

# SmartGrocery - Online Grocery E-Commerce System

**COURSE: OOAD**



ID	NAME
65566	Emaz Ali Khan
65540	Abdul Rafay Zahid
65651	Sumaika Asif

DATE OF SUBMISSION: \_\_\_\_\_

# **SmartGrocery-Online Grocery E-Commerce System**

## **SUBMITTED BY**

Emaz Ali Khan (65566)

Abdul Rafay Zahid (65540)

Sumaika Asif (65651)

## **SUPERVISED BY**

Prof. Maqsood Razi



**REPORT SUBMITTED TO THE FACULTY OF  
COMPUTING, KARACHI INSTITUTE OF ECONOMICS  
AND TECHNOLOGY, IN PARTIAL FULFILMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF BACHELOR OF  
SCIENCE IN COMPUTER SCIENCE**

**SPRING 2025**

# Table of Contents

<b>1. Introduction:</b>	4
1.1 Project Problem Background	4
1.2 Risk Analysis	4
1.3 Problem Statement	4
1.4 Objective	5
1.6 Current System	6
1.7 Proposed System (SmartGrocery)	7
<b>2. Requirement Analysis:</b>	7
2.1 Functional Requirements:	7
2.2 Non-Functional Requirements:	7
2.3 User Roles	8
2.4 System Environment	8
<b>3. Use Case Modeling:</b>	9
3.1 Use Cases:	9
♦ User Use Cases:	9
♦ Admin Use Cases:	9
♦ Driver Use Cases:	10
3.2 Use Case Description	12
♦ User Use Cases:	12
♦ Admin Use Cases:	13
♦ Driver Use Cases:	14
3.3 Flow of Events for User Use Case	15
♦ 1. Register Use Case	15
♦ 2. Login Use Case	16
♦ 3. Add Product to Cart Use Case	17
♦ 4. Remove Product from Cart Use Case	17
♦ 5. View Cart Use Case	18
♦ 6. Place Order (Only if total > 2000) Use Case	18
♦ 7. Choose Payment Method Use Case	19
♦ 8. Receive Driver Details after Order Use Case	20
♦ 9. View Order History Use Case	20
3.4 Flow of Events for Admin Use Case	21
♦ 1. Login Use Case	21
♦ 2. Add Product Use Case	22
♦ 3. Update Product Use Case	22
♦ 4. Delete Product Use Case	23
♦ 5. View All Orders Use Case	23

♦ 6. Monitor Inventory (Low Stock Alerts) Use Case.....	24
♦ 7. View Best-Selling Products Use Case.....	25
♦ 8. Validate User Inputs / Data Use Case.....	25
♦ 9. Assign Drivers Use Case.....	26
3.5 Flow of Events for Driver Use Cases.....	26
♦ 1. Login Use Case.....	26
♦ 2. View Assigned Orders.....	27
♦ 3. View User Details for Delivery (Name, Phone, Address) Use Case.....	28
♦ 4. Check Payment Status (Online or COD) Use Case.....	28
♦ 5. Mark Payment Received (if COD) Use Case.....	29
♦ 6. Mark Order Delivered Use Case.....	29
<b>4. Class Diagram:</b> .....	30
4.1 Relations.....	30
4.2 Class Description.....	32
<b>5. Activity Diagrams:</b> .....	36
<b>6. Collaboration &amp; Object Diagram:</b> .....	37
<b>7. Sequence Diagrams:</b> .....	38
<b>8. State Diagrams:</b> .....	39
<b>9. Component Diagram:</b> .....	40
<b>10. Deployment Diagram:</b> .....	41
<b>11. System Structure:</b> .....	42
11.1 ERD of system.....	42
11.2 System Codes and Outputs.....	42
<b>12. Conclusion:</b> .....	46
12.1 Summary of OOAD Principles.....	46
12.2 Learning Outcomes.....	46
12.3 Challenges Faced.....	46
<b>13. References:</b> .....	46
<b>14. Appendices:</b> .....	47

# 1. Introduction:

## 1.1 Project Problem Background

In today's digital era, the demand for online shopping has grown rapidly, especially after the COVID-19 pandemic, which drastically reshaped consumer behaviour. Grocery shopping—being a frequent, necessity-driven task—has seen a significant shift from traditional in-store purchases to online platforms. However, many local and mid-scale grocery retailers lack the resources or systems to offer a smooth online experience. As a result, customers still face issues like stock unavailability, lack of delivery updates, and limited accessibility.

