**Classroom Activity: Develop a food delivery app using the MERN (MongoDB, Express.js, React.js, Node.js) stack**

**Step 1: Create a Project Structure**

|  |  |
| --- | --- |
| **Back-end** | **Front-end** |
|  |  |

**2. Setting Up the Development Environment**

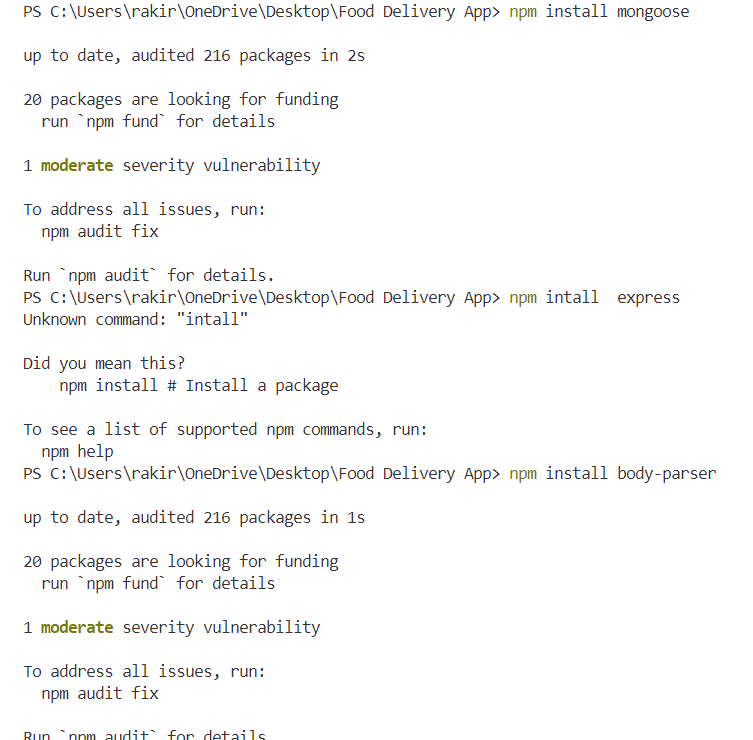
**(i) Initialize the project:**

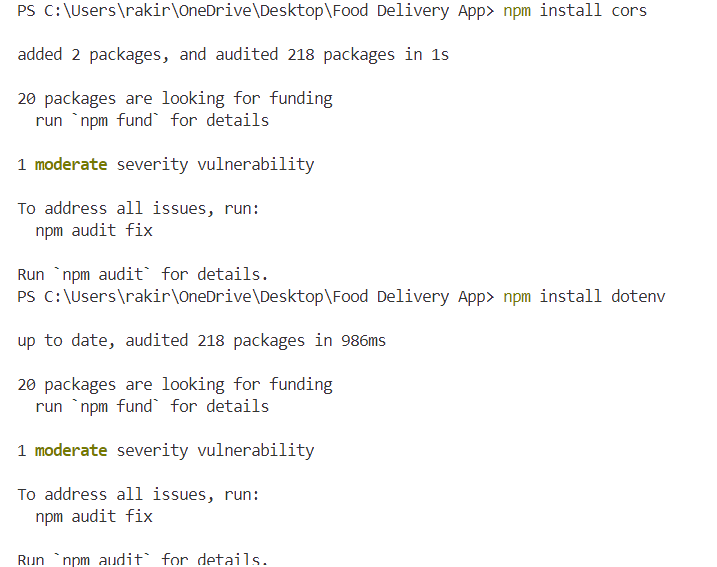
* Create the root directory food-delivery-app.
* Navigate to the directory and run npm init -y to initialize a Node.js project.

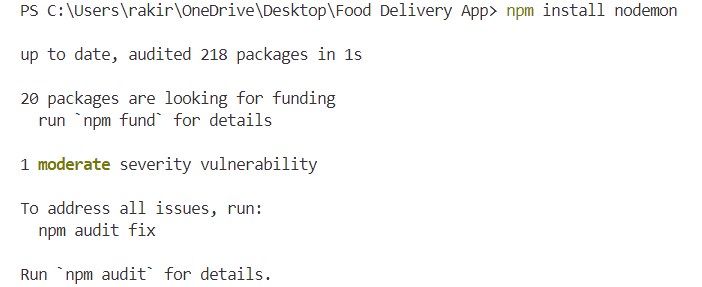


**(ii)Set up the backend**:

* In the backend/ folder, install the required dependencies:







* **create project directory structure at backend side​**



* **Create “.env” file to maintain security for database URL & PORT number and any other sensitive information**

|  |
| --- |
| PORT=5000  MONGO\_URI= "mongodb+srv://ajay4rjava:Ts07ka8375@cluster0.gdleahw.mongodb.net/food?retryWrites=true&w=majority&appName=Cluster0"  WhatisYourName= "My name is Ajay" |

* **Define Models:**

In the models/ folder, define your schemas for Firm, Product, and Vendor.

