Maharaja's Technological Institute,

Thrissur – 680020

REPORT

5009:Internship-II

Submitted by :

LIJO P THOMAS

Register no. 2301131789

Department of computer Engineering 2025-26

College Vision and Misson

Vision

To be a premier institute in the field of engineering to serve society by moulding technically competent professionals.

Mission

Providing value-based quality education through state of the art technologies and eco- friendly environments.

Ensuring a continuously improving teaching learning process to meet the current trends in the industry.

Involving development of society through inculcating leadership qualities and entrepreneurial skills among students.

Department Vision and Misson

Vision

• To be a centre of excellence by producing technically skilled, creative, independent and socially inclined diploma computer engineers contributing towards the ever changing needs of society.

Misson

• To impart high quality professional training with an emphasis in basic principles of computer engineering.

• Inculcate professional and ethical values among students.

• Imbibe social awareness and responsibility in students to serve the society and protect environment.

• Provide opportunities to promote organizational and leadership skills in students.

COMPUTER ENGINEERING

CERTIFICATE

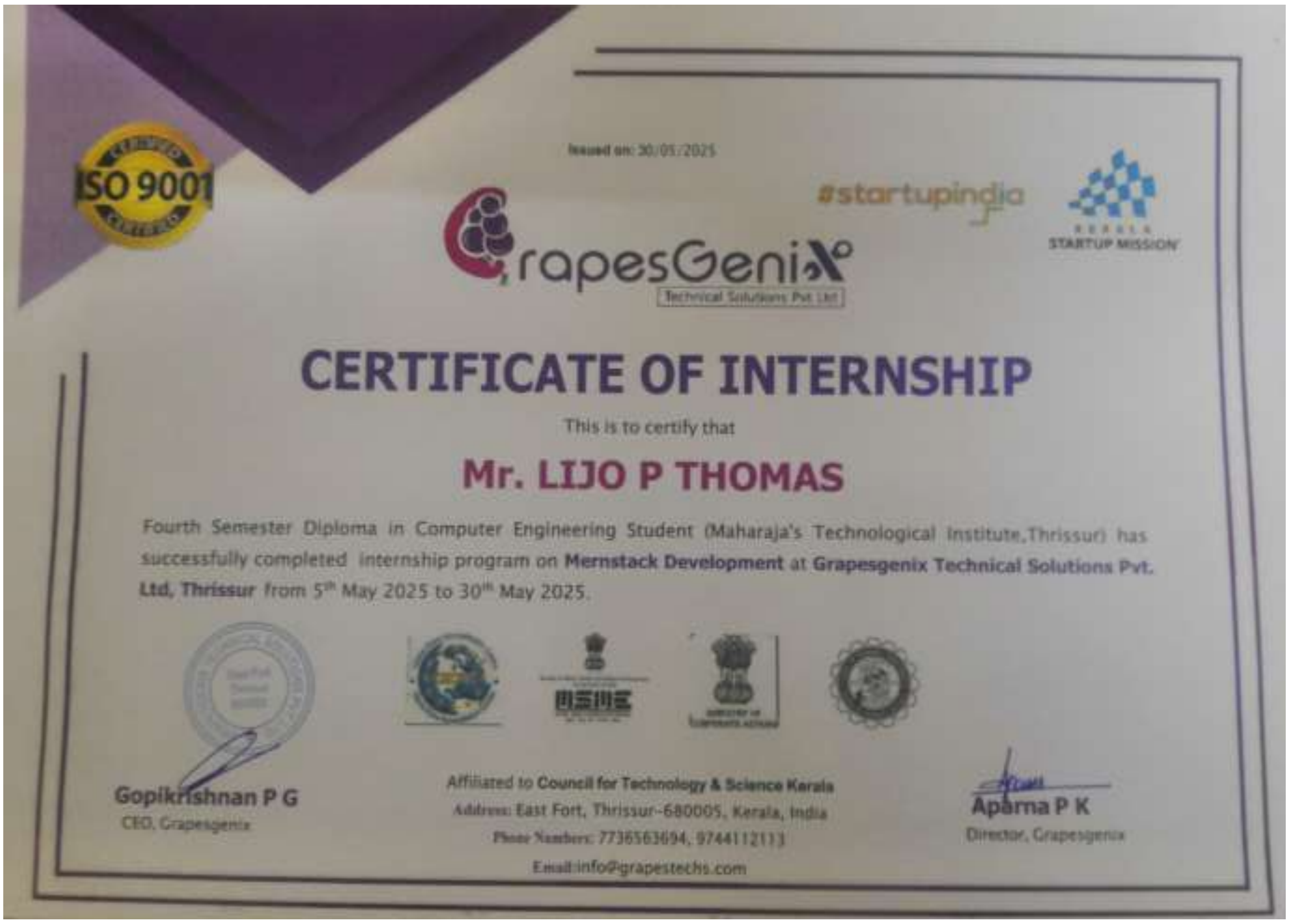
This is to certify that the report entitled **“5009: Internship II Report on E-Commerce Website using MERN Stack”**was prepared by **LIJO P THOMAS** (Reg. No: **2301131789**), a student of the **Fourth Semester Diploma in Computer Engineering**, in partial fulfillment of the academic requirements for the subject **Internship – II** (Subject Code: **5009**).

