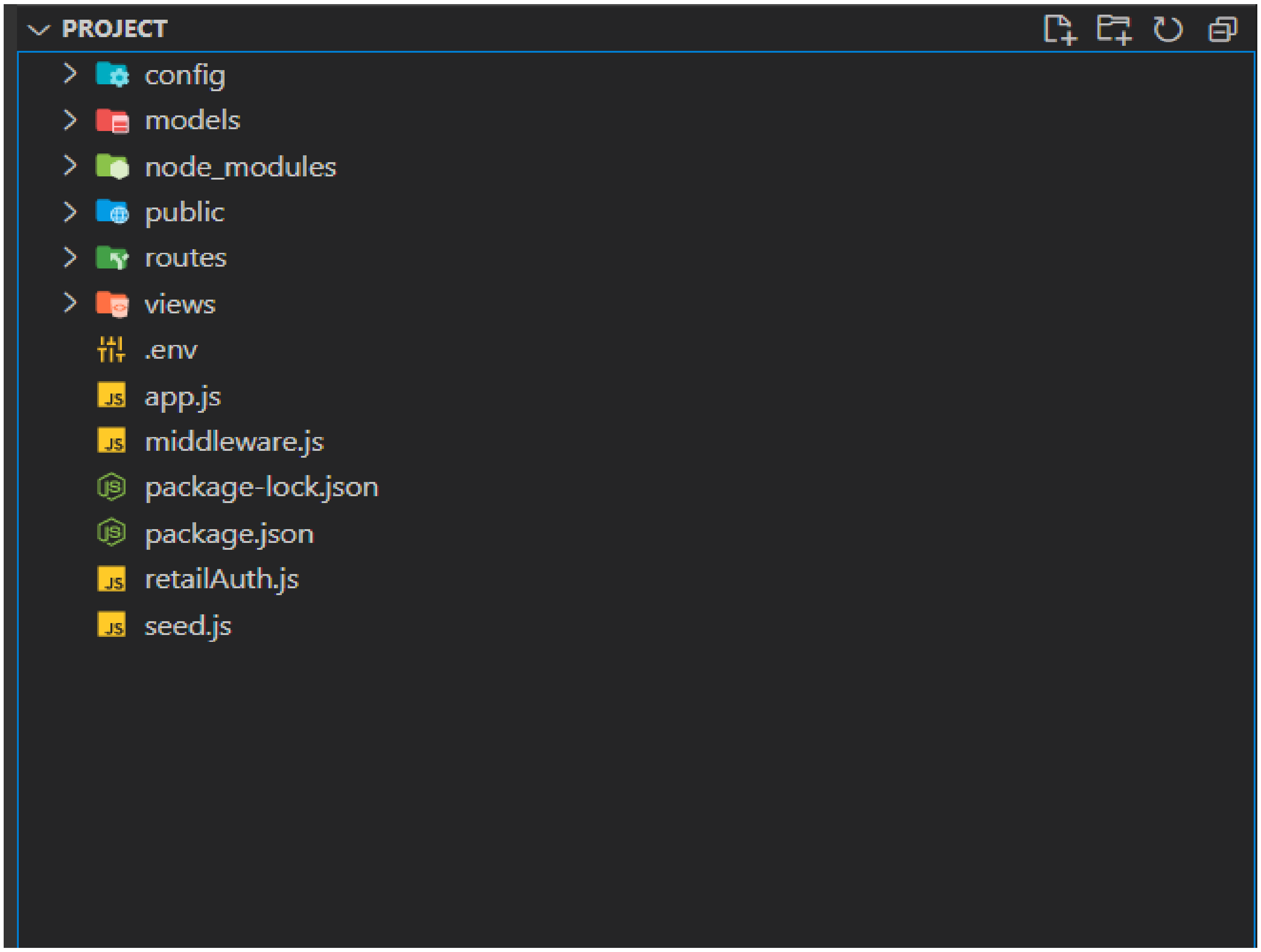
```
    mobNo:{
        type:Number,
        min:10,
        required:true
    },
    Address:{
        type:String,
        required:true
    },
    // below 2 properties are for Forgot Password
    resetPasswordToken:String,
    resetPasswordExpires:Date,
    cart:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Product'
        }
    ],
    orders:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Order'
        }
    ],
})
```

7. ER Diagram


Product
Product._Id
Product.name
Product.image
Product.price
Product.description
Product.reviews[Array]
Has
Have Multiple Products
Have
Reviews
Review.user
Review.rating
Review.comment
Payment
Transaction Id
Amount
Products[Array]
Date
Time
cardNumber
payUmoneyId
Can Have
User/Customer
User.email
User.id
User.firstName
User.lastName
User.username
User.password
User.userType
User.cart[Array]
Can
Give

8.Project Detail with Implementation Snapshots

Folder/File Structure



```
PROJECT
> config
> models
> node_modules
> public
> routes
> views
  .env
  app.js
  middleware.js
  package-lock.json
  package.json
  retailAuth.js
  seed.js
```

Backend files.

Main File app.js(Source code)

```
const express = require("express");
const app = express();
const mongoose = require("mongoose");
const path = require("path");
const seedDB = require('./seed')
const methodOverride = require('method-override')
const session = require('express-session')
const flash = require('connect-flash')
const passport = require('passport')
const LocalStrategy = require('passport-local')
const User = require('./models/user')

const dotenv = require('dotenv')
dotenv.config()

const productRoutes = require('./routes/product')
const authRoutes = require('./routes/auth')
const cartRoutes = require('./routes/cart')
const fpRoutes = require('./routes/forgotpassword')
const paymentRoutes = require('./routes/payment')
const userRoutes = require('./routes/user')

mongoose.connect("mongodb://localhost:27017/shopApp", {
    useNewUrlParser: true,
    useUnifiedTopology: true,
    useFindAndModify:false,
    useCreateIndex:true
  })
  .then(() => {
    console.log("DB connected");
  })
  .catch((error) => {
    console.log("Oh no! ERROR!!");
    console.log(error);
  });

  // seedDB()

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "/views"));
app.use(express.static(path.join(__dirname,'/public')))
app.use(express.urlencoded({extended:true}))
app.use(methodOverride('_method'))

const sessionConfig = {
  secret:'weneedsomebettersecret',
  resave:false,
  saveUninitialized:true
}
app.use(session(sessionConfig))
app.use(flash())

app.use(passport.initialize())
app.use(passport.session())

passport.use(new LocalStrategy(User.authenticate()))

passport.serializeUser(User.serializeUser())
passport.deserializeUser(User.deserializeUser())

app.use((req,res,next)=>{
  res.locals.success = req.flash('success')
  res.locals.error = req.flash('error')
  res.locals.currentUser = req.user
  next()
})


app.get("/", (req, res) => {
  res.render('home');
});

app.use(productRoutes)
app.use(authRoutes)
app.use(cartRoutes)
app.use(fpRoutes)
app.use(paymentRoutes);
app.use(userRoutes)

app.listen(3000, () => {
  console.log("Server started at PORT:3000");
});
```

Seed File (seed.js)

```
const mongoose = require('mongoose')
const Product = require('./models/product')

const products = [
    {
        image:'https://store.storeimages.cdn-apple.com/4982/as-
images.apple.com/is/iphone-12-pro-graphite-hero?wid=940&hei=1112&fmt=png-
alpha&qlt=80&.v=1604021660000',
        name:'Iphone 12',
        price:100000,
        desc:'The iPhone 12 is a dual-
SIM (GSM and GSM) mobile in which you can use both Nano-
SIM and eSIM cards. The smartphone runs iOS 14 and has an inbuilt storage of 64GB
 so that you can store all your local files, songs, videos, pictures, docs and mo
re without any space constraints. In addition to this, the smartphone from Apple
houses a non-
removable battery that supports both wireless and proprietary fast charging.Speak
ing about the camera specifications, the iPhone 12 on the rear features a 12 MP p
rimary camera with f/1.6 aperture and a 12 MP camera with f/2.4 aperture. Also, t
he rear setup has autofocus feature. While on the front, the mobile comes with a
12 MP camera with f/2.2 aperture for clicking some awesome selfies.On iPhone 12,
you get various connectivity options such as Wi-
Fi 802.11 a/b/g/n/ac/Yes, Lightning, Bluetooth v5.00, NFC, GPS, 3G, and 4G (with
support for Band 40 used by some LTE networks in India). Sensors on the smartphon
e include proximity sensor, ambient light sensor, barometer, accelerometer, gyros
cope, and compass/ magnetometer. Also, there is face unlock with 3D face recognit
ion for security purpose.'
    },
    {
        image:'https://store.storeimages.cdn-apple.com/4668/as-
images.apple.com/is/MX8E2_VW_34FR+watch-44-alum-spacegray-nc-
nike6s_VW_34FR_WF_CO_GEO_IN?wid=750&hei=712&trim=1,0&fmt=p-
jpg&qlt=80&.v=1599017871000,1601923999000',
        name:'Apple watch',
        price:60000,
        desc:"Apple Watch comes with an included band (strap) to attach it to the
 user's wrist, which can be easily changed to other types. Third party bands are
compatible with Apple Watch, however Apple produces bands in a variety of materia
ls and colours which are updated each season. The most recent update to the colou
rs occurred in March 2020. Bands designed for the 38mm and 42mm cases are complet
ely compatible with the 40mm and 44mm cases respectively.Starting with Apple Watc
h Series 5, Apple introduced the Apple Watch Studio which allows customers to mix
```

```
 and match bands, eliminating the need to purchase a specific combination and all
ows for a simplification of packaging (since Series 4)."

    },
    {
        image:'https://store.storeimages.cdn-apple.com/4982/as-
images.apple.com/is/macbook-air-gold-select-
201810?wid=892&hei=820&&qlt=80&.v=1603332211000',
        name:'Macbook Air',
        price:200000,
        desc:"The MacBook Air, based on Intel processors, was introduced in Janua
ry 2008 with a 13.3-
inch screen, and was promoted as the world's thinnest notebook, opening a laptop
category known as the ultrabook family. Apple released a second generation MacBoo
k Air in October 2010, with a redesigned tapered chassis, standard solid-
state storage, and added a smaller 11.6-
inch version. Later revisions added Intel Core i5 or i7 processors and Thunderbol
t.[4] The third generation was released in October 2018, with reduced dimensions,
 a Retina display, and combination USB-
C/Thunderbolt 3 ports for data and power. An updated model was released in Februa
ry 2020 with the Magic Keyboard and an option for an Intel Core i7 processor."
    },
    {
        image:'https://store.storeimages.cdn-apple.com/4668/as-
images.apple.com/is/MWP22_AV1?wid=1144&hei=1144&fmt=jpeg&qlt=80&.v=1591634652000'
,
        name:'Apple buds pro',
        price:16000,
        desc:"AirPods Pro were released on October 30, 2019 as a premium option c
ompared to AirPods. They use the same H1 chip found in second generation AirPods,
 and boast a slimmer design, active noise cancellation, adaptive EQ, IPX4 water r
esistance, a new charging case with Qi standard, and include silicone tips"
    },
    {
        image:'https://images-na.ssl-images-
amazon.com/images/I/81g7AiqWrtL._AC_SL1500_.jpg',
        name:'Acer Predator',
        price:126000,
        desc:"Acer Predator is a gamer-
focused brand and line of computer hardware owned by Acer. In 2008, Acer introduc
ed itself in the gaming computer market with a line of desktop computers: the Ace
r Aspire Predator series, later renamed as Acer Predator. The series is character
```

```
ized by the futuristic computer chassis and high performance. In 2016, a complete
 range of Predator desktops, gaming notebooks, tablets and accessories exists"
    },
    {
        image:'https://i.gadgets360cdn.com/products/large/realme-watch-670x800-
1590388807.jpg',
        name:'Realme SmartWatch',
        price:3600,
        desc:"Featuring a large 3.5 cm (1.4) touchscreen, this smartwatch from re
alme is a must-
have accessory. It will help you keep track of your fitness endeavours with its s
et of features such as heart rate monitor, 14 sports modes, and blood oxygen leve
l monitor. Oh, you needn’t take your phone out for everything, as this smartwatch
 supports notifications for texts, calls, and more."
    },
    {
        image:'https://images.puma.com/image/upload/f_auto,q_auto,b_rgb:fafafa,w_
2000,h_2000/global/374765/02/sv01/fnd/IND/fmt/png/Rebound-JOY-SoftFoam+-Shoes',
        name:'PUMA Shoe',
        price:8000,
        desc:"Here’s a basketball-
inspired shoe that’s going to change the game. A dash of retro inspiration combin
es with PUMA’s DNA to create a shoe that looks good both on the court and off. Fe
aturing an upper made with synthetic leather and TPU, it’s got the flexibility yo
u want and, thanks to the SoftFoam+ sockliner, it’s going to give you the support
 you need. A final touch is the perforations, which give this shoe a trendy, more
 aerodynamic look."
    },
    {
        image:'https://5.imimg.com/data5/VS/LV/MY-47202023/tourist-royal-blue-
bag-500x500.jpg',
        name:'Tourist Bag',
        price:1600,
        desc:"Renowned amid one of the credible and eminent business names, we ar
e actively committed in presenting a world class Tourist Royal Blue Bag."
    },
    {
        image:'https://assets.myntassets.com/fl_progressive/h_960,q_80,w_720/v1/a
ssets/images/2365523/2018/2/9/11518161462196-US-Polo-Assn-Men-White-Solid-Polo-
Collar-T-shirt-5091518161462058-1.jpg',
        name:'U.S. POLO t-shirt',
        price:1600,
        desc:"White solid polo T-
shirt, has a polo collar, short sleevesThe basic tee, after all, is the simplest,
 easiest piece of clothing imaginable—its blank-
page quality functions like a screen on which we project our current cultural pre
occupations. A T-shirt can denote working-
class status (if, for example, you’re the Boss)"
    },
    {
        image:'https://cdn.shopify.com/s/files/1/1676/7297/products/Main-
Image_5dd17660-d566-4297-bc2e-e43de833b2fc.jpg?v=1613028178',
        name:'Leaf Headphone',
        price:16000,
        desc:"Leaf Bass’s ergonomic designs helps you cut out the noise around yo
u when you’re using them. They come with noise isolation which allows the headpho
nes to cancel out the ambient noise around you to almost 95% which will help you
zone into your favorite songs. Leaf Bass has ultra-
soft cushion ear cups and headband cushion for superior comfort also ensuring you
r ears don’t experience pain when you put them on for long hours. It’s designed p
erfectly for long listening hours to give you all the comfort required while jamm
ing."
    }


]

const seedDB = async ()=>{
    await Product.insertMany(products)
    console.log('DB SEEDED')
}

module.exports = seedDB
```

