ABSTRACT

The evolution of e-commerce is undergoing a transformative shift, driven by the convergence of advanced technologies and data-centric approaches. This paper explores the integration of LLAMA Vision (a cutting-edge computer vision model) with robust data management systems to redefine the next generation of online shopping experiences.

LLAMA Vision, with its deep learning capabilities, enhances product recognition, user personalization, and real-time interaction within virtual environments, improving the overall consumer experience.

Coupled with efficient and scalable data management frameworks, this integration promises to unlock smarter insights, seamless inventory management, and optimized recommendation systems.

By leveraging the power of artificial intelligence and streamlined data processing, businesses can unlock new growth opportunities, offering consumers a more intuitive, engaging, and tailored shopping journey. This convergence will revolutionize the way businesses operate, empowering e-commerce platforms to respond dynamically to changing customer preferences and market trends.

It ensures efficient inventory control, deeper customer insights, and personalized recommendations. This integration allows businesses to optimize operations, enhance customer experiences, and improve marketing strategies. Ultimately, it promises a smarter, more dynamic e-commerce environment, driving growth and customer loyalty while overcoming challenges like data privacy and scalability.

Table of Contents

List of Figures

Chapter 1. Introduction

The project focuses on developing a comprehensive data management system tailored for an e-commerce platform. The proposed system leverages a hybrid database approach by integrating relational databases (RDBMS) for structured data such as customer details, order history, and product inventory, along with NoSQL databases to manage unstructured data, including customer reviews, multimedia content, and user activity logs. This integration enhances data retrieval, scalability, and performance while ensuring seamless customer interactions.

A key innovation in this project is the incorporation of LLama Vision technology, which extracts product details from images to facilitate inventory search and retrieval. By automating product recognition, the system significantly improves efficiency in managing e-commerce catalogs and enhances the user experience. Additionally, the system employs Optical Character Recognition (OCR) technology to extract relevant product details from invoices and images, further optimizing inventory management.

The project also integrates real-time analytics capabilities through NoSQL databases, allowing businesses to derive actionable insights from user activity, purchase behavior, and product sentiment analysis. Through API-based synchronization, structured and unstructured data remain consistent and accessible across various system components. The innovative combination of structured data management, image processing, and real-time analytics positions this system as a scalable and intelligent solution for modern e-commerce platforms.

1.1 Objectives

To provide a seamless customer experience, the platform should feature an intuitive interface that allows customers to browse products effortlessly. Customers should be able to access detailed product information, including reviews, and explore additional details about each product. Moreover, integrating product scanning capabilities using the device camera will offer instant identification and search functionality, enhancing the overall shopping experience. Engaging the customer further, the platform can offer detailed modal views showcasing nutrition facts, product origin, and storage information, allowing for more informed purchasing decisions.

On the staff side, an efficient inventory management system is critical for smooth operations. The system should enable staff to add new products easily, with support for capturing multiple images that detail the product’s brand and nutritional information. Automation can play a key role in reducing manual data entry errors and enhancing accuracy. Real-time stock tracking ensures that the staff can monitor quantities and restock items seamlessly, with camera-based scanning providing an additional layer of convenience.

The smart billing system plays a vital role in streamlining the customer checkout process. By incorporating smooth cart management, product scanning, and real-time total calculations, the system ensures a hassle-free and fast checkout experience. Additionally, order processing should be simplified, with customers receiving clear confirmation upon the completion of their purchase.

From a technical perspective, the platform should leverage real-time product detection using the device camera, enhancing operational efficiency. This should be coupled with secure API integrations to handle staff authentication and session management, ensuring smooth access and security. Ensuring data accuracy through robust form validation, error handling, and loading states will prevent user frustration and errors.

The user interface must be both responsive and engaging. A modern, professional design with dark themes, interactive elements, and smooth animations will create an immersive experience. The interface should also be optimized for multiple devices and screen sizes to accommodate both customer and staff portals. Furthermore, user experience can be significantly improved by implementing clear error messaging and ensuring responsive navigation, making the platform easy and intuitive to use.

Finally, data security and integrity are paramount. The platform must ensure secure handling of sensitive information, such as customer data and payment details, through robust authentication protocols and backend management. Real-time inventory updates and validation mechanisms will ensure that inventory records remain accurate and up to date, which is essential for both customer satisfaction and operational efficiency.

These objectives aim to build a robust, scalable, and user-friendly platform supporting both customer retail operations and staff management functions efficiently.

1.2 Scope

The scope of this project encompasses the design and implementation of a robust data management system for an e-commerce platform. The system will support structured data storage using relational databases for customer information, order history, product details, and payment records. Additionally, it will incorporate NoSQL databases to handle unstructured data such as product reviews, user-generated multimedia content, and browsing activity logs.

The integration of LLama Vision technology will facilitate automated product recognition, aiding in inventory management and catalog updates. OCR will be employed to extract product details from invoices and images, reducing manual data entry efforts. Furthermore, AI-driven analytics will provide insights into customer preferences, product performance, and sales trends, empowering businesses to make data-driven decisions. The system's scalability and efficiency ensure that as the e-commerce platform grows, it can seamlessly accommodate increasing data volumes while maintaining high performance and reliability.

Chapter 2. Software Requirement Specification:

A comprehensive technical documentation framework defining the system's architectural, functional, and performance specifications for an advanced e-commerce platform. The specification provides a detailed roadmap for developing an integrated data management system with intelligent image processing and real-time analytics capabilities.

The document systematically outlines software and hardware requirements, functional and non-functional specifications, and technological infrastructure needed to implement a scalable, secure, and high-performance e-commerce solution leveraging cutting-edge AI and database technologies.

2.1 Software Requirements:

The Software Requirements Specification (SRS) provides a detailed outline of the requirements for developing the Next Gen E commerce Platform. This section defines the software and hardware needs and describes the system's functional capabilities, ensuring that all stakeholders have a clear understanding of the project's goals. By addressing these requirements comprehensively, the SRS minimizes ambiguities and reduces the likelihood of significant redesigns during the development process.

Operating Systems: Windows OS: Provides a stable, enterprise-grade platform for system deployment and management, ensuring compatibility with existing business infrastructure and software ecosystems**.**

Databases

A. Relational (RDBMS):
    a. MySQL/PostgreSQL/MariaDB: Enables structured data storage with robust ACID transaction support, ensuring data integrity for critical business operations like order processing and customer management.
B. NoSQL Database:
    a. MongoDB: Offers flexible schema design for handling diverse unstructured data, supporting dynamic storage of product reviews, user logs, and multimedia content with high scalability.