**Firm.js:**

|  |
| --- |
| const mongoose = require('mongoose');  const firmSchema = new mongoose.Schema({      firmName: {          type: String,          required: true,          unique: true      },      area: {          type: String,          required: true,      },      category: {          type: [{              type: String,              enum: ['veg', 'non-veg']          }]      },      region: {          type: [{              type: String,              enum: ['south-indian', 'north-indian', 'chinese', 'bakery']          }]      },      offer: {          type: String,      },      image: {          type: String      },      vendor: [{          type: mongoose.Schema.Types.ObjectId,          ref: 'Vendor'      }],      products: [{          type: mongoose.Schema.Types.ObjectId,          ref: 'Product'      }]  });  const Firm = mongoose.model('Firm', firmSchema);  module.exports = Firm |

**Product.js**

|  |
| --- |
| const mongoose = require('mongoose');  const productSchema = new mongoose.Schema({      productName: {          type: String,          required: true      },      price: {          type: String,          required: true      },      category: {          type: [{              type: String,              enum: ['veg', 'non-veg']          }]      },      image: {          type: String      },      bestSeller: {          type: Boolean      },      description: {          type: String      },      firm: [{          type: mongoose.Schema.Types.ObjectId,          ref: 'Firm'      }]  });  const Product = mongoose.model('Product', productSchema);  module.exports = Product |

**Vendor.js:**

|  |
| --- |
| const mongoose = require('mongoose');  const vendorSchema = new mongoose.Schema({      username: {          type: String,          required: true      },      email: {          type: String,          required: true,          unique: true      },      password: {          type: String,          required: true      },      firm: [{          type: mongoose.Schema.Types.ObjectId,          ref: 'Firm'      }]  });  const Vendor = mongoose.model('Vendor', vendorSchema);  module.exports = Vendor; |

* **Create Controllers:**

In the controllers/ folder, implement the logic for handling CRUD operations for firms, products, and vendors.

**firmController.js:**

|  |
| --- |
| const Firm = require('../models/Firm');  const Vendor = require('../models/Vendor');  const multer = require('multer');  const path = require('path');  const storage = multer.diskStorage({      destination: function(req, file, cb) {          cb(null, 'uploads/'); // Destination folder where the uploaded images will be stored      },      filename: function(req, file, cb) {          cb(null, Date.now() + path.extname(file.originalname)); // Generating a unique filename      }  });  const upload = multer({ storage: storage });  const addFirm = async(req, res) => {      try {          const { firmName, area, category, region, offer } = req.body;          const image = req.file ? req.file.filename : undefined;          const vendor = await Vendor.findById(req.vendorId);          if (!vendor) {              res.status(404).json({ message: "Vendor not found" })          }          if (vendor.firm.length > 0) {              return res.status(400).json({ message: "vendor can have only one firm" });          }          const firm = new Firm({              firmName,              area,              category,              region,              offer,              image,              vendor: vendor.\_id          })          const savedFirm = await firm.save();          const firmId = savedFirm.\_id          const vendorFirmName = savedFirm.firmName          vendor.firm.push(savedFirm)          await vendor.save()            return res.status(200).json({ message: 'Firm Added successfully ', firmId, vendorFirmName });      } catch (error) {          console.error(error)          res.status(500).json("intenal server error")      }  }  const deleteFirmById = async(req, res) => {      try {          const firmId = req.params.firmId;          const deletedProduct = await Firm.findByIdAndDelete(firmId);          if (!deletedProduct) {              return res.status(404).json({ error: "No product found" })          }      } catch (error) {          console.error(error);          res.status(500).json({ error: "Internal server error" })      }  }  module.exports = { addFirm: [upload.single('image'), addFirm], deleteFirmById } |

**productController.js:**

|  |
| --- |
| const Product = require("../models/Product");  const multer = require("multer");  const Firm = require('../models/Firm')  const path = require('path');  const storage = multer.diskStorage({      destination: function(req, file, cb) {          cb(null, 'uploads/'); // Destination folder where the uploaded images will be stored      },      filename: function(req, file, cb) {          cb(null, Date.now() + path.extname(file.originalname)); // Generating a unique filename      }  });  const upload = multer({ storage: storage });  const addProduct = async(req, res) => {      try {          const { productName, price, category, bestSeller, description } = req.body;          const image = req.file ? req.file.filename : undefined;          const firmId = req.params.firmId;          const firm = await Firm.findById(firmId);          if (!firm) {              return res.status(404).json({ error: "No firm found" });          }          const product = new Product({              productName,              price,              category,              bestSeller,              description,              image,              firm: firm.\_id          })          const savedProduct = await product.save();          firm.products.push(savedProduct);          await firm.save()          res.status(200).json(savedProduct)      } catch (error) {          console.error(error);          res.status(500).json({ error: "Internal server error" })      }  }  const getProductByFirm = async(req, res) => {      try {          const firmId = req.params.firmId;          const firm = await Firm.findById(firmId);          if (!firm) {              return res.status(404).json({ error: "No firm found" });          }          const restaurantName = firm.firmName;          const products = await Product.find({ firm: firmId });          res.status(200).json({ restaurantName, products });      } catch (error) {          console.error(error);          res.status(500).json({ error: "Internal server error" })      }  }  const deleteProductById = async(req, res) => {      try {          const productId = req.params.productId;          const deletedProduct = await Product.findByIdAndDelete(productId);          if (!deletedProduct) {              return res.status(404).json({ error: "No product found" })          }          res.status(200).json({ message: "Product deleted successfully" });      } catch (error) {          console.error(error);          res.status(500).json({ error: "Internal server error" })      }  }  module.exports = { addProduct: [upload.single('image'), addProduct], getProductByFirm, deleteProductById }; |

