

**System Development Project Design**

Innovation Nexus



**N MBAMBO 22208482**

**BZS KHUMALO 22363276**

**NA NXUMALO 22323727**

**MGABHI NT 21718444**

**MS MTHETHWA 22223731**

**1. Introduction**

**1.1 Purpose**

The purpose of this document is to provide a comprehensive design for the Bright Brightness Shop System, which is intended to streamline and enhance the business operations of the Best Brightness Pongola Shop. This system will address critical business needs, including inventory management, customer management, sales tracking, and e-commerce integration, through a robust web application built using ASP.NET.

**1.2 Scope**

The scope of this system design includes the following:

**Included:**

* Development of a web-based application for inventory management, customer management, and sales tracking.
* Integration of an e-commerce platform to enable online sales.
* Implementation of security measures to protect sensitive data.
* Design of a user-friendly interface for shop administrators and customers.

**Excluded:**

* Development of a mobile application.
* Integration with third-party accounting software.

**1.3 Overview**

This document outlines the system architecture, including hardware, software, and network components. It details the external interfaces, database design, user interface design, and application components. It also considers design aspects such as performance, scalability, security, usability, and dependability.

**2. System Architecture**

**2.1 High-Level Overview**

The system architecture of the Bright Brightness Shop System includes a three-tier architecture:

* **Presentation Layer:** The user interface (UI) components where users interact with the system.
* **Business Logic Layer**: Handles the core functionality, including business rules and operations.
* **Data Access Layer**: Manages interactions with the database, including CRUD operations.

**2.2 Hardware Architecture**

- User Devices: Desktop computers, laptops, tablets, and smartphones used by shop administrators and customers to access the system.

- System Devices: Servers hosting the web application, database, and e-commerce platform.

**2.3 Software Architecture**

**User Applications:**

* Web browsers on user devices for accessing the web application.

**- System Applications:**

* ASP.NET Core web application hosting the business logic and UI.
* SQL Server for managing the database.
* IIS (Internet Information Services) as the web server.

**2.4 Network Architecture**

**- Network Infrastructure:**

* The system components communicate over a secure HTTPS connection.
* The internal network uses Ethernet or Wi-Fi to connect servers and user devices.
* Internet access for remote users and online sales transactions.

**- Network Protocols:**

* HTTP/HTTPS for web communications.
* TCP/IP for internal networking.
* SSL/TLS for secure communications.

**3. External Interfaces**

**3.1 Hardware Interfaces**

* Interface with User Devices: Standard interfaces like USB, Ethernet, and Wi-Fi for connecting user devices to the network.
* Interface with System Devices: Servers connected via Ethernet to ensure fast and reliable communication.

**3.2 Software Interfaces**

- Web API: The web application may expose APIs for future integration with external systems.

- Database Interfaces: The application will interface with the SQL Server database using Entity Framework Core.

**3.3 Communication Interfaces**

1. Internal Communication: ASP.NET Core communicates with SQL Server using Entity Framework Core.
2. External Communication: The system communicates with user devices via HTTPS, ensuring data is encrypted during transmission.

**3.4 System Security and Integrity Controls**

* 1. User Authentication: ASP.NET Identity for managing user authentication and authorization.
  2. Data Encryption: SSL/TLS for encrypting data in transit and AES for data at rest.
  3. Input Validation: To prevent SQL injection and other security threats.

**4. Database Design**

**4.1 Data Requirements**

**- Data Dictionary:**

* Tables: Products, Sales, Customers, Users, Orders, Reviews, Stock Requests, Suppliers, Financial Reports.
* Fields: For Product – ProductId, ProductName, ProductCategory, QuantityInStore, QuantityInWarehouse, ProductPrice, ProductPrice, RestockLevel, SupplierId, DateAdded.

For Sales – SaleId, ProductId, CustomerId, QuantitySold, TotalPrice, SaleDate, DiscountApplied, PaymentMethod, SaleType.

For Customers – CustomerId, CustomerFirstName, CustomerLastName, CustomerEmail, CustomerPhoneNumber, CustomerAddress, CustomerLoyaltyPoints, AccountCreated.

Users – UserId, UserName, UserPasswordHash, UserFirstname, UserLastName, UserEmail, UserRole, DateCreated, IsActive.

Orders – OrderId, CustomerId, OrderDate, OrderStatus, OrderTotalAmount, OrderPaymentStatus, ShippingAddress.

Reviews – ReviewId, ProductId, CustomerId, Rating, Comment, ReviewDate.

Stock Requests – RequestId, ProductId, RequestedBy, QuantityRequested, RequestDate, RequestStatus.

Suppliers – SupplierId, SupplierName, ContactPerson, SupplierPhoneNumber, SupplierEmail, SupplierAddress.

FinancialReport – ReportId, ReportType, ReportStartDate, ReportEndDate, ReportTotalSales, ReportTotalDiscounts, ReportNetProfit, GeneratedBy.

**4.2 Database Schema**

**- Tables and Relationships:**

* Products table with ProductId as the primary key.
* Sales table linked to Products and Users via foreign keys.
* Customers table with a one-to-many relationship with Sales.

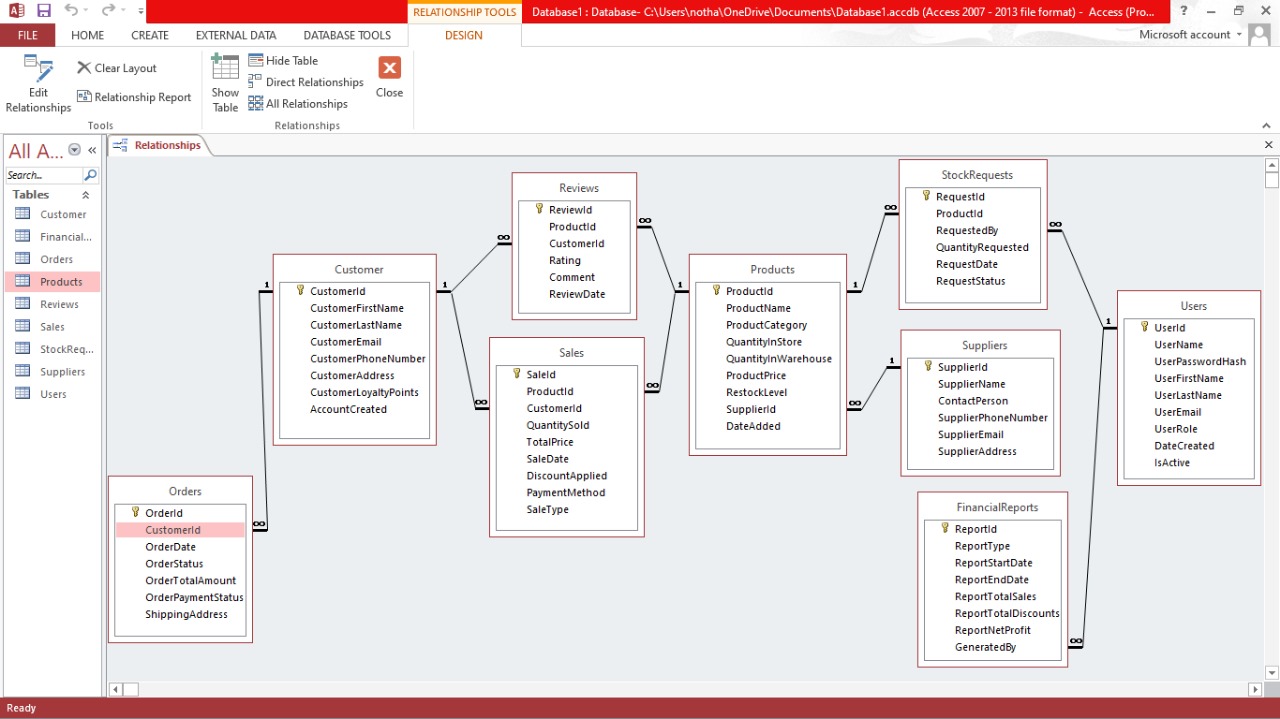
**4.3 Database Physical Design**

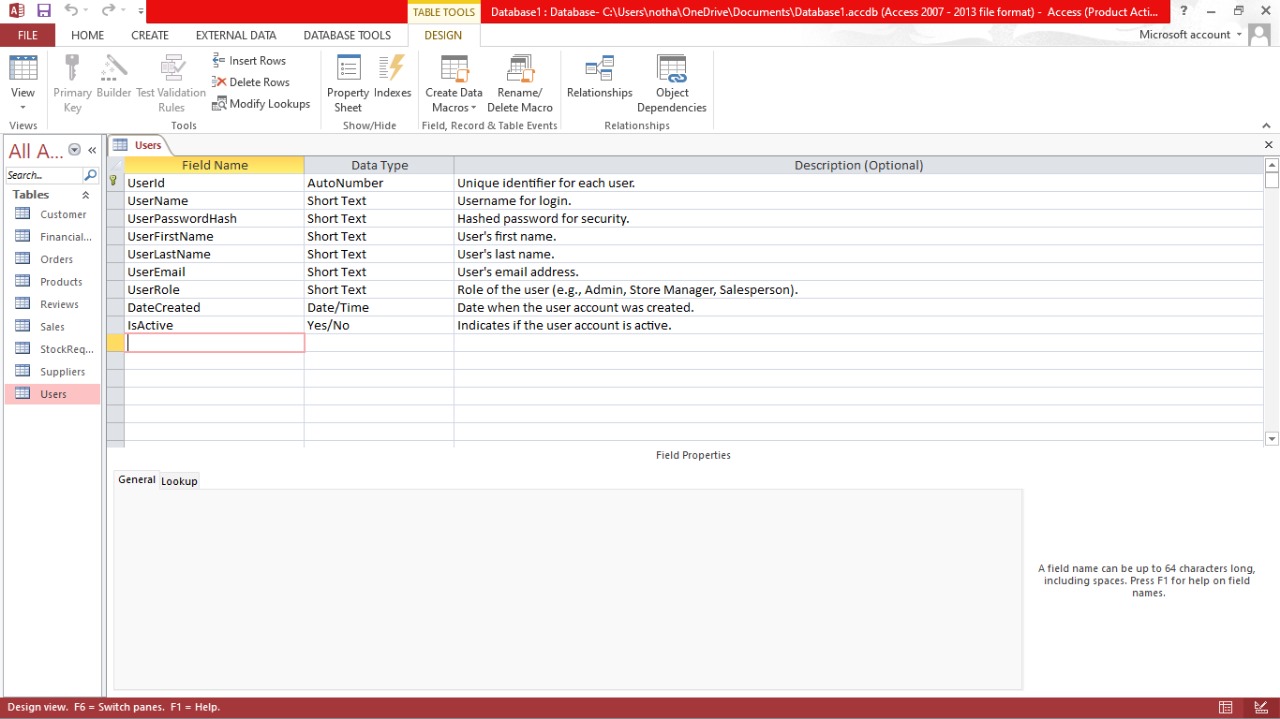
* + Data Storage: SQL Server database hosted on a dedicated server.
  + Storage Size: Adequate to accommodate the shop's growing data, including product details, sales records, and customer information.

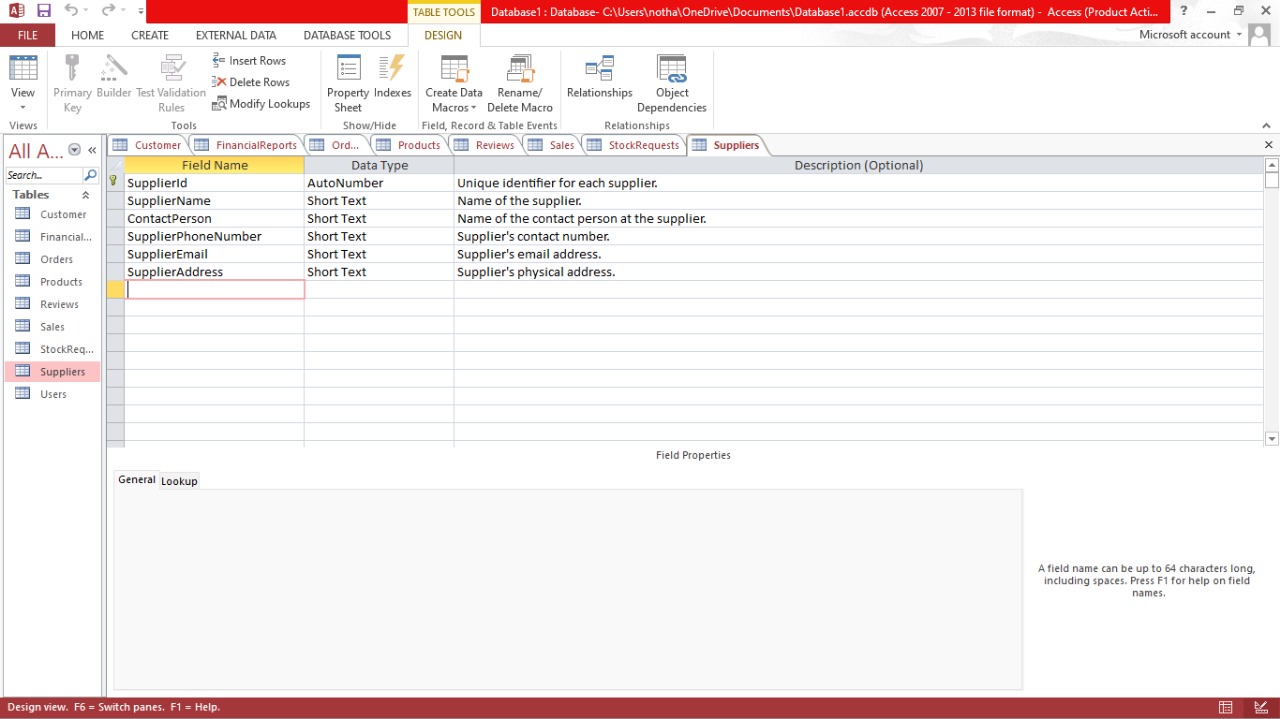
**4.4 Data Security**

**Security Measures:**

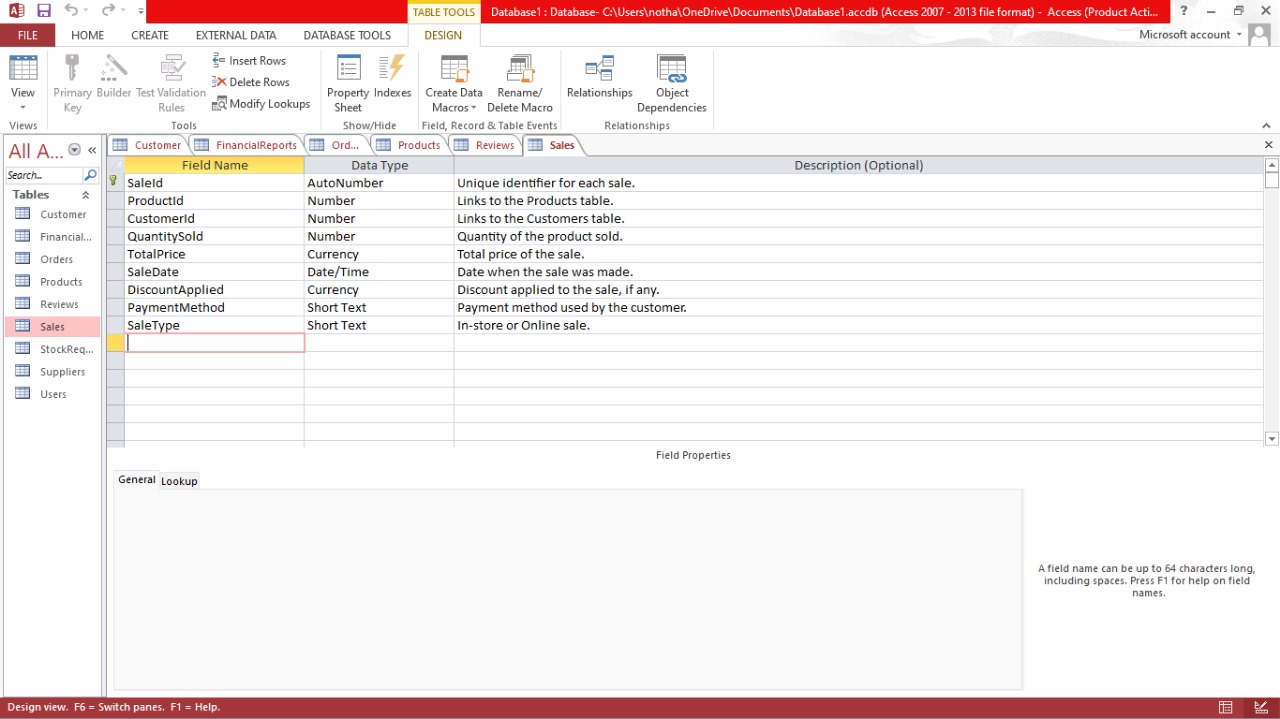
* + Role-based access control to limit access to sensitive data.
  + Regular backups to prevent data loss.
  + Encryption of sensitive fields like passwords and payment information.