Middlewares

isLoggedIn.js

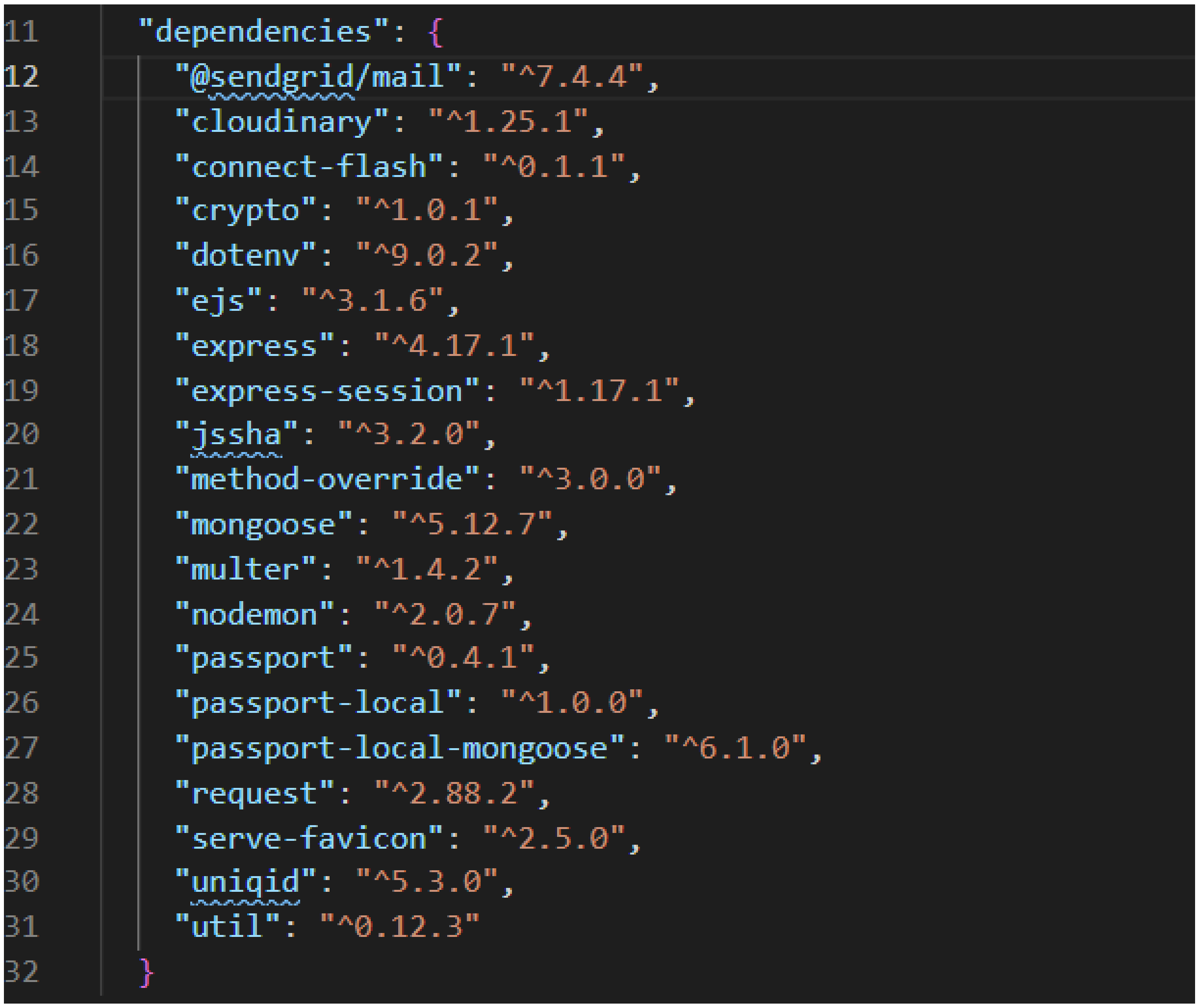
```
const isLoggedIn = (req,res,next) =>{
    if(!req.isAuthenticated()){
        req.flash('error','Please Login First !!')
        return res.redirect('/login')
    }
    next()
}


module.exports = {
    isLoggedIn
}
```

retailAuth.js

```
const User = require('./models/user')



const authRole = (req,res,next)=>{
    if(req.user.role !== 'Retailer'){
        req.flash('error','You are not Authorized')
       return res.redirect('/login')
    }
     next()

}

module.exports = {
    authRole
}
```

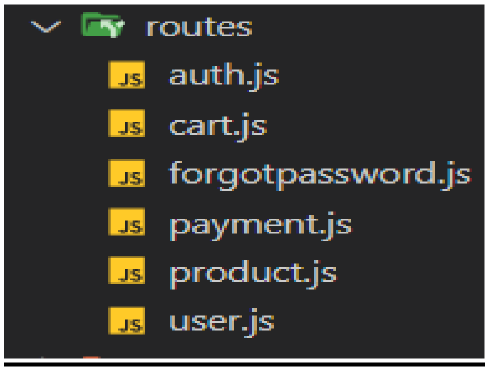
package.json

dependencies



```
"dependencies": {
  "@sendgrid/mail": "^7.4.4",
  "cloudinary": "^1.25.1",
  "connect-flash": "^0.1.1",
  "crypto": "^1.0.1",
  "dotenv": "^9.0.2",
  "ejs": "^3.1.6",
  "express": "^4.17.1",
  "express-session": "^1.17.1",
  "jssha": "^3.2.0",
  "method-override": "^3.0.0",
  "mongoose": "^5.12.7",
  "multer": "^1.4.2",
  "nodemon": "^2.0.7",
  "passport": "^0.4.1",
  "passport-local": "^1.0.0",
  "passport-local-mongoose": "^6.1.0",
  "request": "^2.88.2",
  "serve-favicon": "^2.5.0",
  "uniqid": "^5.3.0",
  "util": "^0.12.3"
}
```

<u>Routes files</u>



```
routes
  auth.js
  cart.js
  forgotpassword.js
  payment.js
  product.js
  user.js
```

Auth.js file

```
const express = require('express')
const passport = require('passport')
const router = express.Router()
const User = require('../models/user')
const cloudinary = require('../config/cloudinaryConfig')
const upload = require('../config/multerConfig')

const path = require('path')




router.get('/register',async(req,res)=>{
    res.render('auth/signup')
})

router.post('/register',upload.single('profileImage'),async(req,res)=>{
    try{

        let imageurl;
        if (! req.file || ! req.file.path) {
            imageurl = 'https://bellfund.ca/wp-content/uploads/2018/03/demo-
user.jpg'
        }else{
        const result = await cloudinary.uploader.upload(req.file.path)
        imageurl = result.secure_url
        }

        const user = new User({
            username:req.body.username,
            profileImage:imageurl,
            email:req.body.email,
            role:req.body.role,
            firstName:req.body.firstName,
            lastName:req.body.lastName,
            mobNo:req.body.mobNo,
            Address:req.body.Address
        })
        const newUser = await User.register(user,req.body.password)
        await user.save()
        req.flash('success','Registered Successfully')
        res.redirect('/login')
}
    catch(e){
        req.flash('error',e.message)
        res.redirect('/register')
    }
})

router.get('/login',(req,res)=>{

  res.render('auth/login')
})

router.post('/login',
passport.authenticate('local',
{
    failureRedirect:'/login',
    failureFlash:true

}),
(req,res)=>{
    req.flash('success',`Hi ${req.user.username}! Welcome Back`)
    res.redirect('/products')
})

router.get('/logout',(req,res)=>{
    if(!req.user){
        req.flash('error','Please Log In first!!!')
        res.redirect('/login')
    }else{
        req.logout()
        req.flash('success','Logged Out Successfully')
        res.redirect('/login')
    }
})



module.exports = router
```

cart.js

```
const express = require('express')
const router = express.Router()
const {isLoggedIn} = require('../middleware')
const Product = require('../models/product')
const User = require('../models/user')


router.get('/user/:userId/cart',isLoggedIn,async(req,res)=>{
    try{
        const user = await User.findById(req.params.userId).populate('cart')

        res.render('cart/showCart',{userCart:user.cart})
    }
    catch(e){
        req.flash('error',"Item couldn't be Added :(")
        res.render('error')
    }

})


router.post('/user/:id/cart',isLoggedIn,async(req,res)=>{
    try{
        const product = await Product.findById(req.params.id)
        const user = req.user
        user.cart.push(product)
        user.save()
        req.flash('success','Added to Cart')
        res.redirect(`/user/${req.user._id}/cart`)
    }
    catch(e){
        req.flash('error','Cannot View Cart Items')
        res.render('error')
    }

})


router.delete('/user/:userId/cart/:id',async(req,res)=>{
    const {userId,id} = req.params
    await User.findByIdAndUpdate(userId,{$pull:{cart:id}})
    res.redirect(`/user/${req.user._id}/cart`)
})
```

```
router.get('/cart/payment', (req, res) => {
    res.render('payment/payment')
})




module.exports = router
```

