Kingdom of Saudi Arabia
Technical and Vocational Training Corporation
Madinah College of Technology
Department of Computer Technology

المملكة العربية السعودية
المؤسسة العامة للتدريب التقني والمهني
الكلية التقنية بالمدينة المنورة
قسم تقنية الحاسب الآلي

عنوان المشروع: متجر إلكتروني

المجموعة:

أيمن سعود سلمان العوفي

هايل جبران

سامر علي

الأرقام التدريبية:

440113203

440121685

439207202

المشرف: م.عمار جوانة

سنة تقديم المشروع: (2022\1443)

بسم الله الرحمن الرحيم

شكر وتقدير

بسم الله الرحمن الرحيم، والحمد لله رب العالمين الذي وفقنا وأعاننا على إنهاء هذا البحث وانطلاقًا من مبدأ أنه لا يشكر الله من لا يشكر الناس، فإننا نتوجه بالشكر الجزيل للمهندس (عمار جوانة) الذي رافقنا في مسيرتنا لإنجاز هذا البحث، ونشكر الأهل والأصدقاء وكل من قدم لنا الدعم المادي أو المعنوي.

الفهرس

جدول الرسومات التوضيحية

المقدمة

بسم الله الرحمن الرحيم، نبدأ تصميم مشروع وهو عبارة عن متجر للبيع والشراء متكون من HTML, Bootstrap, node js.

الفصل الأول:

البحث:

المتطلبات:

1. Business Roles
   - Anonymous users can show products
   - Products can be filtered by category
   - Only Registered users can add products to cart
   - User can view and modify his cart
   - Only Registered users can make orders
   - Registered users can see their orders and track their status
   - Admins can add products
   - Admins can view and manage all users orders
2. Business Scopes
   - Admin
     - Add product
     - Manage user's orders
   - Registered Users
     - View & filter products
     - Add products to cart
     - Manage cart

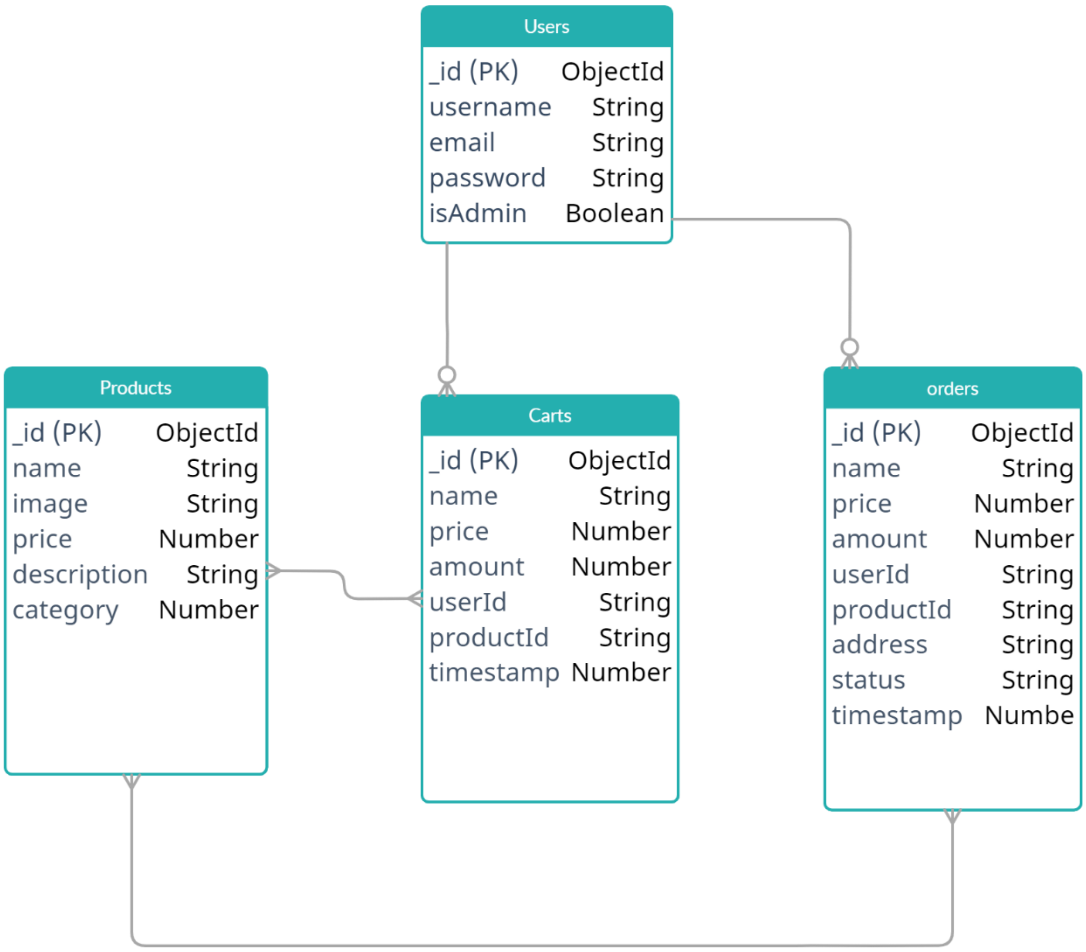
- Make orders
- Track orders

• Anonymous Users

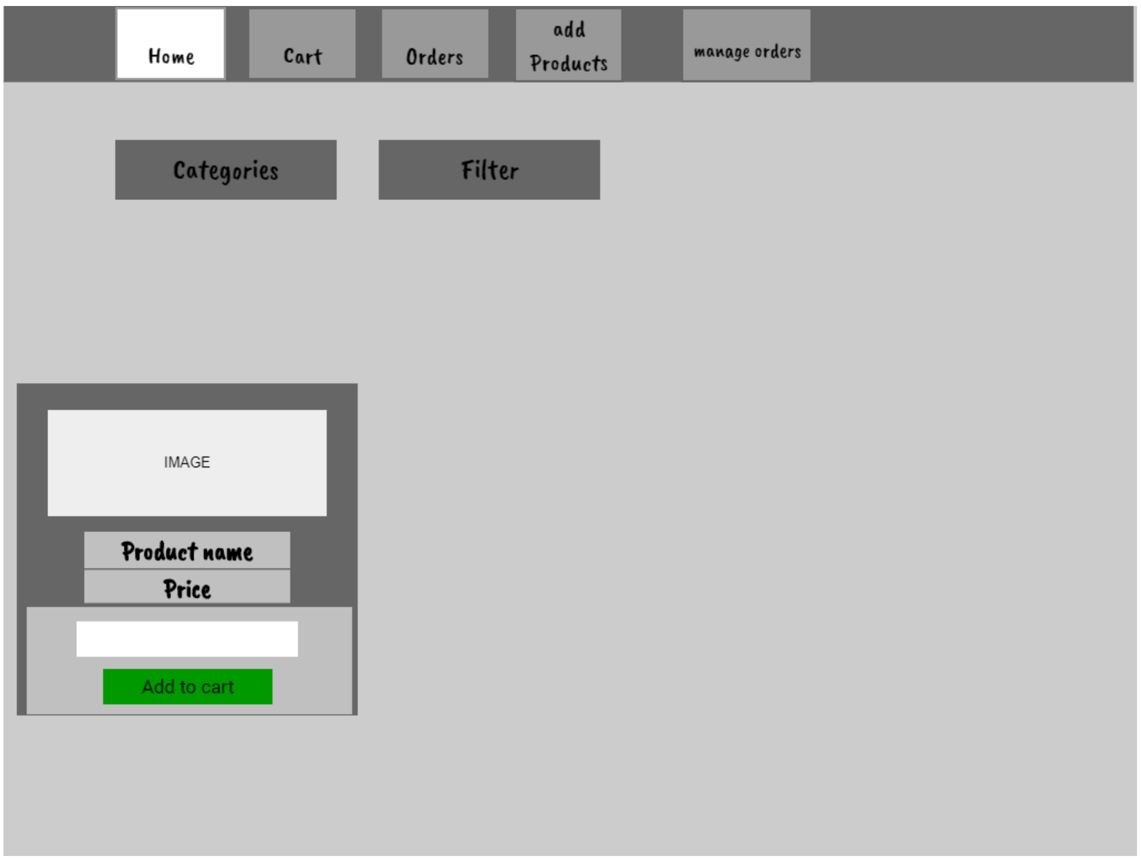
- View & filter products

التصميم:

