AUTOMOBILE DEALERSHIP MANAGEMENT SYSTEM

REVIEW REPORT

Submitted by

K V N HEMANTH (18BCE0538)

K SAI KOUSHIK (18BCE0561)

JAYANTH GOLLAPUDI (18BCI0129)

Prepared For

SOFTWARE ENGINEERING (CSE3001) - PROJECT J COMPONENT

Submitted To

DR. AMUTHA PRABAKAR M

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

ABSTRACT

Nowadays everything is automated using computer systems, so many companies are developing web-applications to provide online services to their customers. Automobile Dealership Management System software is a web-based application developed to manage and track the details of vehicle sales, workshop and service, spare parts inventory and their customers can search and then view the complete specification of each vehicle listing with its features and buy the vehicle as per their requirements. This is a simple yet efficient management system to manage records of dealership data instead of using traditional methods to manage data like manual records or file systems. Their customers can always be in the comfort of their homes to check the various vehicles, spare parts listed in the website for sale as to their choice. Hence reducing the cost, saving time and increasing the comfort of their customers**.**

In the case of a traditional method where the sales and purchase records are stored manually through registers or file systems which is very time consuming and inefficient. Moreover, in this kind of management, there is lots of paperwork involved for the generation of bills, reports, invoices for every transaction which needs to be sorted and maintained manually which are prone to human. The management of these manual records is very hard, productivity is lost using manual records, searching of manual records is tedious, the security of records is unreliable.

This Automobile Dealership management system is a digitalized web-based application where the entire records are maintained by the database management system itself. It provides an intuitive user interface, which is easy to use, for both the dealers and their customers. It will reduce manual work and helps the dealer to save the records related to the details of customers, vehicles sales, transactions efficiently. Any calculations involved in the transactions will be automated increasing the work pace of the system and also reduces the possibilities of incidents of human mistakes. This Automobile Dealership management system tackles these problems through advanced solutions, consumes very less time, error-free compared to manual management. Being an organized data storage system, it simplifies the work of searching by faster search of records and reducing the work delay thus allowing the customers to find the right vehicle or spare part they are searching without any sort of hassle. Also, one fundamental characteristic to highlight is that this management system provides data security and backup service which might be very crucial for the practical implementation of such systems.

INTRODUCTION

The Automobile Dealership management system is a web application designed for dealership agency, which has an intuitive and effortless to use to interface for both its agency owner and its customers. Here owners can log in with their given ID and password in order to manage the web application. The dealership can post the vehicles that needs to be sold by the agency in the vehicle listings. The customers of the agency can register and log into the website. Here the customers can browse through various properties listed on the website which enable the customer to find the vehicle as per his/her requirements.

Customers can search vehicles in the listings and can then view the complete specification of each listing with its features, engine, mileage, price, etc. according to their requirements and can buy the vehicle by initiating a registration with the property. Then the customer needs to pay the agency the preorder price of the vehicle that the customer wants to purchase. After successful completion of the transaction process, the property gets registered under the customer's name and the money gets wired to the agency owner. This method greatly enhances the speed of any process and reduces the overhead of documentation. The dealership also provides details concerning the vehicles that are under service in their garage. Also, the dealership system provides inventory management by managing the sale of accessories of the vehicles to the customers.

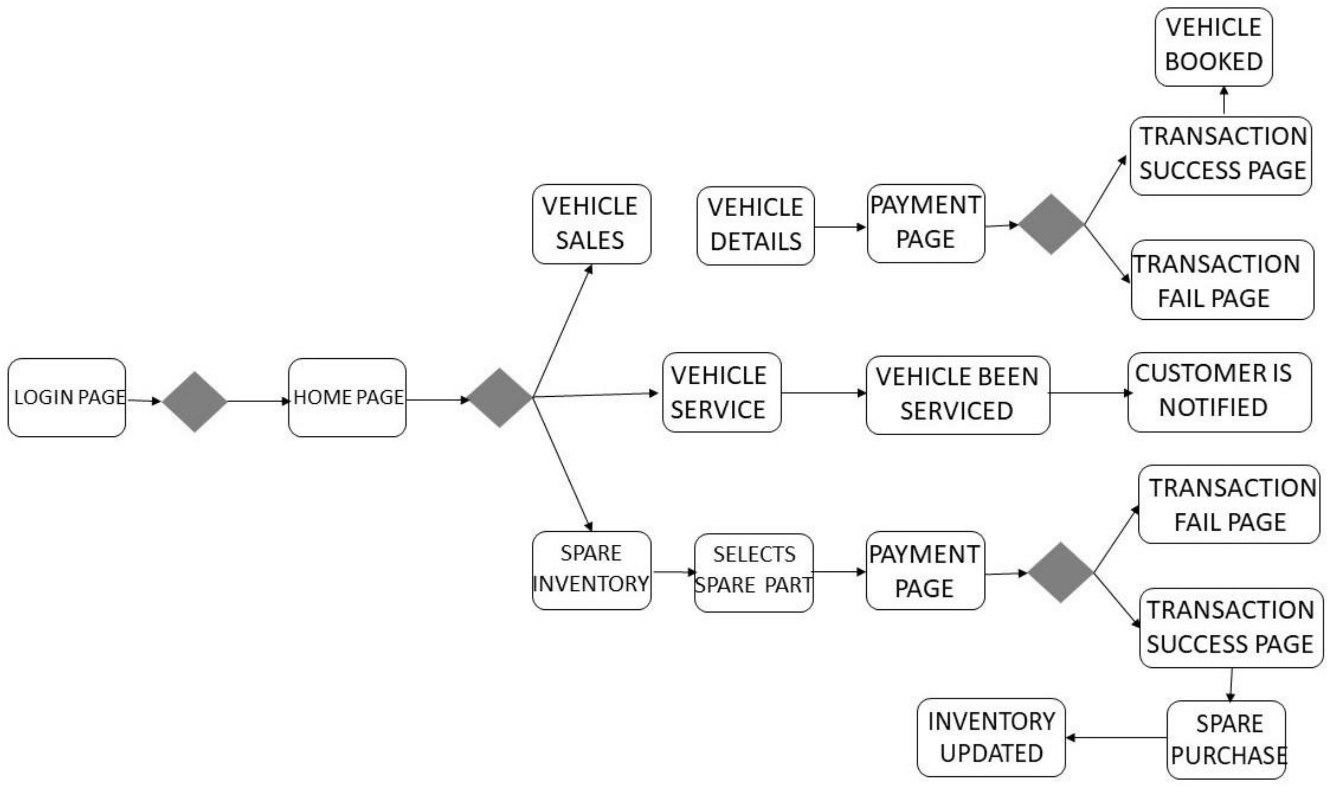
This Automobile Dealership management system is developed using the latest technologies available like HTML5, JavaScript ES6, MongoDB with the intention of incorporating innovative features and enhancing the existing ones to increase the overall efficiency of the system. These helps make the application more user-friendly and makes it more productive. HTML5 provides the basic structural markup for the web app and interactions for the website, CSS improves the aesthetics of the application, JavaScript brings the behavior and basic logical operations to the website by adding features to conveniently search for vehicles and display the list of servicing vehicles and inventory, MongoDB stores the complete records of vehicles, servicing details, inventory details etc. These technologies allow faster search, interaction and deal closure which is profitable to both the agency and the client.

PROPOSED WORK

The objective of the automobile dealership software is to manage and track the details of vehicle sales, workshop and service, spare parts inventory and sales so the dealership could avoid manual work for managing the information which is very time-consuming and inefficient and for the customer who are the end users they can be in the comfort of their homes to browse and check status of vehicles and spare parts. This system tackles these problems by automating the manual work thus consumes very less time, error-free, additionally provide security and backup to the dealership data and increase the efficiency of the dealership. The automobile dealership management software is a web-based system that facilitates the management of an automobile dealership with the help of sub systems each handling different functionalities like vehicle sales which maintains and tracks all the details regarding the sale of vehicles, service which tracks the vehicles that are currently being serviced by the dealership, spare inventory which maintains the stocks of the spares in the inventory. This approach of modular solution helps in resolving the bugs faster and increases the pace of development since these can be developed separated but can be tested together. All this combine to ultimately provide the management of information for the dealership so that the dealer could manage business in an effortless manner.

In this system, the dealer or the customer first needs to authenticate themselves to start using the application. Here dealer can log in with their given ID and password in order to manage the website. The dealer can post the vehicles that needs to be sold by the dealership in vehicle sales. The customers of the dealership can register and log into the website. Here the customers can browse through various vehicles listed on the website which enables the customer to find the vehicle as per his/her requirements. Customers can search vehicles in the sales and can then view the complete specification of each vehicle listing with its features, price, etc. according to their requirements and can book the vehicle by initiating a purchase. Then the customer needs to pay the dealership the price of the property that the customer wants to purchase. After the successful completion of the transaction process, the vehicle will be booked under that customer. The customer can service their vehicle in the dealership and the system tracks this information and informs the customer of pickup. Also, customers can buy spare parts from the dealership with the help of spare part inventory and similarly buy the required components by the customer and can check out.

PROPOSED ARCHITECTURE



The aim of the software modularization process is to partition a software system into subsystems to provide an abstract view of the architecture of the software system. In this system, the dealer or the customer first needs to authenticate themselves to start using the application. Here dealer can log in with their given ID and password in order to manage the website. The dealer can post the vehicles that needs to be sold by the dealership in vehicle sales. The customers of the dealership can register and log into the website. Here the customers can browse through various vehicles listed on the website which enables the customer to find the vehicle as per his/her requirements. Customers can search vehicles in the sales and can then view the complete specification of each vehicle listing with its features, price, etc. according to their requirements and can book the vehicle by initiating a purchase. Then the customer needs to pay the dealership the price of the property that the customer wants to purchase. After the successful completion of the transaction process, the vehicle will be booked under that customer. The customer can service their vehicle in the dealership and the system tracks this information and informs the customer of pickup. Also, customers can buy spare parts from the dealership with the help of spare part inventory and similarly buy the required components by the customer and can check out.

PROPOSED MODULES

LIST OF MODULES

1. Vehicle Sales
2. Vehicle Service
3. Accessories Inventory

VEHICLE SALES

Vehicle sales includes managing the vehicle details which contains the model of the vehicle, price etc. that are currently available, and the vehicles and vehicle Booking includes the details of customer who bought that vehicle and the model of the vehicle that got booked by the customer. Customers can search vehicles in the sales and can then view the complete specification of each vehicle listing with its features, price, etc. according to their requirements and can book the vehicle by initiating a purchase. Payment and Transactions includes the payment option, cost of the product, name of the bank and user id.

PROPOSED IMPLEMENTATION

- Create a base HTML page in which all the components will be embedded with navigation and hyperlinks to rest of the connected pages that are part of the web application.
- Embed the components that are responsible for displaying each unit which hold the details of each vehicle.
- Retrieve the details of each vehicle from the backend database.
- Display the query details onto the application.
- Display each unit data in the order desirable to the customer.
- Selection of a particular vehicle will enable the customer to proceed to booking it.
- Payment handling will be handled using NPM packages in NodeJS.

VEHICLE SERVICE

Vehicle service consists of the no. of the vehicles that are currently being serviced and the vehicles that got serviced and delivered to the customer recently and it consists of the status of vehicles that are currently being served and gets notified to the customer when it completes.

PROPOSED IMPLEMENTATION

- Create a hyperlink that allows the customer to navigate to the service page of the web application through the navigation menu.
- Get customer's vehicle data in order to display the status of the vehicle servicing.
- Component responsible for this action sends query to the database.
- Retrieve the required details of vehicle service status from the backend database across the inputs details taken from customer.
- Display the query details onto the application component.
- Status of the servicing available to the customer to see.
- In case of finishing the service before the customer notices, using NPM package a mail can be sent to the required customer retrieved from the database.