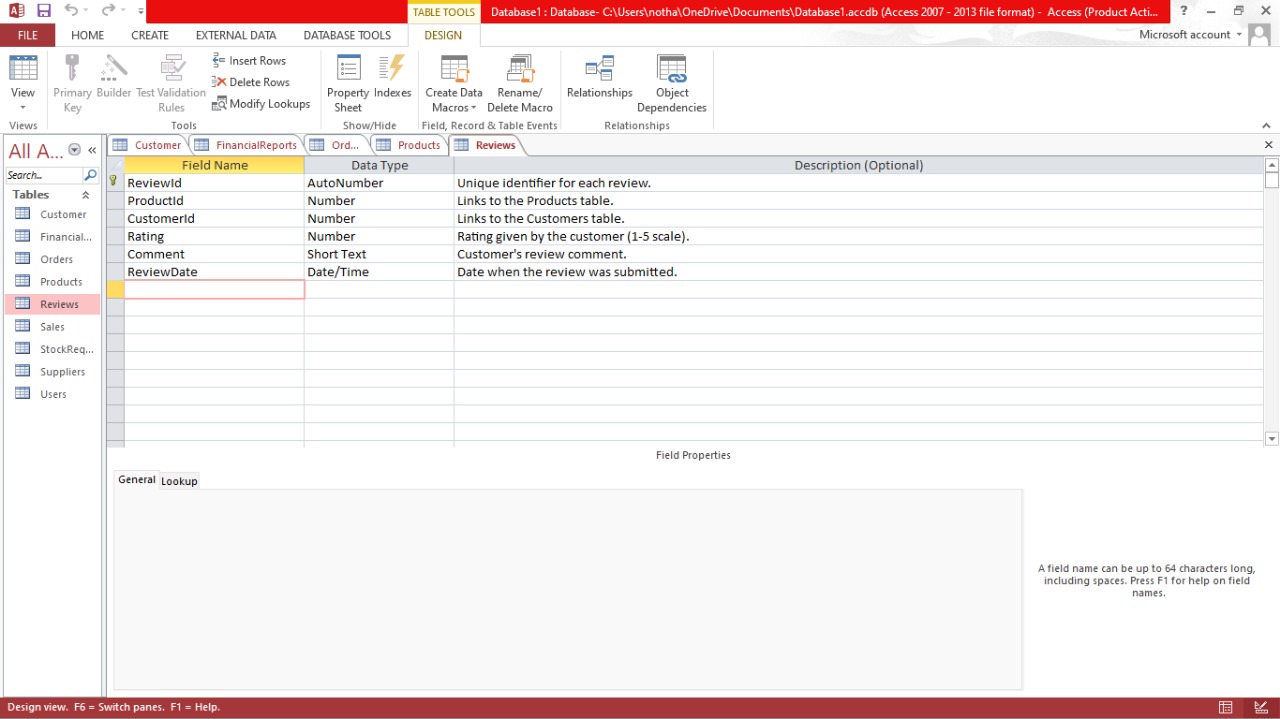


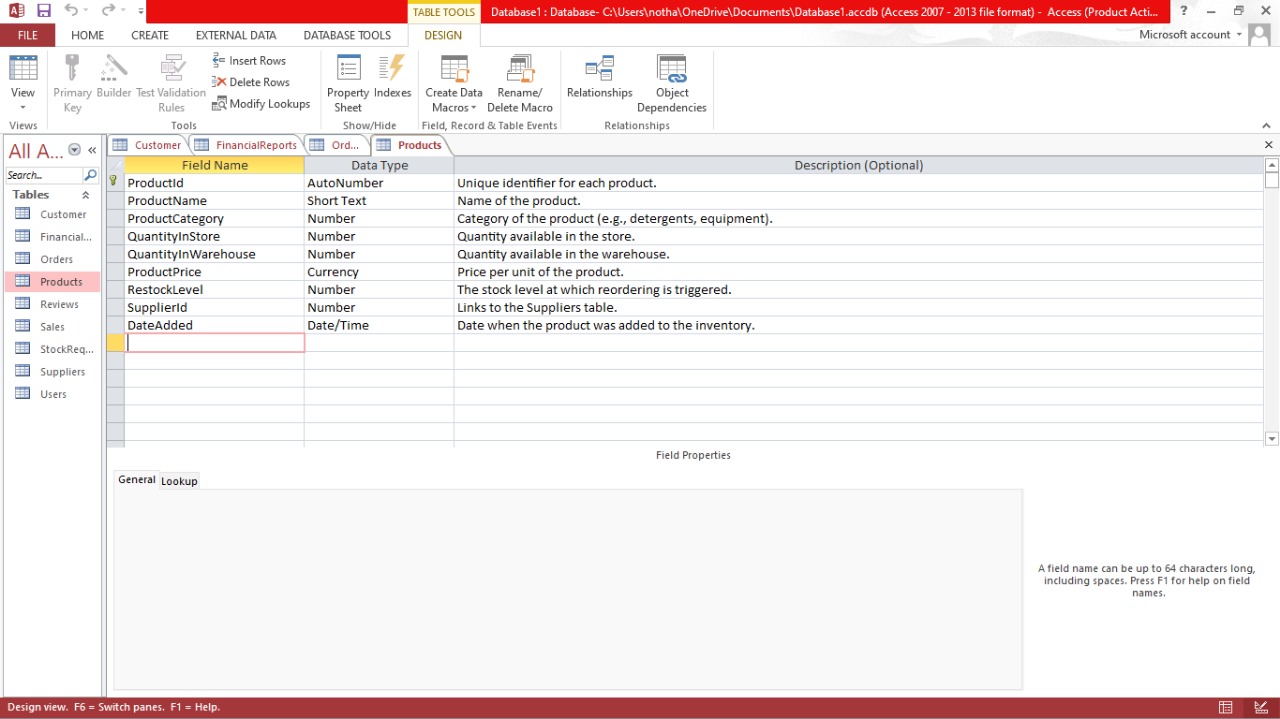


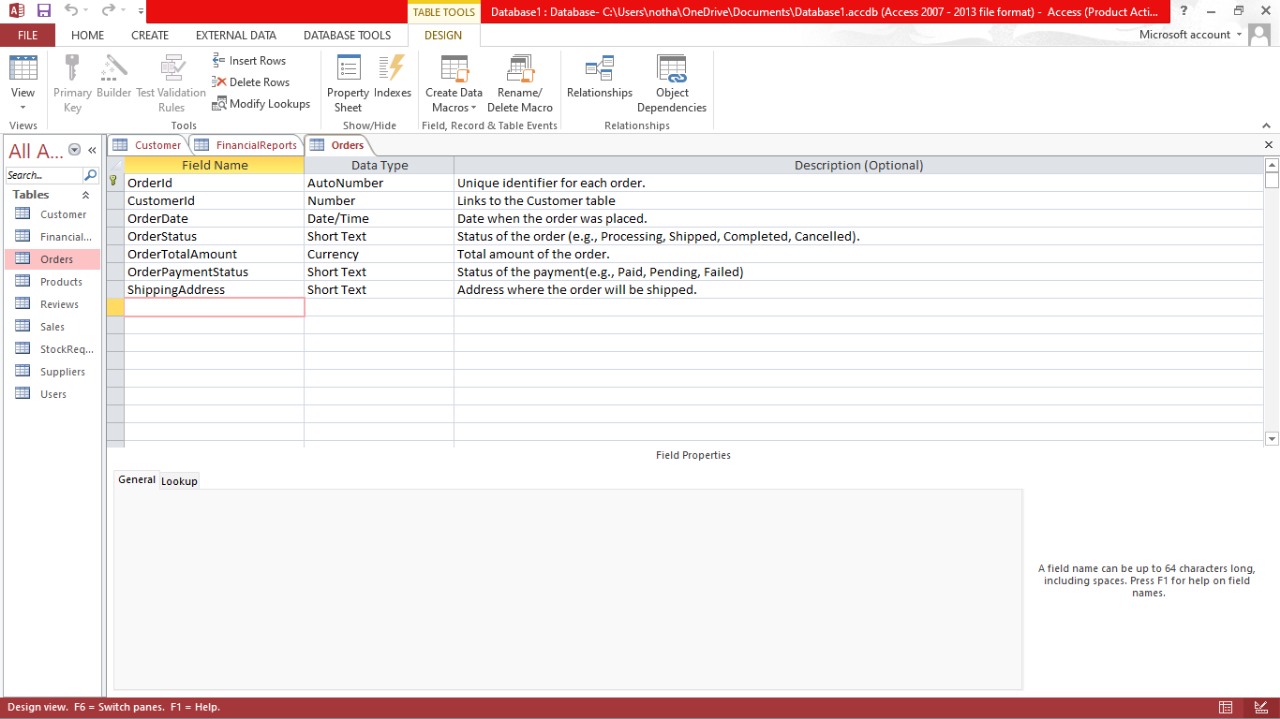
****

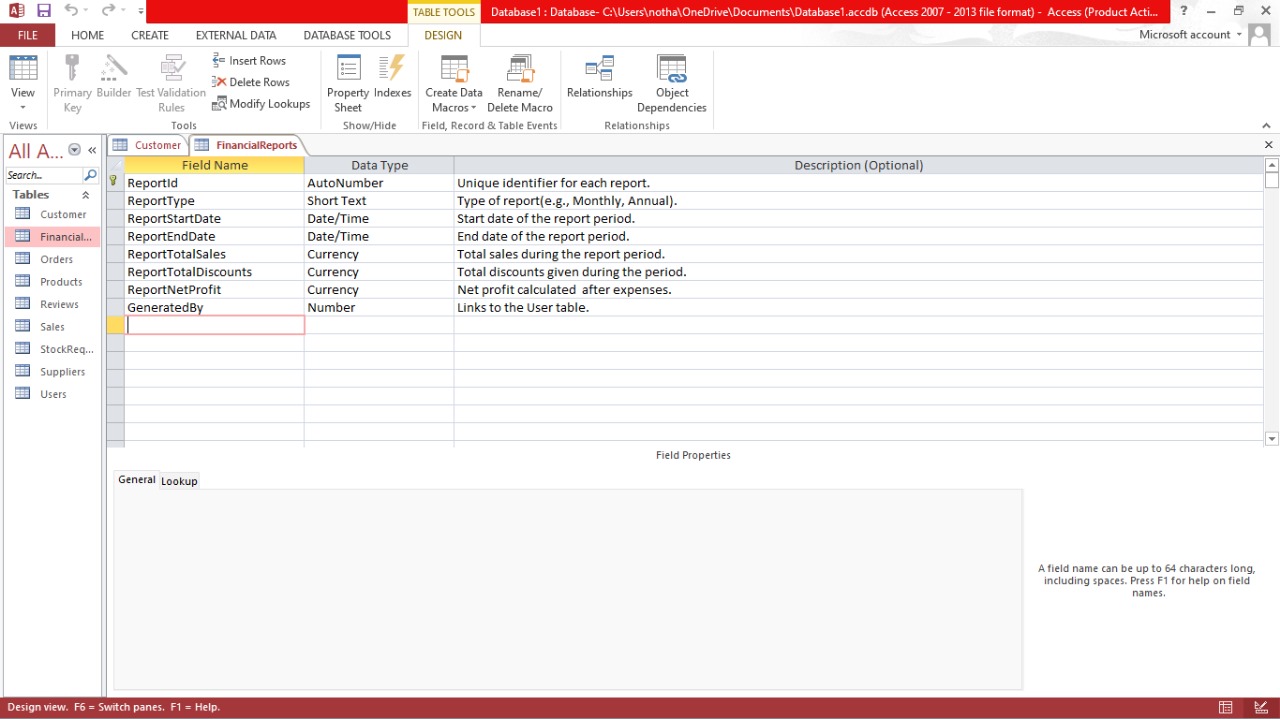
****

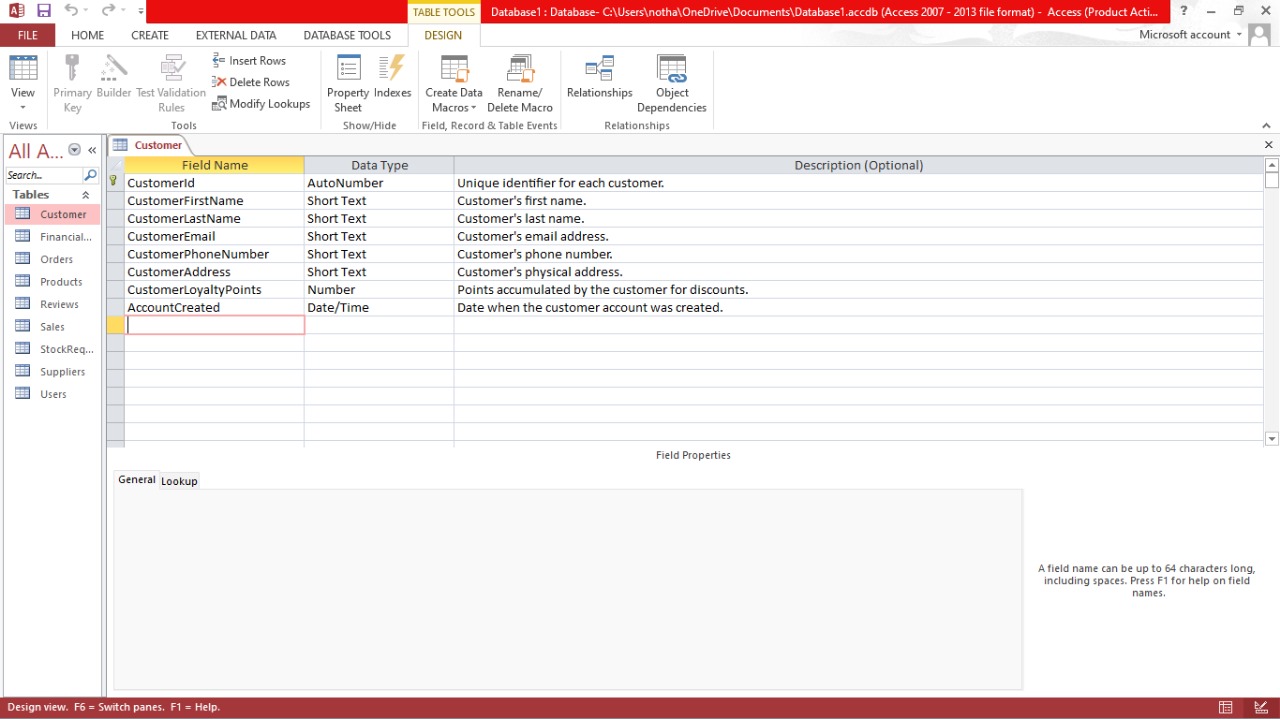
****

****

****

****

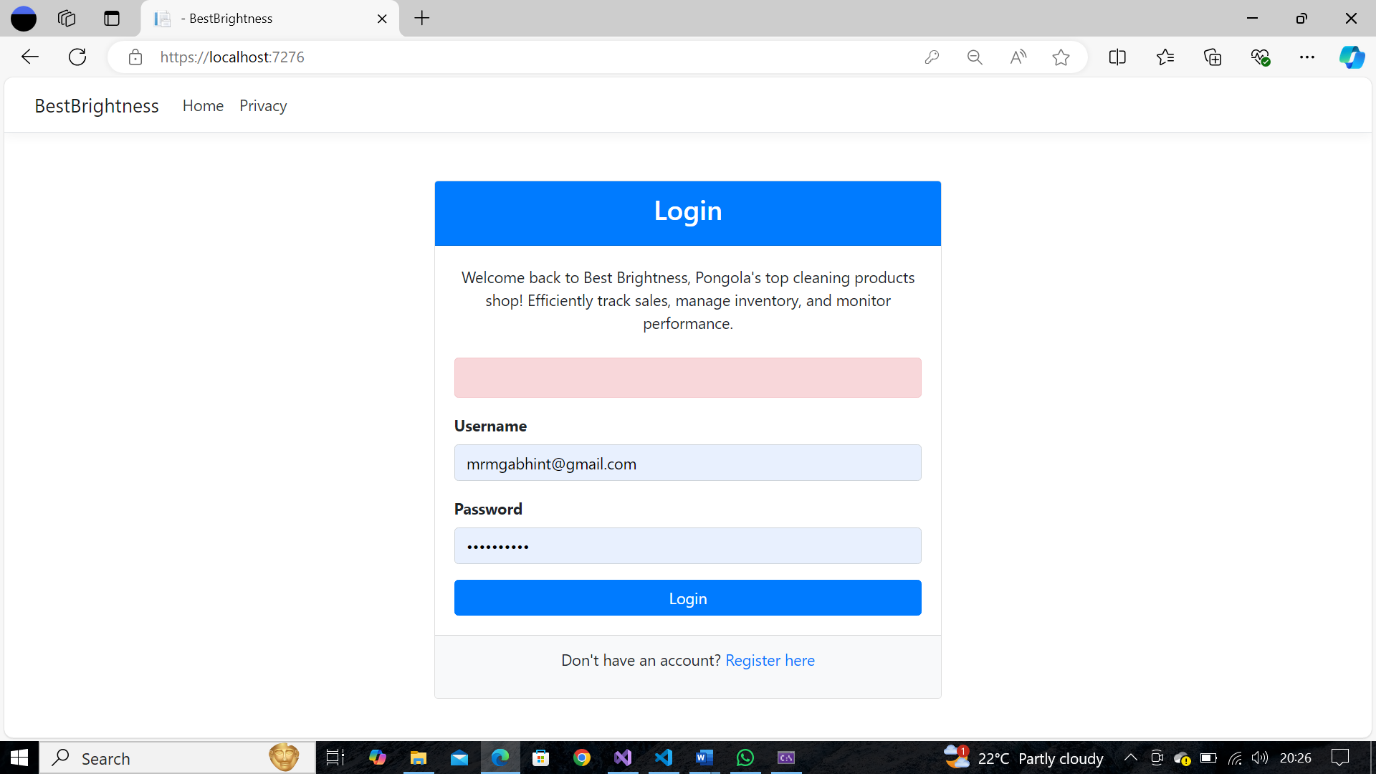
****

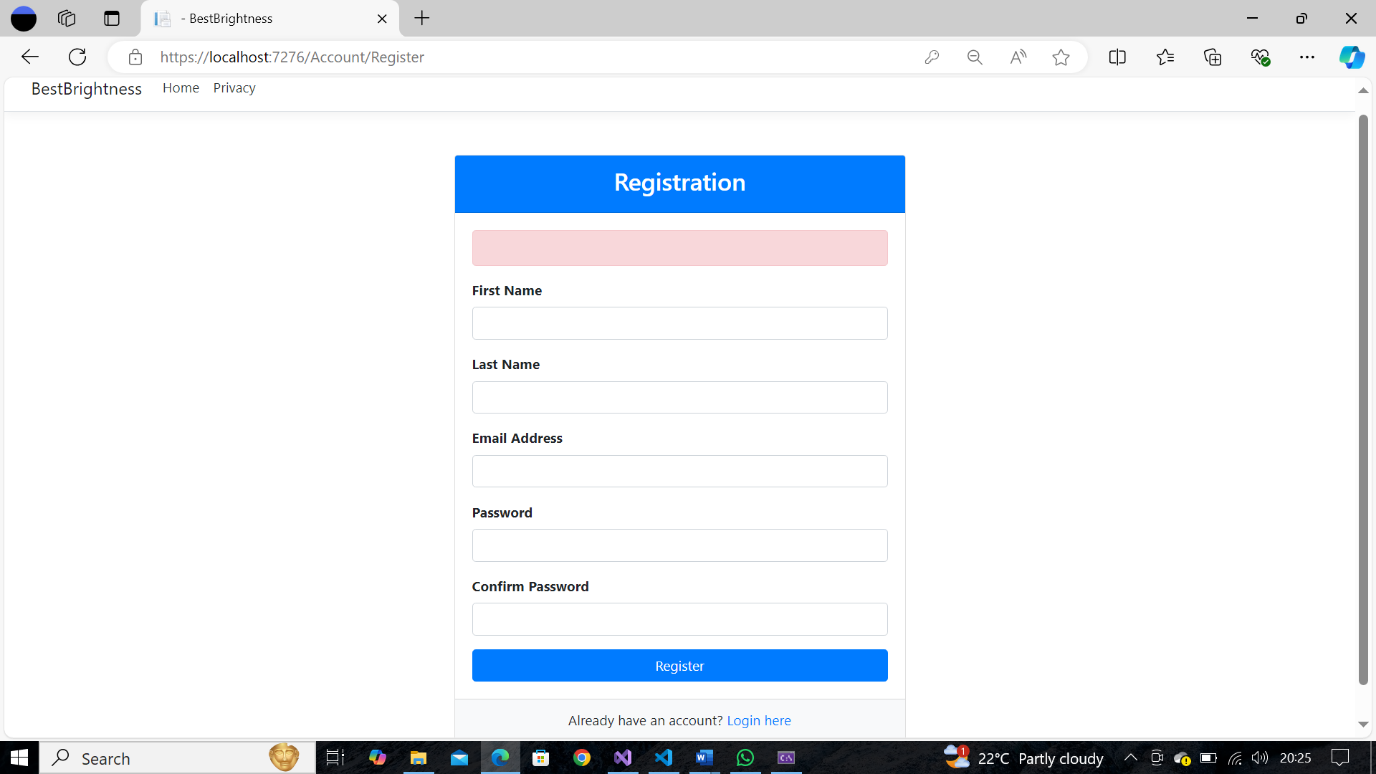
****

**5. User Interface Design**

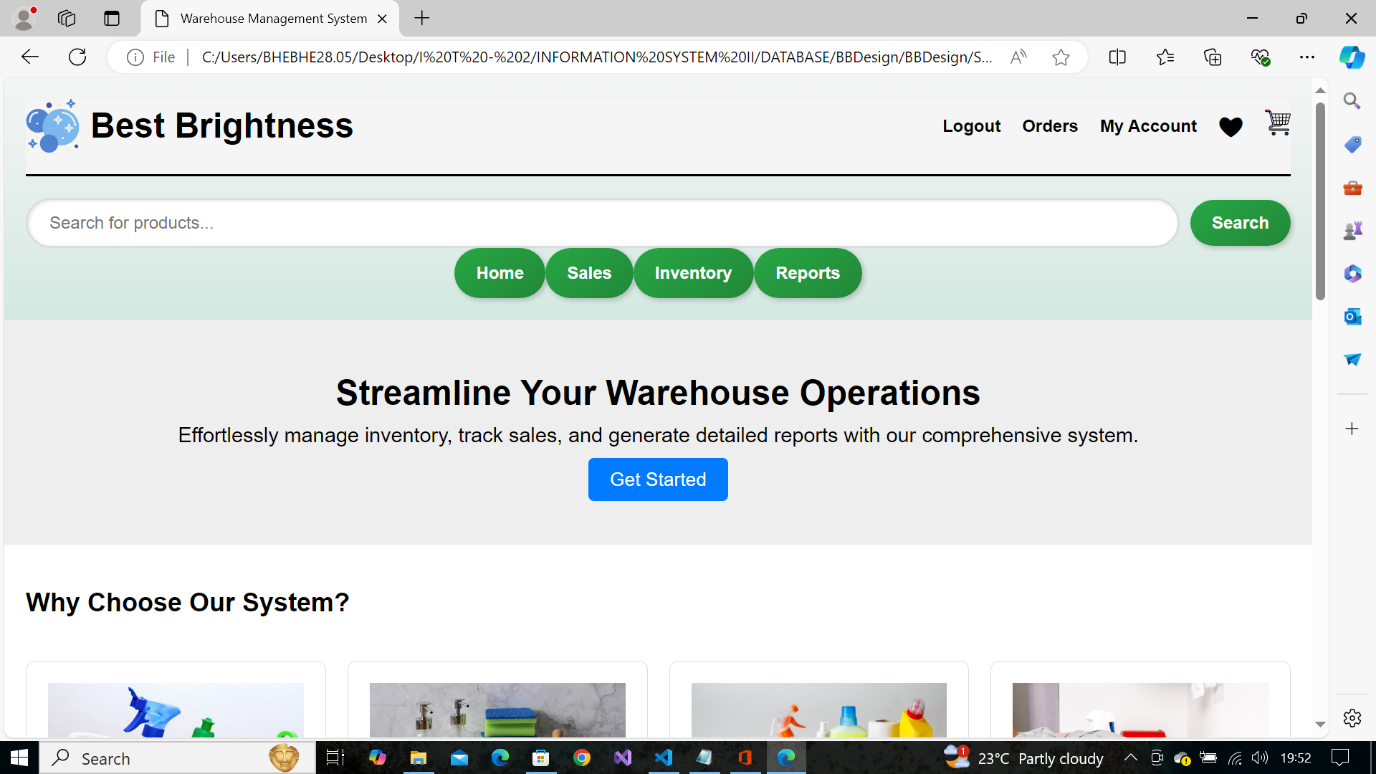
* + Layouts and Mockups:
  + Home Page: Welcome message, login/register buttons, and product overview.
  + Dashboard: Features buttons for Record Sales, Manage Inventory, View Reports, with a clean and modern layout.
  + Forms: Simplified and mobile-friendly forms for login, registration, and sales recording.

