Faculty of Engineering and Technology
Computer Science Department

INTRODUCTION TO NON-STRUCTURED DATABASE
COMP 438
Project Description OF Inventory Management System

Group Members:

Name	ID
Haneen Abu El-Hawa	1210711
Ibraheem Abu Hijleh	1203065
Nour Alaydi	1222810
Yuna Nawahda	1211524

Under Supervision:Dr.Rashid Jayousi

Abstract

The MERN stack—MongoDB, Express, React, and Node.js—was used in the design and development of this system, which is a complete inventory and sales management platform. It is designed to ensure a user-friendly interface and strong functionality while streamlining the management of goods, sellers, sales, and purchases. Modules for user profile administration, seller management, product inventory, user authentication, and sales tracking are all included in the system. Additionally, the system includes a client-facing interface, enabling shoppers to browse available products, manage their carts, view past orders, and access support through the About Us and Contact Us pages. The platform's sleek and contemporary design offers real-time validation, feedback, and thorough tracking of all crucial business operations. By utilizing the MERN stack, the system provides excellent performance, scalability, and flexibility to accommodate a range of business needs.

Table of Contents

1. System Overview

Businesses may effectively manage their products, sellers, purchases, and sales with the help of this inventory and sales management tool.

The client-side interface is designed to provide an intuitive shopping experience. Shoppers can browse the inventory through a visually appealing dashboard, add items to their cart, view their orders, and manage their profile details. The inclusion of an 'About Us' and 'Contact Us' section ensures transparency and accessibility for user support.

The MERN stack, which is used in its construction, has the following benefits:

- ★ **MongoDB**: Structured data, such as user profiles, products, sales, purchases, and client-side data such as orders and carts are stored in a NoSQL database.
- ★ **Express.js**: a backend web framework that manages server-side functionality and API routing.
- ★ **React.js**: A frontend library for creating a responsive and dynamic user interface.
- ★ **Node.js**: an environment for the JavaScript runtime that guarantees scalable and effective server-side functions.

2.Executive Summary

The goal of this project is to employ the MERN stack—MongoDB, Express, React, and Node.js—to create a stable and intuitive inventory management system. By providing a complete solution, the system tackles the difficulties that companies encounter while managing their suppliers, sales, purchases, and inventory.

- Secure password management along with user authentication is one of the main benefits.
- inventory and sales tracking in real time.
- The dashboard is dynamic and shows key performance indicators (KPIs).
- CRUD activities for buyers, sellers, and goods.
- A client-facing interface for shoppers, allowing them to browse inventory, manage their carts, track orders, and update their profiles.
- Accessible "About Us" and "Contact Us" pages to facilitate customer engagement.

The project makes use of contemporary technologies such as Node.js for effective backend processing, React for an interactive user interface, and MongoDB for scalable data storage. For small and medium-sized businesses, this system is made to streamline operations, increase productivity, and facilitate scalability.

3.Introduction

➢ Objectives

This project's goal is to simplify inventory and sales management procedures for small and medium-sized enterprises by offering a centralised platform that will decrease manual labour, increase data accuracy, and improve decision-making capabilities.

➢ Scope

- Inclusions:

  Inventory management (add, update, and delete products), Sales and purchase tracking, User authentication and profile management, Real-time analytics dashboard,

- Exclusions:

  Integration with external accounting systems,

➢ Significance

Inventory management is an essential component of any business, ineffective systems frequently result in stockouts, overstocking, and financial losses, this project solves these issues by providing a scalable, user-friendly, and secure solution that frees businesses to concentrate on expansion rather than administrative duties.

4.Requirements Analysis

➢ Problem Statement

Effectively managing supplier data, sales, and inventory is a common challenge for small firms. Conventional approaches, such as spreadsheets or paper records, are error-prone and don't provide real-time insights. Additionally, small businesses often lack a dedicated platform to engage with customers, resulting in inefficiencies in handling orders, profiles, and inquiries. The goal of this project is to provide a centralized, automated solution to these problems, benefiting both business operations and customer experience.

➢ Requirements Gathering

Functional Requirements:

1) Role management and user authentication for both administrators and customers.
2) CRUD activities for purchases, sales, items, and customer orders.
3) A dynamic dashboard showing KPIs in real time.
4) Client-side features, including a home page with inventory, cart management, order tracking, and user profiles.
5) "About Us" and "Contact Us" pages for improved customer interaction.
6) Search and filtering capabilities across all data entities, including inventory and orders.

Non-Functional Requirements:

1. Performance: Quick database query and user interface update response times.
2. Scalability: The capacity to manage expanding user bases and data.
3. Security: JWT and bcrypt are used for secure authentication.
4. Reliability: continuous operations with 99.9% uptime

➢ Target Users

The system is intended for small and medium-sized enterprises that want an easy-to-use, effective method of managing their sales and inventory information. Additionally, it serves individual customers by providing a user-friendly platform to browse products, manage orders, and interact with the business.

