**BACKEND**

**Assignment-7(E-Commerce)**

-Praveena Manikandan

**BACKEND**

**CAPSTONE PROJECT(Twitter-Clone)**

-Praveena Manikandan

15-Aug-2023

FRONTEND:

Created a folder Frontend.

In Frontend React is used in App.js.

In index.html copied and pasted react bootstrap link and installed the packages react-router-dom and react-bootstrap.

**FRONTEND:s**

**APP.js**

* Create a function using the name App
* Used Routes for route the various pages.
* Import various pages and react-router-dom.

**Components**:

In the components folder

1. **Home.js**
2. **Post**.js
3. **Card.js**
4. Layout.js
5. Profile.js
6. ProfilePic.js
7. UserProfile.js
8. Sidebar.js
9. Home.css
10. Profile.css
11. Sidebar.css

Pages: created a pages folder

1. **Register.js**
2. **Login.js**

**Redux:**

1. Combine Reducer
2. Store
3. User Reducer

**Layout:**

1. **Home**

* Post a tweet in a home page using MODAL from bootstrap.
* Upload image or content or both image and content using form data with post method create tweets for uploading the form data.
* Get a token from authorization.
* Get a user from redux.

1. **Post**

* **After** uploading tweet from home get all the tweets from database using getalltweets from api and delete the tweet using delete post with delete api.

1. **Card**

* Used props for get all the tweets and delete the tweet from Post.
* In card used fontawesome icon for like,comment,retweet.
* Used moment for getting date and time and used useselector for getting user state.
* For getting likes and dislike, use likedislikepost api used.
* For comments set comment box is used and used key event ,while press enter key comment is posted.
* For retweets,used retweet api for getting the data and after retweeted Retweeted with id is displayed.
* For delete the tweet ,used deletepost api is used for getting the details.
* Used toast and toast container for displaying the message.

1. **Layout**

* Layout is used for getting the sidebar and home page.

1. **Profile**

* In profile page ,author image, name ,dateofbirth and location is displayed.
* For displaying date of birth in DD/MM/YYYY,formatted date is used.
* Used Upload button for uploading the image used MODAL .
* Used Edit button for editing the name ,dateofbirth and location is updated from updatedetails and used redux for updating the details used dispatch.
* Used fontawesome icon for location and dateofbirth.
* Used map function for getting user tweets.
* Displayed following and followers details from userprofile.

1. **ProfilePic:**

* **For** Uploading profile picture used Modal.
* used put with upload profile api for changing profile picture.

1. **UserProfile:**

* Other userprofile is viewed and displayed name ,location,dateofbirth is displayed.
* used button with follow and unfollow while click follow followers value is increased in profile page and following is increased.
* Displayed usertweets from usertweets api.

**Viii Home.css**

* Home page is styled using css.

**iX Profile.css**

* Profile page is styled using css.Follow unfollow button is hovered and profile pic in rounded shape.

**X sidebar.css**

* Sidebar is styled using css with position sticky.

**Xi Sidebar.js:**

* Sidebar is displayed in all the pages with col-md-3 size.
* Used Navlink for navigating home,Profile page.
* Used Logout button –while logout remove token from localstorage.
* Displayed profile pic with name and username.

**PAGES:**

**Register.js:**

* For registration, used form with input type as text for name,email for email id,text for username and password for password with placeholder and used button for register.
* Already have an account login button is used and navlink is used to redirect the login page.
* Used Spinner for loading the details.
* Used toast and toast container for displaying the message.

**Login.js:**

* For Login,used form with input type as email for email and password for password with placeholder and used button for login.
* Or Don’t have an account used Link for register with button.
* Used toast and toast container for displaying the message.
* Used spinner for loading the details.

**REDUX:**

**Actiontypes.js:**

* Export Profile request,profile success,profile failure for updating the profile.

**CombineReducer.js:**

* Import combine reducer from redux and user reducer and get user details.
* Export Profile request,profile success,profile failure for updating the profile.