The report is based on the internship carried out at **GrapesGenix Technical Solutions Pvt. Ltd., Thrissur**, and presents the student’s work and learning experiences in developing a full-stack **E-Commerce website** using modern web technologies, namely **MongoDB, Express.js, React.js, and Node.js (MERN Stack)**.

This certificate is issued as part of the academic submission under the curriculum prescribed by the Department of Technical Education, Kerala.

Faculty-in-charge *Head of Department*

Internal Examiner *External Examiner*



Issued on: 30/05/2025

#startupindia

KERALA STARTUP MISSION

GrapesGeniX
Technical Solutions Pvt Ltd

CERTIFICATE OF INTERNSHIP

This is to certify that

Mr. LIJO P THOMAS

Fourth Semester Diploma in Computer Engineering Student (Maharaja's Technological Institute,Thrissur) has successfully completed internship program on **Mernstack Development** at **Grapesgenix Technical Solutions Pvt. Ltd, Thrissur** from 5th May 2025 to 30th May 2025.

Gopikrishnan P G
CEO, Grapesgenix

Affiliated to **Council for Technology & Science Kerala**
Address: East Fort, Thrissur-680005, Kerala, India
Phone Numbers: 7736563694, 9744112113
Email:info@grapestechs.com

Aparna P K
Director, Grapesgenix

Acknowledjement

I would like to express my sincere gratitude to GrapesGenix Technical Solutions Pvt. Ltd., Thrissur, for providing me with the opportunity to undertake a MERN stack-based internship at their esteemed organization. This experience has significantly enhanced my technical skills and practical understanding of modern web development technologies.

A special thanks to Ms. Kripa, our mentor and our trainer, whose valuable guidance, continuous support, and dedicated mentorship made a tremendous impact on my learning journey. Her well-structured sessions and insightful feedback enabled me to understand key concepts clearly and apply them effectively in real-world scenarios.

I am also thankful to the faculty and staff of the Computer Engineering Department at Maharaja's Technological Institute, Thrissur, for their encouragement and support in making this internship possible. Lastly, I extend my heartfelt appreciation to my fellow interns and team members for their collaboration and for creating a positive and enriching learning environment throughout these past four weeks.

Name: LIJO P THOMAS

REG NO: 2301131789

Company profile

GrapesGenix is an initiative with objectives of offering advanced technological solutions in the fields of Information Technology (IT) and Information Technology Enabled services (ITES) including Software Development, Web Designing/Development, Domain Registration/Hosting,Technical Support, Mobile Application Development, Search Engine Optimization and Software Training.

Contact	7736563694
E-mail	mail@grapestechs.com
Address	Grapesgenix Solutions, Near Church, Next Nirmalamatha Technical Lourdes to Central School East Fort, Thrissur.
Working Hours	Mon to Sat : 9:00am - 6:00pm Sun : Closed

INDEX

Internship Overview

INTERN NAME	LIJO P THOMAS
REGISTER NO.	2301131789
INSTITUTION	MAHARAJA'S TECHNOLOGICAL INSTITUTE, THRISSUR
DEPARTMENT	COMPUTER ENGINEERING
COMPANY NAME	Grapesgenix Technical Solutions
INTERNSHIP PERIOD	5 MAY 2025 – 30 MAY 2025
TRAINER	Ms. Kripa – Lead Trainer, MERN Stack Intern

Chapter 1

Introduction

This report offers a detailed look at my four-week MERN Stack development internship at Grapesgenix Technical Solutions thrissur, which was part of my studies at Maharaja's Technological Institute in Thrissur. The goal of the internship was to give us hands-on experience in full-stack web development using MongoDB, Express.js, React.js, and Node.js. We started with the fundamentals of JavaScript programming and progressively moved on to building fully functional and dynamic web applications. Throughout the four weeks, I gained valuable theoretical knowledge and practical skills in frontend and backend development, RESTful API integration, database management, authentication, and application deployment.

