BINUS UNIVERSITY

BINUS INTERNATIONAL

Database Technology

Final Project

Student Information:		Surname:	Given Name:	Student ID
	1.	Gunawan	Darren Gunawan	2702342256
	2.	Jonathan	Kevin Jonathan	2702342823

Course Code : COMP6799001 **Course Name** : Database Technology

Class : L3BC **Lecturer** : YENNY, S.Kom., M.Kom.

Type of Assignment : Final Project Report

Submission Pattern

Due Date : 15 December 2024 **Submission Date** : 15 December 2024

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.

2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.

3. The above information is complete and legible.

4. Compiled pages are firmly stapled.

5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

Binus International seriously regards all forms of plagiarism, cheating, and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity, and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept, and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been notified and accepted in advance

Signature of Student:

Darren Gunawan

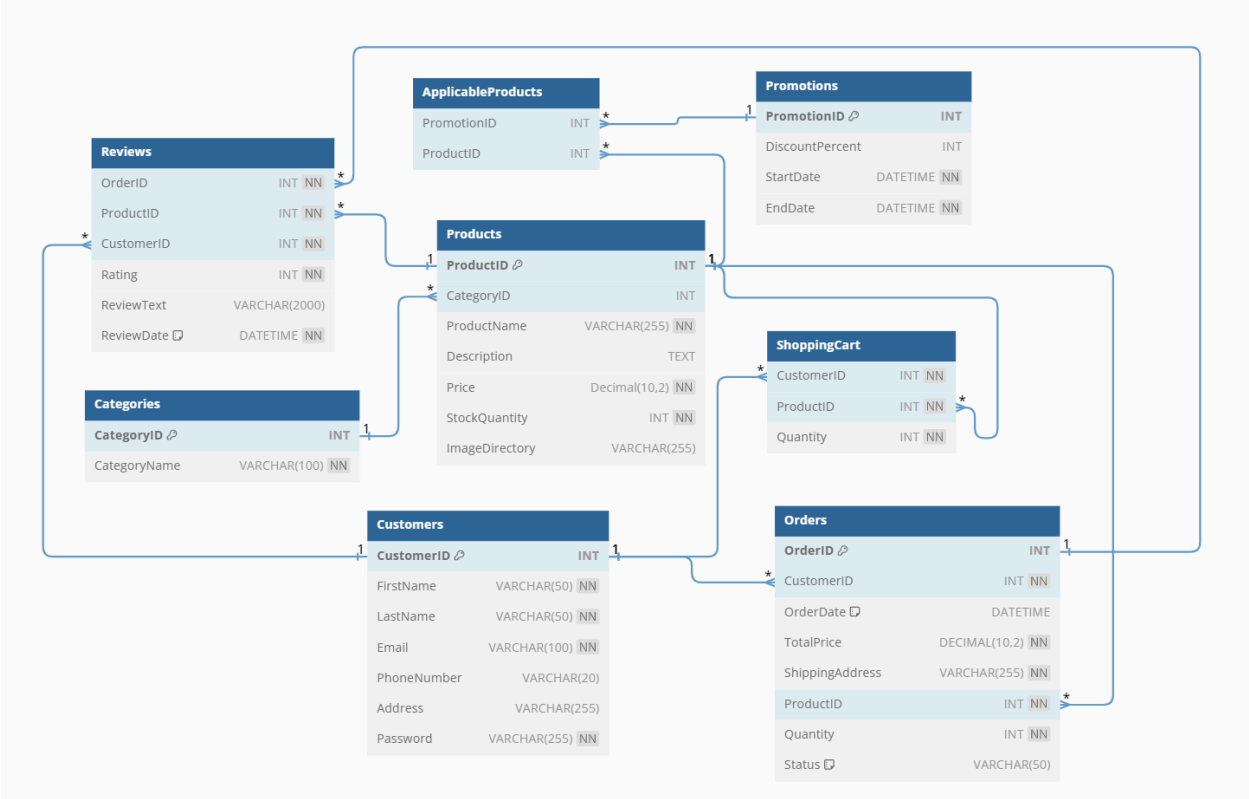
Kevin Jonathan

Table of Contents

I. Introduction

In this digital era where the Internet has become an important part of everyday life, traditional offline businesses are losing sales as many more customers have transitioned and are becoming more accustomed to online businesses. In the United States alone, 44% of female shoppers and 53% of males would rather check over retail items online rather than viewing them in an offline store (Capital One Shopping, 2024). This indicates that there would be less consumers going to an offline store and more will choose to purchase an item through online platforms instead. Other than the amount of customers visiting offline stores decreasing, there are several other challenges that traditional businesses are facing currently. There are 3 common challenges that traditional businesses face and they are inventory mismanagement, lack of accessibility, and limited customer reach. In stores that sell many items, it can be a hassle to keep track of the accurate inventory records for each item. This can lead to overstocking and stockouts, which can result in lost sales and increased cost for the business. Traditional businesses also have a lack of accessibility. Since traditional retail stores are only available in one or few certain places, this makes it so only customers that can physically visit the stores make a purchase. This makes it so the store's customer base is restricted only to the local area around the store. Lastly is limited customer reach, even if a store is locally known, it will be challenging to reach customers beyond the local area they have their store located at. Online retail stores, on the other hand, can have and reach a global audience way easier, which can result in an expansion in the customer base of their store. The rise of online retail stores has provided a solution to these problems by allowing businesses to have streamline operations, and to be accessible at anywhere and any time. This online retail store aims to address these issues by having a platform where customers are able to browse products, add items to their shopping cart, and place orders for these products from any location available. Also this online retail store will have a real time stock management system that will display the accurate amount of stocks available for the product and update it real time. The system objective is to provide a friendly and easy to use interface for the customers to browse and purchase products and offer a real time reliable inventory system that displays accurate stock availability. In addition, the customers will be able to keep track of their orders by seeing their order history and also place reviews on products that they have purchased before.

