EUROPEAN UNIVERSITY OF LEFKE

FACULTY OF ENGINEERING

Graduation Project 2

E-commerce Web Application

Lutfi Katlav

170089

My project is about a bakery shop's e-commerce web aplication, the main idea is to sell products to reach a wider customer base. It will be used because it is easier and more practical to order from a web page and the goal is to reach more customer if user doesn't have time to go and buy product physically.

Supervisor

Asst. Prof. Dr. Vesile Evrim

13.06.2022

Table Of Contents

1.Introduction

1.1 Problem definition

The first e-commerce was made by Pizza Hut in 1994. Thus, Pizza Hut became the "first company to do e-commerce" by selling pizza over the internet. E commerce is not only for the business to costumers(B2C) but there is e commerce between business to business(B2B) trade aswell. And e commerce is being used by all type of companies e.g. grocery markets, flower shops, hunting shops, galleries and so on. It is because in order to compete globally and to be in market. Also during the COVID-19 most of the physical shops couldn't open their shops but internet sales has increased by 30% (April 2020) .[2]

E-commerce is being used and developing since 1994, started by pioneer Jeff Bezos and his company named Amazon, which is among the most popular and well-known e-commerce web sites.

Today, almost every shop, bakery, restaurant, shopping centre etc. have own e-commerce web site, such that when a business doesn't have a website it is being weirded by people.

In my project as a bakery shop e-commerce, it is important to have one to make a name, reach a wider customer base, aiming to increase the customer base with shipping without opening another shop. Also in my website user can register, log-in, and order a product easily and effectively. Although there are several types of e-commerce, my project covers business-to-consumer(B2C) type which is when consumers want to buy online.

The customer, who wants to buy the product, can find the product he/she is looking for in seconds and add it to the shopping cart easily, and can get the desired product, from time to time with special discounts only for the online without wasting time and energy. The aim is to sell more products and increase the profit rate, as well as to create a brand that is known in the city, region, country or even worldwide.

Example-Problems :

- High rental prices, employee, electricity, water, security, expenses .
- Lack of accessing customer comparing to online shop.
- Client may want 24/7 shopping opportunity.
- Client may not have time, has a health problem or simply does not want to go out.

1.2 Goals

- Goals of using e-commerce is the same as goals of businesses wants of grow and make profit.
- Reach wider costumer audience.
- Build a recognizable brand amount the aimed costumer.
- Aiming for more profit with less capital.
- Considering the customer's comfort.
- To provide the opportunity to compare a product's equivalent on the online.

2. Literature Survey

As an example of similar projects that I want to build is an organization that sells sports supplements they built a web site that all the products that customer wants to buy, other brands that sells various clothing items, foods etc. Today we can find anything that is exist in physical stores we can find them on online too.

Similar projects and my project have common functionalities, such as; user registration, user login,

user profile, update user informations, add to shopping cart, add shipping address, select payment method and pay, see past orders on user profile and their details when clicked. Also have admin panel such as; list all product, add, delete and update exist product, list all users, update their name, email and assign to admin, and lastly admin can see list of orders their details(if it is paid, name, surname, order id, user id, shipping address etc.) and set the order as delivered if it is delivered to user.[1]

When I compare my project with other web sites, it has similar functions such as;

Compare1 : I added admin account and there is an option for admin panel to add product with details such as; price, name, quantity and image. Update user profil, assign user to admin, see orders given by users, mark them as delivered if it is delivered.

Compare2: Users can not see their user profile and admin panel.

Compare3: Login/Logout Register screen added. If email is already taken, user need to change it to different mail.

Compare4 : User profile to see user information and if user wants also can update it’s name, email, passwords. User needs to know currently used password to do that.

Compare5: Every product has it’s own details page, such as larger image, detailed description, price and stock info (if it is in stock).

Compare6: After adding to cart, user can see shopping cart, inside of the shopping cart user can see products, total product price, increase or decrease product quantity by clicking + and – buttons and also can remove it from shopping cart completely.

Compare7: If user logged-in can continue process to buy the product which is in shopping cart by clicking continue button, after that user needs to enter it’s shipping address (city, district, street, postal code, detailed address information and contact number).

