

Project Report

ShopSmart: Your Digital Grocery Store Experience

Team ID: LTVIP2025TMID60104

Team Size: 4

Team Members

Team Leader: Bommagoni Yaswanth

Team Member: Chebrolu Reethika

Team Member: Abburi Venkata Prabhas

Team Member: Annamnedi Manoj Kumar

1. Introduction

In the era of rapid digital transformation, the grocery retail sector has witnessed a significant shift toward e-commerce and digital experiences. "ShopSmart" is a full-stack web application designed to provide a seamless digital grocery shopping experience to customers. This project focuses on creating a user-friendly platform that allows users to browse, search, and purchase groceries with ease, while also enabling admin functionalities for product and order management.

2. Objective

- To develop a digital platform that streamlines the grocery shopping process.
- To enhance customer experience through intuitive design and smart features.
- To implement full-stack technologies for robust front-end and back-end functionality.
- To provide admins with tools to manage inventory, orders, and customer data.

3. Key Features

- User Authentication: Secure registration and login system.
- Product Catalog: Browse a wide range of grocery items categorized for convenience.
- Search & Filter: Easily search for products using keywords and apply filters.
- Cart & Checkout: Add items to cart and proceed with order placement.
- Order History: Users can view their past orders and status.
- Admin Panel: Admins can add, update, and delete products; manage users and view orders.
- Responsive Design: Mobile-friendly interface for ease of access.

4. Technologies Used

Technology	Purpose
HTML, CSS, JavaScript	Frontend Structure & Styling
React.js	Frontend Framework (UI Components)
Node.js & Express	Backend Logic & API Handling
MongoDB	NoSQL Database (Product & User Data)
Git & GitHub	Version Control
Postman	API Testing and Validation

5. System Architecture

Frontend (React.js):

- Home page with featured products
- Product listing with filters
- Cart and checkout system
- Login/Register functionality

Backend (Node.js + Express):

- RESTful API for product and user operations
- JWT-based user authentication
- Role-based access for users and admins

Database (MongoDB):

- Collections for Users, Products, Orders
- Structured schema for scalability

6. Modules Overview

- User Module: Registration, login, profile management
- Product Module: Display, add to cart, view details
- Cart Module: Add/remove items, update quantity
- Admin Module: CRUD operations on products, view orders
- Order Module: Place order, update status, track order

7. Development Process

- Phase 1: Requirements gathering and UI/UX design
- Phase 2: Backend API development
- Phase 3: Frontend integration with backend
- Phase 4: Testing and bug fixing
- Phase 5: Final deployment

8. Challenges Faced

- Managing state efficiently in frontend
- Ensuring secure authentication and role management
- Database schema design for scalability
- Integration of frontend and backend APIs

9. Outcomes

- A fully functional digital grocery store web application
- Practical implementation of full-stack technologies
- Improved teamwork and collaborative problem-solving skills
- Better understanding of real-time web application development