Chapter 2

HTML & CSS Introduction

HTML Introduction

The internship commenced with a comprehensive overview of HTML (HyperText Markup Language), the foundational technology used to create web pages. We explored the structure of an HTML document, including the <html>, <head>, and <body> tags, and how they contribute to the rendering of content on web browsers.

Topics Covered

- The significance of HTML in modern web development.
- Syntax rules and document declaration (<!DOCTYPE html>).
- Use of semantic and non-semantic elements.
- Explanation of commonly used tags such as <p>, <h1> to <h6>, <a>, <img>, <br>, and more.
- Introduction to HTML entities (e.g., &nbsp;, &lt;, &gt;) and their importance in rendering special characters.
- Basic attributes like id, class, style, and href.
- Ordered and unordered lists with nesting.
- Embedding and formatting images using <img> tag, understanding attributes like src, alt, width, and height.

- Creating structured data representation using tables with <table>, <tr>, <td>, and <th> tags.
- Introduction to forms as a medium of user interaction.
- Understanding input elements: text fields, radio buttons, checkboxes, submit buttons, and dropdowns.
- CSS syntax and declaration blocks.
- Different methods of including CSS: Inline, Internal, and External.
- Styling of text: color, size, alignment, decoration, and transformation.
- CSS selectors: element, class, id, and group selectors.
- Introduction to the Box Model (margin, border, padding, and content).

Chapter 3
JavaScript Basics & Advanced Concepts

Variable Declarations

- `var` – Function-scoped, can be re-declared and updated
- `let` – Block-scoped, can be updated but not re-declared in same scope
- `const` – Block-scoped, cannot be updated or re-declared

Data Types

- Primitive types: `string`, `number`, `boolean`, `null`, `undefined`
- Complex types: `object`, `array`, `function`

Operators and Expressions

Covered arithmetic, assignment, comparison, logical, and ternary operators, as well as evaluating expressions within conditional structures.

Embedding JavaScript in HTML

We embedded JavaScript code inside `<script>` tags and linked external JS files for better organization.

Conditions, Loops, and Functions

Control flow statements allow decision-making and repetition in JavaScript programs.

Conditional Statements

- `if`, `else if`, `else`
- `switch-case` for multi-branching logic

Looping Constructs

- `for`, `while`, `do...while`
- Iteration through arrays using `for...of` and objects using `for...in`

Functions

- Function declaration vs expression
- Parameters, return values

Events & Modular JavaScript

Modern web applications rely on events and modular design.

Event Handling

- Types: `click`, `mouseover`, `keydown`, `submit`, etc.
- Event listeners using `addEventListener()` method

DOM Manipulation

Accessing and modifying HTML elements using `getElementById`, `querySelector`, and `in-nerHTML`

Modular JavaScript

- Writing reusable code across files
- Import/export modules using **ES6 syntax**
- Introduction to **arrow functions**, `=>`, for cleaner and shorter function expressions

Chapter 4

Web Architecture & Node.js

Client-Server Architecture

- The client sends requests to the server, which processes and responds with data.

HTTP Protocol & Status Codes

- Key methods: `GET`, `POST`, `PUT`, `DELETE`
- Status codes: `200 OK`, `404 Not Found`, `500 Server Error`, etc.

Web Servers

- Introduction to web servers like **Apache**, **IIS**, and **Nginx**

Static vs Dynamic Content : Static pages do not change; dynamic content is rendered based on logic and user interaction.

Node.js Introduction

Node.js allows JavaScript to run on the server, enabling full-stack JS development.

- Non-blocking I/O, event-driven architecture
- Installing Node.js and NPM (Node Package Manager)
- Writing and running `.js` files via the terminal
- Understanding asynchronous vs synchronous execution

Creating & Importing Modules

- Writing modular, reusable code
- `require()` and `module.exports` in CommonJS

NPM & Package Management

- Understanding `package.json`
- Installing dependencies: local vs global packages

Events with `EventEmitter`

- Creating and listening to custom events

File System Operations

- Reading, writing, updating, and deleting files asynchronously

Chapter 5

Express.js & REST API

Setting Up Express

- Installing via NPM, initializing a basic server

Creating Routes

- Defining `GET`, `POST`, `PUT`, `DELETE` routes for handling API requests

Sending Responses

- Using `res.send()` and `res.json()` to deliver responses

Middleware

- Using third-party and custom middleware
- Applying `cors()` to enable Cross-Origin requests