SPARES INVENTORY

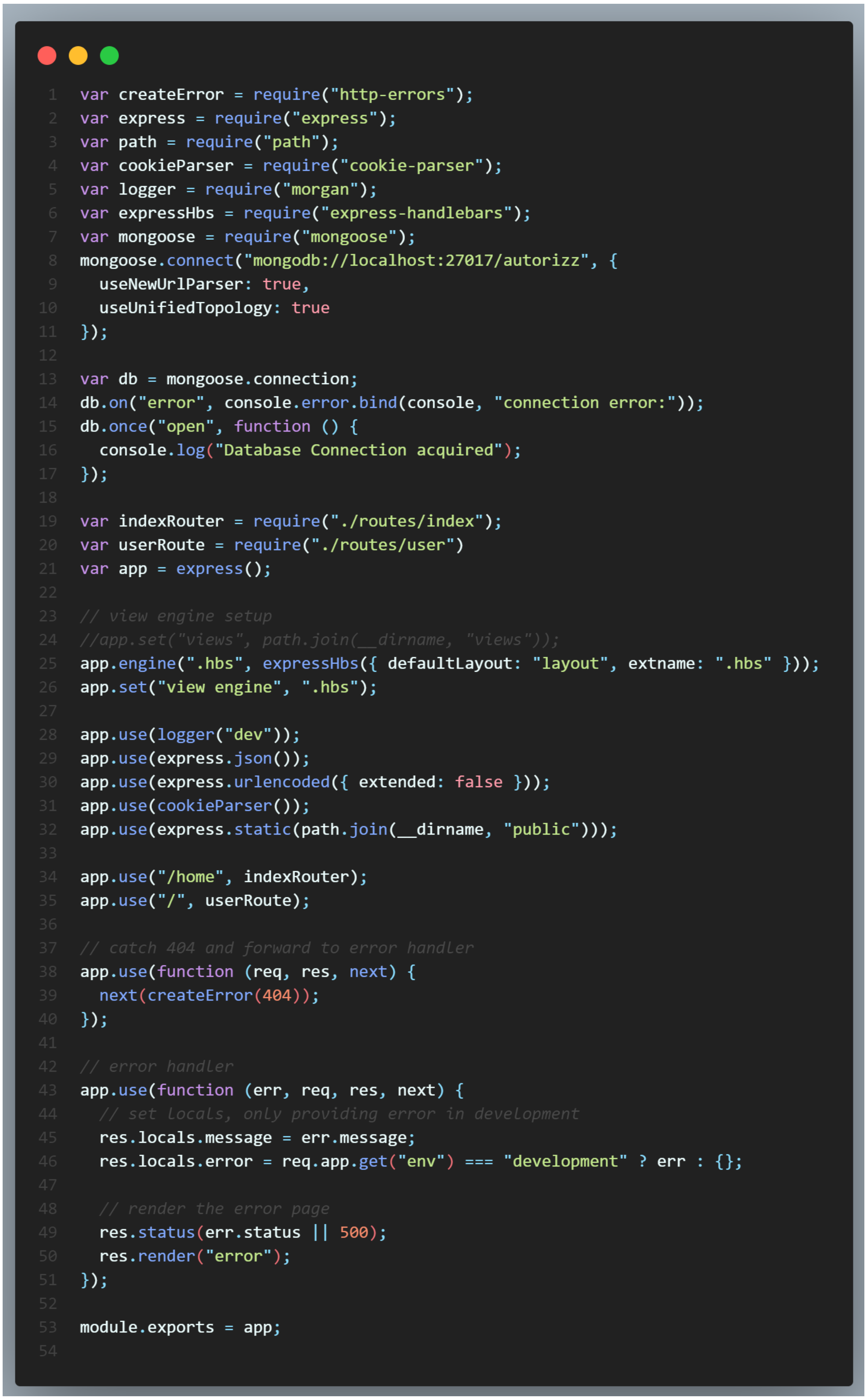
Spares Inventory includes the details of spares available in the inventory and purchase history of spares and it also includes the payment details of the purchase of spare parts. Here customers can buy spare parts from the dealership with the help of spare part inventory and similarly buy the required components by the customer and can check out.

PPROPOSED IMPLEMENTATION:

- Create a hyperlink that allows the customer to navigate to the spares inventory page of the web application through the navigation menu.
- Retrieve the details of each vehicle from the backend database.
- Display each unit containing details of each spare onto the application.
- Spares selected will be phased into temporary cart.
- Confirmation of the cart proceeds the customer to the payment which is handled by an NPM package.
- Generation of invoice follows.

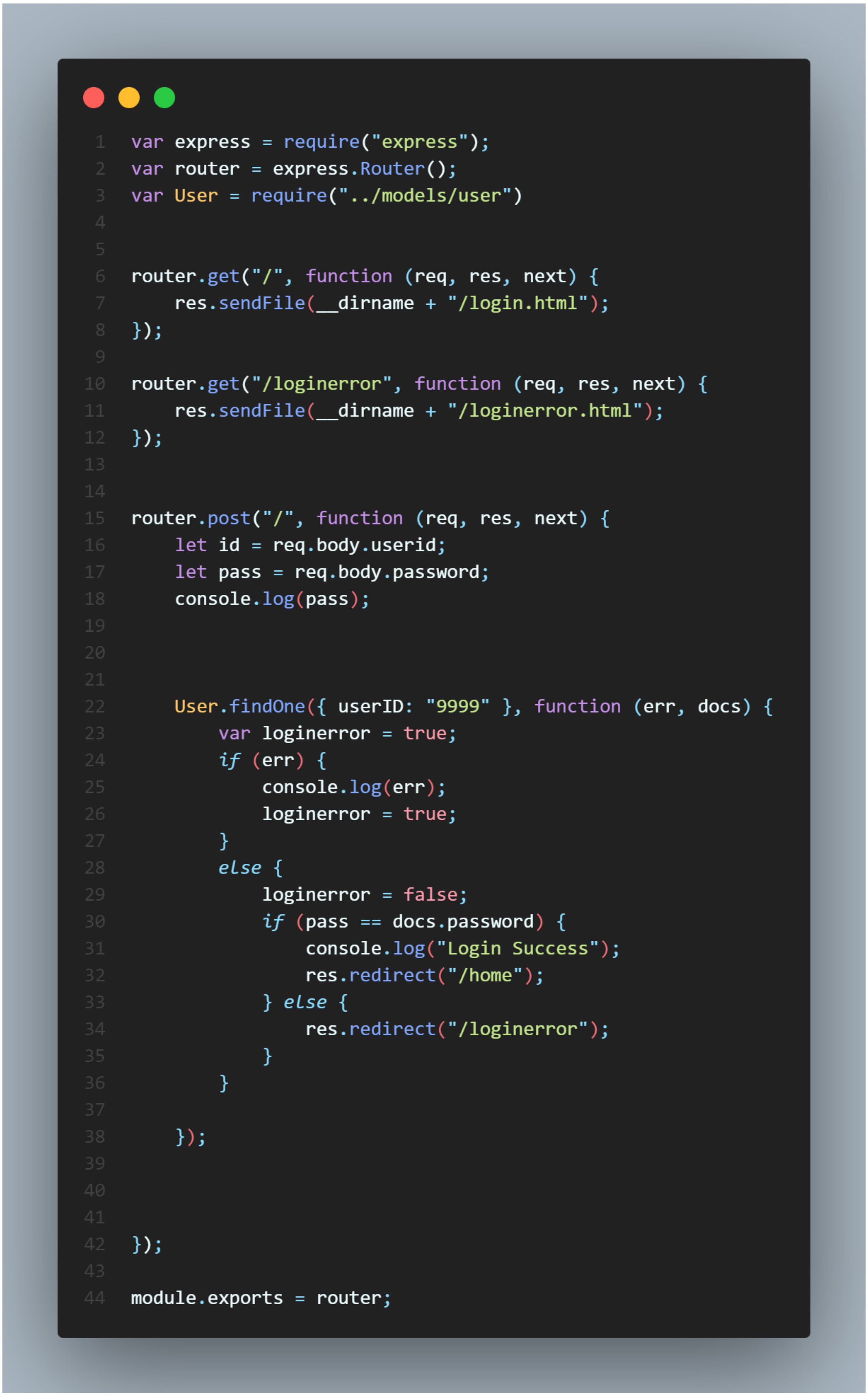
PROGRAM MODULE

MAIN APP MODULE



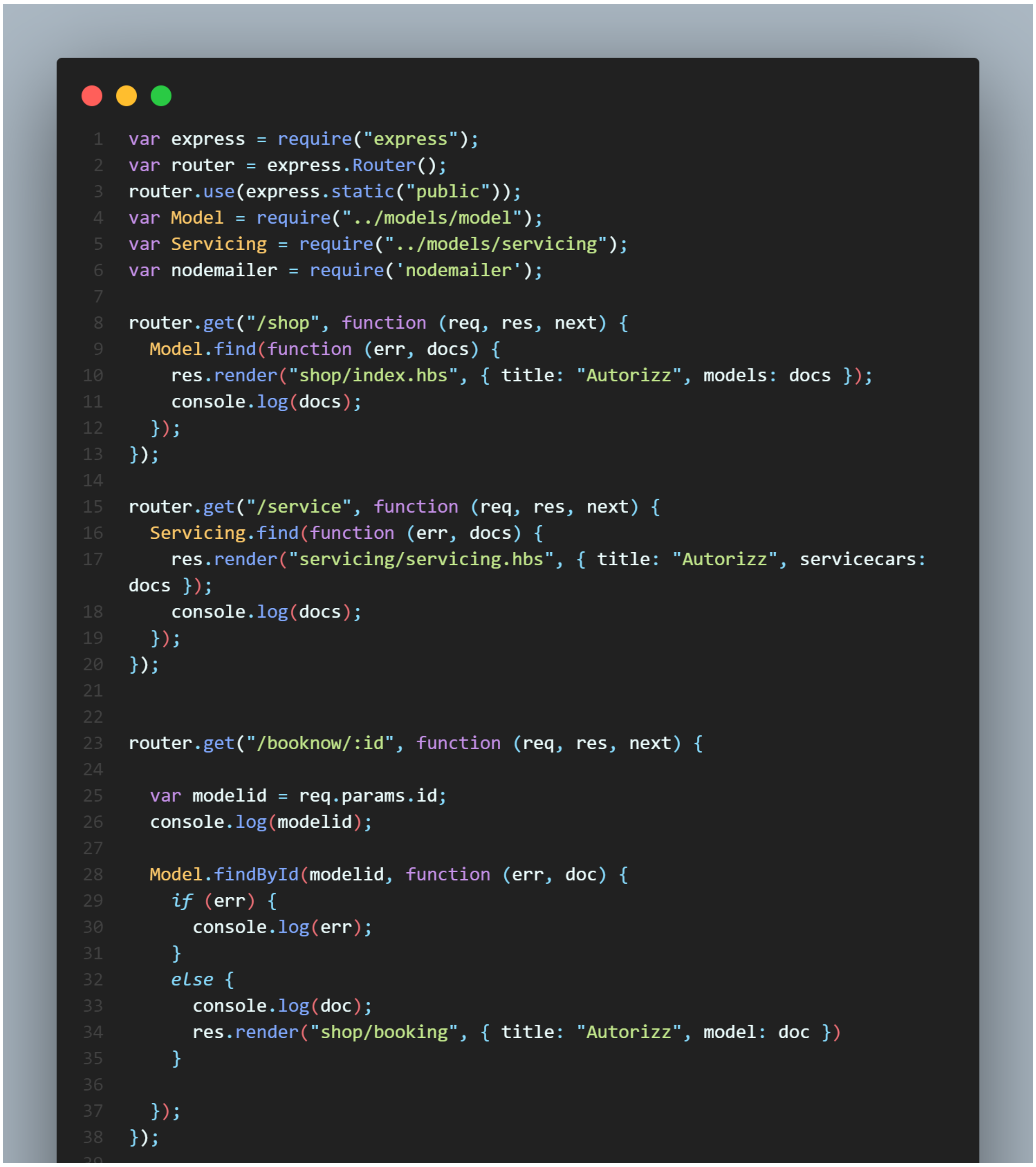
```
var createError = require("http-errors");
var express = require("express");
var path = require("path");
var cookieParser = require("cookie-parser");
var logger = require("morgan");
var expressHbs = require("express-handlebars");
var mongoose = require("mongoose");
mongoose.connect("mongodb://localhost:27017/autorizz", {
  useNewUrlParser: true,
  useUnifiedTopology: true
});

var db = mongoose.connection;
db.on("error", console.error.bind(console, "connection error:"));
db.once("open", function () {
  console.log("Database Connection acquired");
});

var indexRouter = require("./routes/index");
var userRoute = require("./routes/user")
var app = express();

// view engine setup
//app.set("views", path.join(__dirname, "views"));
app.engine(".hbs", expressHbs({ defaultLayout: "layout", extname: ".hbs" }));
app.set("view engine", ".hbs");

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

app.use("/home", indexRouter);
app.use("/", userRoute);

// catch 404 and forward to error handler
app.use(function (req, res, next) {
  next(createError(404));
});

// error handler
app.use(function (err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get("env") === "development" ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render("error");
});

module.exports = app;

```

LOGIN MODULE



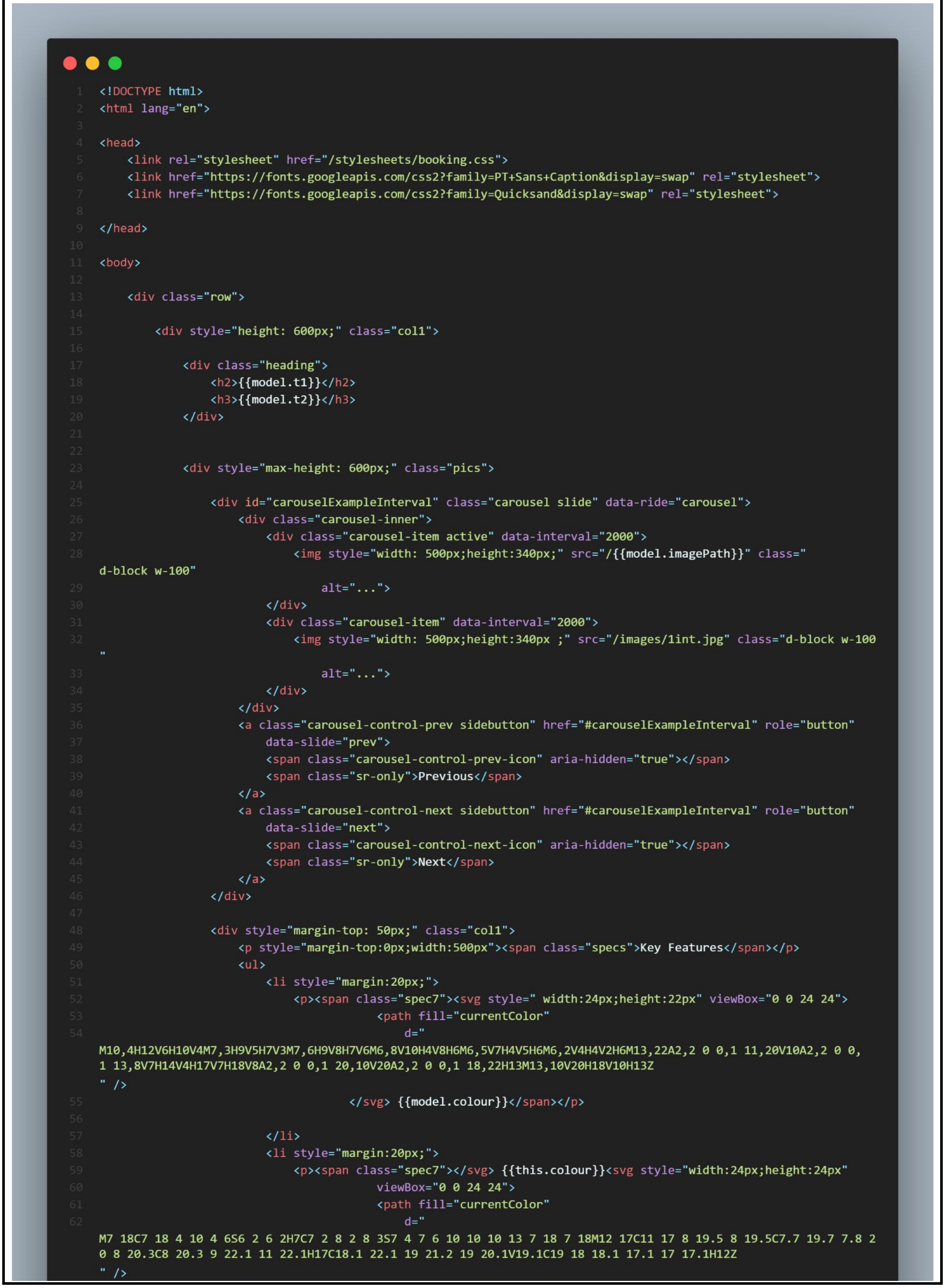
```javascript
var express = require("express");
var router = express.Router();
var User = require("../models/user")


router.get("/", function (req, res, next) {
    res.sendFile(__dirname + "/login.html");
});

router.get("/loginerror", function (req, res, next) {
    res.sendFile(__dirname + "/loginerror.html");
});


router.post("/", function (req, res, next) {
    let id = req.body.userid;
    let pass = req.body.password;
    console.log(pass);



    User.findOne({ userID: "9999" }, function (err, docs) {
        var loginerror = true;
        if (err) {
            console.log(err);
            loginerror = true;
        }
        else {
            loginerror = false;
            if (pass == docs.password) {
                console.log("Login Success");
                res.redirect("/home");
            } else {
                res.redirect("/loginerror");
            }
        }

    });



});

module.exports = router;
```