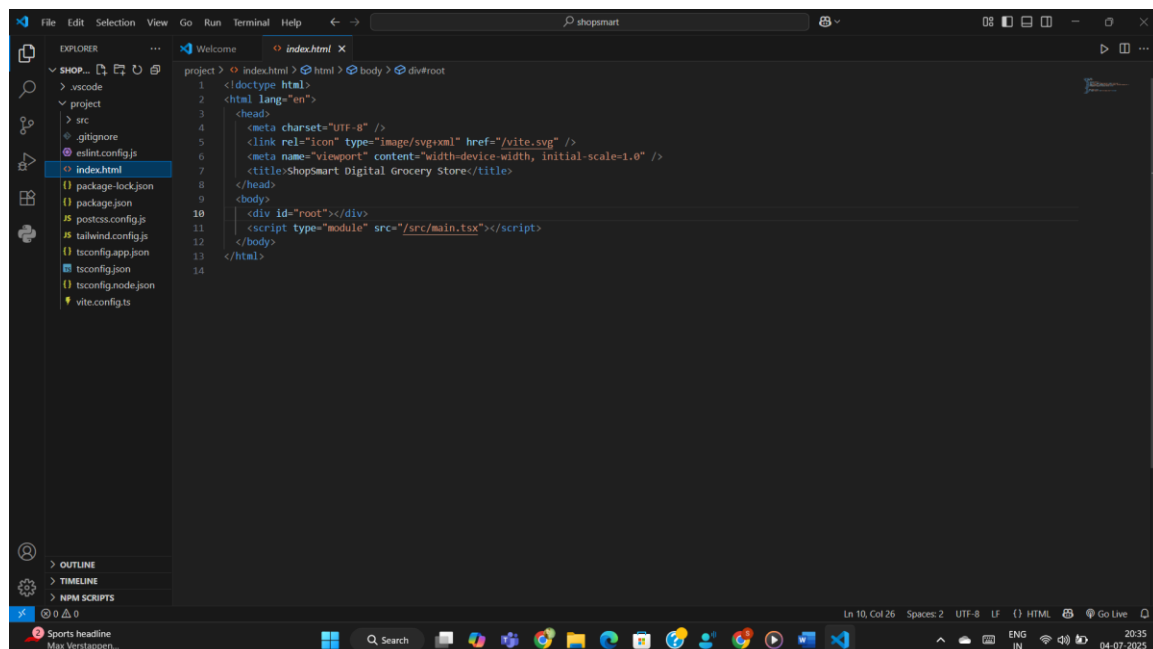
10. Future Enhancements

- Integration with payment gateways (Razorpay, Stripe)
- Real-time inventory updates
- Push notifications for offers and order status
- AI-based product recommendation engine
- Mobile app version using React Native

11. Conclusion

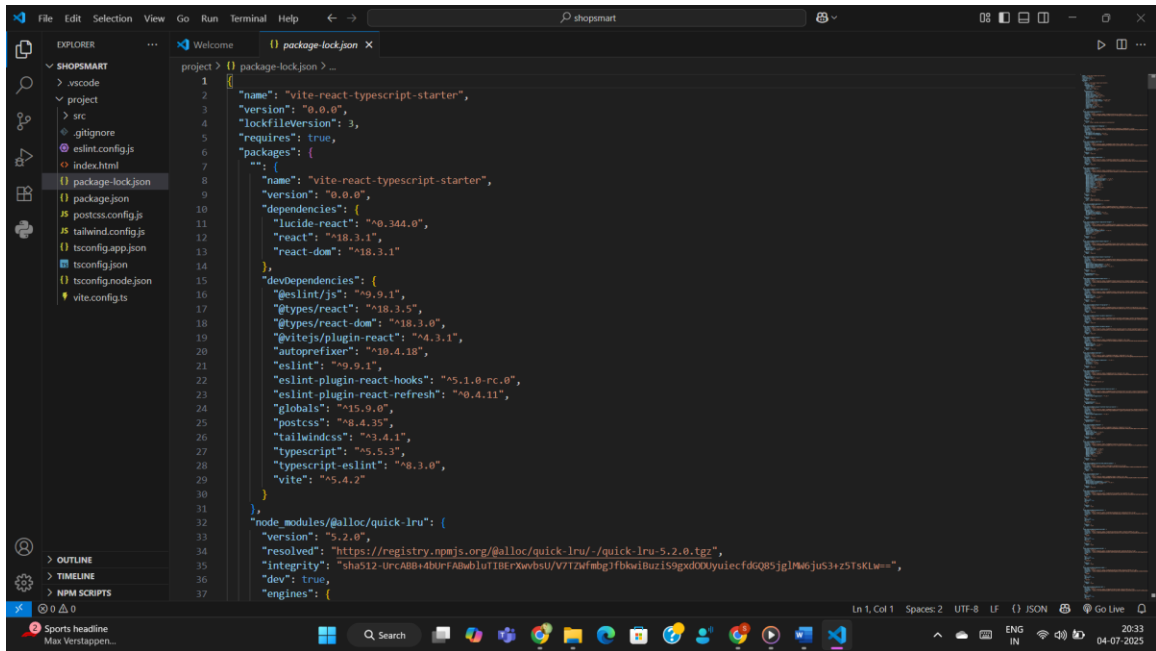
ShopSmart is more than just a digital grocery store – it is a step toward simplifying and enriching the online shopping experience. This project has given our team valuable exposure to full-stack development, project collaboration, and real-world problem-solving. With further improvements and scalability, ShopSmart can be turned into a commercially viable platform.

Executable code :