RESTful API Concepts

- Designing resource-based URIs
- Using tools like **Postman** for API testing
- Understanding headers, parameters, body, and query strings

Chapter 6

MongoDB & Mongoose

MongoDB Basics

- Collections, documents, and CRUD operations (`insert`, `find`, `update`, `delete`)
- Working with the Mongo shell

BSON vs JSON

- Understanding the data formats used internally

Mongoose Integration

- Defining schemas and models
- Schema types, validation rules, and relationships
- Querying, updating, and removing documents using Mongoose methods

Middleware and Hooks

- Using `pre` and `post` hooks for custom logic during database operations

Chapter 7

Authentication& React.js Frontend

JWT (JSON Web Tokens)

- Secure token generation for authenticated sessions
- Using `jsonwebtoken` library to create and verify tokens

Protecting Routes

- Writing middleware to check for valid tokens
- Creating login and registration endpoints

Password Hashing

- Using `bcrypt` to securely hash and compare passwords

Introduction to React

Virtual DOM

- React creates a virtual copy of the DOM and only updates what changes, improving performance.

JSX Syntax

- JavaScript + HTML combined using JSX

Component-Based Architecture

- Functional and class-based components
- Passing data using **props**

Project Setup

- Creating apps using `create-react-app` (CRA)
- File and folder structure

State and Lifecycle

useState and useEffect

- Managing dynamic data and side effects like API calls

Form Handling

- Controlled vs uncontrolled components
- Two-way data binding

Rendering Lists and Conditional UI

- Using `.map()` and conditional (`&&`, ternary) rendering

Component Lifecycle

- Mounting, updating, and unmounting behavior using hooks

Chapter 8

Final Project: MERN Stack E-commerce Website

We applied everything learned to build an E-commerce web application using the MERN stack.

Backend:

- Built with Node.js and Express
- API endpoints for products
- MongoDB as the database
- Mongoose for schema modeling
- Seed script to populate initial product data

Frontend:

- React components for products, cart, filtering
- State management using React context
- Axios for fetching data from backend
- Custom styling and responsive UI
- Functionality for filtering, sorting, cart updates

Deployment:

- Server and client setup
- Package structure for both backend and frontend
- `npm start` commands for development

Prerequisites:

- React
- Node
- Express
- Mongodb and Mongoose
- CORS
- MERN Stack

Approach to create E-commerce Website:

- Server-side API for fetching product data
- React components for displaying product information
- MongoDB for storing product details
- CORS middleware to handle cross-origin requests

Steps to Create the E-commerce Website with MERN Stack:

1. Setting up backend

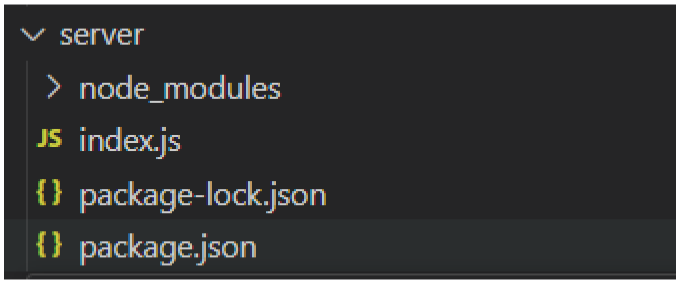
Create a directory for project

```
mkdir server
cd server
```

Step 2: Initialized the Express app and installing the required packages

```
npm init -y
npm i express mongoose cors
```

Project Structure(Backend):



```
server
  node_modules
  index.js
  package-lock.json
  package.json
```

The updated dependencies in package.json .

```
"dependencies": {
   "cors": "^2.8.5",
   "express": "^4.18.3",
   "mongoose": "^8.2.1",
```