SmartGrocery aims to fill this gap by providing an online grocery shopping system that benefits both customers and store management. By automating operations like inventory tracking, order placement, and delivery handling, SmartGrocery ensures a more reliable and accessible grocery buying process.

---

## 1.2 Risk Analysis

While developing and deploying the SmartGrocery platform, several risks may arise:

- **Technical Complexity:** Integrating payment methods, driver assignment, and real-time inventory tracking may introduce complexity that increases development time.
- **Time Constraints:** As this is a semester-long academic project, certain advanced features may need to be postponed or simplified.
- **Data Integrity:** Ensuring that user data, product data, and payment records remain accurate and secure is critical.
- **Scalability:** Since the project is currently designed for a single grocery store scenario, expanding it to multiple stores or cities may require a major architectural update.
- **User Error & Input Validation:** Poorly validated inputs might lead to crashes or incorrect order records.

To mitigate these, we follow Agile methodology, prioritise core functionalities first, and plan time for testing and incremental improvements.

---

## 1.3 Problem Statement

With the increasing demand for online shopping, especially in the grocery sector, many traditional grocery stores are finding it difficult to keep up with the changing trends. Customers often experience difficulties like long wait times, limited store hours, and inconvenience in locating specific items. As a result, an online platform that allows customers to easily order and track their groceries has become necessary.

The **SmartGrocery** platform aims to address these challenges by offering a streamlined and user-friendly e-commerce platform for grocery shopping. It will allow users to:

- Add products to their cart and place orders.
- Choose between payment methods (online or cash on delivery).
- Track their order history and receive order updates.

For admins, the system will provide capabilities to manage the product inventory, process orders, and receive alerts for products with low stock. Furthermore, drivers will be able to see their assigned orders, deliver them, and ensure payment collection if needed.

By developing the SmartGrocery platform, we aim to simplify the grocery shopping experience for users, optimize inventory management for admins, and provide an efficient order delivery mechanism for drivers.

---

## 1.4 Objective

The primary objective of the **SmartGrocery** project is to develop a comprehensive and user-friendly **online grocery shopping platform** that bridges the gap between traditional retail and modern consumer needs. The system is designed to streamline operations for customers, administrators, and delivery drivers, ensuring a seamless grocery shopping and delivery experience.

### Key objectives include:

- To provide customers with a simple and efficient interface to browse products, manage a cart, and place orders online with flexible payment options (online/COD).
- To equip administrators with tools to manage products, monitor inventory levels (with low-stock alerts), view sales reports, and assign delivery drivers.
- To enable drivers to access order details, update delivery statuses, and track payment collection (especially for COD orders).
- To ensure real-time communication and data synchronisation across all modules (user, admin, and driver).
- To enhance data accuracy and reduce manual inventory and order management errors through automation.
- To implement essential input validation and session management for a secure and reliable user experience.

- To create a modular and scalable system structure that can support future enhancements such as multi-store support, loyalty points, and more personalized recommendations.

---

## 1.5 Scope of the Project

The SmartGrocery project aims to deliver a fully functional online grocery shopping platform that serves three main actors: users (customers), admins, and delivery drivers. The system will allow users to register, browse products, manage a shopping cart, and place orders using selected payment methods. Admins will be able to manage the product inventory, monitor stock levels, and assign drivers to orders. Drivers will have access to their assigned deliveries and update the status of each delivery.

**Customer functionalities** include registration, login, product browsing, cart management, order placement (with a minimum order value), payment selection (online or cash on delivery), and viewing order history along with delivery status updates.

**Admin functionalities** include secure login, adding/editing/deleting products, viewing and managing orders, tracking low-stock items, and assigning drivers.