**vendorController.js:**

|  |
| --- |
| const Vendor = require('../models/Vendor');  const jwt = require('jsonwebtoken');  const bcrypt = require('bcryptjs');  const dotEnv = require('dotenv');  dotEnv.config();  const secretKey = process.env.WhatIsYourName    const vendorRegister = async(req, res) => {      const { username, email, password } = req.body;      try {          const vendorEmail = await Vendor.findOne({ email });          if (vendorEmail) {              return res.status(400).json("Email already taken");          }          const hashedPassword = await bcrypt.hash(password, 10);          const newVendor = new Vendor({              username,              email,              password: hashedPassword          });          await newVendor.save();          res.status(201).json({ message: "Vendor registered successfully" });          console.log('registered')      } catch (error) {          console.error(error);          res.status(500).json({ error: "Internal server error" })      }  }  const vendorLogin = async(req, res) => {      const { email, password } = req.body;      try {          const vendor = await Vendor.findOne({ email });          if (!vendor || !(await bcrypt.compare(password, vendor.password))) {              return res.status(401).json({ error: "Invalid username or password" })          }          const token = jwt.sign({ vendorId: vendor.\_id }, secretKey, { expiresIn: "1h" })          const vendorId = vendor.\_id;          res.status(200).json({ success: "Login successful", token, vendorId })          console.log(email, "this is token", token);      } catch (error) {          console.log(error);          res.status(500).json({ error: "Internal server error" });      }  }  const getAllVendors = async(req, res) => {      try {          const vendors = await Vendor.find().populate('firm');          res.json({ vendors })      } catch (error) {          console.log(error);          res.status(500).json({ error: "Internal server error" });      }  }  const getVendorById = async(req, res) => {      const vendorId = req.params.apple;      try {          const vendor = await Vendor.findById(vendorId).populate('firm');          if (!vendor) {              return res.status(404).json({ error: "Vendor not found" })          }          const vendorFirmId = vendor.firm[0].\_id;          res.status(200).json({ vendorId, vendorFirmId, vendor })          console.log(vendorFirmId);      } catch (error) {          console.log(error);          res.status(500).json({ error: "Internal server error" });      }  }  module.exports = { vendorRegister, vendorLogin, getAllVendors, getVendorById } |

* **Set Up Routes:**

Define the routes in the routes/ folder, mapping them to the respective controllers.

**firmRoutes.js:**

|  |
| --- |
| const express = require('express');  const firmController = require('../controllers/firmController');  const verifyToken = require('../middlewares/verifyToken');  const router = express.Router()  router.post('/add-firm', verifyToken, firmController.addFirm);  router.get('/uploads/:imageName', (req, res) => {      const imageName = req.params.imageName;      res.header('Content-Type', 'image/jpeg');      res.sendFile(path.join(\_\_dirname, '..', 'uploads', imageName));  });  router.delete('/:firmId', firmController.deleteFirmById);  module.exports = router; |

**productRoutes.js:**

|  |
| --- |
| const express = require('express');  const productController = require("../controllers/productController");  const router = express.Router();  router.post('/add-product/:firmId', productController.addProduct);  router.get('/:firmId/products', productController.getProductByFirm);  router.get('/uploads/:imageName', (req, res) => {      const imageName = req.params.imageName;      res.header('Content-Type', 'image/jpeg');      res.sendFile(path.join(\_\_dirname, '..', 'uploads', imageName));  });  router.delete('/:productId', productController.deleteProductById);  module.exports = router; |

**vendorRoutes.js:**

|  |
| --- |
| const vendorController = require('../controllers/vendorController');  const express = require('express');  const router = express.Router();  router.post('/register', vendorController.vendorRegister);  router.post('/login', vendorController.vendorLogin);  router.get('/all-vendors', vendorController.getAllVendors);  router.get('/single-vendor/:apple', vendorController.getVendorById)  module.exports = router; |

* **Middleware:**

Implement middleware in the middlewares/ folder, such as token verification (VerifyToken.js), to protect certain routes.

**VerifyToken.js:**

|  |
| --- |
| const Vendor = require('../models/Vendor');  const jwt = require('jsonwebtoken');  const dotEnv = require('dotenv');  dotEnv.config()  const secretKey = process.env.WhatIsYourName  const verifyToken = async(req, res, next) => {      const token = req.headers.token;      if (!token) {          return res.status(401).json({ error: "Token is required" });      }      try {          const decoded = jwt.verify(token, secretKey)          const vendor = await Vendor.findById(decoded.vendorId);          if (!vendor) {              return res.status(404).json({ error: "vendor not found" })          }          req.vendorId = vendor.\_id          next()      } catch (error) {          console.error(error)          return res.status(500).json({ error: "Invalid token" });      }  }  module.exports = verifyToken |

* **Start the Server:**

|  |
| --- |
| npm start |

This will start your backend server on a specified port (e.g., http://localhost:5000).

