Customer order management and Invoice generator

Project Report Submitted in Partial Fulfilment of the Requirements for the Degree of

Bachelor of Engineering *in* Information Technology

Submitted by

Chandra Prakash Goyal

(Roll No. 22UITE5009)

Under the Supervision of

Dr. Simran Chaudhary

Professor , CSE

Department of Computer Science and Engineering

MBM University, Jodhpur

May, 2025

Department of Computer Science & Engineering

Faculty of Engineering & Architecture, MBM University

Ratanada, Jodhpur, Rajasthan, India –342011

CERTIFICATE

This is to certify that the work contained in this report entitled **"Customer Order management and Invoice generator"** is submitted by Mr. Chandra Prakash Goyal (Roll. No: 22UITE5009) to the Department of Computer Science & Engineering, MBM University, Jodhpur, for the partial fulfilment of the requirements for the degree of **Bachelor of Engineering in Information Technology.**

They have carried out their work under my supervision. This work has not been submitted else-where for the award of any other degree or diploma.

The project work in our opinion, has reached the standard fulfilling of the requirements for the degree of Bachelor of Engineering in accordance with the regulations of the University.

Dr. Simran Chaudhary,
(Mentor)
Prof, CSE Department

Dr. Shrawan Ram
(Head of Department, CSE)

DECLARATION

I, Chandra Prakash Goyal (IVth Year), hereby declare that this training report titled "Customer Order management and Invoice generator" is a record of original work done by me under the supervision and guidance of Mentor Dr. Simran Chaudhary.

I, further certify that this work has not formed the basis for the award of the Degree/Diploma/Associateship/Fellowship or similar recognition to any candidate of any university and no part of this report is reproduced as it is from any other source without appropriate reference and permission.

(Chandra Prakash Goyal)

8th Semester, Information Technology

Roll No. – 22UITE5009

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who have supported and guided me throughout the development of my project titled "Customer Order Management and Invoice Generator."

I would like to express my heartfelt gratitude to Dr. Simran Chaudhary, Professor, Department of Computer Science and Engineering, for her invaluable mentorship, constant guidance, and encouragement throughout the development of my project titled *"Customer Order Management and Invoice Generator."* Her insightful feedback and unwavering support played a pivotal role in shaping this project and enhancing my understanding of the subject matter.

I am also grateful to the Department of Computer Science and Engineering, MBM University, Jodhpur, for their continuous academic support and for equipping me with the foundational knowledge that enabled me to successfully complete this project.

I wish to acknowledge the support of Dr. Shrawan Ram Balach, Head of the Department, CSE, for providing access to essential resources and fostering an environment that encourages academic excellence and innovation. His leadership and commitment to student development significantly contributed to the success of this project.

ABSTRACT

In today's fast-paced business environment, efficient and accurate order processing is crucial for maintaining customer satisfaction and operational effectiveness. The Customer Order Management and Invoice Generator project aims to streamline the entire order handling process — from order placement to invoice generation — within a single, integrated system. This project provides a digital solution that automates and simplifies customer interactions, product ordering, and financial documentation.

The system is designed to manage customer data, track orders, monitor inventory levels, and generate professional invoices with minimal manual input. It features a user-friendly interface that allows customers or sales representatives to create and modify orders quickly. Upon confirmation, the system automatically calculates totals, taxes, and discounts, and generates downloadable or printable invoices. This not only enhances accuracy but also reduces processing time and minimizes human error.

Key functionalities of the system include real-time order tracking, customer information management, inventory checks, dynamic pricing, and customizable invoice templates. Admin users can access analytics and order histories to gain insights into sales trends and customer behavior. By implementing this solution, businesses can improve workflow efficiency, ensure timely billing, and maintain organized financial records.

Overall, this project provides a practical, scalable tool for businesses of all sizes to manage their customer orders and invoicing processes digitally, ultimately leading to improved customer service and better financial management.