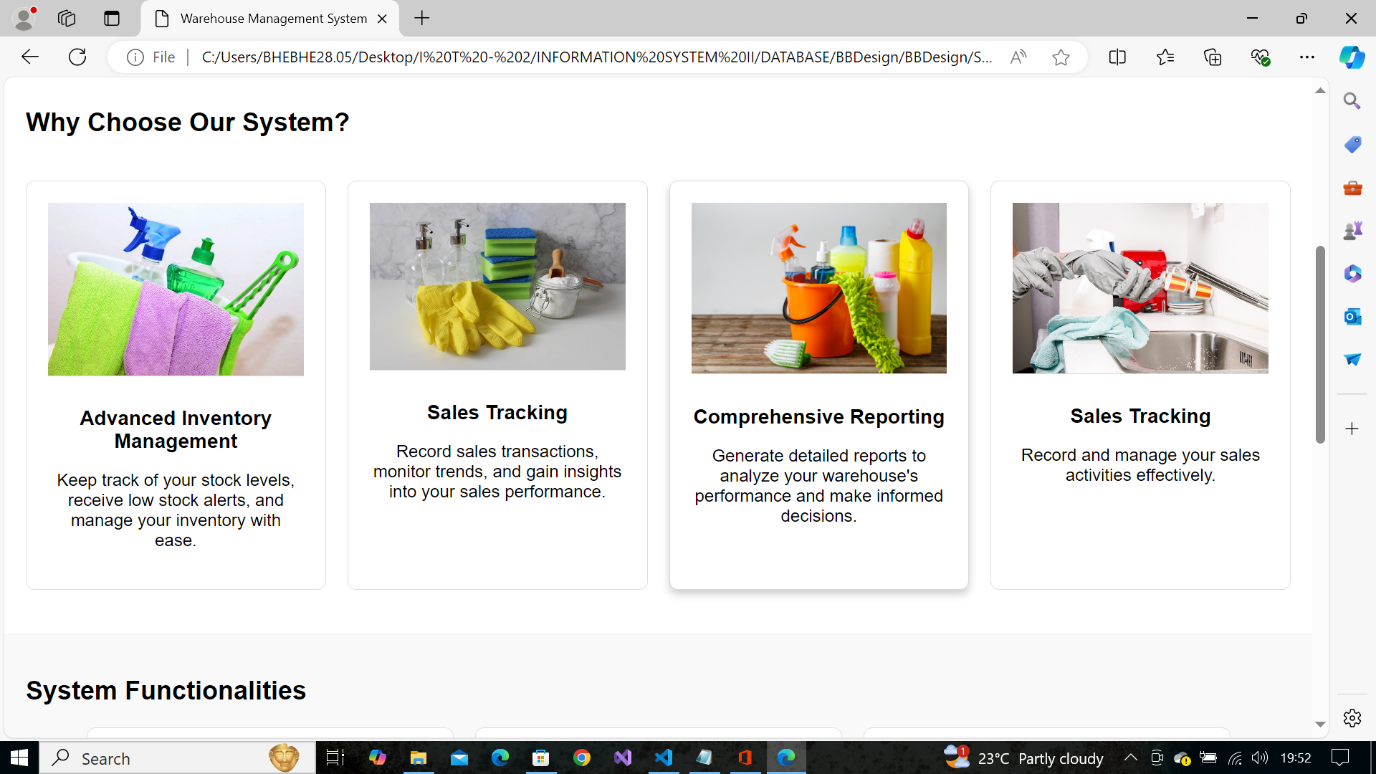
**LOGIN AND REGISTRATION**

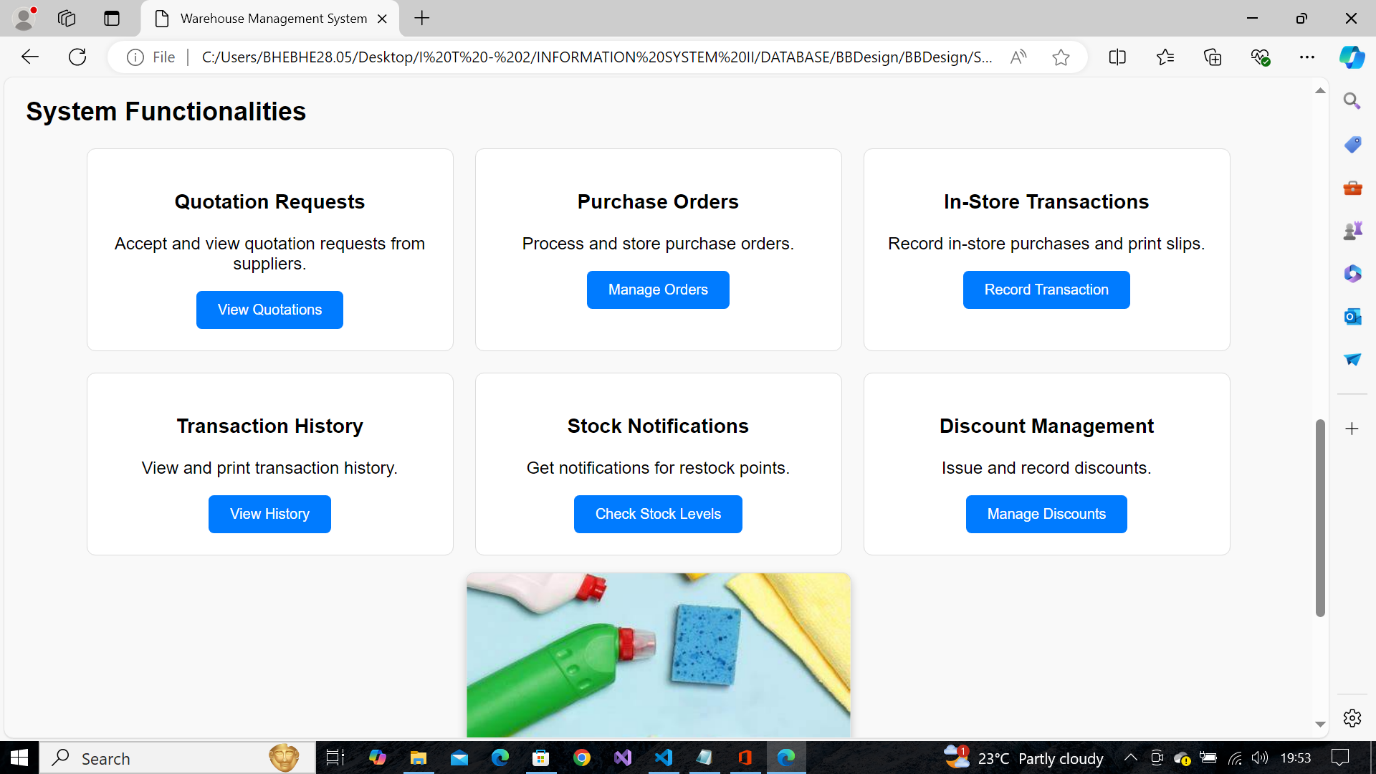
****

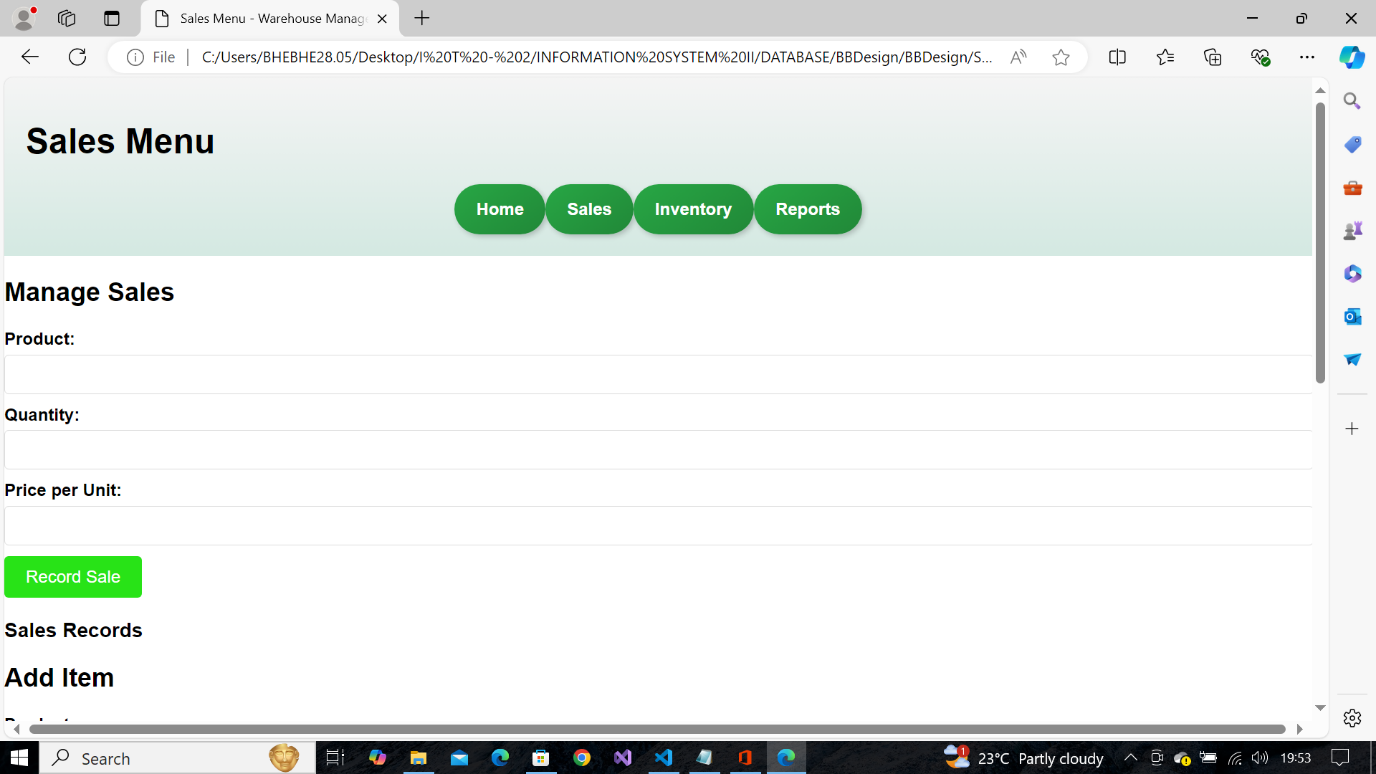
****

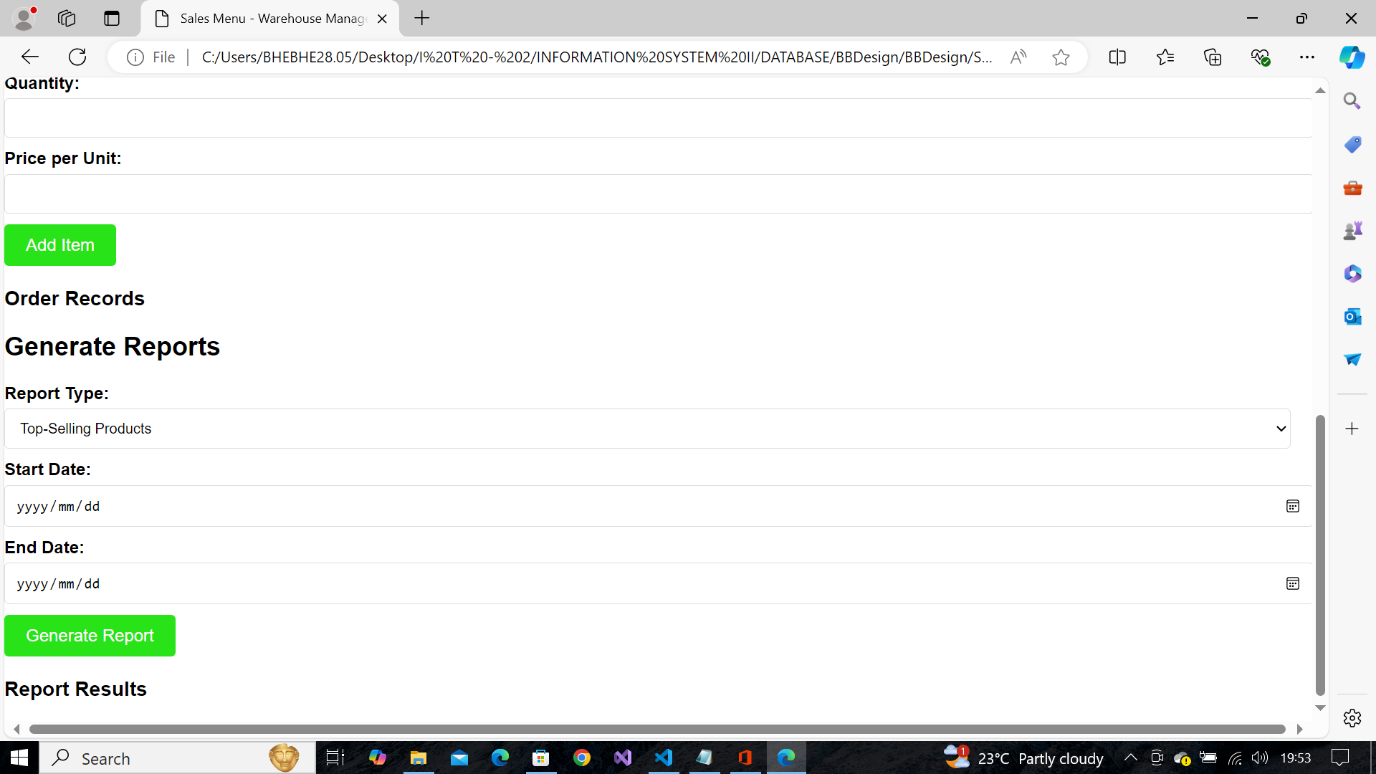
**SALES MENU**

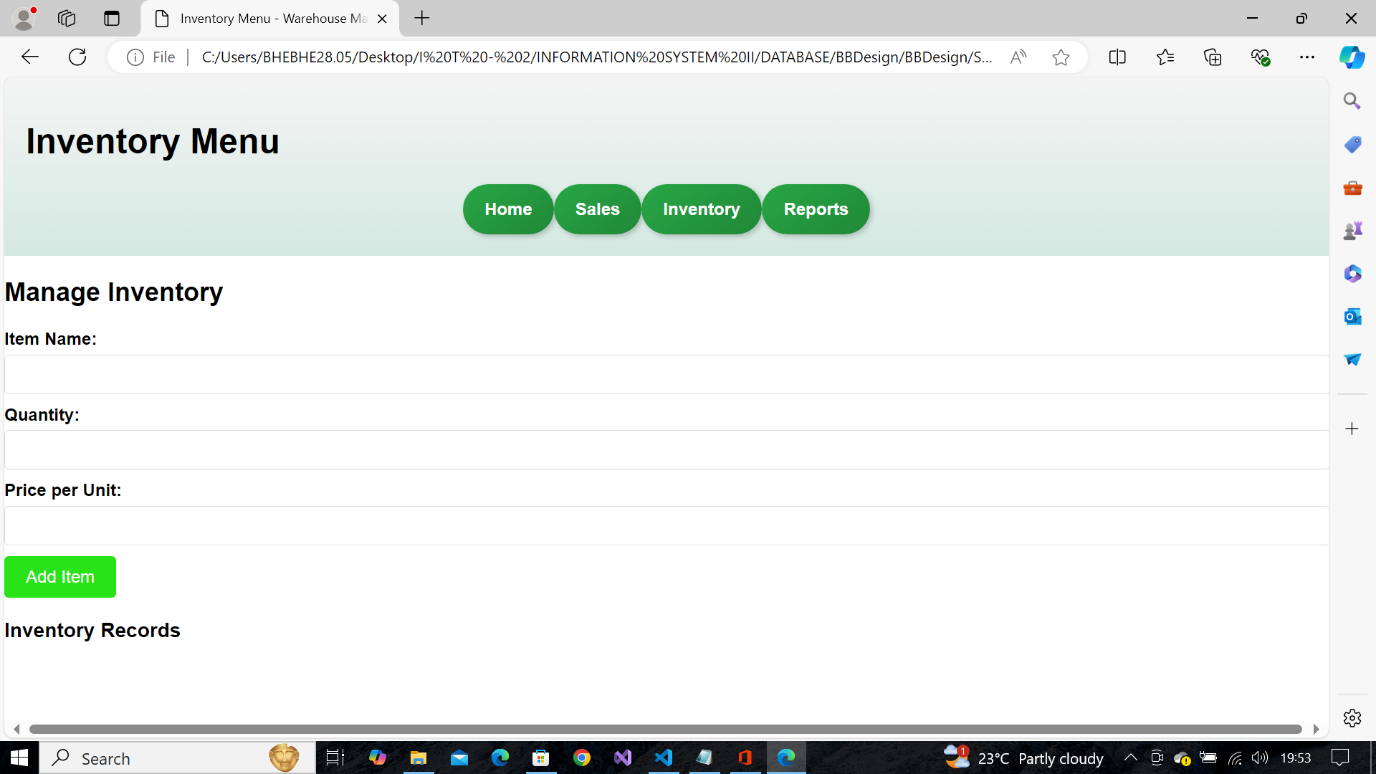