**(iii)Set up the frontend**:

* Create the frontend/ folder and set up a React application

|  |
| --- |
| C:\>CRUD\_MERN>npx create-react-app dashboard​  C:\>CRUD\_MERN>cd dashboard  C:\>CRUD\_MERN>npm start |

**dist/ Folder:**

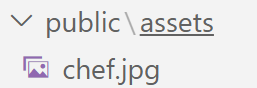
* Contains the production-ready build of your frontend application, including minified and optimized assets.
* The assets/ subfolder holds compiled images, CSS, and JavaScript files.

**node\_modules/ Folder**:

* Holds all the third-party libraries and dependencies installed via npm.

**public/ Folder**:

* This folder typically holds static assets like images or fonts that need to be accessible without any modifications.
* Sometimes contains a favicon.ico or other static files.



**src/ Folder**:

* **components/**: Houses reusable UI components such as buttons, input fields, headers, etc.

**AllProducts.jsx:**

|  |
| --- |
| import React,{useState, useEffect} from 'react'  import { API\_URL } from '../data/apiPath';  const AllProducts = () => {      const [products, setProducts]= useState([]);      const productsHandler = async()=>{              const firmId = localStorage.getItem('firmId');          try {                  const response = await fetch(`${API\_URL}/product/${firmId}/products`);                  const newProductsData = await response.json();                  setProducts(newProductsData.products);                  console.log(newProductsData);          } catch (error) {              console.error("failed to fetch products", error);              alert('failed to fetch products')          }      }      useEffect(()=>{          productsHandler()          console.log('this is useEffect')      },[])      const deleteProductById = async(productId)=>{                  try {                          const response = await fetch(`${API\_URL}/product/${productId}`,{                              method: 'DELETE'                          })                      if(response.ok){                          setProducts(products.filter(product =>product.\_id !== productId));                          confirm("are you sure, you want to delete?")                          alert("Product deleted Successfully")                      }                  } catch (error) {                      console.error('Failed to delete product');                      alert('Failed to delete product')                  }      }      return (      <div className='productSection'>          {!products ? (              <p>No products added</p>          ) : (              <table className="product-table">                  <thead>                      <tr>                          <th>Product Name</th>                          <th>Price</th>                          <th>Image</th>                          <th>Delete</th>                      </tr>                  </thead>                  <tbody>                      {products.map((item)=>{                              return (                                  <>                                      <tr key={item.\_id}>                                          <td>{item.productName}</td>                                          <td>₹{item.price}</td>                                      <td>                                          {item.image && (                                              <img src={`${API\_URL}/uploads/${item.image}`}                                              alt={item.productName}                                              style={{ width: '50px', height:'50px'  }}                                              />                                          )}                                      </td>                                      <td>                                          <button onClick={()=>deleteProductById(item.\_id)}                                          className='deleteBtn'                                          >Delete</button>                                      </td>                                      </tr>                                  </>                              )                      })}                  </tbody>              </table>           )}      </div>    )  }  export default AllProducts |

**NavBar.jsx:**

|  |
| --- |
| import React from 'react'  const NavBar = ({showLoginHandler, showRegisterHandler, showLogOut, logOutHandler}) => {      const firmName = localStorage.getItem('firmName')    return (      <div className="navSection">            <div className="company">              Vendor Dashboard          </div>          <div className="firmName">              <h4>Firname : {firmName}</h4>          </div>          <div className="userAuth">            {!showLogOut ?  <>             <span onClick={showLoginHandler}>Login / </span>            <span onClick={showRegisterHandler}>Register</span>            </> : <span onClick={logOutHandler}            className='logout'            >Logout</span>  }            </div>      </div>    )  }  export default NavBar |

**NotFound.jsx:**

|  |
| --- |
| import React from 'react'  import { Link } from 'react-router-dom'  const NotFound = () => {    return (      <>      <div className='errorSection'>      <Link to="/" style={{ fontSize:'1.5rem', color: 'darkblue'}}>          <p>go back</p>      </Link>          <h1>404</h1>          <div>Page Not found</div>      </div>      </>      )  }  export default NotFound |

**SideBar.jsx:**

|  |
| --- |
| import React from "react";  const SideBar = ({    showFirmHandler,    showProductHandler,    showAllProductsHandler,    showFirmTitle  }) => {    return (      <div className="sideBarSection">        <ul>          {showFirmTitle ? <li onClick={showFirmHandler}>Add Firm</li> : "" }          <li onClick={showProductHandler}>Add Product</li>          <li onClick={showAllProductsHandler}>All Products</li>          <li>User Details</li>        </ul>      </div>    );  };  export default SideBar; |

**Welcome.jsx:**

|  |
| --- |
| import React from 'react'  const Welcome = () => {      const firmName = localStorage.getItem("firmName")    return (      <div className='welcomeSection'>          <h2>Welcome {firmName}</h2>          <div className="landingImage">            <img src='/assets/chef.jpg' alt='welcome' />          </div>      </div>    )  }  export default Welcome |