II. Entity-Relationship Diagram



Entities:

1. **Products**

   The products that are going to be sold in the market.

2. **Categories**

   The categories for the product that will be used to filter down the data.

3. **Customers**

   The main stakeholders for this system, storing data for login purposes.

4. **Reviews**

   Captures the reviews and ratings provided by customers for specific products.

5. **ShoppingCart**

   Products that customers intend to buy.

6. **Orders**

   The purchased products made by the customers.

7. **Promotions**

   Storing the discount for the products.

Relationships:

1. **Products and Categories:** One-to-many relationship.

   A category can have many products, but a product belongs to only one category.

2. **Products and Reviews:** One-to-many relationship.

   A product can have many reviews, but a review is associated with only one product.

3. **Customers and Reviews:** One-to-many relationship.

   A customer can write many reviews, but a review is written by only one customer.

4. **Customers and ShoppingCart:** One-to-many relationship.

   A customer can have many products in the shopping carts, but each shopping cart belongs to only one customer.

5. **Customers and Orders:** One-to-many relationship.

   A customer can place many orders, but an order is placed by only one customer.

6. **ShoppingCart and Products:** Many-to-many relationship.

   A shopping cart can contain many products, and a product can be in many shopping carts.

7. **Orders and Products:** Many-to-many relationship.

   An order can contain many products, and a product can be in many orders.

8. **Promotions and Applicable Products:** One-to-many relationship.

   A promotion can be applied to many products, but a product can only have one promotion applied to it.

Assumptions:

1. **Single Category Assignment:**

   Each product is strictly associated with one category, meaning it cannot be in several categories simultaneously.

2. **Multiple Reviews per Product:**

   The same product can have more than one review, and by different customers. In this way, a comprehensive view can be developed about the customer's response to a particular product.

3. **Multiple Reviews by one Customer:**

   A customer can have more than one review about different products. This gives customers the ability to state their opinions on different kinds of products they may come across.

4. **TotalPrice in Orders table:**

   Storing the price when the user bought the product is crucial, as prices may be updated, the price when it is bought shouldn’t change.

5. **Promotions are Single Used:**

   If the promotion ends, like 11.11 or 12.12, it will need to be recreated for next year's usage. While the promotion may be used for multiple products still, but with the same duration.

III. Normalization

---

1. First Normal Form (1NF)

Ensuring each attribute to be atomic, the attributes stored are atomic and can't be divided anymore. Also, we managed to eliminate repeating groups within the tables. However, we decided to keep an attribute that should be removed as part of calculation as it conflicts with the 1NF. This decision was made to store the historical data related to the total price, as it shouldn't be changed even if the price of the product had been updated.

Orders	**Products**	**ShoppingCart**	**Customers**
OrderID	*ProductID*	*CustomerID*	*CustomerID*
OrderDate	ProductName	*ProductID*	FirstName
TotalPrice	Description	Quantity	LastName
ShippingAddress	Price		Email
Quantity	StockQuantity		PhoneNumber
Status	ImageDirectory		Address
	CategoryID		Password
	CategoryName		
	PromotionID		
	StartDate		
	EndDate		
	DiscountPercent		
	Rating		
	ReviewText		
	ReviewDate		

Notes:
= Primary Key
= Composite Keys

2. Second Normal Form (2NF)

For the next normal form, we separated all the tables that might require a primary key. All non-key attributes should be fully dependent on the primary key. This is done to solve the update anomalies that might be needed when we save the products in the cart and update them, and also when trying to update the profile.

Orders	**Customers**	**Products**	**Categories**
OrderID	*CustomerID*	*ProductID*	*CategoryID*
CustomerID	FirstName	CategoryID	CategoryName
ProductID	LastName	ProductName	
OrderDate	Email	Description	
TotalPrice	PhoneNumber	Price	
ShippingAddress	Address	StockQuantity	
Quantity	Password	ImageDirectory	
Status		PromotionID	

Promotions	**Reviews**	**ShoppingCart**
PromotionID	*OrderID*	*CustomerID*
DiscountPercent	*ProductID*	*ProductID*
StartDate	*CustomerID*	Quantity
EndDate	Rating	
	ReviewText	
	ReviewDate	

Notes:
= Primary Key
= Composite Keys

3. **Third Normal Form (3NF)**

Lastly, we have to eliminate the transitive dependencies, it aims to remove all the redundancy happening in the database. We implemented the 3NF by making sure that data like promotion ID would not be duplicated, while each promotion ID can still apply for many products. Therefore we separated the table as it would have a transitive dependency. This also ensures that all of the attributes in the table are only dependent on the primary key, and eliminates all dependencies and most importantly to reduce the redundancy in the database.

Orders	**Customers**	**Products**	**Categories**
OrderID	CustomerID	*ProductID*	*CategoryID*
CustomerID	FirstName	CategoryID	CategoryName
OrderDate	LastName	ProductName	
TotalPrice	Email	Description	
ShippingAddress	PhoneNumber	Price	
ProductID	Address	StockQuantity	
Quantity	Password	ImageDirectory	
Status			

Applicable Products	**Promotions**	**Reviews**	**ShoppingCart**
PromotionID	*PromotionID*	*OrderID*	*CustomerID*
ProductID	DiscountPercent	*ProductID*	*ProductID*
	StartDate	*CustomerID*	Quantity
	EndDate	Rating	
		ReviewText	
		ReviewDate	