Contents

Chapter 1

Introduction

In today's digital era, businesses increasingly rely on software systems to manage everyday operations with precision and efficiency. Order management and invoice generation are two critical aspects of running a business, especially for service providers and retailers. Manual methods, such as handwritten invoices or spreadsheet-based order tracking, are not only time-consuming but also prone to errors and mismanagement. To address these challenges, this project presents a web-based solution titled **"Customer Order Management and Invoice Generator"**, which automates the order-to-invoice process, including tax calculation. Developed using the MERN stack with TypeScript, this system aims to simplify workflows, improve accuracy, and offer a smooth user experience for both administrators and customers.

1.1 History & Background:

Among the growing complexity of business transactions, especially in small to medium-sized enterprises (SMEs), there is a growing need for digital tools that can streamline routine operations. Managing customer details, tracking orders, and generating accurate invoices are essential for business success, but performing these tasks manually can lead to inefficiencies. Automation not only reduces the chances of error but also saves time and improves record-keeping. Moreover, calculating taxes such as GST or VAT manually can be inconsistent and may lead to legal and financial discrepancies.

To meet these demands, the **Customer Order Management and Invoice Generator** system was developed as a comprehensive platform that combines order handling, tax calculation, and invoice generation into a single web-based solution. The system ensures

that every transaction is logged, every invoice is accurate, and customer satisfaction is maintained through timely and professional documentation.

1.2 Problem Statement:

Manual processes for handling orders and generating invoices can lead to several operational bottlenecks. These include data entry mistakes, delayed invoice generation, difficulty in tracking customer transactions, and inconsistent tax calculations. These challenges become even more pronounced as the number of customers and orders increases. Businesses using outdated methods often struggle to maintain organized records and accurate financial statements.

This project seeks to resolve these issues by developing a user-friendly, automated solution that streamlines the entire process from order creation to invoice generation, including dynamic tax calculation. By leveraging modern technologies, the system offers better accuracy, efficiency, and scalability for businesses looking to modernize their operations.

1.3 Objectives of the Project:

The primary objective of this project is to design and implement a full-stack web application that enables businesses to manage customer orders and generate tax-compliant invoices automatically. The system should be easy to use, responsive, and scalable for future needs.

Key objectives include:

- Designing a clean and intuitive user interface using React.
- Building a secure and scalable back-end using Node.js and Express.
- Storing and managing data efficiently using MongoDB.
- Providing features to add, update, and delete customer and order information.
- Enabling users to view, download, or print invoices in professional formats (e.g., PDF).

1.4 Scope of the Project:

The scope of this project is limited to building a web application suitable for small and medium businesses that require a structured system to handle customer orders and billing. The system includes modules for customer management, product listings, order processing, tax-inclusive invoice generation, and administrative control.

While the system provides essential features for order and invoice handling, it does not include real-time inventory tracking or payment gateway integration in the current version. However, it has been designed in a modular way to allow easy future integration of such features. The application focuses on simplicity, reliability, and accuracy, ensuring it meets the core needs of business users without unnecessary complexity.

1.5 Methodology overview:

The development of the Customer Order Management and Invoice Generator system follows a modular, full-stack approach using the MERN stack (MongoDB, Express.js, React, Node.js) combined with TypeScript. This stack was chosen for its flexibility, scalability, and strong support for web-based applications. Each component of the system was developed and tested in phases to ensure smooth integration and optimal performance.

The project began with requirement analysis, where the core functionalities were identified, including customer and order management, invoice creation, and tax computation. Following this, the system design phase involved creating data flow diagrams, defining the database schema, and planning the component structure for the front end. In the implementation phase, the React-based front end was developed with TypeScript for better code organization and type safety. Simultaneously, the Express and Node.js back end was configured to handle API requests, manage authentication, and perform business logic.

1.6 Significance of the project:

This project holds considerable significance in both academic and practical contexts. From a practical standpoint, it offers a solution to a common problem faced by many

businesses — managing orders and generating accurate, tax-inclusive invoices. The automation of these processes helps reduce errors, improve efficiency, and ensure compliance with tax regulations. By using this system, businesses can enhance their operational workflow, improve record-keeping, and deliver a more professional experience to their clients.

Academically, this project demonstrates the effective application of full-stack development using modern web technologies. It showcases how a real-world problem can be analysed, planned, and solved through structured software development practices. It also emphasizes key software engineering concepts such as modular design, responsive front-end development, secure API creation, and database integration.