**Components/forms:**

**AddFirm.jsx:**

|  |
| --- |
| import React, {useState} from 'react'  import { API\_URL } from '../../data/apiPath';  import { ThreeCircles } from 'react-loader-spinner';  const AddFirm = () => {    const [firmName, setFirmName] = useState("");    const [area, setArea] = useState("");    const [category, setCategory] = useState([]);    const [region, setRegion] = useState([]);    const [offer, setOffer] = useState("");    const [file, setFile] = useState(null);    const [loading, setLoading] = useState(false);    const handleCategoryChange = (event)=>{        const value = event.target.value;          if(category.includes(value)){            setCategory(category.filter((item)=> item !== value));          }else{            setCategory([...category, value])          }    }    const handleRegionChange = (event)=>{        const value = event.target.value;          if(region.includes(value)){            setRegion(region.filter((item)=> item !== value));          }else{            setRegion([...region, value])          }    }      const handleImageUpload =(event)=>{        const selectedImage = event.target.files[0];        setFile(selectedImage)    }    const handleFirmSubmit= async(e)=>{          e.preventDefault();      setLoading(true);     try {          const loginToken = localStorage.getItem('loginToken');          if(!loginToken){              console.error("User not authenticated");          }          const formData = new FormData();            formData.append('firmName', firmName);            formData.append('area', area);            formData.append('offer', offer);            formData.append('image', file)            category.forEach((value)=>{              formData.append('category', value)            });            region.forEach((value)=>{              formData.append('region', value)            })            const response = await fetch(`${API\_URL}/firm/add-firm`,{              method:'POST',              headers:{                'token': `${loginToken}`              },              body: formData            });            const data = await response.json()            if(response.ok){              console.log(data);              setFirmName("");              setArea("")              setCategory([]);              setRegion([]);              setOffer("");              setFile(null)              alert("Firm added Successfully")            }else if(data.message === "vendor can have only one firm"){                alert("Firm Exists 🥗. Only 1 firm can be added  ")            } else{                alert('Failed to add Firm')            }                 const mango = data.firmId;            const vendorRestuarant = data.vendorFirmName            localStorage.setItem('firmId', mango);            localStorage.setItem('firmName', vendorRestuarant)            window.location.reload()     } catch (error) {        console.error("failed to add Firm")        alert("failed to add Firm")     } finally {      setLoading(false);    }    }    return (          <div className="firmSection">     {loading &&        <div className="loaderSection">          <ThreeCircles            visible={loading}            height={100}            width={100}            color="#4fa94d"            ariaLabel="three-circles-loading"            wrapperStyle={{}}            wrapperClass=""          />        </div>}           {!loading &&   <form className="tableForm" onSubmit={handleFirmSubmit}>              <h3>Add Firm</h3>                  <label >Firm Name</label>                  <input type="text" name='firmName' value={firmName} onChange={(e)=>setFirmName(e.target.value)}/>                  <label >Area</label>                  <input type="text"  name='area' value={area} onChange={(e)=>setArea(e.target.value)} />                  {/\* <label >Category</label>                  <input type="text"  /> \*/}      <div className="checkInp">        <label >Category</label>            <div className="inputsContainer">            <div className="checboxContainer">                    <label>Veg</label>                    <input type="checkbox" checked ={category.includes('veg')}  value="veg" onChange={handleCategoryChange}/>                  </div>                  <div className="checboxContainer">                    <label>Non-Veg</label>                    <input type="checkbox" checked ={category.includes('non-veg')} value="non-veg" onChange={handleCategoryChange}/>                  </div>            </div>      </div>      <label >Offer</label>                  <input type="text" name='offer' value={offer} onChange={(e)=>setOffer(e.target.value)}/>      <div className="checkInp">        <label >Region</label>            <div className="inputsContainer">            <div className="regBoxContainer">                    <label>South Indian</label>                    <input type="checkbox" value="south-indian"   checked ={region.includes('south-indian')}                    onChange={handleRegionChange}                    />                  </div>                  <div className="regBoxContainer">                    <label>North-Indian</label>                    <input type="checkbox" value="north-indian"  checked ={region.includes('north-indian')}                    onChange={handleRegionChange}                    />                  </div>                  <div className="regBoxContainer">                    <label>Chinese</label>                    <input type="checkbox" value="chinese" checked ={region.includes('chinese')}                    onChange={handleRegionChange}                    />                  </div>                  <div className="regBoxContainer">                    <label>Bakery</label>                    <input type="checkbox" value="bakery" checked ={region.includes('bakery')}                    onChange={handleRegionChange}                    />                  </div>            </div>      </div>                    <label >Firm Image</label>                  <input type="file" onChange={handleImageUpload} />                  <br />              <div className="btnSubmit">          <button type='submit'>Submit</button>      </div>             </form>}          </div>    )  }  export default AddFirm |