index.js

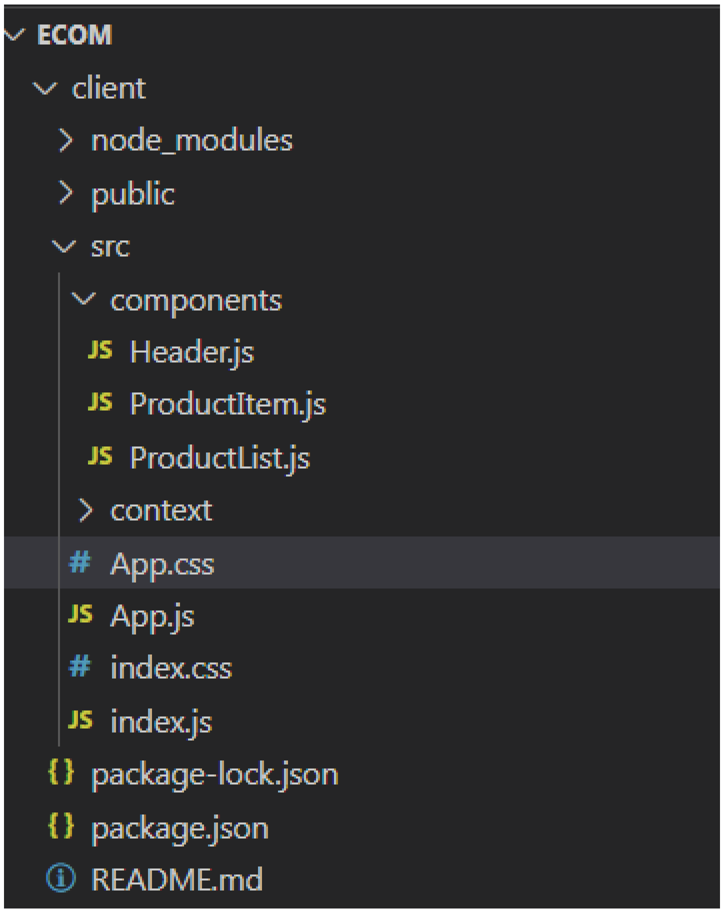
```
const express = require('express');
const mongoose = require('mongoose');
const app = express();
const PORT = process.env.PORT || 5000;
const cors = require('cors');
 //replace the link with your mongodb atlas link
mongoose.connect('your_mongodb_url',
  {
   useNewUrlParser: true,
   useUnifiedTopology: true
  }
);

app.use(express.json());
app.use(cors()); // Use the cors middleware

const productSchema = new mongoose.Schema({
  name: String,
  type: String,
  description: String,
  price: Number,
  image: String,
});

const Product = mongoose.model('Product', productSchema);

// Function to seed initial data into the database
const seedDatabase = async () => {
  try {
    await Product.deleteMany(); // Clear existing data

    const products = [
      {
        name: "Men's Casual T-shirt",
        type: 'Men',
        description: 'Comfortable and stylish casual T-shirt for men',
        price: 350,
        image:
'https://mediaorg/wp-content/uploads/20230407153931/gfg-tshirts.jpg'
      },
      {
        name: 'Luxury bag',
        type: 'Not Applicable',
        description: 'Elegant luxury bag with leather strap',
        price: 2500,
        image:
'https://media.org/wp-content/uploads/20230407154213/gfg-bag.jpg'
      },
      {
        name: "Hoodie",
        type: 'Men',
        description: 'Light and classy hoodies for every seasons ',
        price: 450,
        image:
'https://media.org/wp-content/uploads/20230407153938/gfg-hoodie.jpg'
      },
      {
```

```
    name: 'Remote Control Toy car',
    type: 'Not Applicable',
    description: 'High-quality Toy car for fun and adventure',
    price: 1200,
    image:
'https://media.org/wp-content/uploads/20240122182422/images1.jpg'
  },
  {
    name: 'Books',
    type: 'Women',
    description: 'You wll have a great time reading .',
    price: 5000,
    image:
'https://media.org/wp-content/uploads/20240110011854/reading-925589_640.jpg'
  },
  {
    name: 'Bag',
    type: 'Men',
    description: 'Comfortable and supportive Bag ',
    price: 800,
    image:
'https://media.org/wp-content/uploads/20230407154213/gfg-bag.jpg'
  },
  {
    name: 'Winter hoodies for women',
    type: 'Women',
    description: 'Stay cozy in style with our womens hoodie, crafted for comfort ',
    price: 250,
    image:
'https://media.org/wp-content/uploads/20230407153938/gfg-hoodie.jpg'
  },

  {
    name: 'Honda car ',
    type: 'Men',
    description: 'Powerful Honda car with comfy driving',
    price: 700,
    image:
'https://media.org/wp-content/uploads/20240122184958/images2.jpg'
  }
  ];

  await Product.insertMany(products);
  console.log('Database seeded successfully');
 } catch (error) {
  console.error('Error seeding database:', error);
 }
};

// Seed the database on server startup
seedDatabase();

// Define API endpoint for fetching all products
app.get('/api/products', async (req, res) => {
 try {
  // Fetch all products from the database
  const allProducts = await Product.find();

  // Send the entire products array as JSON response
  res.json(allProducts);
 } catch (error) {
  console.error(error);
  res.status(500)
   .json({ error: 'Internal Server Error' });
 }
});

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

Start the backend server with the following command:

```
node index.js
```

2. Creating FrontEnd



```
ECOM
  client
    node_modules
    public
    src
      components
        Header.js
        ProductItem.js
        ProductList.js
      context
      App.css
      App.js
      index.css
      index.js
    package-lock.json
    package.json
    README.md
```

Step 1: Set up React frontend using the command.

