

Project Name:
ONLINE STORE
Project Charter®

Project:	Online Store
Title:	Project Charter
Version	0.1
Document status:	Draft
	Ravjot Singh : 10042347
Author:	
Responsible:	Ravjot Singh : 10042347
Date created:	10.10.2023
Protection class:	"For internal use only"

Document history

Version	Date	Author	Comment / Change
0.1	10.10.2023		Draft
0.2	21.11.2023		Draft

Contents

	Page
1 Background/Project purpose or justification.....	4
2 Goals	5
2.1 Goals.....	6
2.2 Milestones	6
3 Project product description	6
4 Delivery units	6
4.1 Delivery units/services	7
4.2 E-R Diagram.....	7
4.3 Database Creation Using SQL Commands.....	8
4.4 DML Commands.....	10
5 Project success criteria.....	16
6 High-level risks	17
7 Key stakeholders	18
8 Project startup	18
9 Project end	19
9.1 Signatures for release.....	
Annexture	

1. Background/Project purpose or justification

The initial purpose of the Online Store Database Project is to improve the general functionality and efficiency of the online store. The Online Grocery Store project aims to create a user-friendly and efficient platform for customers to conveniently browse, select, and purchase grocery items from the comfort of their homes. We are going to build a database that can help us keep track of our inventory and our customers information. There have been times that due to mis-management within the inventory and number of orders, our team got inefficient and had to deal with various problems. With the rapid rise of e-commerce and rising customer demands, an organization must spend in modernizing and optimizing its database architecture. This projects targets stores inventory management, customers details and orders status by designing and implementing an efficient database.

2. Goals

Here are some key goals highlighting the importance of a database in the context of an online store.

2.1 Goals

Goal	Description
Data Storage:	A large amount of data is generated and managed by an online business, including product information, customer information, sales transactions, and inventory levels. A database is required to efficiently store and organize this data.
Product Catalogue:	A database is used to keep track of a product catalogue, which includes names, descriptions, pricing, photos, and stock levels. Customers may successfully browse and search for products as a result of this.
Inventory Management:	Having inventory data in a database allows the store to track product availability, manage restocking, and avoid overselling.
Order Processing:	When a customer makes a purchase, their order details must be stored and maintained in a database. This comprises the ordered items, customer information, payment information, and order status.
Customer Information:	An online store database saves customer profiles, purchase histories, shipping addresses, and contact information, allowing for personalized shopping experiences and efficient customer care.
Shopping Cart:	A shopping cart is a temporary storage area for things that a client wishes to purchase that is frequently included as part of a database system. It keeps the items chosen for purchase until the buyer is ready to checkout.
Payment Processing:	To support secure financial transactions, payment data and transaction records are securely stored within the database.
Reporting and Analytics:	Data on customer behavior, sales trends, and website performance can be collected using a well-structured database. This information is essential for making sound business decisions and optimizing the online store.
Security and Compliance:	Databases play an important role in ensuring data security and compliance with privacy requirements. They include techniques for encrypting sensitive data, restricting access, and auditing data modifications.

2.2 Milestones

Schedule	Description
October 12	Project Charter draft
November 9	E-R diagram draft
December 7	Video Project description
December 7	Final report delivery

3. Project product description

The project's product is the Database, which will be utilized to store important data about online business. The Online Store Database is the online store's backbone, coordinating the management, retrieval, and storage of important data required for the online store to function. It contains all information on users, carts, products, and reviews, making it a trustworthy and efficient platform for people to browse, buy, and engage.

The Database will store the information as:

USERS: UserID, Name, Address, Contact Information etc.

PRODUCTS: Product ID, name, description, price, stock, quantity etc.

CART : cart ID, user ID, Product ID, quantity, etc.

REVIEWS: Review ID, User ID , product ID, rating, comment, date etc.