Forgotpassword.js

```
const express = require('express')
const router = express.Router()
const User = require('../models/user')
const passport = require('passport')
const sgMail = require('@sendgrid/mail')
const util = require('util')
const crypto = require('crypto')


sgMail.setApiKey(process.env.SGMAIL_APIKEY)


router.get('/forgotpassword',(req,res)=>{
    res.render('forgot')
})


router.put('/forgotpassword',async(req,res)=>{
    const token = await crypto.randomBytes(20).toString('hex')
    const {email} = req.body
    const user = await User.findOne({email:email})
    //Check if email-id exists in database
    if(!user){
        console.log('User doesnt exists')
        req.flash('error','You are not registered')
        return res.redirect('/forgotpassword')
    }
    //if exists then put the token and expire time in database
    user.resetPasswordToken = token
    user.resetPasswordExpires = Date.now() + 3600000
    await user.save()
    const msg = {
        to: email,
        from: 'easyshop303@gmail.com',
        subject: 'Reset Password',
        text: `You are receiving this because you (or someone else) have requeste
d the reset of the password for your account. Please click on the following link,
 or paste this into your browser to complete the process:
        http://${req.headers.host}/reset/${token}
        If you did not request this, please ignore this email and
        your password will remain unchanged.`.replace(/    /g, ''),


        html: `You are receiving this because you (or someone else) have requeste
d the reset of the password for your account. Please click on the following link,
 or paste this into your browser to complete the process:
        http://${req.headers.host}/reset/${token}
        If you did not request this, please ignore this email and
        your password will remain unchanged.`,
    }
    await sgMail.send(msg)
    console.log('An Email has sent to '+email)
    res.redirect('/forgotpassword')
})

router.get('/reset/:token',async(req,res)=>{
    const token = req.params.token
    const user = await User.findOne({
        resetPasswordToken:token,
        resetPasswordExpires:{$gt:Date.now()}
    })
    //if user doesn't exist or the token has been expired
    if(!user){
        console.log("Password Reset Token Has Expired")
        req.flash('error','Password Reset Token Has Expired')
        return res.redirect('/forgotpassword')
    }

    //if token is valid then show reset password page
    res.render('reset',{token})
})
```

```
router.put('/reset/:token',async(req,res)=>{

    const token = req.params.token
    const user = await User.findOne({
        resetPasswordToken:token,
        resetPasswordExpires:{$gt:Date.now()}
    })

    //if user doesn't exist or the token has been expired
    if(!user){
        console.log("Password Reset Token Has Expired")
        req.flash('error','Password Reset Token Has Expired')
        return res.redirect('/forgotpassword')
    }
    //If everything goes fine then check the password and confirm fields
    if(req.body.password === req.body.confirm){
        //update the new password
        await user.setPassword(req.body.password)
        //set token to null
        user.resetPasswordToken = null
        user.resetPasswordExpires = null
        //save the user and login
        await user.save()
        const login = util.promisify(req.login.bind(req))
        await login(user)
    }//if password doesn't match ask them to enter again
    else{
        console.log('Passwords does not match')
        req.flash('error','Passwords does not match')
        return res.redirect(`/reset/${token}`)
    }
    res.redirect('/products')
})


module.exports = router
```

payment.js

```
const express = require('express');
const router = express.Router();
const request = require('request');
const jsSHA = require('jssha');
const uniqid = require('uniqid');
const { isLoggedIn } = require('../middleware');
const User = require('../models/user');
const Order = require('../models/order');

router.post('/payment_gateway/payumoney',isLoggedIn,(req,res)=>{
    req.body.txnid = uniqid.process()
    req.body.email = req.user.email
    req.body.firstname = req.user.firstName

    const pay = req.body


    const hashString = process.env.MERCHANT_KEY //store in in different file
     + '|' + pay.txnid
     + '|' + pay.amount
     + '|' + pay.productinfo
     + '|' + pay.firstname
     + '|' + pay.email
     + '|' + '||||||||||'
     + process.env.MERCHANT_SALT //store in in different file


    const sha = new jsSHA('SHA-512', "TEXT");
    sha.update(hashString);
    //Getting hashed value from sha module
     const hash = sha.getHash("HEX");

     //We have to additionally pass merchant key to API

     pay.key = process.env.MERCHANT_KEY //store in in different file;
     pay.surl = 'http://localhost:3000/payment/success';
     pay.furl = 'http://localhost:3000/payment/fail';
     pay.hash = hash;

     //Making an http/https call with request
     request.post({
        headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json'
        },
        url: 'https://sandboxsecure.payu.in/_payment', //Testing url
        form: pay
     }, function (error, httpRes, body) {
        if (error)
            res.send(
                {
                    status: false,
                    message:error.toString()
                }
     );
     if (httpRes.statusCode === 200) {
        res.send(body);
     } else if (httpRes.statusCode >= 300 &&
            httpRes.statusCode <= 400) {
            res.redirect(httpRes.headers.location.toString());

     }
     })
})



router.post('/payment/success',isLoggedIn, async(req, res) => {
    //Payumoney will send Success Transaction data to req body.
    //  Based on the response Implement UI as per you want

    try {
        const order= {
            txnid: req.body.txnid,
            amount: req.body.amount,
            orderedProducts:req.user.cart
        }

        const placedOrder=await Order.create(order);

        req.user.orders.push(placedOrder);
        await req.user.save();

        req.flash('success','Your Order has been Successfully Placed.Thanks for S
hopping with us!')
        res.redirect(`/user/${req.user._id}/me`);
    }
    catch (e) {
        console.log(e.message);
        req.flash('error', 'Cannot Place the Order at this moment.Please try agai
n later!');
        res.render('error');
    }
})

router.post('/payment/fail',isLoggedIn, (req, res) => {
    //Payumoney will send Fail Transaction data to req body.
    //  Based on the response Implement UI as per you want
    req.flash('error', `Your Payment Failed.Try again after sometime`);
    res.render('error');
})


module.exports = router;
```

product.js

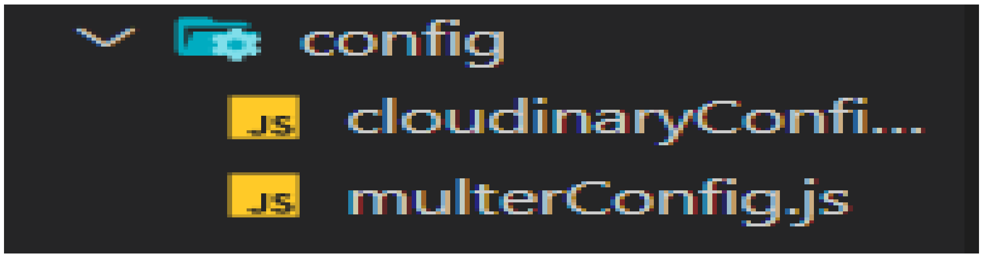
```
const express = require('express')
const router = express.Router()
const Product = require('../models/product')
const Review = require('../models/review')
const {isLoggedIn} = require('../middleware')
const {authRole} = require('../retailAuth')
const favicon = require('serve-favicon')
router.get('/products',async (req,res)=>{

    try{
        const products = await Product.find({})
        res.render('products/index',{products})
    }
    catch(e){
        console.log("Something Went Wrong")
        req.flash('error','Cannot find Products')
        res.redirect('/error')
    }
})

// Get the form for new Product
router.get('/products/new',isLoggedIn,authRole,(req,res)=>{
    res.render('products/new')
```

```
})

// Create New Product

router.post('/products',isLoggedIn,authRole, async(req,res)=>{
    try{
        const {product} = req.body
        await Product.create(product)
        req.flash('success','Product Created Successfully')
        res.redirect('/products')
    }
    catch(e){
        console.log(e.message)
        req.flash('error',"Couldn't Create a new Product!")

        res.redirect('error')
    }
})

router.get('/products/:id',async(req,res)=>{
    try{
        const product = await Product.findById(req.params.id).populate('reviews')
        res.render('products/show',{product})
    }
    catch(e){
        console.log(e.message)
        req.flash('error',"Couldn't find that product :(")
        res.redirect('/error')
    }
})

router.get('/products/:id/edit',isLoggedIn,authRole,async(req,res)=>{
    const product = await Product.findById(req.params.id)
    res.render('products/edit',{product})
})

router.patch('/products/:id',isLoggedIn,authRole,async(req,res)=>{
    try{
        const {product} = req.body
        await Product.findByIdAndUpdate(req.params.id,product)
        req.flash('success','Product Updated!')
        res.redirect(`/products/${req.params.id}`)
    }
    catch(e){
        console.log(e.message)
        req.flash('error',"Product Updation failed!")
        res.redirect('error')
    }
})

router.delete('/products/:id',isLoggedIn,authRole,async(req,res)=>{
    await Product.findByIdAndRemove(req.params.id)
    res.redirect('/products')
})

router.get('/error',(req,res)=>{
    res.status(404).render('error')
})



// Creating a new comment on a Product

router.post('/products/:id/review',isLoggedIn,async(req,res)=>{
    try{
        const product = await Product.findById(req.params.id)
        const review = new Review({
            user:req.user.username,
            ...req.body
        })

        product.reviews.push(review)

        await review.save()
        await product.save()
        req.flash('success','Review added Successfully!')
        res.redirect(`/products/${req.params.id}`)
    }
    catch(e){
        console.log(e.message)
        req.flash('error','Cannot Add Review To This Product!!')
        res.redirect('/error')
    }
})

router.delete('/products/:id/review/:reviewId',isLoggedIn,async(req,res)=>{
    const {id,reviewId} = req.params
    await Product.findByIdAndUpdate(id,{$pull:{review:reviewId}})
    await Review.findByIdAndDelete(reviewId)
    res.redirect(`/products/${id}`)
})


module.exports = router
```

User.js

```
const express = require('express');
const router = express.Router();
const { isLoggedIn } = require('../middleware');
const User = require('../models/user');
const Product = require('../models/product');


router.get('/user/:id/me',isLoggedIn,async(req,res)=>{
    try{
        const userInfo = await User.findById(req.params.id).populate({
            path:'orders',
            populate:{
                path:'orderedProducts',
                model:'Product'
            }
        })

        res.render('user/user',{orders:userInfo.orders})
    }
    catch(e){
        console.log(e.message);
        req.flash('error', 'Cannot Place the Order at this moment.Please try agai
n later!');
        res.render('error');
    }
})

module.exports = router
```

Config



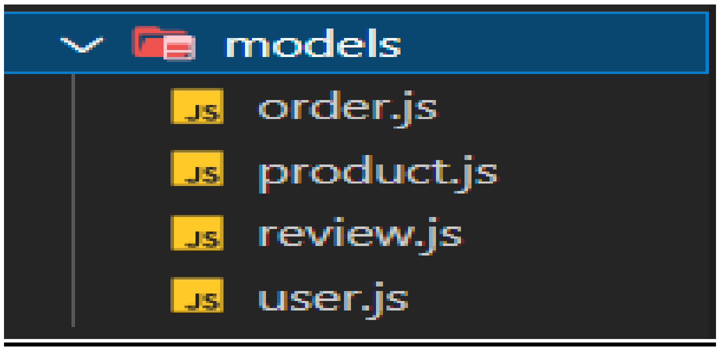
cloudinaryConfig.js

```
const cloudinary = require('cloudinary').v2
 cloudinary.config({
    cloud_name: process.env.CLOUDINARY_CLOUD_NAME,
    api_key: process.env.CLOUDINARY_API_KEY,
    api_secret: process.env.CLOUDINARY_API_SECRET,
});


module.exports = cloudinary
```

multerConfig.js

```
const multer = require('multer')
const path = require('path')

module.exports = multer({
    storage:multer.diskStorage({}),
    fileFilter:(req,file,cb) => {
        let ext = path.extname(file.originalname)
        if(ext !== '.jpg' && ext !== '.jpeg' && ext !== '.png'){
            cb(new Error('File Type Not Supported'), false)
            return;
        }
        cb(null,true)
    }
})
```