```
npx create-react-app client cd client
```

Step 2: Install the required dependencies.

```
npm i @fortawesome/free-solid-svg-icons
npm i @fortawesome/react-fontawesome
```

Project Structure(Frontend):

The updated dependencies in package.json file of frontend will look like:

```
"dependencies": {
   "@fortawesome/free-solid-svg-icons": "^6.5.1",
   "@fortawesome/react-fontawesome": "^0.2.0",
   "axios": "^1.6.7",
   "react": "^18.2.0",
   "react-dom": "^18.2.0",
   "react-router-dom": "^6.22.3",
   "react-scripts": "5.0.1",
   "sass": "^1.71.1",    "web-vitals": "^2.1.4"
}
```

/* App.css */

```
.cart-items {
  border-radius: 50%;
  background-color: rgb(20, 158, 105);
  font-weight: 700;
  color: aliceblue;
  width: 30px;
  height: 30px;
  font-size: 1.5rem;
  padding: 10px;
  position: relative;
  top: 10px;
  left: 30px;
}

.header {
  display: flex;
  justify-content: space-evenly;
  align-items: center;
  padding: 10px;
  border-bottom: 1px solid #ccc;
}
```

// src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root =
   ReactDOM.createRoot(document.getElementById('root'));
root.render(
   <React.StrictMode>
      <App />
   </React.StrictMode>
);
```

// App.js

```
import React from 'react';

import ProductList
   from './components/ProductList';
import Header
   from './components/Header';
import './App.css'
import CustomItemContext
   from './context/ItemContext';

const App = () => {
   return (
      <CustomItemContext>
         <Header />
         <ProductList />
      </CustomItemContext>
   );
};

export default App;
```

// client/src/components/Header.js

```
import React, { useContext } from 'react';
import { FontAwesomeIcon }
   from '@fortawesome/react-fontawesome'
import { faCartShopping }
   from '@fortawesome/free-solid-svg-icons'
import { itemContext } from '../context/ItemContext';

const Navbar = () => {

   const { itemsInCart, totalPrice } = useContext(itemContext);

   return (
      <nav className='navbar'>
         <div className='navbar-brand'>
            <h1 className='ecom'>
               My Ecommerce Website
            </h1>
         </div>
         <div className='navbar-items'>
            <h3 style={{ color: "green" }}>
               Total Price: {totalPrice}
            </h3>
            <div className='cart-num'>
               <FontAwesomeIcon
                  icon={faCartShopping} size="2x" />
               <div className='cart-items'>
                  {itemsInCart}
               </div>
            </div>
         </div>
      </nav>
   );
};

export default Navbar;
```

// client/src/components/ProductItem.js

```
import React, { useContext } from 'react';
import { itemContext } from '../context/ItemContext';

const ProductItem = ({ product }) => {
   const { addToCart, removeFromCart } = useContext(itemContext)
   const handleAddToCart = (product) => {
      console.log(product)
      addToCart(product)

   };
   const handleRemoveToCart = (product) => {
      console.log("product removed", product)
      removeFromCart(product)

   };
   return (
      <div className="product-card">
         <img className="product-image"
            src={product.image}
            alt={product.name} />
         <div className="product-details">
            <h3 style={{ fontWeight: "700" }}>
               {product.name}
            </h3>
            <p style={{ fontWeight: "300" }}>
               {product.description}
            </p>
            <p style={{ fontWeight: "500" }}>
               Price: {product.price} Rs
            </p>
            <button onClick={
               () => handleAddToCart(product)
            }>
               Add to Cart
            </button>
            <button onClick={
               () =>
                  handleRemoveToCart(product)
            }>
```

```
                -
              </button>
          </div>
      </div>
   );
};

export default ProductItem;
```