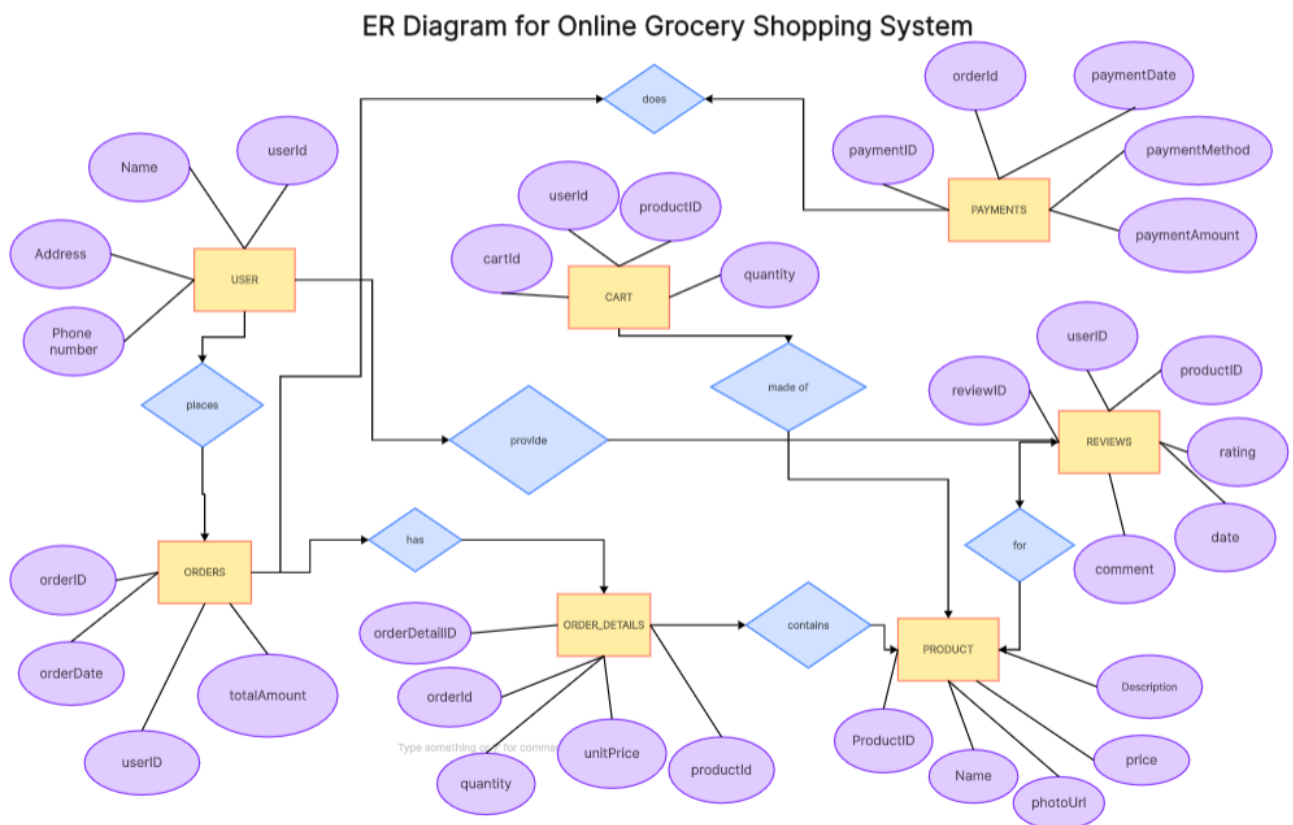
As far as the user is concerned, The Online Grocery Store project seeks to build a user-friendly and efficient platform for customers to browse, select, and purchase grocery items from the comfort of their own homes. Key features include a User-Friendly Interface that will assist users in seamless navigation across multiple devices for a tailored shopping experience, as well as an Extensive Product Catalogue that includes a Comprehensive listing of grocery items with detailed descriptions, images, and nutritional information for easy exploration, including fresh produce, dairy, beverages, and household goods.

4. Delivery Units:

4.1 Delivery units/services

Delivery unit	Description/Comment
Project Charter	Final version of the charter approved
Project Report	A concise document that compiles your whole experience with the project
Design document	E-R Diagram describing the conceptual level of the database
DDL documents	All SQL statements used to create the database (Internal level)
Sample Queries	DML commands showing sample Queries used in regular use of the database
Supplements	Additional documents providing background in the chosen topic. (Optional)

4.2 E-R Diagram



4.3- Database Creation Using SQL Commands

-- Create the Online Grocery Store schema

```
CREATE SCHEMA OnlineGroceryStore;
```