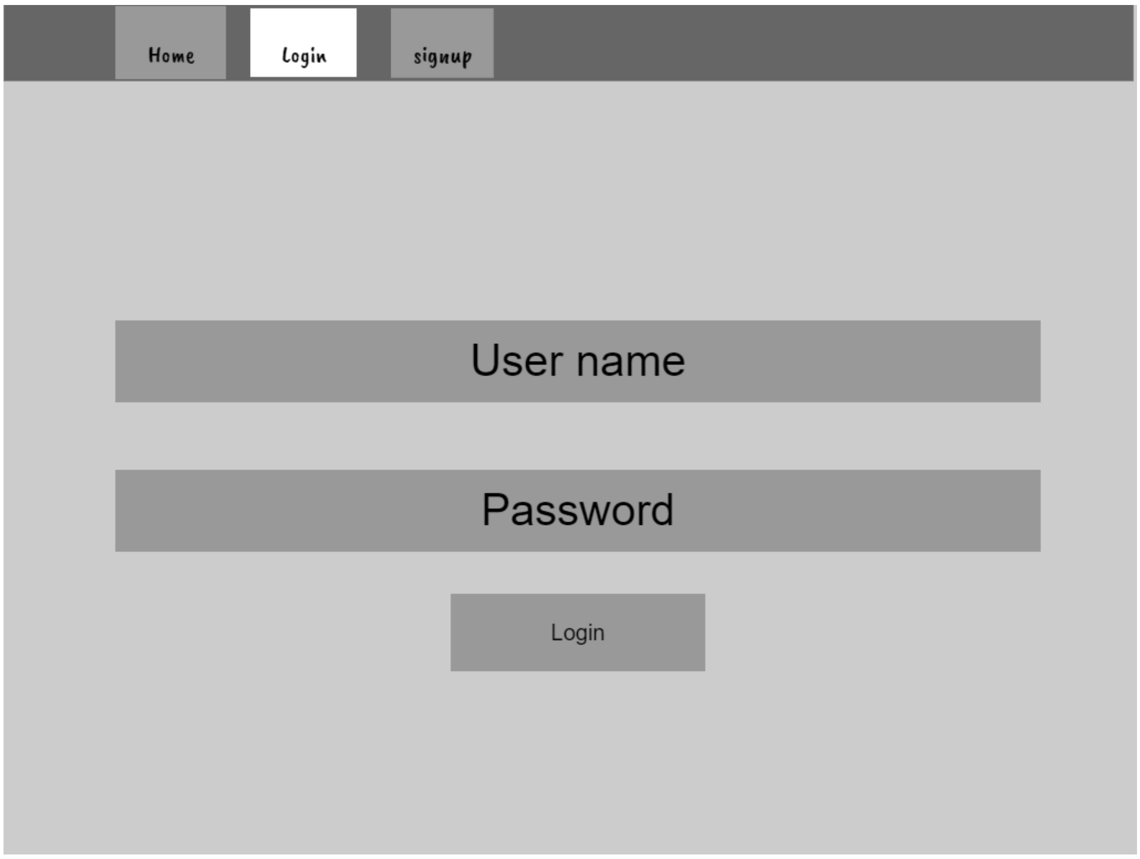
DATABASE DESGIN



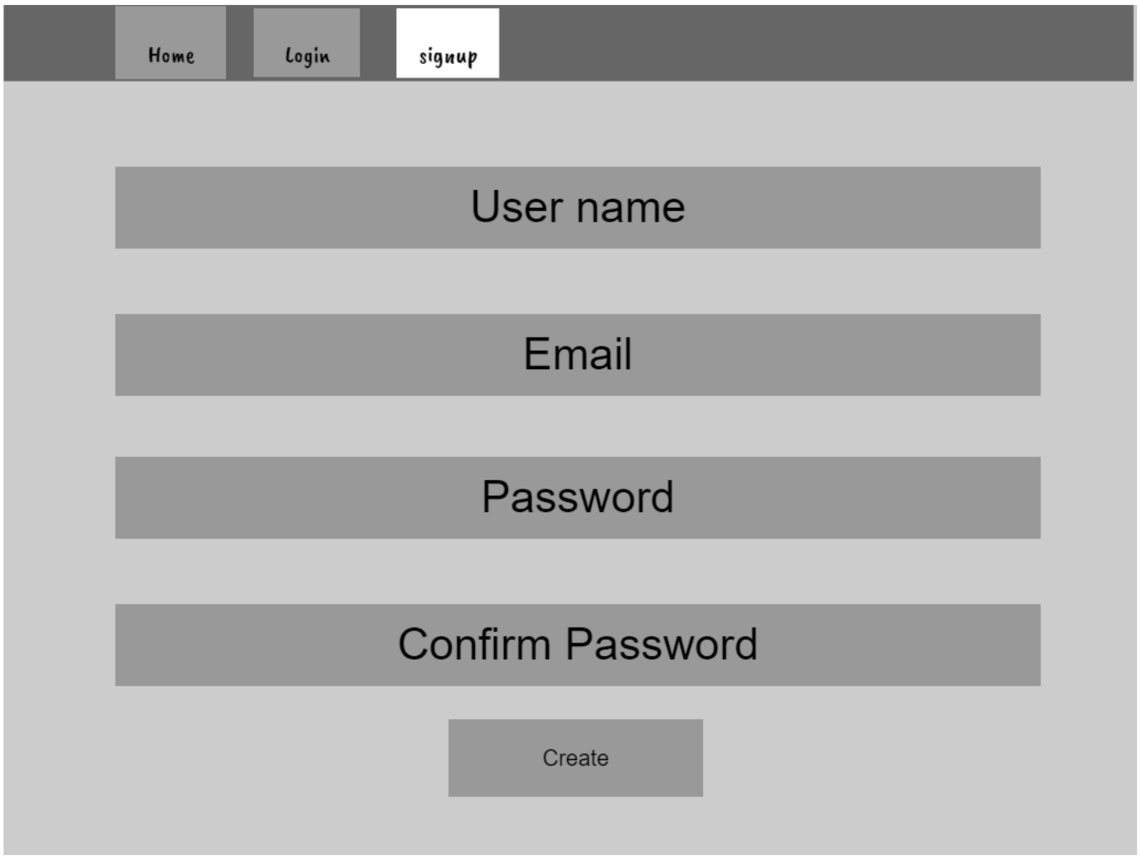
Low Level Design



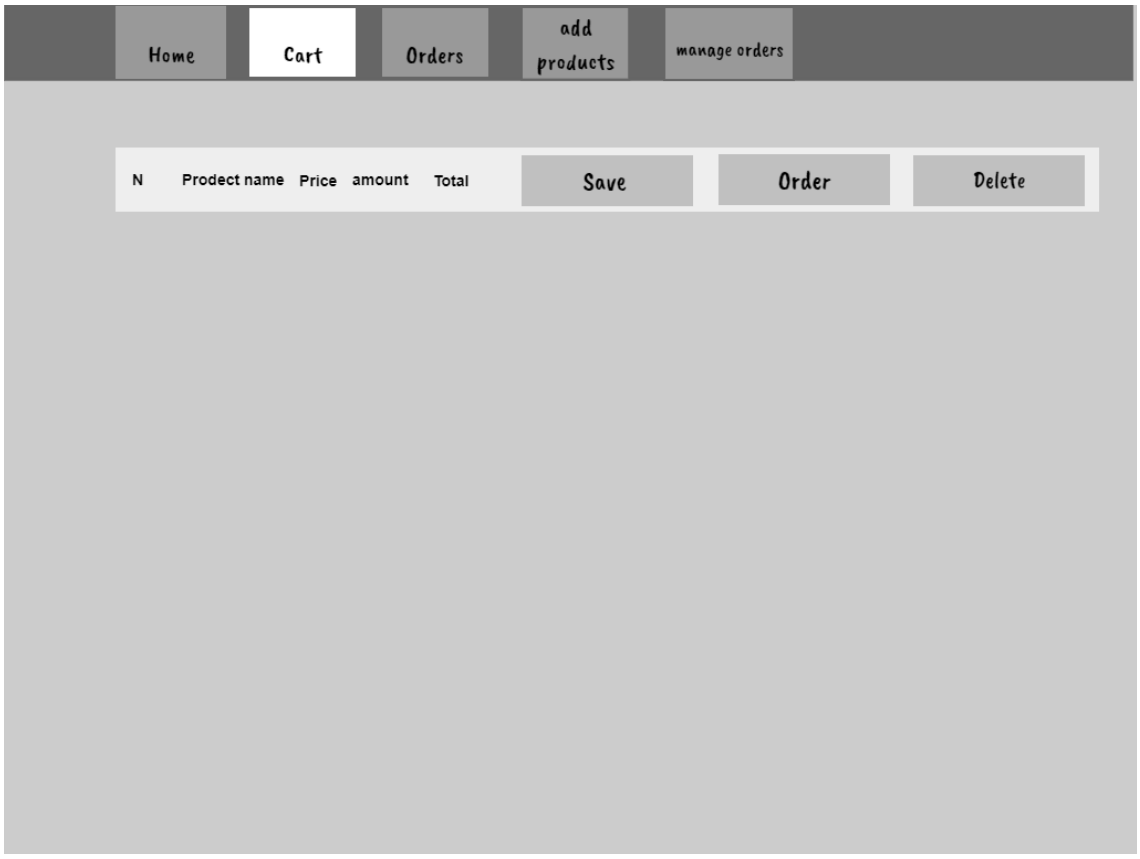
1 HOME



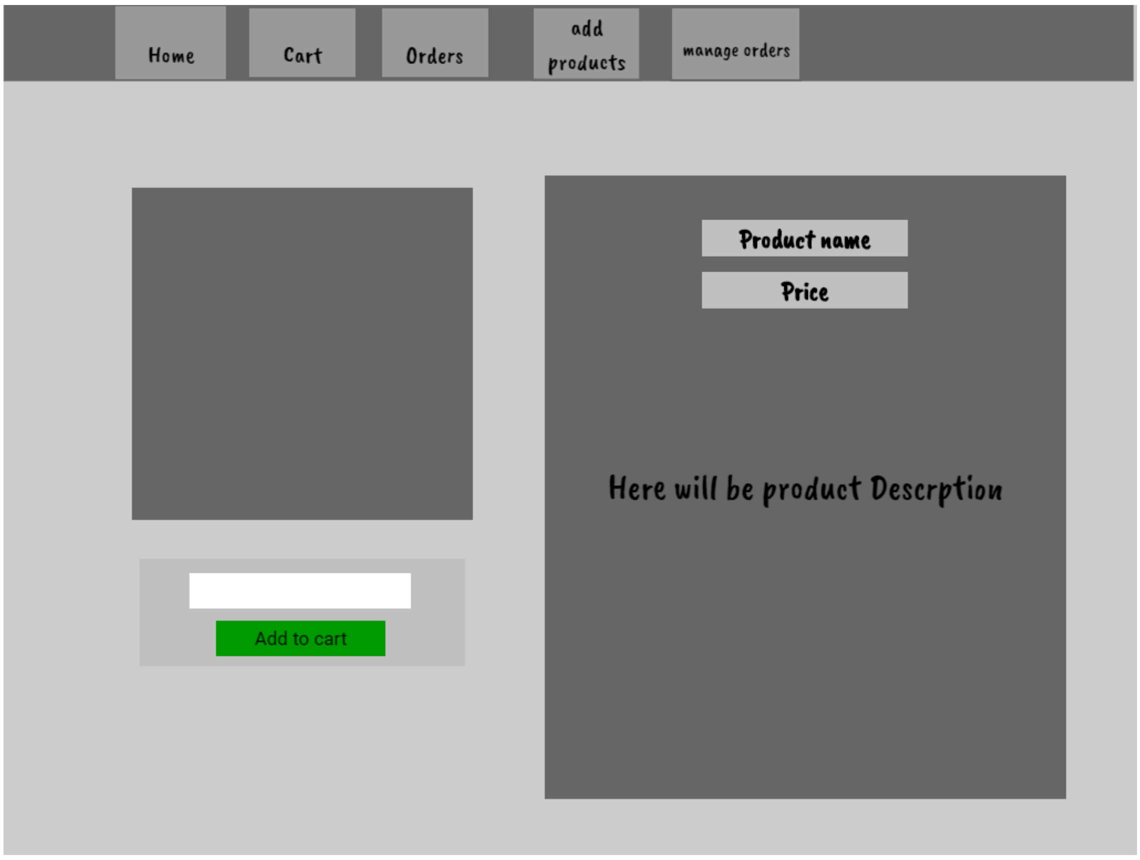
2LOGIN



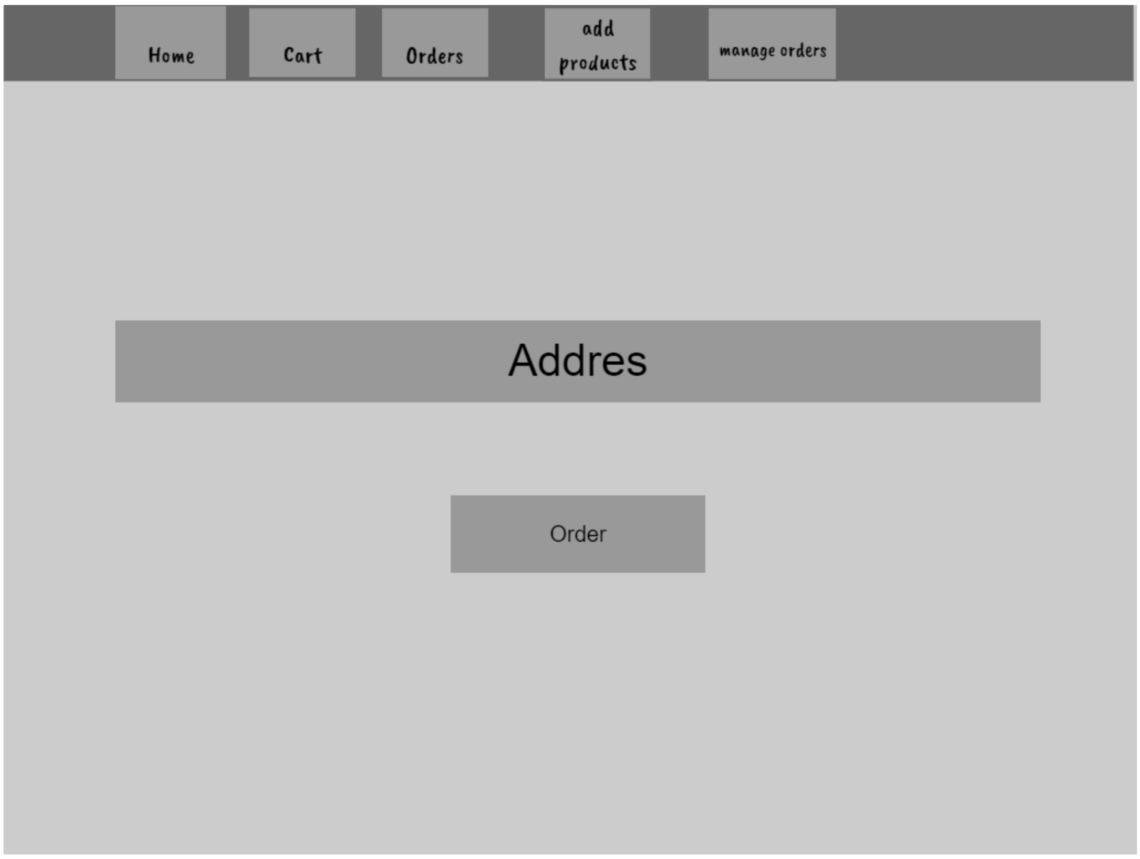
3SIGNUP



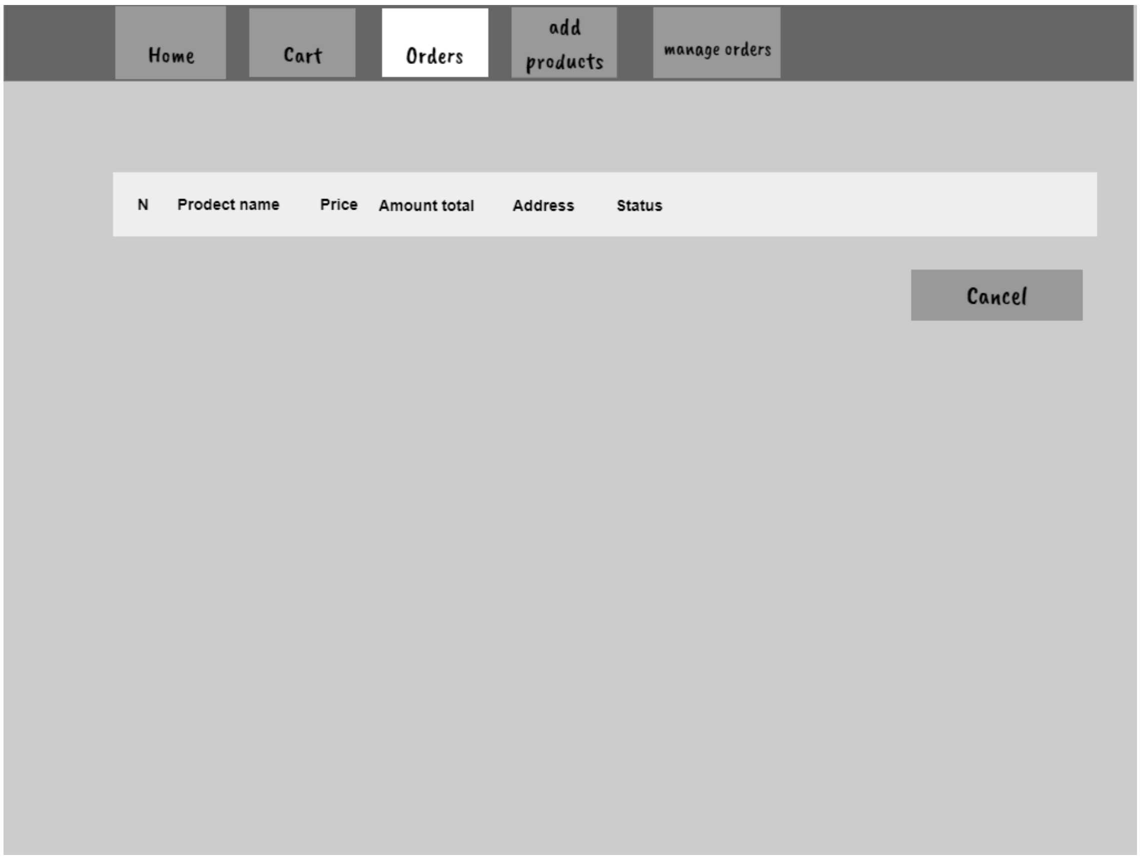
4CART



5product details

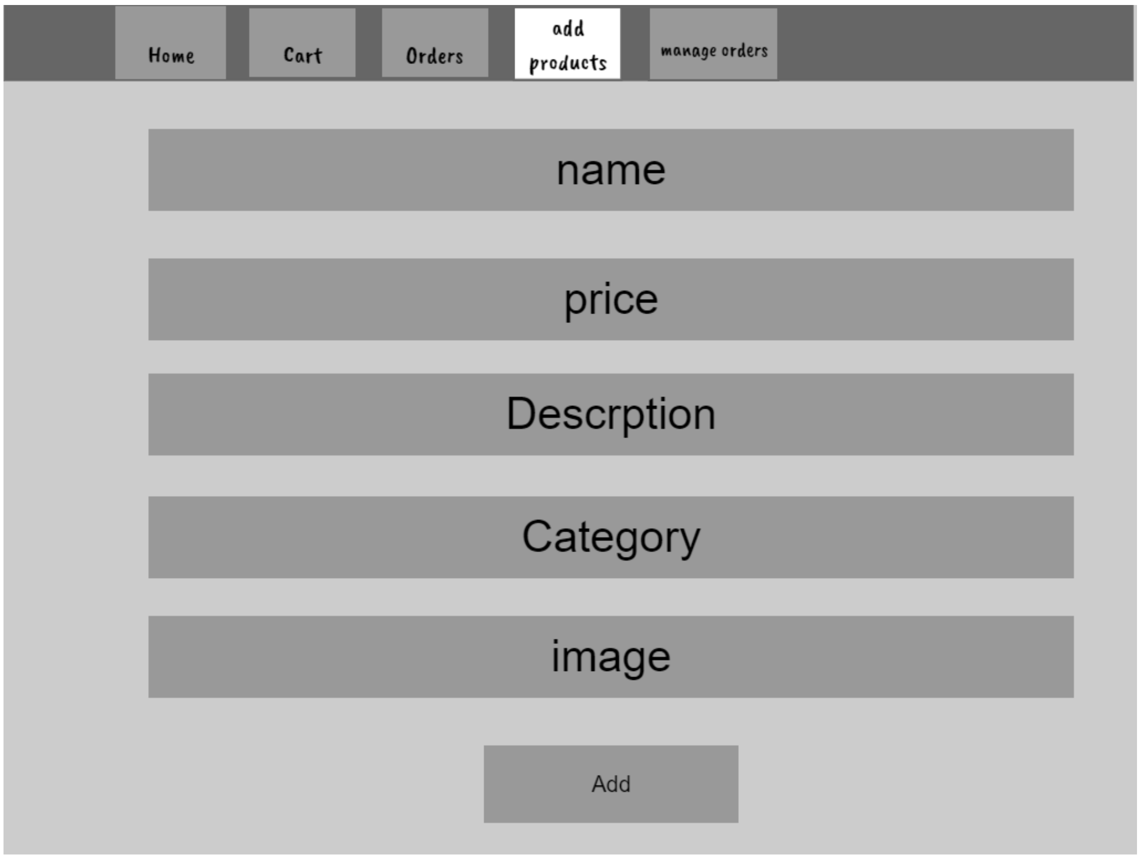


6verify order

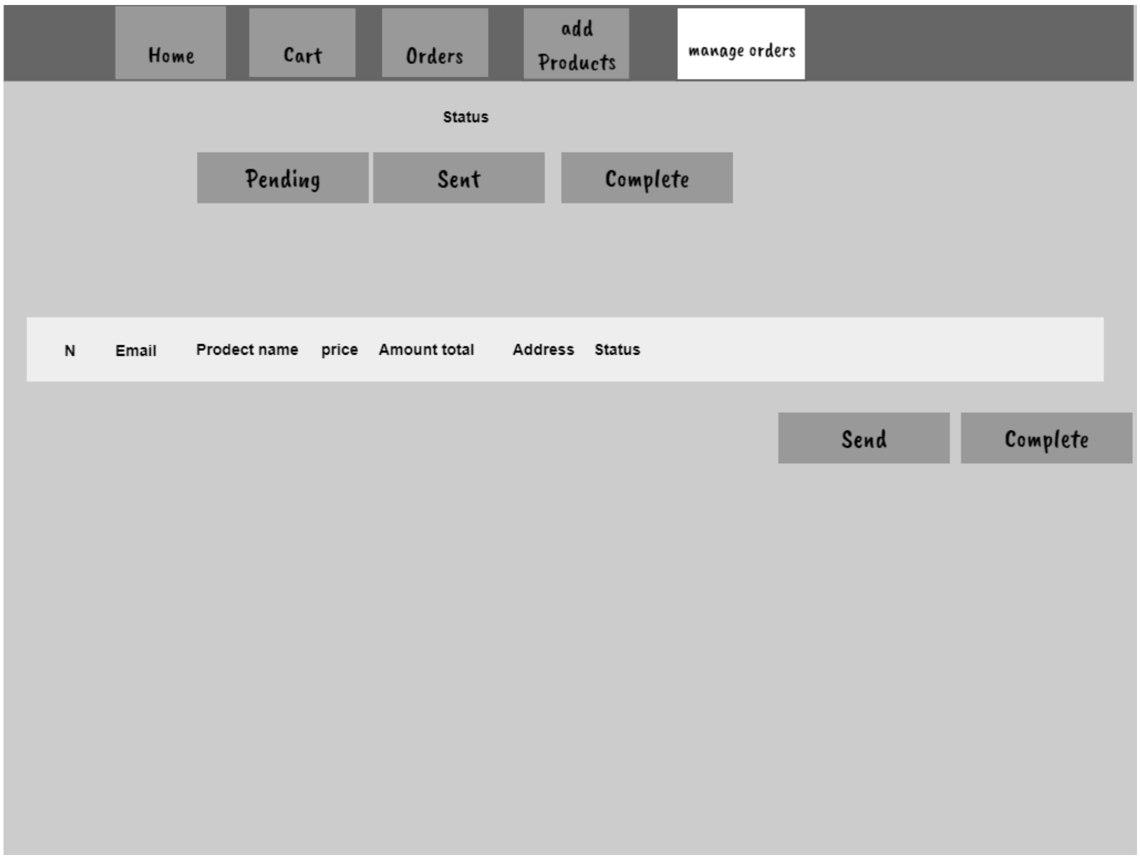


N	Prodect name	Price	Amount total	Address	Status

7 ORDERS



8ADD PRODUCTS



9 MANAGE ORDERS

الفصل الثاني: التنفيذ

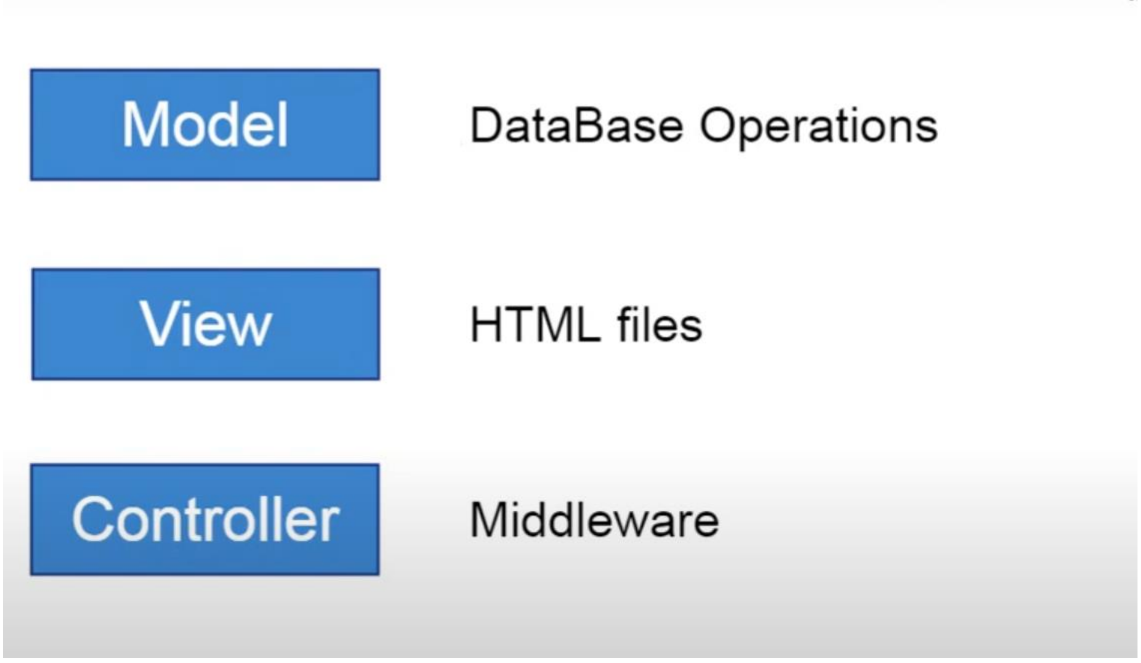
نمط المشروع – Architectural Pattern

MVC- Model, View, Controller

Controller يربط الModel ف View

Model هي الداتا.

Viewما يرأه المستخدم.


Model DataBase Operations

View HTML files

Controller Middleware

Code – الكود

Controllers

admin.controller.js

```
const productsModel = require("../models/products.model");
const ordersModel = require("../models/order.model");
const validationResult = require("express-
validator").validationResult;

exports.getAdd = (req, res, next) => {
    res.render("add-product", {
        validationErrors: req.flash("validationErrors"),
        isUser: true,
        isAdmin: true,
        productAdded: req.flash("added")[0],
        pageTitle: "Add Product"
    });
};




exports.postAdd = (req, res, next) => {
    if (validationResult(req).isEmpty()) {
        req.body.image = req.file.filename;
        productsModel
            .addNewProduct(req.body)
            .then(() => {
                req.flash("added", true);
                res.redirect("/admin/add");
            })
            .catch(err => {
                res.redirect("/error");
            });
    } else {
        req.flash("validationErrors",
validationResult(req).array());
        res.redirect("/admin/add");
    }
};

exports.getOrders = (req, res, next) => {
    ordersModel
        .getAllOrders()
        .then(items => {
            res.render("manage-orders", {
                pageTitle: "Manage Orders",
                isUser: true,
                isAdmin: true,
                items: items
            });
        })