// client/src/components/ProductList.js

```
import React,
{
   useContext,
   useEffect,
   useState
} from 'react';
import ProductItem from './ProductItem';
import { itemContext } from '../context/ItemContext';

const ProductList = () => {
   const { products } = useContext(itemContext);
   // Keep a local state for sorted products
   const [sortedProducts, setSortedProducts] =
      useState([...products]);
   const [minPrice, setMinPrice] = useState(0);
   const [maxPrice, setMaxPrice] = useState(3000);
   // 'all' represents no type filter
   const [selectedType, setSelectedType] = useState('all');

   useEffect(() => {
      setSortedProducts([...products])
   }, [products])

   const handleSortByPrice = () => {
      const sorted = [...sortedProducts]
         .sort((a, b) => a.price - b.price);
      setSortedProducts(sorted);
   };
```

```
const handleFilterByPriceRange = () => {
  const filtered =
    products.filter(
      (product) =>
        product.price >= minPrice &&
        product.price <= maxPrice);
  setSortedProducts(filtered);
};
const handleFilterByType = () => {
  if (selectedType === 'all') {
    // Reset the type filter
    setSortedProducts([...products]);
  } else {
    const filtered =
      products.filter(
        (product) =>
          product.type === selectedType);
    setSortedProducts(filtered);
  }
};
return (
  <div className='prdt-list'>
    <h2>Product List</h2>
    <div className='filter-btn'>
      <button onClick={handleSortByPrice}>
        Sort by Price
      </button>
      <label>
        Min Price:
        <input type='number' value={minPrice}
          onChange={
            (e) =>
              setMinPrice(Number(e.target.value))
          } />
      </label>
      <label>
        Max Price:
        <input type='number' value={maxPrice}
          onChange={
            (e) =>
```

```
                setMaxPrice(Number(e.target.value))
            } />
        </label>
        <button onClick={() => handleFilterByPriceRange()}>
          Filter by Price Range
        </button>
        <label>
          Filter by Type:
          <select value={selectedType}
            onChange={
              (e) =>
                setSelectedType(e.target.value)
            }>
            <option value='all'>
              All
            </option>
            <option value='Men'>Men</option>
            <option value='Women'>Women</option>
          </select>
        </label>

        <button onClick={handleFilterByType}>
          Filter by Type
        </button>
      </div>
      <ul className='item-card'>
        {sortedProducts.map((product) => (
          <ProductItem key={product._id}
            product={product} />
        ))}
      </ul>
      <div className='buy-now-btn'>Buy Now</div>
    </div>
  );
};

export default ProductList;
```