Compare8: If user fills the shipping address can finally see overview&checkout screen, in this screen; it is provided entered shipping address information, cart information and order summary with shipping price. If user wants to continue, can select “Credit Card” as a payment method and enter test credit card information.

Compare9: After payment is successfully done user can see orders in user profile under the “MY ORDERS”, also can click “Details” button to see order details on different screen.

3. Background Information

3.1 Required & Used software

- **React.js:**
  React is a JavaScript library for rendering UI components, it is working logic is to divide every screen into components and render as a single page application. I will use React because it is the most powerful and popular library for JavaScript.
- **Express.js:**
  It is a framework of Node.js based web or mobile applications. I used Express to ease my work on listening request on a port. I used this framework to communicate with server via routers, it has simple and understandable syntax.
- **MongoDB:**
  MongoDB for NoSQL database for storing products, customers and orders data. Since it is an open-source database which means it is different than SQL databases such as MySQL, Oracle etc. I used MongoDB for easy to use, fast, and since I don’t work with complex datas with rules. It has large ecosystem that uses it, that means when there is a problem I can find a solution easily.Also it is document based database which uses BSON and sends data as JSON which is easy to work with. I created an account and created a cluster after that I downloaded MongoDBCompass which is an GUI for MongoDB.
- **Visual Studio Code :**
  Excellent code editor to work on a project, it amongs the most common used code editors in the world. Also it is easy to work on.
- **Mongoose:**
  Mongoose is a framework of MongoDB, I used Mongoose to communicate with database with more understandable syntax and creating different data models
- **Axios:**

  It is a library of JavaScript, I used this library on client side to communicate with server side, efficient use of PUT, GET, POST,DELETE, requests on client side with different HTTP request.

- **HTML/CSS/JS Knowledge:**

It is mandatory to have knowledge of HTML/CSS and JavaScript in order to work on this project, because React is a library of JavaScript and also it uses JSX syntax which is uses HTML with JavaScript codes and also for styling I used quite many inline CSS in this project.

- **Npm Packages:**

  Stripe: For use credit card API, bcrypt: for hash passwords, bootstrap: for styling, concurrently: for run both backend and frontend together at the same time, dotenv: for store secret, key or url init, jsonwebtoken: for authorization, multer: for upload images when creating or updating a product, nodemon: whenever I save a file it restarts automatically both server and client, react-dom: for routing purposes (BrowserRouter as Router, Routes, Route) used in App.js to give compenents a route.

3.2 Other software

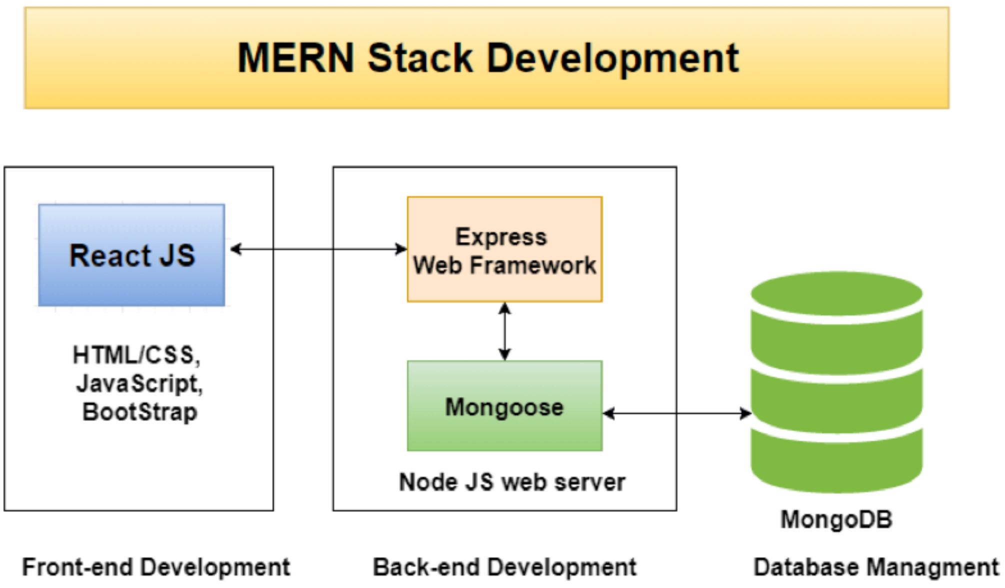
- **AdobePhotoshop :**

  For designing poster.