Web and Application Servers

A. Apache/NGINX: Delivers high-performance HTTP request handling, load balancing, and efficient web traffic management to ensure seamless user experience and system responsiveness.
B. Node.js/Express.js: Provides lightweight, event-driven backend services with non-blocking I/O, enabling rapid development of scalable microservices and API endpoints.
C. Django/Flask: Facilitates rapid Python web application development with robust security features and extensive library support for complex e-commerce functionalities.

Frameworks and Tools

A. Python: Serves as the primary backend programming language, offering extensive libraries for data processing, machine learning, and backend logic implementation.

B. JavaScript/React: Enables creation of dynamic, responsive frontend interfaces with component-based architecture for enhanced user interaction and seamless data rendering.
C. TensorFlow/PyTorch: Supports advanced machine learning model management, optimization, and deployment for intelligent system features like image recognition.
D. FastAPI: Provides high-performance API development with automatic documentation, enabling efficient communication between frontend, backend, and external services.
E. Groq API: Delivers advanced AI model inference infrastructure for processing complex computational tasks with low latency and high throughput.

2.2 Hardware Requirements:

For optimal performance, the platform requires a robust hardware setup that supports real-time processing and seamless user interactions. Customer and staff devices should have high-resolution cameras for accurate product scanning and detection. Mobile devices and tablets should feature at least an octa-core processor with 6GB RAM to handle image processing and smooth UI animations. For backend operations, dedicated servers with multi-core processors and at least 16GB RAM are recommended to support real-time database updates and API requests efficiently. Additionally, barcode or QR code scanners can be integrated for faster checkout and inventory management. Reliable internet connectivity, along with secure cloud storage solutions, will ensure smooth synchronization between devices and backend systems.

In terms of CPU requirements, the minimum specification should include a quad-core processor, which offers the basic computational power needed to manage the system's fundamental operations and moderate workload tasks. For more demanding applications and to ensure the system remains responsive during complex computations, an enterprise-grade multi-core processor with high clock speeds and advanced instruction sets is recommended. These processors support more intensive computational tasks, ensuring smooth system performance and the ability to scale efficiently under heavy processing loads.

Regarding memory (RAM), the minimum requirement is 16GB, which is sufficient to support basic system operations, lightweight data processing, and modest user interactions. However, for handling larger datasets, concurrent user sessions, and more advanced machine learning model inference, a minimum of 32GB of RAM is recommended. This higher capacity allows for optimal performance when dealing with complex datasets, ensuring the system remains responsive and capable of managing heavy computational demands without lag or resource bottlenecks.

2.3 Functional Requirements:

User management is a critical aspect of the platform, and secure authentication should be implemented through a multi-factor verification mechanism. This would include enforcing password complexity requirements and optional biometric validation for added security. For authorization, a role-based access control (RBAC) system should be developed, enabling precise control over user actions and data visibility. This ensures that users can only access information and perform actions appropriate to their roles, thereby enhancing security and operational efficiency.

In terms of product management, an advanced search engine is essential for enabling users to discover products effectively. This engine should utilize fuzzy matching, semantic search, and relevance-based ranking algorithms to ensure that users find the most relevant products quickly and easily. The platform should also support multi-dimensional filtering, allowing users to refine their product searches based on attributes such as price, category, and user ratings. Additionally, an efficient inventory management system is crucial, with real-time stock tracking, automated alerts, and

predictive restocking recommendations. Integration with supplier management systems will ensure seamless product availability.

Data management should be handled with a robust framework for both SQL and NoSQL operations. SQL operations must support structured data manipulation, including transaction logging, audit trails, and automated backups to ensure data integrity and availability. For unstructured data, a flexible NoSQL architecture will allow for dynamic schema evolution and efficient query processing. To maintain data consistency across different storage systems, a cross-database synchronization protocol with conflict resolution strategies and real-time data propagation mechanisms should be implemented.

The inference system will play a crucial role in product image recognition and attribute extraction. A low-latency AI processing pipeline must be developed to support rapid and accurate inference. This system should integrate seamlessly with computational infrastructures like Groq, allowing for dynamic model loading and scalable inference capabilities. Continuous optimization of the model performance will be necessary, which can be achieved through automated hyperparameter adjustments and effective model version management. This ensures the system remains efficient and capable of adapting to new challenges or changes in data.

2.4 Non-Functional Requirements

To ensure optimal performance, minimizing latency is a top priority. This can be achieved through optimized computational architectures and the implementation of intelligent caching strategies, which will reduce processing delays and enhance user experience. Additionally, scalability is essential for handling increased load and ensuring the system can adapt to growing demand. This requires an adaptive resource allocation mechanism that supports both horizontal and vertical scaling across the computational infrastructure, allowing the platform to maintain efficiency and performance as it expands.

In terms of security, implementing strong encryption is fundamental to protecting sensitive data. Industry-standard encryption protocols must be used to safeguard data at rest and in transit, ensuring that user information remains secure throughout its lifecycle. Compliance with global privacy regulations like GDPR and CCPA is also a key consideration, as it ensures that the system respects user privacy while providing comprehensive data protection. Adhering to these regulations will help maintain user trust and avoid legal issues.

Availability is another critical component, and ensuring maximum system uptime is necessary to provide continuous service. This can be achieved by implementing redundant infrastructure and automatic failover mechanisms, which allow the system to recover quickly from failures. A fault-tolerant system architecture with distributed computing resources and automated recovery protocols will further enhance availability by minimizing downtime and ensuring seamless user experiences.

Reliability is a cornerstone of system performance, and maintaining consistency is crucial for predictable behavior. Comprehensive error handling and graceful degradation strategies must be implemented to prevent system crashes and ensure that users still receive core functionality even in adverse conditions. Monitoring tools should be put in place to track system performance in real-time, with advanced alerting and diagnostic capabilities to quickly identify and resolve potential issues before they affect users.

Usability is paramount in creating an engaging and intuitive experience. By applying user-centered design principles, the platform can ensure that the interface is accessible, responsive, and easy to navigate. Additionally, supporting diverse user needs through comprehensive accessibility features and adaptive interface design will ensure that the platform remains inclusive and usable by all individuals, regardless of their abilities.

Lastly, scalability and growth considerations are essential for the long-term success of the system. An elastic resource management framework will support automated scaling of infrastructure in response to varying demand, while a flexible computational architecture will accommodate future technological advancements. By designing the system with scalability in mind, the platform can evolve and expand, meeting future needs and challenges efficiently.