/* ProductItem.css */

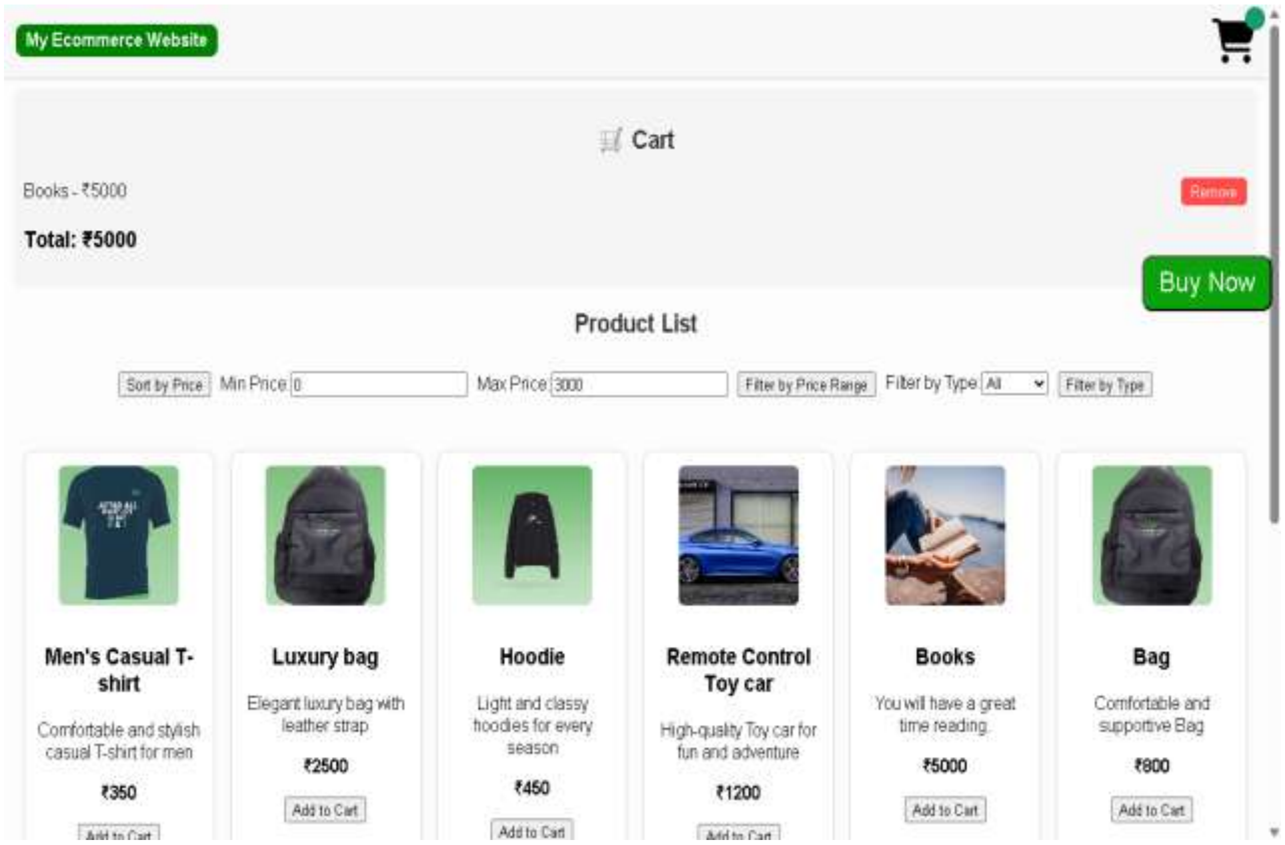
```css
.product-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  width: fit-content;
  padding: 16px;
  margin: 16px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  background-color: #fff;
  display: flex;
  flex-direction: column;
  align-items: center;
}

.product-image {
  width: 200px;
  height: 200px;
  object-fit: cover;
  border-radius: 10px;
  margin-bottom: 12px;
  transition: transform 0.3s ease-in-out;
}

.product-image:hover {
  transform: scale(1.1);
  /* Enlarge the image on hover */
}

.product-details {
  text-align: center;
}

.item-card {
  display: flex;
  flex-wrap: wrap;
}

h2 {
  text-align: center;
  color: #333;
```

```css
}

.filter-btn {
  display: flex;
  flex-direction: row;
  padding: 10px;
  gap: 10px;
  justify-content: center;
}

.prdt-list {
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.cart-num {
  margin-bottom: 40px;
  cursor: pointer;
  font-size: 1.2rem;
}

.buy-now-btn {
  background-color: rgb(11, 162, 11);
  color: white;
  padding: 8px 15px;
  border-radius: 10px;
  font-size: 1.5rem;
  position: fixed;
  top: 30%;
  right: 10px;
  cursor: pointer;
}

.buy-now-btn:hover {
  background-color: rgb(113, 230, 113);
  color: brown;
}

.ecom {
  background-color: green;
  color: white;
  padding: 8px 15px;
  border-radius: 10px;
  font-size: 1.2rem;
}
```

Start frontend code:

```
npm start
```

Output:



Chapter 9

Conclusion

The 20-day internship at GrapesGenix Technical Solutions offered an invaluable opportunity to gain hands-on experience in full-stack web development using the MERN stack — MongoDB, Express.js, React.js, and Node.js. This internship was not just a learning journey but a practical dive into the real-world processes of building dynamic and scalable web applications.

Beginning with the foundational layers of HTML and CSS, we gradually advanced through JavaScript programming, explored backend development with Node.js, built APIs using Express.js, and implemented robust database operations with MongoDB. The integration of Mongoose allowed us to structure and manage data efficiently, while React introduced us to the component-based architecture essential for creating modern, responsive user interfaces.

Each phase of the internship was thoughtfully designed to build on the previous one, ensuring a smooth and logical progression. The hands-on assignments, live coding sessions, and collaborative tasks deepened our understanding and helped us transition from theoretical concepts to practical implementation with confidence.

By the end of the internship, we were able to successfully develop and deploy a complete E-commerce application — a testament to the skills and knowledge we had acquired. This experience has not only strengthened our technical foundation but also instilled in us a problem-solving mindset, teamwork spirit, and the ability to adapt to fast-evolving web technologies.

Overall, this internship served as a stepping stone toward becoming proficient full-stack developers, and it has left us better prepared for the challenges and expectations of the tech industry.

Chapter 10

References

1. **MongoDB Documentation** – https://www.mongodb.com/docs/
2. **Express.js Guide** – https://expressjs.com/
3. **React Official Documentation** – https://react.dev/
4. **Node.js Documentation** – https://nodejs.org/en/docs/
5. **Socket.io Documentation** – https://socket.io/docs/
6. **JWT (JSON Web Tokens)** – https://jwt.io/introduction
7. **W3Schools Tutorials** – https://www.w3schools.com/