VEHICLE SALES IMPLEMENTATION

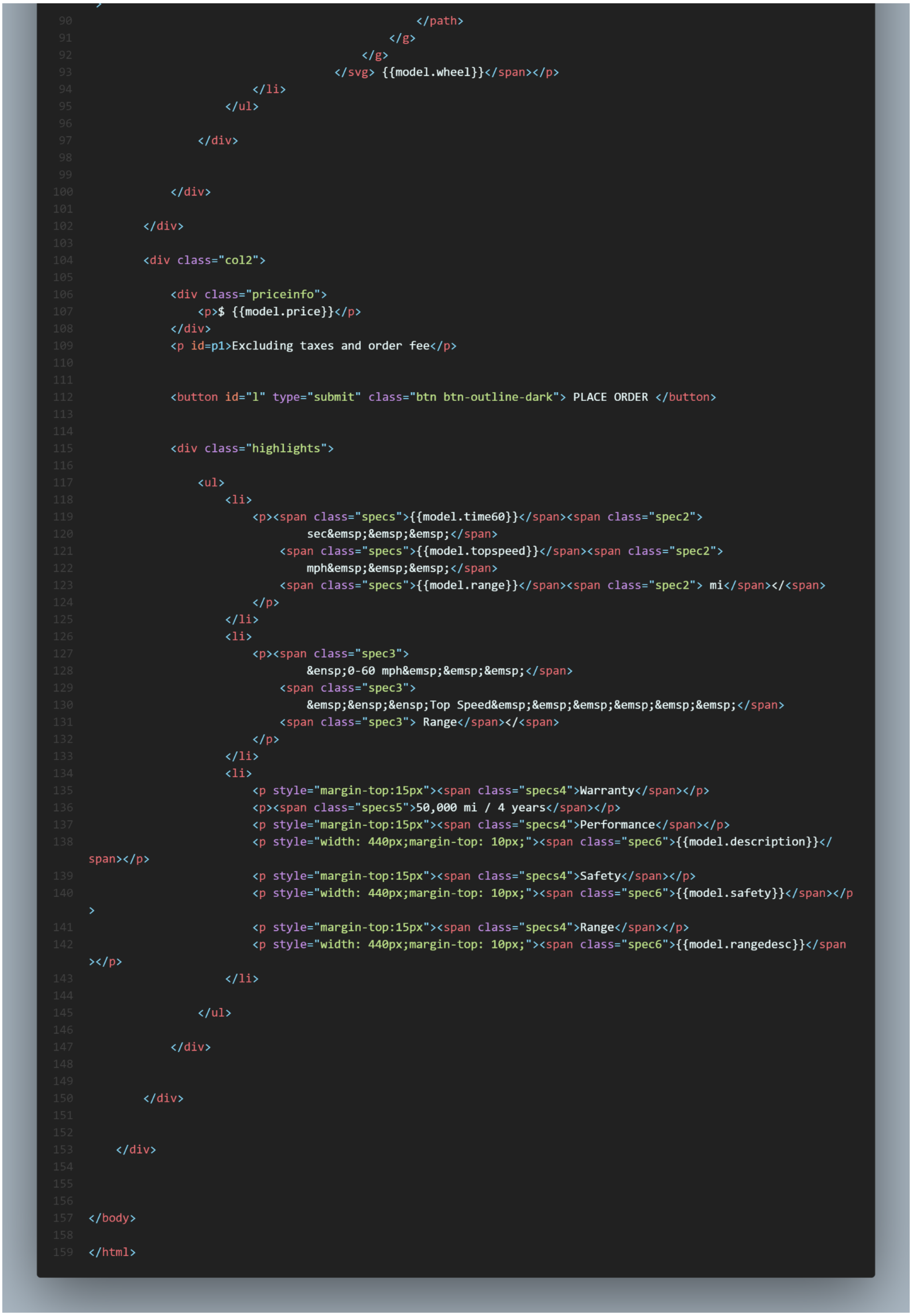


```
var express = require("express");
var router = express.Router();
router.use(express.static("public"));
var Model = require("../models/model");
var Servicing = require("../models/servicing");
var nodemailer = require('nodemailer');

router.get("/shop", function (req, res, next) {
  Model.find(function (err, docs) {
    res.render("shop/index.hbs", { title: "Autorizz", models: docs });
    console.log(docs);
  });
});

router.get("/service", function (req, res, next) {
  Servicing.find(function (err, docs) {
    res.render("servicing/servicing.hbs", { title: "Autorizz", servicecars:
docs });
    console.log(docs);
  });
});


router.get("/booknow/:id", function (req, res, next) {

  var modelid = req.params.id;
  console.log(modelid);

  Model.findById(modelid, function (err, doc) {
    if (err) {
      console.log(err);
    }
    else {
      console.log(doc);
      res.render("shop/booking", { title: "Autorizz", model: doc })
    }

  });
});
```

VEHICLE SALES APP PAGE:

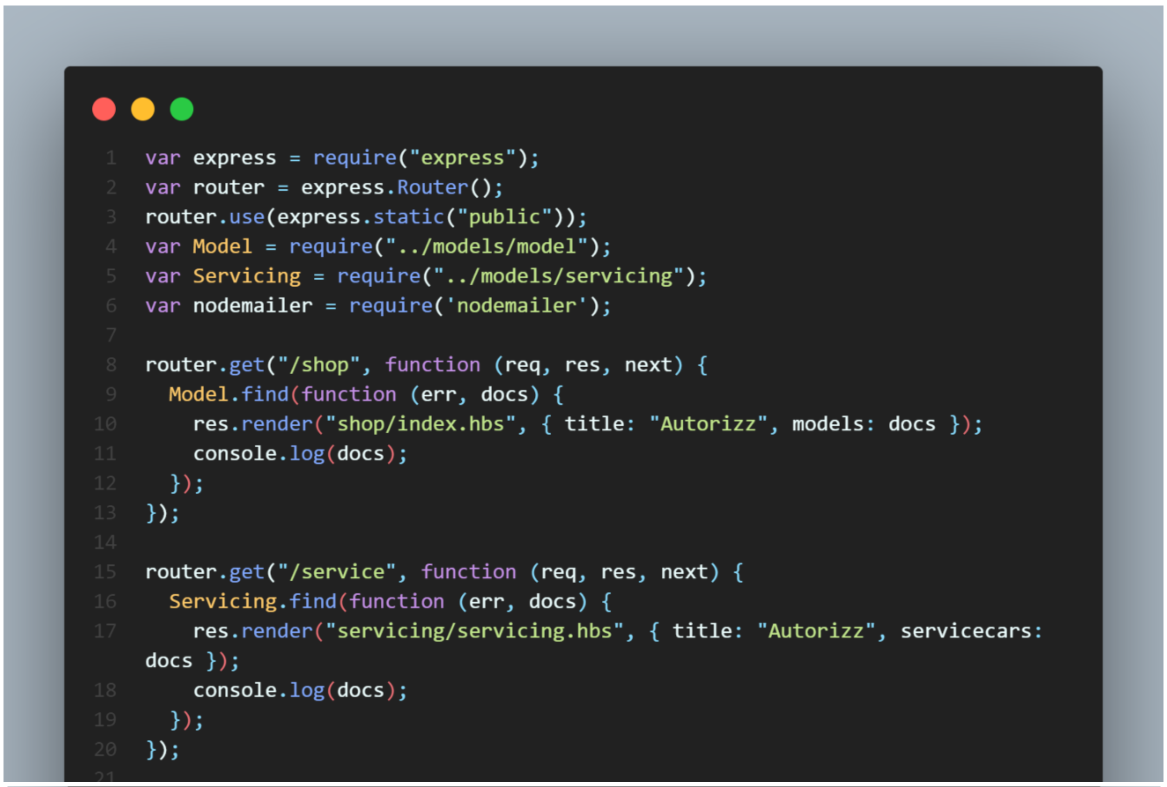


```html
<!DOCTYPE html>
<html lang="en">

<head>
    <link rel="stylesheet" href="/stylesheets/booking.css">
    <link href="https://fonts.googleapis.com/css2?family=PT+Sans+Caption&display=swap" rel="stylesheet">
    <link href="https://fonts.googleapis.com/css2?family=Quicksand&display=swap" rel="stylesheet">

</head>

<body>

    <div class="row">

        <div style="height: 600px;" class="col1">

            <div class="heading">
                <h2>{{model.t1}}</h2>
                <h3>{{model.t2}}</h3>
            </div>


            <div style="max-height: 600px;" class="pics">

                <div id="carouselExampleInterval" class="carousel slide" data-ride="carousel">
                    <div class="carousel-inner">
                        <div class="carousel-item active" data-interval="2000">
                            <img style="width: 500px;height:340px;" src="/{{model.imagePath}}" class="
d-block w-100"
                                alt="...">
                        </div>
                        <div class="carousel-item" data-interval="2000">
                            <img style="width: 500px;height:340px ;" src="/images/1int.jpg" class="d-block w-100
"
                                alt="...">
                        </div>
                    </div>
                    <a class="carousel-control-prev sidebutton" href="#carouselExampleInterval" role="button"
                        data-slide="prev">
                        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                        <span class="sr-only">Previous</span>
                    </a>
                    <a class="carousel-control-next sidebutton" href="#carouselExampleInterval" role="button"
                        data-slide="next">
                        <span class="carousel-control-next-icon" aria-hidden="true"></span>
                        <span class="sr-only">Next</span>
                    </a>
                </div>

                <div style="margin-top: 50px;" class="col1">
                    <p style="margin-top:0px;width:500px"><span class="specs">Key Features</span></p>
                    <ul>
                        <li style="margin:20px;">
                            <p><span class="spec7"><svg style=" width:24px;height:22px" viewBox="0 0 24 24">
                                        <path fill="currentColor"
                                            d="
M10,4H12V6H10V4M7,3H9V5H7V3M7,6H9V8H7V6M6,8V10H4V8H6M6,5V7H4V5H6M6,2V4H4V2H6M13,22A2,2 0 0,1 11,20V10A2,2 0 0,
1 13,8V7H14V4H17V7H18V8A2,2 0 0,1 20,10V20A2,2 0 0,1 18,22H13M13,10V20H18V10H13Z
" />
                                    </svg> {{model.colour}}</span></p>

                        </li>
                        <li style="margin:20px;">
                            <p><span class="spec7"></svg> {{this.colour}}<svg style="width:24px;height:24px"
                                        viewBox="0 0 24 24">
                                        <path fill="currentColor"
                                            d="
M7 18C7 18 4 10 4 6S6 2 6 2H7C7 2 8 2 8 3S7 4 7 6 10 10 10 13 7 18 7 18M12 17C11 17 8 19.5 8 19.5C7.7 19.7 7.8 2
0 8 20.3C8 20.3 9 22.1 11 22.1H17C18.1 22.1 19 21.2 19 20.1V19.1C19 18 18.1 17.1 17 17.1H12Z
" />
```