        .catch(err => res.redirect("/error"));
};

exports.postOrders = (req, res, next) => {
    ordersModel
        .editOrder(req.body.orderId, req.body.status)
        .then(() => res.redirect("/admin/orders"))
        .catch(err => {
            res.redirect("/error");
        });
};
```

auth.controller.js

```
const authModel = require("../models/auth.model");
const validationResult = require("express-
validator").validationResult;

exports.getSignup = (req, res, next) => {
    res.render("signup", {
        authError: req.flash("authError")[0],
        validationErrors: req.flash("validationErrors"),
        isUser: false,
        isAdmin: false,
        pageTitle: "Signup"
    });
};



exports.postSignup = (req, res, next) => {
    if (validationResult(req).isEmpty()) {
        authModel
            .createNewUser(req.body.username, req.body.email,
req.body.password)
            .then(() => res.redirect("/login"))
            .catch(err => {
                req.flash("authError", err);
                res.redirect("/signup");
            });
    } else {
        req.flash("validationErrors",
validationResult(req).array());
        res.redirect("/signup");
    }
};

exports.getLogin = (req, res, next) => {
    res.render("login", {
        authError: req.flash("authError")[0],
        validationErrors: req.flash("validationErrors"),
        isUser: false,
        isAdmin: false,
        pageTitle: "Login"
    });
};

exports.postLogin = (req, res, next) => {
    if (validationResult(req).isEmpty()) {
        authModel
            .login(req.body.email, req.body.password)
            .then(result => {
                req.session.userId = result.id;
                req.session.isAdmin = result.isAdmin;
                res.redirect("/");
            })
            .catch(err => {
                req.flash("authError", err);
                res.redirect("/login");
            });
    } else {
        req.flash("validationErrors",
validationResult(req).array());
        res.redirect("/login");
    }
};

exports.logout = (req, res, next) => {
    req.session.destroy(() => {
        res.redirect("/");
    });
};
```