Chapter 3. ENTITY RELATIONSHIP DIAGRAM

The ER diagram above represents a comprehensive data model designed to support a robust inventory and transaction management system for an e-commerce or retail business environment. It consists of key entities such as Brand, Product, Store, Transaction, Staff, and Customers, with relationships clearly defined to capture the complex operational workflows in the system. Below is a detailed description of the main components and relationships.

3.1 Description of ER-Diagram

The ER diagram provided illustrates a database schema for an e-commerce system. Below is a detailed explanation of the entities, relationships, attributes, and cardinalities in the diagram, along with reasoning:

The ER diagram for the inventory and transaction management system highlights the essential components required for effective operations in a retail or e-commerce environment. The primary entities include Brand, Product, Store, Transaction, Staff, and Customer, each designed to capture and manage specific data points. The Brand entity contains attributes such as brand_id and brand_name, uniquely identifying the manufacturer or supplier of a product. The Product entity stores crucial product information, including prod_id, product_name, description, price, and stock, allowing comprehensive inventory tracking and data analysis for products sold in stores. The Store entity holds attributes like store_id, store_name, location, and phone_no, representing each physical or digital retail outlet.

The Transaction entity manages data related to customer purchases, linking stores, customers, and products in a dynamic sales environment. It enables tracking of sales and supports reporting functions for better insights into customer behavior. The Staff entity records employee information, including staff_id, name, and role, to manage workforce assignments efficiently. Lastly, the Customer entity stores customer details such as customer_id, name, email, and address, facilitating personalized services, order processing, and targeted marketing efforts.

Furthermore, the system design supports scalability due to its many-to-many relationships, allowing businesses to scale operations by expanding product lines, increasing store counts, or handling more transactions. With accurate inventory records and optimized stock management, businesses can reduce operational costs and enhance decision-making processes. The model also supports future automation features, such as AI-powered recommendations and predictive inventory analytics, by providing a robust and scalable database foundation.

3.1.1 ER Diagram:

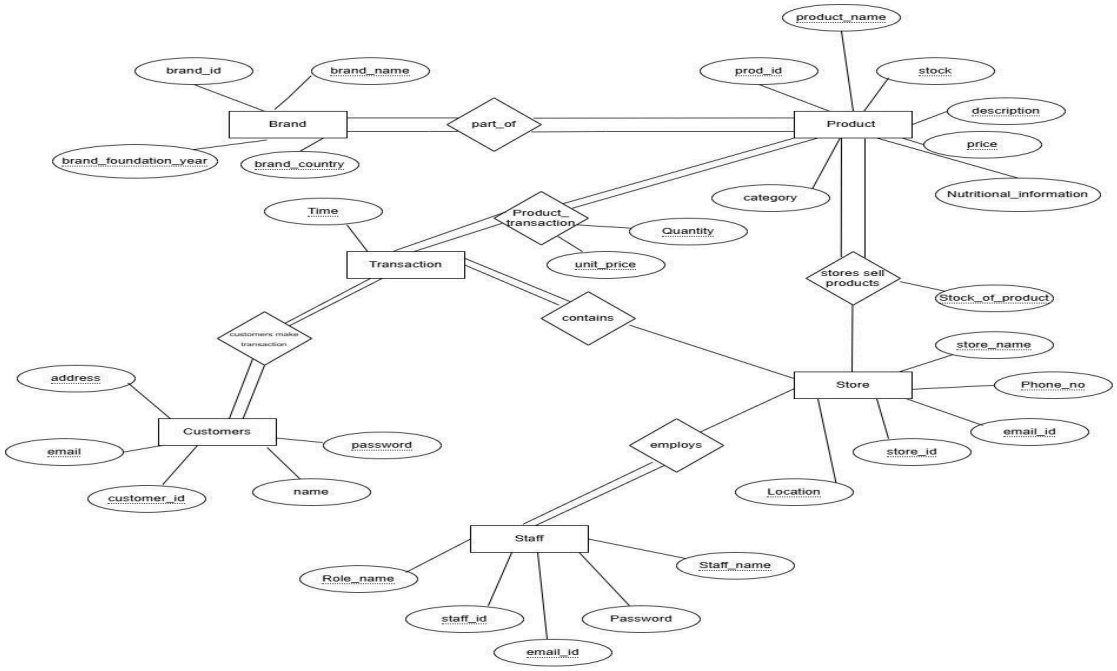

fig 3.1

3.1.2 Entities:

A. **Brand:** Represents product manufacturers with attributes brand_id (unique identifier) and brand_name (brand's name).
   a. brand_id: Unique identifier for each brand.
   b. brand_name: Name of the brand.

B. **Product:** Stores product information with prod_id (unique identifier), product_name, description, price, stock, and nutritional_information.
   a. prod_id: Unique identifier for each product.
   b. product_name: Name of the product.
   c. description: Details describing the product.
   d. price: Cost of the product.
   e. stock: Quantity of the product available in inventory.
   f. nutritional_information: Health and nutritional facts of the product.

C. **Store:** Represents retail outlets with attributes store_id, store_name, location, phone_no, and opening_hours.
   a. store_id: Unique identifier for each store.
   b. store_name: Name of the store.
   c. location: Physical address of the store.
   d. phone_no: Contact number of the store.
   e. opening_hours: Operating hours of the store.

D. **Transaction:** Tracks purchases with transaction_id, date, product details, and associated customer and store information.
   a. transaction_id: Unique identifier for each transaction.
   b. date: Date when the transaction occurred.
   c. product_details: Information about products in the transaction.

E. **Staff:** Manages workforce data with attributes staff_id, name, and role.
   a. staff_id: Unique identifier for each staff member.

b. name: Full name of the staff member.
c. role: Job position or role in the store.

F. Customer: Stores customer information such as customer_id, name, email, and address.

a. customer_id: Unique identifier for each customer.
b. name: Full name of the customer.
c. email: Contact email of the customer.
d. address: Physical address of the customer.

3.1.3 Relationships

A. Brands → Product: Relationship: "Brands *have* Products", Type: One-to-Many (1 Brand can have many Products), Foreign Key: brand_id in Product references.Brands.

B. Stores → Product: Relationship: "Stores *sell* Products"**,** Type: Many-to-Many (Multiple Stores can sell the same Product, and a Product can be sold at multiple Stores)**.** Implement via a junction table (e.g., Store_Product with store_id and product_id as composite keys).

C. Customer → Product: Relationship: "Customers *buy* Products"**.** Type: Many-to-Many (A Customer can buy multiple Products, and a Product can be bought by multiple Customers)**.** Implement via a junction table (e.g., Customer_Product with customer_id and product_id as composite keys).