Models



Order.js

```
const mongoose = require('mongoose')
const Product = require('./product')

const paymentSchema = new mongoose.Schema({
    txnid:{
        type:String,
        required:true,
        unique:true
    },
    amount:{
        type:String,
        required:true
    },
    date:{
        type:Date,
        default:Date.now
    },
    orderedProducts:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Product'
        }
    ]
})

const Order = mongoose.model('Order',paymentSchema)

module.exports = Order
```

Product.js

```
const mongoose = require('mongoose')
const Review = require('./review')
const productSchema = new mongoose.Schema({
   image:{
        type:String
    },
    name:{
        type:String,
        required:true
    },
    price:{
        type:Number,
        min:0
    },
    desc:{
        type:String
    },
    reviews:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Review'
        }
    ]
})

const Product = mongoose.model('Product',productSchema)
module.exports = Product
```

review.js

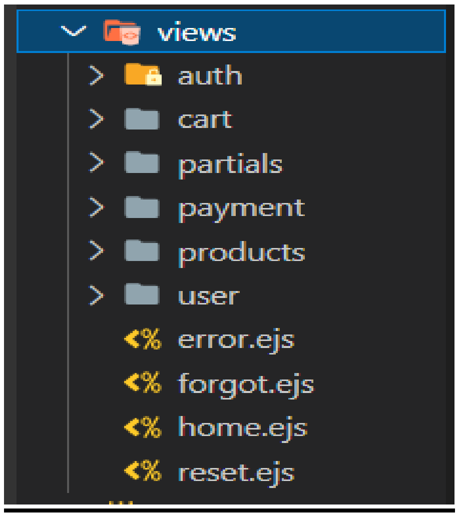
```
const mongoose = require('mongoose')


const reviewSchema = new mongoose.Schema({
    user:{
        type:String,
        require:true
    },
    rating:{
        type:Number,
        min:0,
        max:5
    },
    comment:{
        type:String,
        required:true
    }
})

const Review = mongoose.model('Review',reviewSchema)

module.exports = Review
```

user.js

```
const mongoose = require('mongoose')
const passportLocalMongoose = require('passport-local-mongoose')
const Product = require('./product')
const Order = require('./order')

const userSchema = new mongoose.Schema({
    email:{
        type:String,
        required:true,
        unique:true
    },
    profileImage:String,
    role:{
        type:String,
        required:true
    },
    firstName:{
        type:String,
        required:true
    },
    lastName:{
        type:String,
        required:true
    },
    mobNo:{
        type:Number,
        min:10,
        required:true
```

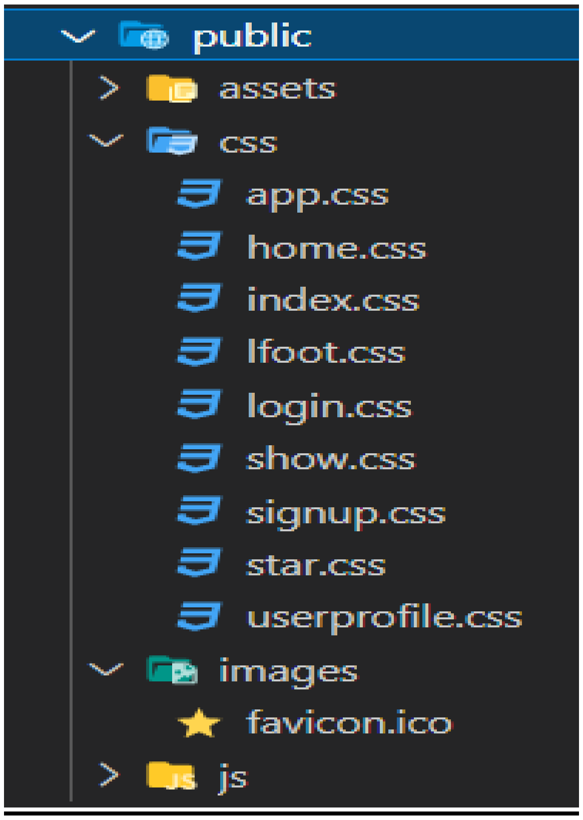
```
    },
    Address:{
        type:String,
        required:true
    },
    // below 2 properties are for Forgot Password
    resetPasswordToken:String,
    resetPasswordExpires:Date,
    cart:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Product'
        }
    ],
    orders:[
        {
            type:mongoose.Schema.Types.ObjectId,
            ref:'Order'
        }
    ],
})

userSchema.plugin(passportLocalMongoose)

const User = mongoose.model('User',userSchema)
module.exports = User
```

Views folder (EJS templating engine used)



```
views
  auth
  cart
  partials
  payment
  products
  user
  error.ejs
  forgot.ejs
  home.ejs
  reset.ejs
```

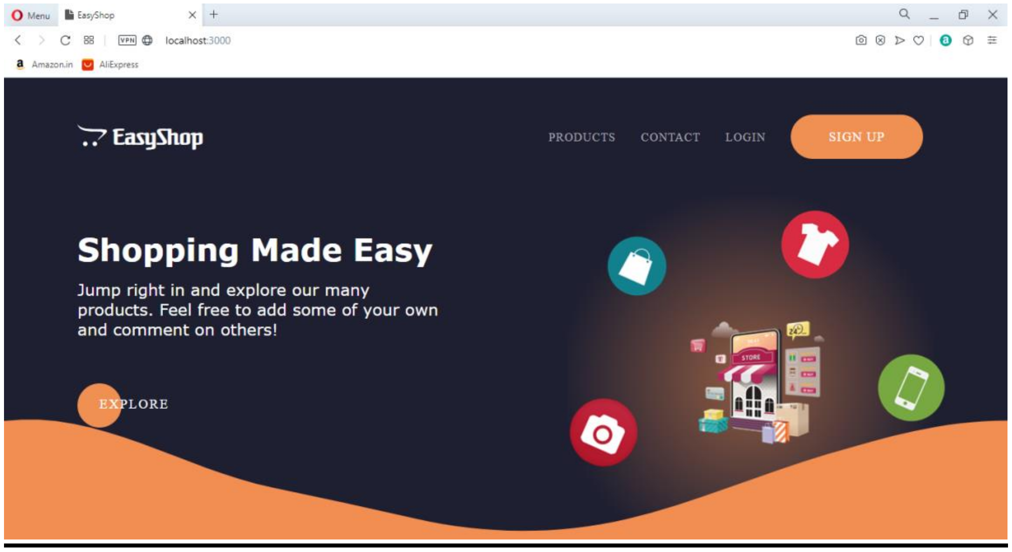
Public folder



```
public
  > assets
  v css
      app.css
      home.css
      index.css
      lfoot.css
      login.css
      show.css
      signup.css
      star.css
      userprofile.css
  v images
      favicon.ico
  > js
```

Front End

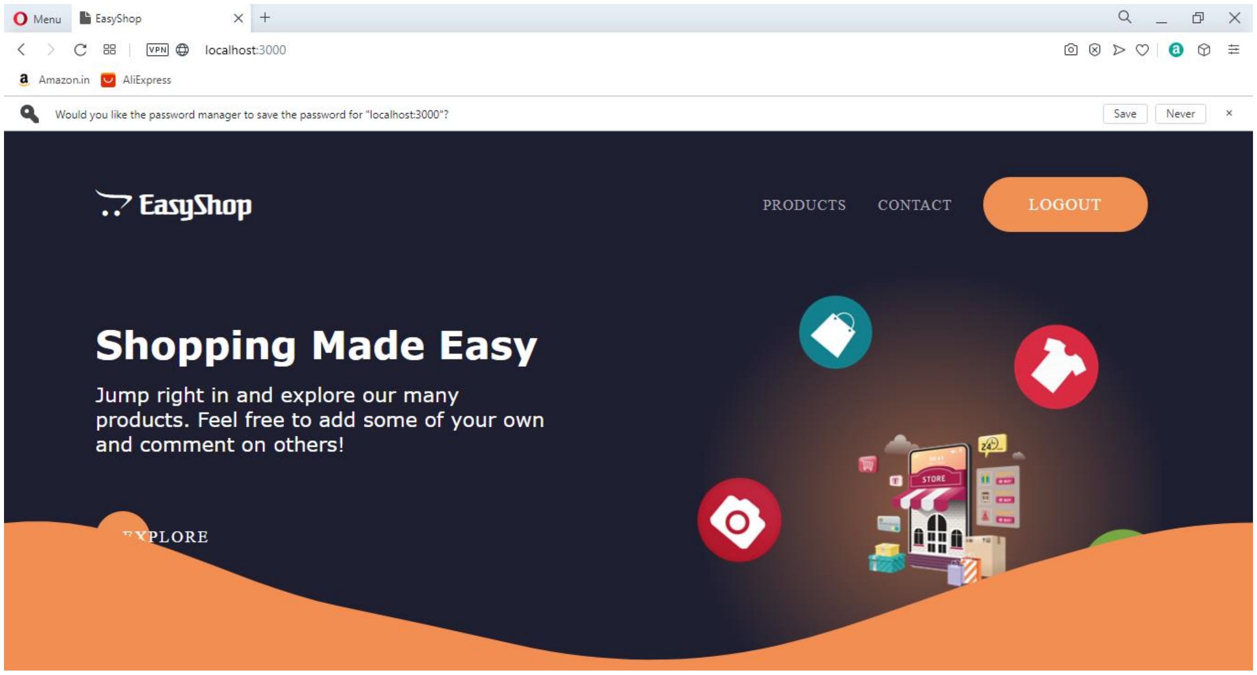
Home Page (without login)