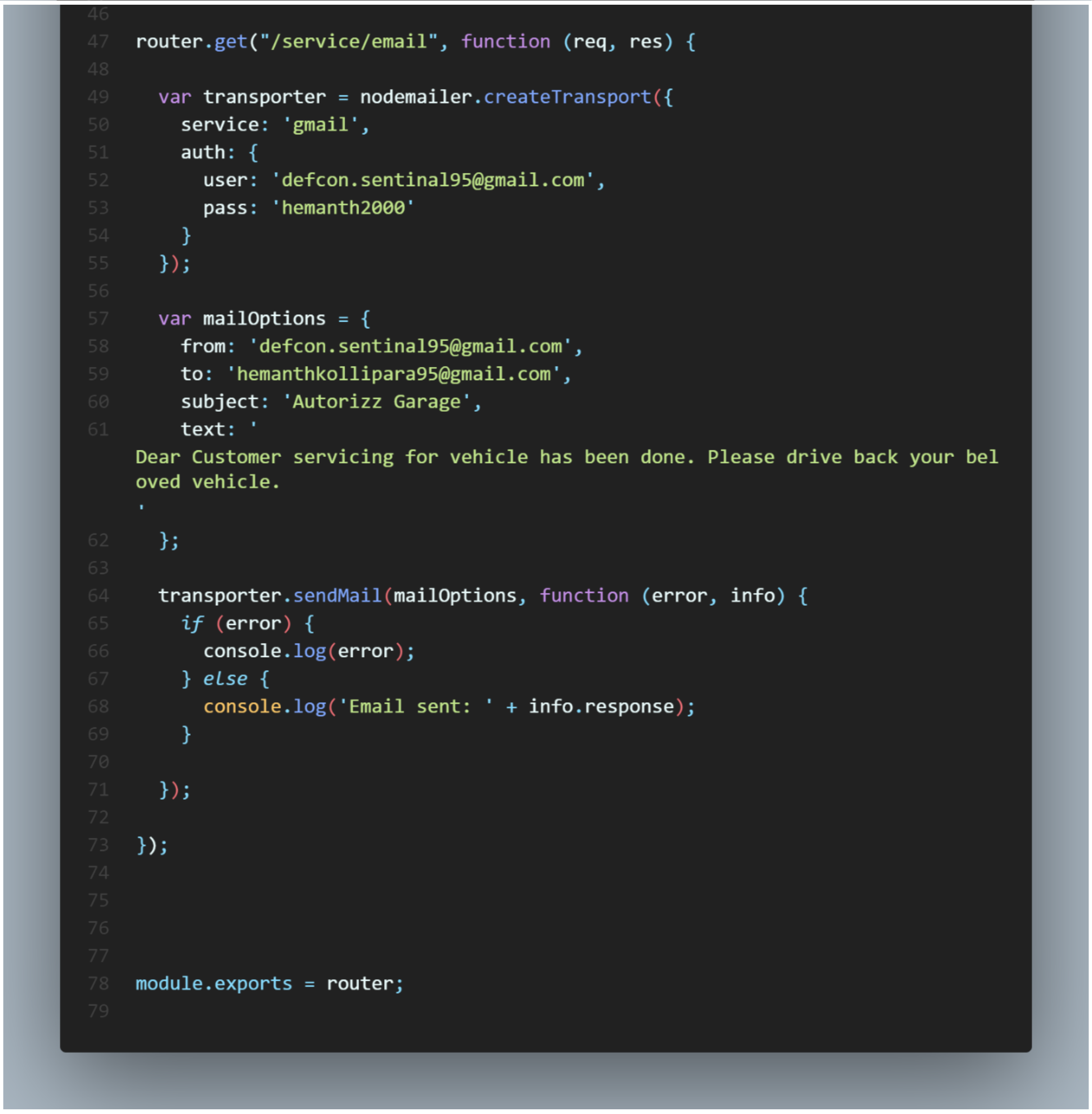


```html
                                                </path>
                                            </g>
                                        </g>
                                    </svg> {{model.wheel}}</span></p>
                        </li>
                    </ul>

                </div>


            </div>

        </div>

        <div class="col2">

            <div class="priceinfo">
                <p>$ {{model.price}}</p>
            </div>
            <p id=p1>Excluding taxes and order fee</p>


            <button id="l" type="submit" class="btn btn-outline-dark"> PLACE ORDER </button>


            <div class="highlights">

                <ul>
                    <li>
                        <p><span class="specs">{{model.time60}}</span><span class="spec2">
                                sec&emsp;&emsp;&emsp;</span>
                            <span class="specs">{{model.topspeed}}</span><span class="spec2">
                                mph&emsp;&emsp;&emsp;</span>
                            <span class="specs">{{model.range}}</span><span class="spec2"> mi</span></<span>
                        </p>
                    </li>
                    <li>
                        <p><span class="spec3">
                                &ensp;0-60 mph&emsp;&emsp;&emsp;</span>
                            <span class="spec3">
                                &emsp;&ensp;&ensp;Top Speed&emsp;&emsp;&emsp;&emsp;&emsp;&emsp;</span>
                            <span class="spec3"> Range</span></<span>
                        </p>
                    </li>
                    <li>
                        <p style="margin-top:15px"><span class="specs4">Warranty</span></p>
                        <p><span class="specs5">50,000 mi / 4 years</span></p>
                        <p style="margin-top:15px"><span class="specs4">Performance</span></p>
                        <p style="width: 440px;margin-top: 10px;"><span class="spec6">{{model.description}}</span></p>
                        <p style="margin-top:15px"><span class="specs4">Safety</span></p>
                        <p style="width: 440px;margin-top: 10px;"><span class="spec6">{{model.safety}}</span></p>
                        <p style="margin-top:15px"><span class="specs4">Range</span></p>
                        <p style="width: 440px;margin-top: 10px;"><span class="spec6">{{model.rangedesc}}</span></p>
                    </li>

                </ul>

            </div>


        </div>


    </div>



</body>

</html>
```

SERVICE MODULE

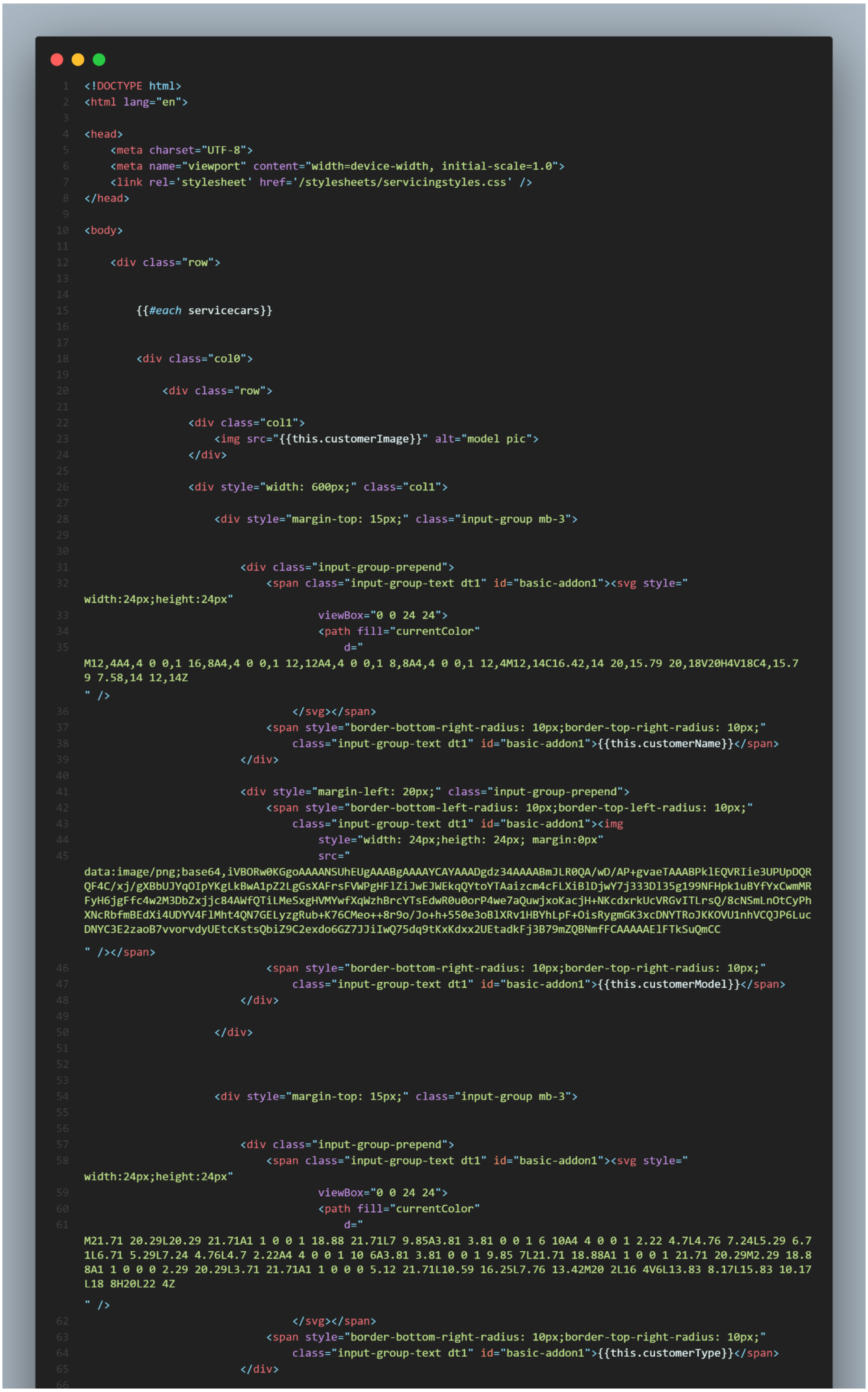


```javascript
var express = require("express");
var router = express.Router();
router.use(express.static("public"));
var Model = require("../models/model");
var Servicing = require("../models/servicing");
var nodemailer = require('nodemailer');

router.get("/shop", function (req, res, next) {
  Model.find(function (err, docs) {
    res.render("shop/index.hbs", { title: "Autorizz", models: docs });
    console.log(docs);
  });
});

router.get("/service", function (req, res, next) {
  Servicing.find(function (err, docs) {
    res.render("servicing/servicing.hbs", { title: "Autorizz", servicecars: docs });
    console.log(docs);
  });
});
```



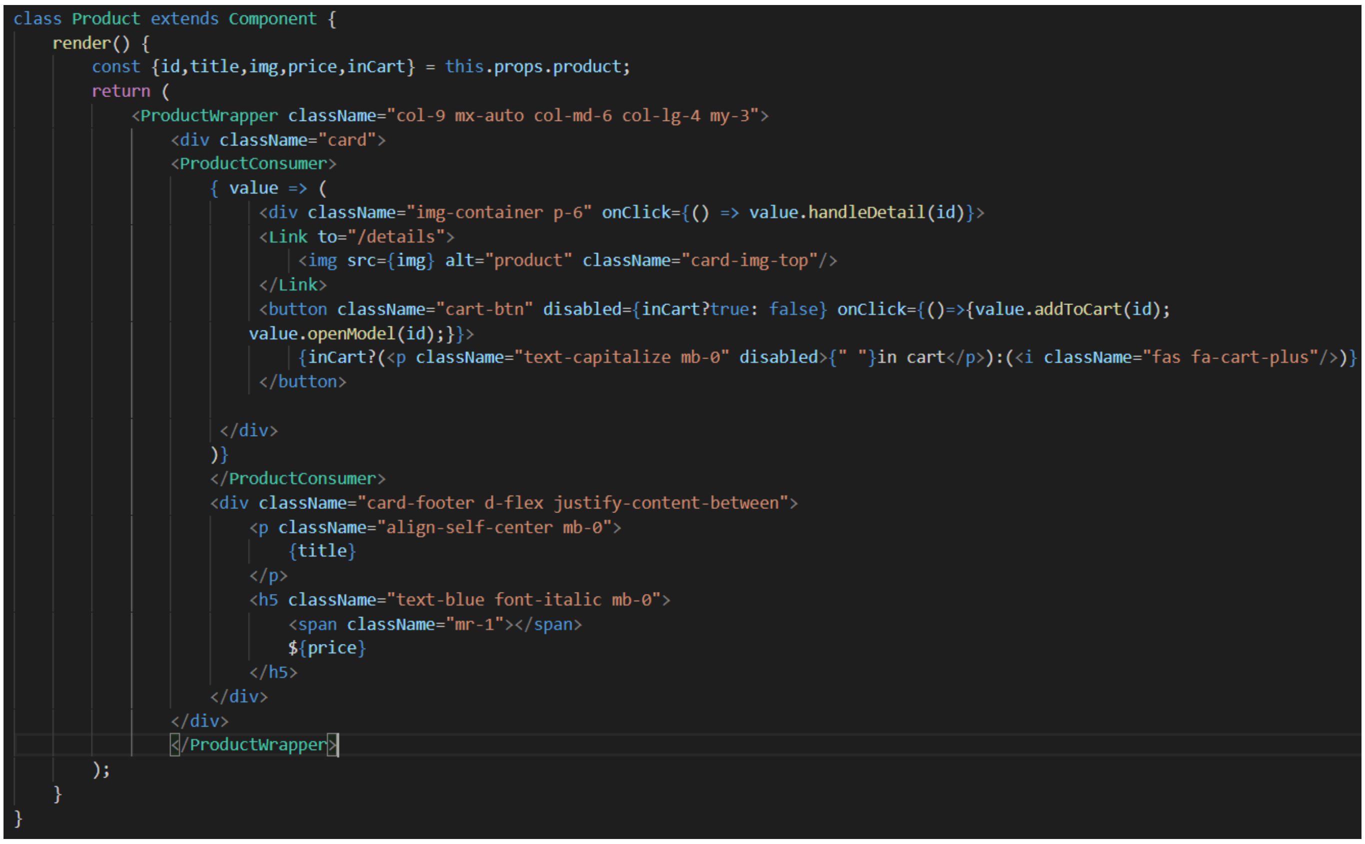
```javascript
router.get("/service/email", function (req, res) {

  var transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
      user: 'defcon.sentinal95@gmail.com',
      pass: 'hemanth2000'
    }
  });

  var mailOptions = {
    from: 'defcon.sentinal95@gmail.com',
    to: 'hemanthkollipara95@gmail.com',
    subject: 'Autorizz Garage',
    text: '
Dear Customer servicing for vehicle has been done. Please drive back your beloved vehicle.
'
  };

  transporter.sendMail(mailOptions, function (error, info) {
    if (error) {
      console.log(error);
    } else {
      console.log('Email sent: ' + info.response);
    }

  });

});




module.exports = router;
```