cart.controller.js

```javascript
const cartModel = require("../models/cart.model");
const validationResult = require("express-
validator").validationResult;

exports.getCart = (req, res, next) => {
    cartModel
        .getItemsByUser(req.session.userId)
        .then(items => {
            res.render("cart", {
                items: items,
                isUser: true,
                isAdmin: req.session.isAdmin,
                pageTitle: "Cart"
            });
        })
        .catch(err => {
            res.redirect("/error");
        });
};

exports.postCart = (req, res, next) => {
    if (validationResult(req).isEmpty()) {
        cartModel
            .addNewItem({
                name: req.body.name,
                price: req.body.price,
                amount: req.body.amount,
                productId: req.body.productId,
                userId: req.session.userId,
                timestamp: Date.now()
            })
            .then(() => {
                res.redirect("/cart");
            })
            .catch(err => {
                res.redirect("/error");
            });
    } else {
        req.flash("validationErrors",
validationResult(req).array());
        res.redirect(req.body.redirectTo);
    }
};

exports.postSave = (req, res, next) => {
```

home.controller.js

```
const productsModel = require("../models/products.model");

exports.getHome = (req, res, next) => {
    let category = req.query.category;
    let validCategories = ["clothes", "phones", "computers"];
    let productsPromise;
    if (category && validCategories.includes(category))
        productsPromise =
productsModel.getProductsByCategory(category);
    else productsPromise = productsModel.getAllProducts();
    productsPromise
        .then(products => {
            res.render("index", {
                products: products,
                isUser: req.session.userId,
                isAdmin: req.session.isAdmin,
                validationError: req.flash("validationErrors")[0],
                pageTitle: "Home"
            });
        })
        .catch(err => {
            console.log(err);
        });
};
```

order.constroller.js

```
const cartModel = require("../models/cart.model");
const orderModel = require("../models/order.model");

const validationResult = require("express-
validator").validationResult;

exports.getOrderVerify = (req, res, next) => {
    cartModel
        .getItemById(req.query.order)
        .then(cartItem => {
            res.render("verify-order", {
                cart: cartItem,
                isUser: true,
                isAdmin: req.session.isAdmin,
                pageTitle: "Verify Order",
                validationError: req.flash("validationErrors")[0]
            });
        })
        .catch(err => res.redirect("/error"));
};

exports.getOrder = (req, res, next) => {
    orderModel
        .getOrdersByUser(req.session.userId)
        .then(items => {
            res.render("orders", {
                pageTitle: "Orders",
                isUser: true,
                isAdmin: req.session.isAdmin,
                items: items
            });
        })
        .catch(err => res.redirect("/error"));
};

exports.postOrder = (req, res, next) => {
    if (validationResult(req).isEmpty())
        orderModel
            .addNewOrder(req.body)
            .then(() => res.redirect("/orders"))
            .catch(err => {
                res.redirect("/error");
            });
    else {
        req.flash("validationErrors",
validationResult(req).array());
        res.redirect("/verify-order?order=" + req.body.cartId);
    }
};

exports.postCancel = (req, res, next) => {
    orderModel
        .cancelOrder(req.body.orderId)
        .then(() => res.redirect("/orders"))
        .catch(err => {
            res.redirect("/error");
        });
};
```

product.controller.js

```
const productsModel = require("../models/products.model");

exports.getProduct = (req, res, next) => {
    productsModel
        .getFirstProduct()
        .then(product => {
            res.render("product", {
                product: product,
                isUser: req.session.userId,
                isAdmin: req.session.isAdmin,
                pageTitle: "Product Details"
            });
        })
        .catch(err => res.redirect("/error"));
};

exports.getProductById = (req, res, next) => {
    let id = req.params.id;
    productsModel
        .getProductById(id)
        .then(product => {
            res.render("product", {
                product: product,
                isUser: req.session.userId,
                isAdmin: req.session.isAdmin,
                pageTitle: product.name
            });
        })
        .catch(err => res.redirect("/error"));
};
```

Models

auth.model.js

```
const mongoose = require("mongoose");

const bcrypt = require("bcrypt");

const DB_URL = "mongodb://localhost:27017/shop";
const userSchema = mongoose.Schema({
    username: String,
    email: String,
    password: String,
    isAdmin: {
        type: Boolean,
        default: false
    }
});

const User = mongoose.model("user", userSchema);

exports.createNewUser = (username, email, password) => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return User.findOne({ email: email });
            })
            .then(user => {
                if (user) {
                    mongoose.disconnect();
                    reject("email is used");
                } else {
                    return bcrypt.hash(password, 10);
                }
            })
            .then(hashedPassword => {
                let user = new User({
                    username: username,
                    email: email,
                    password: hashedPassword
                });
                return user.save();
            })
            .then(() => {
                mongoose.disconnect();
                resolve();
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.login = (email, password) => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => User.findOne({ email: email }))
            .then(user => {
                if (!user) {
                    mongoose.disconnect();
                    reject("there is no user matches this email");
                } else {
                    bcrypt.compare(password,
user.password).then(same => {
                        if (!same) {
                            mongoose.disconnect();
                            reject("password is incorrect");
                        } else {
                            mongoose.disconnect();
                            resolve({
                                id: user.email,
                                isAdmin: user.isAdmin
                            });
                        }
                    });
                }
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};
```

cart.model.js

```
const mongoose = require("mongoose");

const DB_URL = "mongodb://localhost:27017/shop";

const cartSchema = mongoose.Schema({
    name: String,
    price: Number,
    amount: Number,
    userId: String,
    productId: String,
    timestamp: Number
});

const CartItem = mongoose.model("cart", cartSchema);

exports.addNewItem = data => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                let item = new CartItem(data);
                return item.save();
            })
            .then(() => {
                mongoose.disconnect();
                resolve();
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getItemsByUser = userId => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return CartItem.find(
                    { userId: userId },
                    {},
                    { sort: { timestamp: 1 } }
                );
            })
            .then(items => {
                mongoose.disconnect();
                resolve(items);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.editItem = (id, newData) => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => CartItem.updateOne({ _id: id }, newData))
            .then(items => {
                mongoose.disconnect();
                resolve(items);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.deleteItem = id => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => CartItem.findByIdAndDelete(id))
            .then(() => {
                mongoose.disconnect();
                resolve();
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getItemById = id => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => CartItem.findById(id))
            .then(item => {
                mongoose.disconnect();
                resolve(item);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};
```

order.model.js

```
const mongoose = require("mongoose");

const cartModel = require("./cart.model");

const DB_URL ='mongodb://localhost:27017/shop';

const orderSchema = mongoose.Schema({
    name: String,
    price: Number,
    amount: Number,
    userId: String,
    productId: String,
    timestamp: Number,
    address: String,
    status: {
        type: String,
        default: "pending"
    },
    timestamp: Number
});

const Order = mongoose.model("order", orderSchema);

exports.addNewOrder = data => {
    return new Promise((resolve, reject) => {
        cartModel
            .deleteItem(data.cartId)
            .then(() => mongoose.connect(DB_URL))
            .then(() => {
                data.timestamp = Date.now();
                let order = new Order(data);
                return order.save();
            })
            .then(() => {
                mongoose.disconnect();
                resolve();
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getOrdersByUser = userId => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Order.find(
                    { userId: userId },
                    {},
                    { sort: { timestamp: 1 } }
                );
            })
            .then(items => {
                mongoose.disconnect();
                resolve(items);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.cancelOrder = id => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => Order.findByIdAndDelete(id))
            .then(() => {
                mongoose.disconnect();
                resolve();
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getAllOrders = () => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Order.find({}, {}, { sort: { timestamp: 1 }
});
            })
            .then(items => {
                mongoose.disconnect();
                resolve(items);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.editOrder = (id, newStatus) => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Order.updateOne({ _id: id }, { status:
newStatus });
            })
            .then(items => {
                mongoose.disconnect();
                resolve(items);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};
```

products.model.js

```
const mongoose = require("mongoose");

const DB_URL = "mongodb://localhost:27017/shop";
const productSchema = mongoose.Schema({
    name: String,
    image: String,
    price: Number,
    description: String,
    category: String
});

const Product = mongoose.model("product", productSchema);

exports.addNewProduct = data => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                let newProduct = new Product(data);
                return newProduct.save();
            })
            .then(products => {
                mongoose.disconnect();
                resolve(products);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getAllProducts = () => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Product.find({});
            })
            .then(products => {
                mongoose.disconnect();
                resolve(products);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getProductsByCategory = category => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Product.find({ category: category });
            })
            .then(products => {
                mongoose.disconnect();
                resolve(products);
            })
            .catch(err => {
                mongoose.disconnect(reject(err));
            });
    });
};

exports.getProductById = id => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Product.findById(id);
            })
            .then(product => {
                mongoose.disconnect();
                resolve(product);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};

exports.getFirstProduct = () => {
    return new Promise((resolve, reject) => {
        mongoose
            .connect(DB_URL)
            .then(() => {
                return Product.findOne({});
            })
            .then(product => {
                mongoose.disconnect();
                resolve(product);
            })
            .catch(err => {
                mongoose.disconnect();
                reject(err);
            });
    });
};
```

Views

Parts

footer.ejs

```
<!--link bootstrap js-->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.
bundle.min.js"
        integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
        crossorigin="anonymous"></script>
</body>

</html>
```

header.ejs

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

<head>
    <meta charset="UTF-8">
    <title><%= pageTitle%></title>
    <!--link bootstrap css-->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstra
p.min.css" rel="stylesheet"
        integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
</head>

<body>
```

navbar.ejs

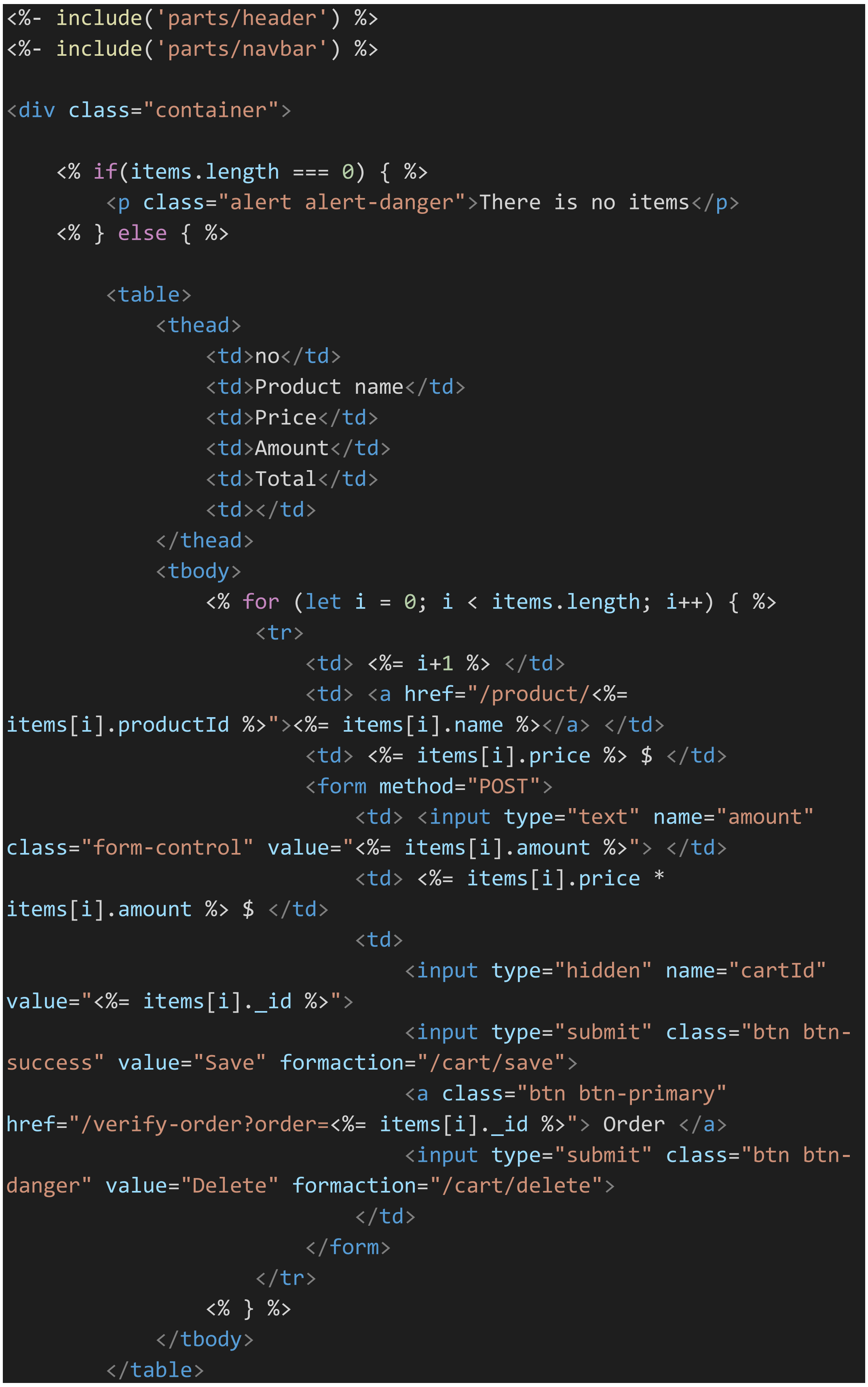
```
<nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <div class="container">
        <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse"
id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item <%= pageTitle === 'Home' ?
'active' : '' %>">
                    <a class="nav-link" href="/">Home</a>
                </li>
                <% if(isUser) { %>
                    <li class="nav-item <%= pageTitle === 'Cart' ?
'active' : '' %>">
                        <a class="nav-link" href="/cart">Cart</a>
                    </li>
                    <li class="nav-item <%= pageTitle === 'Orders' ?
'active' : '' %>">
                        <a class="nav-link"
href="/orders">Orders</a>
                    </li>
                    <% } else { %>
                        <li class="nav-item <%= pageTitle ===
'Login' ? 'active' : '' %>">
                            <a class="nav-link"
href="/login">Login</a>
                        </li>
                        <li class="nav-item <%= pageTitle ===
'Signup' ? 'active' : '' %>">
                            <a class="nav-link"
href="/signup">Signup</a>
                        </li>
                        <% } %>
                            <% if(isAdmin) { %>
                                <li class="nav-item <%= pageTitle
=== 'Add Product' ? 'active' : '' %>">
                                    <a class="nav-link"
href="/admin/add">Add Product</a>
                                </li>
                <li class="nav-item <%= pageTitle
=== 'Manage Orders' ? 'active' : '' %>">
                  <a class="nav-link"
href="/admin/orders">Manage Orders</a>
                </li>
                <% } %>
      </ul>
      <% if(isUser) { %>
        <form action="/logout" method="POST">
          <input type="submit" value="Logout" class="btn
btn-danger">
        </form>
        <% } %>
    </div>
  </div>
</nav>
```

add-product.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container text-center">

  <% if(productAdded) { %>
  <p class="alert alert-success">
    Product added successfully
  </p>
  <% } %>

  <h3>Add Product</h3>
  <form action="/admin/add" method="POST" enctype="multipart/form-
data">
    <input type="text" class="form-control" name="name"
placeholder="Name">
    <% let nameError = validationErrors.find(err => err.param
=== 'name') %>
    <% if (nameError) { %>
      <p class="alert alert-danger">
        <%= nameError.msg %>
      </p>
    <% } %>
    <input type="number" class="form-control" name="price"
placeholder="Price">
    <% let priceError = validationErrors.find(err => err.param
=== 'price') %>
```

```
        <% if (priceError) { %>
            <p class="alert alert-danger">
                <%= priceError.msg %>
            </p>
        <% } %>
        <textarea type="password" class="form-control"
name="description" placeholder="Description"></textarea>
        <% let descriptionError = validationErrors.find(err =>
err.param === 'description') %>
        <% if (descriptionError) { %>
            <p class="alert alert-danger">
                <%= descriptionError.msg %>
            </p>
        <% } %>
        <select type="password" class="form-control"
name="category">
            <option value="" selected>Choose a category</option>
            <option value="clothes">Clothes</option>
            <option value="phones">Phones</option>
            <option value="computers">Computers</option>
        </select>
        <% let categoryError = validationErrors.find(err =>
err.param === 'category') %>
        <% if (categoryError) { %>
            <p class="alert alert-danger">
                <%= categoryError.msg %>
            </p>
        <% } %>
        <input type="file" class="form-control" name="image"
placeholder="Image">
        <% let imageError = validationErrors.find(err => err.param
=== 'image') %>
        <% if (imageError) { %>
            <p class="alert alert-danger">
                <%= imageError.msg %>
            </p>
        <% } %>
        <input type="submit" value="Add Product" class="btn btn-
primary">
    </form>
</div>

<%- include('parts/footer')%>
```