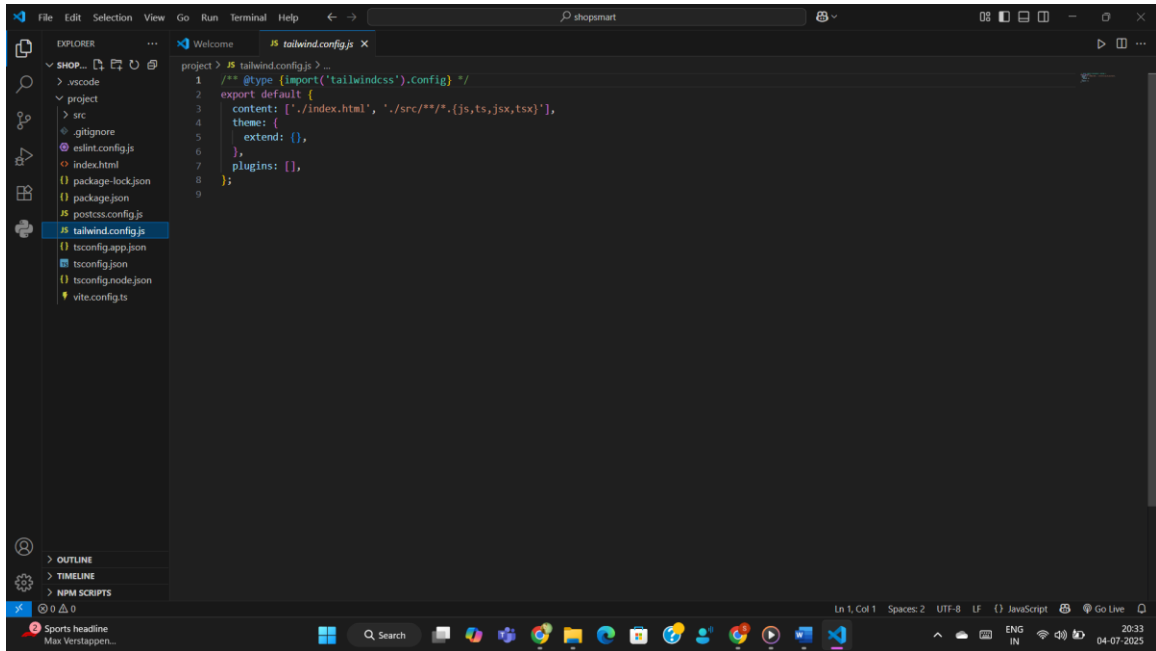
The screenshot shows a Visual Studio Code editor window with a dark theme. The Explorer sidebar on the left displays a file tree for a project named 'SHOP...'. The file 'index.html' is selected and its content is shown in the main editor area. The code is a standard HTML5 boilerplate for a web application. The status bar at the bottom indicates the cursor is at line 10, column 26, in a UTF-8 file with 2 spaces.

```
1 <!doctype html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8" />
5   <link rel="icon" type="image/svg+xml" href="/vite.svg" />
6   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7   <title>ShopSmart Digital Grocery Store</title>
8 </head>
9 <body>
10  <div id="root"></div>
11  <script type="module" src="/src/main.tsx"></script>
12 </body>
13 </html>
14
```



This screenshot shows the Visual Studio Code editor with the `package-lock.json` file open. The Explorer sidebar on the left shows the project structure, including files like `package.json`, `tailwind.config.js`, and `tsconfig.json`. The main editor area displays the content of `package-lock.json`, which is a JSON file containing metadata for the project's dependencies. The file includes fields for `name`, `version`, `lockfileVersion`, `requires`, `packages`, `devDependencies`, `node_modules/@alloc/quick-lru`, `resolved`, `integrity`, `dev`, and `engines`. The status bar at the bottom indicates the current line and column (Ln 1, Col 1) and the file encoding (UTF-8).

```
1 {
2   "name": "vite-react-typescript-starter",
3   "version": "0.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "vite-react-typescript-starter",
9       "version": "0.0.0",
10      "dependencies": {
11        "lucide-react": "^0.344.0",
12        "react": "^18.3.1",
13        "react-dom": "^18.3.1"
14      },
15      "devDependencies": {
16        "eslint/js": "^9.9.1",
17        "@types/react": "^18.3.5",
18        "@types/react-dom": "^18.3.0",
19        "@vitejs/plugin-react": "^4.3.1",
20        "autoprefixer": "^10.4.18",
21        "eslint": "^9.9.1",
22        "eslint-plugin-react-hooks": "^5.1.0-rc.0",
23        "eslint-plugin-react-refresh": "^0.4.11",
24        "globals": "^15.9.0",
25        "postcss": "^8.4.35",
26        "tailwindcss": "^3.4.1",
27        "typescript": "^5.5.3",
28        "typescript-eslint": "^8.3.0",
29        "vite": "^5.4.2"
30      },
31    },
32    "node_modules/@alloc/quick-lru": {
33      "version": "5.2.0",
34      "resolved": "https://registry.npmjs.org/@alloc/quick-lru/-/quick-lru-5.2.0.tgz",
35      "integrity": "sha512-UrcAD9IRG/foeWk7Hk4Js1Nk+T8nOy1BdlPw+u8XUYWkMXu4+wZcQbH8o7ryx1eNtZdWKhjVMS9H4Xp17g==",
36      "dev": true,
37      "engines": {
```



This screenshot shows the Visual Studio Code editor with the `tailwind.config.js` file open. The Explorer sidebar on the left shows the project structure, including files like `package.json`, `tailwind.config.js`, and `tsconfig.json`. The main editor area displays the content of `tailwind.config.js`, which is a JavaScript file used to configure Tailwind CSS. The file includes comments and code for `export default`, `content`, `theme`, `extend`, and `plugins`. The status bar at the bottom indicates the current line and column (Ln 1, Col 1) and the file encoding (UTF-8).

```
1 /** @type {import('tailwindcss').Config} */
2 export default {
3   content: ['./index.html', './src/**/*.{js,ts,jsx,tsx}'],
4   theme: {
5     extend: {},
6   },
7   plugins: [],
8 };
9
```

Output :