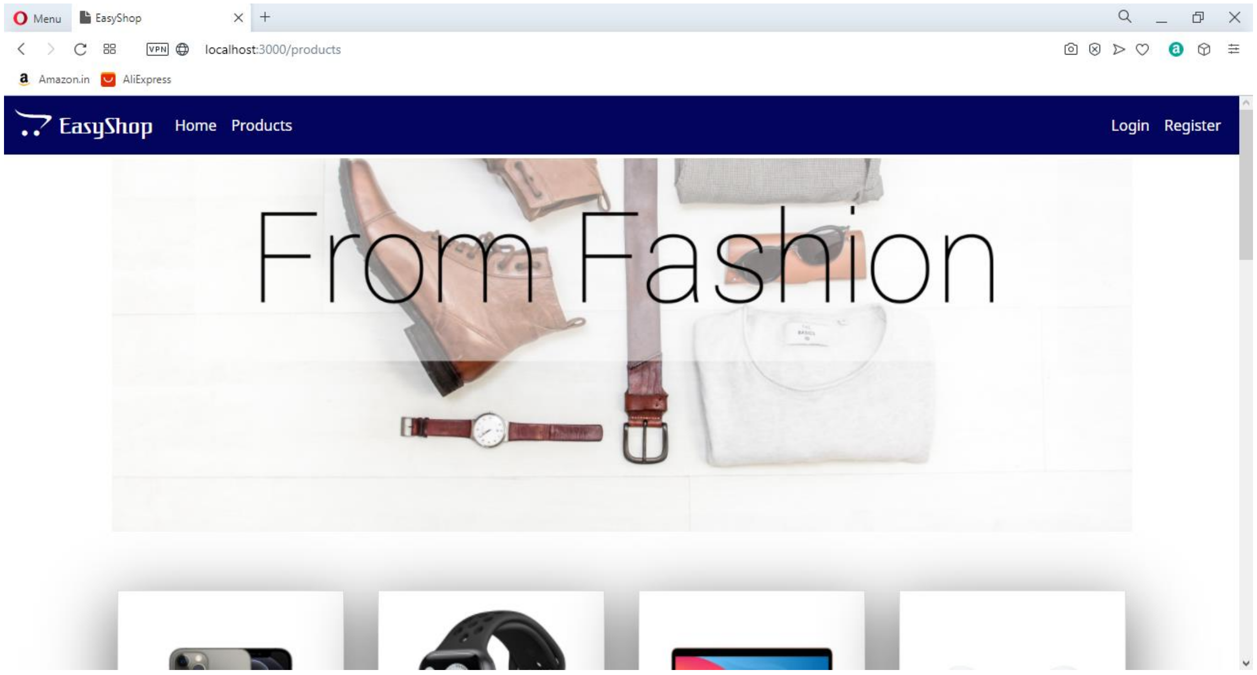
Shopping Made Easy

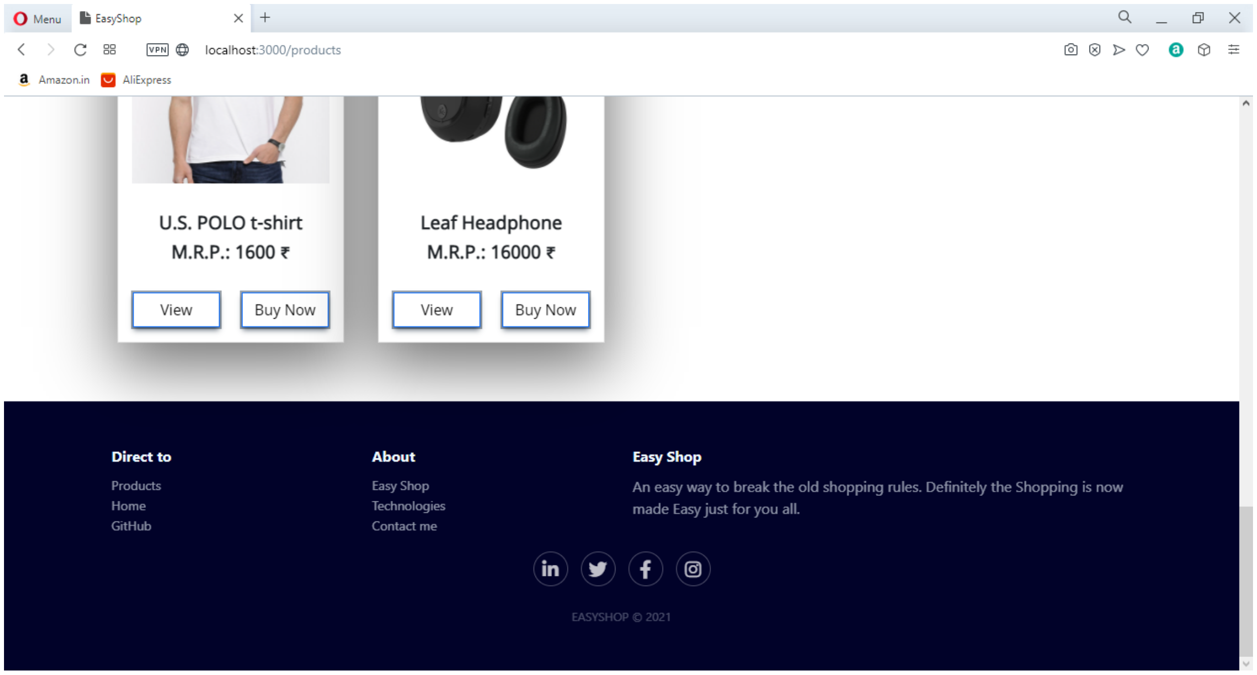
Jump right in and explore our many products. Feel free to add some of your own and comment on others!

Home Page (with login)


Menu
EasyShop
localhost:3000
Amazon.in
AliExpress
Would you like the password manager to save the password for "localhost:3000"?
Save
Never
EasyShop
PRODUCTS
CONTACT
LOGOUT
Shopping Made Easy
Jump right in and explore our many products. Feel free to add some of your own and comment on others!
EXPLORE

Products page (without login)





localhost:3000/products

U.S. POLO t-shirt
M.R.P.: 1600 ₹

Leaf Headphone
M.R.P.: 16000 ₹

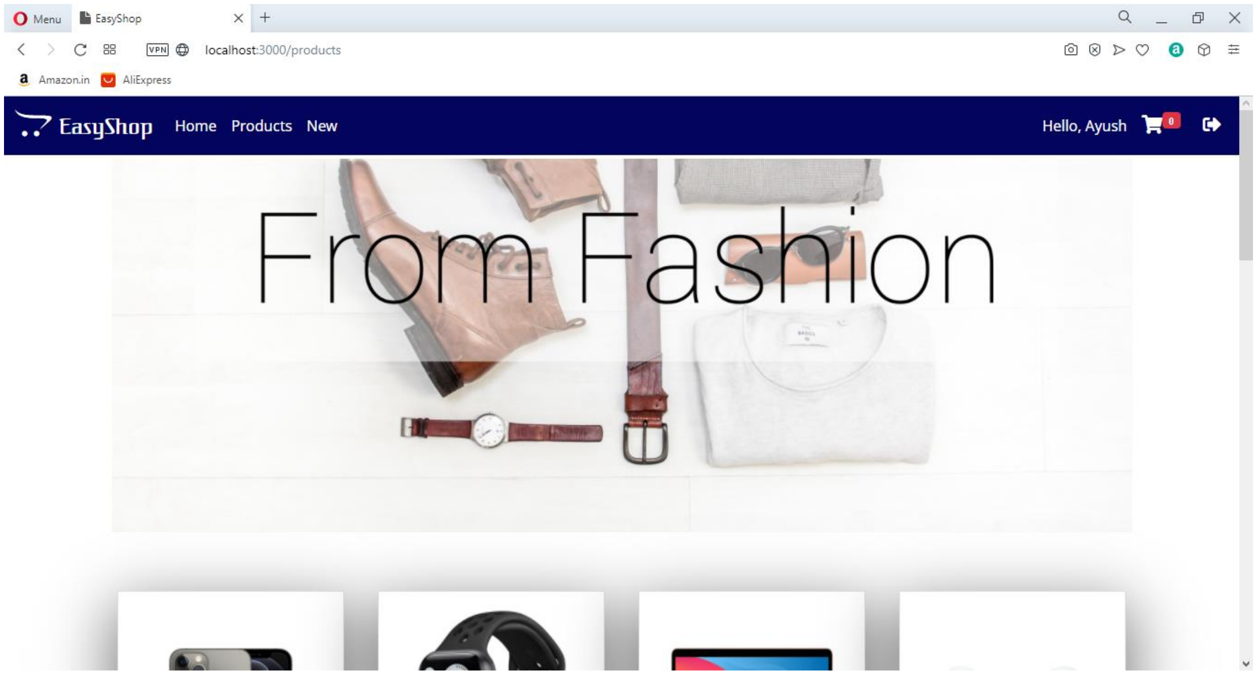
Direct to
Products
Home
GitHub

About
Easy Shop
Technologies
Contact me

Easy Shop
An easy way to break the old shopping rules. Definitely the Shopping is now made Easy just for you all.

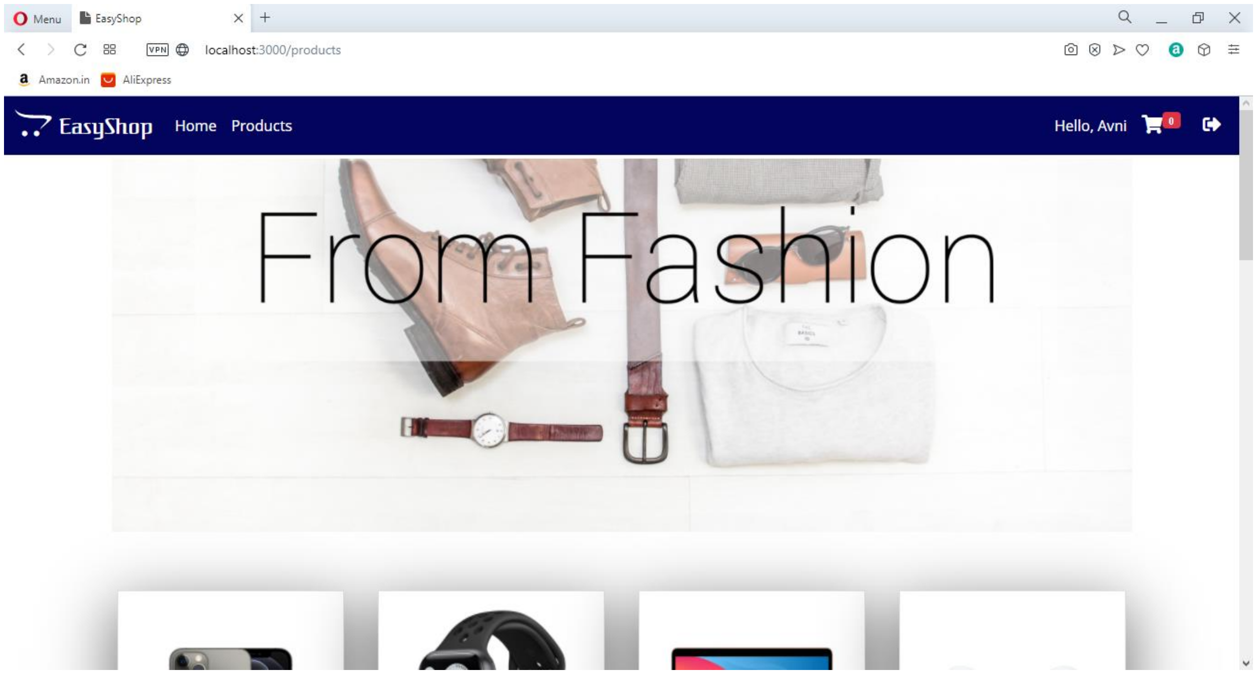
EASYSHOP © 2021

Products page (with login as retailer)

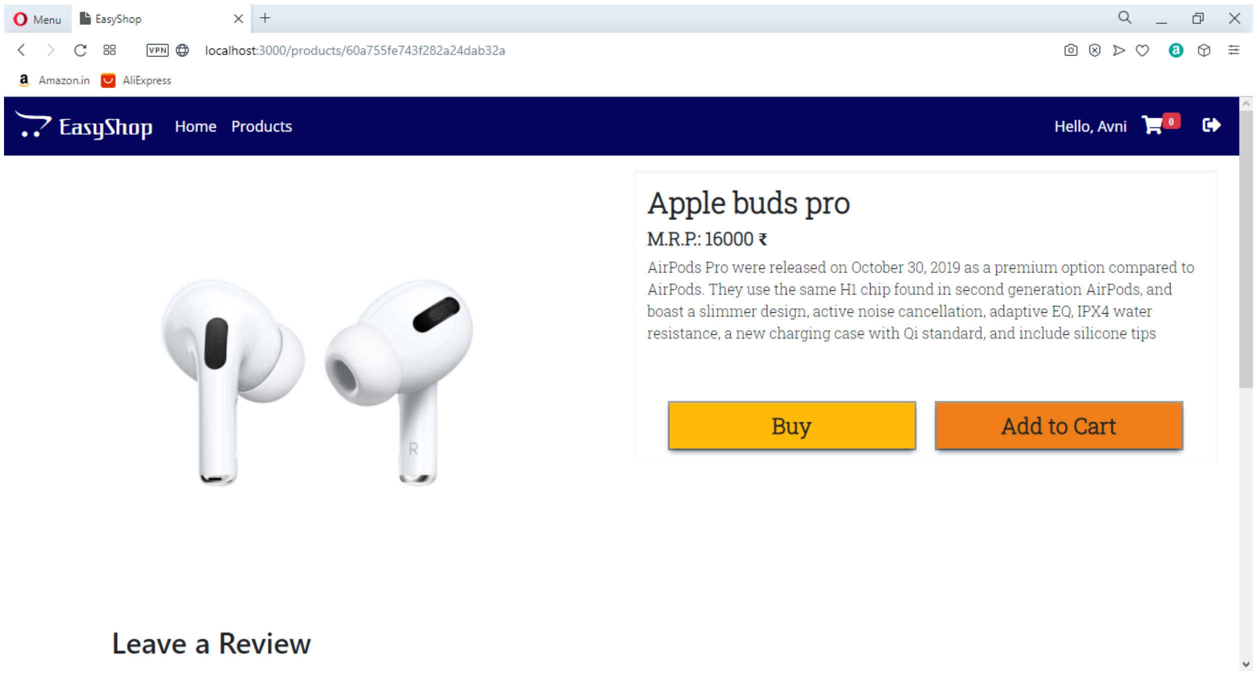


localhost:3000/products

Products page (with login as customer)