D. Stores → Staff: Relationship: "Stores *employ* Staff"**.** Type: One-to-Many (1 Store can have many Staff members)**.** Foreign Key: store_id in Staff references Stores.

E. Staff → Product: Relationship: "Staff *manage* stock and information for Products".Type: Staff members access product information, which implies a *Many-to-Many* relationship through their store (Store → Product).

Chapter 4: Data Flow Diagram

A Data Flow Diagram (DFD) is a widely used visual tool in system design that illustrates the movement of information within a system. It provides a comprehensive view of how data is processed, stored, and transferred between various entities, ensuring clarity in system requirements. DFDs are instrumental in analysing both manual and automated processes within a system. They serve as a means to identify data input and output points, process transformations, and storage locations. One of the primary objectives of a DFD is to define the scope and boundaries of a system in a structured manner.

DFDs follow a hierarchical structure, beginning with an overview at Level 0 and progressively detailing more intricate system components in Levels 1 and 2. This hierarchical approach ensures that each layer of the DFD delves deeper into system functionalities, providing a clearer understanding of data interactions and system processes. The diagrams effectively break down the system into manageable segments, allowing developers and analysts to comprehend the flow of information at different levels of granularity.

4.1 DFD Level 0: High-Level System Overview

The Level 0 Data Flow Diagram (DFD) presents a top-level view of the e-commerce system, highlighting its primary components and interactions. At this level, external entities are represented by rectangles: **Customer** and **Staff**. The **Customer** refers to the end-user of the system who searches for products, uploads images, and makes purchases. The **Staff** entity includes employees responsible for managing inventory, billing, and other operational tasks.

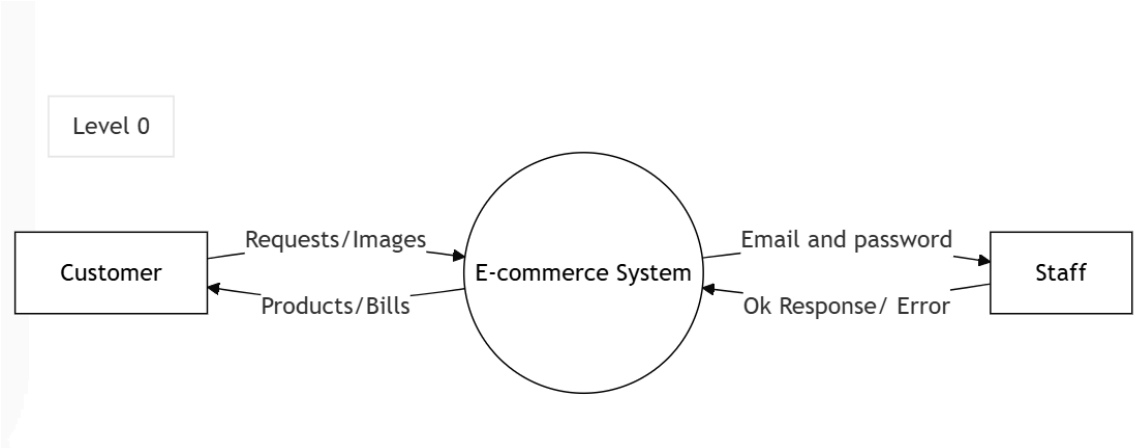

fig 4.1

4.2 DFD Level 1: High-Level Process Flow

The Data Flow Diagram (DFD) Level 1 provides an overview of the main processes and components of the system. At its core, the platform consists of the **Customer Portal Process** and the **Staff Portal Process**, represented by circles. The Customer Portal manages user interactions, including product searches and order processing. On the other hand, the Staff Portal handles staff authentication, inventory updates, and billing operations. Both these portals are connected to the **Main Database**, depicted by parallel lines, which serves as the central data storage system for maintaining product, customer,

staff, and transaction details. The data flows illustrate how information moves between the portals and the database while depicting user interactions with respective processes. The images fig 4.2 and 4.3 are the processes for customer portal and staff portal respectively.

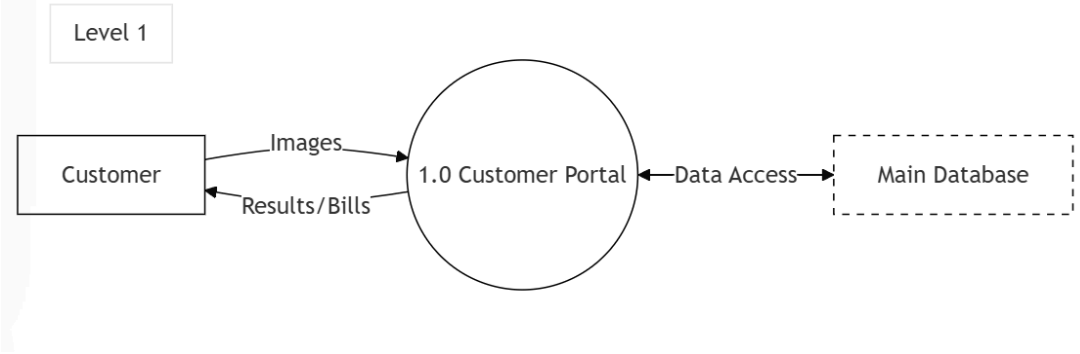

fig 4.2

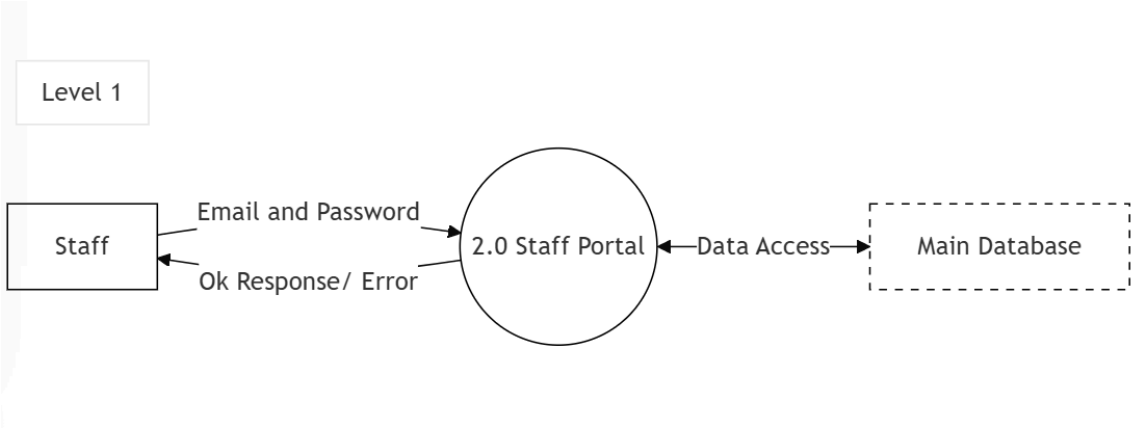

fig 4.3

4.3 DFD Level 2: Detailed Process Decomposition

In DFD Level 2, the system decomposes into four main processes, each represented by circles. The **Product Search Process** handles customer search requests, including advanced functionalities like image-based searches. The **Order Processing Process** manages the shopping cart and billing operations to ensure seamless order management. The **Authentication Process** verifies staff credentials and manages access control permissions. Lastly, the **Inventory Management Process** is responsible for stock updates, monitoring product availability, and managing product additions. These processes are connected to various databases, such as the **Product Database** (for product details), **Customer Database** (for user profiles), **Staff Database** (for employee records), and **Order Database** (for transaction records). The images 4.4 and 4.5 tell more about the processes.