The inclusion of tax calculation and PDF generation makes this project more complete and realistic, aligning closely with actual business needs. Furthermore, the choice of the MERN stack with TypeScript reflects a modern and in-demand skill set in the field of web development, making the project not only technically sound but also professionally relevant.

Chapter 2

Requirement Modelling

Understanding and modelling the requirements of a system is a crucial step in the software development lifecycle. It ensures that the final product meets the intended goals and satisfies user needs. This chapter outlines the functional and non-functional requirements of the **Customer Order Management and Invoice Generator** system, followed by a brief description of the technologies and tools selected to implement them. The goal of requirement modelling is to define what the system should do, how it should behave, and what constraints or performance expectations it must meet. Clear and accurate modelling also helps in ensuring scalability, maintainability, and usability of the system.

2.1 Functional Requirements:

Functional requirements describe the specific behavior or functions of the system. These are the core features that the system must support to be considered successful.

- **Customer management:** The system should allow administrators to add, update, and delete customer records, including contact details and order history.
- **Product Management**: Admins should be able to add and manage products, including names, prices, and descriptions.
- **Order Handling**: The system should allow placing new orders, modifying existing orders, and tracking the status of all active and completed orders.
- **Invoice Generation**: The system must automatically generate invoices based on order data, including item details, quantity, unit price and total cost.
- **Authentication and Authorization**: Users must register and log in securely to access the platform. Admin-level users should have access to all controls, while regular users should have limited access.

- **PDF Export and Printing**: Invoices should be viewable, downloadable as PDFs, and printable in a standardized format.
- **Real-time Validation**: The system should validate user inputs (e.g., required fields, valid email formats) in real time.

2.2 Non-Functional Requirements:

Non-functional requirements describe how the system performs rather than what it does. They impact user experience, system architecture, and long-term maintainability.

- **Scalability**: The system must be able to handle an increasing number of users and transactions as the business grows.
- **Security**: Authentication and data transmission must be secure. JWT (JSON Web Tokens) is used for secure token-based session management, and Google reCAPTCHA is integrated to prevent bot attacks during login and signup.
- **Performance**: The system should provide fast response times, even during peak usage.
- **Usability**: The interface should be intuitive and responsive across devices, ensuring a good experience for users with varying technical skills.
- **Maintainability**: The code should be modular and well-documented to allow for future updates and bug fixes.
- **Availability**: The system should aim for high availability, minimizing downtime during usage.

2.3 Technological Stack and Tools:

The system is built using the **MERN stack**, which includes MongoDB, Express.js, React, and Node.js. These technologies were selected for their speed, scalability, and active community support.

- **MongoDB** is used for storing customer, product, and invoice data in a flexible, schema-less format, making it easier to manage evolving data structures.
- **Express.js** serves as the back-end web framework, handling HTTP requests and middleware.

- **React.js with TypeScript** is used to create the front-end interface, offering type safety and better code organization.
- **Node.js** powers the server environment, enabling fast, asynchronous operations.
- **JWT (JSON Web Token)** is used for secure authentication, allowing users to remain logged in while ensuring that each request is verified.
- **Google reCAPTCHA** is integrated into the authentication system to protect against automated sign-ups and brute-force attacks.
- **CSS and component libraries** are used for responsive UI design and to enhance user experience.

2.4 User Roles and System Access:

The system supports multiple user roles with different access levels:

- **Administrator**: Has full access to customer, product, order, and invoice management. Admins can also view analytics and perform updates or deletions.
- **Authenticated User (Customer)**: Can view their own orders and invoices but cannot access or manage other user data.
- **Guest**: Can browse public pages (if any) but cannot access or interact with secure data without registration and login.
- **Secure Access**: Access to customer data and order history is restricted exclusively to the authenticated user, ensuring privacy.

2.5 System Workflow:

Understanding how different components of the system interact is essential for effective development and user experience. The workflow of the Customer Order Management and Invoice Generator system is designed to be intuitive, with clear navigation from one functionality to another. Below is a simplified description of the main process flow:

- **User Authentication**: Users are required to register or log in. During login, Google reCAPTCHA is used to ensure that the request is made by a human, and JWT tokens are issued to manage session authentication securely.
- **Dashboard Access**:Admins are redirected to the admin dashboard where they can manage customers, products, and orders.

