





COURSE NAME: MERNSTACK BY MONGODB

NAANMUDHALVAN ID : BF575A44AA4503DB97C4E820C47F248F

PROJECT TITLE:[ GROCERY WEBAPP ]

PROJECT SUBMITTED TO: SMART INTERNZ

NAME : K.ASMATHDILLFAR

DEPARTMENT: COMPUTER SCIENCE & ENGINEERING

YEAR: 2024-25

SEMESTER: VII

COLLEGE NAME : RRASE COLLEGE OF ENGINEERING

COLLEGE CODE : 4118

**INTRODUCTION**

The rise of e-commerce and online services has transformed the way people shop for groceries. A **Grocery Web Application** provides a convenient, efficient, and user-friendly platform for users to shop for groceries from the comfort of their homes. Our project focuses on building such an application, leveraging the **MERN stack (MongoDB, Express.js, React.js, and Node.js)** to deliver a seamless experience.

**MERN STACK MongoDB**

**MERN stack MongoDB** refers to the use of MongoDB as the database component in a MERN stack application. Let's break this down:

**MERN Stack:**

**MERN** is a popular JavaScript-based technology stack used to build full-stack web applications. It includes:

1. **M: MongoDB** - A NoSQL database.
2. **E: Express.js** - A backend framework for Node.js.
3. **R: React.js** - A library for building the frontend.
4. **N: Node.js** - A JavaScript runtime environment for server-side development.

**What is MongoDB?**

**** MongoDB is a NoSQL database, meaning it stores data in a flexible, JSON-like format instead of traditional tables and rows like SQL databases.

 Data is organized into documents (like JSON objects) and stored in collections (similar to tables in relational databases).

 It is schema-less, meaning you don't need to define a rigid structure for the data, making it highly flexible for modern web apps.

**Why Use MongoDB in MERN?**

1. **Flexible Data Model**:
   * MongoDB's document-based structure aligns well with the JSON format used in React.js and Node.js. Data flows smoothly between the frontend and backend.
2. **Scalability**:
   * MongoDB is highly scalable, making it ideal for apps that may grow in terms of data and users.
3. **Easy Integration**:
   * MongoDB integrates seamlessly with Node.js through libraries like **Mongoose**, which simplifies database interactions.
4. **Real-Time Performance**:
   * MongoDB is fast and efficient for handling large amounts of unstructured or semi-structured data.

**MongoDB in a MERN Project**

* In a MERN stack grocery web app, MongoDB could be used to store:
  + **User information** (name, email, address, etc.)
  + **Product details** (name, price, category, stock, etc.)
  + **Orders** (user purchases, order history, etc.)
  + **Cart items** (temporary data for what a user wants to buy)

By using MongoDB, the application can efficiently handle diverse and dynamic data, providing a smooth and responsive user experience.

CODING 1.App.js

import React from 'react';

import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';

import GroceryList from './components/GroceryList';

const App = () => {

  return (

    <div>

      <h1>Grocery List</h1>

      <GroceryList />

    </div>

  );

}

const App = () => (

  <Router>

    <Routes>

      <Route path="/" element={<GroceryList />} />

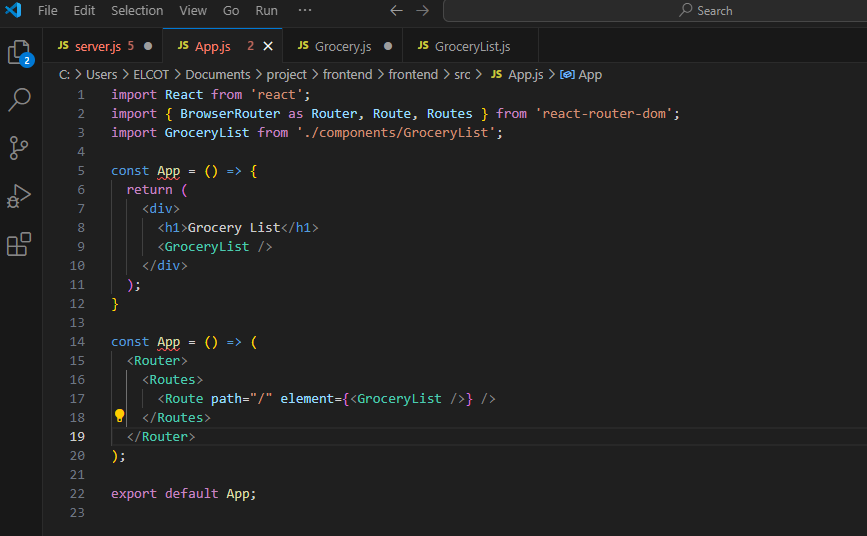
    </Routes>

  </Router>

);

export default App;