cart.ejs



```
<%- include('parts/header') %>
<%- include('parts/navbar') %>

<div class="container">

    <% if(items.length === 0) { %>
        <p class="alert alert-danger">There is no items</p>
    <% } else { %>

        <table>
            <thead>
                <td>no</td>
                <td>Product name</td>
                <td>Price</td>
                <td>Amount</td>
                <td>Total</td>
                <td></td>
            </thead>
            <tbody>
                <% for (let i = 0; i < items.length; i++) { %>
                    <tr>
                        <td> <%= i+1 %> </td>
                        <td> <a href="/product/<%=
items[i].productId %>"><%= items[i].name %></a> </td>
                        <td> <%= items[i].price %> $ </td>
                        <form method="POST">
                            <td> <input type="text" name="amount"
class="form-control" value="<%= items[i].amount %>"> </td>
                            <td> <%= items[i].price *
items[i].amount %> $ </td>
                            <td>
                                <input type="hidden" name="cartId"
value="<%= items[i]._id %>">
                                <input type="submit" class="btn btn-
success" value="Save" formaction="/cart/save">
                                <a class="btn btn-primary"
href="/verify-order?order=<%= items[i]._id %>"> Order </a>
                                <input type="submit" class="btn btn-
danger" value="Delete" formaction="/cart/delete">
                            </td>
                        </form>
                    </tr>
                <% } %>
            </tbody>
        </table>
    <% } %>

</div>

<%- include('parts/footer') %>
```

error.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container">
    <p class="alert alert-danger">Something went wrong</p>
</div>

<%- include('parts/footer')%>
```

index.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container">

    <form class="filter" action="/" method="GET">
        <div class="row">
            <div class="col col-md-10">
                <select name="category" class="form-control">
                    <option value="all">All</option>
                    <option value="clothes">Clothes</option>
                    <option value="phones">Phones</option>
                    <option value="computers">Computers</option>
                </select>
            </div>
            <div class="col col-md-2">
                <input type="submit" class="btn btn-primary"
value="Filter">
            </div>
        </div>
    </form>

    <% if(validationError) { %>
        <p class="alert alert-danger">
            <%= validationError.msg%>
        </p>
    <% } %>

    <% if(products.length === 0) { %>
    <div class="alert alert-danger">
        there is no products
    </div>
    <% } else { %>
    <div class="row">
        <% for(let product of products) { %>
            <div class="col col-12 col-md-6 col-lg-4 col-xl-3">
                <div class="card" style="width: 100%;">
                    <img src="/<%= product.image%>" class="card-img-
top" style="height: 200px;">
                    <div class="card-body">
                        <h5 class="card-title">
                            <a href="/product/<%= product._id%>"><%=
product.name%></a>
                                <p>Price: <%= product.price%> $</p>
                        </h5>
                        <form action="/cart" method="POST"
class="add-to-cart-form">
                            <input type="number" name="amount"
class="form-control">
                            <input type="hidden" name="name"
value="<%= product.name%>">
                            <input type="hidden" name="price"
value="<%= product.price%>">
                            <input type="hidden" name="productId"
value="<%= product._id%>">
                            <input type="hidden" name="redirectTo"
value="/">
                            <input type="submit" value="Add to cart"
class="btn btn-primary">
                        </form>
                    </div>
                </div>
            </div>
        <% } %>
    </div>
    <% } %>
</div>

<%- include('parts/footer')%>
```

login.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container text-center">
    <h3>Login</h3>
    <form action="/login" method="POST">
        <input type="email" class="form-control" name="email"
placeholder="E-mail">
        <% let emailError = validationErrors.find(err => err.param
=== 'email') %>
        <% if (emailError) { %>
            <p class="alert alert-danger">
                <%= emailError.msg %>
            </p>
        <% } %>
        <input type="password" class="form-control" name="password"
placeholder="Password">
        <% let passwordError = validationErrors.find(err =>
err.param === 'password') %>
        <% if (passwordError) { %>
            <p class="alert alert-danger">
                <%= passwordError.msg %>
            </p>
        <% } %>
        <input type="submit" value="Login" class="btn btn-primary">
    </form>
    <% if(authError) { %>
        <br>
        <p class="alert alert-danger">
            <%= authError%>
        </p>
    <% } %>
    <br>
    <div>
        Don't have account ? <a href="/signup">Create Account</a>
    </div>
</div>

<%- include('parts/footer')%>
```

manage-orders.ejs

```
<%- include('parts/header') %>
<%- include('parts/navbar') %>

<div class="container">

    <% if(items.length === 0) { %>
        <p class="alert alert-danger">There is no items</p>
    <% } else { %>

        <table>
            <thead>
                <td>no</td>
                <td>Email</td>
                <td>Product name</td>
                <td>Price</td>
                <td>Amount</td>
                <td>Total</td>
                <td>Address</td>
                <td>Status</td>
                <td></td>
            </thead>
            <tbody>
                <% for (let i = 0; i < items.length; i++) { %>
                    <tr>
                        <td> <%= i+1 %> </td>
                        <td><%= items[i].userId %></td>
                        <td> <a href="/product/<%=
items[i].productId %>"><%= items[i].name %></a> </td>
                        <td> <%= items[i].price %> $ </td>
                        <td> <%= items[i].amount %> </td>
                        <td> <%= items[i].price * items[i].amount %>
$ </td>
                        <td> <%= items[i].address %> </td>
                        <td> <%= items[i].status %> </td>
                        <td>
                            <% if (items[i].status === 'pending') {
%>
                            <form style="display: inline-block"
method="POST" action="/admin/orders">
                                <input type="hidden" name="orderId"
value="<%= items[i]._id %>">
                                <input type="hidden" name="status"
value="sent">
                                <input type="submit" class="btn btn-
primary" value="Send">
```

```
                            </form>
                            <% } %>
                            <% if (items[i].status === 'pending' ||
items[i].status === 'sent') { %>
                            <form style="display: inline-block"
method="POST" action="/admin/orders">
                                <input type="hidden" name="orderId"
value="<%= items[i]._id %>">
                                <input type="hidden" name="status"
value="complete">
                                <input type="submit" class="btn btn-
success" value="Complete">
                            </form>
                            <% } %>
                        </td>
                    </tr>
                <% } %>
            </tbody>
        </table>

    <% } %>

</div>

<%- include('parts/footer') %>
```

not-found.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container">
    <p class="alert alert-danger">Page not found</p>
</div>

<%- include('parts/footer')%>
```

not-admin.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container">
    <p class="alert alert-danger">you are not admin</p>
</div>

<%- include('parts/footer')%>
```

orders.ejs

```
<%- include('parts/header') %>
<%- include('parts/navbar') %>

<div class="container">

    <% if(items.length === 0) { %>
        <p class="alert alert-danger">There is no items</p>
    <% } else { %>

        <table>
            <thead>
                <td>no</td>
                <td>Product name</td>
                <td>Price</td>
                <td>Amount</td>
                <td>Total</td>
                <td>Address</td>
                <td>Status</td>
                <td></td>
            </thead>
            <tbody>
                <% for (let i = 0; i < items.length; i++) { %>
                    <tr>
                        <td> <%= i+1 %> </td>
                        <td> <a href="/product/<%=
items[i].productId %>"><%= items[i].name %></a> </td>
                        <td> <%= items[i].price %> $ </td>
                        <form method="POST">
                            <td> <%= items[i].amount %> </td>
                            <td> <%= items[i].price *
items[i].amount %> $ </td>
                            <td> <%= items[i].address %> </td>
                            <td> <%= items[i].status %> </td>
                            <td>
```

```
                                <input type="hidden" name="orderId"
value="<%= items[i]._id %>">
                                <input type="submit" class="btn btn-
danger" value="Cancel" formaction="/orders/cancel">
                            </td>
                        </form>
                    </tr>
                <% } %>
            </tbody>
        </table>

    <% } %>

</div>

<%- include('parts/footer') %>
```

product.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container">
    <% if(!product) { %>
    <div class="alert alert-danger">
        there is no product matches this id
    </div>
    <% } else {%>
    <div class="row">
        <div class="col col-12 col-md-6">
            <div class="card" style="width: 100%;">
                <img src="/<%= product.image%>" class="card-img-top"
style="max-height: 400px">
                <div class="card-body">
                    <form action="/cart" method="POST" class="add-
to-cart-form">
                        <input type="number" name="amount"
class="form-control">
                        <input type="submit" value="Add to cart"
class="btn btn-primary">
                    </form>
                </div>
            </div>
        </div>
        <div class="col col-12 col-md-6 product-detail">
            <h3><%= product.name%></h3>
            <h5>Price: <%= product.price%> $</h5>
            <p><%= product.description%></p>
        </div>
    </div>
    <% } %>
</div>

<%- include('parts/footer')%>
```

signup.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container text-center">
    <h3>Create Account</h3>
    <form action="/signup" method="POST">
        <input type="text" class="form-control" name="username"
placeholder="Username">
        <% let usernameError = validationErrors.find(err =>
err.param === 'username') %>
        <% if (usernameError) { %>
            <p class="alert alert-danger">
                <%= usernameError.msg %>
            </p>
        <% } %>
        <input type="email" class="form-control" name="email"
placeholder="E-mail">
        <% let emailError = validationErrors.find(err => err.param
=== 'email') %>
        <% if (emailError) { %>
            <p class="alert alert-danger">
                <%= emailError.msg %>
            </p>
        <% } %>
        <input type="password" class="form-control" name="password"
placeholder="Password">
        <% let passwordError = validationErrors.find(err =>
err.param === 'password') %>
        <% if (passwordError) { %>
            <p class="alert alert-danger">
                <%= passwordError.msg %>
            </p>
        <% } %>
        <input type="password" class="form-control"
name="confirmPassword" placeholder="Confirm Password">
        <% let passwordConfirmError = validationErrors.find(err =>
err.param === 'passwordConfirm') %>
        <% if (passwordConfirmError) { %>
            <p class="alert alert-danger">
```

```
                <%= passwordConfirmError.msg %>
            </p>
        <% } %>
        <input type="submit" value="Create Account" class="btn btn-
primary">
    </form>
    <% if(authError) { %>
        <br>
        <p class="alert alert-danger">
            <%= authError%>
        </p>
    <% } %>