**AddProduct.jsx:**

|  |
| --- |
| import React, {useState} from 'react'  import { API\_URL } from '../../data/apiPath';  import { ThreeCircles } from 'react-loader-spinner';  const AddProduct = () => {      const [productName, setProductName] = useState("");      const [price, setPrice] = useState("");      const [category, setCategory] = useState([]);      const [bestSeller, setBestSeller] = useState(false);      const [image, setImage] = useState(null);      const [description, setDescription] = useState("");    const [loading, setLoading] = useState(false);      const handleCategoryChange = (event)=>{        const value = event.target.value;          if(category.includes(value)){            setCategory(category.filter((item)=> item !== value));          }else{            setCategory([...category, value])          }    }    const handleBestSeller =(event)=>{      const value = event.target.value === 'true'        setBestSeller(value)    }    const handleImageUpload =(event)=>{      const selectedImage = event.target.files[0];      setImage(selectedImage)  }    const handleAddProduct = async(e)=>{        e.preventDefault()      setLoading(true);        try {          const loginToken = localStorage.getItem('loginToken');            const firmId = localStorage.getItem('firmId')            if(!loginToken || !firmId){                console.error("user not authenticated")            }            const formData = new FormData();          formData.append('productName', productName);          formData.append('price', price);          formData.append('description', description);          formData.append('bestSeller', bestSeller)          formData.append('image', image)          category.forEach((value)=>{            formData.append('category', value)          });              const response = await fetch(`${API\_URL}/product/add-product/${firmId}`, {              method:'POST',              body: formData            })              const data = await response.json()              if(response.ok){                alert('Product added succesfully')              }              setProductName("")              setPrice("");              setCategory([])              setBestSeller(false);              setImage(null);              setDescription("")        } catch (error) {            alert('Failed to add Product')        }finally {          setLoading(false);        }    }      return (      <div className="firmSection">  {loading &&         <div className="loaderSection">          <ThreeCircles            visible={loading}            height={100}            width={100}            color="#4fa94d"            ariaLabel="three-circles-loading"            wrapperStyle={{}}            wrapperClass=""          />          <p>Please wait, your product is being added...</p>        </div>}    {!loading &&      <form className="tableForm" onSubmit={handleAddProduct}>      <h3>Add Product</h3>          <label >Product Name</label>          <input type="text" value={productName} onChange={(e)=>setProductName(e.target.value)} />          <label >Price</label>          <input type="text" value={price} onChange={(e)=>setPrice(e.target.value)}/>          <div className="checkInp">       <label >Category</label>           <div className="inputsContainer">           <div className="checboxContainer">                   <label>Veg</label>                   <input type="checkbox" value="veg" checked ={category.includes('veg')}  onChange={handleCategoryChange}/>                 </div>                 <div className="checboxContainer">                   <label>Non-Veg</label>                   <input type="checkbox" value="non-veg" checked ={category.includes('non-veg')} onChange={handleCategoryChange} />                 </div>           </div>     </div>     <div className="checkInp">       <label >Best Seller</label>           <div className="inputsContainer">           <div className="checboxContainer">                   <label>Yes</label>                   <input type="radio" value="true" checked = {bestSeller=== true} onChange={handleBestSeller}/>                 </div>                 <div className="checboxContainer">                   <label>No</label>                   <input type="radio" value="false" checked = {bestSeller=== false} onChange={handleBestSeller}/>                 </div>           </div>     </div>            <label >Description</label>          <input type="text" value={description} onChange={(e)=>setDescription(e.target.value)} />          <label >Firm Image</label>          <input type="file" onChange={handleImageUpload} />          <br />      <div className="btnSubmit">  <button type='submit'>Submit</button>  </div>     </form>    }   </div>    )  }  export default AddProduct |

**Login.jsx:**

|  |
| --- |
| import React, {useState} from 'react'  import { API\_URL } from '../../data/apiPath';  import { ThreeCircles } from 'react-loader-spinner';  const Login = ({showWelcomeHandler}) => {    const [email, setEmail] = useState("");    const [password, setPassword] = useState("");    const [loading, setLoading] = useState(false);    const [showPassword, setShowPassword] = useState(false)    const handleShowPassword = ()=>{      setShowPassword(!showPassword);    }      const loginHandler = async(e)=>{        e.preventDefault();      setLoading(true);        try {            const response = await fetch(`${API\_URL}/vendor/login`, {              method: 'POST',              headers:{                'Content-Type': 'application/json'              },              body: JSON.stringify({email, password})            })            const data = await response.json();            if(response.ok){              alert('Login success');              setEmail("");              setPassword("");              localStorage.setItem('loginToken', data.token);              showWelcomeHandler()            }            const vendorId = data.vendorId            console.log("checking for VendorId:",vendorId)            const vendorResponse = await fetch(`${API\_URL}/vendor/single-vendor/${vendorId}`)            window.location.reload()            const vendorData = await vendorResponse.json();            if(vendorResponse.ok){              const vendorFirmId = vendorData.vendorFirmId;              const vendorFirmName = vendorData.vendor.firm[0].firmName;              localStorage.setItem('firmId', vendorFirmId);              localStorage.setItem('firmName', vendorFirmName)            }        } catch (error) {            alert("login fail")        } finally {          setLoading(false);        }    }    return (      <div className="loginSection">  {loading &&        <div className="loaderSection">          <ThreeCircles            visible={loading}            height={100}            width={100}            color="#4fa94d"            ariaLabel="three-circles-loading"            wrapperStyle={{}}            wrapperClass=""          />          <p>Login in process... Please wait</p>        </div>}       {!loading &&    <form  className='authForm' onSubmit={loginHandler} autoComplete='off'>          <h3>Vendor Login</h3>              <label>Email</label>              <input type="text" name='email' value = {email} onChange={(e)=>setEmail(e.target.value)} placeholder='enter your email'/><br />              <label>Password</label>              <input   type={showPassword? "text":"password"} name='password' value={password} onChange={(e)=>setPassword(e.target.value)} placeholder='enter your password'/><br />              <span className='showPassword'                onClick={handleShowPassword}                >{showPassword ? 'Hide' : 'Show'}</span>      <div className="btnSubmit">          <button type= 'submit'>Submit</button>      </div>          </form>}      </div>    )  }  export default Login |