ScreenShot 1. App.js :



CODING 2.Server.js

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const bodyParser = require('body-parser');

require('dotenv').config();

const app = express();

const PORT = process.env.PORT || 5000;

app.use(cors());

app.use(bodyParser.json());

// MongoDB Connection

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

  .then(() => console.log('MongoDB Connected'))

  .catch((err) => console.log(err));

// Routes

app.get('/', (req, res) => res.send('API is running'));

// Start Server

app.listen(PORT, () => console.log(Server running on port ${PORT}));

const Grocery = require('./models/Grocery');

app.get('/groceries', async (req, res) => {

  const groceries = await Grocery.find();

  res.json(groceries);

});

app.post('/groceries', async (req, res) => {

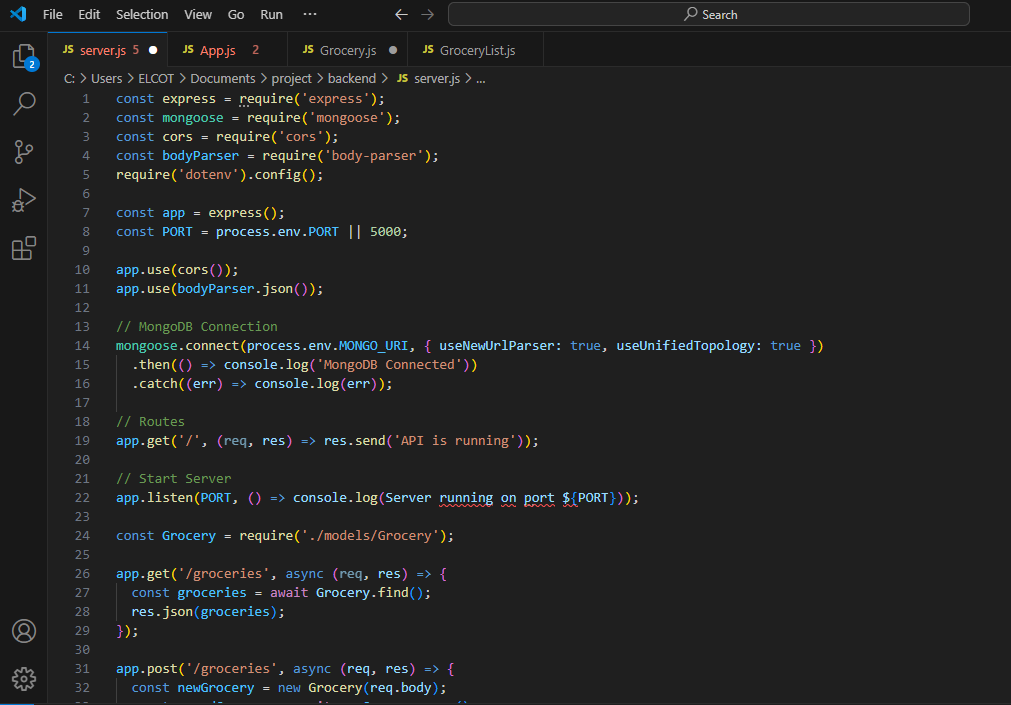
  const newGrocery = new Grocery(req.body);

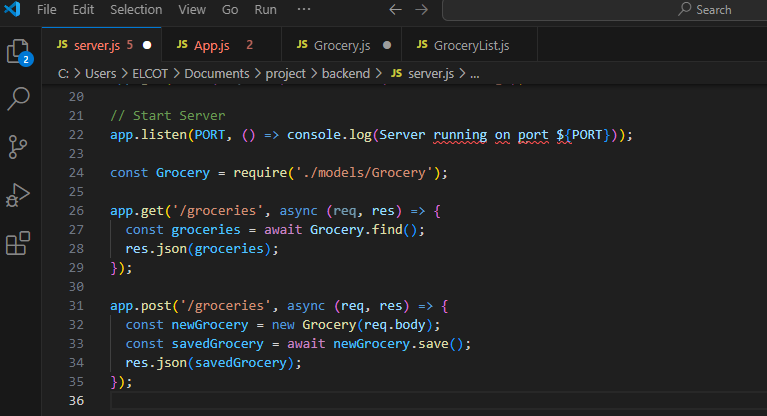
  const savedGrocery = await newGrocery.save();

  res.json(savedGrocery);