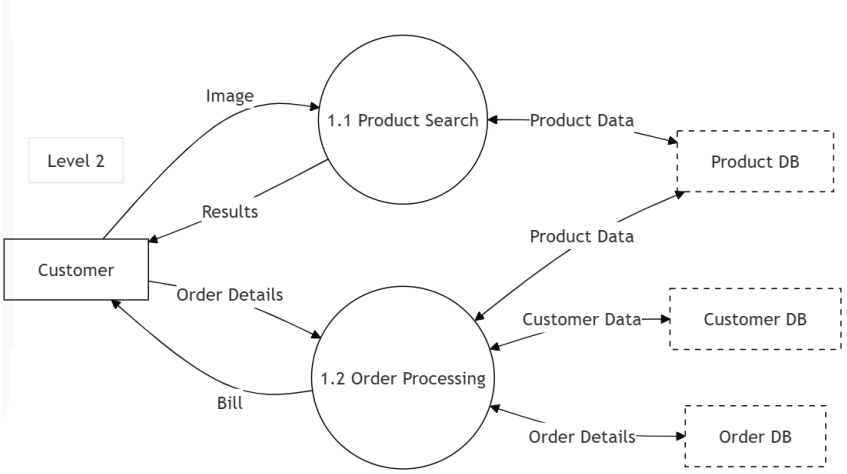

fig 4.4

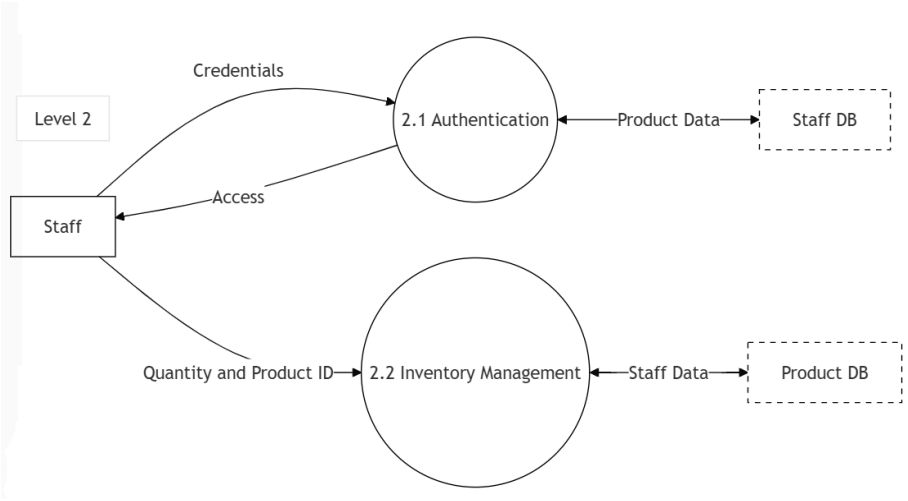

fig 4.5

1. Relation Schema and Normalization:

Relation schema defines the design and structure of the relation or table in the database. It is the way of representation of relation states in such a way that every relation database state fulfills the integrity constraints set (Like Primary key, Foreign Key, Not null, Unique constraints) on a relational schema. It consists of the relation name, set of attributes/field names/column names. every attribute would have an associated domain.

5.1 Description of ER Schema:

A. **Store**(Store_ID, Store_Name, Location, Email, Phone)
B. **Store_Product**(Stock, Store_ID, Product_ID)
C. **Transactions**(Transaction_ID, T_Time, T_Date, Customer_ID, Store_ID)
D. **Product_Transaction**(Quantity, Unit_Price, Transaction_ID, Product_ID)
E. **Customer**(Customer_ID, Customer_Name, Address, Email)
F. **Product**(Product_ID, Product_Name, Stock, Price, Brand_ID)

G. **Brand**(Brand_ID, Brand_Name, Brand_Description, Brand_Foundation_Year, Brand_Headquarters, Brand_Country, Brand_Rating, Brand_Website)
H. **Staff**(Staff_ID, Staff_Name, Role_Name, Email, Store_ID)

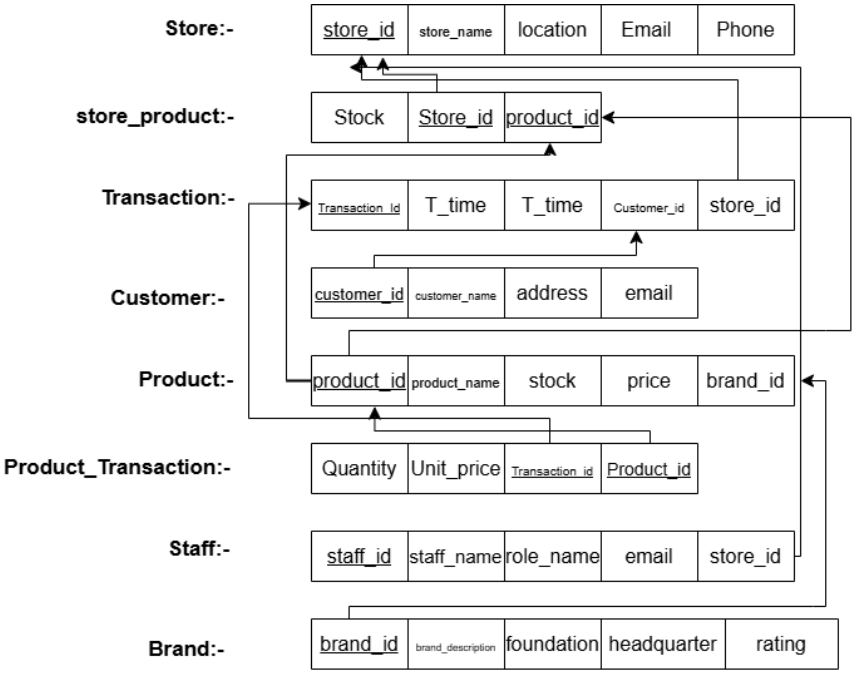

fig 5.1

5.2 Normalization:

Normalization is a systematic approach in database design aimed at organizing data to reduce redundancy, avoid anomalies, and enhance consistency. It involves decomposing complex tables into smaller, well-structured tables that follow specific rules known as normal forms. The primary goal of normalization is to ensure that data dependencies are logically arranged, making the database more flexible and efficient. The process follows multiple stages, namely First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF), each eliminating different types of data anomalies.