SERVICE APP PAGE

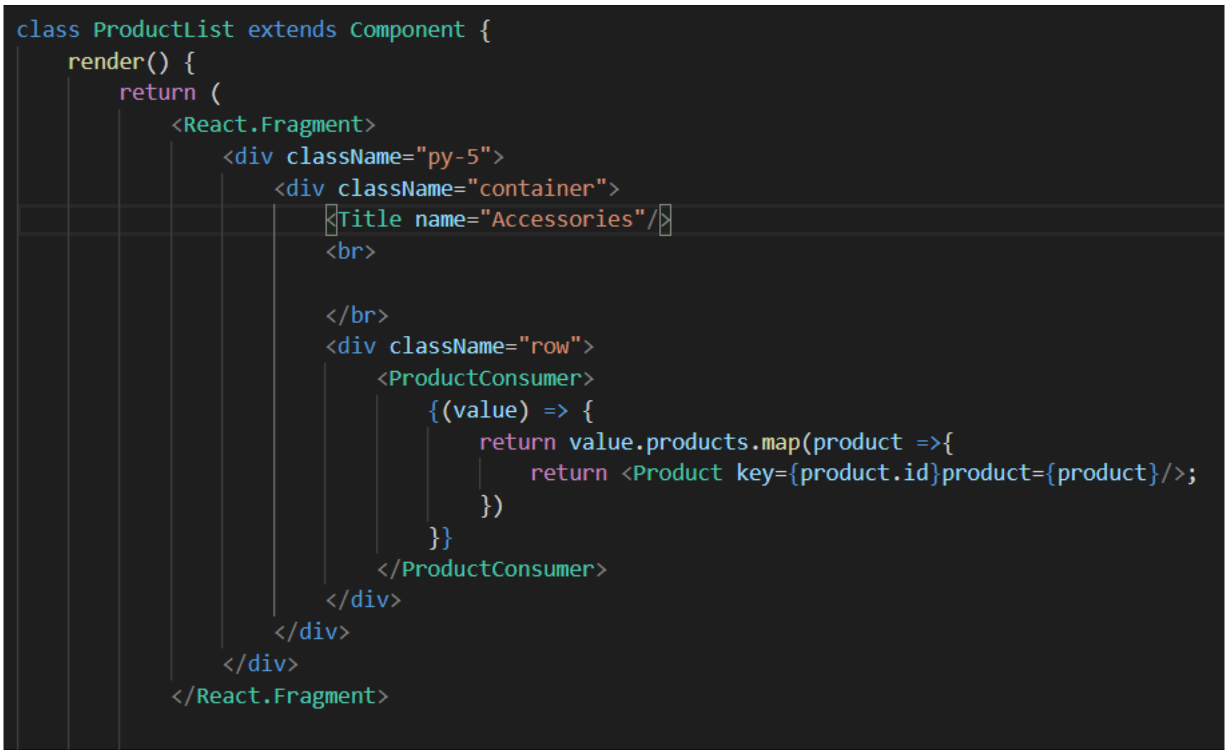


```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel='stylesheet' href='/stylesheets/servicingstyles.css' />
</head>

<body>

    <div class="row">


        {{#each servicecars}}


        <div class="col0">

            <div class="row">

                <div class="col1">
                    <img src="{{this.customerImage}}" alt="model pic">
                </div>

                <div style="width: 600px;" class="col1">

                    <div style="margin-top: 15px;" class="input-group mb-3">


                        <div class="input-group-prepend">
                            <span class="input-group-text dt1" id="basic-addon1"><svg style="
width:24px;height:24px"
                                    viewBox="0 0 24 24">
                                    <path fill="currentColor"
                                        d="
M12,4A4,4 0 0,1 16,8A4,4 0 0,1 12,12A4,4 0 0,1 8,8A4,4 0 0,1 12,4M12,14C16.42,14 20,15.79 20,18V20H4V18C4,15.7
9 7.58,14 12,14Z
" />
                                </svg></span>
                            <span style="border-bottom-right-radius: 10px;border-top-right-radius: 10px;"
                                class="input-group-text dt1" id="basic-addon1">{{this.customerName}}</span>
                        </div>

                        <div style="margin-left: 20px;" class="input-group-prepend">
                            <span style="border-bottom-left-radius: 10px;border-top-left-radius: 10px;"
                                class="input-group-text dt1" id="basic-addon1"><img
                                    style="width: 24px;heigth: 24px; margin:0px"
                                    src="
data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABgAAAAYCAYAAADgdz34AAAABmJLR0QA/wD/AP+gvaeTAAABPklEQVRIie3UPUpDQR
QF4C/xj/gXBbUJYqOIpYKgLkBwA1pZ2LgGsXAFrsFVWPgHFlZiJwEJWEkqQYtoYTAaizcm4cFLXiBlDjwY7j333Dl35g199NFHpk1uBYfYxCwmMR
FyH6jgFfc4w2M3DbZxjjc84AWfQTiLMeSxgHVMYwfXqWzhBrcYTsEdwR0u0orP4we7aQuwjxoKacjH+NKcdxrkUcVRGvITLrsQ/8cNSmLnOtCyPh
XNcRbfmBEdXi4UDYV4FlMht4QN7GELyzgRub+K76CMeo++8r9o/Jo+h+550e3oBlXRv1HBYhLpF+OisRygmGK3xcDNYTRoJKKOVU1nhVCQJP6Luc
DNYC3E2zaoB7vvorvdyUEtcKstsQbiZ9C2exdo6GZ7JJiIwQ75dq9tKxKdxx2UEtadkFj3B79mZQBNmfFCAAAAAElFTkSuQmCC
" /></span>
                            <span style="border-bottom-right-radius: 10px;border-top-right-radius: 10px;"
                                class="input-group-text dt1" id="basic-addon1">{{this.customerModel}}</span>
                        </div>

                    </div>



                    <div style="margin-top: 15px;" class="input-group mb-3">


                        <div class="input-group-prepend">
                            <span class="input-group-text dt1" id="basic-addon1"><svg style="
width:24px;height:24px"
                                    viewBox="0 0 24 24">
                                    <path fill="currentColor"
                                        d="
M21.71 20.29L20.29 21.71A1 1 0 0 1 18.88 21.71L7 9.85A3.81 3.81 0 0 1 6 10A4 4 0 0 1 2.22 4.7L4.76 7.24L5.29 6.7
1L6.71 5.29L7.24 4.76L4.7 2.22A4 4 0 0 1 10 6A3.81 3.81 0 0 1 9.85 7L21.71 18.88A1 1 0 0 1 21.71 20.29M2.29 18.8
8A1 1 0 0 0 2.29 20.29L3.71 21.71A1 1 0 0 0 5.12 21.71L10.59 16.25L7.76 13.42M20 2L16 4V6L13.83 8.17L15.83 10.17
L18 8H20L22 4Z
" />
                                </svg></span>
                            <span style="border-bottom-right-radius: 10px;border-top-right-radius: 10px;"
                                class="input-group-text dt1" id="basic-addon1">{{this.customerType}}</span>
                        </div>
```

VIEW PAGE

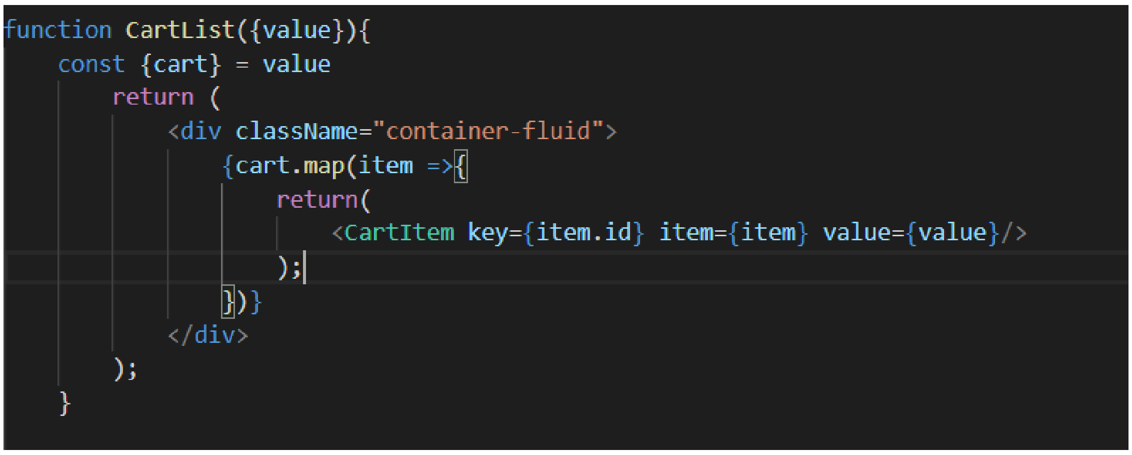


```
class Product extends Component {
    render() {
        const {id,title,img,price,inCart} = this.props.product;
        return (
            <ProductWrapper className="col-9 mx-auto col-md-6 col-lg-4 my-3">
                <div className="card">
                <ProductConsumer>
                    { value => (
                        <div className="img-container p-6" onClick={() => value.handleDetail(id)}>
                        <Link to="/details">
                            <img src={img} alt="product" className="card-img-top"/>
                        </Link>
                        <button className="cart-btn" disabled={inCart?true: false} onClick={()=>{value.addToCart(id);
                       value.openModel(id);}}>
                            {inCart?(<p className="text-capitalize mb-0" disabled>{" "}in cart</p>):(<i className="fas fa-cart-plus"/>)}
                        </button>

                      </div>
                    )}
                    </ProductConsumer>
                    <div className="card-footer d-flex justify-content-between">
                        <p className="align-self-center mb-0">
                            {title}
                        </p>
                        <h5 className="text-blue font-italic mb-0">
                            <span className="mr-1"></span>
                            ${price}
                        </h5>
                    </div>
                </div>
                </ProductWrapper>
        );
    }
}
```



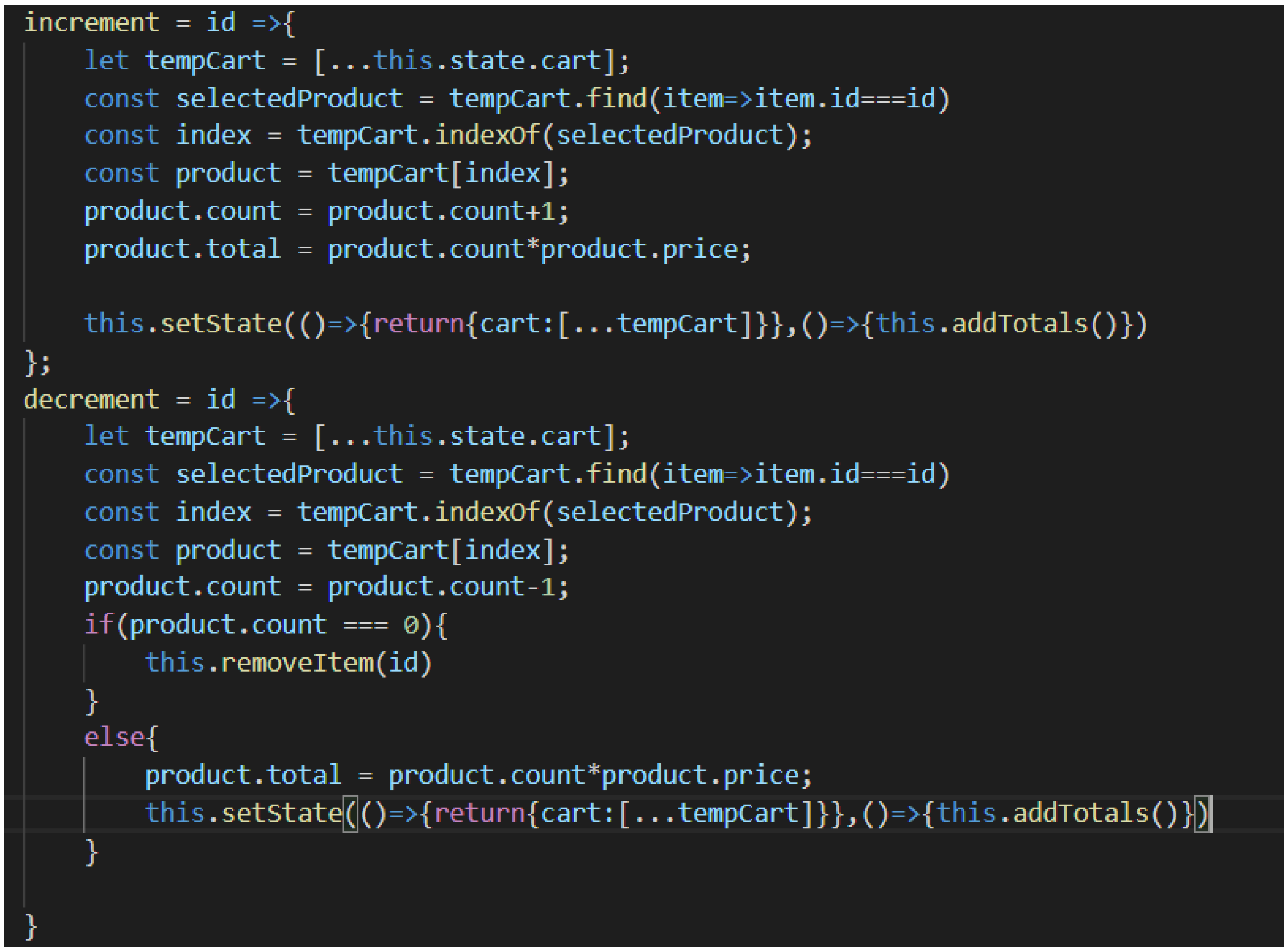
```
class ProductList extends Component {
    render() {
        return (
            <React.Fragment>
                <div className="py-5">
                    <div className="container">
                        <Title name="Accessories"/>
                        <br>

                        </br>
                        <div className="row">
                            <ProductConsumer>
                                {(value) => {
                                    return value.products.map(product =>{
                                        return <Product key={product.id}product={product}/>;
                                    })
                                }}
                            </ProductConsumer>
                        </div>
                    </div>
                </div>
            </React.Fragment>
```