- **Order Placement**: Admins add new orders by selecting customer and product details. Orders are stored in the MongoDB database via nodeJs, express.

2.6 Security Considerations:

Security is a critical part of any web application, especially when dealing with sensitive business and customer data. The system includes several layers of security to protect against unauthorized access and data breaches:

- **JWT Tokenization**: Upon successful login, the system issues a JSON Web Token that is required for accessing protected routes. This token is securely stored on the client side and verified on every request.
- **Password Hashing**: User passwords are stored using secure hashing algorithms (such as bcrypt) to prevent plain-text storage.
- **Role-Based Access Control (RBAC)**: Different user roles (admin and customer) have different access permissions, ensuring that critical operations can only be performed by authorized personnel.
- **Google reCAPTCHA**: Protects the system from spam bots and brute-force attacks during authentication by verifying that the user is human.
- **CORS and HTTPS Enforcement**: Cross-Origin Resource Sharing (CORS) policies are implemented to prevent unauthorized domain access, and HTTPS is recommended for secure communication.

Chapter 3

System Design

System design is the blueprint that transforms user requirements into a technically feasible and structured software architecture. It defines the overall structure, components, data flow, interfaces, and logic that enable the system to function effectively. In this chapter, we outline the architectural and component-level design of the **Customer Order Management and Invoice Generator** application, developed using the MERN stack and enhanced with TypeScript, JWT authentication, and Google reCAPTCHA. The goal of this design phase is to ensure clarity, maintainability, and scalability before moving into full implementation.

3.1 System Architecture:

The system follows a three-tier architecture composed of the presentation layer, application (logic) layer, and data layer. These layers are modular, allowing for independent development, testing, and scaling:

- **Presentation Layer (Frontend)**: Built using React.js with TypeScript, the frontend is responsible for all user interactions. It displays dynamic forms for customer input, product selection, order review, and invoice previews. This layer communicates with the backend through RESTful APIs and ensures data validation and user feedback.
- **Application Layer (Backend)**: The backend, developed with Node.js and Express.js, handles business logic, routing, API endpoints, and integrates third-party services such as reCAPTCHA. It processes order creation, performs tax calculations, and manages authentication and authorization via JWT tokens.
- **Data Layer (Database)**: MongoDB is used as the NoSQL database for storing user data, customer records, orders, and invoices. The schema-less nature of MongoDB offers flexibility and adaptability for evolving requirements.

3.2 Component Design:

Customer Order management and Invoice generator

The application is broken down into key functional components for both the frontend and backend:

Frontend Components:

- **Authentication Pages**: Includes login and registration forms with Google reCAPTCHA integration.
- **Dashboard**: Separate dashboards for Admin and Customer, displaying relevant metrics and navigation.
- **Customer & Product Forms**: Interfaces for creating and updating records.
- **Order Management Interface**: Allows users to view or manage orders depending on their roles.

Backend Components:

- **User Controller**: Handles authentication, role assignment, and secure token generation using JWT.
- **Order Controller**: Manages order logic, including order creation, status updates, and linking with customers/products.
- **Middleware**: Validates JWT tokens, reCAPTCHA responses, and user roles before granting access to protected routes.

3.3 Data Flow Design:

The data flow within the system occurs through RESTful APIs and follows a request-response pattern:

- User logs in/registers via frontend. The request includes reCAPTCHA verification.
- Backend authenticates the user and returns a JWT token.
- When placing an order, data is sent to the backend, processed, and stored in MongoDB.

3.4 User Interface Design:

- **Navigation Menus** tailored to user roles.
- **Forms with real-time validation**, ensuring data accuracy before submission.
- **Dashboard Visuals** to help admins manage and track orders at a glance.
- **PDF invoice layout** that is structured, professional, and consistent with business standards.