-- PRODUCTS Table

```
CREATE TABLE OnlineGroceryStore.PRODUCTS (  
    ProductID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Description TEXT,  
    Price DECIMAL(10, 2),  
    Stock INT,  
    PhotoURL VARCHAR(255)  
);
```

-- USERS Table

```
CREATE TABLE OnlineGroceryStore.USERS (  
    UserID INT PRIMARY KEY,  
    Name VARCHAR(255),  
    Address VARCHAR(255),  
    ContactInformation VARCHAR(255)  
);
```

-- ORDERS Table

```
CREATE TABLE OnlineGroceryStore.ORDERS (  
    OrderID INT PRIMARY KEY,  
    UserID INT,  
    OrderDate DATE,  
    TotalAmount DECIMAL(10, 2),  
    FOREIGN KEY (UserID) REFERENCES OnlineGroceryStore.USERS(UserID)  
);
```

-- ORDER_DETAILS Table


```

CREATE TABLE OnlineGroceryStore.ORDER_DETAILS (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT,
    UnitPrice DECIMAL(10, 2),
    FOREIGN KEY (OrderID) REFERENCES OnlineGroceryStore.ORDERS(OrderID),
    FOREIGN KEY (ProductID) REFERENCES OnlineGroceryStore.PRODUCTS(ProductID)
);

```

-- PAYMENTS Table

```

CREATE TABLE OnlineGroceryStore.PAYMENTS (
    PaymentID INT PRIMARY KEY,
    OrderID INT,
    PaymentDate DATE,
    PaymentAmount DECIMAL(10, 2),
    PaymentMethod VARCHAR(50),
    FOREIGN KEY (OrderID) REFERENCES OnlineGroceryStore.ORDERS(OrderID)
);

```

-- REVIEWS Table

```

CREATE TABLE OnlineGroceryStore.REVIEWS (
    ReviewID INT PRIMARY KEY,
    UserID INT,
    ProductID INT,
    Rating INT,
    Comment TEXT,
    Date DATE,
    FOREIGN KEY (UserID) REFERENCES OnlineGroceryStore.USERS(UserID),
    FOREIGN KEY (ProductID) REFERENCES OnlineGroceryStore.PRODUCTS(ProductID)
);

```

-- CART Table

```

CREATE TABLE OnlineGroceryStore.CART (
    CartID INT PRIMARY KEY,
    UserID INT,

```

```

ProductID INT,
Quantity INT,
FOREIGN KEY (UserID) REFERENCES OnlineGroceryStore.USERS(UserID),
FOREIGN KEY (ProductID) REFERENCES OnlineGroceryStore.PRODUCTS(ProductID));

```

4.4 DML Commands

-- Insert sample data into the PRODUCTS table

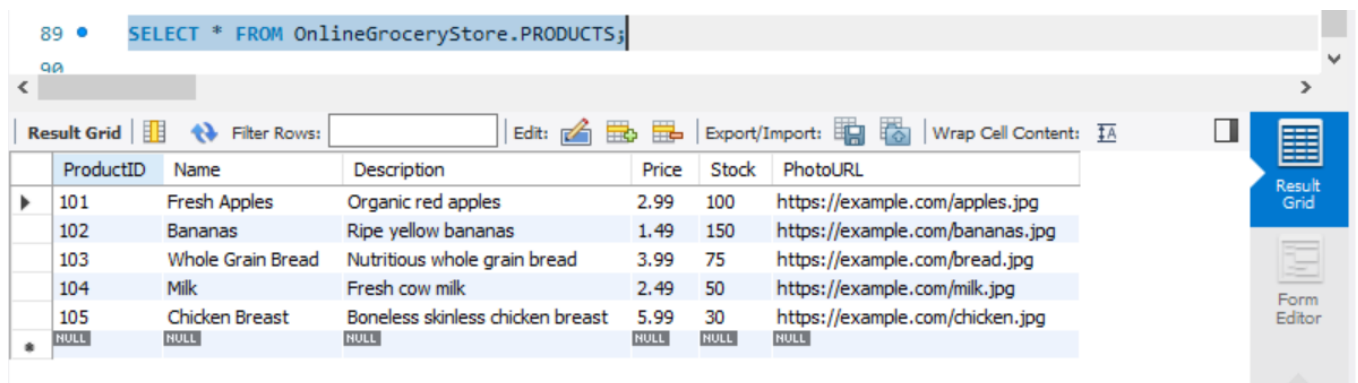
```

INSERT INTO OnlineGroceryStore.PRODUCTS (ProductID, Name, Description, Price, Stock, PhotoURL)
VALUES
(101, 'Fresh Apples', 'Organic red apples', 2.99, 100, 'https://example.com/apples.jpg'),
(102, 'Bananas', 'Ripe yellow bananas', 1.49, 150, 'https://example.com/bananas.jpg'),
(103, 'Whole Grain Bread', 'Nutritious whole grain bread', 3.99, 75,
'https://example.com/bread.jpg'),
(104, 'Milk', 'Fresh cow milk', 2.49, 50, 'https://example.com/milk.jpg'),
(105, 'Chicken Breast', 'Boneless skinless chicken breast', 5.99, 30,
'https://example.com/chicken.jpg' );