Product Show page (when logged out or logged in as Customer)



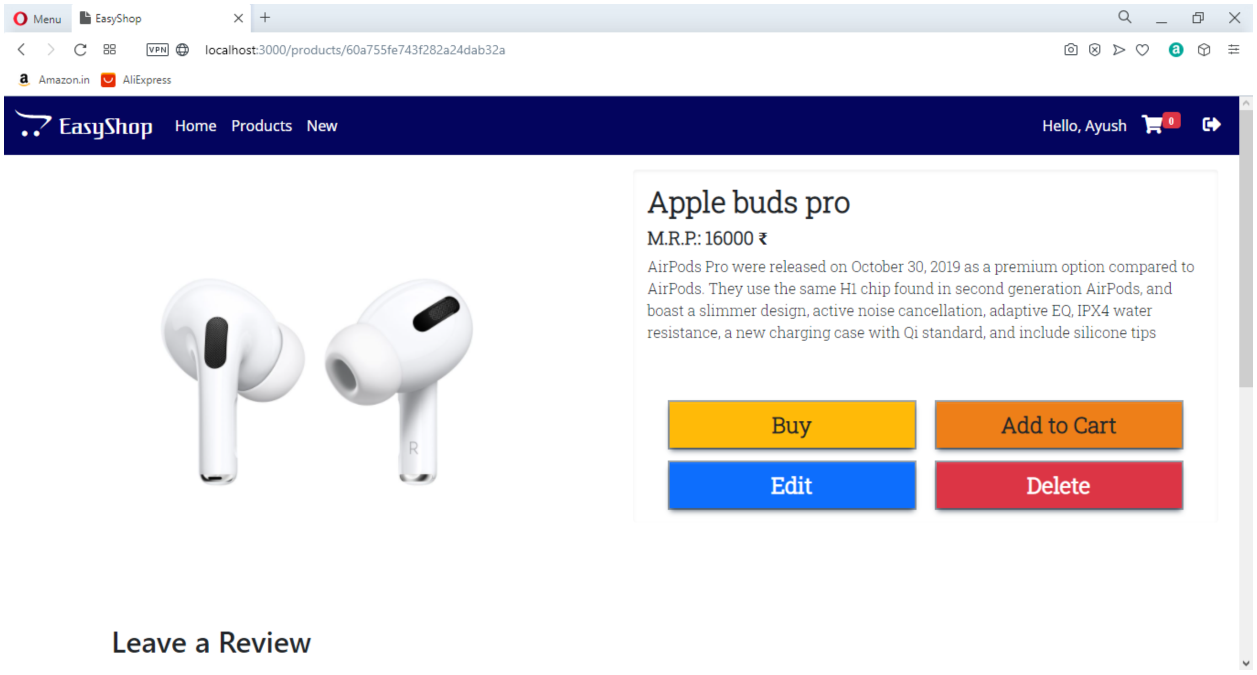
Apple buds pro

M.R.P.: 16000 ₹

AirPods Pro were released on October 30, 2019 as a premium option compared to AirPods. They use the same H1 chip found in second generation AirPods, and boast a slimmer design, active noise cancellation, adaptive EQ, IPX4 water resistance, a new charging case with Qi standard, and include silicone tips

Leave a Review

<u>Product Show page (when logged in as Retailer)</u>



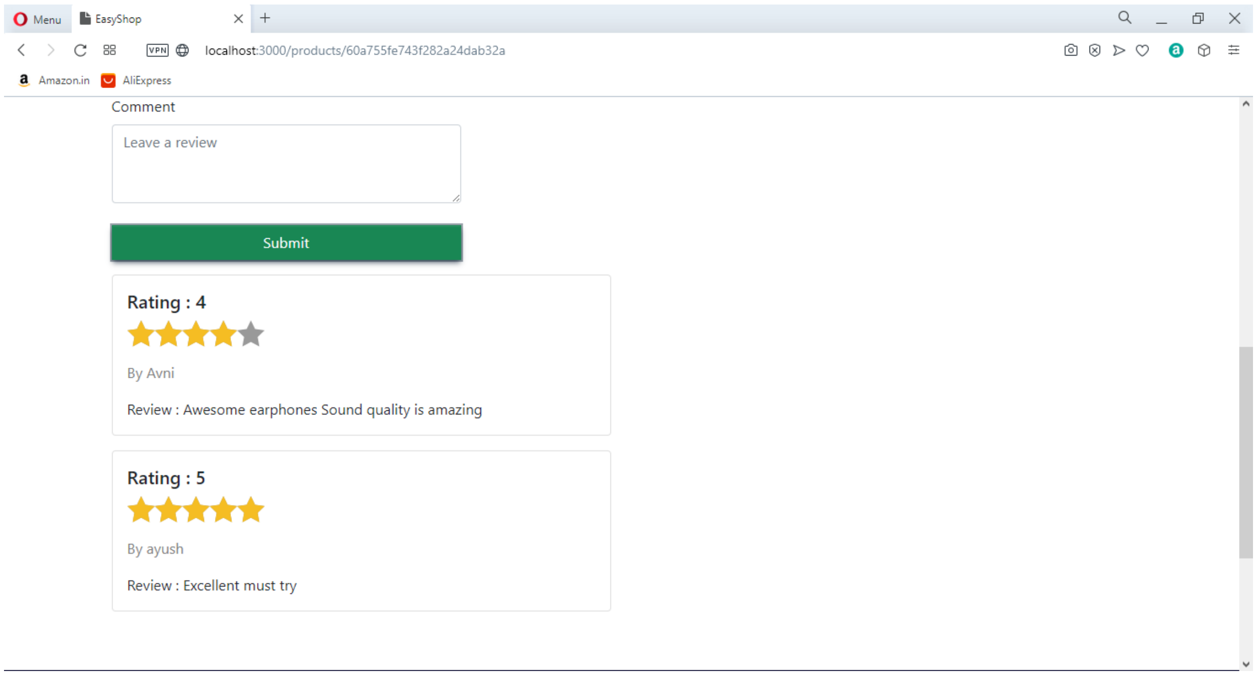
Apple buds pro

M.R.P.: 16000 ₹

AirPods Pro were released on October 30, 2019 as a premium option compared to AirPods. They use the same H1 chip found in second generation AirPods, and boast a slimmer design, active noise cancellation, adaptive EQ, IPX4 water resistance, a new charging case with Qi standard, and include silicone tips

Leave a Review

<u>Reviews</u>



Rating : 4

By Avni

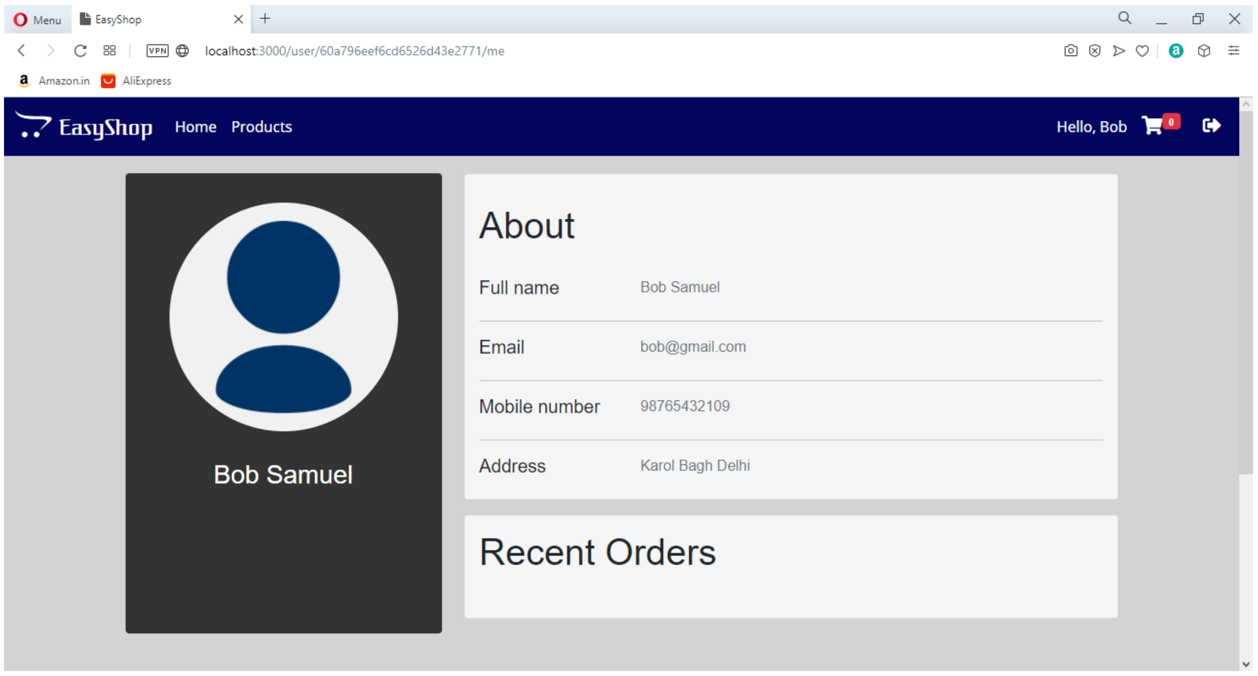
Review : Awesome earphones Sound quality is amazing

Rating : 5

By ayush

Review : Excellent must try

User Profile (who provided no image during signup)



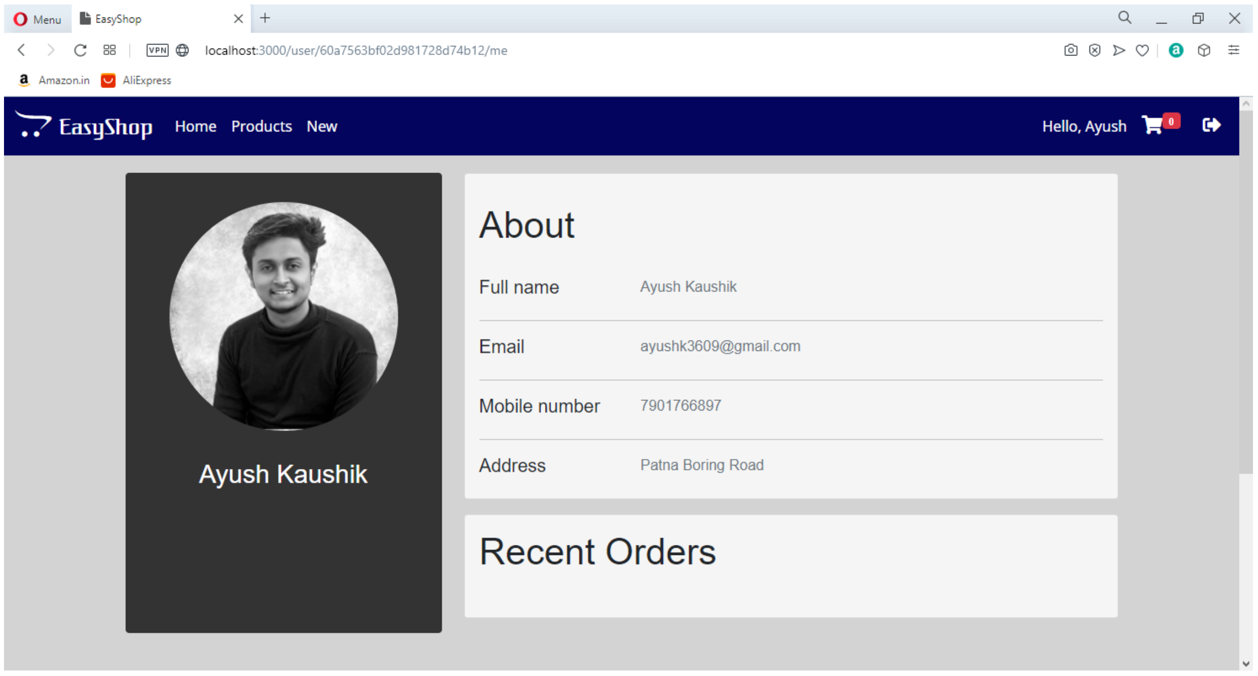
Bob Samuel

About

Full name	Bob Samuel
Email	bob@gmail.com
Mobile number	98765432109
Address	Karol Bagh Delhi