Notes:
= Primary Key
= Composite Keys

IV. Table Structure

Customers Table

Column Name	Data Type	Primary / Foreign Key	Constraints
CustomerID	INT	Primary Key	
FirstName	VARCHAR(50)		Not Null
LastName	VARCHAR(50)		Not Null
Email	VARCHAR(100)		Not Null Email field contains a valid email address format
PhoneNumber	VARCHAR(20)		PhoneNumber field contains only numeric values and has a length between 10 and 15 digits.
Address	VARCHAR(255)		
Password	VARCHAR(255)		Not Null Password field contains at least 8 characters for security purposes

Products Table

Column Name	Data Type	Primary / Foreign Key	Constraints
ProductID	INT	Primary Key	
CategoryID	INT	Foreign Key	Not Null
ProductName	VARCHAR(255)		Not Null
Description	TEXT		
Price	DECIMAL(10,2)		Not Null
StockQuantity	INT		Not Null
ImageDirectory	VARCHAR(255)		

Categories Table

Column Name	Data Type	Primary / Foreign Key	Constraints
CategoryID	INT	Primary Key	
CategoryName	VARCHAR(100)		Not Null

ApplicableProducts Table

Column Name	Data Type	Primary / Foreign Key	Constraints
PromotionID	INT	Foreign Key	Not Null
ProductID	INT	Foreign Key	Not Null

Promotions Table

Column Name	Data Type	Primary / Foreign Key	Constraints
PromotionID	INT	Primary Key	
DiscountPercent	INT		Not Null
StartDate	DATETIME		Not Null
EndDate	DATETIME		Not Null

ShoppingCart Table

Column Name	Data Type	Primary / Foreign Key	Constraints
CustomerID	INT	Foreign Key	Not Null
ProductID	INT	Foreign Key	Not Null
Quantity	INT		Not Null

Orders Table

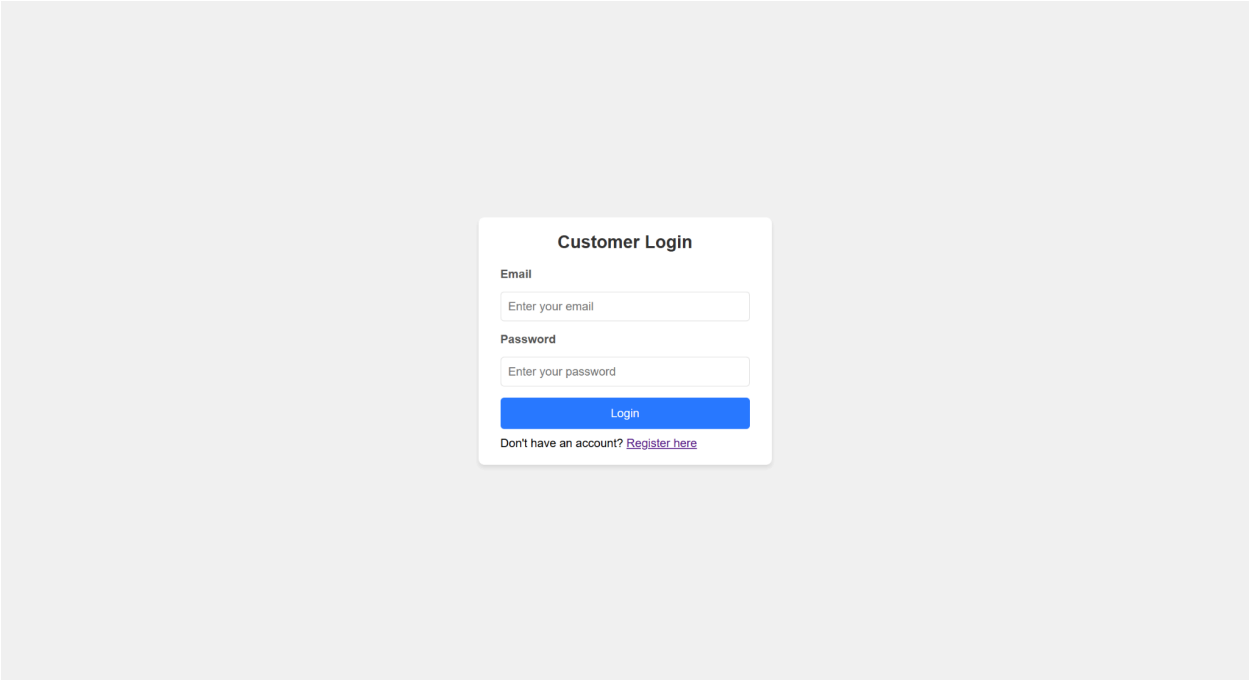
Column Name	Data Type	Primary / Foreign Key	Constraints
OrderID	INT	Primary Key	
CustomerID	INT	Foreign Key	Not Null
OrderDate	DATETIME		Not Null
TotalPrice	DECIMAL(10,2)		Not Null
ShippingAddress	VARCHAR(255)		Not Null
ProductID	INT	Foreign Key	Not Null
Quantity	INT		Not Null
Status	VARCHAR(50)		Not Null