3.5 Security Design:

- **JWT Authentication**: Stateless and secure session management for route protection.
- **Role-Based Access Control**: Only authorized users can perform restricted actions like modifying products or generating invoices.
- **Google reCAPTCHA**: Prevents bots and unauthorized automated access.
- **Input Validation**: Both server and client-side checks prevent XSS and injection attacks.
- **Password Hashing**: Ensures secure credential storage using hashing algorithms like bcrypt.

Chapter 4

Project Implementation

Project implementation is the phase where the planned system design is translated into actual working software through code, configuration, and integration. This chapter presents the step-by-step execution of the Customer Order Management and Invoice Generator system using the MERN stack (MongoDB, Express.js, React.js, Node.js), enhanced with TypeScript, JWT-based authentication, and Google reCAPTCHA.

4.1 Setting Up the Development Environment:

The development environment was initialized by setting up separate workspaces for the frontend and backend using Node.js package management tools. Tools used include:

- VS Code as the primary code editor.
- Node.js & npm for managing dependencies.
- TypeScript for adding type safety and enhanced developer tooling.
- MongoDB Atlas as a cloud-hosted database solution.
- Postman for API testing and verification during backend development.
- Git and GitHub for version control and collaboration.

4.2 Backend Implementation with Node.js and Express:

The backend serves as the core of the application, managing business logic, database interactions, and API routing. Key steps in backend implementation include:

- **Project Initialization**: Express was used to scaffold the server. Required middleware like `cors`, `dotenv`, and `body-parser` was integrated.
- **Database Connectivity**: MongoDB was connected using the Mongoose ODM to manage collections and schemas for users, customers, products, orders, and invoices.
- **API Development**: RESTful APIs were created for CRUD operations across all entities. Separate controllers and routes were built for:
  - User registration and login

- Customer management
- Product management
- Order creation and retrieval
- Invoice generation and PDF export
-

- **JWT Token Authentication**: User authentication was implemented using JWT. Upon successful login, tokens are issued and used to access protected routes.
- **Google reCAPTCHA**: Integrated into login and signup endpoints to prevent bot activity.

4.3 Frontend Implementation with React and TypeScript:

The frontend was implemented using React along with TypeScript to improve code robustness and catch errors during development. Key components and features include:

- **Authentication Pages**: Signup and login forms connected to the backend via API calls. reCAPTCHA v2 was embedded to validate human users.
- **Role-Based Routing**: React Router was used to differentiate between Admin and Customer views, ensuring restricted pages are protected based on user role.
- **Customer & Product Dashboards**: Separate views and forms were created to manage customer details and product inventories.
- **Order Management**: Admin users can create new orders by selecting customers and products. Orders are submitted through API and stored in the database.

4.4 Security and Validation:

Security features were thoroughly implemented to safeguard user data and ensure proper access control:

- **JWT Middleware**: Custom middleware verifies token validity for each request to sensitive API routes.
- **Form Validation**: Both client-side and server-side validation ensures that data such as emails, passwords, and numeric inputs are formatted correctly.

- **ReCAPTCHA Validation**: Login and registration forms are protected using Google reCAPTCHA, reducing the risk of automated attacks.
- **Password Hashing**: All passwords are hashed using `bcrypt` before being stored in MongoDB to prevent plain-text exposure.

4.5 Testing and Debugging:

Throughout the development process, testing was conducted to validate correctness and usability:

- **Unit Testing**: Functions responsible for tax calculations, JWT validation, and data formatting were tested in isolation.
- **Manual Testing**: The complete flow from user login to invoice download was manually tested for various scenarios and roles.
- **Error Handling**: Proper error messages were returned from the backend and displayed on the frontend to inform users of issues like login failures, validation errors, or API issues.