- **Git :**

  Used for bitbucket commits.

4. Design Documents



5. Methodology

After npx create-react app ;

Screens:

Index.js: Renders App.js

App.js : It routes components and pass data to some of the child components also have cart functionality init.

Home Screen: Fetched product data with axios and mapped the all data to send each data to Product.js component and render the data there to screen to show all products in database.User can add to cart from home screen by clicking "ADD TO CART" button.

Product Screen: Fetched product by it's id and showed the product's name, image, description, price and stock information. User can also add to cart 1 product or more than 1 product by clicking "Add to Cart " button. And also there is a go back button which navigates to homes page when clicking on that button.

Cart Screen: It's functionality comes from App.js, handleClick() is to add to cart functionality; if product exist only increase the number of it, if doesn't exist add to cart by localStorage and set the quantity of it to 1. handleChange() is to increase or decrease the quantity of the product functionality, if it's quantitity is 0 set to 1 and if + increase if - decrease. handleRemove() if user wants to remove the product from cart user can do it by clicking Remove button it simply filters the array. handlePrice() is to calculate price of the individual product and sum of all products.

Register Screen: If user doesn't have an account, can create an account by entering name, email, password, confirm password. It checks if the paswords are same and also checks if email is already taken. Inside of the form tag with the help of the React hook of useState I assigned every field to them and post the data with axios to backend after submit with handleSubmit function, if user already logged-in can not see this page. If register is successful it redirects to login page. If it's not successful it shows an alert on the top.

Login Screen: If user have an account, simply can enter email and password after entering data it is read by onChange event and assign it to the useState hooks. And post the entered data to backend to compara entered data with registered users, if user didn't register before it warns the user, and if user wants to register, user can click to sign up text and it redirects it. If user successully logs-in, it will redirect to the homepage, if it fails it will warn the user based on the error(Invalid password or Invalid email and/or password).

User Profile Screen: With the help of the useEffect I fetched the data from backend with axios and assign it to useState, and filled the empty texts as user info, if user wants to change it's name, email or password can do it very easily after entering currently used password. And also user can see it's orders under the MY ORDERS tag again I fetched it with axios from backend orderRoutes. User can click on Details button if wants to see detailed version of it's past order.

Shipping Address Screen: If user wants to continue to process to buy, firstly needs to enter address and contact number information.(City, District, Street, Postal Code, Detailed Address, Contact Number fields.) After getting all entered data with useState, I assigned all data to localStorage shipping Address and if all the fields are entered and clicked on continue it navigates to payment screen.

Payment Screen: User sees an overview of shipping address, order details and order summary(product price with shipping price) and after user is sure abot all of the info that user entered, select payment method and click on continue button, after that continue button will be disabled, order has been created with the isPaid is false and stripe API shows up which is a test credit card API with the Order ID on the top of it after entered the Card number MM/YY CVC and click on Pay button it will send a put request to backend to change the isPaid true. After it is successfully paid button will be changed to Check My Orders, if user clicks on that button it will redirect to user profile screen to see all past orders.

Order Detail Screen: User can see detailed order screen by clicking Details on user profile under the MY ORDERS tag. This page covers shipping address details, product details, and order summary.

It is fetched with axios get request to backend orderRoutes with passing order params.

Admin User List Screen: It is fetched from axios get request from backend userRoutes, it fetches all the registered user info such as name, id, email, isAdmin, created, updated times. Admin can remove a user with axios delete request by passing user id.

Admin Edit User Profile Screen: When admin clicks on Edit button it redirects to admin/user/edit/userId location, it fetches user information and fills the empty texts, Admin can set user’s name, email or admin status by entering admin password.

Admin Product List Screen: Admin can fetch all the products that has been created before with the help of axios get request to backend productRoutes, and takes product info such as product id, name, category, rating, reviews, price, stock, created at, updated at. If admin wants to delete a product, admin can simply click on delete button, it will fire handleDeleteProduct function and deletes it in a second.