****

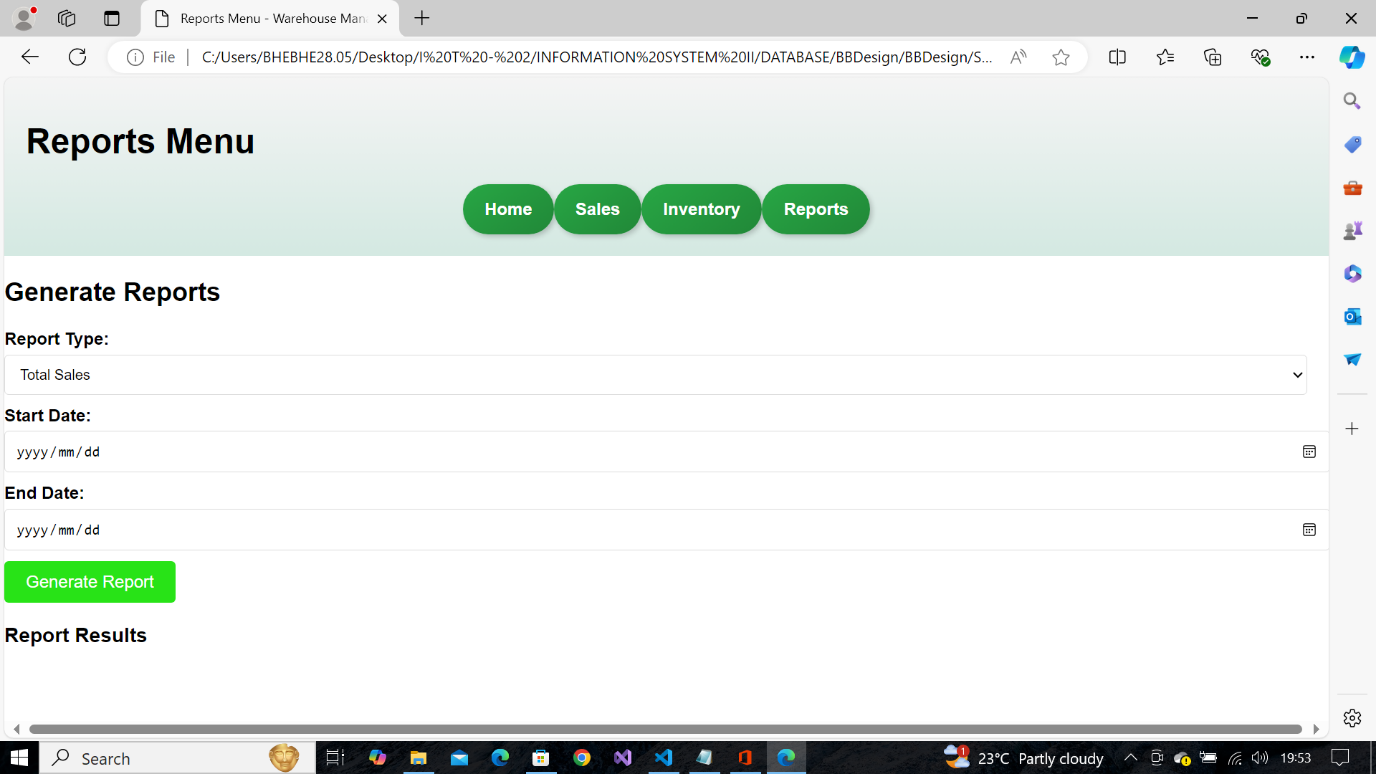
****



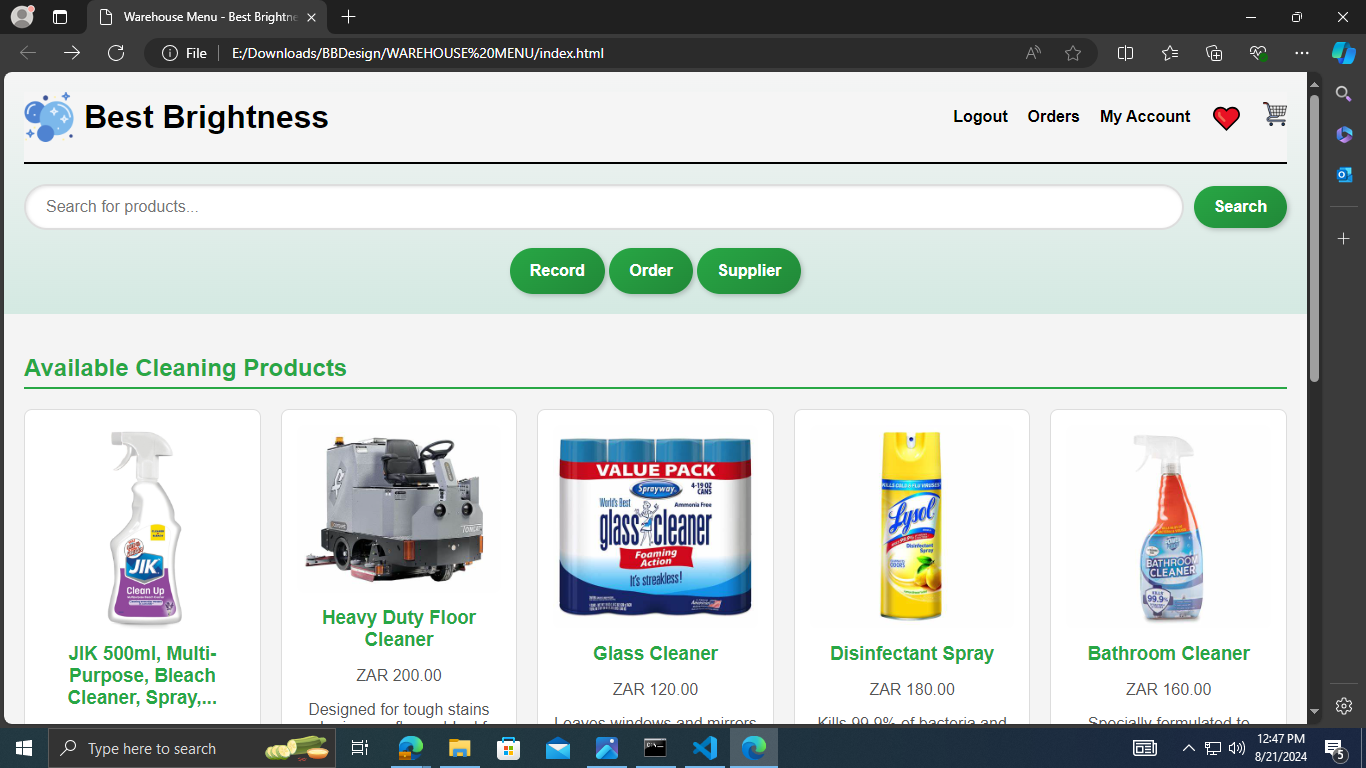
****

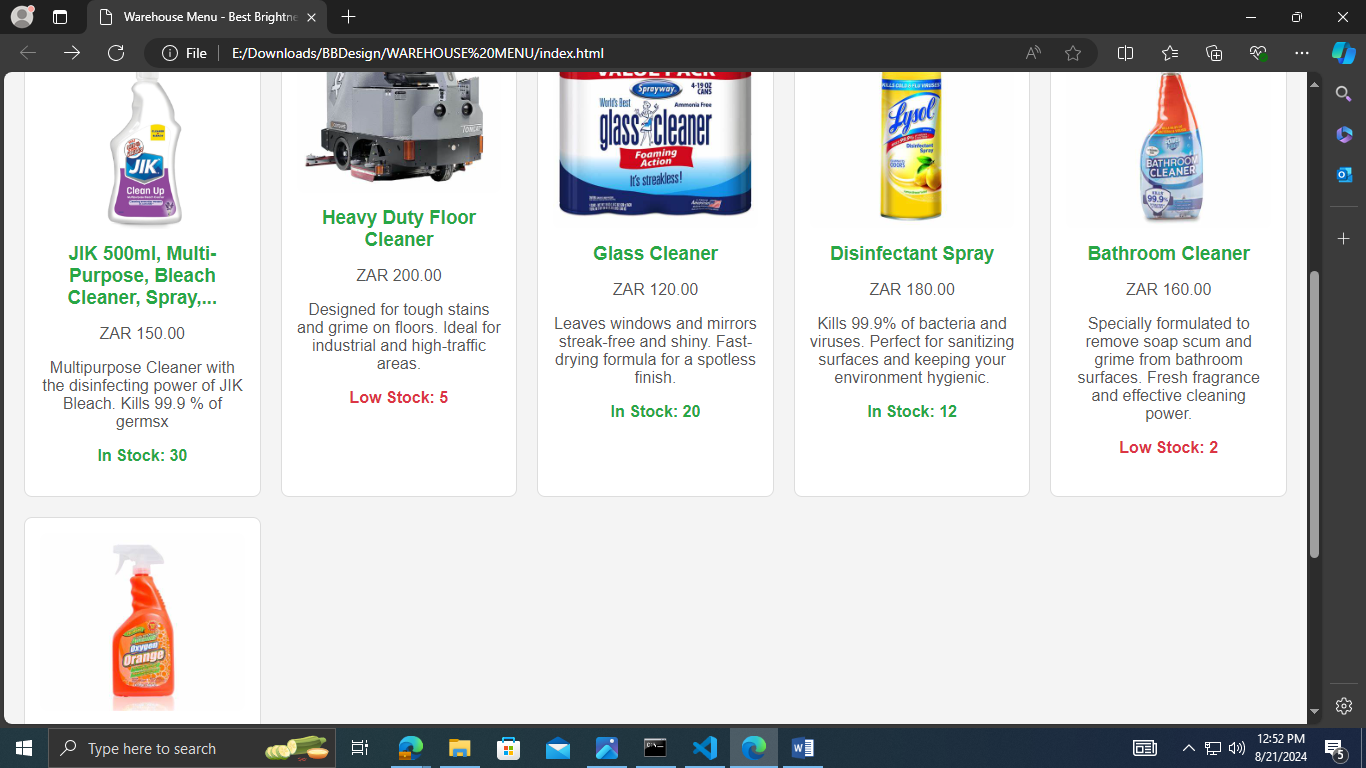
****

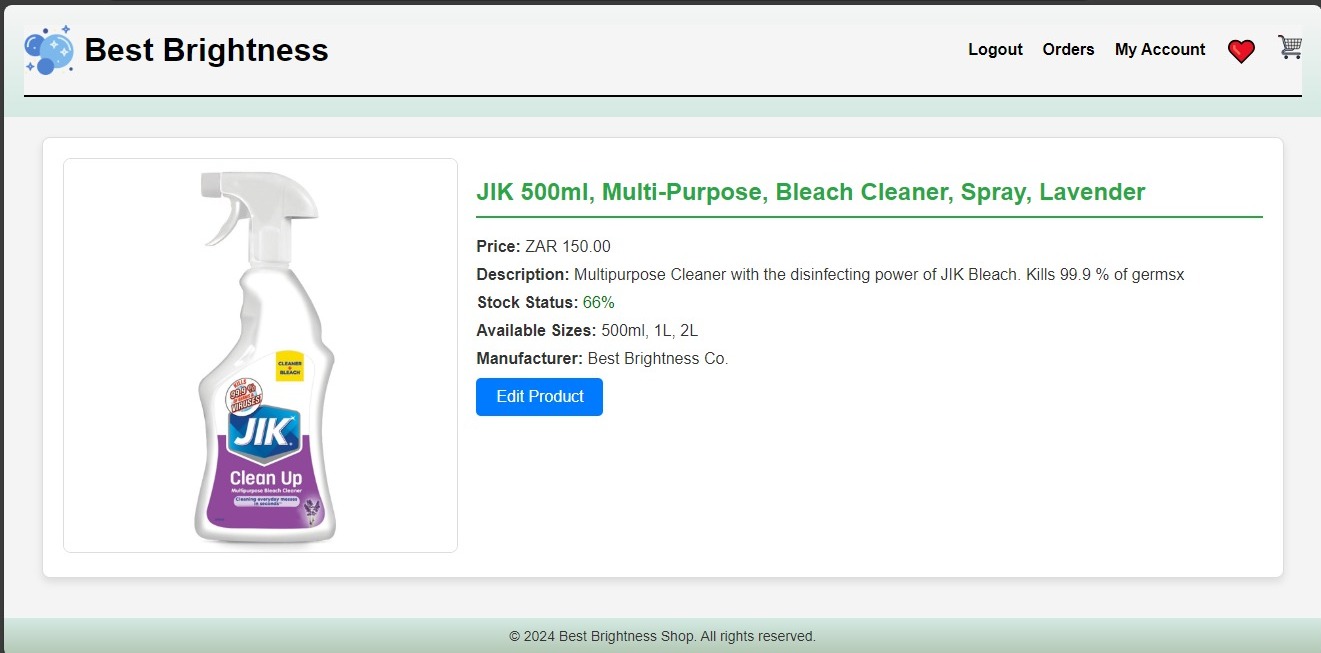
****

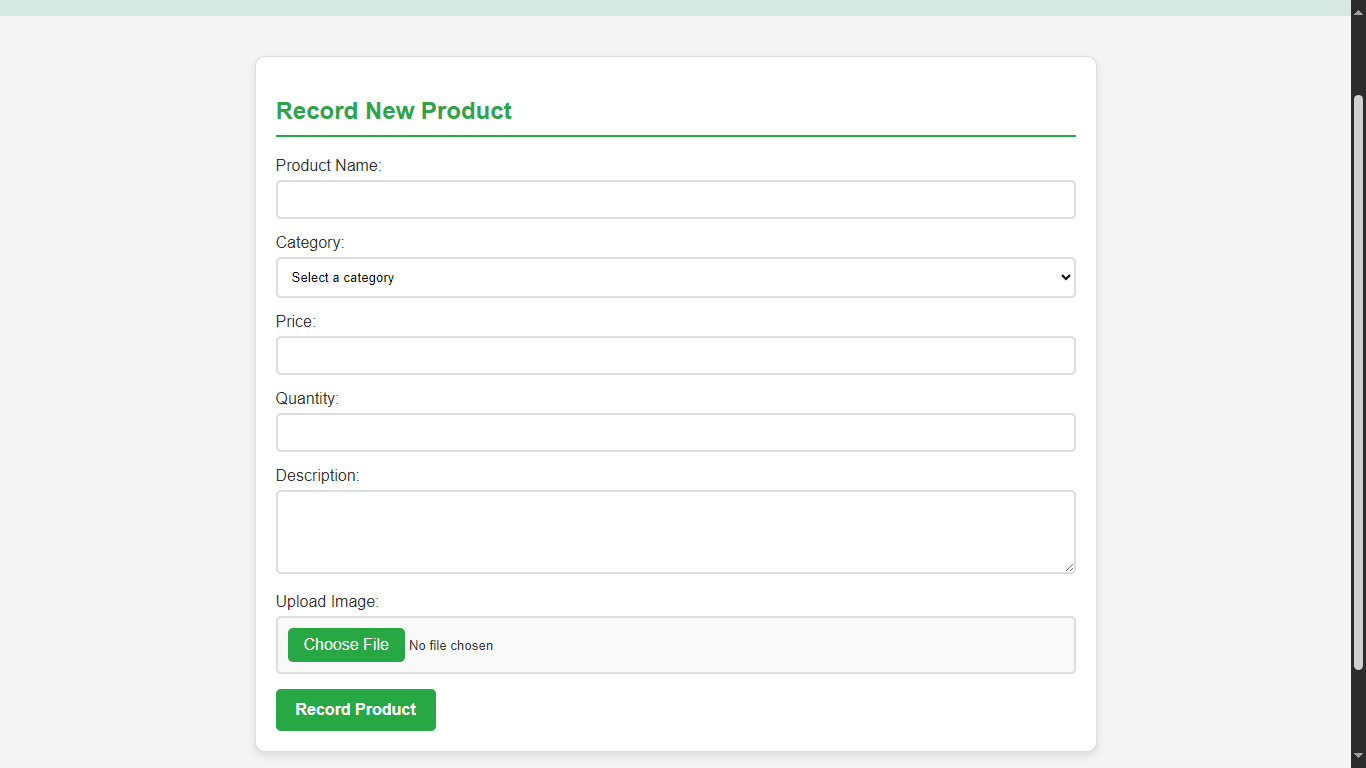
****

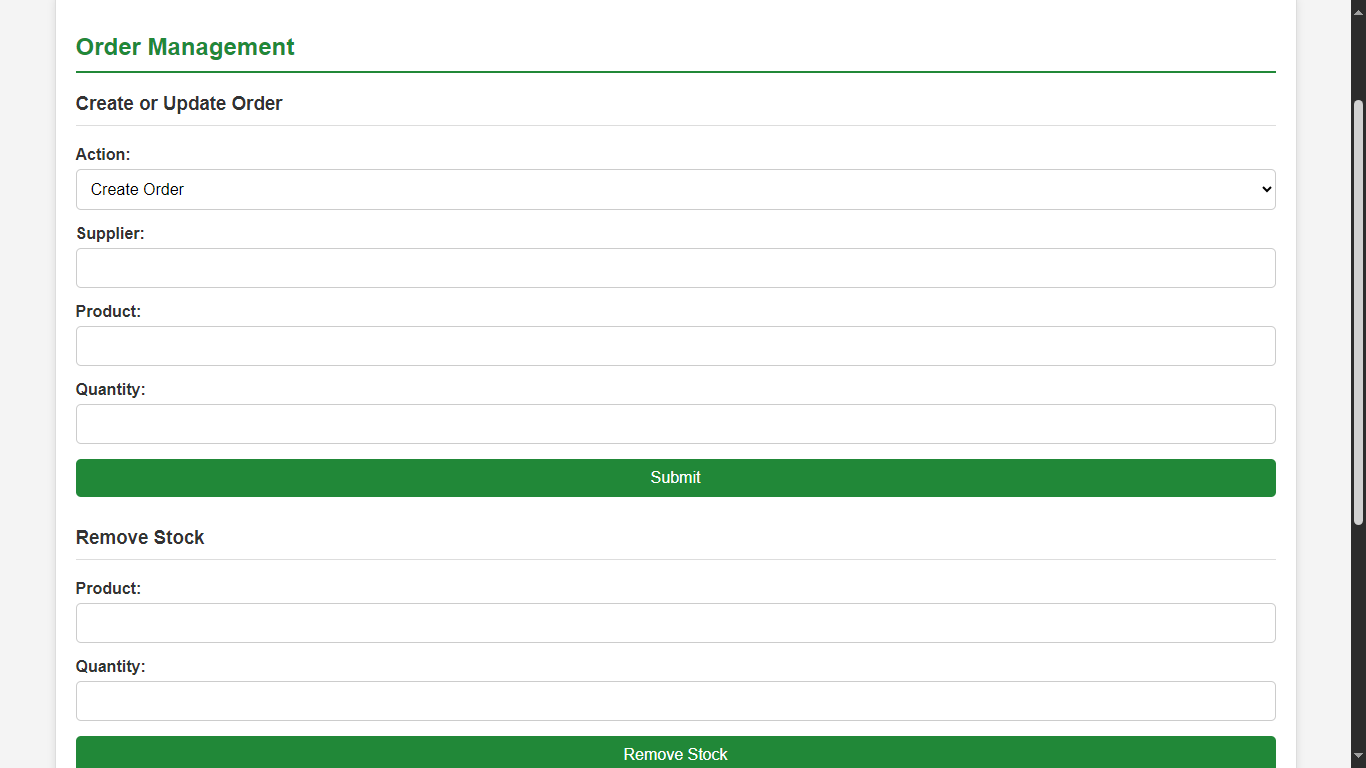