```

-- Get details of all products in the catalog.

```
SELECT * FROM OnlineGroceryStore.PRODUCTS;
```



The screenshot shows a database query tool interface. At the top, the SQL command `SELECT * FROM OnlineGroceryStore.PRODUCTS;` is entered in a text area. Below the text area, a toolbar contains various icons for editing and exporting. The main area displays a table with the following data:

ProductID	Name	Description	Price	Stock	PhotoURL
101	Fresh Apples	Organic red apples	2.99	100	https://example.com/apples.jpg
102	Bananas	Ripe yellow bananas	1.49	150	https://example.com/bananas.jpg
103	Whole Grain Bread	Nutritious whole grain bread	3.99	75	https://example.com/bread.jpg
104	Milk	Fresh cow milk	2.49	50	https://example.com/milk.jpg
105	Chicken Breast	Boneless skinless chicken breast	5.99	30	https://example.com/chicken.jpg
NULL	NULL	NULL	NULL	NULL	NULL

-- Retrieve Available Products:

- Get products that are currently in stock over 70 in number.

```

SELECT * FROM OnlineGroceryStore.PRODUCTS
WHERE Stock > 70;

```

```

89 • SELECT * FROM OnlineGroceryStore.PRODUCTS
90 WHERE Stock > 70;
91

```

Result Grid

ProductID	Name	Description	Price	Stock	PhotoURL
101	Fresh Apples	Organic red apples	2.99	100	https://example.com/apples.jpg
102	Bananas	Ripe yellow bananas	1.49	150	https://example.com/bananas.jpg
103	Whole Grain Bread	Nutritious whole grain bread	3.99	75	https://example.com/bread.jpg
NULL	NULL	NULL	NULL	NULL	NULL

-- Insert a new user into the USERS table

```

INSERT INTO OnlineGroceryStore.USERS (UserID, Name, Address, ContactInformation)
VALUES (1, 'John Doe', '123 Main St', '555-1234');

```

```

SELECT * FROM OnlineGroceryStore.USERS;

```

```

89 • SELECT * FROM OnlineGroceryStore.USERS;
90

```

Result Grid

UserID	Name	Address	ContactInformation
1	John Doe	123 Main St	555-1234
NULL	NULL	NULL	NULL

-- Place an Order:

- **Create a new order for a user.**

```

INSERT INTO OnlineGroceryStore.ORDERS (OrderID, UserID, OrderDate, TotalAmount)
VALUES (1, 1, CURRENT_DATE, 50.99);

```

-- Add products to the order details.

```

INSERT INTO OnlineGroceryStore.ORDER_DETAILS (OrderDetailID, OrderID, ProductID,
Quantity, UnitPrice)
VALUES (1, 1, 101, 2, 10.99),
      (2, 1, 102, 3, 8.99);

```

```

SELECT * FROM OnlineGroceryStore.ORDER_DETAILS

```

95 • `SELECT * FROM OnlineGroceryStore.ORDER_DETAILS`

96

<

Result Grid | Filter Rows: | Edit: | Export/Import:

	OrderDetailID	OrderID	ProductID	Quantity	UnitPrice
▶	1	1	101	2	10.99
	2	1	102	3	8.99
*	NULL	NULL	NULL	NULL	NULL