Reviews Table

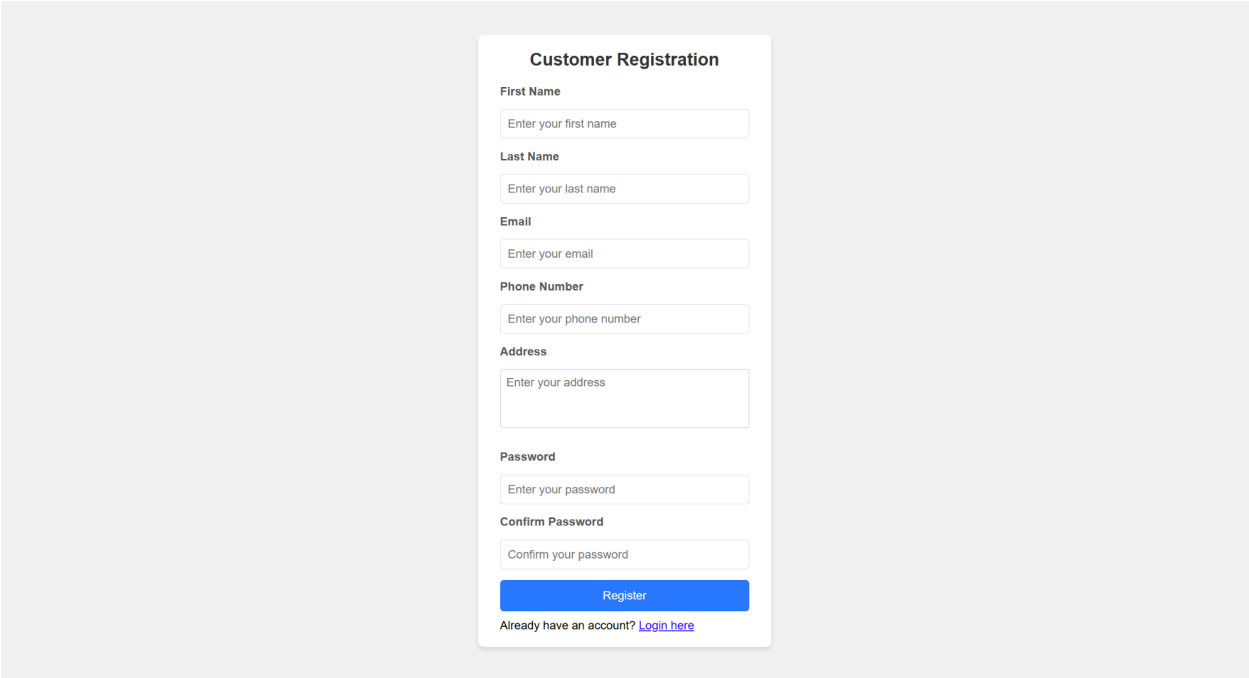
Column Name	Data Type	Primary / Foreign Key	Constraints
OrderID	INT	Foreign Key	Not Null
ProductID	INT	Foreign Key	Not Null
CustomerID	INT	Foreign Key	Not Null
Rating	INT		Not Null Between 1 AND 5
ReviewText	VARCHAR(2000)		
ReviewDate	DATETIME		Not Null