VIEW PURCHASE FUNCTION



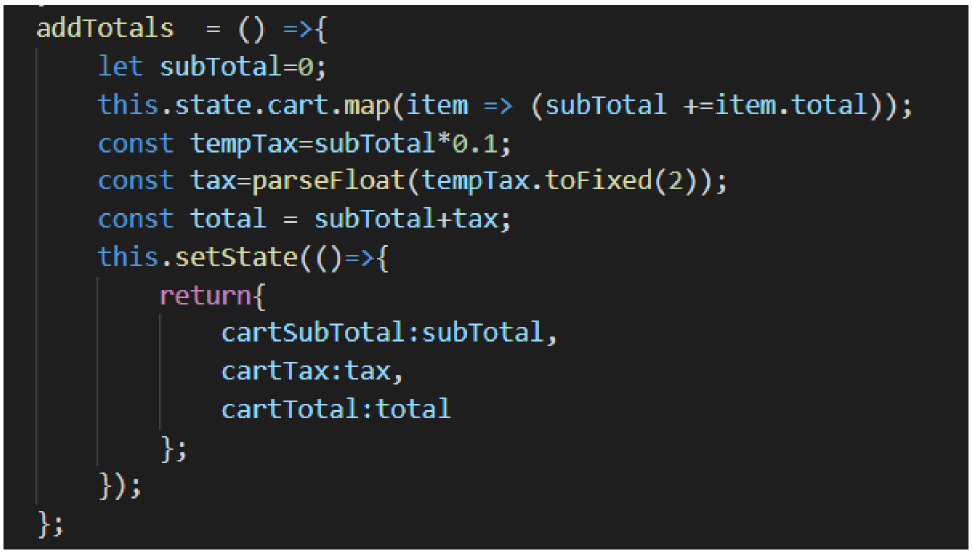
```
function CartList({value}){
    const {cart} = value
        return (
            <div className="container-fluid">
                {cart.map(item =>{
                    return(
                        <CartItem key={item.id} item={item} value={value}/>
                    );
                })}
            </div>
        );
    }
```

UPDATING CART MODULE

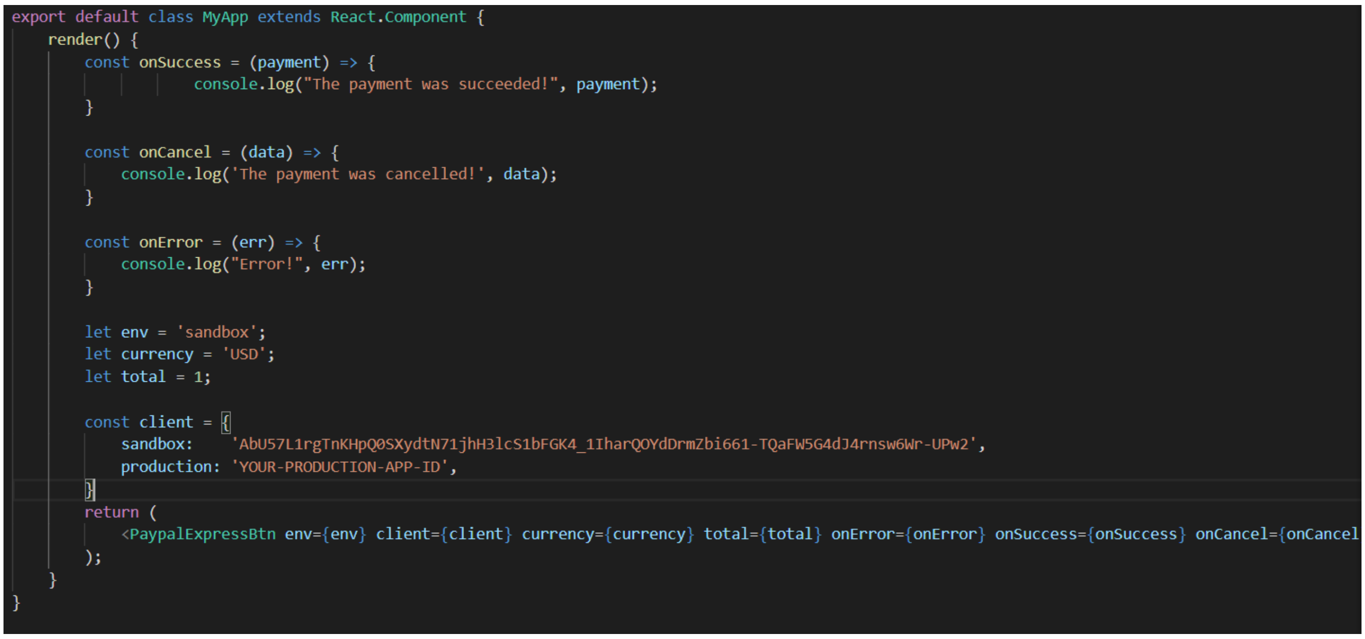


```
increment = id =>{
    let tempCart = [...this.state.cart];
    const selectedProduct = tempCart.find(item=>item.id===id)
    const index = tempCart.indexOf(selectedProduct);
    const product = tempCart[index];
    product.count = product.count+1;
    product.total = product.count*product.price;

    this.setState(()=>{return{cart:[...tempCart]}},()=>{this.addTotals()})
};
decrement = id =>{
    let tempCart = [...this.state.cart];
    const selectedProduct = tempCart.find(item=>item.id===id)
    const index = tempCart.indexOf(selectedProduct);
    const product = tempCart[index];
    product.count = product.count-1;
    if(product.count === 0){
        this.removeItem(id)
    }
    else{
        product.total = product.count*product.price;
        this.setState(()=>{return{cart:[...tempCart]}},()=>{this.addTotals()})
    }

}
```

BILLING PAGE:

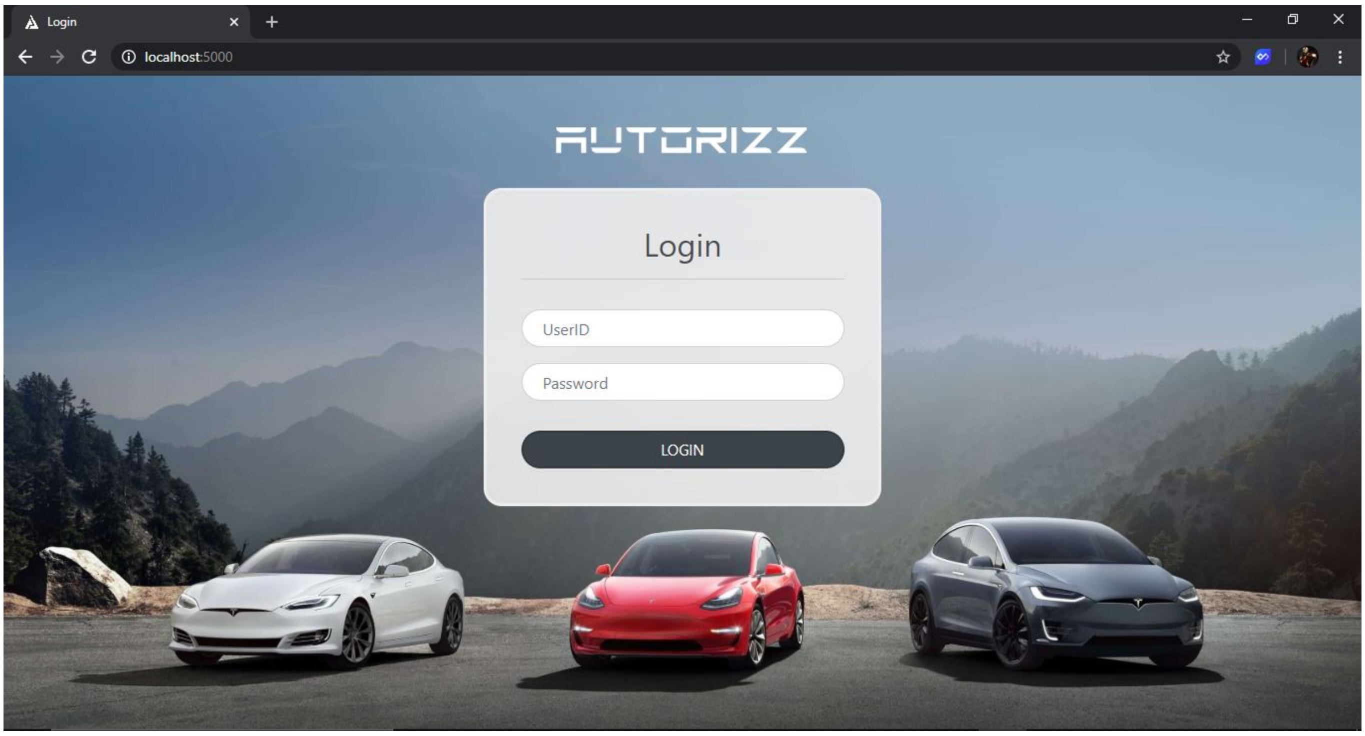

```
addTotals  = () =>{
    let subTotal=0;
    this.state.cart.map(item => (subTotal +=item.total));
    const tempTax=subTotal*0.1;
    const tax=parseFloat(tempTax.toFixed(2));
    const total = subTotal+tax;
    this.setState(()=>{
        return{
            cartSubTotal:subTotal,
            cartTax:tax,
            cartTotal:total
        };
    });
};
```

PAYMENT HANDLING FUNCTION

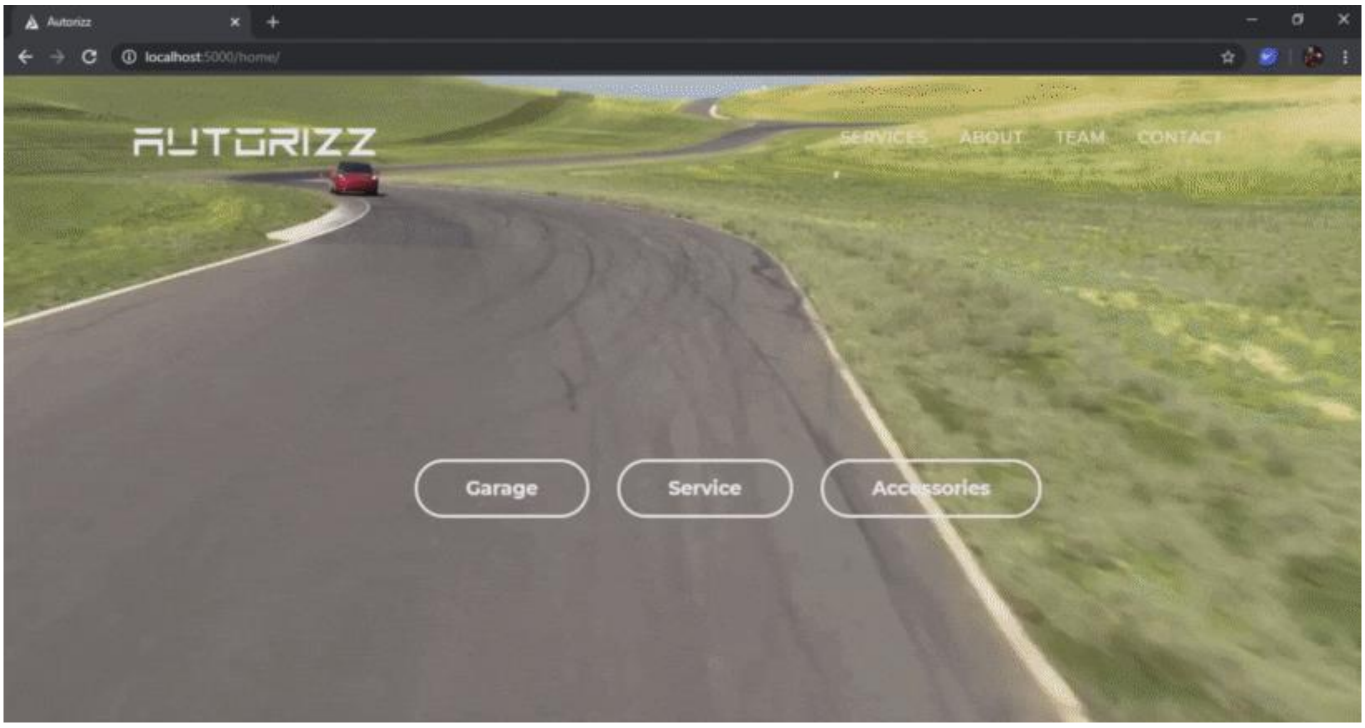

```
export default class MyApp extends React.Component {
    render() {
        const onSuccess = (payment) => {
                    console.log("The payment was succeeded!", payment);
        }

        const onCancel = (data) => {
            console.log('The payment was cancelled!', data);
        }

        const onError = (err) => {
            console.log("Error!", err);
        }

        let env = 'sandbox';
        let currency = 'USD';
        let total = 1;

        const client = {
            sandbox:    'AbU57L1rgTnKHpQ0SXydtN71jhH3lcS1bFGK4_1IharQOYdDrmZbi661-TQaFW5G4dJ4rnsw6Wr-UPw2',
            production: 'YOUR-PRODUCTION-APP-ID',
        }
        return (
            <PaypalExpressBtn env={env} client={client} currency={currency} total={total} onError={onError} onSuccess={onSuccess} onCancel={onCancel
        );
    }
}
```