4.6 Project layout and UI:

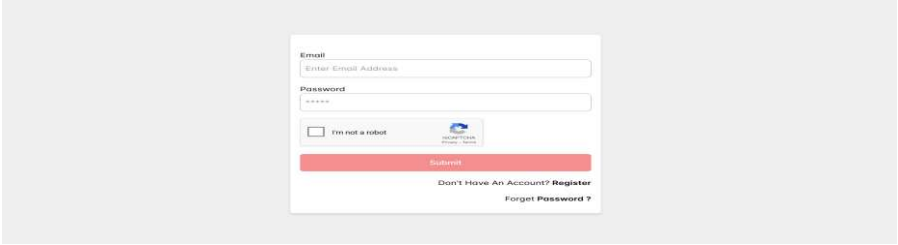

Figure 4.6.1 login page with captcha

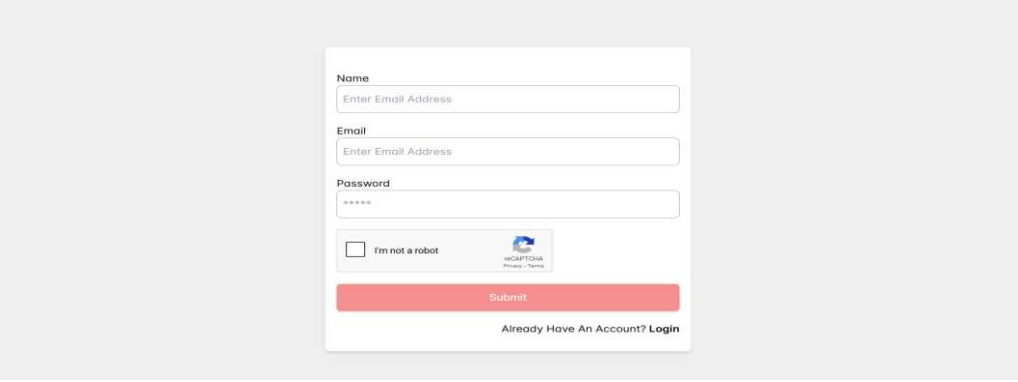

Figure 4.6.2 signup page

Figure 4.6.3 Order summary page

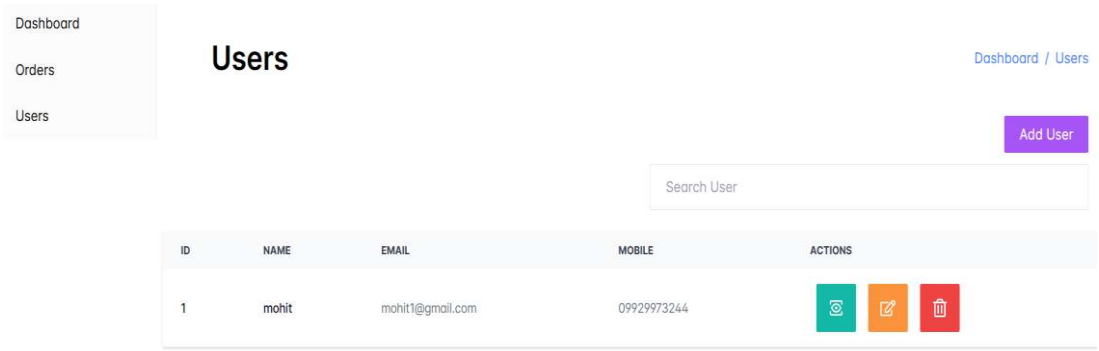

Figure 4.6.4 users detail page

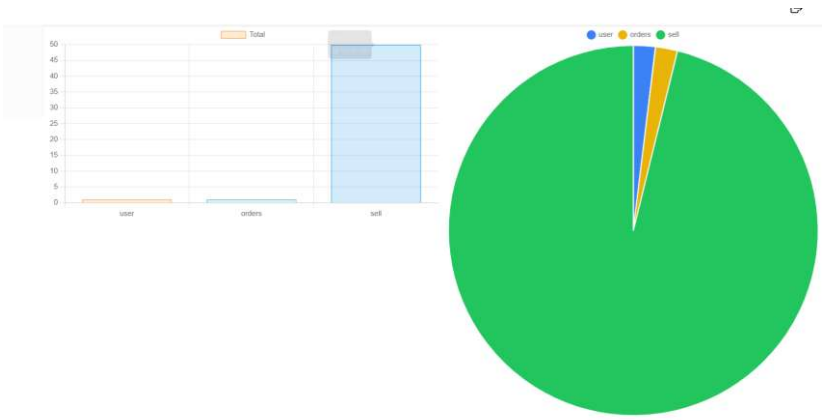

Figure 4.6.5 Pie Chart of user, order ,sale

Figure 4.6.6 code of backend/server.js

Figure 4.6.7 code of backend/app.js

Chapter 5

Future Work & Conclusion

While the system in its current form meets the core requirements of customer, order, and invoice management, there are numerous opportunities to enhance its functionality, user experience, and scalability. The following points highlight possible directions for future development:

- **Multi-Currency and Tax Region Support:**
  Introducing dynamic currency conversion and support for region-specific tax rules (like GST, VAT, etc.) would allow the system to serve international clients and comply with localized financial regulations.