    <br>
    <div>
        Already have account ? <a href="/login">Login</a>
    </div>
</div>



<%- include('parts/footer')%>
```

verify-order.ejs

```
<%- include('parts/header')%>
<%- include('parts/navbar')%>

<div class="container text-center">
    <h3>Verify Order</h3>
    <% if(validationError) { %>
    <p class="alert alert-danger">
    <%= validationError.msg %>
    </p>
    <% } %>
    <form action="/orders" method="POST">
        <input type="hidden" name="cartId" value="<%= cart._id%>">
        <input type="hidden" name="name" value="<%= cart.name%>">
        <input type="hidden" name="price" value="<%= cart.price%>">
        <input type="hidden" name="amount" value="<%=
cart.amount%>">
        <input type="hidden" name="productId" value="<%=
cart.productId%>">
        <input type="hidden" name="userId" value="<%=
cart.userId%>">
        <input type="text" name="address" placeholder="Enter the
Address" class="form-control">
        <input type="submit" value="Verify" class="btn btn-primary">
    </form>
</div>

<%- include('parts/footer')%>
```

Routes

Guards

admin.guard.js

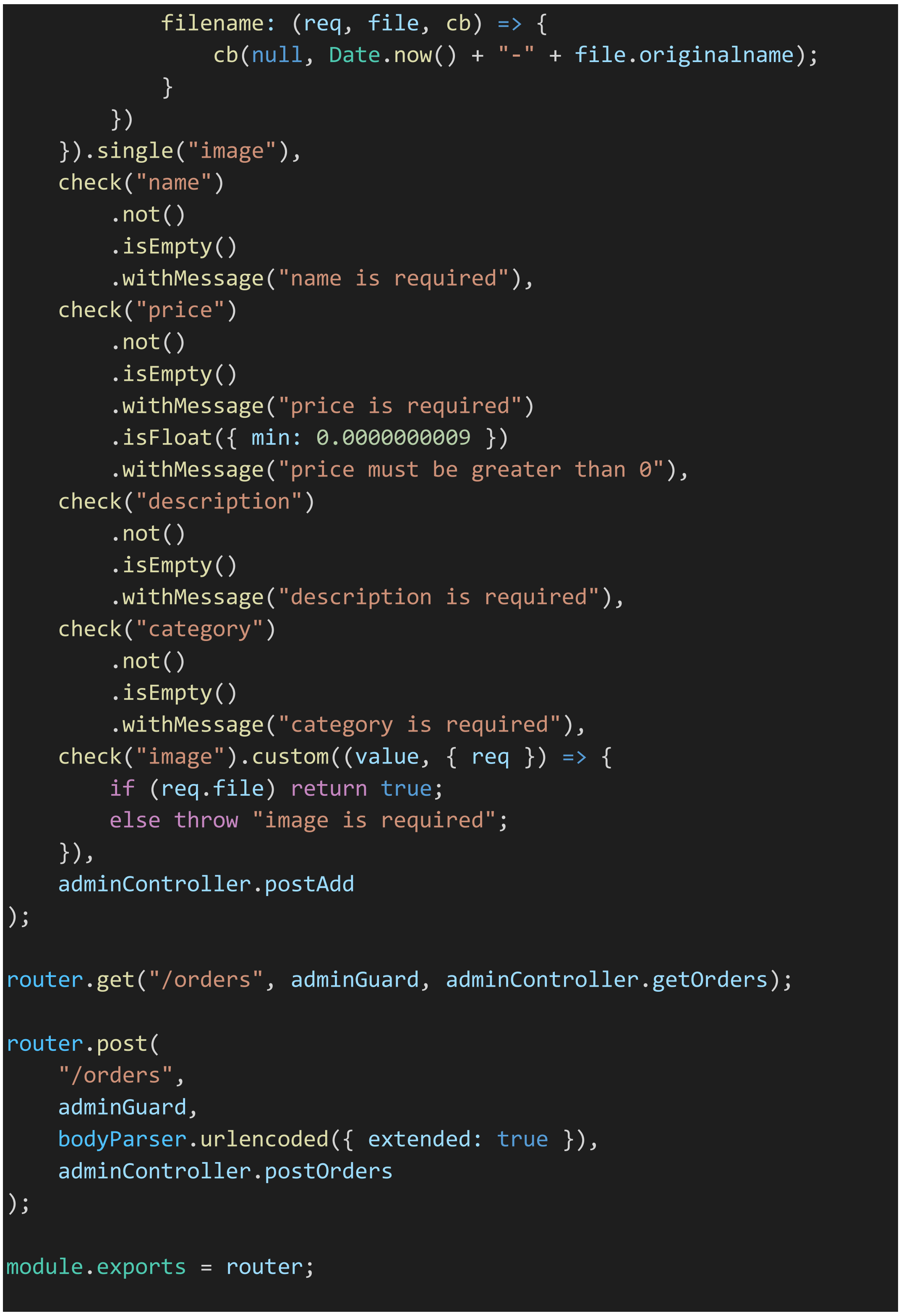
```
module.exports = (req, res, next) => {
    if (req.session.isAdmin) next();
    else res.redirect("/not-admin");
};
```

auth.guard.js

```
exports.isAuth = (req, res, next) => {
    if (req.session.userId) next();
    else res.redirect("/login");
};

exports.notAuth = (req, res, next) => {
    if (!req.session.userId) next();
    else res.redirect("/");
};
```

admin.route.js

```
const router = require("express").Router();
const check = require("express-validator").check;
const multer = require("multer");
const bodyParser = require("body-parser");

const adminController = require("../controllers/admin.controller");
const adminGuard = require("./guards/admin.guard");

router.get("/add", adminGuard, adminController.getAdd);

router.post(
    "/add",
    adminGuard,
    multer({
        storage: multer.diskStorage({
            destination: (req, file, cb) => {
                cb(null, "images/");
            },
```



```
            filename: (req, file, cb) => {
                cb(null, Date.now() + "-" + file.originalname);
            }
        })
    }).single("image"),
    check("name")
        .not()
        .isEmpty()
        .withMessage("name is required"),
    check("price")
        .not()
        .isEmpty()
        .withMessage("price is required")
        .isFloat({ min: 0.0000000009 })
        .withMessage("price must be greater than 0"),
    check("description")
        .not()
        .isEmpty()
        .withMessage("description is required"),
    check("category")
        .not()
        .isEmpty()
        .withMessage("category is required"),
    check("image").custom((value, { req }) => {
        if (req.file) return true;
        else throw "image is required";
    }),
    adminController.postAdd
);

router.get("/orders", adminGuard, adminController.getOrders);

router.post(
    "/orders",
    adminGuard,
    bodyParser.urlencoded({ extended: true }),
    adminController.postOrders
);

module.exports = router;
```

auth.route.js

```
const router = require("express").Router();
const bodyParser = require("body-parser");
const check = require("express-validator").check;

const authGuard = require("./guards/auth.guard");

const authController = require("../controllers/auth.controller");

router.get("/signup", authGuard.notAuth, authController.getSignup);

router.post(
    "/signup",
    authGuard.notAuth,
    bodyParser.urlencoded({ extended: true }),
    check("username")
        .not()
        .isEmpty()
        .withMessage("username is required"),
    check("email")
        .not()
        .isEmpty()
        .withMessage("email is required")
        .isEmail()
        .withMessage("invalid format"),
    check("password")
        .not()
        .isEmpty()
        .withMessage("password is required")
        .isLength({ min: 6 })
        .withMessage("password must be at least 6 charachters"),
    check("confirmPassword").custom((value, { req }) => {
        if (value === req.body.password) return true;
        else throw "passwords dont equal";
    }),
    authController.postSignup
);

router.get("/login", authGuard.notAuth, authController.getLogin);

router.post(
    "/login",
    authGuard.notAuth,
    bodyParser.urlencoded({ extended: true }),
    check("email")
        .not()
        .isEmpty()
        .withMessage("email is required")
        .isEmail()
        .withMessage("invalid format"),
    check("password")
        .not()
        .isEmpty()
        .withMessage("password is required")
        .isLength({ min: 6 })
        .withMessage("password must be at least 6 charachters"),
    authController.postLogin
);

router.all("/logout", authGuard.isAuth, authController.logout);

module.exports = router;
```

cart.route.js

```
const router = require("express").Router();
const bodyParser = require("body-parser");
const check = require("express-validator").check;

const authGaurd = require("./guards/auth.guard");

const cartController = require("../controllers/cart.controller");

router.get("/", authGaurd.isAuth, cartController.getCart);

router.post(
    "/",
    authGaurd.isAuth,
    bodyParser.urlencoded({ extended: true }),
    check("amount")
        .not()
        .isEmpty()
        .withMessage("amount is required")
        .isInt({ min: 1 })
        .withMessage("amount must be grater then 0"),
    cartController.postCart
);

router.post(
    "/save",
    authGaurd.isAuth,
    bodyParser.urlencoded({ extended: true }),
    check("amount")
        .not()
        .isEmpty()
        .withMessage("amount is required")
        .isInt({ min: 1 })
        .withMessage("amount must be grater then 0"),
    cartController.postSave
);

router.post(
    "/delete",
    authGaurd.isAuth,
    bodyParser.urlencoded({ extended: true }),
    cartController.postDelete
);

module.exports = router;
```

home.route.js

```
const router = require("express").Router();

const homeController = require("../controllers/home.controller");

router.get("/", homeController.getHome);

module.exports = router;
```

home.routes.js

```
const router = require('express').Router();


const homeController = require('../controllers/home.controller');

router.get('/', homeController.getHome);

module.exports = router;
```

orders.route.js

```
const router = require("express").Router();
const bodyParser = require("body-parser");
const check = require("express-validator").check;

const orderController = require("../controllers/order.controller");
const authGuard = require("./guards/auth.guard");

router.get("/verify-order", authGuard.isAuth,
orderController.getOrderVerify);

router.get("/orders", authGuard.isAuth, orderController.getOrder);

router.post(
    "/orders",
    authGuard.isAuth,
    bodyParser.urlencoded({ extended: true }),
    check("address")
        .not()
        .isEmpty()
        .withMessage("address is required"),
    orderController.postOrder
);
```

```
router.post(
    "/orders/cancel",
    authGuard.isAuth,
    bodyParser.urlencoded({ extended: true }),
    orderController.postCancel
);

module.exports = router;
```

product.route.js

```
const router = require("express").Router();

const productController =
require("../controllers/product.controller");

router.get("/", productController.getProduct);

router.get("/:id", productController.getProductById);

module.exports = router;
```

app.js

هذا أول ملف يشتغل فالنظام وهنا يوجد لُب المشروع والملفات الثابتة والقالب المستخدم مثل EJS والمنفذ للسيرفر وهو 8080وحالات الخطأ

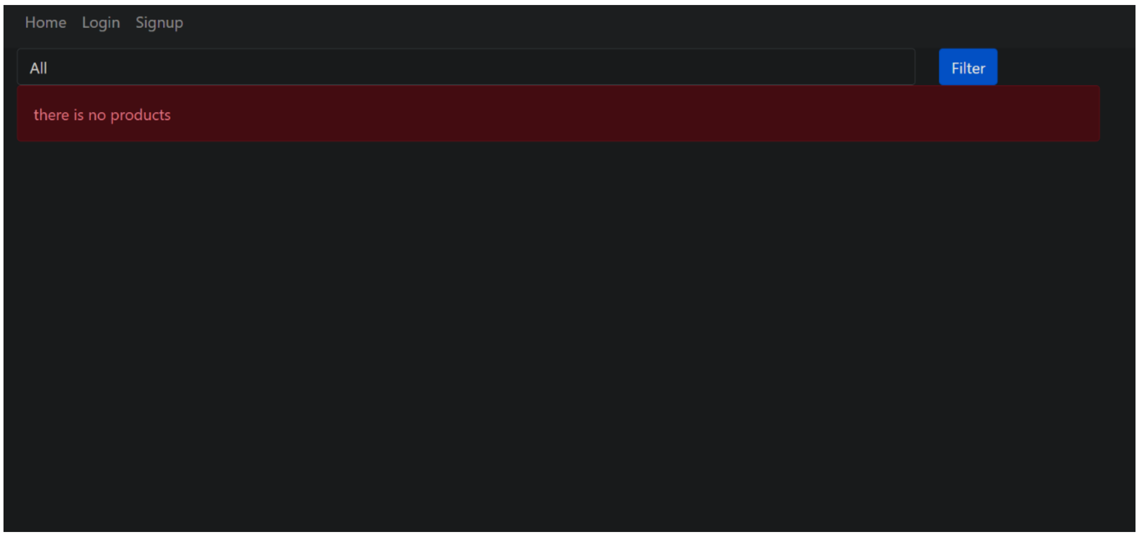
```javascript
const express = require("express");
const path = require("path");

const session = require("express-session");
const SessionStore = require("connect-mongodb-session")(session);
const flash = require("connect-flash");

const homeRouter = require("./routes/home.route");
const productRouter = require("./routes/product.route");
const authRouter = require("./routes/auth.route");
const cartRouter = require("./routes/cart.route");
const adminRouter = require("./routes/admin.route");
const orderRouter = require("./routes/orders.route");

const app = express();