**Driver functionalities** include login, viewing assigned orders, accessing delivery information, checking payment mode, marking payment received (in case of COD), and updating delivery status.

The system will be built using ASP.NET MVC with ADO.NET for database interaction. It will support real-time updates, input validation, and secure session handling. The scope is limited to a single-store setup but is designed in a modular way to allow future scaling to support multiple stores, cities, or platforms such as mobile apps.

---

## 1.6 Current System

In many local grocery stores, the current system for shopping and order management is largely manual and traditional. Customers typically visit the store physically or call to place orders without any assurance about product availability. This often results in inconvenience due to stock shortages or long wait times. The stores operate within fixed hours, limiting customer accessibility.

Inventory management is handled manually by store staff, which leads to errors in stock tracking and delays in restocking. Delivery services, if available, are uncoordinated, and drivers receive

orders informally without clear communication or tracking. Customers cannot monitor their orders, and payment is usually limited to cash transactions. Furthermore, data related to orders, stock, and payments are poorly maintained, making it difficult to analyze or optimize store operations.

---

## **1.7 Proposed System (SmartGrocery)**

The SmartGrocery platform introduces a modern, automated solution to these problems. Customers can browse products online, add them to a cart, and place orders conveniently from their devices. The system provides real-time updates on stock availability, ensuring users are always informed. It operates 24/7, removing the limitations of store hours and location.

Admins benefit from automated inventory tracking with low-stock alerts, and can easily add, update, or delete products. Orders placed by users are systematically assigned to drivers, who can log in to view their tasks, customer contact details, and delivery addresses. The system also supports both online payments and cash on delivery, giving customers more flexibility.

Order tracking, payment confirmation, and historical order data are all managed through the system, improving transparency and operational efficiency. In short, SmartGrocery provides a reliable, user-friendly, and scalable solution for modern grocery shopping needs.

---

## **2. Requirement Analysis:**

### **2.1 Functional Requirements:**

- User Registration and Login
- Cart Management (Add/Remove/View Items)
- Order Placement and Order History
- Online or Cash-on-Delivery Payment Options
- Admin Product Management (Add/Edit/Delete)
- Driver Assignment
- Driver Login and Order Viewing
- Payment Tracking for COD and Online
- Inventory Monitoring with Low Stock Alerts
- Persistent User Sessions
- Input/Data Validation



## 2.2 Non-Functional Requirements:

- Fast database access using Dapper ORM.
  - User-friendly front end developed in ASP.NET.
  - Responsive system performance for small and medium-scale usage.
  - Scalable structure for future expansion.
- 

## 2.3 User Roles

### Actors:

1. User (Customer)
2. Admin
3. Driver

### User (Customer):

- Register and log in to the system.
- Browse and search for grocery products.
- Add or remove products from their shopping cart.
- Place orders only if the cart total exceeds 2000.
- Select payment method: Online or Cash on Delivery (COD).
- Receive details of the assigned delivery driver after order confirmation.
- View their order history and track past orders.

### Admin:

- Log in to the admin panel to manage the system backend.
- Add, update, or delete products and categories.
- Monitor inventory and receive alerts on low stock.
- View all orders placed by users.
- Assign delivery drivers to specific orders.
- Validate data inputs to maintain system integrity.
- Generate reports such as best-selling products (if applicable).

### Driver:

- Log in to access assigned delivery orders.
  - View customer details including name, phone, and address.
  - Confirm payment collection when COD is selected.
  - Mark orders as delivered after successful delivery.
-

## 2.4 System Environment

The SmartGrocery system is a web-based application built using the following technologies:

- **Frontend:** Developed with ASP.NET MVC to deliver a responsive and user-friendly interface accessible on desktops and mobile devices through modern web browsers.
- **Backend:** Uses C# with ADO.NET and Dapper ORM for efficient communication with the database.
- **Database:** Microsoft SQL Server stores all data related to users, products, orders, drivers, and payments.
- **Payment Integration:** Supports both online payment gateways (such as EasyPaisa) and cash-on-delivery options.
- **Security:** Includes user authentication, session management, and input validation to ensure data privacy and system integrity.