Recent Orders

User Profile (who provided image during signup)



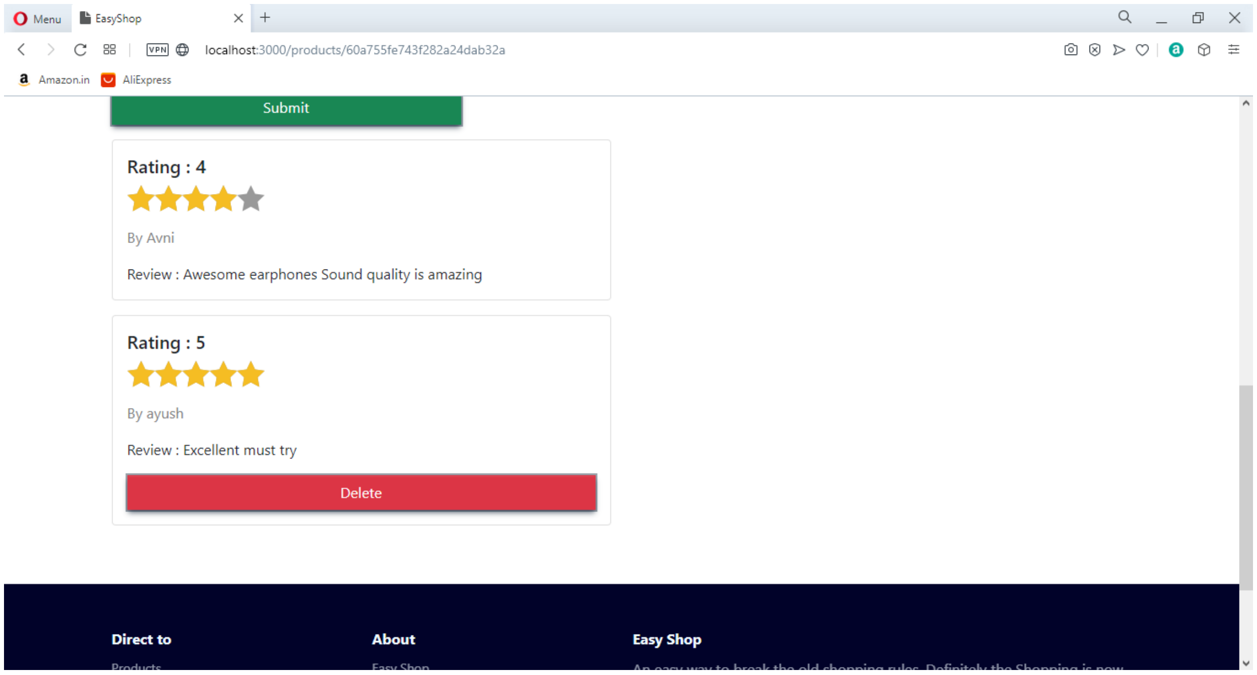
Ayush Kaushik

About

Full name	Ayush Kaushik
Email	ayushk3609@gmail.com
Mobile number	7901766897
Address	Patna Boring Road

Recent Orders

<u>User can delete their own review only</u>



Rating : 4

By Avni

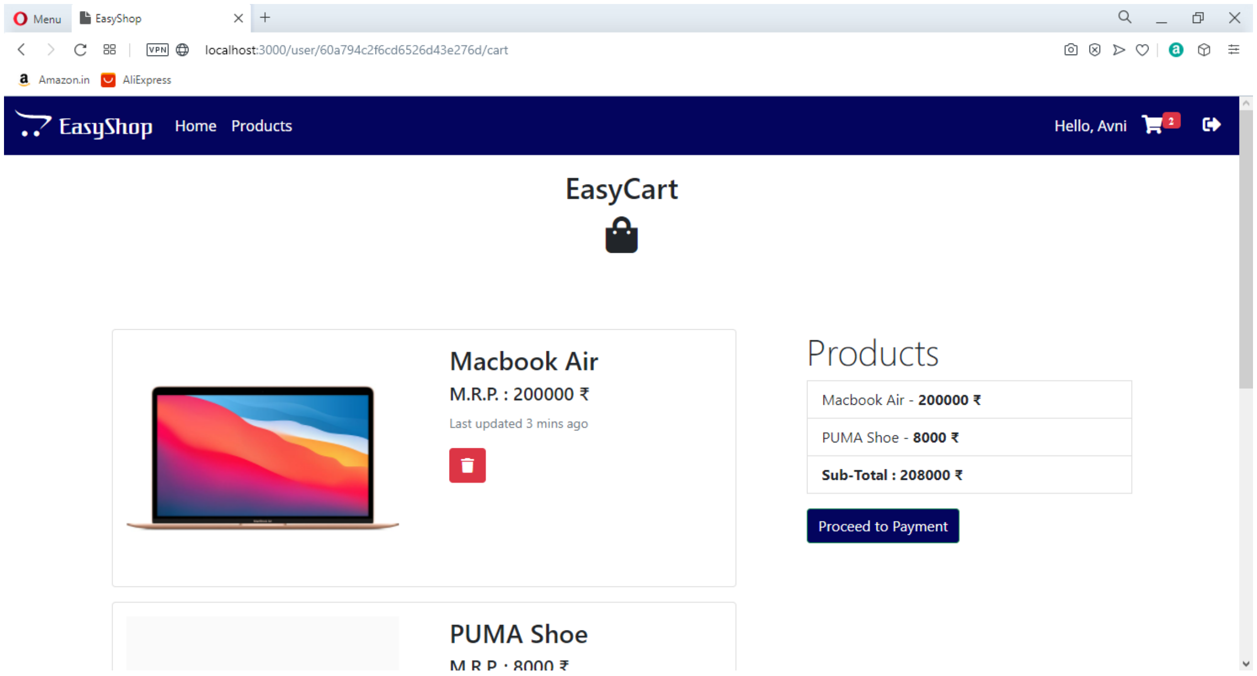
Review : Awesome earphones Sound quality is amazing

Rating : 5

By ayush

Review : Excellent must try

<u>Cart page (when items available)</u>



Macbook Air

M.R.P. : 200000 ₹

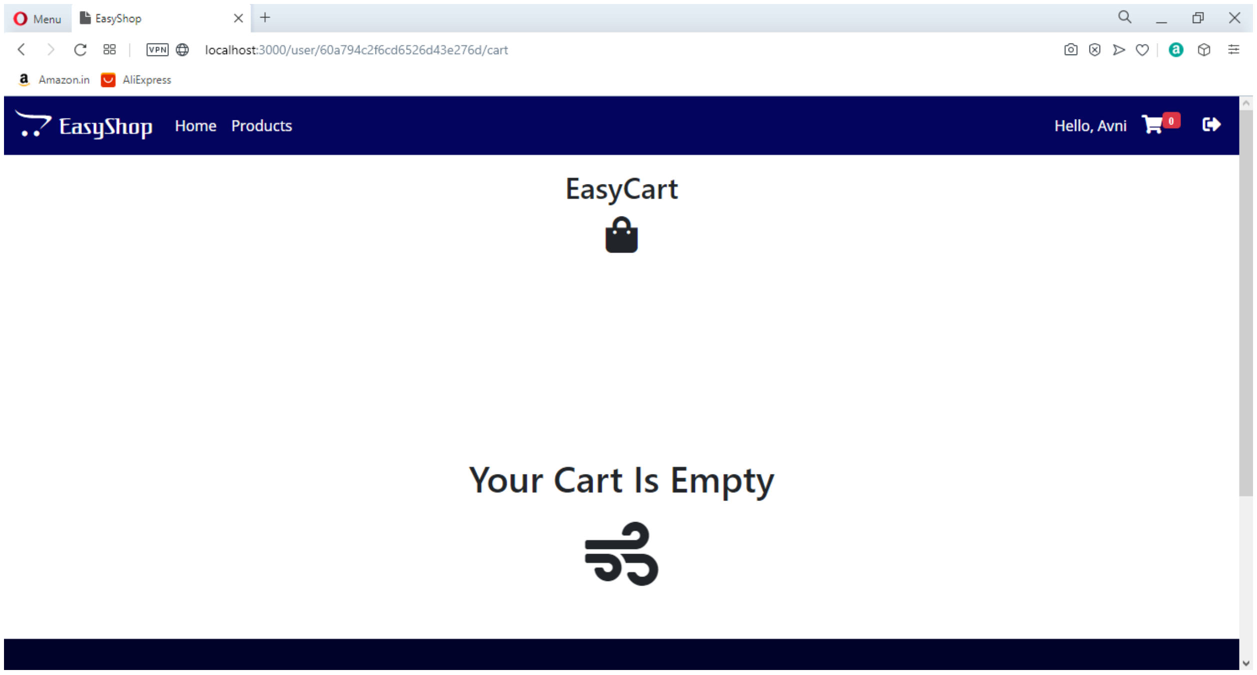
Last updated 3 mins ago

PUMA Shoe

Products

Macbook Air - **200000 ₹**
PUMA Shoe - **8000 ₹**
Sub-Total : 208000 ₹

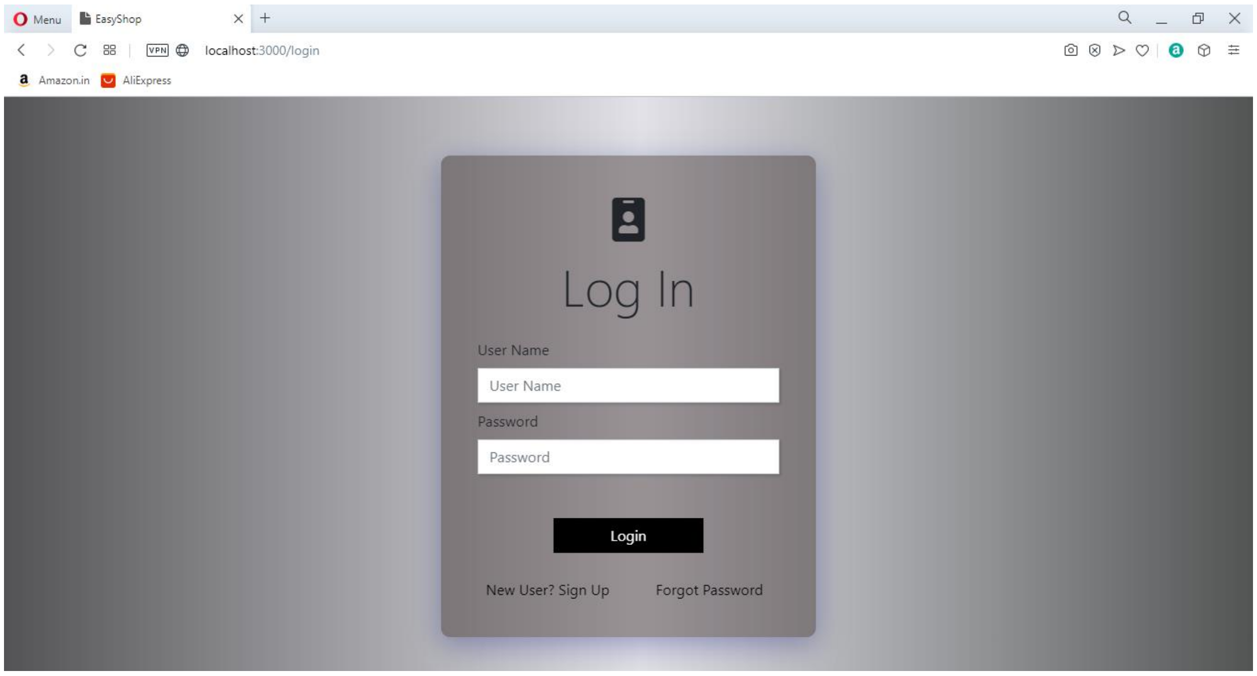
Cart page (when no items are available)



