**STORE MANAGEMENT SYSTEM**

**Purpose of the Store Management System**

The Store Management System is a Spring Boot-based project designed to provide an efficient and user-friendly platform for managing a store's inventory, sellers, products and orders data. This application aims to simplify the process of managing a store by providing an easy-to-use interface that allows users to manage all aspects of their store's operations.

**Reasoning Behind Database Design Choices**

The Store Management System uses a MySQL database to store and manage data related to sellers, products, stocks and orders. The following are the key database design choices that were made while developing this application:

**Entity Relationship Diagram (ERD)**

The ERD for the Store Management System consists of five primary entities: Products, Seller, Orders, Stocks and Users.

The **Product** entity represents the various products available for sale in the store.

The **Seller** entity represents the various ssellers(brands) available for sale in the store.

The **Orders** entity represents individual users who purchase products from the store.

The **Users** entity represents the user whether as an admin or user. Admin has an access to add the sellers, products, stocks and orders the products.

The **Stocks** entity represents the quantity of the products available for sale in the store.

A Seller can have multiple Products

A Product can have multiple Sales.

A User can add multiple Sellers and Products.

A User can order multiple Products.

**CRUD Operation:**

**Create/Insert:** User creates/adds sellers and products data into the database using the **Add-seller, Add-Sellers, Add-Product and Add-Products API’s**.

**Read/Retrieve:** User can read the sellers and products data from the database using the **get-sellers, get-products, seller-item and fetch-product API’s**.

**Update:** User can update the stock of the product into the database using the **add-stocks API** and order the product using the **order-item API**.

**Delete:** User can delete the seller from the database using the **remove-sellers API**.

**Conclusion**

In summary, the database design choices for the Store Management System were made with the goal of ensuring data consistency, reducing redundancy, and improving data integrity. By using a normalized database schema, the application is able to efficiently manage the store's sales, products, stock data while minimizing the risk of errors and inconsistencies.