This environment supports a reliable, scalable, and secure platform for all users, admins, and drivers.

---

## 3. Use Case Modeling:

### 3.1 Use Cases:

#### ◆ User Use Cases:

1. Register
2. Login
3. Add Product to Cart
4. Remove Product from Cart
5. View Cart
6. Place Order (Only if total > 2000)
7. Choose Payment Method (Online / Cash on Delivery)
8. Receive Driver Details after Order
9. View Order History

---

#### ◆ Admin Use Cases:

1. Login
2. Add Product
3. Update Product
4. Delete Product

5. View All Orders
  6. Monitor Inventory (Low Stock Alerts)
  7. View Best-Selling Products
  8. Validate User Inputs / Data
  9. Assign Drivers
- 

◆ **Driver Use Cases:**

1. Login
  2. View Assigned Orders
  3. View User Details for Delivery (Name, Phone, Address)
  4. Check Payment Status (Online or COD)
  5. Mark Payment Received (if COD)
  6. Mark Order Delivered
-

# **3.1**

# **Use Case Diagram**

## 3.2 Use Case Description

### ◆ User Use Cases:

#### 1. Register

- **Description:** The user can create an account by providing necessary details like name, email, and password. After registration, they will have access to the system for browsing and placing orders.
- **Why it's needed:** Registration helps users personalize their shopping experience, manage their orders, and receive notifications related to their account.

#### 2. Login

- **Description:** Users log into their account using their registered credentials (email and password).
- **Why it's needed:** Login is crucial for maintaining user-specific sessions, tracking orders, and personalizing the shopping experience.

#### 3. Add Product to Cart

- **Description:** After browsing, users can add items to their shopping cart for purchase.
- **Why it's needed:** It gives users the flexibility to pick items, review them, and then proceed to checkout.

#### 4. Remove Product from Cart

- **Description:** Users can remove items from the cart if they change their mind about purchasing them.
- **Why it's needed:** This provides flexibility and control over the shopping cart, ensuring the user can adjust their order before finalizing.

#### 5. View Cart

- **Description:** Users can view all items added to their cart, along with quantities and the total price.
- **Why it's needed:** This allows users to review their selected items before proceeding to checkout.

#### 6. Place Order (Only if total > 2000)

- **Description:** Users place an order once they've confirmed the items in their cart. The system checks if the total amount exceeds 2000 before allowing the order to be placed.

- **Why it's needed:** This ensures that smaller orders, which may not be profitable for delivery, are excluded. It also sets a minimum spending threshold.
  - 7. **Choose Payment Method (Online / Cash on Delivery)**
    - **Description:** Users can choose between online payment (via EasyPaisa or other services) or Cash on Delivery (COD) for their purchase.
    - **Why it's needed:** Offers flexibility to the user in terms of payment options and accommodates different preferences.
  - 8. **Receive Driver Details after Order**
    - **Description:** After placing the order, users receive the name and phone number of the driver assigned to deliver their order.
    - **Why it's needed:** This is crucial for communication between the user and driver, particularly if there are any issues or delays with the delivery.
  - 9. **View Order History**
    - **Description:** Users can view their previous orders, including product details, order status, payment status, and delivery information.
    - **Why it's needed:** This helps users keep track of their shopping activity and enables re-ordering of frequently purchased items.
- 

#### ◆ **Admin Use Cases:**

1. **Login**
  - **Description:** Admin logs in to manage the platform's operations.
  - **Why it's needed:** It's essential to restrict the access of system settings and management features to authorized personnel only.
2. **Add Product**
  - **Description:** Admin can add new products to the system, including details like product name, category, price, and stock.
  - **Why it's needed:** This is necessary to keep the inventory updated with new items available for purchase.
3. **Update Product**
  - **Description:** Admin can modify existing product details, such as price, stock quantity, and description.
  - **Why it's needed:** Ensures that product information is up-to-date, accurate, and reflects any changes in pricing or availability.
4. **Delete Product**

- **Description:** Admin can remove products from the inventory that are no longer available or relevant.
- **Why it's needed:** Helps maintain an accurate product catalog and remove outdated or discontinued items.