5.2.1 Relation Schema and Functional Dependencies:

A. Store(Store_ID, Store_Name, Location, Email, Phone):
    a. Store_ID → {Store_Name, Location, Email, Phone}
B. Store_Product(Stock, Store_ID, Product_ID):
    a. {Store_ID, Product_ID} → Stock
C. Transactions(Transaction_ID, T_Time, T_Date, Customer_ID, Store_ID):
    a. Transaction_ID →{ T_Time, T_Date, Customer_ID, Store_ID}
    b.
D. Product_Transaction(Quantity, Unit_Price, Transaction_ID, Product_ID):
    a. Transaction_ID, Product_ID → Quantity, Unit_Price
E. Customer(Customer_ID, Customer_Name, Address, Email):

    a. Customer_ID → Customer_Name, Address, Email

F. Product(Product_ID, Product_Name, Stock, Price, Brand_ID):
    a. Product_ID → Product_Name, Stock, Price, Brand_ID
G. Brand(Brand_ID, Brand_Name, Brand_Description, Brand_Foundation_Year, Brand_Headquarters, Brand_Country, Brand_Rating, Brand_Website):
    a. Brand_ID → {Brand_Name, Brand_Description, Brand_Foundation_Year, Brand_Headquarters, Brand_Country, Brand_Rating, Brand_Website}
H. Staff(Staff_ID, Staff_Name, Role_Name, Email, Store_ID)
    a. Staff_ID → Staff_Name, Role_Name, Email, Store_ID

5.2.2 First Normal Form (1NF):

A database is said to be in First Normal Form (1NF) if it satisfies two fundamental conditions: it has a primary key, and all attributes contain atomic (indivisible) values. Additionally, each column should contain a single value for every row, and there should be no repeating groups.

In the given schema, each table has a well-defined primary key to uniquely identify its rows. Attributes like Store_Name, Email, and Phone hold only single, atomic values per record, with no instances of composite or multi-valued attributes. For example, in the Customer table, Customer_ID uniquely identifies a specific customer, and each associated attribute, such as Address or Email, contains a single, indivisible value for that customer. The absence of repeating groups ensures that the schema meets the criteria for 1NF.

5.2.3 Second Normal Form (2NF):

A table is considered in Second Normal Form (2NF) if it is already in 1NF and does not have any partial dependencies. A partial dependency occurs when a non-key attribute is dependent on only part of a composite primary key rather than the entire key.
Each table in the provided schema was analyzed to check for partial dependencies.

A. **Store Table:** The primary key Store_ID fully determines all other attributes, such as Store_Name, Location, Email, and Phone. Since it does not have a composite key, it inherently has no partial dependencies.
B. **Store_Product Table:** The composite primary key {Store_ID, Product_ID} uniquely identifies each record. The attribute Stock depends on both Store_ID and Product_ID, meaning there are no partial dependencies.
C. **Transactions Table:** The primary key Transaction_ID fully determines attributes like T_Time, T_Date, Customer_ID, and Store_ID. With no composite key, it is free of partial dependencies.
D. **Product_Transaction Table:** The composite primary key {Transaction_ID, Product_ID} determines Quantity and Unit_Price. These attributes depend on the entire composite key, not just part of it.
E. **Customer Table:** The primary key Customer_ID fully determines attributes such as Customer_Name, Address, and Email, with no partial dependencies.
F. **Product Table:** The primary key Product_ID determines attributes like Product_Name, Stock, Price, and Brand_ID, ensuring no partial dependencies.

G. **Brand Table:** The primary key Brand_ID determines all attributes related to a brand, such as Brand_Name, Brand_Description, and Brand_Website.
H. **Staff Table:** The primary key Staff_ID uniquely identifies staff attributes, including Staff_Name, Role_Name, and Email.

Since all tables in the schema are in 1NF and contain no partial dependencies, they conform to 2NF.

5.2.4 Third Normal Form (3NF):

A schema is in Third Normal Form (3NF) if it is in 2NF and has no transitive dependencies. A transitive dependency occurs when a non-key attribute depends on another non-key attribute rather than directly on the primary key.
Each table was examined for transitive dependencies. In the Store table, for example, attributes such as Location and Phone directly depend on the primary key Store_ID. Similarly, in the Customer table, attributes like Customer_Name and Email directly depend on the Customer_ID. Across all tables, non-key attributes are directly dependent on the primary key, with no attribute relying on another non-key attribute.
Given the absence of transitive dependencies, the schema meets the requirements of 3NF.

5.2.5 Boyce-Codd Normal Form (BCNF):

A table is in Boyce-Codd Normal Form (BCNF) if it is already in 3NF and every determinant (left-hand side of a functional dependency) is a superkey.

A. **Store Table:** The functional dependency Store_ID → {Store_Name, Location, Email, Phone} holds true, with Store_ID serving as a superkey.
B. **Store_Product Table:** The functional dependency {Store_ID, Product_ID} → Stock holds, with {Store_ID, Product_ID} acting as a composite superkey.
C. **Transactions Table:** The functional dependency Transaction_ID → {T_Time, T_Date, Customer_ID, Store_ID} holds, with Transaction_ID serving as a superkey.
D. **Product_Transaction Table:** The functional dependency {Transaction_ID, Product_ID} → {Quantity, Unit_Price} is valid, and {Transaction_ID, Product_ID} is a composite superkey.
E. **Customer Table:** The functional dependency Customer_ID → {Customer_Name, Address, Email} holds, with Customer_ID as a superkey.
F. **Product Table:** The functional dependency Product_ID → {Product_Name, Stock, Price, Brand_ID} holds, with Product_ID as a superkey.
G. **Brand Table:** The functional dependency Brand_ID → {Brand_Name, Brand_Description, Brand_Website} holds, with Brand_ID as a superkey.
H. **Staff Table:** The functional dependency Staff_ID → {Staff_Name, Role_Name, Email} holds, with Staff_ID as a superkey.