V. User-Interface Design

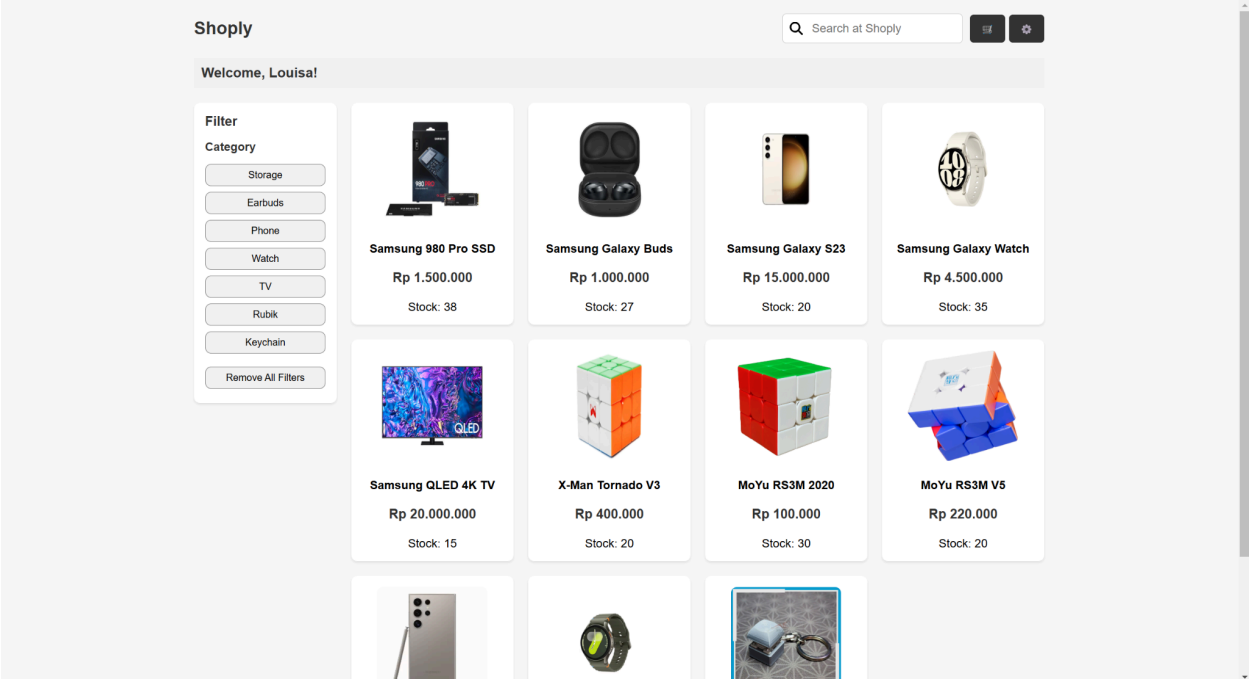
Login Page



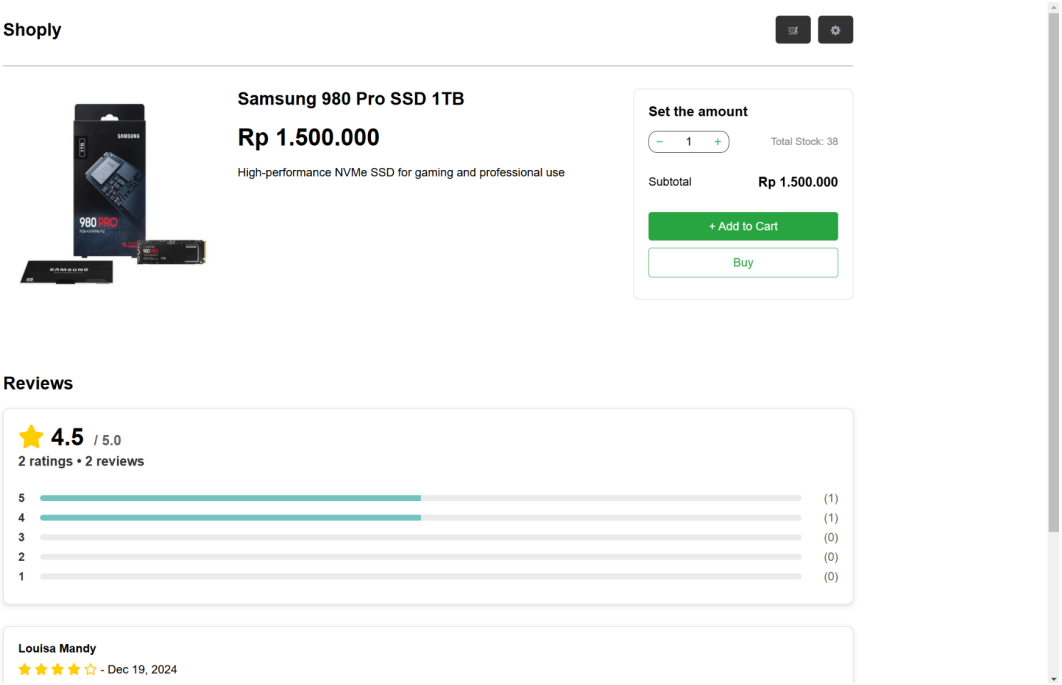
Registration Page



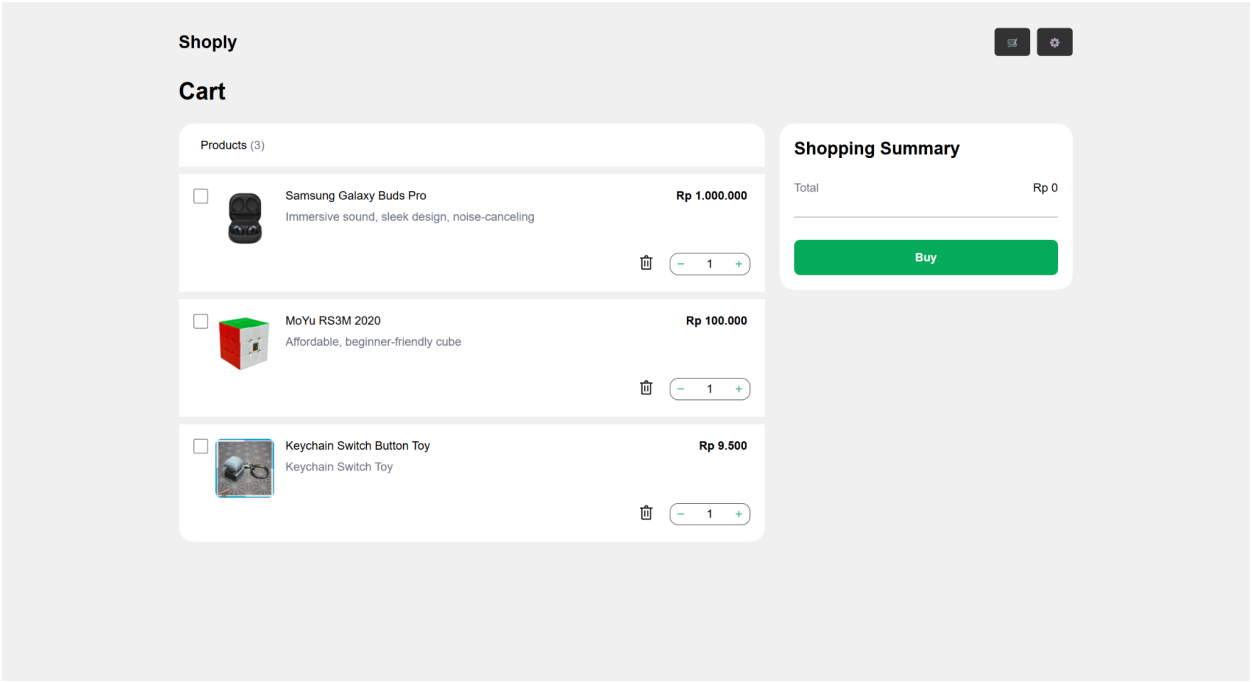
Home Page



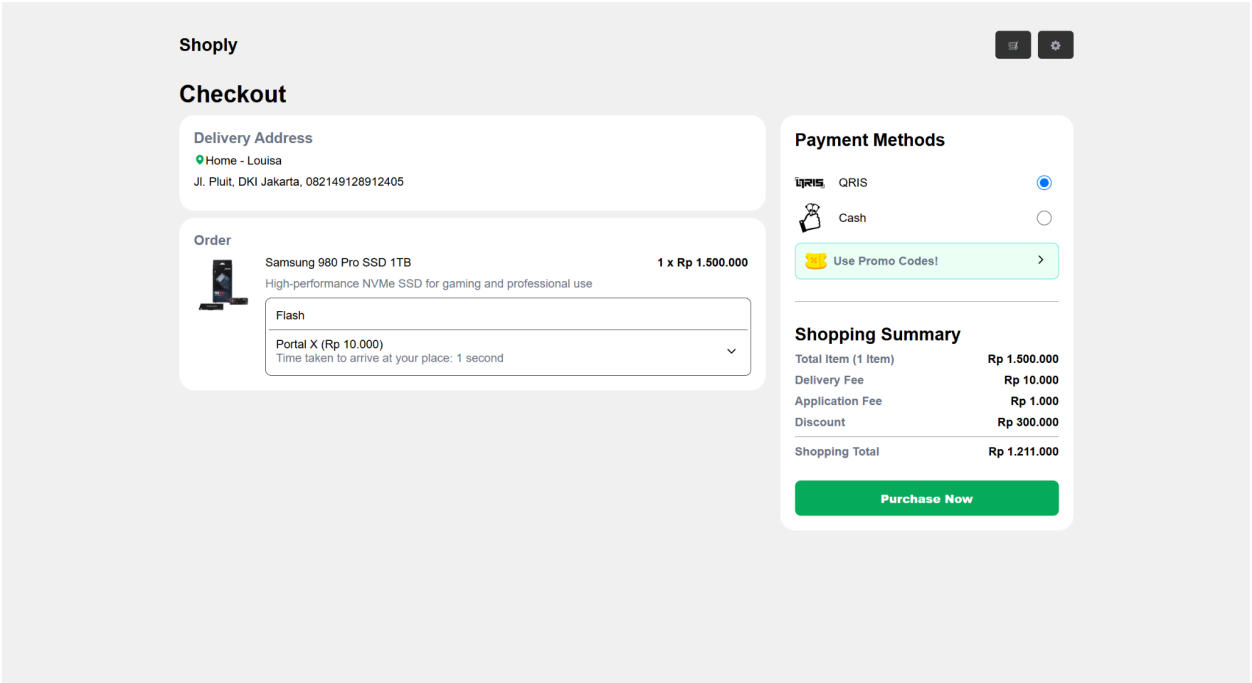
Product Page



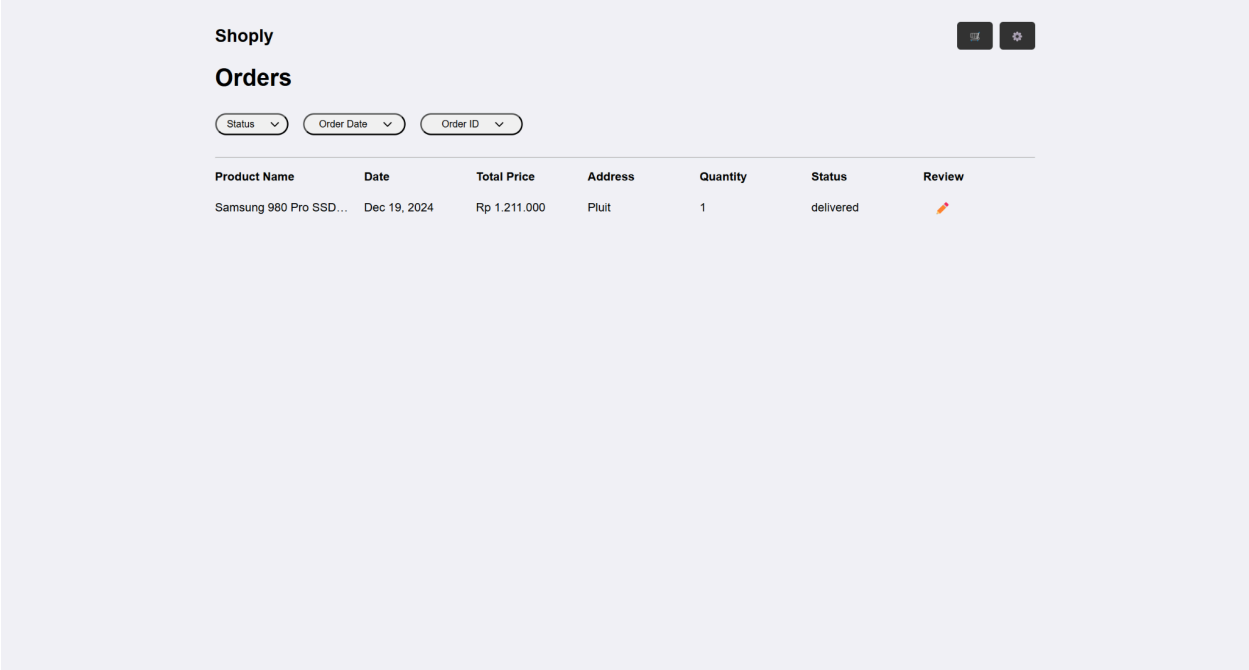
Shopping Cart Page



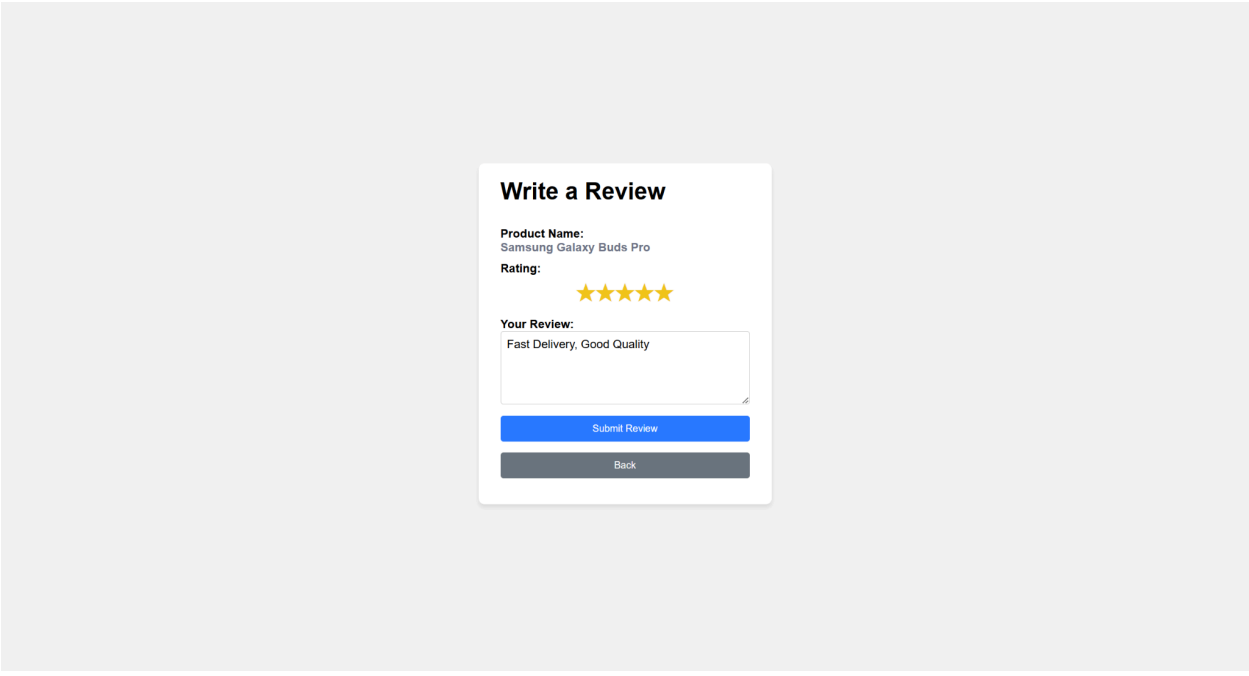
Checkout Page



Orders Page



Reviews Page



Key Features:

1. **Login Page For Security**

   We used a login page to ensure that only authorized users can access the system. It’s used to protect the user's information.

2. **Search For Products**

   This allows users to quickly find specific products within the catalog. It helps the user to navigate easier and find what they’re looking for in seconds, though saving their time.

3. **Filter Products By Category**

   Enables the user to narrow down the catalog by finding which one they choose to explore. It helps them to find the products that are relevant with their interests, also improving the user experience as it organizes the catalog and making it easier for them to browse.

4. **Add Products To Cart**

   Allows users to temporarily store products that they intend to purchase. Enabling the user to easily find the item when they want to purchase it later on.

5. **Purchasing The Products**

   It represents the actual transaction process for the users to buy the items. We also offer various payment methods for users to pick according to their convenience.

6. **Write Reviews**