#### 5. View All Orders

- **Description:** Admin can view the details of all orders placed by users, including product details, payment status, and delivery status.
- **Why it's needed:** Admins need to manage and track all user orders, ensuring that the system runs smoothly and customers receive their orders in a timely manner.

#### 6. Monitor Inventory (Low Stock Alerts)

- **Description:** Admin is notified when a product's stock falls below a certain threshold.
- **Why it's needed:** Keeps the admin informed about inventory levels, ensuring that the store can replenish stock before items run out.

#### 7. View Best-Selling Products

- **Description:** Admin can see a list of the most popular products based on sales data.
- **Why it's needed:** Identifying best-selling items helps the admin focus on popular products, streamline inventory, and improve marketing strategies.

#### 8. Validate User Inputs / Data

- **Description:** Admin can review and validate user inputs during registration or order placement.
- **Why it's needed:** Ensures that the data entered by users is accurate, preventing issues with orders or delivery.

#### 9. Assign Drivers

- **Description:** Admin system automatically assigns available drivers to orders based on predefined criteria such as delivery area, driver availability, and workload.
- **Why it's needed:** Automating driver assignment ensures faster and more efficient delivery by matching orders with drivers familiar with the area, reduces manual workload on the admin, and minimizes delivery delays due to unavailability or misassignment.

---

### ◆ Driver Use Cases:

#### 1. Login

- **Description:** The driver logs into the system to view assigned orders.
  - **Why it's needed:** Ensures that only authorized drivers can access delivery details.
2. **View Assigned Orders**
- **Description:** Drivers see orders assigned to them.
  - **Why it's needed:** Ensures that deliveries are assigned fairly among drivers and helps maintain a balanced workload.
3. **View User Details for Delivery (Name, Phone, Address)**
- **Description:** Drivers see the orders assigned to them. They can view customer details such as name, phone number, and delivery address. If the payment method is Cash on Delivery, the driver collects the payment and updates the system to confirm delivery..
  - **Why it's needed:** This helps the driver with delivery coordination and contact in case of issues.
4. **Check Payment Status (Online or COD)**
- **Description:** The driver can check whether the user has paid online or if they need to collect cash on delivery.
  - **Why it's needed:** Helps the driver plan the delivery accordingly and confirm payment status.
5. **Mark Payment Received (if COD)**
- **Description:** If the user opts for cash on delivery, the driver marks the payment as received in the system.
  - **Why it's needed:** This ensures that the payment process is recorded, and no outstanding payment is missed.
6. **Mark Order Delivered**
- **Description:** The driver confirms that the order has been delivered.
  - **Why it's needed:** Helps close the order and notify the system that the delivery was successful.
- 

### 3.3 Flow of Events for User Use Case

#### 1. Register Use Case

##### Precondition:

1. The user is not yet registered in the system and is trying to create a new account.



**Main Flow:**

1. The user navigates to the registration page.
2. The system prompts the user to enter necessary information: name, email, password, and phone number.
3. The user enters the required information and clicks the "Register" button.
4. The system validates the entered data (e.g., checks if the email is already used).
5. If validation is successful, the system creates a new account and sends a confirmation email to the user.
6. The user is successfully registered and redirected to the login page.

**Subflow:**

- If the user tries to register with an existing email:
  1. The system displays an error message indicating the email is already in use.
  2. The user is prompted to provide a different email.

**Alternative Flow:**

- If the user fails to provide all the required fields:
    1. The system highlights the missing fields and asks the user to fill them in.
    2. The user corrects the data and resubmits the registration form.
- 

## 2. Login Use Case

**Precondition:**

1. The user has already registered in the system.

**Main Flow:**

1. The user navigates to the login page.
2. The system prompts the user to enter their email and password.
3. The user enters the credentials and clicks the "Login" button.
4. The system validates the credentials against the stored data.
5. If credentials are correct, the user is logged in successfully and directed to the homepage or dashboard.
6. The user can now access their account, browse products, and place orders.

**Subflow:**

- If the user enters incorrect credentials:
  1. The system displays an error message (e.g., "Invalid email or password").
  2. The user is prompted to try again.