});

ScreenShot 2. Server.js :





CODING 3.Grocery.js

const mongoose = require('mongoose');

const GrocerySchema = new mongoose.Schema({

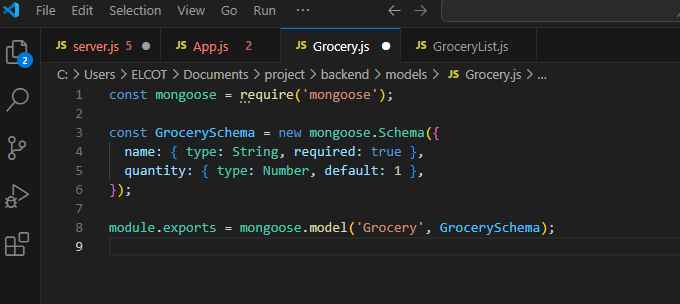
  name: { type: String, required: true },

  quantity: { type: Number, default: 1 },

});

module.exports = mongoose.model('Grocery', GrocerySchema);

ScreenShot 3. Grocery.js :



CODING 4.GroceryList.js

import React, { useState, useEffect } from 'react';

import axios from 'axios';

const GroceryList = () => {

  const [groceries, setGroceries] = useState([]);

  useEffect(() => {

    axios.get('http://localhost:5000/groceries')

      .then((response) => setGroceries(response.data))

      .catch((error) => console.error(error));

  }, []);

  return (

    <div>

      <h1>Grocery List</h1>

      <ul>

        {groceries.map((item) => (

          <li key={item.\_id}>{item.name}</li>

        ))}

      </ul>

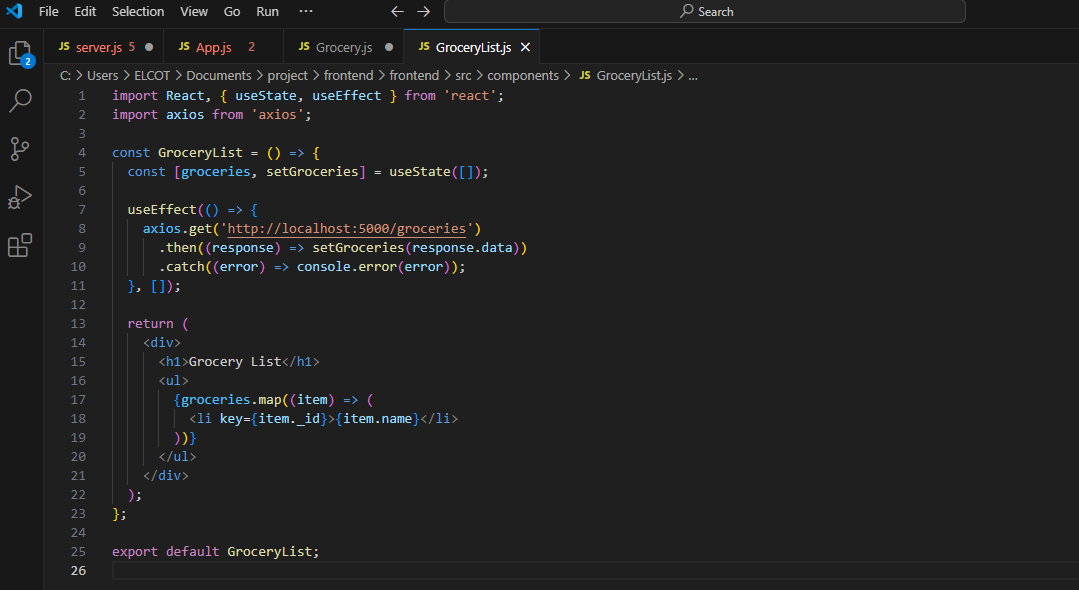
    </div>

  );

};

export default GroceryList;

ScreenShot 4.GroceryList.js :



**CONCLUTION**

The development of a **Grocery Web Application** using the MERN stack demonstrates the power of modern web technologies in creating efficient, scalable, and user-friendly solutions. By leveraging **MongoDB** for data management, **Express.js** and **Node.js** for backend services, and **React.js** for a responsive and interactive frontend, the application ensures a seamless shopping experience.