**User Reducer.js:**

* Import update profile request,success and failure from actiontypes.
* Initialize the user ,error and loading as empty..for getting updated data.
* Set userReducer state s initial state i.e., empty and action for getting updated action.
* Using switch for login success or failure.
* If it is success then state gets updated with the current user.
* If it is error return the empty state.
* Then for getting the profile details profile success state gets updated.
* Set default as initial state.

**Index.js**

* Store and Provider is set.

**BACKEND:**

In backend

SERVER.js

1. Use express,cors and mongoose for mongodb as database
2. Create connection between mongoose with mongodb atlas and express and establish the connection.
3. Use cors and express as json .
4. Create .env file.
5. Then use uploads file for storing the images with express.static.
6. Then use user,auth and tweet with user routes, auth routes and tweet routes.

**.ENV:**

1. Localhost:5500,
2. Used secret key:secret,
3. MONGO\_URL=mongodb://127.0.0.1:27017/twitter

**SEED USER:**

* In seed user created 2 user 1.user
* 2.Admin

**SEED PRODUCTS:**

* In seed products created 10 product using product model .

**CONFIG:**

**Connect:**

* Establish a connection between mongoose with mongodb atlas url.

MIDDLEWARE:

**Protected Resources:**

1. Used jwt as jsonwebtoken for authorization and used JWT\_SECRET for encryption using bcrypt.
2. Created authorization in the headers and replace the token with bearer.
3. If not authorization user is not logged in.
4. Get the id from payload if the id matches then user goes to next.

**Is admin:**

* Check the user is admin using findById and check whether the user!==1 then it is unauthorized or else it will move to next.

**File Upload:**

* Used multer for storing images and allowed only jpg,jpeg and png file upto 5MB.
* Used storage and file destination with folder uploads.

**Models:**

* Category model

1. Created a schema for category with name type as string and used timestamps are current date.

* Product model

1. Created a schema for product with name, description,price,quantity,image,category used reference ,shipping,rating and reviews with timestamps.

* Order model

1. Created a schema for order get products from product model reference,payment,buyer used user model for reference and status with timestamps.

* User model

1. Created a schema for user like name,address,phone,email,password,isadmin set default to 0 and set timestamps.

ROUTE:

1. User routes
2. product routes
3. category routes
4. user routes:

* created routes for login and register as post with authorization and is admin from middleware.
* Created a user and admin as get.
* Created a profile for user as put for updating the details.
* Created an order and all orders as get for getting the order details.
* Get an order Id as put for updating the order status to user.

2. product routes:

* Created a path for downloading the images and file upload for getting images
* Created CRUD operations for product to add new product,get all product,get product by id ,delete product,update product ,update images,product filters,product list,product count,brain tree token and payment.

3. category routes:

* Created a route with get,post,put and delete the category routes.

Controllers:

1. user controller
2. product controller
3. category controller
4. USER CONTROLLER:

* Used bcrypt,jwt,secret key from .env file.
* Get user and order model for the details.

**Register:**

* Used findone for finding the email if the email matches the user is already registered.
* Used bcrypt password with 10 rounds for encrypting the password.
* Create a new for registration with user details.

**Login:**

* Used find one for getting the email and check the password using bcrypt .compare for compare the password.
* Then used jwt-sign for getting the token and check with secret key from env file.
* If it matches then get the user details.

**Profile:**

* For updating the user profile get the user details from the body and used findbyId to get the user id.
* Get the encrypted password and if both matches then update the user by using findby id and update is used for updating the user details.

**Order:**

* Get the user id as buyer from order model and populate the products,user name and used sort for listing the data in descending order using -1.

**All Order:**

* Get all the user order for admin using find for getting all the orders.

**Order Status:**

* After getting order placed using order id and status from body order status gets updated by admin.

GIT HUB LINK: https://github.com/Praveena-manis/Assignment7

15-Aug-2023

BACKEND: deployed in <https://ecommercebckend.onrender.com>

FRONTEND: deployed in: https://main--comfy-boba-22fea8.netlify.app/

Created a folder Frontend.

In Frontend React is used in App.js.

In index.html copied and pasted react bootstrap link and installed the packages react-router-dom and react-bootstrap.