**Register.jsx:**

|  |
| --- |
| import React, { useState } from 'react';  import { API\_URL } from '../../data/apiPath';  import { ThreeCircles } from 'react-loader-spinner';  const Register = ({ showLoginHandler }) => {    const [username, setUsername] = useState("");    const [email, setEmail] = useState("");    const [password, setPassword] = useState("");    const [error, setError] = useState("");    const [loading, setLoading] = useState(false); // Set loading to false initially    const [showPassword, setShowPassword] = useState(false);    const handleShowPassword = () => {      setShowPassword(!showPassword);    };    const handleSubmit = async (e) => {      e.preventDefault();      setLoading(true); // Set loading to true when the request starts      try {        const response = await fetch(`${API\_URL}/vendor/register`, {          method: 'POST',          headers: {            'Content-Type': 'application/json'          },          body: JSON.stringify({ username, email, password })        });        const data = await response.json();        if (response.ok) {          console.log(data);          setUsername("");          setEmail("");          setPassword("");          alert("Vendor registered successfully");          showLoginHandler();        } else {          setError(data.error);          alert("Registration Failed, Contact Admin")        }      } catch (error) {        console.error("Registration failed", error);        alert("Registration failed");      } finally {        setLoading(false);      }    };    return (      <div className="registerSection">       {loading &&        <div className="loaderSection">        <ThreeCircles          visible={loading}          height={100}          width={100}          color="#4fa94d"          ariaLabel="three-circles-loading"          wrapperStyle={{}}          wrapperClass=""        />        <p>Hi, Your Registration under process</p>      </div>       }  {!loading &&     <form className='authForm' onSubmit={handleSubmit} autoComplete='off'>  <h3>Vendor Register</h3>  <label>Username</label>  <input type="text" name='username' value={username} onChange={(e) => setUsername(e.target.value)} placeholder='enter your name' /><br />  <label>Email</label>  <input type="text" name='email' value={email} onChange={(e) => setEmail(e.target.value)} placeholder='enter your email' /><br />  <label>Password</label>  <input type={showPassword ? "text" : "password"} value={password} onChange={(e) => setPassword(e.target.value)} name='password' placeholder='enter your password' /><br />  <span className='showPassword'    onClick={handleShowPassword}  >{showPassword ? 'Hide' : 'Show'}</span>  <div className="btnSubmit">    <button type='submit'>Submit</button>  </div>  </form>}        </div>    );  };  export default Register; |

* Handles API interactions, making it easier to communicate with the backend.

**Components/data:**

**apiPath.jsx:**

|  |
| --- |
| //export const API\_URL= "http://localhost:5000";  export const API\_URL= "https://vercel.com/ajay4ugits-projects/food/D7WWQy9tN2PRCdmuF6sqruiCWj7Q"; |

* **pages/**: Each page corresponds to a route in the app, like the homepage, vendor management page, etc.

|  |
| --- |
| import React, {useState, useEffect} from 'react'  import NavBar from '../components/NavBar'  import SideBar from '../components/SideBar'  import Login from '../components/forms/Login'  import Register from '../components/forms/Register'  import AddFirm from '../components/forms/AddFirm'  import AddProduct from '../components/forms/AddProduct'  import Welcome from '../components/Welcome'  import AllProducts from '../components/AllProducts'  const LandingPage = () => {    const [showLogin, setShowLogin] = useState(false)    const [showRegister, setShowRegister] = useState(false)    const [showFirm, setShowFirm] = useState(false)    const [showProduct, setShowProduct] = useState(false)    const [showWelcome, setShowWelcome] = useState(false)    const [showAllProducts, setShowAllProducts] = useState(false);    const [showLogOut, setShowLogOut] = useState(false)    const [showFirmTitle, setShowFirmTitle] = useState(true)    useEffect(()=>{      const loginToken = localStorage.getItem('loginToken');      if(loginToken){          setShowLogOut(true)          setShowWelcome(true)      }    }, [])    useEffect(()=>{        const firmName = localStorage.getItem('firmName');        const firmId = localStorage.getItem('firmId')        if(firmName || firmId ){            setShowFirmTitle(false)            setShowWelcome(true)        }    },[])      const logOutHandler =()=>{      confirm("Are you sure to logout?")        localStorage.removeItem("loginToken");        localStorage.removeItem("firmId");        localStorage.removeItem('firmName');        setShowLogOut(false)        setShowFirmTitle(true)        setShowWelcome(false)    }  const showLoginHandler =()=>{      setShowLogin(true)      setShowRegister(false)      setShowFirm(false)      setShowProduct(false)      setShowWelcome(false)      setShowAllProducts(false)  }  const showRegisterHandler = ()=>{      setShowRegister(true)      setShowLogin(false)      setShowFirm(false)      setShowProduct(false)      setShowWelcome(false)      setShowAllProducts(false)  }  const showFirmHandler = ()=>{    if(showLogOut){      setShowRegister(false)      setShowLogin(false)      setShowFirm(true)      setShowProduct(false)      setShowWelcome(false)      setShowAllProducts(false)    }else{      alert("please login");      setShowLogin(true)    }  }  const showProductHandler = ()=>{    if(showLogOut){      setShowRegister(false)      setShowLogin(false)      setShowFirm(false)      setShowProduct(true)      setShowWelcome(false)      setShowAllProducts(false)      }else{          alert("please login")          setShowLogin(true)      }  }  const showWelcomeHandler = ()=>{      setShowRegister(false)      setShowLogin(false)      setShowFirm(false)      setShowProduct(false)      setShowWelcome(true)      setShowAllProducts(false)  }  const showAllProductsHandler = ()=>{    if(showLogOut){      setShowRegister(false)      setShowLogin(false)      setShowFirm(false)      setShowProduct(false)      setShowWelcome(false)      setShowAllProducts(true)  }else{      alert("please login")      setShowLogin(true)   }  }    return (      <>          <section className='landingSection'>              <NavBar showLoginHandler = {showLoginHandler} showRegisterHandler = {showRegisterHandler}              showLogOut = {showLogOut}              logOutHandler = {logOutHandler}              />              <div className="collectionSection">              <SideBar showFirmHandler={showFirmHandler} showProductHandler ={showProductHandler}              showAllProductsHandler = {showAllProductsHandler}              showFirmTitle={showFirmTitle}              />            {showFirm && showLogOut && <AddFirm />}            {showProduct && showLogOut && <AddProduct />}            {showWelcome && <Welcome />}            {showAllProducts && showLogOut && <AllProducts />}            {showLogin && <Login showWelcomeHandler ={showWelcomeHandler}/>}            {showRegister && <Register showLoginHandler = {showLoginHandler}/>}                </div>          </section>      </>    )  }  export default LandingPage |