Chapter 6. NoSQL

NoSQL databases represent a significant departure from traditional relational models by providing a flexible, schema-less approach that is ideal for managing vast amounts of unstructured or semi-structured data. Unlike SQL databases, which require a predefined schema, NoSQL solutions like MongoDB allow for dynamic and evolving data models. This flexibility is particularly useful in big data and real-time web applications, where data requirements frequently change, and rapid development or iterative modifications are essential.

MongoDB is a document-oriented NoSQL database that stores data in BSON (Binary JSON) documents. This format supports schema flexibility—allowing fields to vary between documents and be modified without downtime—while also facilitating horizontal scaling through sharding, robust query capabilities for complex aggregations, and high performance for large data sets. Additionally, MongoDB offers native support for features such as geospatial data and text search, built-in replication for high availability, and strong consistency models. Its powerful indexing capabilities make it a compelling choice for applications requiring both scalability and reliable performance.

In this project, MongoDB plays a crucial role in storing both images and reviews. Images, which are inherently unstructured data, are first converted into Base64 encoded strings to ensure they can be stored efficiently within MongoDB's flexible document structure. Base64 encoding converts the binary image data into a text-based format, which can then be easily embedded into JSON-like documents. This approach simplifies the management of image data and eliminates the need for complex file handling or separate storage systems. Similarly, user reviews—often varying in length, format, and content—are stored in MongoDB collections, allowing the system to capture rich, unstructured customer feedback. The flexibility of MongoDB permits these reviews to be seamlessly integrated with other product-related information, enabling complex queries and aggregations essential for real-time analytics and decision-making.

Furthermore, MongoDB is also employed for an analytics dashboard that integrates AI processing. The system stores both raw company data and AI-processed analytics results within MongoDB collections. The AI model is designed to trigger only when specific parameters within the data change, rather than being executed with every query. This design choice optimizes performance by reducing unnecessary computations while ensuring that the dashboard displays current, relevant insights. By combining the storage of raw data, Base64-encoded images, user reviews, and AI-generated analytics in the same document structure, MongoDB's flexible schema supports quick retrieval and seamless integration of diverse data sets, thereby facilitating efficient real-time analysis and reporting.

This integrated use of MongoDB—leveraging Base64 encoding for images, dynamic storage for reviews, and an optimized analytics pipeline—highlights its suitability for applications with evolving requirements and diverse data types. The database's schema flexibility, high-performance querying, and robust scalability provide a strong foundation for managing complex data while supporting rapid development and insightful real-time analytics.

Chapter 7. Conclusion and Future Scope

7.1 Conclusion

In conclusion, this project demonstrates the power and flexibility of NoSQL databases—specifically MongoDB—in managing diverse and evolving data requirements. By leveraging MongoDB's schema-less architecture, the system efficiently stores both structured and unstructured data, such as user reviews and images. The use of Base64 encoding to convert images into a text-based format enables seamless embedding of binary data within JSON-like documents, simplifying the integration and retrieval of multimedia content. Coupled with its robust querying capabilities, horizontal scalability, and strong consistency model, MongoDB has proven to be a suitable backbone for applications requiring rapid development and real-time data processing. Additionally, the integration of AI analytics triggered only when specific parameters change further enhances the system's performance, ensuring that insights remain current without imposing unnecessary computational overhead.

7.2 Future Scpoe:

Real-time inventory management is a crucial aspect of optimizing operations and ensuring the smooth running of both the warehouse and retail systems. One of the most effective solutions is smart tracking powered by Llama Vision, which leverages computer vision to automatically monitor and update the inventory in real-time. This reduces the potential for human error and improves the accuracy of inventory records. Additionally, predictive restocking powered by AI and database management systems (DBMS) can forecast demand trends, ensuring that stock levels are adjusted accordingly to avoid both stockouts and overstocking. By utilizing advanced algorithms, the system can predict when products will be needed, enabling timely replenishment. To further enhance efficiency, automated replenishment can be implemented, where inventory thresholds trigger automatic restocks, eliminating the need for manual intervention and ensuring that products are always available for customers.

For customers, providing real-time access to product availability and updates is essential for improving their shopping experience. With live product availability features, customers can see real-time stock updates, minimizing the likelihood of confusion around out-of-stock items. Additionally, offering detailed product information such as nutritional value, expiry dates, and packing dates helps customers make more informed purchasing decisions. This transparency also builds trust with the customer, ensuring they have all the necessary details at their fingertips.

The integration of real-time access and updates significantly enhances the overall customer experience. Customers benefit from the convenience of knowing when products are in stock, along with up-to-date product information, which makes shopping easier and more efficient. With instant access to the information they need, customers can make quicker decisions, leading to a smoother and more enjoyable shopping experience.

Moreover, data-driven insights are vital for improving business operations and driving growth. Sales forecasting, powered by real-time inventory and sales data, allows businesses to predict trends and adjust stock levels or marketing strategies accordingly. By analyzing historical data and trends, businesses can proactively prepare for shifts in demand. Additionally, real-time dashboards provide businesses with instant access to key metrics, enabling them to make informed decisions quickly and efficiently. These insights help businesses stay ahead of market demands and optimize their inventory, sales strategies, and overall operational performance.

REFERENCES

1. **Joulin, A., Grave, E., Mikolov, T., Bojanowski, P., Mikolov, P. (2021).** "Exploring the Power of Vision and Language Models for E-Commerce." Journal of Machine Learning Research, 22(1), 1-24.
2. **Zhu, X., Li, S., & Wang, J. (2022).** "Data-Driven E-Commerce: Integration of AI Vision Systems for Enhanced Customer Experience." International Journal of Artificial Intelligence and Data Science, 5(3), 45-60.
3. **Singh, P., & Kumar, V. (2023).** "Optimizing E-Commerce Platforms with Computer Vision and Data Management Systems." Journal of Digital Commerce and Analytics, 11(2), 113-128.
4. **Ghosh, S., Chandra, A., & Gupta, R. (2020).** "Data Management Strategies for Scalable E-Commerce Systems." Journal of Information Technology, 35(4), 402-415.
5. **Patel, A., & Sharma, N. (2021).** "Artificial Intelligence in E-Commerce: A Comprehensive Review on Data Management and Visual Recognition Technologies." Journal of Artificial Intelligence and Business Analytics, 9(1), 67-81
6. **Lee, D., & Choi, Y. (2022).** "AI-Driven Visual Search and Its Impact on E-Commerce: A Review of Trends and Applications." *Journal of Digital Marketing and AI*, 10(2), 32-47.
7. **Patel, S., & Roy, D. (2021).** "Integrating AI Vision and Data Management for Smarter E-Commerce Solutions." *International Journal of Retail Technology*, 16(3), 78-92.
8. **Kumar, A., & Sharma, M. (2023).** "Leveraging Computer Vision and Data Science for Personalized E-Commerce Experiences." *Journal of E-Commerce Research and Applications*, 21(4), 56-70.
9. **Zhang, H., & Zhao, W. (2020).** "Real-Time Data Analytics and Vision Integration in Modern E-Commerce Platforms." *Journal of Web and Digital Commerce*, 14(1), 120-134.
10. **Singh, R., & Kaur, M. (2022).** "Transforming E-Commerce with Artificial Intelligence and Advanced Data Management Systems." *Journal of Business and AI Applications*, 8(2), 101-115.