-- Change the price of a specific product.

```
UPDATE OnlineGroceryStore.PRODUCTS
SET Price = 12.99
WHERE ProductID = 101;
```

-- Get the order history for a specific user.

```
SELECT * FROM OnlineGroceryStore.ORDERS
WHERE UserID = 1;
```

92 • `SELECT * FROM OnlineGroceryStore.ORDERS`

93 `WHERE UserID = 1;`

<

Result Grid | Filter Rows: | Edit: | Export/Import:

	OrderID	UserID	OrderDate	TotalAmount
▶	1	1	2023-12-08	50.99
*	NULL	NULL	NULL	NULL

-- Post a Review:

- **Add a review for a product.**

```
INSERT INTO OnlineGroceryStore.REVIEWS (ReviewID, UserID, ProductID, Rating,
Comment, Date)
VALUES (1, 1, 101, 4, 'Great product!', CURRENT_DATE);
```

```
SELECT * FROM OnlineGroceryStore.REVIEWS
```

94 • `SELECT * FROM OnlineGroceryStore.REVIEWS`

95 `WHERE UserID = 1`

Result Grid

	ReviewID	UserID	ProductID	Rating	Comment	Date
▶	1	1	101	4	Great product!	2023-12-08
*	NULL	NULL	NULL	NULL	NULL	NULL

-- Adding 2 units of product with ProductID 101 and 3 units of product with ProductID 102 to the cart for User with UserID 1

```
INSERT INTO OnlineGroceryStore.CART (CartID, UserID, ProductID, Quantity)
VALUES
  (1, 1, 101, 2),
  (2, 1, 102, 3);
```

--Retrieve items in a user's shopping cart.

```
SELECT * FROM OnlineGroceryStore.CART
WHERE UserID = 1;
```

96

97 • `SELECT * FROM OnlineGroceryStore.CART`

98 `WHERE UserID = 1;`

Result Grid

	CartID	UserID	ProductID	Quantity
▶	1	1	101	2
	2	1	102	3
*	NULL	NULL	NULL	NULL

--Remove an item from the shopping cart.

```
DELETE FROM OnlineGroceryStore.CART
WHERE CartID = 1;
```

```

98 • DELETE FROM OnlineGroceryStore.CART
99 WHERE CartID = 1;
100 • SELECT * FROM OnlineGroceryStore.CART
101 WHERE UserID = 1;

```

Result Grid | Filter Rows: | Edit: | Export/Import:

	CartID	UserID	ProductID	Quantity
▶	2	1	102	3
*	NULL	NULL	NULL	NULL

-- Update Stock After Placing an Order

```

UPDATE OnlineGroceryStore.PRODUCTS p
SET p.Stock = p.Stock - (
    SELECT od.Quantity
    FROM OnlineGroceryStore.ORDER_DETAILS od
    WHERE od.ProductID = p.ProductID
)
WHERE EXISTS (
    SELECT 1
    FROM OnlineGroceryStore.ORDER_DETAILS od
    WHERE od.ProductID = p.ProductID
);

```

```
SELECT * FROM OnlineGroceryStore.PRODUCTS
```

```
97 • SELECT * FROM OnlineGroceryStore.PRODUCTS
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	ProductID	Name	Description	Price	Stock	PhotoURL
▶	101	Fresh Apples	Organic red apples	12.99	96	https://example.com/apples.jpg
	102	Bananas	Ripe yellow bananas	1.49	144	https://example.com/bananas.jpg
	103	Whole Grain Bread	Nutritious whole grain bread	3.99	75	https://example.com/bread.jpg
	104	Milk	Fresh cow milk	2.49	50	https://example.com/milk.jpg
	105	Chicken Breast	Boneless skinless chicken breast	5.99	30	https://example.com/chicken.jpg
*	NULL	NULL	NULL	NULL	NULL	NULL

-- Get the average rating for each product based on reviews.