app.use(express.static(path.join(__dirname, "images")));
app.use(flash());

const STORE = new SessionStore({
    uri: "mongodb://localhost:27017/shop",
    collection: "sessions"
});

app.use(
    session({
        secret: " ......",
        saveUninitialized: false,
        store: STORE
    })
);

app.set("view engine", "ejs");
app.set("views", "views");

app.use("/", homeRouter);
app.use("/", authRouter);
app.use("/product", productRouter);
app.use("/cart", cartRouter);
app.use("/admin", adminRouter);
app.use("/", orderRouter);
app.get("/error", (req, res, next) => {
    res.status(500);
    res.render("error.ejs", {
        isUser: req.session.userId,
        isAdmin: req.session.isAdmin,
        pageTitle: "Error"
    });
});

app.get("/not-admin", (req, res, next) => {
    res.status(403);
    res.render("not-admin", {
        isUser: req.session.userId,
        isAdmin: false,
        pageTitle: "Not Allowed"
    });
});

app.use((req, res, next) => {
    res.status(404);
    res.render("not-found", {
        isUser: req.session.userId,
        isAdmin: req.session.isAdmin,
        pageTitle: "Page Not Found"
    });
});


app.listen(8080, () => {
    console.log("server listen on port 8080");
});
```

package.json

هنا معلومات المشروع المؤلف، وأول ملف يشتغل app.js بالتطبيق والمكاتب المستخدمة

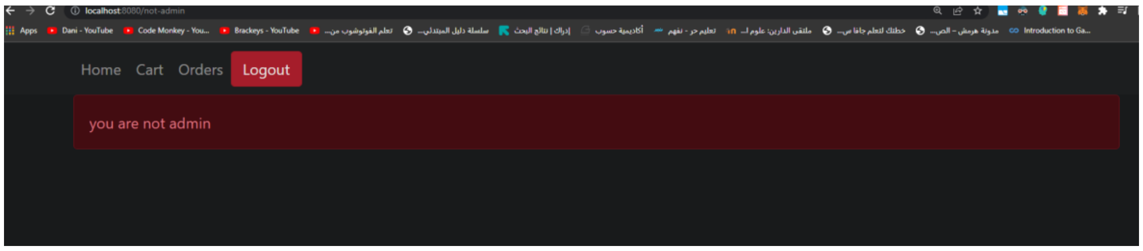
```json
{
  "name": "shop",
  "version": "1.0.0",
  "description": "Graduate project",
  "main": "app.js",
  "scripts": {
    "start": "nodemon app.js"
  },
  "author": "ayman Al-awfi",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.0.1",
    "body-parser": "^1.19.2",
    "connect-flash": "^0.1.1",
    "connect-mongodb-session": "^3.1.1",
    "ejs": "^3.1.6",
    "express": "^4.17.3",
    "express-session": "^1.17.2",
    "express-validator": "^6.14.0",
    "mongodb": "^4.5.0",
    "mongoose": "^6.3.1",
    "multer": "^1.4.4"
  },
  "devDependencies": {
    "nodemon": "^2.0.15"
  }
}
```

الإختبار



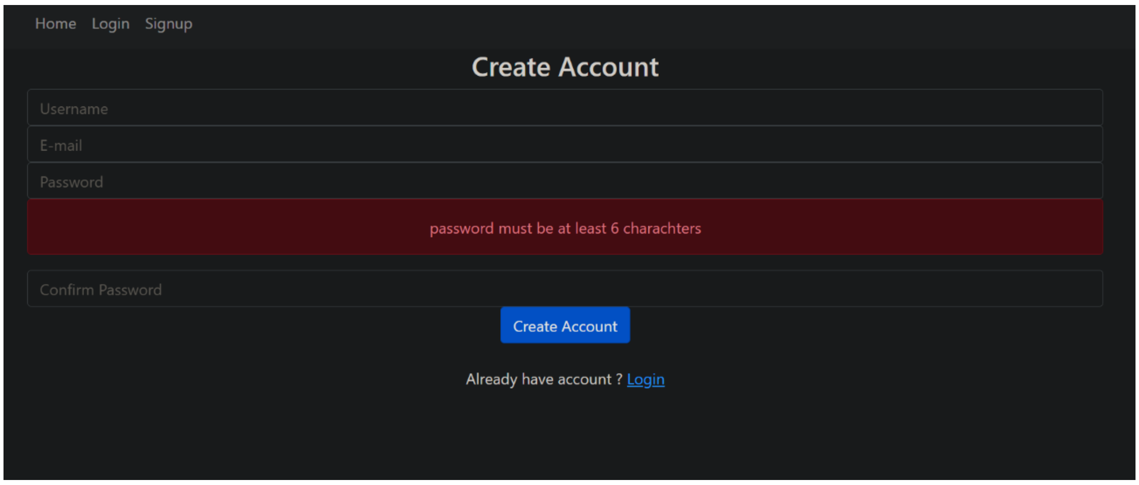
there is no products

10TEST



you are not admin

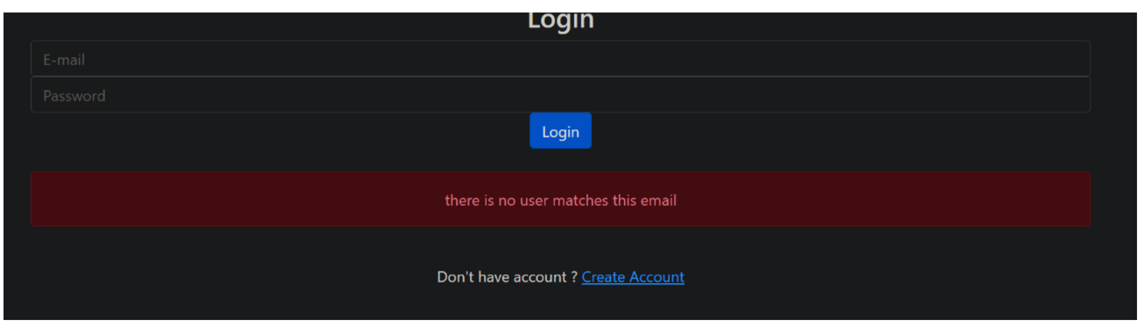
*TEST**11***



password must be at least 6 charachters

Already have account ? Login

*TEST**12***



Login

E-mail

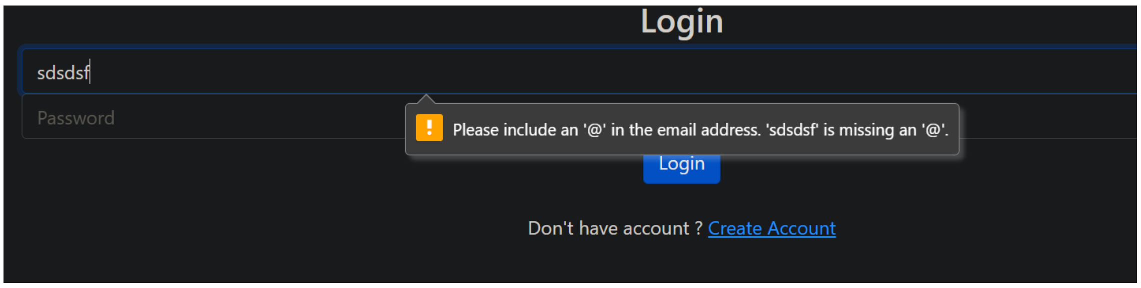
Password

Login

there is no user matches this email

Don't have account ? Create Account

TEST13



Login

sdsdsf

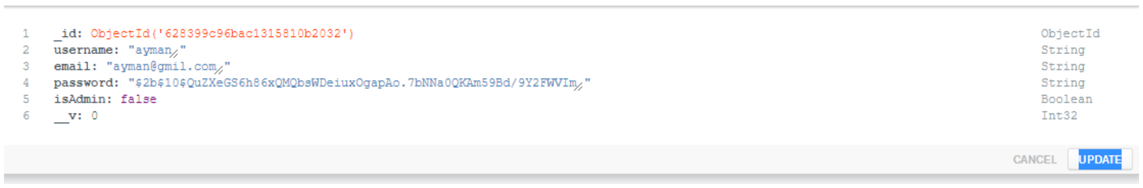
Password

Please include an '@' in the email address. 'sdsdsf' is missing an '@'.

Login

Don't have account ? Create Account

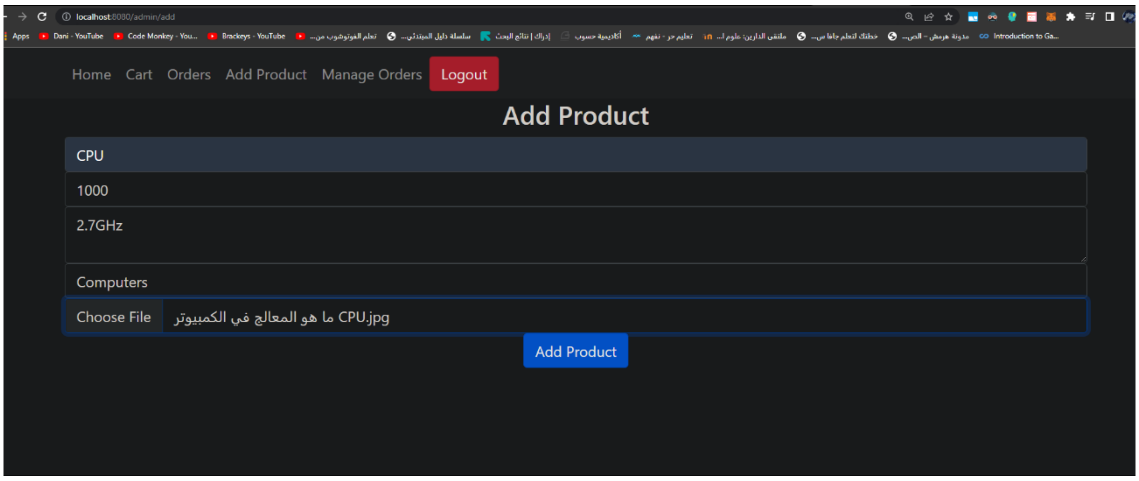
TEST14