**WAREHOUSE MENU**

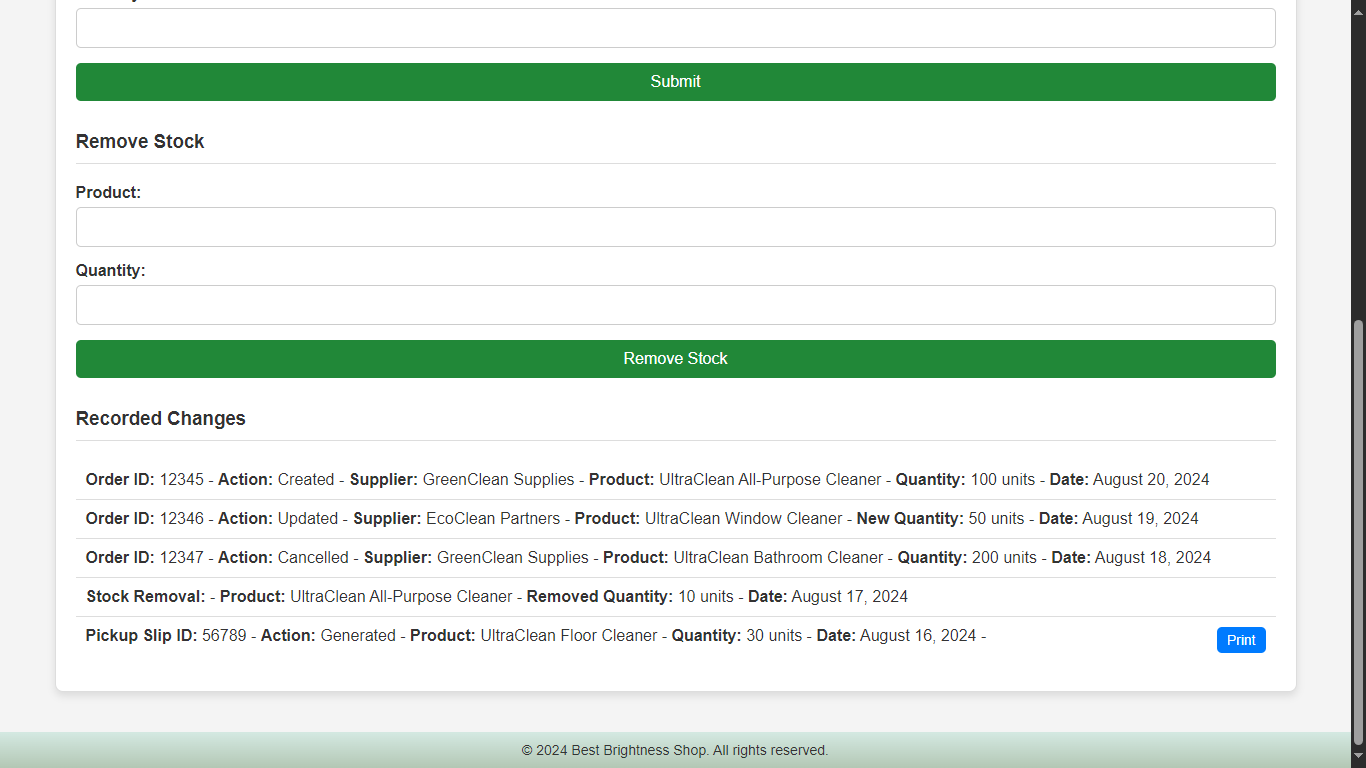


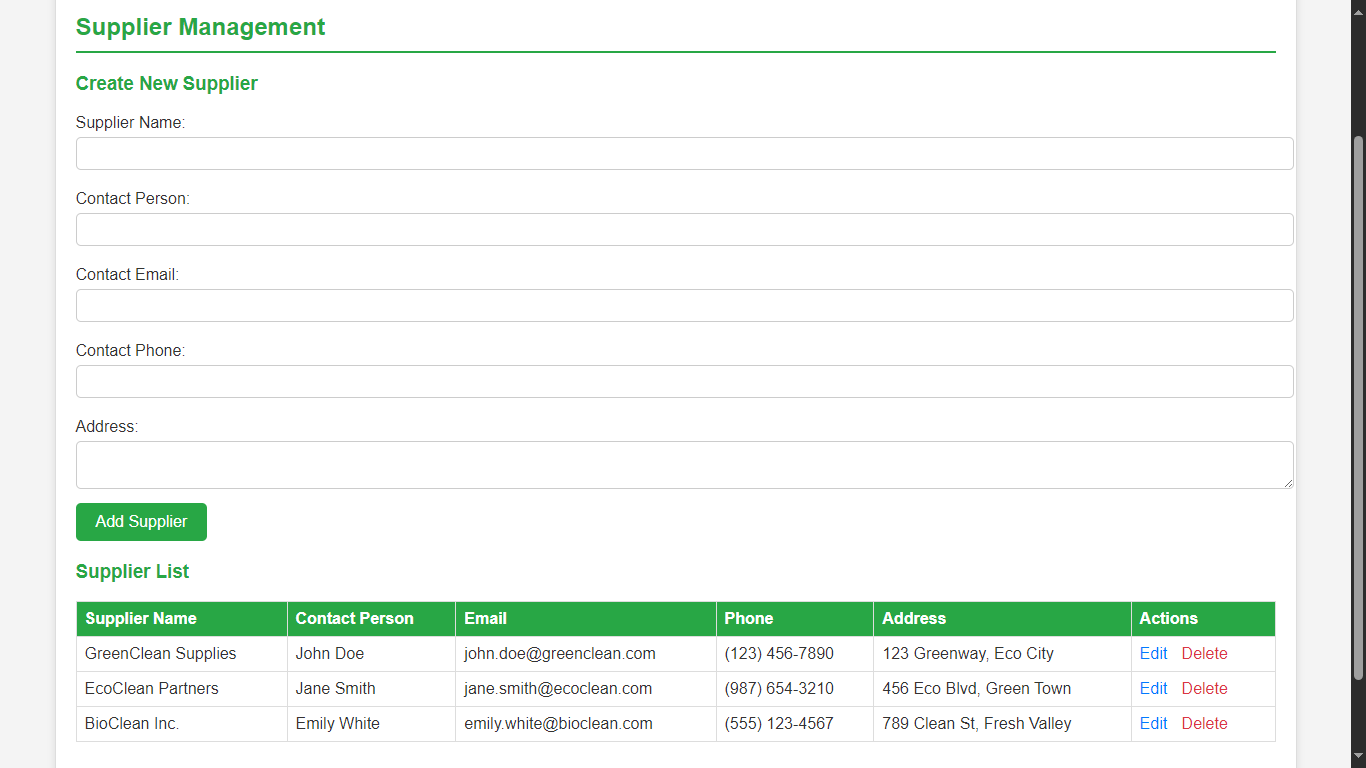




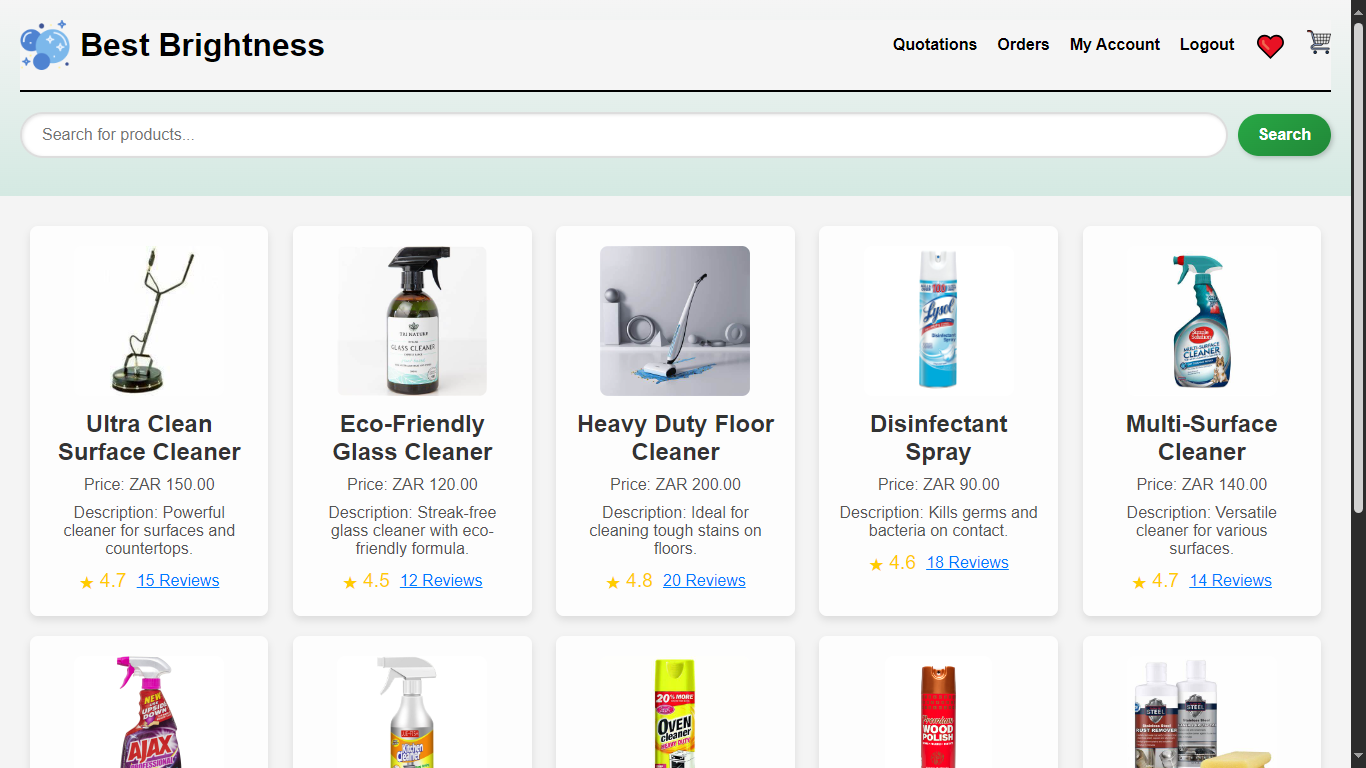


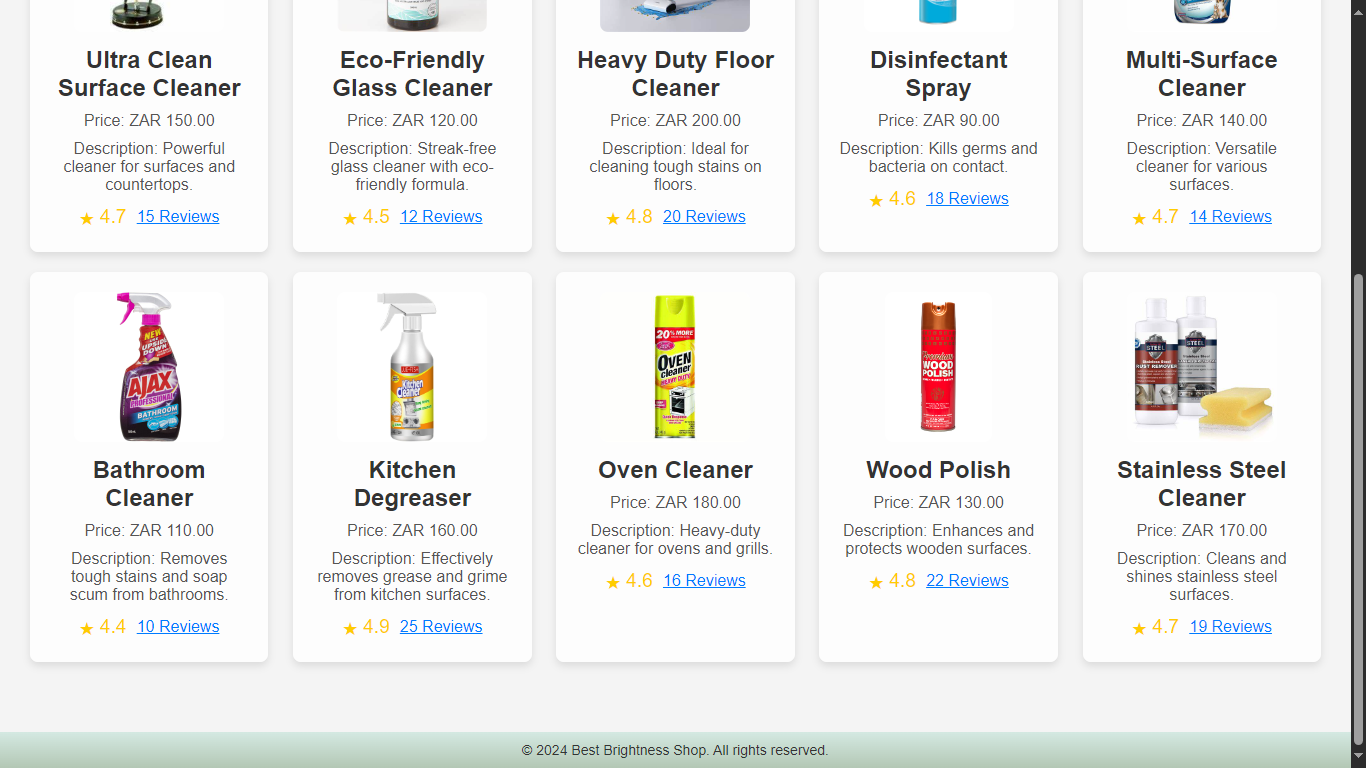


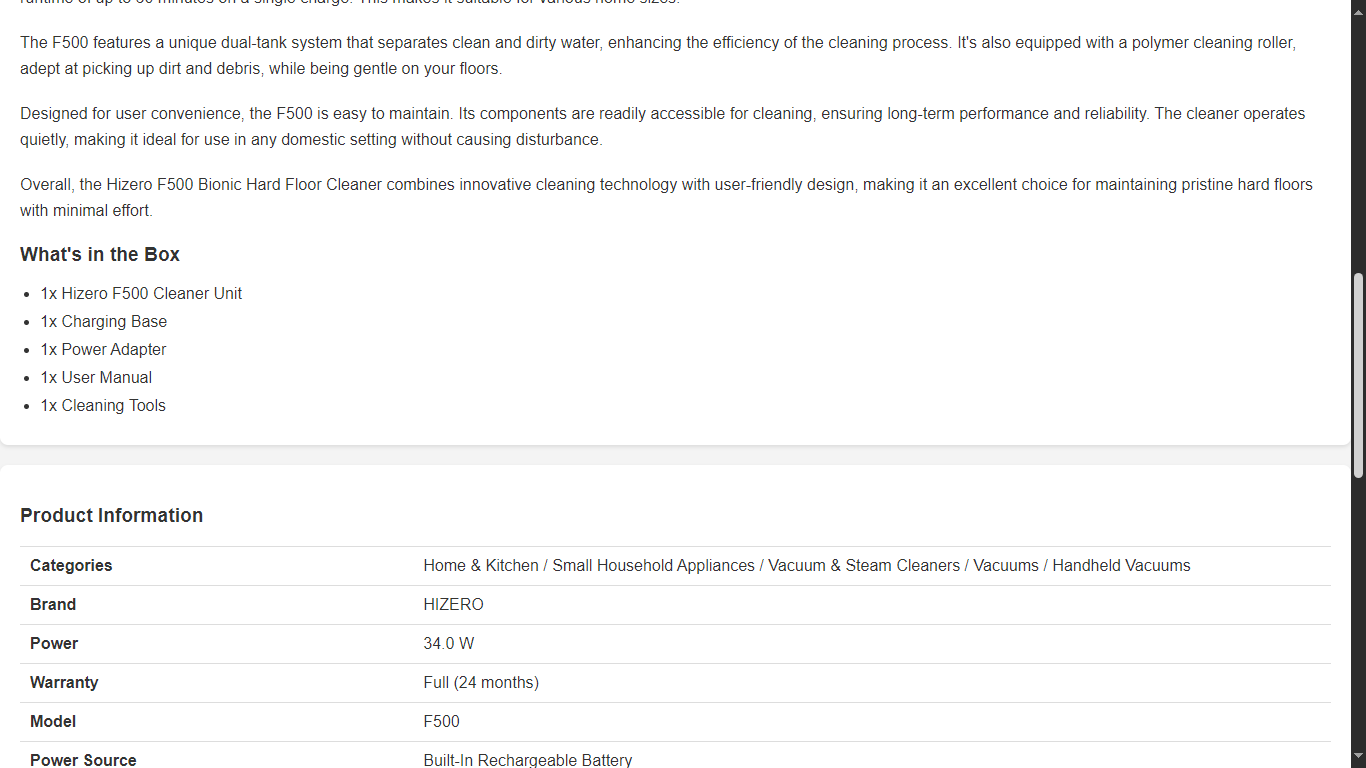
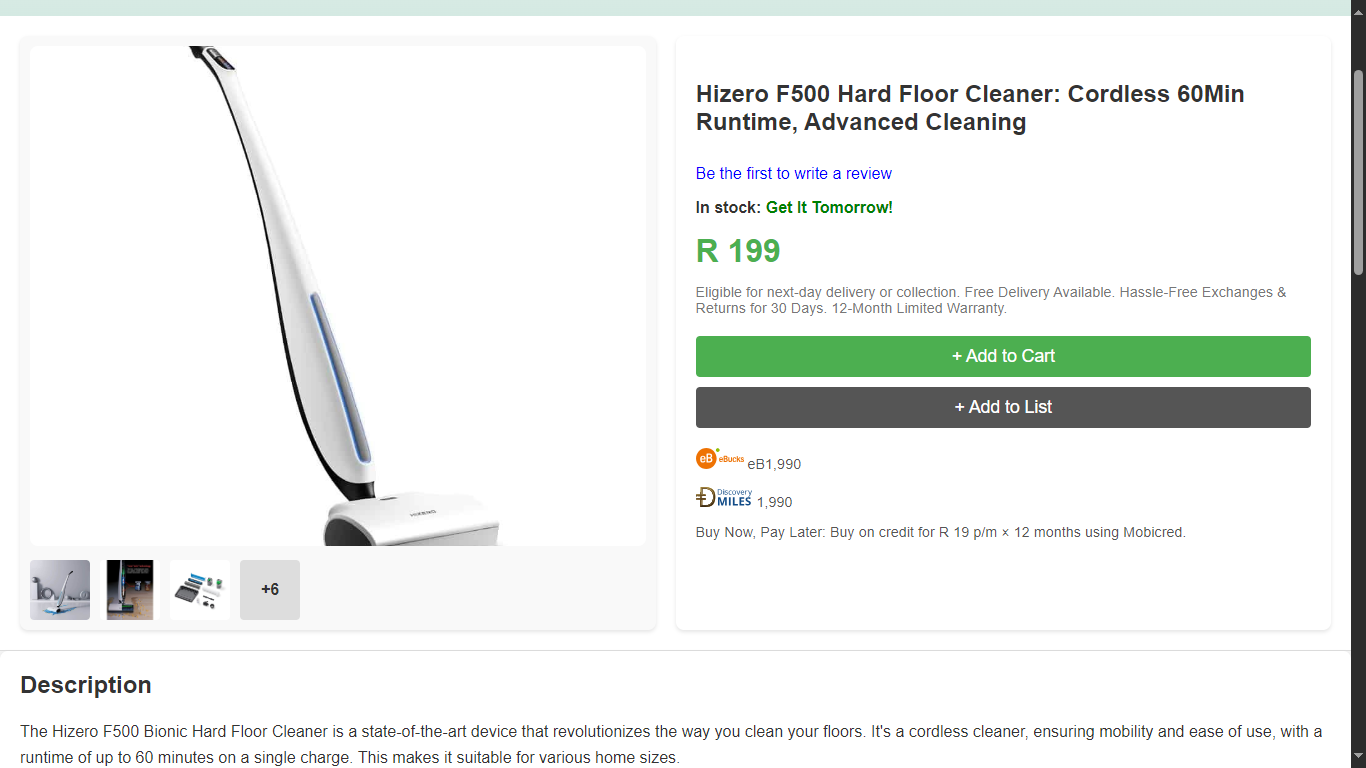