OUTPUT SCREEN

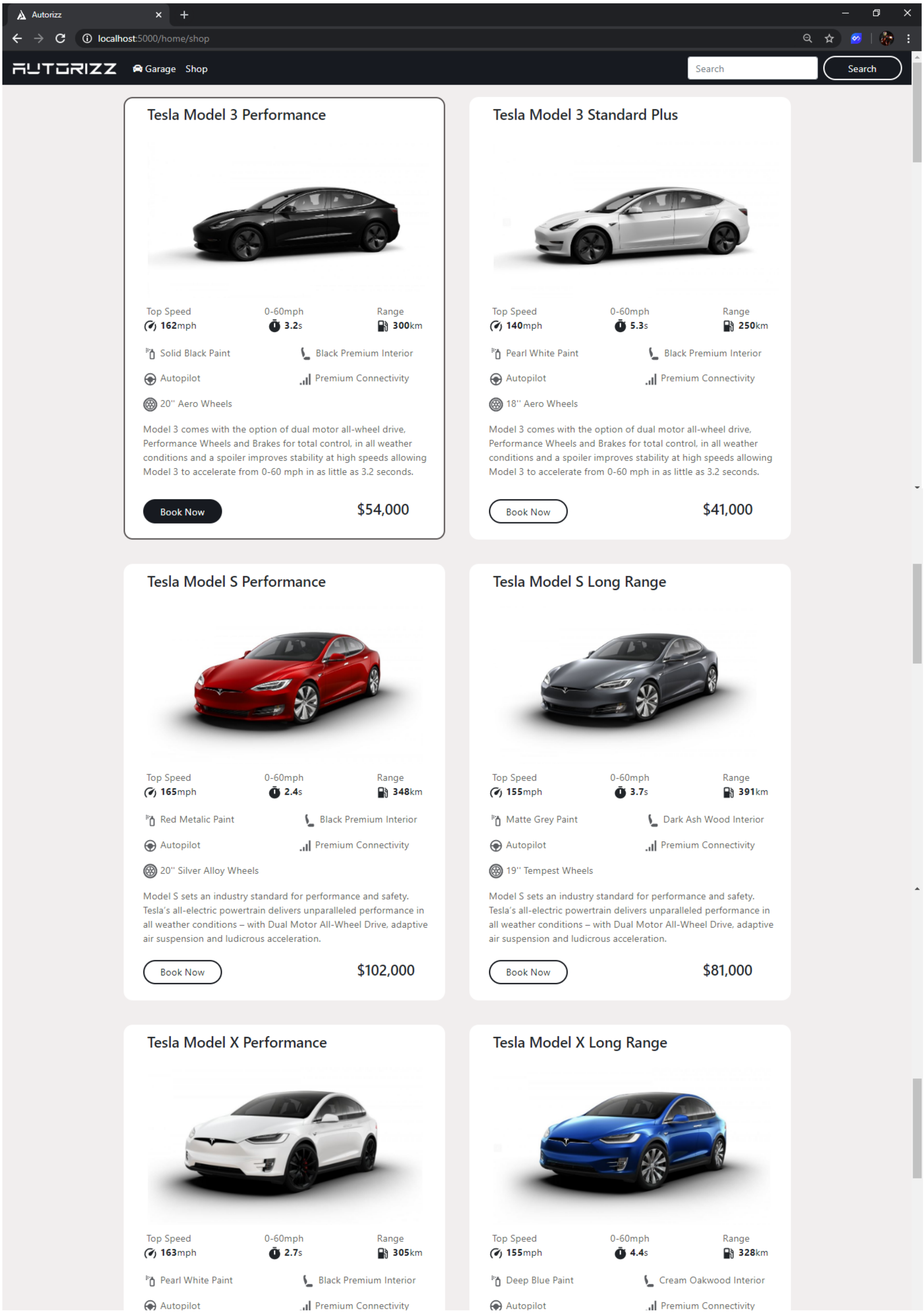
LOGIN PAGE



HOME PAGE



VEHICLE SALES PAGE



Autorizz

localhost:5000/home/shop

AUTORIZZ Garage Shop Search Search

Tesla Model 3 Performance

Top Speed 162mph | 0-60mph 3.2s | Range 300km

Solid Black Paint | Black Premium Interior

Autopilot | Premium Connectivity

20'' Aero Wheels

Model 3 comes with the option of dual motor all-wheel drive, Performance Wheels and Brakes for total control, in all weather conditions and a spoiler improves stability at high speeds allowing Model 3 to accelerate from 0-60 mph in as little as 3.2 seconds.

Book Now $54,000

Tesla Model 3 Standard Plus

Top Speed 140mph | 0-60mph 5.3s | Range 250km

Pearl White Paint | Black Premium Interior

Autopilot | Premium Connectivity

18'' Aero Wheels

Model 3 comes with the option of dual motor all-wheel drive, Performance Wheels and Brakes for total control, in all weather conditions and a spoiler improves stability at high speeds allowing Model 3 to accelerate from 0-60 mph in as little as 3.2 seconds.

Book Now $41,000

Tesla Model S Performance

Top Speed 165mph | 0-60mph 2.4s | Range 348km

Red Metalic Paint | Black Premium Interior

Autopilot | Premium Connectivity

20'' Silver Alloy Wheels

Model S sets an industry standard for performance and safety. Tesla's all-electric powertrain delivers unparalleled performance in all weather conditions – with Dual Motor All-Wheel Drive, adaptive air suspension and ludicrous acceleration.

Book Now $102,000

Tesla Model S Long Range

Top Speed 155mph | 0-60mph 3.7s | Range 391km

Matte Grey Paint | Dark Ash Wood Interior

Autopilot | Premium Connectivity

19'' Tempest Wheels

Model S sets an industry standard for performance and safety. Tesla's all-electric powertrain delivers unparalleled performance in all weather conditions – with Dual Motor All-Wheel Drive, adaptive air suspension and ludicrous acceleration.

Book Now $81,000

Tesla Model X Performance

Top Speed 163mph | 0-60mph 2.7s | Range 305km

Pearl White Paint | Black Premium Interior

Autopilot | Premium Connectivity

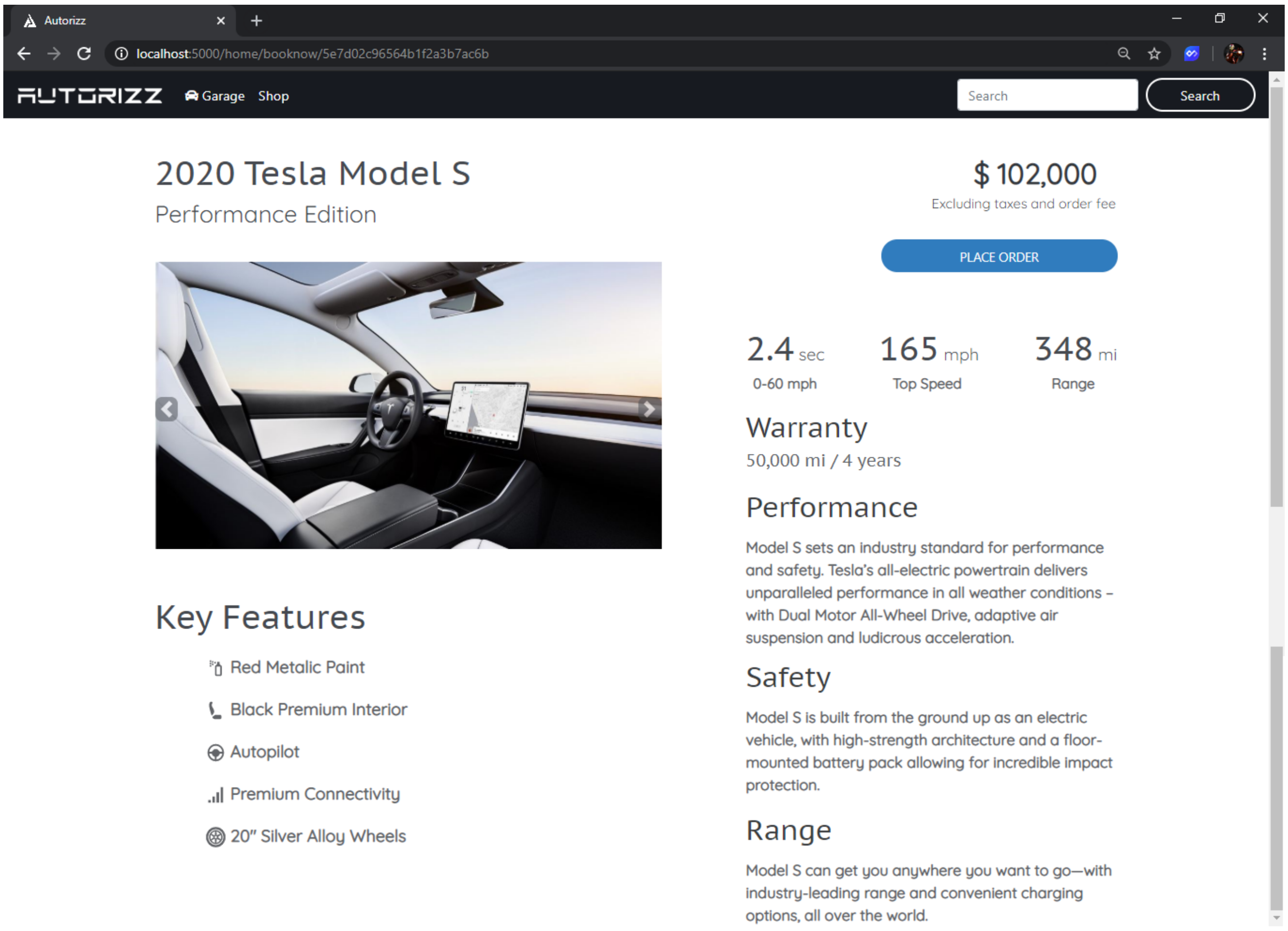
Tesla Model X Long Range

Top Speed 155mph | 0-60mph 4.4s | Range 328km

Deep Blue Paint | Cream Oakwood Interior

Autopilot | Premium Connectivity

BOOKING PAGE



Autorizz

localhost:5000/home/booknow/5e7d02c96564b1f2a3b7ac6b

AUTORIZZ Garage Shop Search Search

2020 Tesla Model S

Performance Edition

$ 102,000

Excluding taxes and order fee

PLACE ORDER

2.4 sec 0-60 mph

165 mph Top Speed

348 mi Range

Warranty

50,000 mi / 4 years

Performance

Model S sets an industry standard for performance and safety. Tesla's all-electric powertrain delivers unparalleled performance in all weather conditions – with Dual Motor All-Wheel Drive, adaptive air suspension and ludicrous acceleration.

Safety

Model S is built from the ground up as an electric vehicle, with high-strength architecture and a floor-mounted battery pack allowing for incredible impact protection.

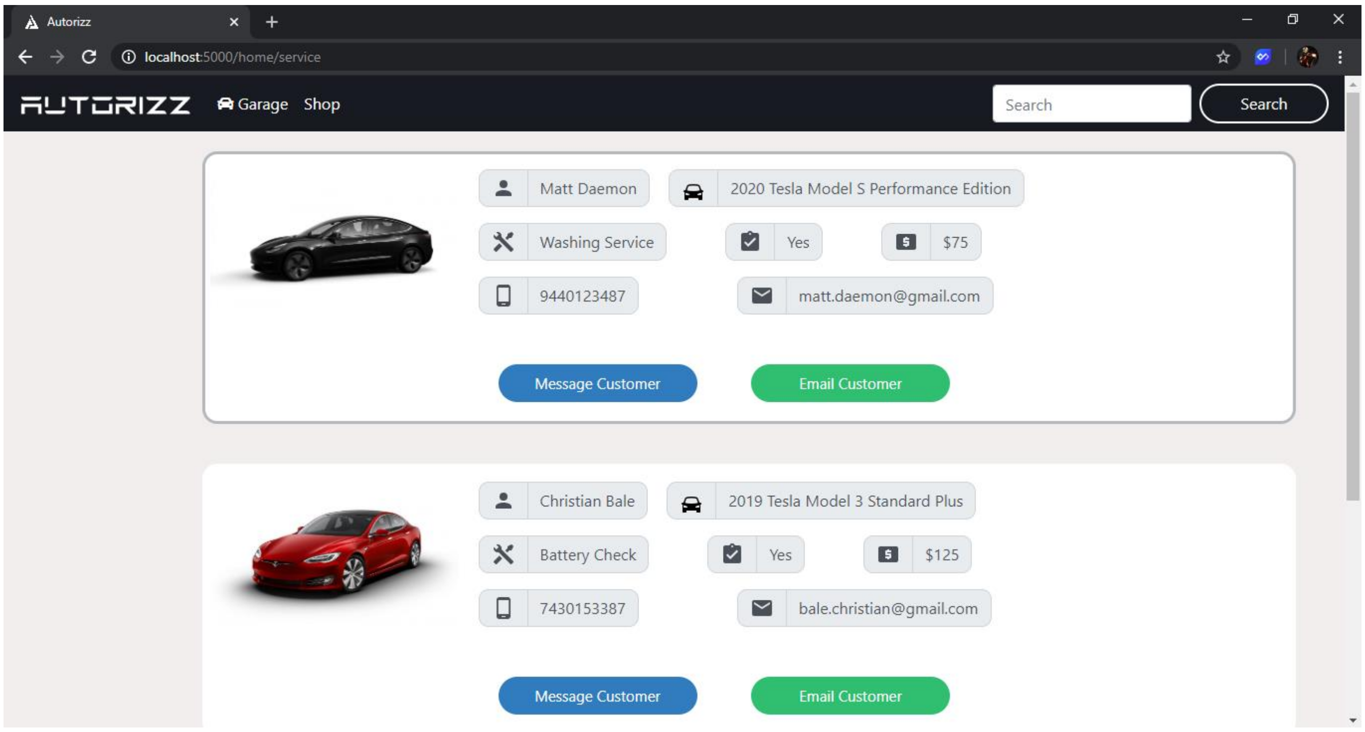
Range

Model S can get you anywhere you want to go—with industry-leading range and convenient charging options, all over the world.