```
_id: ObjectId('628399c96bac1315810b2032')                                   ObjectId
username: "ayman"                                                            String
email: "ayman@gmil.com"                                                      String
password: "$2b$10$QuZXeGS6h86xQMQbsWDeiuxOgapAo.7bNNa0QKAm59Bd/9Y2FWVIm"     String
isAdmin: false                                                               Boolean
__v: 0                                                                       Int32
```

CANCEL UPDATE

TEST15



Home Cart Orders Add Product Manage Orders Logout

Add Product

CPU

1000

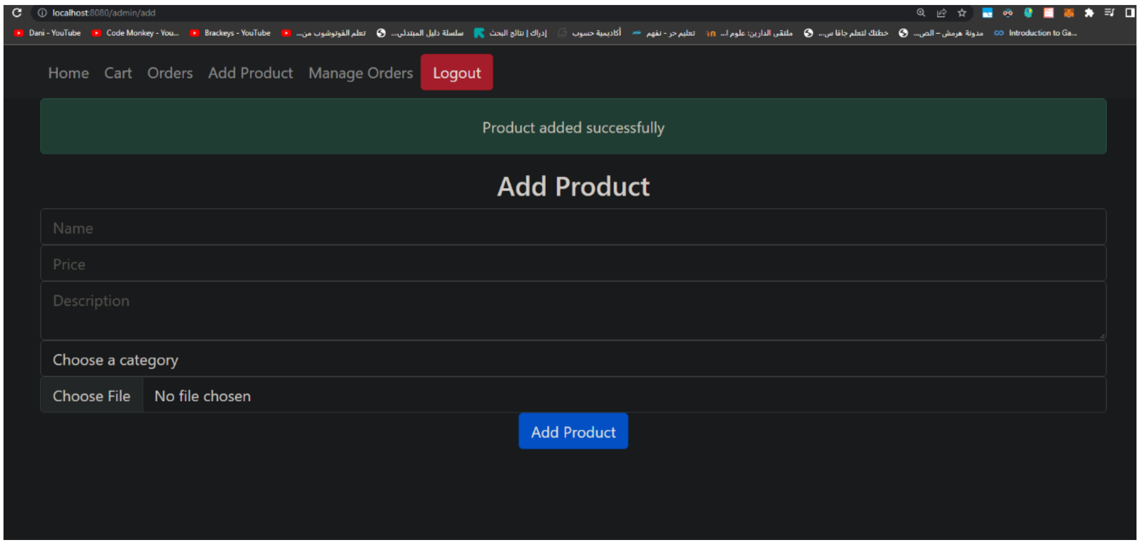
2.7GHz

Computers

Choose File ما هو المعالج في الكمبيوتر CPU.jpg

Add Product

TEST16


localhost:8080/admin/add

Home Cart Orders Add Product Manage Orders Logout

Product added successfully

Add Product

Name

Price

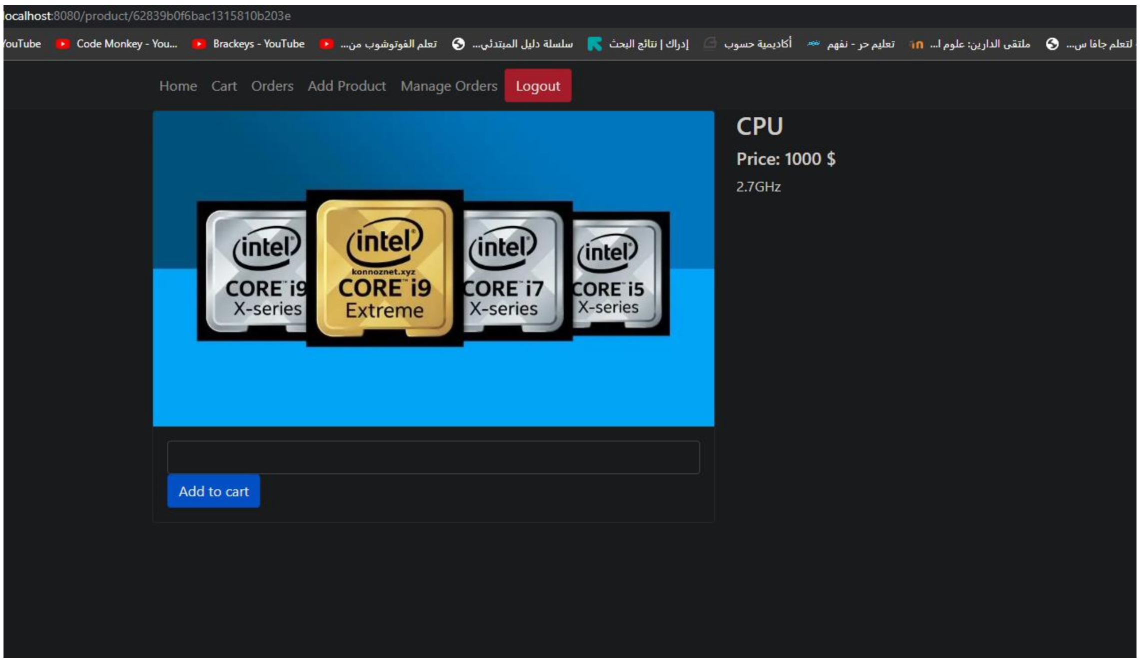
Description

Choose a category

Choose File No file chosen

Add Product

TEST17


localhost:8080/product/62839b0f6bac1315810b203e

Home Cart Orders Add Product Manage Orders Logout

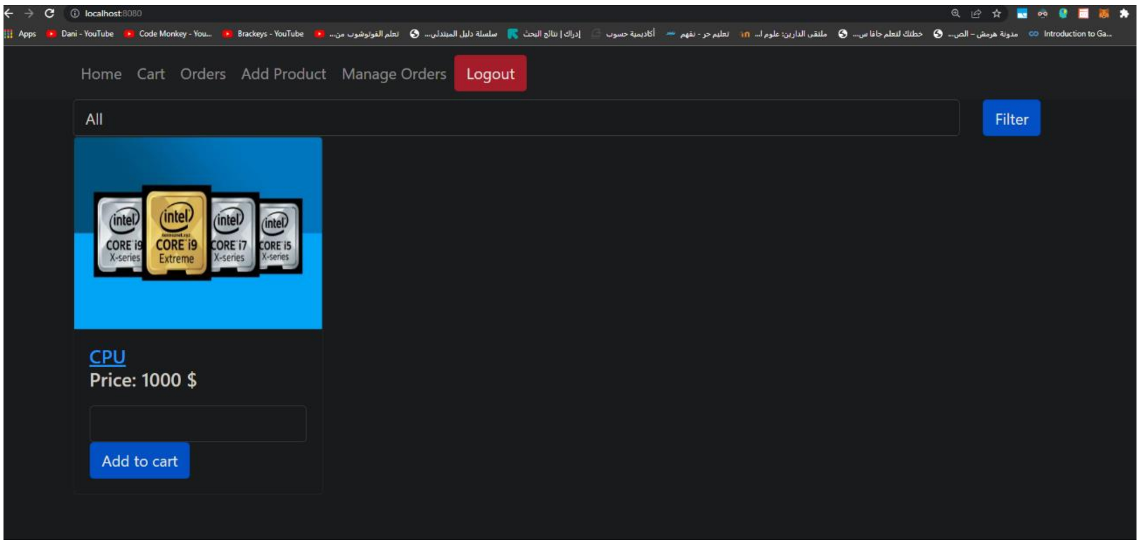
CPU

Price: 1000 $

2.7GHz

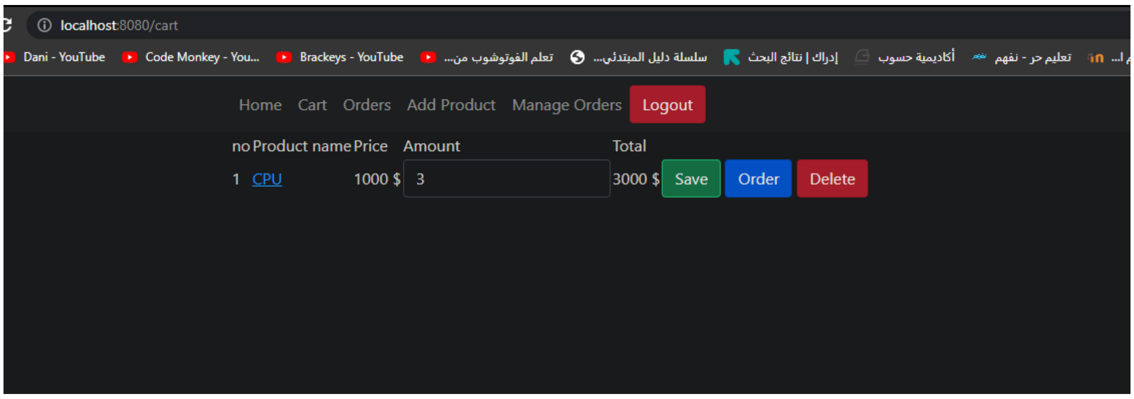
Add to cart

18 TEST



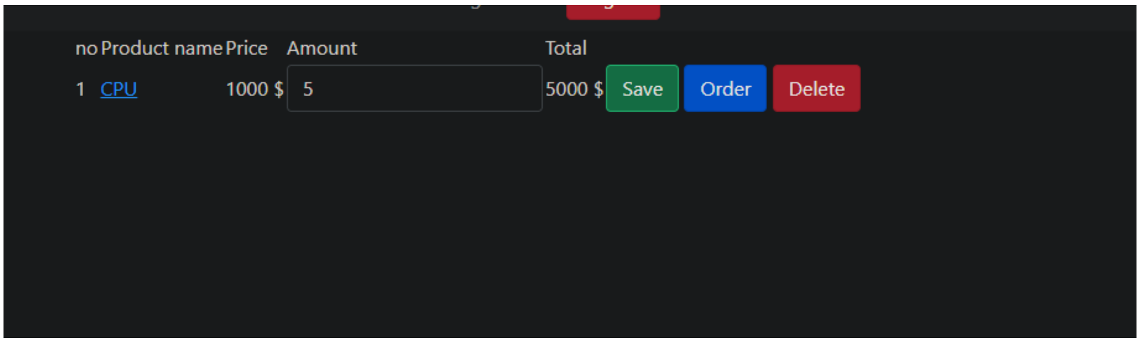
Price: 1000 $

TEST19



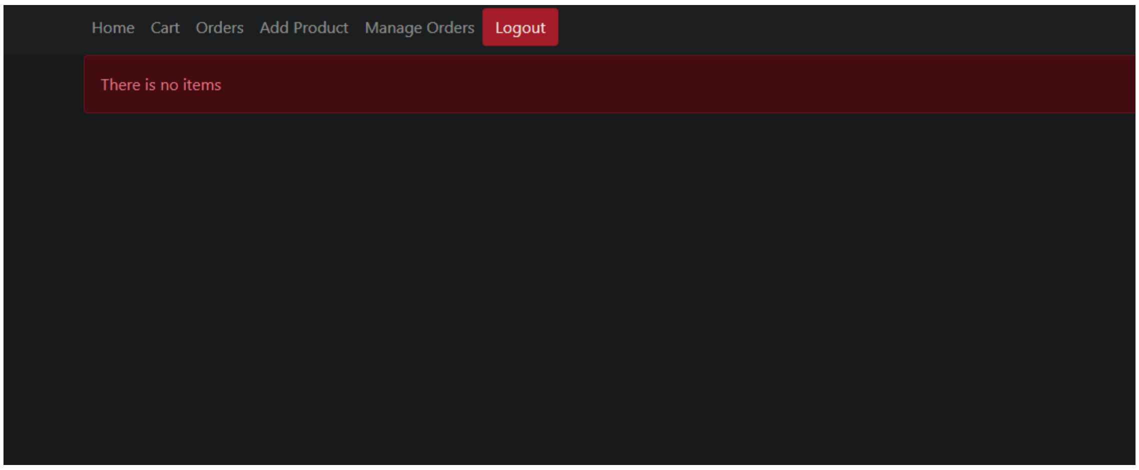
no	Product name	Price	Amount	Total
1	CPU	1000 $	3	3000 $

20 TEST



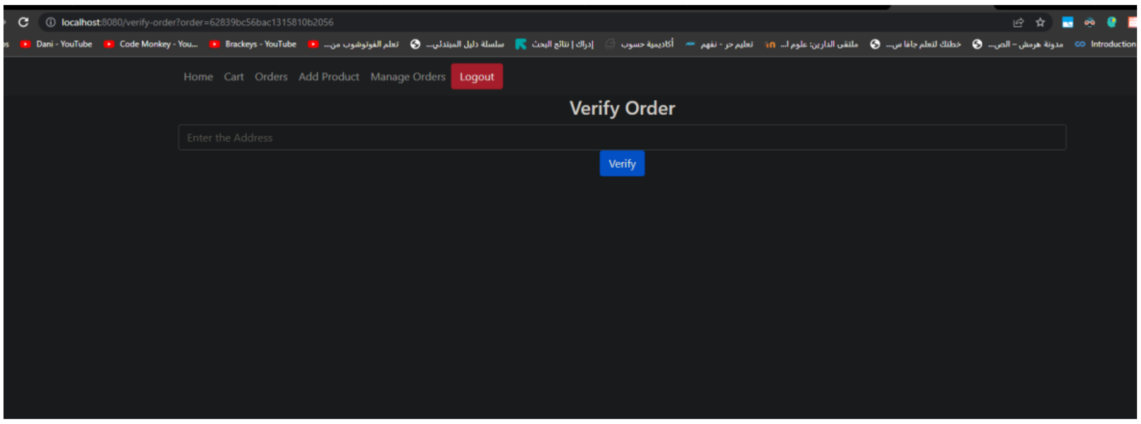
no	Product name	Price	Amount	Total
1	CPU	1000 $	5	5000 $

TEST21



There is no items

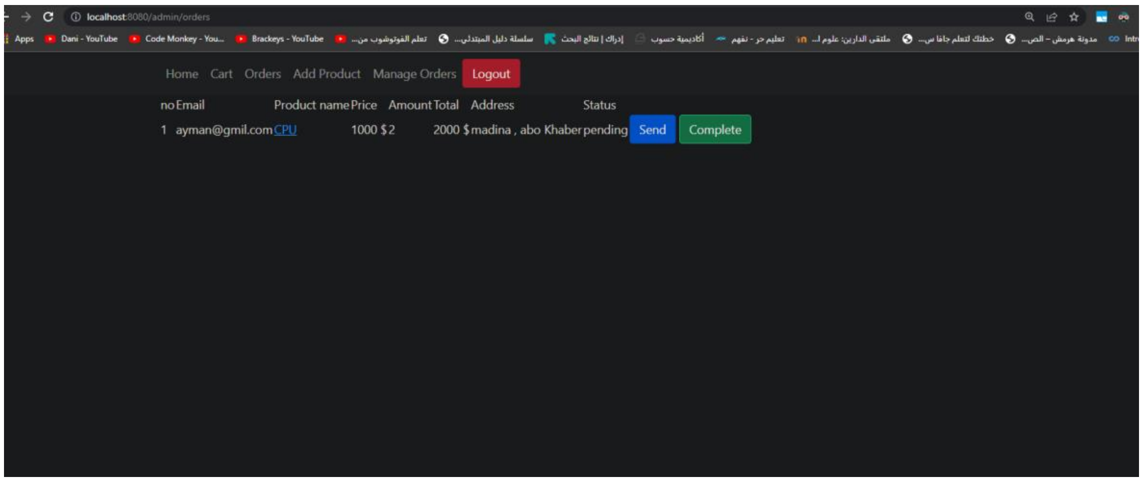
TEST22



localhost:8080/verify-order?order=62839bc56bac1315810b2056

Verify Order

TEST23

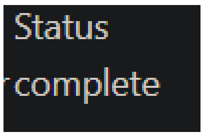


localhost:8080/admin/orders

no	Email	Product name	Price	Amount	Total	Address	Status
1	ayman@gmil.com	CPU	1000 $	2	2000 $	madina , abo Khaber	pending

TEST24

TEST25



Status
complete

*TEST**26***

الخاتمة

في خاتمة هذا البحث، الحمد لله رب العالمين الذي وفقنا لختام هذا البحث الذي تحدثنا فيه عن مشروع (متجر إلكتروني)

الملاحق:

- pencil project
- / https://app.creately.com/