**FRONTEND:s**

**APP.js**

* Create a function using the name App
* Used Routes for route the various pages.
* Import various pages and react-router-dom.

**Components**:

In the components folder

1. **Form**-created a category Form for storing categories.
2. **Layout-**Inside the layout created
3. Admin Menu,
4. Header,
5. Footer,
6. Layout,
7. User Menu
8. header.css
9. **Routes-**Inside the routes created
10. Admin
11. Auth
12. Private Route
13. Slider.js
14. Slider.css
15. Home
16. Home cover
17. Home cover.css
18. Pagination
19. Prices
20. Spinner

**Context:**

Inside the context folder

Created a i. auth

ii. Cart

Pages: created a pages folder

1. **Admin**
2. addproduct.js
3. addproduct.css
4. admin dashboard
5. admin orders
6. create category
7. product.js
8. product.css
9. update product
10. **Auth**
11. Login
12. Register.css
13. Register.js
14. **User**
15. Dashboard
16. Home
17. Orders
18. Product details
19. Profile
20. Star Rating.css
21. Star Rating.js
22. Cart Page

**Redux:**

1. Combine Reducer
2. Store
3. User Reducer

**Layout:**

1. **Admin Menu**

* In the admin menu listed admin operations with Nav Link for
* create Category
* Admin Dashboard
* Add Products
* All Products
* Admin Orders

For their corresponding pages

* Used Font Awesome icon

1. **Header**

* Created a Header with nav-bar and toggle for the responsive
* Used Logo in the header with the height:80 and width:120
* Used form with input type as search and placeholder with button type as submit.
* Used nav-item with nav-link to products, register, login.
* Used nav –link dropdown toggle for displaying user name.
* While logout user details are removed from local storage.
* Used badge and displaying the cart length and used cart font awesome icon.
* Use use Auth Context Api for getting user details in local storage.
* By default set user as null and token as empty.

1. **Footer**

* Created footer with the nav link for various pages and makes responsive.

1. **Header.css**

* Created a various style for form, icon, nav-link and media screen

1. **Layout**

* In layout import helmet,toast-container for displaying toast messages with header,children and footer.

1. **User menu**

* In user menu displayed user operations like
* User Dashboard
* Profile
* Orders
* Used font awesome and link to various pages

**ROUTES:**

1. **Admin routes:**

* Using context api and get method is used and get admin details from the axios and get method from backend.
* Based on token check the user if it matches then it will move to the further link in admin dashboard app.js or spinner will load.

1. **Private Route:**

* Private route is used getting user details from the backend using axios.
* Based on token check the user if it matches then it will move to the further link in admin dashboard app.js or spinner will load.

**Slider:**

* In the slider page created a carousel slider with 4 images per page .
* Make the Slider responsive.

**Home:**

* In the home page, created a layout with header,cover,slider and footer.

**Home cover:**

* In the home cover page, created a banner for large screen and small screen.

**Spinner:**

* Created a spinner for loading and set count up to 3 for one second then navigate to the corresponding location.

**Prices:**

* Created an array with price for filtering with the price range up to 0 to 1000.

**CONTEXT:**

**Auth:**

* Use create context for authentication .the context will hold the authentication state
* Initially the state is set as null after login the state will updated.
* Set the default authorization header.
* Then state and token gets updated in local storage.
* Auth provider.context acts as a provider for the auth context.
* Use useContext hook to access the authentication context.
* Finally export authprovider to index.js.
* In any component we want to access the user information use useAuth().

**Cart:**

* Use creates context and context provider for cart page.
* For storing and updating the cart data and use usecontext hook for cart value.
* The same process repeated for cart instead of user.

**PAGES:**

**Create Category:**

* Use useState for storing the values and used axios and post name for create category.
* The use axios and get for getting the category using get category.
* Used axios and put for updating the category using update category with their ID.
* Used axios and delete for deleting the category using their ID.
* Used category form for updating the category.
* Used map function for listing the category.
* Created button with edit and delete with onClick event using ID.