Admin Edit Product Screen: It fetches data with the help of axios get reques to backend productRoutes and fulfills the empty texts with setting useState with the data comes from backend. If Admin wants to update the product, simply can rewrite desired info or select new image. After set the new info it will be submitted to backend productRoutes with axios put request.

Admin Create New Product Screen: When admin wants to add a new product, simply can click on Create a Product button and it redirects it to admin/create-product screen, after admin fills the name, category, price, stock info, image, description, admin needs to enter admin password to create a new product. And it will be send backend productRoutes with the help of axios post request.

Admin Order List Screen: Admin can see all the orders made by users, and see order id, user id, username, price, paid info, ordered at, delivered, and delivered at info. If admin sets an paid order to mark as delivered, it will send a put request to backend orderRoutes with axios and it will change the value of isPaid to true and a delivered time assignes after clicking on it. After clicking button it will be disabled(it will be disabled if order is not paid too).

Admin Order Details Screen: If admin wants to see detailed screen of ordered made by user, admin can click on Order Details button and it will redirect to order details screen with the shipping details(name,province etc.), order details(product), and order summary (product price, shipping price, total price).

Components:

Header component: It is a navbar that shows brand(if clicked redirects to main page), cart button and if user not logged in Login/Signup button, if user logged-in it will shows user name, if user clicks on it's name it will dropdown and there will be two options; Profile and Logout. If admin logs-in and clicks on it's name, admin will see Profile, User List, Product List, Order List and Logout with dropdown.

Rating Component:

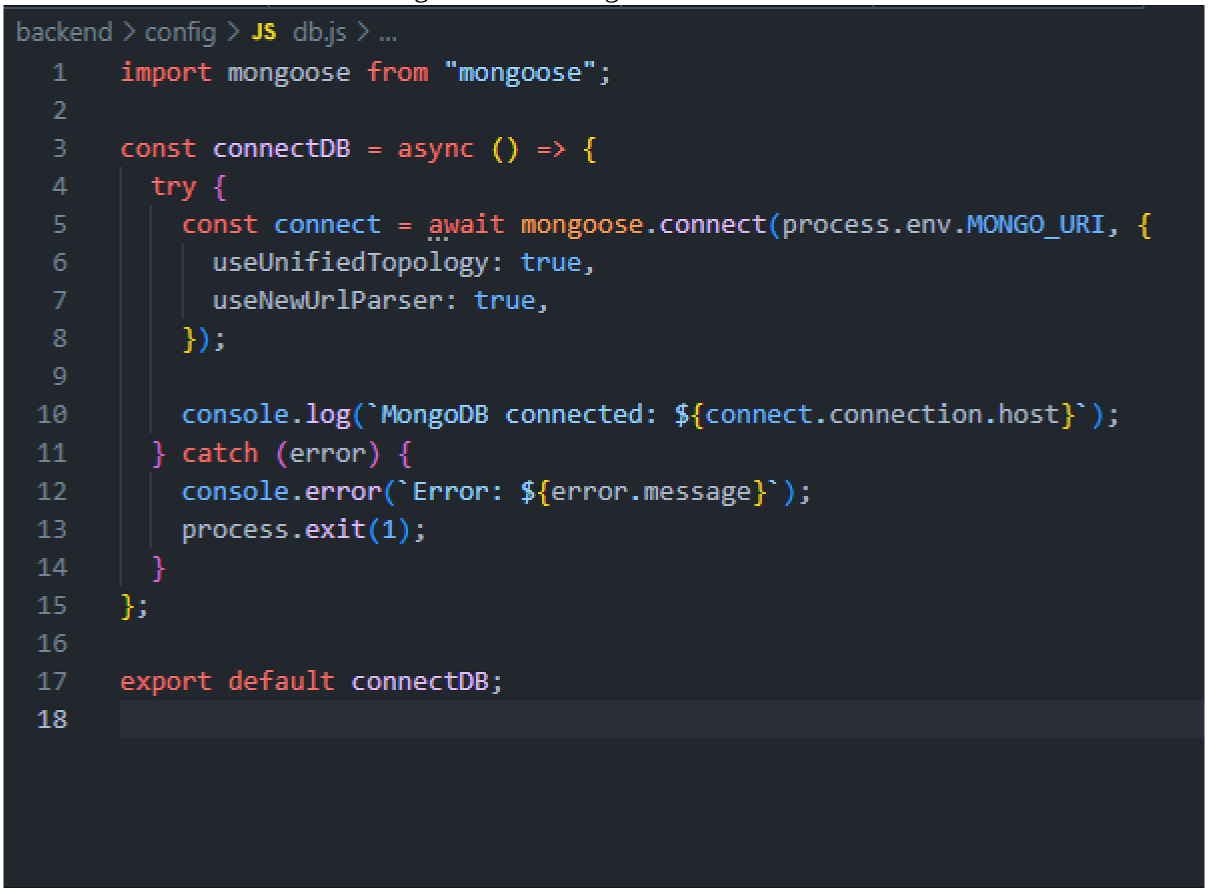
This component made for assigning stars according to rate that user gives, if user gives 4.5 start it will fulfills 4.5 stars.