   This allows users to share their feedback and experiences with the products they bought. It helps other users to be informed and aware about the product before buying. It also fosters a sense of community, where users can interact with others.

7. **Edit Profile**

   We also enable the users to update their personal information and preferences within the application, they can also change their password.

VI. Conclusion

Our key achievements for this project was to first design a database for an online retail store that is normalized. We made sure that each table is following the normalization principles from 1NF to 3NF, so that there will be less redundancy of data and improve data integrity. The database system is able to store, display, and modify customer data, customers' orders, product details, and many more. In addition, we also made a functional, responsive, and user-friendly interface to demonstrate user operations such as data entry, retrieval, and display of information from the database. It allows the users to create an account and login, browse over all the store's products seamlessly, buy products, view order history, and also to write down reviews for each product they have ever purchased.

For the challenges we encountered, deciding the tables we needed and normalizing the tables was a very big challenge for us. Identifying all the important entities and relationships between each entity is very important and we also need to analyze them to ensure that it captures all the processes that are required in an online retail store. Normalizing the tables also made us confused because there were some scenarios where breaking the normalization rules seemed more practical. For example, in the "Orders" table there is a column called "TotalPrice". Putting calculations in a table can result in redundancy and potential for inconsistencies, but in our scenario, we wanted to store the value of the total price of the products that the users ordered with the price of the products during that time. If the total price was calculated using queries instead of storing the value in a table, when the price of the product changes, the total price that the users pay for that product will also change and that is completely wrong. That is why we decided to store the value of total price inside the "Orders" table. Making the user interface and connecting the database to the UI was also not an easy task to do.

Despite all the challenges, we have learned many valuable things from doing this project. For hard skills, we learned how to normalize tables, how to write queries in MySQL, and also how to connect the database onto a webpage. For soft skills, we learned that time management and communication skills are important in a group work setting to make sure that all members are working at a good pace to finish the project on time. By mentioning the communication

skills, we meant to highlight the importance of having communication tools, such as WhatsApp that is frequently used nowadays, by reading the chats and replying to the chats would be a great progress. We also learned that it is better to have less members that actually work rather than having more group members, but they do not contribute anything to the team. Overall this project was fun and would be useful for us if we are working with databases again in the future.