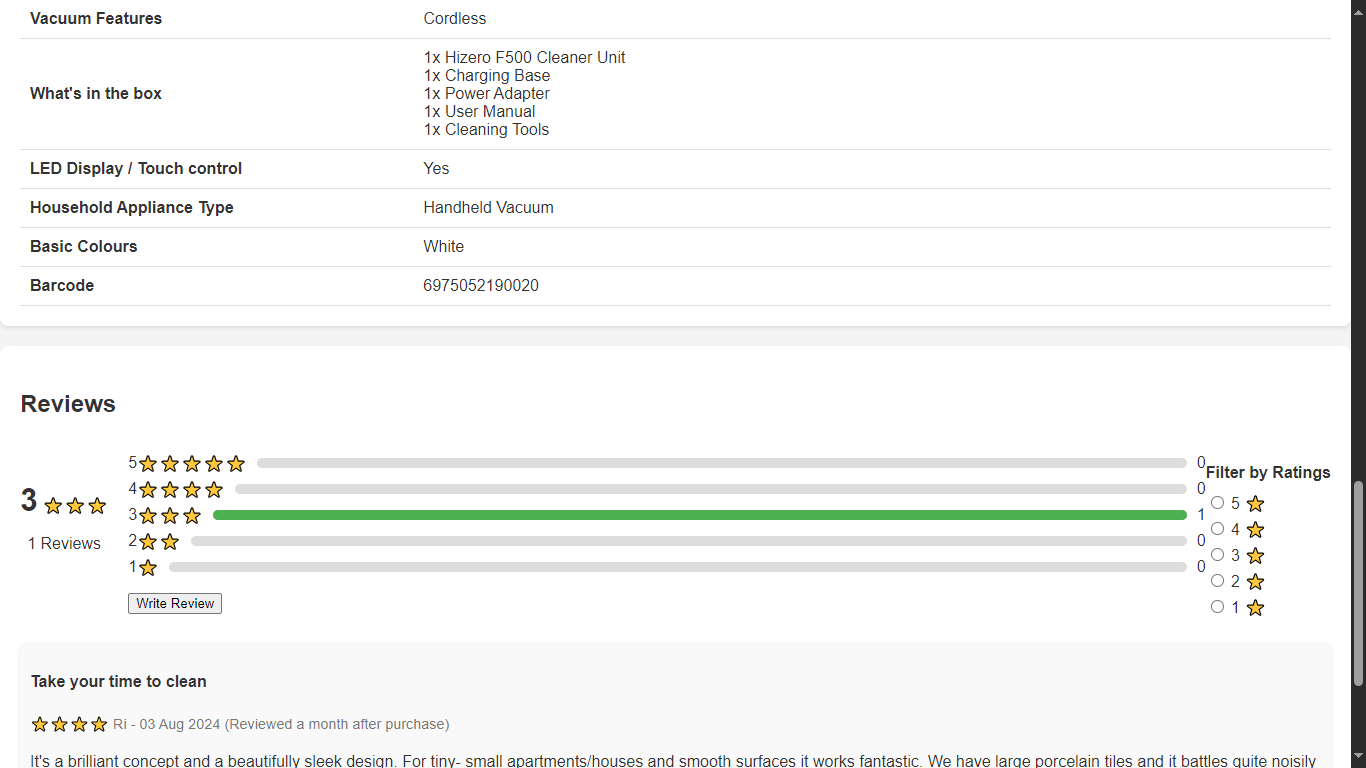


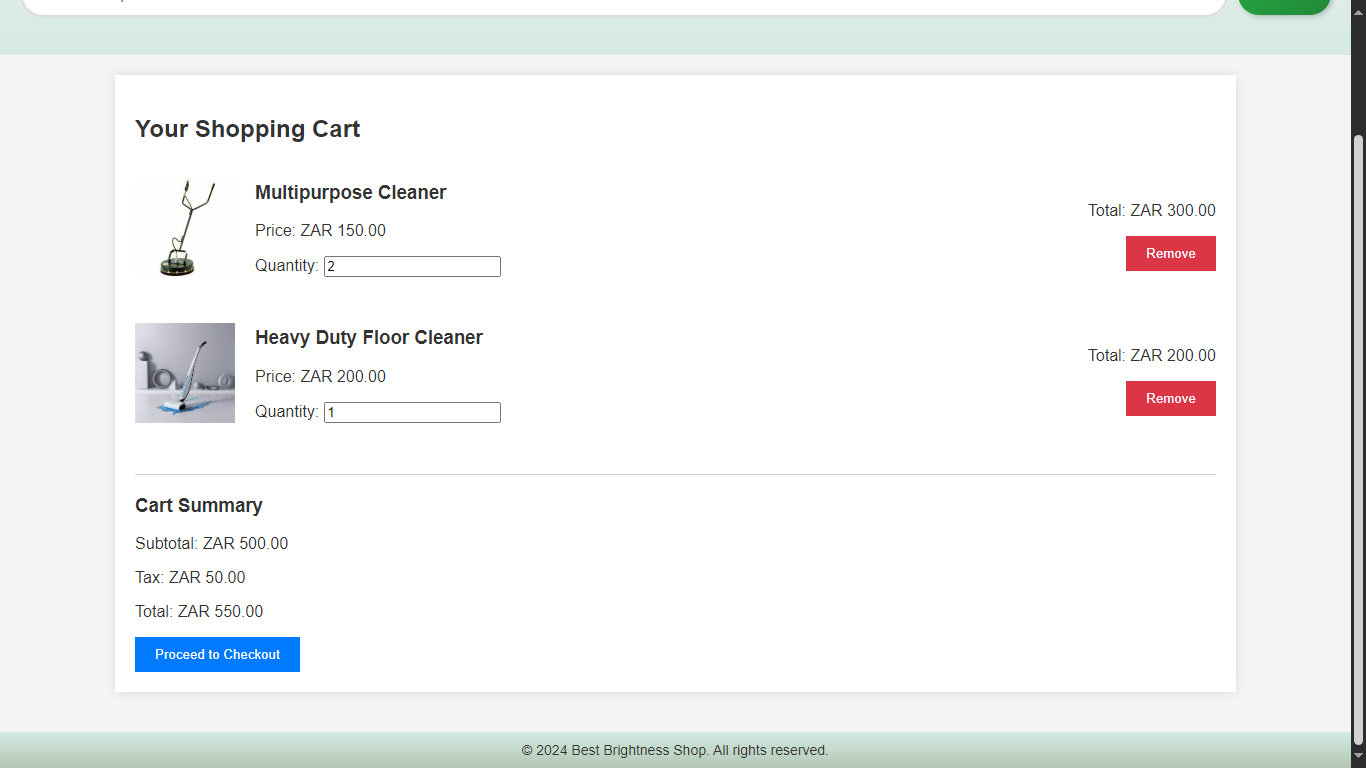
**CUSTOMER (ONLINE) MENU**

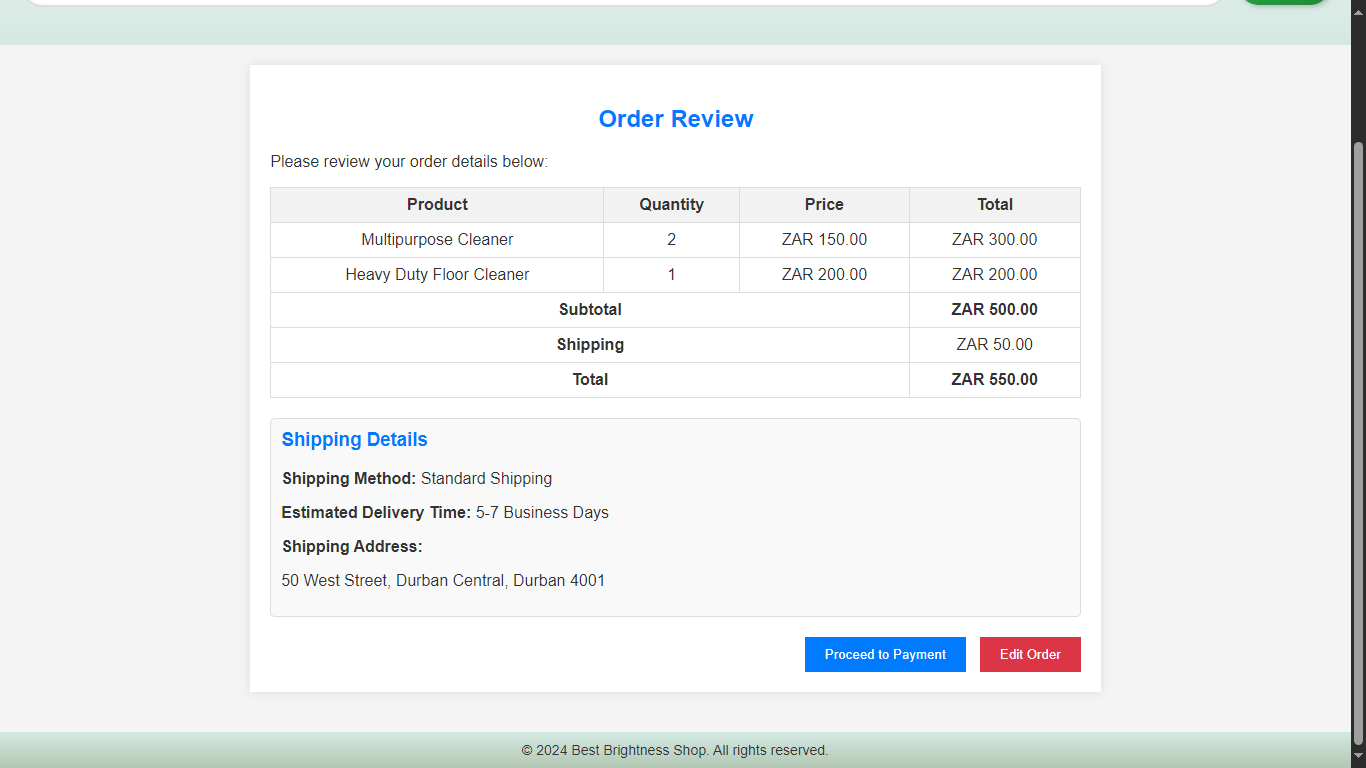
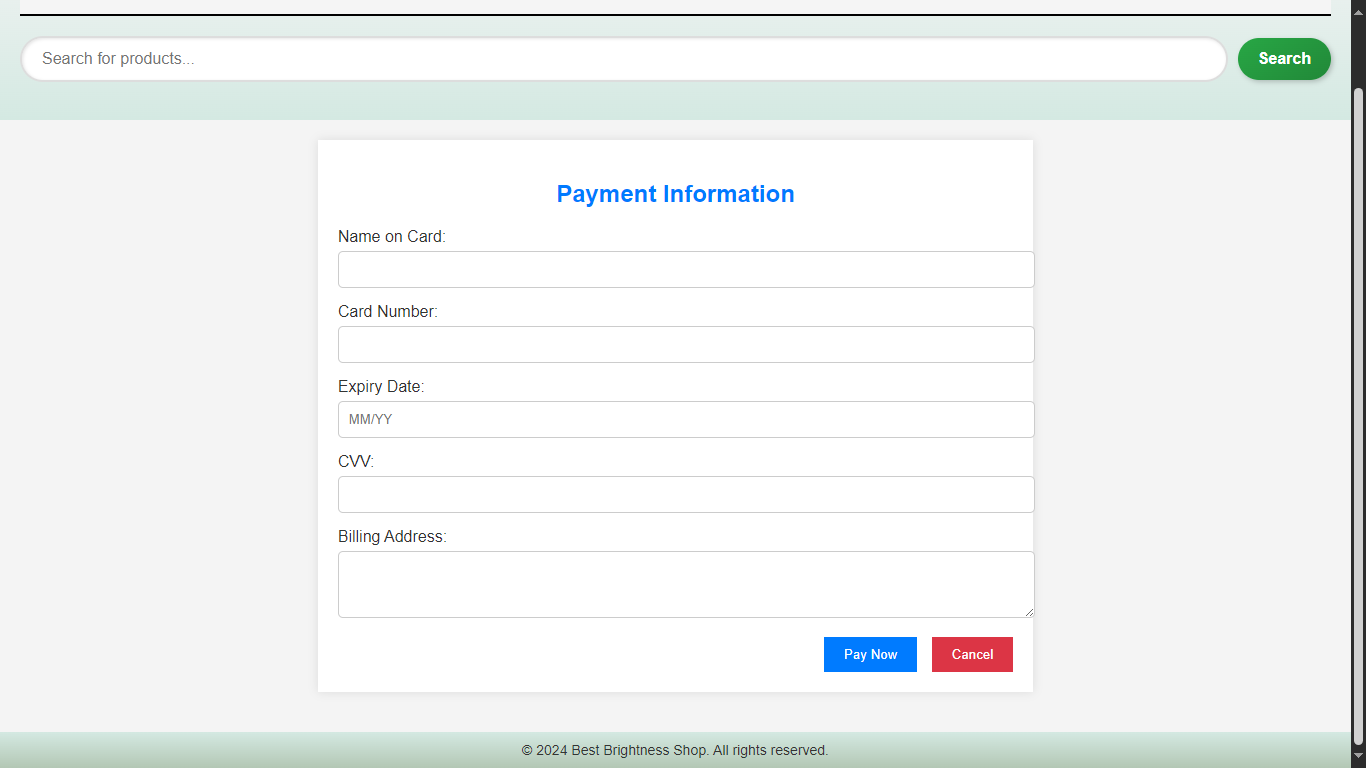