Footer Component:

It will have business name with copyright text.

In Public folder: There is an images folder with index.html

In backend/config folder there is an db.js file to import mongoose and set a function named connectDB. This will connect mongoose with MongoDB.

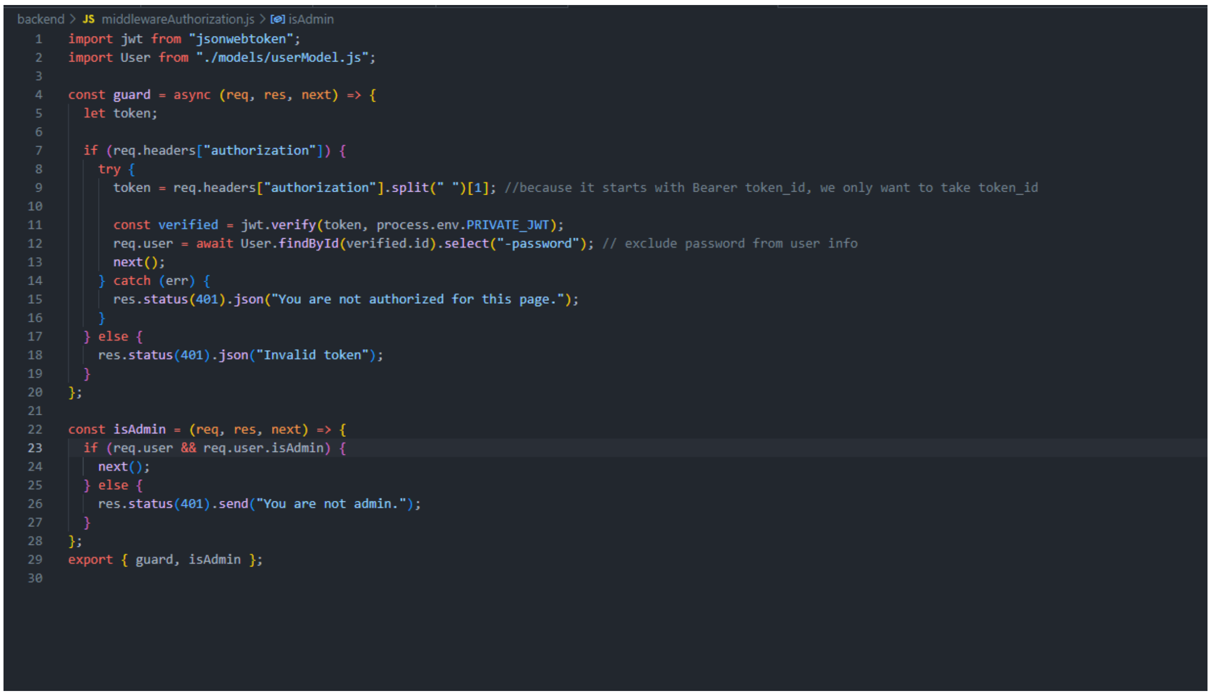


```
import mongoose from "mongoose";

const connectDB = async () => {
  try {
    const connect = await mongoose.connect(process.env.MONGO_URI, {
      useUnifiedTopology: true,
      useNewUrlParser: true,
    });

    console.log(`MongoDB connected: ${connect.connection.host}`);
  } catch (error) {
    console.error(`Error: ${error.message}`);
    process.exit(1);
  }
};

export default connectDB;
```

In backend/data folder there is startup dummy data as product.js and users.js.