- **Email and SMS Notifications:**
  Automating communication through email or SMS for actions like order confirmation, invoice delivery, due date reminders, and account alerts would greatly improve customer engagement and operational efficiency.

- **Payment Gateway Integration:**
  Integrating secure payment gateways such as Razorpay, Stripe, or PayPal will enable customers to pay their invoices directly through the platform, turning it into a complete transaction processing solution.

- **Mobile Application Development:**
  A dedicated mobile app built using React Native or Flutter would allow customers and administrators to manage orders and invoices from anywhere, improving accessibility and convenience.

- **Advanced Analytics dashboard:**
  A visual analytics panel with graphs, charts, and reports could provide insights into sales trends, customer behavior, and product performance, enabling better decision-making.

- **Role expansion and audit logs:**
  Introducing additional user roles (e.g., sales executive, accountant) and detailed audit logging can improve accountability, traceability, and internal control in multi-user environments.

- **Offline Mode support:**
  Implementing offline-first capabilities using Progressive Web App (PWA) techniques would allow users to create and view orders or invoices even without internet connectivity, syncing automatically when back online.

- **Multi company support:**
  Enabling the system to handle data for multiple businesses or departments under a single login would expand its use cases, particularly for accounting firms or franchises.

- **Recurruring Invoices and subscriptions:**
  Adding support for recurring invoices would be beneficial for businesses with subscription-based models. The system could automatically generate and send invoices on scheduled intervals.

- **Inventory management module:**
  Expanding the system to track stock levels, manage suppliers, and receive low-stock alerts would provide comprehensive product management capabilities.

- **Integration with accounting software:**
  Linking with third-party accounting tools like QuickBooks, Zoho Books, or Tally could help sync invoices and transaction data for easier bookkeeping.

- **Localization and Multilingual support:**

  Supporting multiple languages and regional settings would improve accessibility for non-English speaking users and broaden the system's usability in different countries.

Conclusion:

The Customer Order Management and Invoice Generator project successfully demonstrates the development of a modern, secure, and scalable full-stack web application using the MERN stack and TypeScript. By integrating robust features such as JWT-based authentication, Google reCAPTCHA, and automated tax-inclusive invoice generation, the system effectively meets the needs of businesses that require streamlined order tracking and billing solutions.

Through a modular and layered architecture, the system ensures maintainability, scalability, and high usability. Features like dynamic invoice creation, role-based access control, and secure authentication workflows contribute to the reliability and professionalism of the platform.

This project not only reflects technical competence in full-stack development but also lays the groundwork for future enhancements that can expand its functionality into a complete business management tool. It serves as a strong foundation for real-world application and continued innovation in cloud-based enterprise systems.

References

- React Documentation. https://reactjs.org/docs/getting-started.html
- Node.js Documentation. https://nodejs.org/en/docs
- Express.js Guide. https://expressjs.com/en/starter/installing.html
- MongoDB Official Documentation. https://www.mongodb.com/docs/manual/
- TypeScript Documentation. https://www.typescriptlang.org/docs/
- JWT (JSON Web Token) Guide. Auth0. https://auth0.com/docs/secure/tokens/json-web-tokens
- Google reCAPTCHA Documentation. https://developers.google.com/recaptcha
- jsPDF – HTML to PDF Converter. https://github.com/parallax/jsPDF
- Mozilla Developer Network (MDN) Web Docs. https://developer.mozilla.org/en-US/
- GitHub – Version Control for Project Codebase. https://github.com
- Stack Overflow – Community Discussions and Troubleshooting. https://stackoverflow.com
- Postman API Platform. https://www.postman.com/