```

SELECT p.ProductID, p.Name, AVG(r.Rating) AS AverageRating
FROM OnlineGroceryStore.PRODUCTS p
LEFT JOIN OnlineGroceryStore.REVIEWS r ON p.ProductID = r.ProductID

```

GROUP BY p.ProductID, p.Name;

```
101 • SELECT p.ProductID, p.Name, AVG(r.Rating) AS AverageRating
102 FROM OnlineGroceryStore.PRODUCTS p
103 LEFT JOIN OnlineGroceryStore.REVIEWS r ON p.ProductID = r.ProductID
104 GROUP BY p.ProductID, p.Name;
```

ProductID	Name	AverageRating
101	Fresh Apples	4.0000
102	Bananas	NULL
103	Whole Grain Bread	NULL
104	Milk	NULL
105	Chicken Breast	NULL

-- Calculate the total amount spent by a user across all orders.

```
SELECT u.UserID, u.Name, SUM(o.TotalAmount) AS TotalAmountSpent
FROM OnlineGroceryStore.USERS u
LEFT JOIN OnlineGroceryStore.ORDERS o ON u.UserID = o.UserID
GROUP BY u.UserID, u.Name;
```

```
100
101 • SELECT u.UserID, u.Name, SUM(o.TotalAmount) AS TotalAmountSpent
102 FROM OnlineGroceryStore.USERS u
103 LEFT JOIN OnlineGroceryStore.ORDERS o ON u.UserID = o.UserID
104 GROUP BY u.UserID, u.Name;
```

UserID	Name	TotalAmountSpent
1	John Doe	50.99

-- Insert payment data for an order

```
INSERT INTO OnlineGroceryStore.PAYMENTS (PaymentID, OrderID, PaymentDate,
PaymentAmount, PaymentMethod)
VALUES
(1, 1, CURRENT_DATE, 50.99, 'Credit Card');
```

```
SELECT * FROM OnlineGroceryStore.PAYMENTS
```

108 • `SELECT * FROM OnlineGroceryStore.PAYMENTS`

< 109

Result Grid Filter Rows: Edit: Export/Import:

	PaymentID	OrderID	PaymentDate	PaymentAmount	PaymentMethod
▶	1	1	2023-12-08	50.99	Credit Card
*	NULL	NULL	NULL	NULL	NULL

- Assume a refund of \$10.00 for OrderID 1
- Update the original payment amount for OrderID 1

```
INSERT INTO OnlineGroceryStore.PAYMENTS (OrderID, PaymentDate, PaymentAmount,
PaymentMethod)
SET PaymentAmount = PaymentAmount - 10.00
WHERE OrderID = 1;
```

97 • `SELECT * FROM OnlineGroceryStore.PAYMENTS;`

98

< 99

Result Grid Filter Rows: Edit: Export/Import: W

	PaymentID	OrderID	PaymentDate	PaymentAmount	PaymentMethod
▶	1	1	2023-12-08	40.99	Credit Card
*	NULL	NULL	NULL	NULL	NULL

5. Project success criteria

Project success criteria
The Project Report addresses of reasoning used to generate the design of the database. Data needed and information generated.
Internal and conceptual level definitions are free of errors and complete.
The queries created retrieve useful information for the business.
The project presentation conveys a concise summary of the projects experience.
The team shows understanding of the business, conceptual design and MySQL.

6. High-level risks

Risk	Possible impacts on the project
Data Security Breach:	Making weak password, misconfiguration in database where users have more control over the database, third-party vendors that are connected can create data security breach, accidental human error or deletion of data can also lead the data security breach
Data migration difficulties:	Serious data validation needs to be performed, there is a risk of operations being disrupted, and there's a possibility of potential data loss and corruption.
Problems with scalability:	Any incorrect inventory data in the database may result in decreased system performance, loss of sales during heavy traffic periods, and lower client confidence.
Issues with Third-Party Integrations:	Insufficient involvement or feedback from key stakeholders may lead to misalignment with business goals and requirements, potentially causing delays.
bad infrastructure	Will hinder the Scalability and efficiency of the Business.

7. Key stakeholders

Name	Role
Divya Sehdev	Database Administrator
Prachi Sehgal	Database Developer
Jobanpreet Singh	Database Administrator
Ravjot Singh	Database Developer
Cesar Lopez Castellanos	Database Instructor

8. Project startup

The project is deemed started with the following signatures:

	Instructor	Communications Officer	Project manager
Signature			
Name			
Date			

5. Project end

Planned project end:

a. Signatures for release

The project manager is released with the signatures provided here following the project closing phase:

	Instructor	Communications Officer	Project manager
Signature			
Name			
Date			