In backend/models folder; there is 3 mongoose models file;

In backend/midllewareAuthorization: It is for authorization it checks if the user’s token is valid, and also it check is the user isAdmin.

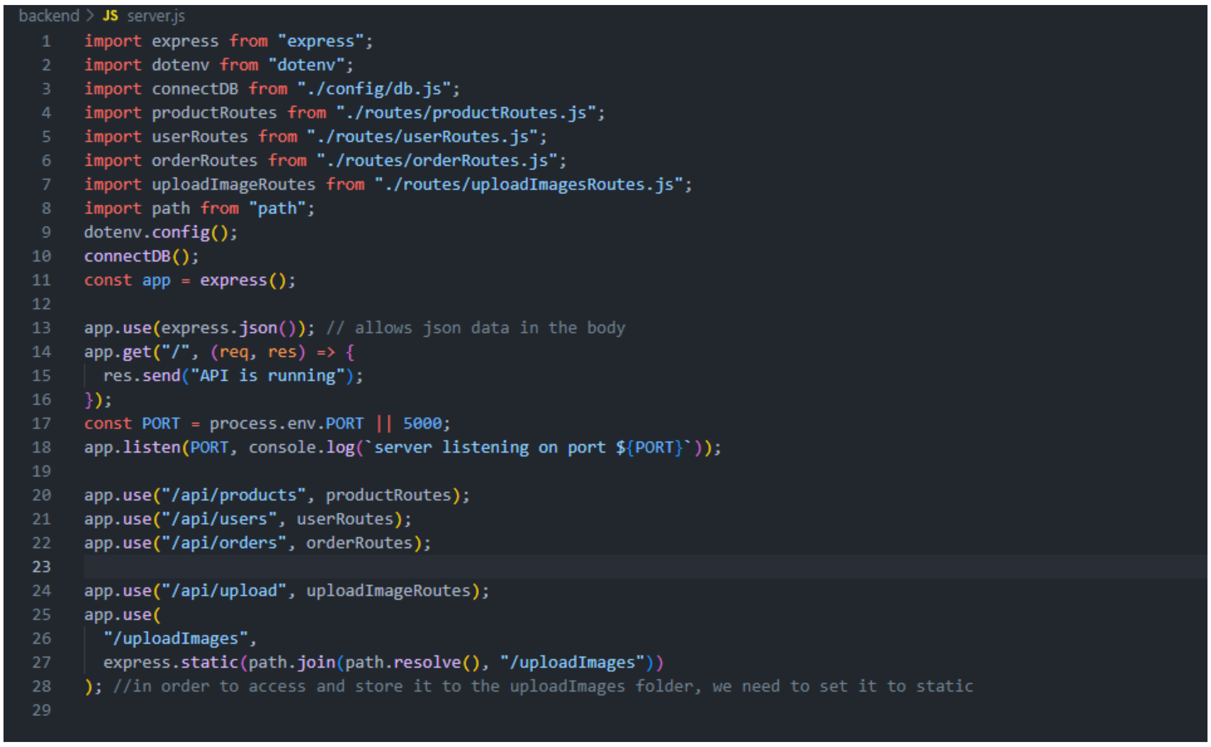


backend > JS middlewareAuthorization.js > isAdmin

```
import jwt from "jsonwebtoken";
import User from "./models/userModel.js";

const guard = async (req, res, next) => {
  let token;

  if (req.headers["authorization"]) {
    try {
      token = req.headers["authorization"].split(" ")[1]; //because it starts with Bearer token_id, we only want to take token_id

      const verified = jwt.verify(token, process.env.PRIVATE_JWT);
      req.user = await User.findById(verified.id).select("-password"); // exclude password from user info
      next();
    } catch (err) {
      res.status(401).json("You are not authorized for this page.");
    }
  } else {
    res.status(401).json("Invalid token");
  }
};

const isAdmin = (req, res, next) => {
  if (req.user && req.user.isAdmin) {
    next();
  } else {
    res.status(401).send("You are not admin.");
  }
};
export { guard, isAdmin };
```

In backend/seeder: it is dummy file, at first I created it to push and delete data to test database.