**Alternative Flow:**

- If login credentials are incorrect:
    1. The user is prompted to re-enter the correct login details.
- 

**♦ 3. Add Product to Cart Use Case****Precondition:**

1. The user is logged into the system.
2. The user has browsed and selected a product.

**Main Flow:**

1. The user selects a product they wish to buy.
2. The system displays the product's details (e.g., quantity, price).
3. The user selects the desired quantity of the product.
4. The user clicks "Add to Cart."
5. The system adds the product to the user's shopping cart and updates the cart's total price.
6. The user is shown a confirmation message that the product was successfully added to the cart.

**Subflow:**

- If the product stock is insufficient for the selected quantity:
  1. The system displays an error message and asks the user to select a smaller quantity.

**Alternative Flow:**

- The user decides to add multiple products from different categories to the cart:
    1. The user continues browsing and adding items until they are done.
    2. The cart is updated in real-time as new items are added.
- 

**♦ 4. Remove Product from Cart Use Case****Precondition:**

1. The user is logged in and has products in their cart.

**Main Flow:**

1. The user navigates to the cart page.
2. The system displays a list of items in the user's cart.
3. The user clicks on the "Remove" button next to a product.
4. The system removes the product from the cart and updates the total price accordingly.
5. The user is shown a confirmation message that the product has been removed successfully.

**Subflow:**

- If the cart is empty after removing the last item:
  1. The system displays a message indicating that the cart is empty, and the user can continue shopping.

**Alternative Flow:**

- The user wants to adjust the quantity instead of removing a product:
    1. The user updates the quantity and the cart's total price is automatically adjusted.
- 

♦ **5. View Cart Use Case**

**Precondition:**

1. The user is logged in and has products in their cart.

**Main Flow:**

1. The user navigates to the cart page.
2. The system displays a list of products in the cart, including quantity, price per item, and total price.
3. The user reviews the cart, and can either proceed to checkout or continue shopping.
4. The user can see the option to proceed with the payment by choosing either "Pay Online" or "Cash on Delivery."

**Subflow:**

- If the cart is empty:
  1. The system displays a message informing the user that there are no items in the cart.

**Alternative Flow:**

- The user decides to change the quantity of an item or remove an item from the cart before proceeding to checkout:
  1. The cart is updated accordingly with real-time adjustments.

---

## ◆ 6. Place Order (Only if total > 2000) Use Case

### Precondition:

1. The user has added items to the cart and the total amount is greater than 2000.

### Main Flow:

1. The user clicks the "Place Order" button.
2. The system checks if the total amount is above the minimum threshold (2000).
3. If the total is greater than 2000, the system prompts the user to choose a payment method: Cash on Delivery or EasyPaisa (online).
4. The user selects a payment method.
5. The system confirms the order and sends a confirmation message to the user, along with the assigned driver's name and phone number for delivery.
6. The order is logged in the system and passed to the admin and driver for further processing.

### Subflow:

- If the total is less than 2000:
  1. The system displays an error message indicating that the order amount must exceed 2000 to be placed.

### Alternative Flow:

- The user decides to cancel the order before confirming:
  1. The system discards the cart and allows the user to continue shopping.

---

## ◆ 7. Choose Payment Method Use Case

### Precondition:

1. The user has added products to the cart and is ready to place an order.

### Main Flow:

1. The user selects their preferred payment method: Cash on Delivery or Online Payment.
2. If the user selects online payment, the system redirects them to the payment gateway (e.g., EasyPaisa) to complete the transaction.
3. Once the online payment is confirmed, the system updates the order status as "Paid."
4. If the user selects Cash on Delivery, the order status is marked as "Pending Payment."

**Subflow:**

- If the online payment fails:
  1. The system displays an error message and prompts the user to try again or choose another payment method.

**Alternative Flow:**

- The user decides to change the payment method before confirming:
    1. The user selects another payment option and proceeds.
- 

**♦ 8. Receive Driver Details after Order Use Case****Precondition:**

1. The user has placed an order successfully.

**Main Flow:**

1. After the order is placed, the system assigns a driver based on the user's delivery location.
2. The system sends the driver's name and phone number to the user's registered contact information (email/phone).
3. The user can use the contact information to coordinate with the driver if needed.