* **App.jsx**: Acts as the root component that brings together all other components and sets up the routing.

**App.jsx:**

|  |
| --- |
| import React from 'react'  import LandingPage from './vendorDashBoard/pages/LandingPage'  import { Routes, Route } from 'react-router-dom'  import "./App.css"  import NotFound from './vendorDashBoard/components/NotFound'  const App = () => {    return (      <div>        <Routes>            <Route path='/' element = {<LandingPage />}/>            <Route path='/\*' element = {<NotFound />} />        </Routes>      </div>    )  }  export default App |

* **main.jsx**: The entry point for the React application, where the root component (App.jsx) is rendered into the DOM.

|  |
| --- |
| import React from 'react'  import ReactDOM from 'react-dom/client'  import App from './App.jsx'  import {BrowserRouter} from 'react-router-dom'  import './index.css'  ReactDOM.createRoot(document.getElementById('root')).render(    <React.StrictMode>      <BrowserRouter>      <App />      </BrowserRouter>    </React.StrictMode>,  ) |

* **.gitignore**:

Lists files and directories that should not be tracked by Git, such as node\_modules, .env files, and build outputs.

|  |
| --- |
| # Logs  logs  \*.log  npm-debug.log\*  yarn-debug.log\*  yarn-error.log\*  pnpm-debug.log\*  lerna-debug.log\*  node\_modules  dist  dist-ssr  \*.local  # Editor directories and files  .vscode/\*  !.vscode/extensions.json  .idea  .DS\_Store  \*.suo  \*.ntvs\*  \*.njsproj  \*.sln  \*.sw? |

* **index.html**:

The main HTML file where your React app is mounted. This file is usually in the public folder and is the template used by the index.js to inject the React components.

|  |
| --- |
| <!doctype html>  <html lang="en">    <head>      <meta charset="UTF-8" />      <link rel="icon" type="image/svg+xml" href="/vite.svg" />      <meta name="viewport" content="width=device-width, initial-scale=1.0" />      <title>Vite + React</title>    </head>    <body>      <div id="root"></div>      <script type="module" src="/src/main.jsx"></script>    </body>  </html> |

* **package.json**:

Describes the project’s dependencies, scripts, and metadata. It is essential for managing project configurations and dependencies.

|  |
| --- |
| {    "name": "dashboard",    "private": true,    "version": "0.0.0",    "type": "module",    "scripts": {      "dev": "vite",      "build": "vite build",      "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",      "preview": "vite preview"    },    "dependencies": {      "dashboard": "file:",      "gh-pages": "^6.1.1",      "mailgun-js": "^0.22.0",      "react": "^18.3.1",      "react-devtools": "^5.2.0",      "react-dom": "^18.3.1",      "react-loader-spinner": "^6.1.6",      "react-router-dom": "^6.23.1"    },    "devDependencies": {      "@types/react": "^18.3.3",      "@types/react-dom": "^18.3.0",      "@vitejs/plugin-react": "^4.3.1",      "eslint": "^8.57.0",      "eslint-plugin-react": "^7.34.2",      "eslint-plugin-react-hooks": "^4.6.2",      "eslint-plugin-react-refresh": "^0.4.7",      "vite": "^5.3.1"    },    "description": "This template provides a minimal setup to get React working in Vite with HMR and some ESLint rules.",    "main": "vite.config.js",    "keywords": [],    "author": "",    "license": "ISC"  } |

* **package-lock.json**:

Automatically generated to maintain consistency across different environments when installing dependencies.