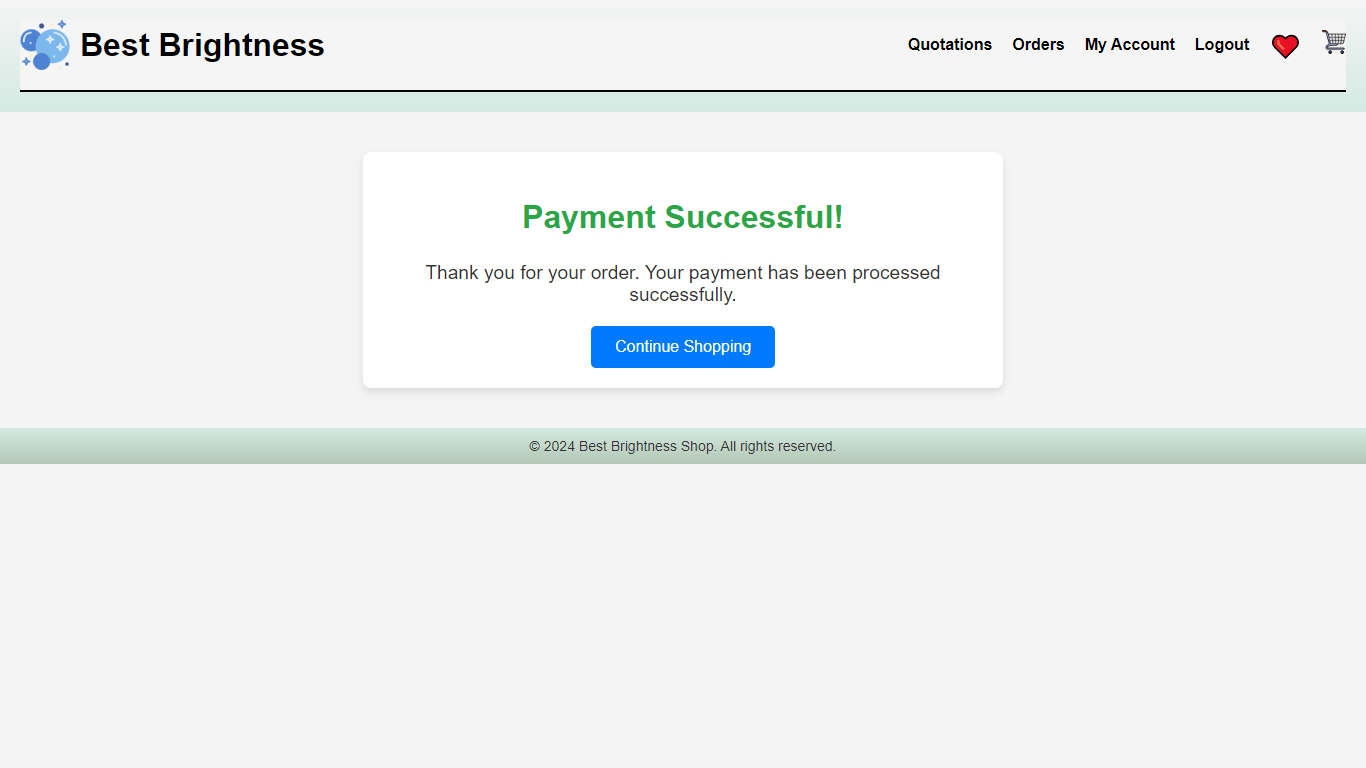




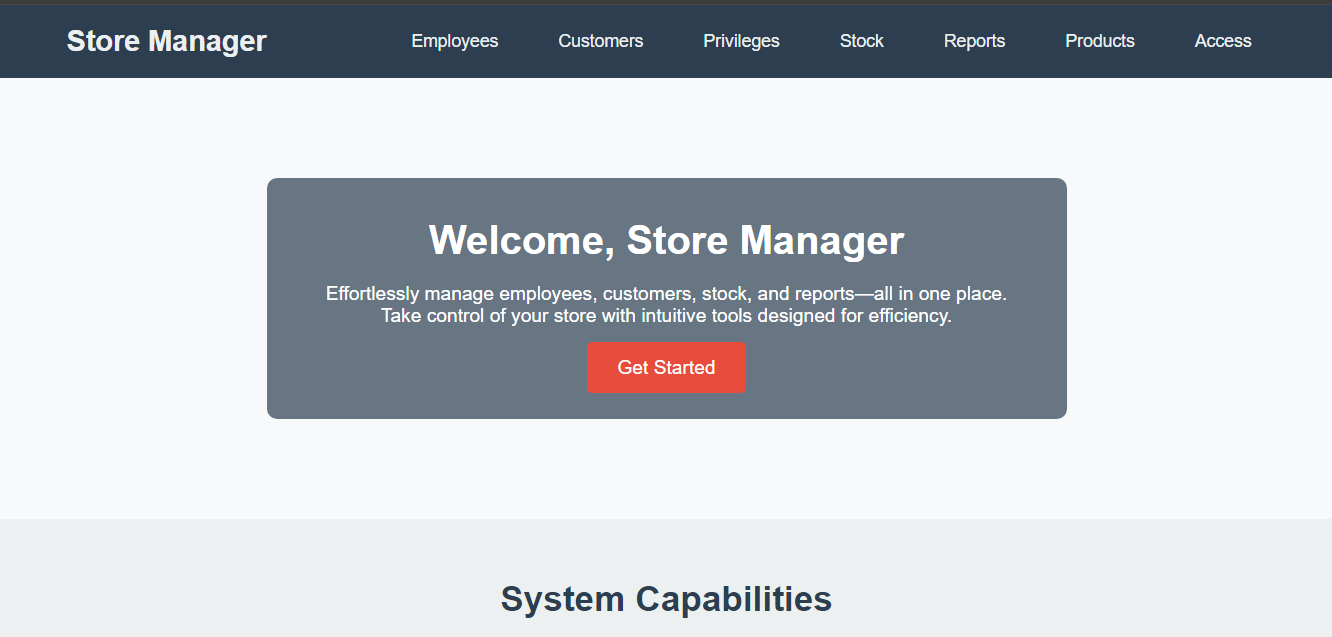


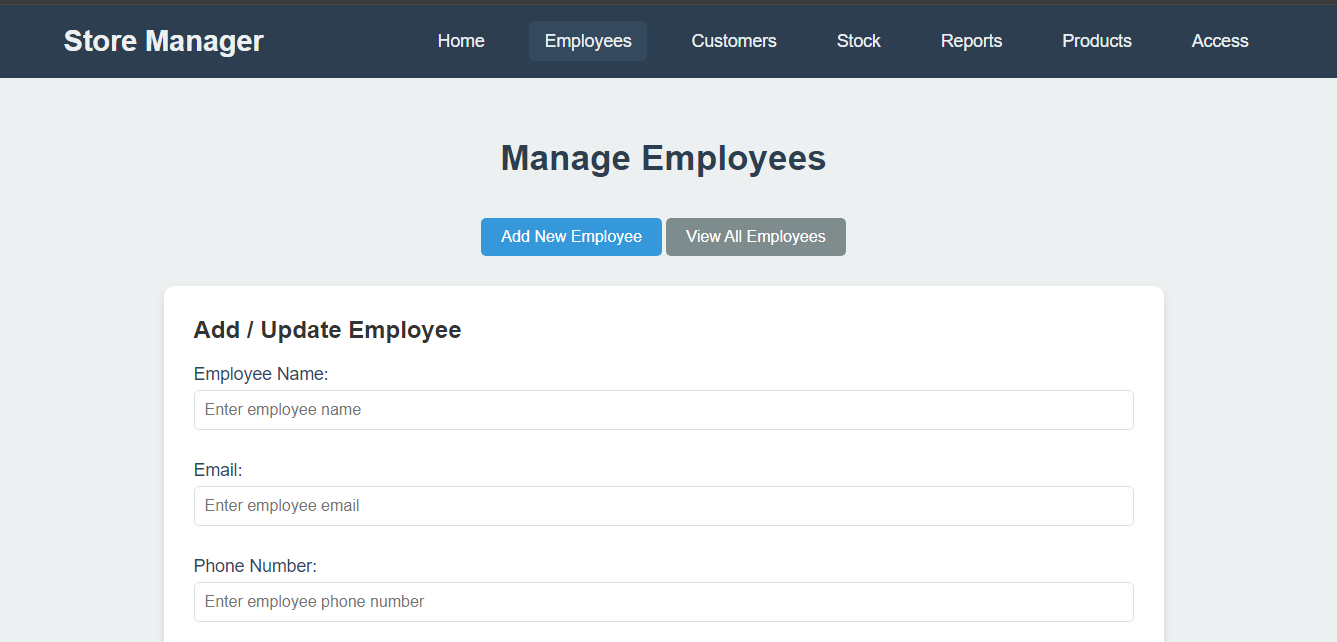




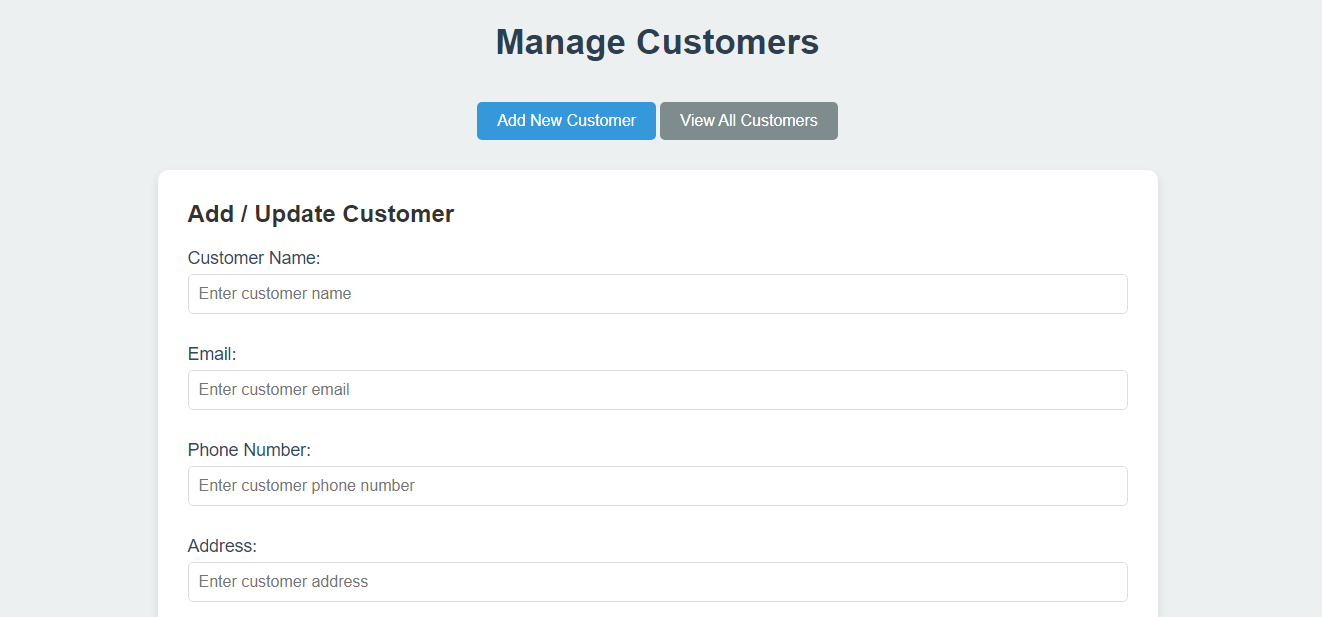


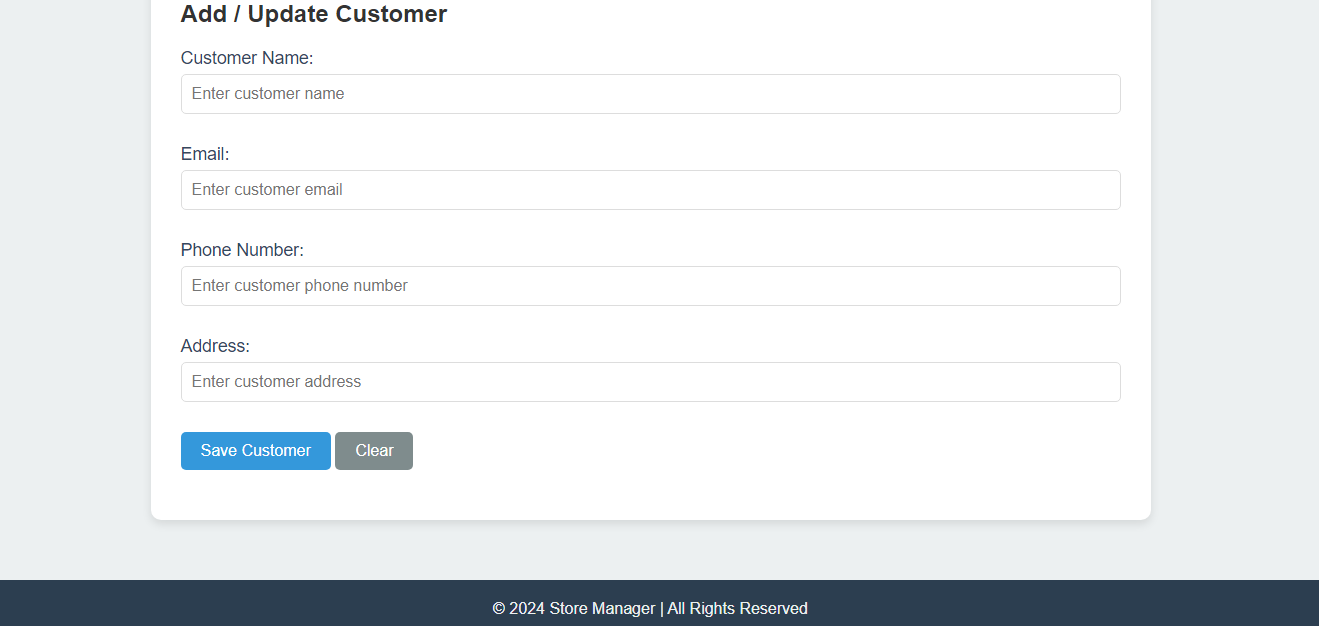
**STORE MANAGER MENU**

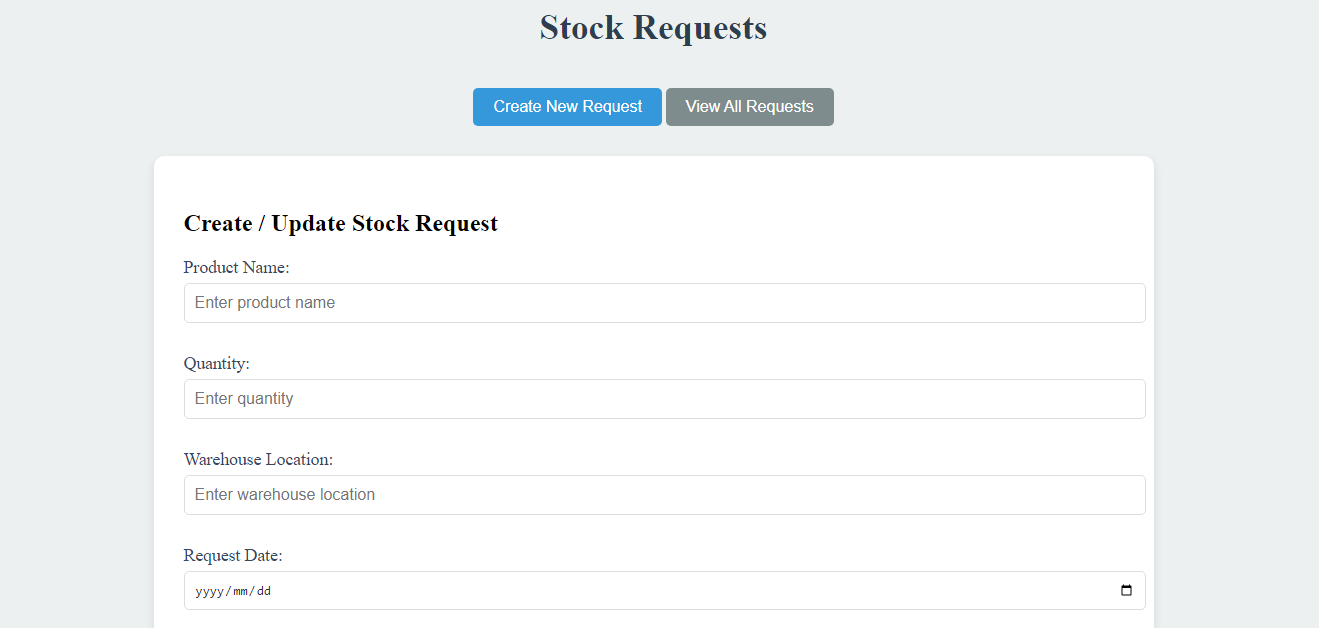
****

****

****

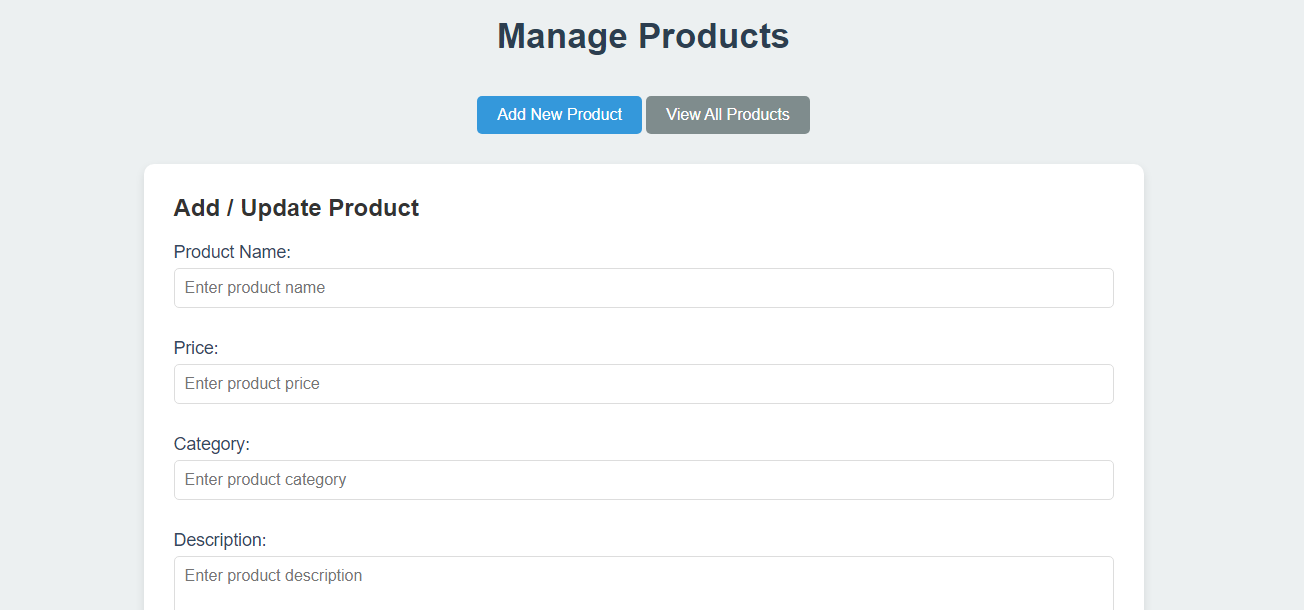
****

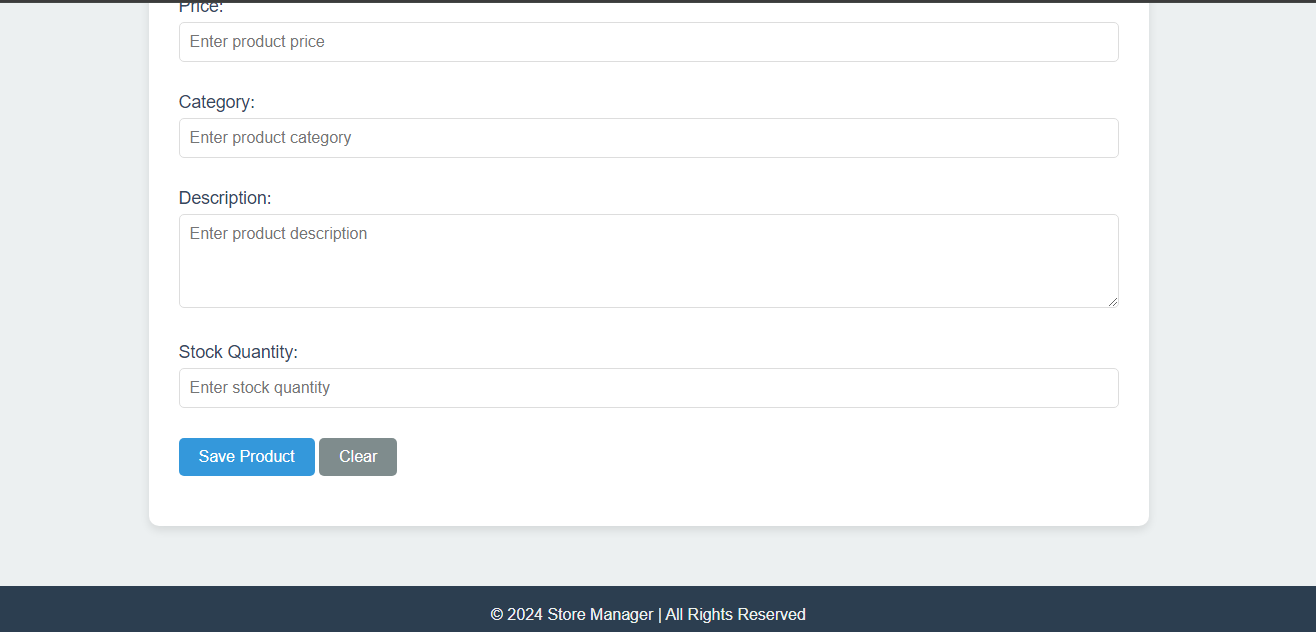
****

****

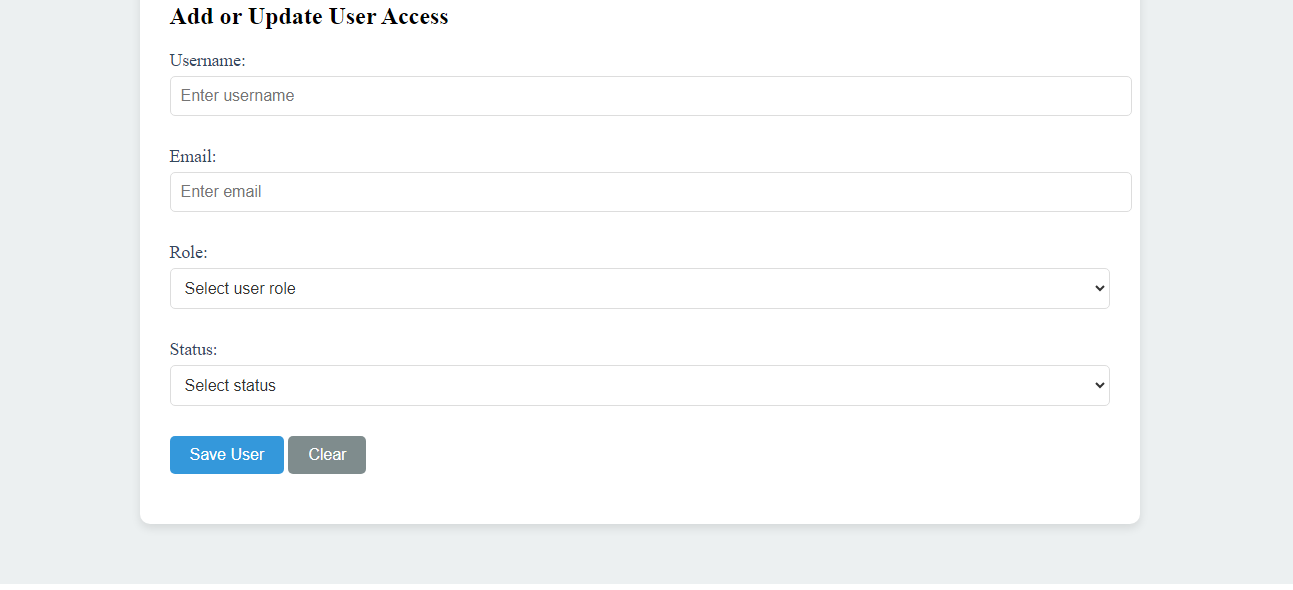
****

****

****

****

****

****

**6. Application Components Design**

**6.1 Component Overview**

**Subsystems:**

* + Inventory Management: Tracks product stock levels, reordering, and adjustments.
  + Sales Management: Manages sales transactions and records.
  + Customer Management: Stores customer details and purchase history.
  + E-commerce Integration: Facilitates online sales and order processing.

**6.2 Detailed Component Design**

* + Purpose: Describe the purpose of the component.
  + Data Structures: Describe any data structures used by the component.
  + Interaction: Describe how the component interacts with other components.

**7. Design Considerations**

1. **Performance:**

* **Efficiency:** The system is designed to handle high volumes of transactions efficiently. ASP.NET Core’s support for asynchronous programming ensures that the application remains responsive under load, minimizing wait times for users.
* **Caching:** To enhance performance, frequently accessed data such as product details and sales reports are cached using in-memory caching techniques. This reduces the load on the database and improves response times for users.
* **Load Balancing:** The system utilizes load balancing to distribute incoming requests evenly across multiple servers. This approach ensures high availability and optimizes resource utilization, preventing any single server from becoming a bottleneck.
* **Database Optimization:** Queries are optimized, and database indexing is implemented to speed up data retrieval. Regular maintenance tasks such as database deindexing and query optimization are scheduled to maintain performance levels.

1. **Scalability:**

* **Horizontal Scaling:** The system architecture supports horizontal scaling, allowing the addition of more servers to handle increased traffic. This ensures that the application can accommodate growth without compromising performance.
* **Modular Design:** The system is built with a modular architecture, allowing independent scaling of different components such as inventory management, sales tracking, and customer management. This modular approach facilitates targeted scaling based on specific needs.
* **Cloud Integration:** The application is designed to leverage cloud services for scalability. Cloud-based infrastructure allows for dynamic scaling of resources based on demand, ensuring the system can handle peak loads efficiently.

1. **Security:**

* **Authentication and Authorization:** ASP.NET Identity is used to manage user authentication and role-based authorization. This ensures that only authorized users have access to sensitive features and data within the system.
* **Data Encryption:** Data transmitted between users and the server is encrypted using SSL/TLS protocols. Additionally, sensitive data stored in the database, such as user credentials and financial information, is encrypted using AES encryption.
* **Input Validation and Sanitization:** The system implements strict input validation and sanitization to prevent common web vulnerabilities such as SQL injection and cross-site scripting (XSS). This helps protect the application from malicious attacks.
* **Security Audits:** Regular security audits and vulnerability assessments are conducted to identify and address potential security risks. This proactive approach ensures that the system remains secure against emerging threats.

4. **Usability:**

* **User-Friendly Interface:** The system features an intuitive and user-friendly interface designed to enhance the user experience. Consistent layouts, clear labeling, and responsive design elements contribute to ease of use across various devices.
* **Accessibility:** The application adheres to accessibility standards, ensuring that users with disabilities can navigate and interact with the system effectively. Features such as alternative text for images and keyboard navigation are included to support accessibility.
