STORE MANAGEMENT

Team: Raghul D

           Vasanthabalan M

ABSTRACT:

This Project is about Store Management and this paper proposes an architecture that involves both *connected* and *disconnected* components, with separate interfaces for administrators and customers. The system allows store *owners*(admin) to manage their product id, product name, quantity are available, cost, and customer information, as well as track customer products and generate Total Amount of their purchasing product. *Customers* can use the system to view products, place orders by using product id and quantity, and check order status with total amount , So that the customer can buy a product and pay the purchasing products . The connected architecture involves a central database that stores all information, while the disconnected architecture involves individual databases on each store location that are periodically synced with the central database. This approach provides both scalability and fault tolerance. The system also incorporates security features such as encryption and authentication to ensure that sensitive information is protected. The proposed architecture offers a flexible and efficient solution for managing store operations in a modern retail environment.

Admin Flow:

The admin flow in a store management system typically involves logging into the admin interface, where they can perform various tasks such as adding new products, product id, product name, quantity are available, cost, and customer information, as well as track customer products and generate Total Amount of their purchasing product.

Customer Flow:

On the other hand, the customer flow in a store management system involves visiting the customer-facing interface, typically an online store or mobile app. Customers can browse products, view product details, add items to their cart, and check out. They can also view their order history and check the status of their current orders. Some store management systems may also allow customers to provide reviews and ratings for products they have purchased, which can help other customers make informed decisions.

DB SCHEMA:

CustomerProd
- customer_id
- product_id
- product_quatity
- cost

Customer
- 🔑 customer_id
- customer_name
- phone
- email

ProductData
- 🔑 product_id
- product_name
- quantity_available
- cost