**Add Product:**

* In the add product, admin have an access to add the product.
* **Using** axios and get for getting category then use useState for storing the input from the form using onchange event with input type as text and number.
* Used input type as file for uploading images in FormData and used append for updating values.
* Linked admin menu and used select for selecting the category and used select for selecting the shipping values.
* Used button with onclick event for adding the product.

**ADMIN DASHBOARD:**

* Created admin dashboard using context api for getting user details and it will check that user is admin or not.
* If the user is admin then it will redirect to admin dashboard page and display the admin name, email address and contact number.

**ALL PRODUCTS:**

* Use useState for storing the values in array and get all the product using axios .
* Use map function for displaying the product details in card.
* Created button with link for navigating to update and delete product with their ID.
* Admin can delete the product and update the product.

**UPDATE AND DELETE PRODUCTS:**

* Used axios and get for getting single product and set the previous product in their corresponding data.
* Used axios and get method for getting category
* Then append the new value in the form data using put and axios the values get updated.
* Using their ID the product gets deleted using delete and axios method.
* Used map function and form for getting the category and updating the values used input type as text and file for image.
* Used EDIT and DELETE button onclick event for their function.

**ADMIN ORDERS:**

* User placed order then admin can manage the order by getting all the order and processed the status to user.
* Get all orders using get and axios method and get order ID.
* After getting the details of the orders used map function for getting the order details and product details.
* Used moment for maintain the current time.

**AUTH:**

**Register:**

* Registration use axios and post method to post user details like name ,email,password,phone,address and password.
* Using container with rows and cols for responsive.
* Using spinner for loading.
* Using forms with on submit event and used input type as text for name address with onchange event for updating the user details while registration and used value for storing and placeholder.
* Used input type as email for email and password for password and tel for mobile number.
* Using button type as submit.
* Using navlink to login page while the user is already registered.

**Login:**

* For login get the registered user details and enter the email and password using form with on Submit and input type as email for email and password for password.
* For loading used spinner from bootstrap.
* After enter the email and password, used post and axios method to check the user details.
* If the user details matched with status then display the success message with toast and navigate to home page and the user details and token in local storage.

USER:

**Dashboard:**

* Created user dashboard and linked user menu and get the user details

**Home:**

* In home page list the products with filters
* Category filter
* Price filter
* Used map method for getting categories in checkbox.
* For category filter used axios and get method is used for getting categories.
* For getting price created price components with array and used map function for getting the price filter using radio box.
* Based on the filter, products will display using map function from get all products.
* For pagination get the product count and set product count per page and product gets loaded using get and axios method.
* After the products get displayed and used ADD to CART button and MORE DETAILS button with on click event for changes.
* While clicking add to cart button, items get added to the cart using cart context api, cart item is updated.
* While clicking more details button, used navigate button to link product details page.
* Page loading used react –icons load more per page 4 products will displayed.

**Product Details:**

* After clicking more details navigate to product details page. In product details page used params.id for getting product details.
* Using id the product image,name,price,description and reviews and used star from font awesome used.
* Used add to cart button for items get stored in cart.

**Orders:**

* Get user order details after the payment is successful get user orders from get and axios method.
* After getting the details, used map function for displaying the user details like product details, id,status,payment status,for getting current date used moment package and quantity.

**User Profile:**

* Maintain user details in profile update the profile details name, email, password, and address using get and axios method in profile.
* Get user details from context api and gets updated in profile.
* Using form and input type for user details using on change event and placeholder is used.
* Using button type as submit for submitting the profile details.

**Star Rating:**

* Used star font awesome icon for rating.
* Used array function and set hover for getting star reviews.

**Cart Page:**

* In cart page get user details using auth context and cart details using use Cart.
* Cart page is divided in to two:
* First is used for displaying product details with image,price,and name.
* Then used handle increment and handle decrement for increasing the quantity.
* Displayed the user detail that is placing the order. suppose the user is guest first login then only allowed to checkout the details and payment.
* Second is used to display total amount and summary details and payment.
* Used total price function for calculating the total amount.
* Used remove cart item function for removing the id from the cart and removed details from local storage.
* Used map function for getting the product details from the cart.
* Then used update address before placing the order or the address is updated then allowed to place the order by paying amount using card and paypal.
* Before using payment used to get client token from Braintree-paypal.
* For payment used dropin for both card and paypal.
* While placing an order using card with details :

Card no:4242 4242 4242 4242

Valid thuru:12/25 pin:123

* Or use pay pal.
* While placing an order using paypal I used Braintree and linked paypal and Braintree with paypal id ,private and public key.
* After payment is successful used toast for displaying messages and navigate to order page.