Key Features

- Red Metalic Paint
- Black Premium Interior
- Autopilot
- Premium Connectivity
- 20" Silver Alloy Wheels

VEHICLE SERVICE PAGE



Autorizz

localhost:5000/home/service

AUTORIZZ Garage Shop Search Search

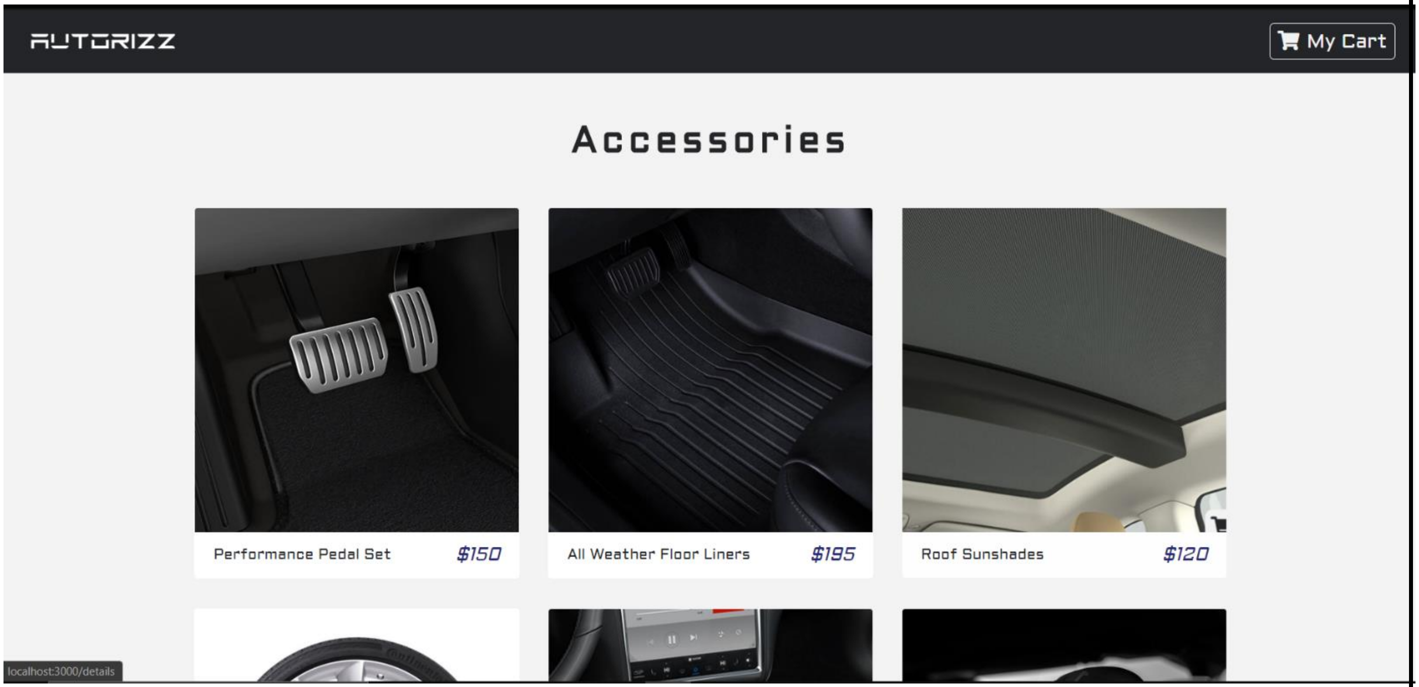
Matt Daemon	2020 Tesla Model S Performance Edition	
Washing Service	Yes	$75
9440123487	matt.daemon@gmail.com	

Message Customer Email Customer

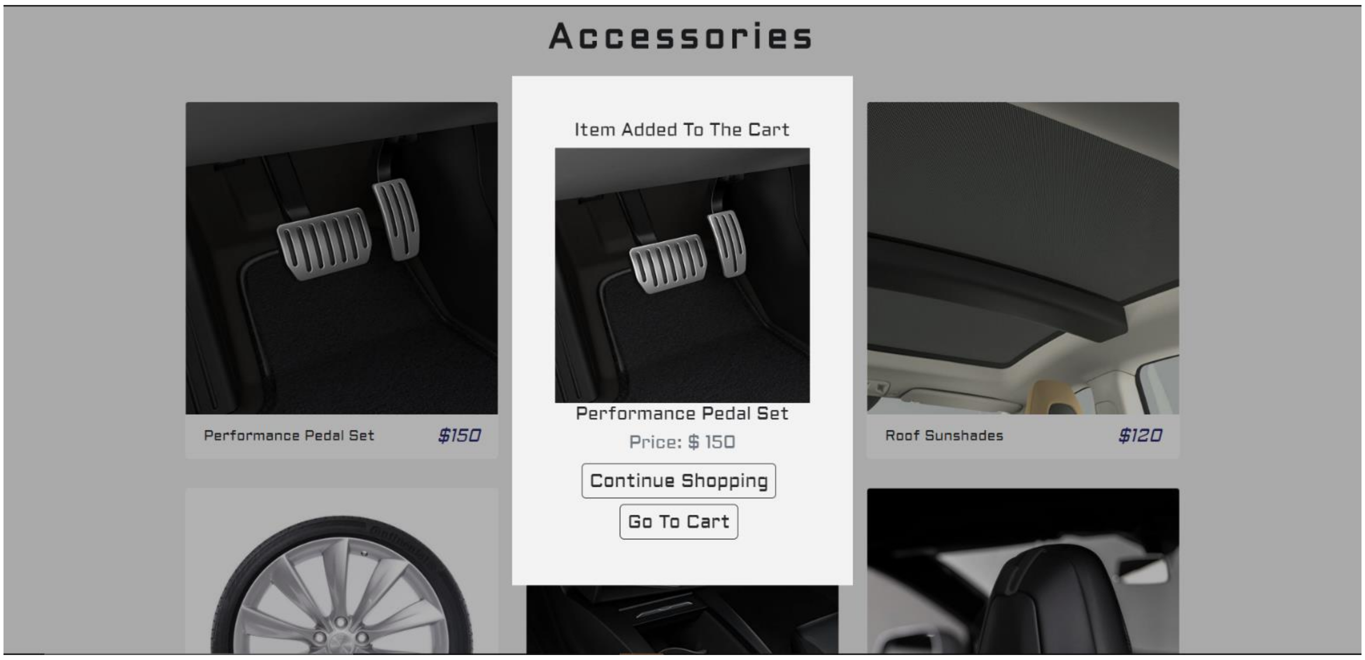
Christian Bale	2019 Tesla Model 3 Standard Plus	
Battery Check	Yes	$125
7430153387	bale.christian@gmail.com	

Message Customer Email Customer

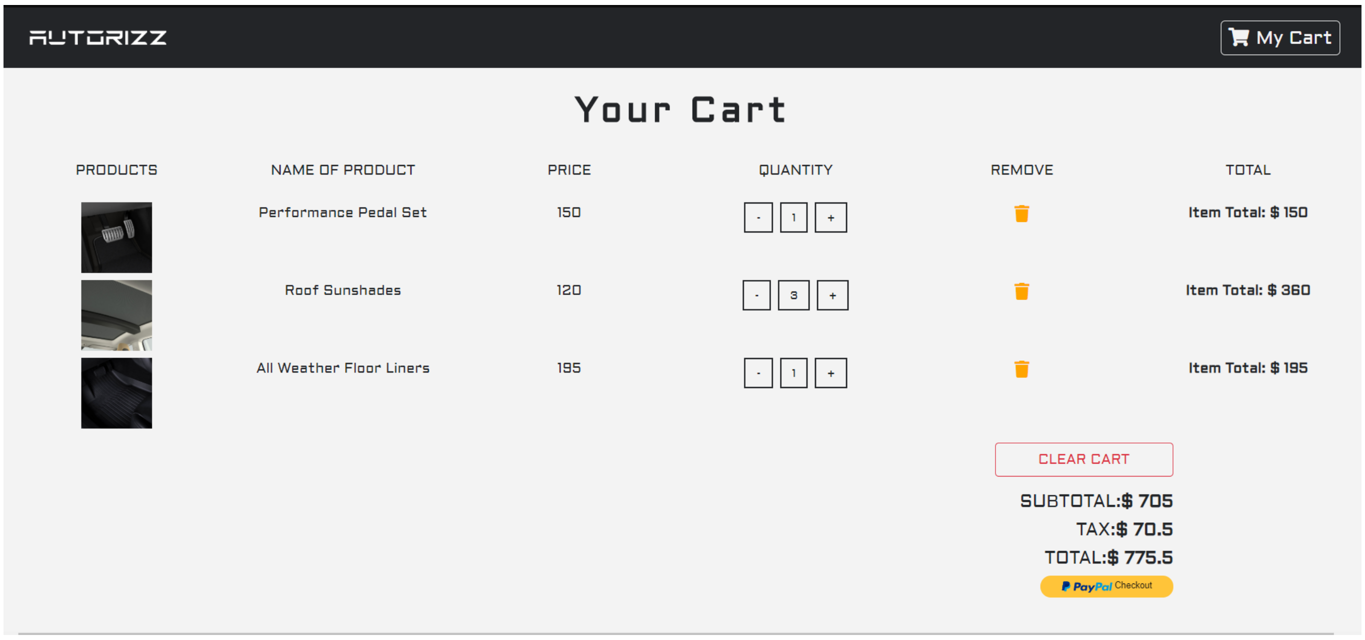
ACCESSORIES PAGE



CART PAGE



CART AND PAYMENT


AUTORIZZ
My Cart

Your Cart

PRODUCTS	NAME OF PRODUCT	PRICE	QUANTITY	REMOVE	TOTAL
	Performance Pedal Set	150	- 1 +	🗑	Item Total: $ 150
	Roof Sunshades	120	- 3 +	🗑	Item Total: $ 360
	All Weather Floor Liners	195	- 1 +	🗑	Item Total: $ 195

CLEAR CART
SUBTOTAL:$ 705
TAX:$ 70.5
TOTAL:$ 775.5
PayPal Checkout

CONCLUSION

Thus, we have successfully build automobile dealership management software that facilitates the management of an automobile dealership with the help of sub systems each handling different functionalities like vehicle sales which maintains and tracks all the details regarding the sale of vehicles, service which tracks the vehicles that are currently being serviced by the dealership, spare inventory which maintains the stocks of the spares in the inventory. This approach of modular solution helps in resolving the bugs faster and increases the pace of development since these can be developed separated but can be tested together. All this combine to ultimately provide the management of information for the dealership so that the dealer could manage business in an effortless manner.

SOFTWARE REQUIREMENTS:

FRONTEND RESOURCES:

- **HTML 5**

  It is a standard markup language that is used for creating the website and provides the base structure and layout for a web page that is displayed in a browser.
- **CSS 3**

  It is a stylesheet language used to describe how the html elements should be displayed on the screen.
- **JavaScript ES6**

  It is a high-level interpreted scripting language that gives behavior to the website to give a dynamic and interactive experience for the user by manipulating the html elements of the webpage.
- **jQuery v3.4.1**

  It is an opensource fast and concise JavaScript library designed to simplify html element manipulation and event handling.
- **Chrome browser**

  A browser is a software application that can interpret and display HTML Web pages, applications, JavaScript and other content.

BACKEND RESOURCES:

- **Node.js 10.16.3LTS**

  It is an opensource JavaScript runtime server environment that can execute JavaScript code to generate dynamic page content by modifying files on the server and can collect user data from the website , transmit it to the server and also can modify data in the database.
- **Express.js v4.17.1**

  Express.js is an essential web framework of Node.js. Express is a layer built on the top of the Node.js that helps manage a server and routes that provides robust features to develop web applications.

DATABASE RESOURCES:

- **MongoDB v4.2.0**

  MongoDB is a cross-platform document-oriented , dynamic and scalable database program. It is a NoSQL database program that uses JSON-like documents with schema. Unlike SQL it stores data in the form of JSON instead of 2-dimensional row-column table.

- **Mongoose v5.7.5**

  It is an object data modelling library for MongoDB and Node.js. Since Node.js can only access MongoDB through its native MongoDB driver Mongoose is used to translate between objects in code and the representation of those objects in MongoDB. It basically simplifies writing the MongoDB validation and boilerplate code.

HARDWARE REQUIREMENTS:

- PROCESSOR: At least Intel Core 2 Duo processor and above or

  At least Athlon 64 FX-60 processor and above.
- RAM: At least 2 GB of RAM
- HARD DISK SPACE: At least 20GB of disk space