**Subflow:**

- If the driver cannot be assigned:
  1. The system notifies the admin to reassign the delivery to an available driver.

**Alternative Flow:**

- If the user is unavailable or unable to reach the driver:
    1. The user can call customer support for assistance in locating the driver.
- 

**♦ 9. View Order History Use Case****Precondition:**

1. The user is logged in and has placed at least one order.

**Main Flow:**

1. The user navigates to the "Order History" section.
2. The system displays a list of all past orders, including product details, status, delivery date, and payment status.
3. The user can view detailed information about each order.

**Subflow:**

- If no past orders exist:
  1. The system displays a message indicating there are no past orders.

**Alternative Flow:**

- The user filters orders based on date, status, or other criteria:
    1. The system updates the order history display according to the filter settings.
- 

## 3.4 Flow of Events for Admin Use Case

### ♦ 1. Login Use Case

**Precondition:**

1. The admin has an existing account in the system.

**Main Flow:**

1. The admin navigates to the login page.
2. The system prompts the admin to enter their email and password.
3. The admin enters the credentials and clicks the "Login" button.
4. The system validates the credentials.
5. If valid, the system logs the admin in and directs them to the admin dashboard.
6. If invalid, the system displays an error message and prompts the admin to try again.

**Subflow:**

- If the admin enters incorrect credentials:
  1. The system displays an error message (e.g., "Invalid email or password").
  2. The admin is prompted to re-enter the credentials or use the "Forgot Password" link.

**Alternative Flow:**

- If the admin forgets their password:
  1. The admin clicks the "Forgot Password" link.
  2. The system asks for the admin's email and sends a password reset link.

---

## ♦ 2. Add Product Use Case

### Precondition:

1. The admin is logged in.

### Main Flow:

1. The admin navigates to the “Manage Products” section.
2. The system displays the current product list with an “Add Product” button.
3. The admin clicks “Add Product.”
4. The system prompts the admin to enter product details (name, category, price, stock).
5. The admin enters the required details and submits the form.
6. The system validates the input and adds the new product to the product database.
7. The system confirms the addition of the product.

### Subflow:

- If the admin enters incomplete or invalid data:
  1. The system displays an error message asking for the missing or invalid information.
  2. The admin corrects the input and resubmits.

### Alternative Flow:

- If the admin cancels adding the product:
  1. The admin can click the “Cancel” button, and the system returns to the product list.

---

## ♦ 3. Update Product Use Case

### Precondition:

1. The admin is logged in, and a product exists in the system.

### Main Flow:

1. The admin navigates to the “Manage Products” section.
2. The system displays the list of products with options to update.
3. The admin selects the “Update” option next to the product they want to edit.
4. The system loads the current product details.
5. The admin modifies product details (e.g., price, stock).

6. The admin clicks “Save” to submit the changes.
7. The system updates the product in the database and confirms the change.

**Subflow:**

- If the admin doesn’t make any changes and clicks “Save”:
  1. The system notifies the admin that no changes were made.

**Alternative Flow:**

- If the admin wants to cancel the update:
    1. The admin clicks “Cancel” to return to the product list without saving changes.
- 

♦ **4. Delete Product Use Case**

**Precondition:**

1. The admin is logged in, and a product exists in the system.

**Main Flow:**

1. The admin navigates to the “Manage Products” section.
2. The system displays the list of products with options to delete.
3. The admin selects the “Delete” option next to the product they want to remove.
4. The system asks the admin to confirm the deletion.
5. The admin confirms by clicking “Delete.”
6. The system removes the product from the database and displays a confirmation message.

**Subflow:**

- If the product is linked to an existing order or has stock remaining:
  1. The system prevents deletion and displays an error message explaining the situation.