EasyCart

Your Cart Is Empty

Login page



Log In

User Name

User Name

Password

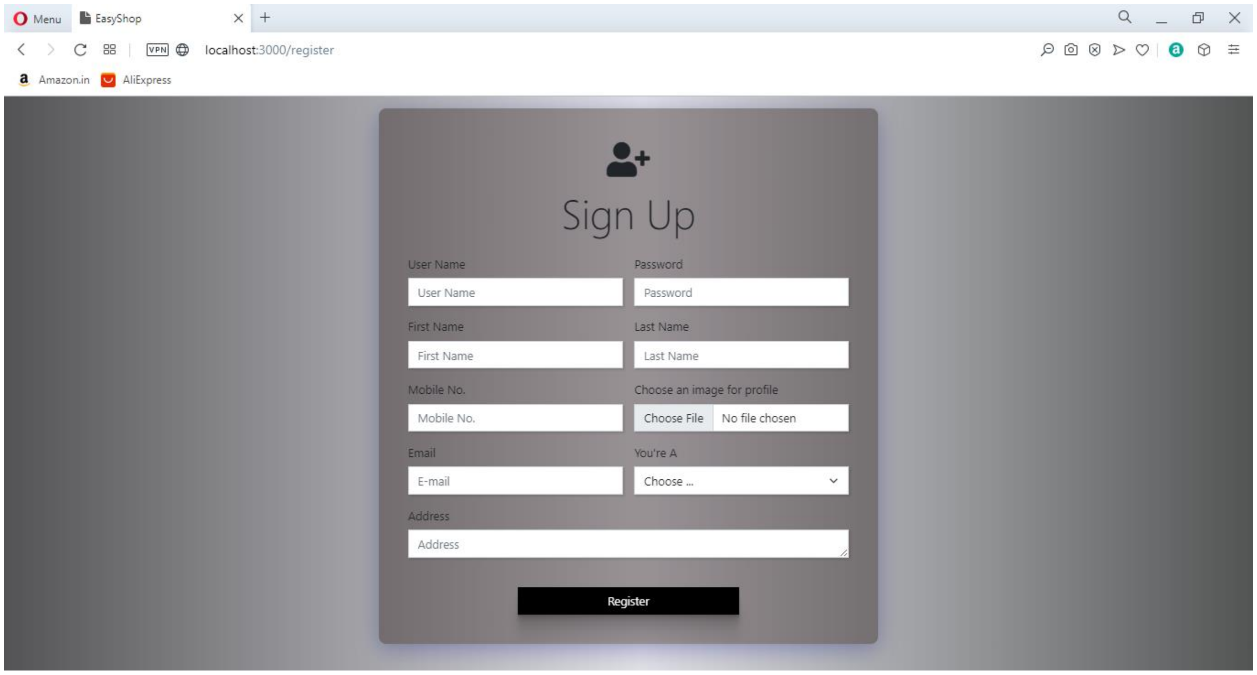
Password

Login

New User? Sign Up

Forgot Password

Signup page



Sign Up

User Name: User Name
Password: Password
First Name: First Name
Last Name: Last Name
Mobile No.: Mobile No.
Choose an image for profile: Choose File No file chosen
Email: E-mail
You're A: Choose ...
Address: Address

Register

Add new Product form for Retailers

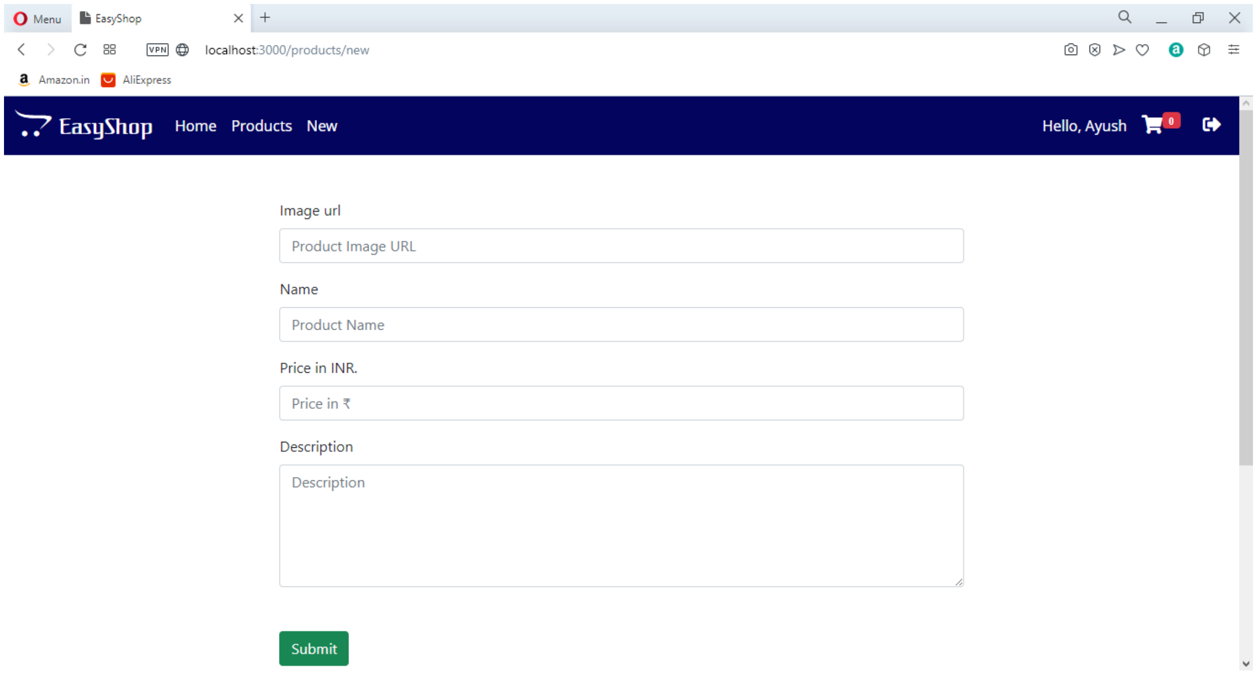


Image url: Product Image URL
Name: Product Name
Price in INR.: Price in ₹
Description: Description

Submit

Edit Page for Retailers

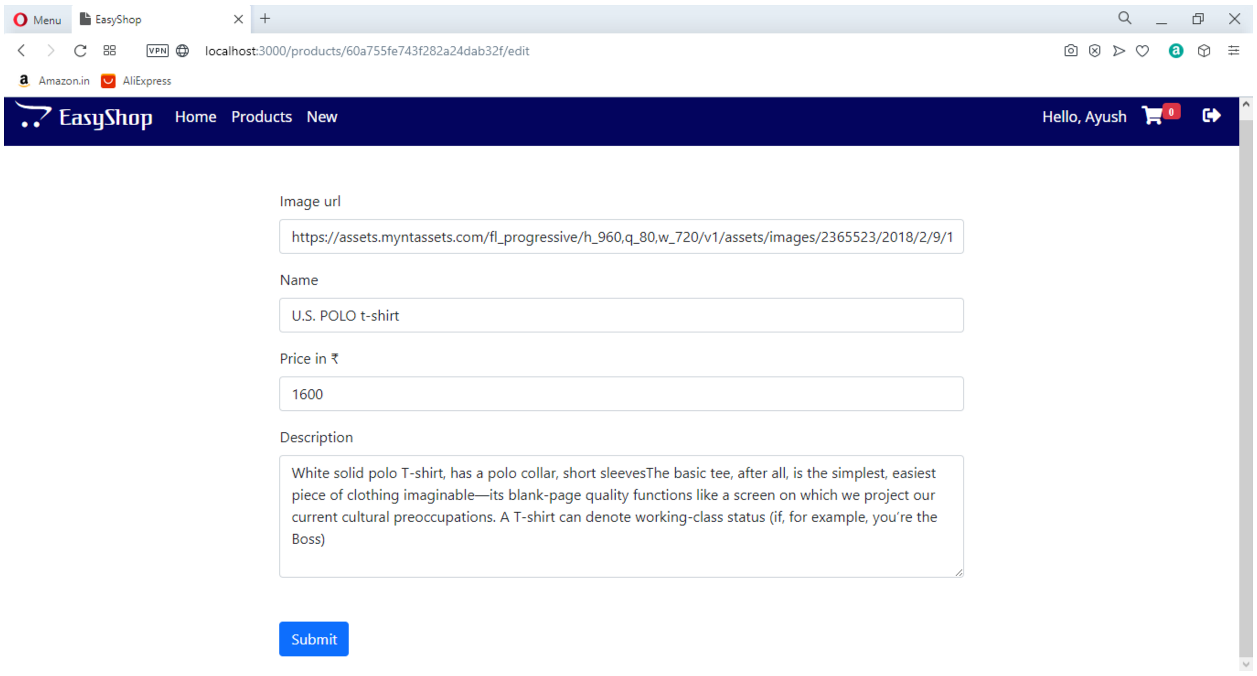


Image url

https://assets.myntassets.com/fl_progressive/h_960,q_80,w_720/v1/assets/images/2365523/2018/2/9/1

Name

U.S. POLO t-shirt

Price in ₹

1600

Description

White solid polo T-shirt, has a polo collar, short sleevesThe basic tee, after all, is the simplest, easiest piece of clothing imaginable—its blank-page quality functions like a screen on which we project our current cultural preoccupations. A T-shirt can denote working-class status (if, for example, you're the Boss)

9.Testing:

Test Case 1 :

Testing Type : Unit Testing

Input: Password

Objective: Checking Password Security

Actual Output: Authorized users are only accessed

Expected Output Satisfied

<u>Test Case 2:</u>

Testing Type: Unit Testing

Input : Data

Objective: Testing Successful Data Entry in Database

Actual Output: Data Entry Successfully done

Expected Output Satisfied

<u>Test Case 3:</u>

Testing Type: Unit Testing

Input: Product Details

Objective: Checking Successful addition of products

Actual Output: Product Successfully added and is shown

Expected Output Satisfied

<u>Test Case 4:</u>

Testing Type: Unit Testing

Input: Edit

Objective: Editing Product Details By Retailer

Actual Output: Product Updated Successfully and changes are seen

Expected Output Satisfied

Bibliography

- https://www.udemy.com/course/the-complete-web-developer-zero-to-mastery/
- https://codingblocks.com/
- https://cloudinary.com/
- https://developer.mozilla.org/en-US/
- http://www.passportjs.org/
- https://mongoosejs.com/
- https://nodejs.org/en/docs/
- https://expressjs.com/en/guide/routing.html
- https://docs.npmjs.com/packages-and-modules
- https://www.youtube.com/