5.Implementation

Technologies Used

1. Frontend: React.js (with state management hooks and the Context API , and dynamic routing using React Router for smooth navigation pages).
2. Backend: Express.js and Node.js.
3. MongoDB with Mongoose ORM is the database.
4. JWT and bcrypt are used for authentication.

System Architecture

- Frontend: React-built to create interactive and user-friendly elements for both admin and client interfaces
- Backend: APIs made to manage CRUD tasks.
- Database: MongoDB-based non-relational schema that is tuned for quick queries.

6.Testing

Testing Strategy

Functionality and performance were verified through a combination of automated and manual testing. API testing was done with tools like Postman, while React component unit testing was done with Jest.

Test Cases

Test case ID	Description	Input	Expected Output	Actual Output	status	client/admin page
TC 001	User login functionality	Email, Password	Login success	Login success	Passed	admin
TC 002	Add product functionality	Product details	Product added	Product added	Passed	admin
TC 003	Add item to cart	Product details , quantity	Item added to cart	Item added to cart	passed	Client
TC 004	Place order	Car items ,address	Order placed	Order placed	Passed	Client
TC 005	View product detail	Product id	Product detail displayed	Product detail displayed	Passed	Client
TC 006	Search for product	Search query	Matching products displayed	Matching products displayed	Passed	Client

Results

- Every significant feature passed testing.
- Small errors in the validation logic were found and fixed.

7.Challenges and Solutions

Challenges:

1. Dashboard updated in real time: avoiding page reloads and achieving smooth updates.
2. Secure authentication: Using strong hashing to protect user data.
3. Handling Product Inventory Updates: Real-time updates for inventory management, including adding or removing products, required careful synchronization to prevent stock inconsistencies.

Solutions:

1. State management in React was put into practice for real-time changes.
2. used JWT for safe authentication and bcrypt for password hashing.
3. Role based control to ensure different user roles (admin, customer) only have access to relevant sections of the system.
4. For accurate product availability and real-time inventory updates, RESTful APIs were designed to handle add, update, and delete operations efficiently

8.System Design

Schema Diagram

Include a database schema diagram showing tables for:

- Users
- Products
- Sales
- Purchases
- Sellers
- Brands
- Categories
- Cart
- Orders
- Messages
- Contacts

System Architecture

Diagram illustrating:

- Frontend: UI/UX using React.js.
- Backend: Express.js and Node.js for developing APIs.
- Database: MongoDB for storing information.
- Communication: Frontend and backend data transfer via RESTful APIs.

9. Features of the System

9.1 Login and Registration

★ **Login Page**:
- Users can use their password and email to authenticate.
- Secure authentication with JWT and bcrypt tokens is ensured by integration with the backend.
- If the user that will login is not registered on the system it will display an alert message

★ **Registration Page**:
- By entering their name, email address,phone number ,and password, new users can register.
- Strong user validation is ensured by password confirmation.
- User information is stored in MongoDB via communication between the Express backend and the React frontend.
- If the user already registered it shows alert .

9.2 Admin Dashboard

★ **Purpose**: provides a summary of important indicators including income, total inventory, and items sold represented on graphs and clear way .

★ **Features**:
- Real-time statistic changes through integration with MongoDB.
- React.js was used to create a dynamic user interface that shows key performance indicators (KPIs).
- Node.js and Express.js-designed backend APIs effectively retrieve pertinent data.
- A graphs implemented to give real user experience.

9.3 Add New Product to database

- ★ **Features**:
    - Product details, including name, price, stock, description, and size, are entered into form fields.
    - drop-down menus that are dynamically updated from the database to choose a seller, category, and brand.
    - Validation logic makes sure that required fields are filled up.
    - RESTful APIs for inserting data in real time into MongoDB.
    - Display alert for the successful or failed actions .

9.4 Manage Products

- ★ **Purpose**: gives users the ability to manage, search, and filter product inventory also to sell a products , add the products to the stock or delete the product also to update too.
- ★ **Features**:
    - To maximize performance, price range, category, and brand filters are applied to both the frontend and backend.
    - The text indexing capabilities of MongoDB enable search functionality.
    - React manages product list dynamic updates without requiring a page reload.
    - If the user want to add product to the stock it asks for the quantity.
    - For the delete functionality it will ask if the user sure he want to delete the item.
    - For the update functionality it will ask for the all fields to be updated and sure to be
    - For the add to stock feature it will ask the user to enter the buyer name and choose the date from date picker and specify the quantity of the items.