In backend/server.js: it is the server file it listens to the port and activates the routes.



backend > JS server.js

```
import express from "express";
import dotenv from "dotenv";
import connectDB from "./config/db.js";
import productRoutes from "./routes/productRoutes.js";
import userRoutes from "./routes/userRoutes.js";
import orderRoutes from "./routes/orderRoutes.js";
import uploadImageRoutes from "./routes/uploadImagesRoutes.js";
import path from "path";
dotenv.config();
connectDB();
const app = express();

app.use(express.json()); // allows json data in the body
app.get("/", (req, res) => {
  res.send("API is running");
});
const PORT = process.env.PORT || 5000;
app.listen(PORT, console.log(`server listening on port ${PORT}`));

app.use("/api/products", productRoutes);
app.use("/api/users", userRoutes);
app.use("/api/orders", orderRoutes);

app.use("/api/upload", uploadImageRoutes);
app.use(
  "/uploadImages",
  express.static(path.join(path.resolve(), "/uploadImages"))
); //in order to access and store it to the uploadImages folder, we need to set it to static
```

1.userModel: it is an user model, first I created an user schema with name, email, password, and isAdmin and assign it to a model.

2.productModel: it is a product model with 2 schema. 1. reviewSchema with name, rating and comment of product object of objects. 2. productSchema with user.id(comes from userModel which admin created or updated this product), name, category, image, description, reviews, rating, numReviews,price, countInStock and defaultCartStock(the reason I created this to handle cart screen)

and assigned it to model

3.orderModel: it is an order model with user.id(which user ordered), orderedItems array(name, image,price,defaultCartStock, product), shippingAddress array(fullAddress,city,district,street, postalCode, contactNumber), paymentMethod, shippingPrice, itemsPrice, totalPrice, isPaid, paidAt, isDelivered, deliveredAt and assigned it to mongoose.model.

In backend/routes folder there are 4 routes file. I created them to fill models and manipulate the data with what I want

1. userRoutes: I produced a token with JWT(JSON web tokens) to authorize users,

   router.post("/login) : it takes email and password from request.body and search if it is in the database, if finds it with email match it compares with passwords if password is matches too it sends data back with the new produced token.

   router.post("/register"): it takes name,email and password from user and searches if it is already exist in database, if it is exist in database it returns res.status(400)
   if not, it will create a new user and set it's simple password to bcrypt hash password to make the password stronger. After user created it will send data back with produced new token.
   router.route("/profile).get : It is protected with auth middleware because it is private page, it take req.user._id and sends data back.

   router.route("profile").put : it is protected with auth middleware because it is private page, it updates the user with the requested data, if user changes it's name, email or password with the correct entered currently used password, it sends data back with new set values.

   router.route("/").get: it is protected with auth middleware and also it is protected with isAdmin, it means user can not get the data if it is not admin, basically it sends all user list back.

   router.route("/:id).delete: it is protected with token and isAdmin auth, it's for deleting user

   from database by an admin account.

   router.route("/:id").get : it is protected with token and isAdmin auth, it's for fetching desired user data.

   router.route("/:id").put : it is protected with token and isAdmin auth, it's for updating user data by an admin account.

   2.productRoutes:

   router.get("/") : send all products in database.

Router.get("/:id): send desired product's information

Router.route("/").get : it is protected with token and isAdmin auth, it's for fetching all products.

Router.route("/:id).delete : it is protected with token and isAdmin auth, it's for deleting a specific product by an admin acc.

Router.route("/:id").put : it is protected with token and isAdmin auth, it's for updating a specific product's information.

Router.route("/new-product").post : it is protected with token and isAdmin auth, it's for creating a new product.

3.orderRoutes:

Router.route("/").post : it is protected with token auth, user can give an order with this route.

Router.route("/").get : it is protected with token auth, user can see all the past orders that user made.

Router.route("/:id").put : it is protected with token auth, when user pays successfully it sets isPaid to true.