**Index.js:**

* Used auth provider and cart provider in index file.
* Used provider to wrap the store from reducer.

**BACKEND:**

In backend

SERVER.js

1. Use express,cors and mongoose for mongodb as database
2. Create connection between mongoose with mongodb atlas and express and establish the connection.
3. Use cors and express as json .
4. Create .env file.
5. Then use uploads file for storing the images with express.static.
6. Then use user,category and product with user routes, category routes and product routes.

**.ENV:**

1. Localhost:5500,

Mongodb atlas link: mongodb+srv://apraveenamanikandan92:test123@ecommerce.iobyyfd.mongodb.net/ecommerce

1. Used secret key:secret
2. Linked Braintree Id,private and public key.

**SEED USER:**

* In seed user created 2 user 1.user
* 2.Admin

**SEED PRODUCTS:**

* In seed products created 10 product using product model .

**CONFIG:**

**Connect:**

* Establish a connection between mongoose with mongodb atlas url.

MIDDLEWARE:

**Protected Resources:**

1. Used jwt as jsonwebtoken for authorization and used JWT\_SECRET for encryption using bcrypt.
2. Created authorization in the headers and replace the token with bearer.
3. If not authorization user is not logged in.
4. Get the id from payload if the id matches then user goes to next.

**Is admin:**

* Check the user is admin using findById and check whether the user!==1 then it is unauthorized or else it will move to next.

**File Upload:**

* Used multer for storing images and allowed only jpg,jpeg and png file upto 5MB.
* Used storage and file destination with folder uploads.

**Models:**

* Category model

1. Created a schema for category with name type as string and used timestamps are current date.

* Product model

1. Created a schema for product with name, description,price,quantity,image,category used reference ,shipping,rating and reviews with timestamps.

* Order model

1. Created a schema for order get products from product model reference,payment,buyer used user model for reference and status with timestamps.

* User model

1. Created a schema for user like name,address,phone,email,password,isadmin set default to 0 and set timestamps.

ROUTE:

1. User routes
2. product routes
3. category routes
4. user routes:

* created routes for login and register as post with authorization and is admin from middleware.
* Created a user and admin as get.
* Created a profile for user as put for updating the details.
* Created an order and all orders as get for getting the order details.
* Get an order Id as put for updating the order status to user.

2. product routes:

* Created a path for downloading the images and file upload for getting images
* Created CRUD operations for product to add new product,get all product,get product by id ,delete product,update product ,update images,product filters,product list,product count,brain tree token and payment.

3. category routes:

* Created a route with get,post,put and delete the category routes.

Controllers:

1. user controller
2. product controller
3. category controller
4. USER CONTROLLER:

* Used bcrypt,jwt,secret key from .env file.
* Get user and order model for the details.

**Register:**

* Used findone for finding the email if the email matches the user is already registered.
* Used bcrypt password with 10 rounds for encrypting the password.
* Create a new for registration with user details.

**Login:**

* Used find one for getting the email and check the password using bcrypt .compare for compare the password.
* Then used jwt-sign for getting the token and check with secret key from env file.
* If it matches then get the user details.

**Profile:**

* For updating the user profile get the user details from the body and used findbyId to get the user id.
* Get the encrypted password and if both matches then update the user by using findby id and update is used for updating the user details.

**Order:**

* Get the user id as buyer from order model and populate the products,user name and used sort for listing the data in descending order using -1.

**All Order:**

* Get all the user order for admin using find for getting all the orders.

**Order Status:**

* After getting order placed using order id and status from body order status gets updated by admin.

GIT HUB LINK: https://github.com/Praveena-manis/Assignment7