9.5 Manage Sales

- ★ **Purpose**: offers a well-organized interface for monitoring and controlling sales transactions by editing or deleting the sales.
- ★ **Features**:
    - a searchable list of sold items that includes information on the buyer, the quantity, and the total cost.
    - Accurate sales tracking is ensured by the MongoDB database's real-time changes.
    - Sales data retrieval is effectively managed by backend APIs.

- For the delete functionality it will ask if the user sure he want to delete the sales.

9.6 Manage Sellers

★ **Purpose**: Allows users to maintain a database of product sellers by delete or edit the sellers .

★ **Features**:

- contains spaces for the seller's name, email address, and phone number.
- To display and update seller details, the React frontend interfaces with Express APIs.
- MongoDB provides a reliable repository for all seller data.
- For the delete functionality it will ask if the user sure he want to delete the salers.

9.7 Manage Purchases

★ **Purpose**: Tracks purchases made from sellers by deleting or editing the purchases .

★ **Features**:

- shows the product name, seller name, quantity, price per unit, and total cost of the purchase.
- MongoDB ensures scalability for expanding enterprises by storing purchase history.
- RESTful APIs make it easy to retrieve and change data.
- For the delete functionality it will ask if the user sure he want to delete the purchases.

9.8 User Profile

★ **Purpose**: Allows users to view and edit their account details.

★ **Features**:

- Users can change their social media links, email address, name, and address.
- Real-time profile updates are dynamically rendered by React.js.
- Bcrypt hashing is used to secure password management.
- MongoDB makes ensuring that user information is stored securely.
- Provide the go back button to return to the original page .

9.9 Change Password

- ★ **Purpose**: Enhances account security by allowing users to update their passwords.
- ★ **Features**:
    - The current password must be used for verification when changing a password.
    - Both the backend (Node.js) and frontend (React.js) have robust validation logic in place.
    - Secure user credential changes are ensured by backend integration with MongoDB.
    - Provide the go back button to return to the original page .

9.10 Client Home page :

- ★ Displays the available inventory categorized by type, brand, or other attributes.
- ★ Provides promotional banners or featured products, best sellers

---

9.11 Orders:

- ★ Clients can view their complete purchase history, with details such as the order date, items purchased, quantities, and total cost. This history is stored securely in the database and is available for future reference.
- ★ To enhance the user experience, users can filter order date

---

9.12 Cart Management:

- ★ Clients have the ability to easily add or remove items from their shopping cart.
- ★ Displays the total cost dynamically as items are added or removed.

9.13 About Us/Contact Us:

- Offers details about the company and its values in the "About Us" section.

- the contact section includes a form for customers to reach out to the business directly. It consists of fields for name, email, and message, allowing users to submit inquiries, feedback, or issues easily.

9.14 Shop Page:

The Shop Page serves as the main browsing section for users to explore and purchase products. It provides an organized, easy-to-navigate interface with various features designed to enhance the shopping experience:

- Products are displayed in a grid,, with clear images, names, prices, and a brief description for each item.
- Users can easily scroll through product
- The page allows users to filter products based on various attributes
- A dropdown menu provides quick access to filtering options, enhancing usability.
- Clicking on a product leads users to a dedicated product detail page where they can view images, detailed descriptions, specifications, user reviews, and availability.
- Each product has an "Add to Cart" button, enabling users to add items to their shopping cart with a single click.
- A quantity selector allows users to specify how many units of an item they want to purchase.
- A search bar allows users to quickly find specific products by entering keywords

10. Advantages of Using the MERN Stack

- ★ **Scalability**: MongoDB allows for flexible and scalable data storage.
- ★ **Real-Time Interactivity**:Real-time data rendering with React.js guarantees a fluid and dynamic user experience.
- ★ **Unified Codebase**: Using JavaScript throughout the stack improves maintainability and reduces development complexity.
- ★ **High Performance**: Non-blocking I/O from Node.js guarantees quick backend operations.
- ★ **Robust APIs**: Frontend and backend communication is made easier with RESTful APIs created with Express.js.

11.Future Enhancements

1. Include support for several languages.
2. Connect to third-party payment processors such as Stripe or PayPal.
3. Implement advanced data analytics tools for better business insights. Features such as customizable reports on sales trends, inventory turn
4. While the web version would benefit from mobile optimization, developing a dedicated mobile app for iOS and Android could further enhance the customer experience, particularly for users who prefer a native app over a mobile website.

Conclusion

Using the MERN stack, this project effectively provides a complete inventory management system. For small and medium-sized organizations, it enhances user experience, scalability, and efficiency. Throughout the implementation phase, the team acquired invaluable knowledge in full-stack development and overcoming practical obstacles.

THANK U!!!!!