Router.route("/myorders").get: it is protected with token auth when user wants to see all past orders.

Router.route("/id").get : it is protected token auth, user can see order details

Router.route("/deliver/:id").put: it is protected with token and isAdmin auth, it's for whenever the order is delivered it sets isDelivered to true.

4.uploadImagesRoutes: it is created for when admin wants a create new product or update a product's image, this route helps to do that function with the help of npm multer.

uploadImages folder is for uploadImagesRoutes

-Risk Analysis

Customer's privacy and security may not be secured 100%.

Returned products by customer may not be acceptable.

Shipment may go to the wrong address.

Custommer service may not be able to solve customer's problem.

Payment by customer may not work as supposed to be.

Maintanance may be too long.

The site may not work properly in high traffic.

User may not register or login.

If user is old, user may not understand how the site works

6. Conclusion

6.1 Benefits

a. Benefits to users :

1. Customer saves time.
2. Comparison with subtitutes to find the one with better price.
3. More Variety.
4. Saves money to not going with transportation.

b. Benefits to me :

1. I have gained experience on web development.
2. I have learned working logic of an e-commerce web application
3. I understood how to combine different technologies into together.

6.2 Ethics

• Responsibility is to hold the decisions that business made and act according to that decisions. The idea is the accept the duties[4].
• Accountability is businesses have to take account on the consequences of the actions, individual or the whole organization. “Systems and institutions in which it is impossible to find out who took what action are inherently incapable of ethicalanalysis or ethical action” (Grewal & Shivani 2012) [4].
• Liability is widen the meaning of the responsibility and accountability, it is related with political regulations[4].

- Results also show that reliability/fulfilment and non-deception are the most effective relationship-building dimensions. In addition, relationship quality has a positive effect on buyer repurchase intentions and loyalty.[7]
- The current study examines the differences between e-commerce ethics between the UK and Egypt in the context of B2C and B2B e-commerce. The conceptual model is then tested with a total of 980 completed questionnaires collected from two sample countries; namely, Egypt and the UK. These were analysed through a multivariate analysis using a variance-based statistical technique known as Partial Least Squares Structural Equation Modelling. The findings of this study show significant support for the proposed model. As predicted, BPSE is a second-order construct composed of seven dimensions (i.e., security, privacy, fulfilment, non-deception, service recovery, communication, and shared value). Trust and commitment mediate the relationship between BPSE and satisfaction. In addition, II reliability/fulfilment and non-deception are the most effective dimensions in BPSE. Byer’s perceptions regarding sellers’ ethics (BPSE) has a significant influence on consumer satisfaction. No major differences between the two country models were found.[8]

Why did I choose this project?

I choosed this project because I want to master on web development and want a career on this area, so I thought after graduation this project would be a good reference and practice for me, also I will be pleased to learn and implement all of them in one project. I choosed this project to push my limits in a limited time and learn new technologies and also always wanted to create a website that has functionalities in it. The main idea here is to understand working logic of an full stack web application and how to implement it.

6.3 Future Works

I will continue to work on this project, I will make user interface better, add more functionalities such as; communication API between admin and user, add session/cookies, make security more solid, make redirects better, make requests better, will learn and implement Redux, will make cross platform mobile application of it.

7. References

[1]https://www.westga.edu/~bquest/2011/ecommerce11.pdf

[2] https://www.researchgate.net/profile/Ahmed-Khan-67/publication/342736799_E-commerce_trends_during_COVID-19_Pandemic/links/5f04603c458515505091c291/E-commerce-trends-during-COVID-19-Pandemic.pdf

[3]https://dergipark.org.tr/en/download/article-file/1123318

[4] https://www.theseus.fi/bitstream/handle/10024/119487/Final Thesis-KhanhNguyen.pdf?sequence=1&isAllowed=y

[5] https://www.formationsdirect.com/blog/physical-retail-stores-vs-online/

[6] https://www.indeed.com/career-advice/career-development/what-is-ecommerce

[7] http://irep.ntu.ac.uk/id/eprint/33173/2/10694_Agag.pdf

[8] https://pearl.plymouth.ac.uk/handle/10026.1/4588