VII. Appendix

Data Definition Language

```
CREATE TABLE Products(
ProductID INT AUTO_INCREMENT PRIMARY KEY,
CategoryID INT,
ProductName VARCHAR(255) NOT NULL,
Description TEXT,
Price Decimal(10,2) NOT NULL,
StockQuantity INT NOT NULL,
ImageDirectory VARCHAR(255),
FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);
```

```
CREATE TABLE Customers(
    CustomerID INT AUTO_INCREMENT PRIMARY KEY,
    FirstName VARCHAR(50) NOT NULL,
    LastName VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    PhoneNumber VARCHAR(20),
    Address VARCHAR(255),
    Password VARCHAR(255) NOT NULL, -- Added password field
    CONSTRAINT Check_CustomerEmail CHECK (Email REGEXP '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'),
    CONSTRAINT Check_CustomerPhoneNumber CHECK (PhoneNumber REGEXP '^[0-9]{10,15}$'),
    CONSTRAINT Check_PasswordStrength CHECK (CHAR_LENGTH(Password) >= 8) -- Ensure a minimum length of 8 characters
);
```

```
CREATE TABLE Orders (
    OrderID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT NOT NULL,
    OrderDate DATETIME DEFAULT CURRENT_TIMESTAMP,
    TotalPrice DECIMAL(10, 2) NOT NULL,
    ShippingAddress VARCHAR(255) NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    Status VARCHAR(50) DEFAULT 'processed',
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

```sql
CREATE TABLE ShoppingCart (
    CartID INT AUTO_INCREMENT PRIMARY KEY,
    CustomerID INT NOT NULL,
    ProductID INT NOT NULL,
    Quantity INT NOT NULL,
    -- Foreign key relationships with Customers and Products tables
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

```sql
CREATE TABLE Reviews (
    ReviewID INT AUTO_INCREMENT PRIMARY KEY,
    ProductID INT NOT NULL,
    CustomerID INT NOT NULL,
    Rating INT NOT NULL CHECK (Rating BETWEEN 1 AND 5),
    ReviewText VARCHAR(2000),
    ReviewDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT FK_Reviews_Customers FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
    CONSTRAINT FK_Reviews_Products FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);
```

Data Manipulation Language

```
const query = `
                INSERT INTO Orders (CustomerID, OrderDate, TotalPrice, ShippingAddress, ProductID, Quantity, Status)
                VALUES (?, NOW(), ?, ?, ?, ?, ?)
              `;
```

```
db.query(`
    SELECT Reviews.*, Products.ProductName
    FROM Reviews
    JOIN Products ON Reviews.ProductID = Products.ProductID
    WHERE Reviews.CustomerID = ? AND Reviews.ProductID = ?`,
    [customerId, productId], (err, results) => {
        if (err) reject(err);
        else resolve(results);
    });
});
```

```
// Check if the review already exists for this customer and product
const checkQuery = `SELECT * FROM Reviews WHERE CustomerID = ? AND ProductID = ?`;
db.query(checkQuery, [customerId, productId], (err, result) => {
    if (err) {
        console.error('Error checking review:', err);
        return res.status(500).send('Server error');
    }

    // If a review is found, update it
    if (result.length > 0) {
        const updateQuery = `UPDATE Reviews SET Rating = ?, ReviewText = ? WHERE CustomerID = ? AND ProductID = ?`;
        db.query(updateQuery, [rating, reviewText, customerId, productId], (err, result) => {
            if (err) {
                console.error('Error updating review:', err);
                return res.status(500).send('Server error');
            }
            res.redirect('/orders');
        });
    } else {
        const insertQuery = `INSERT INTO Reviews (CustomerID, ProductID, Rating, ReviewText) VALUES (?, ?, ?, ?)`;
        db.query(insertQuery, [customerId, productId, rating, reviewText], (err, result) => {
            if (err) {
                console.error('Error inserting review:', err);
                return res.status(500).send('Server error');
            }
            res.redirect('/orders');
        });
    }
});
```

```
db.query(`
    SELECT ROUND(AVG(Rating), 1) AS averageRating, COUNT(*) AS totalReviews,
    SUM(CASE WHEN Rating = 5 THEN 1 ELSE 0 END) AS fiveStar,
    SUM(CASE WHEN Rating = 4 THEN 1 ELSE 0 END) AS fourStar,
    SUM(CASE WHEN Rating = 3 THEN 1 ELSE 0 END) AS threeStar,
    SUM(CASE WHEN Rating = 2 THEN 1 ELSE 0 END) AS twoStar,
    SUM(CASE WHEN Rating = 1 THEN 1 ELSE 0 END) AS oneStar,
    SUM(CASE WHEN Rating >= 1 AND Rating <= 5 THEN 1 ELSE 0 END) AS totalRatings
    FROM Reviews WHERE ProductID = ?`, [productId], (err, results) => {
    if (err) reject(err);
    else resolve(results);
});
```

```
db.query(`
    SELECT shoppingcart.CartID, shoppingcart.ProductID, shoppingcart.Quantity, products.ProductName,
    products.Price, products.ImageDirectory, products.Description, products.StockQuantity
    FROM shoppingcart
    JOIN products ON ShoppingCart.ProductID = products.ProductID
    WHERE shoppingcart.CustomerID = ?`, [req.session.customerid], (err, results) => {
    if (err) reject(err);
    else resolve(results);
});
```

```
const address = await new Promise((resolve, reject) => {
    db.query(`SELECT Address FROM Customers WHERE CustomerID = ?`, [req.session.customerid], (err, results) => {
        if (err) reject(err);
        else resolve(results);
    });
});

const shippingAddress = address[0].Address;

// Insert the order
const query = `
                INSERT INTO Orders (CustomerID, OrderDate, TotalPrice, ShippingAddress, ProductID, Quantity, Status)
                VALUES (?, NOW(), ?, ?, ?, ?, ?)
              `;

db.query(query, [customerId, price, shippingAddress, productId, quantity, 'delivered'], (err, result) => {
```

```
const updateStockQuery = `UPDATE Products SET StockQuantity = StockQuantity - ? WHERE ProductID = ?`;

db.query(updateStockQuery, [quantity, productId], (err, result) => {
    if (err) {
        console.error('Error updating product stock:', err);
        return res.status(500).send('Server error');
    }

    // Successfully updated stock, redirect to the orders page
    res.redirect('/orders');
});
```

VIII. References

Online shopping demographics (2024): Trends & Total Shoppers. Capital One Shopping. (2024, March 12). https://capitaloneshopping.com/research/online-shopping-demographics/