* **Feedback Mechanisms:** Users can provide feedback and report issues through built-in mechanisms within the application. This feedback is used to make continuous improvements to the user interface and overall usability.

5. **Dependability:**

* **Redundancy:** The system architecture includes redundancy at both hardware and software levels. Redundant servers, databases, and network components are implemented to minimize the impact of hardware failures and ensure continuous operation.
* **Backup and Recovery:** A comprehensive backup strategy is in place to protect against data loss. Regular backups are performed, and a well-defined recovery plan ensures that data can be restored quickly in the event of a failure.
* **Monitoring and Alerts:** The system includes monitoring tools to track performance, availability, and security. Alerts are configured to notify administrators of critical issues, enabling prompt response and resolution to maintain system dependability.

**8. Appendices**

A. Glossary of Terms

* ASP.NET Core: An open-source framework for building modern web applications and services. It is cross-platform and allows for building high-performance applications.
* Entity Framework Core (EF Core): An Object-Relational Mapper (ORM) that provides a higher-level abstraction to interact with databases using .NET objects.
* SQL Server: A relational database management system developed by Microsoft, used to store and manage the application's data.
* SSL/TLS: Protocols used to secure communication over a network by encrypting data transmitted between clients and servers.
* Load Balancer: A device or software application that distributes network or application traffic across multiple servers to ensure no single server becomes overwhelmed.

**B. Acronyms**

* UI: User Interface
* CRUD: Create, Read, Update, Delete
* HTTPS: Hypertext Transfer Protocol Secure
* AES: Advanced Encryption Standard
* HTTP: Hypertext Transfer Protocol
* TCP/IP: Transmission Control Protocol/Internet Protocol

**C. System Requirements**

* Hardware Requirements:
  + Server: Minimum 4 CPUs, 16 GB RAM, and 500 GB SSD for hosting the application and database.
  + Client: Desktop or laptop with a modern web browser (Chrome, Firefox, Edge) and internet access.
* Software Requirements:
  + Server: Windows Server 2019 or later, .NET Core Runtime, SQL Server 2019 or later.
  + Client: Latest version of web browsers (Chrome, Firefox, Edge) for optimal performance.

**D. Design Diagrams**

* Architecture Diagram: Illustrates the three-tier architecture including presentation layer, business logic layer, and data access layer.
* Database Schema Diagram: Shows the tables and their relationships in the SQL Server database.
* User Interface Mockups: Provides visual representations of key user interfaces such as the home page, dashboard, and various forms.

**E. Implementation Plan**

* Phase 1: Requirements Gathering and Analysis
  + Duration: 2 weeks
  + Activities: Gather detailed requirements, analyze business needs, and document functional and non-functional requirements.
* Phase 2: System Design
  + Duration: 3 weeks
  + Activities: Create design documents, develop architecture diagrams, and finalize database schema.
* Phase 3: Development
  + Duration: 8 weeks
  + Activities: Develop application components, implement user interfaces, and integrate with the database.
* Phase 4: Testing
  + Duration: 4 weeks
  + Activities: Conduct unit tests, integration tests, and user acceptance tests.
* Phase 5: Deployment
  + Duration: 2 weeks
  + Activities: Deploy the application to the production environment, perform final validation, and provide user training.
* Phase 6: Maintenance
  + Duration: Ongoing
  + Activities: Monitor system performance, apply updates and patches, and address user feedback and issues.

**F. Contact Information**

* Project Manager: Mbekezeli Sibusiso Mthethwa, Email: mbekezelimthethwa@gmail.com, Phone: +27 71 832 7848

**9. References**

* ASP.NET Core Documentation: <https://docs.microsoft.com/aspnet/core>
* Available at [[10,000+ Free Cleaning Products & Cleaning Images - Pixabay](https://pixabay.com/images/search/cleaning%20products/)]
* Entity Framework Core Documentation: <https://docs.microsoft.com/ef/core>
* SQL Server Documentation: <https://docs.microsoft.com/sql/sql-server>
* OWASP (Open Web Application Security Project): <https://owasp.org> - For security best practices and guidelines.