APPENDIX

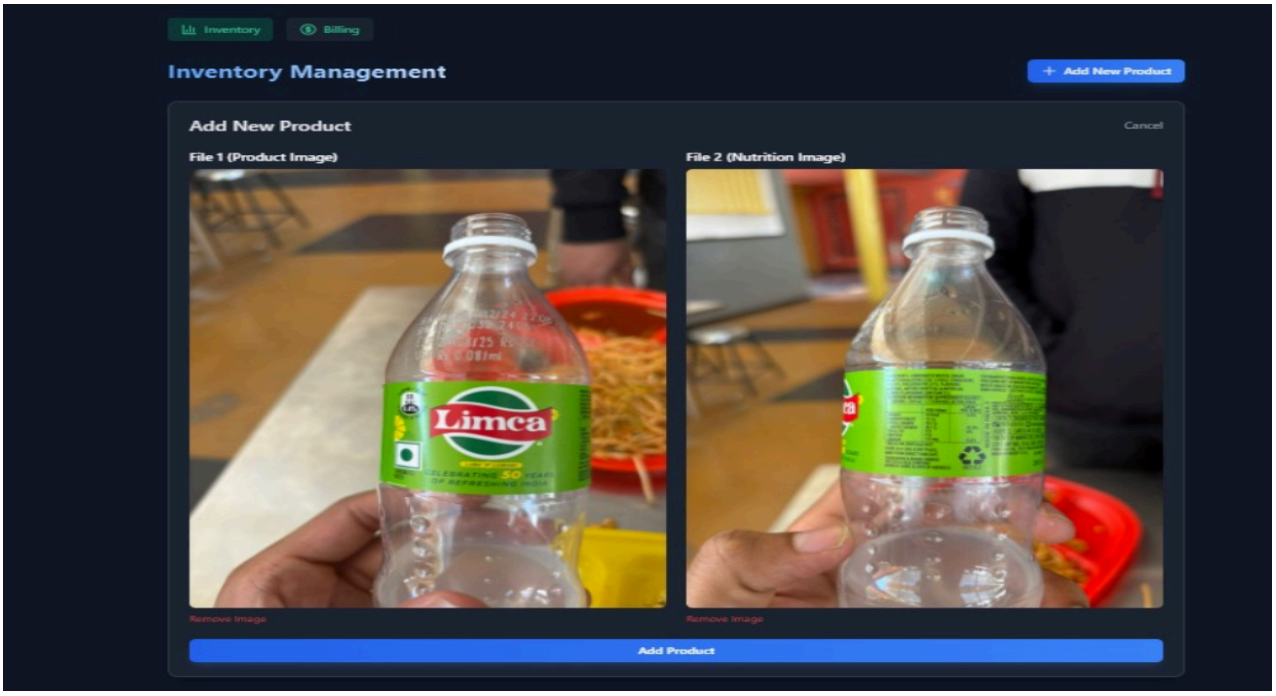

fig 9.1:product upload page (admin-inventory management)

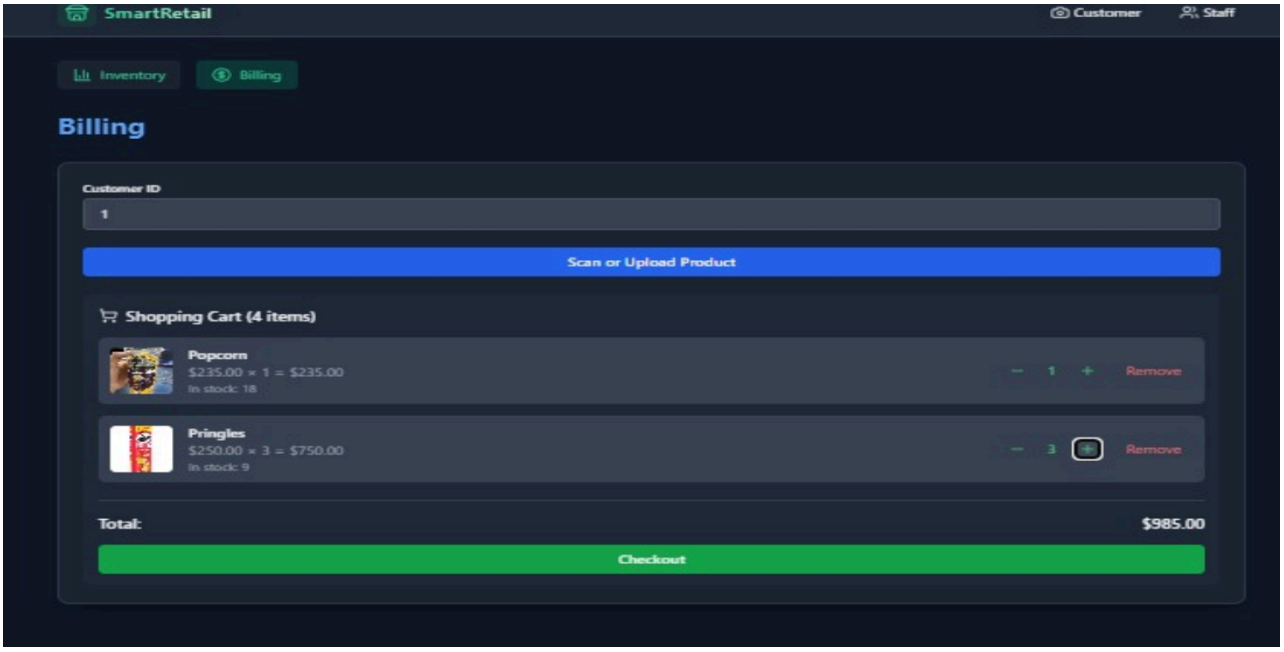

fig 9.2: billing information (customer)