**Alternative Flow:**

- If the admin cancels the deletion:
    1. The system returns to the product list without making any changes.
- 

♦ **5. View All Orders Use Case**

**Precondition:**



1. The admin is logged in.

**Main Flow:**

1. The admin navigates to the “View Orders” section.
2. The system displays all orders placed by users, including order ID, customer name, order details, and status.
3. The admin can filter and sort orders by status, date, or payment type.
4. The admin selects an order to view detailed information (e.g., product list, quantities, user details).
5. The admin can update the order status (e.g., mark as shipped).

**Subflow:**

- If there are no orders in the system:
  1. The system displays a message indicating no orders have been placed yet.

**Alternative Flow:**

- If the admin wants to export order data:
  1. The admin can export the order history to a CSV or PDF file.

---

◆ **6. Monitor Inventory (Low Stock Alerts) Use Case**

**Precondition:**

1. The admin is logged in and has product inventory data.

**Main Flow:**

1. The admin navigates to the “Low Inventory” section in the admin dashboard.
2. The system automatically identifies products with stock levels below a predefined threshold.
3. The system displays a list of products with low stock and their current quantities.
4. The admin can take necessary actions like restocking or removing products.

**Subflow:**

- If no products have low inventory:
  1. The system displays a message indicating no products are running low on stock.

**Alternative Flow:**

- The admin decides to reorder products:
  1. The admin selects a product and indicates the quantity to be restocked.

---

## ◆ 7. View Best-Selling Products Use Case

### **Precondition:**

1. The admin is logged in and the system has sales data.

### **Main Flow:**

1. The admin navigates to the “Best Selling Products” section.
2. The system generates and displays a list of top-selling products based on total sales for a given period.
3. The admin can filter the best-sellers by time period (e.g., daily, weekly, or monthly).
4. The admin views product names, quantities sold, and total sales.

### **Subflow:**

- If no sales data is available for the selected period:
  1. The system displays a message indicating no data is available.

### **Alternative Flow:**

1. The admin chooses to download the best-selling products report.

---

## ◆ 8. Validate User Inputs / Data Use Case

### **Precondition:**

1. The admin is logged in.

### **Main Flow:**

1. The system validates all inputs entered by the admin for product management, user registration, etc.
2. If the data entered is valid, the system allows the action to be completed (e.g., adding a product or updating a user).
3. If the data is invalid, the system displays an error message and requests corrections.

### **Subflow:**

- If the admin enters invalid data (e.g., price is a negative number):
  1. The system displays a validation error message asking for the correct input.

**Alternative Flow:**

- If the admin decides to cancel the action after validation errors:
    1. The system returns the admin to the main screen.
- 

**♦ 9. Assign Drivers Use Case****Precondition:**

1. The admin is logged in and drivers exist in the system.

**Main Flow:**

1. The admin navigates to the “Driver Management” section.
2. The system displays a list of available drivers.
3. When a new delivery is created, the system automatically assigns it to an available driver.
4. The system saves the assignment and updates the driver’s status.
5. The assigned driver is notified of their new delivery.

**Subflow:**

- If a driver is already assigned.
  1. The system automatically checks driver availability. If a selected driver is already assigned to another delivery, the system alerts the admin and automatically selects the next available driver for assignment.

**Alternative Flow:**

- The admin cancels the assignment:
    1. The system returns the admin to the driver list without saving changes.
- 

**3.5 Flow of Events for Driver Use Cases****♦ 1. Login Use Case****Precondition:**

1. The driver has an existing account and is assigned to an area.

**Main Flow:**

1. The driver navigates to the login page.
2. The system prompts the driver to enter their username/email and password.
3. The driver enters the credentials and clicks "Login."
4. The system validates the credentials.
5. If valid, the system logs the driver in and displays their dashboard with assigned orders for the area.
6. If invalid, the system displays an error message and prompts the driver to try again.

**Subflow:**

- If the driver enters incorrect credentials:
  1. The system displays an error message (e.g., "Invalid username or password").
  2. The driver is prompted to re-enter the credentials or use the "Forgot Password" link.

**Alternative Flow:**

- If the driver forgets their password:
  1. The driver clicks the "Forgot Password" link.
  2. The system asks for the driver's email and sends a password reset link.

---

♦ **2. View Assigned Orders**

**Precondition:**

1. The driver is logged in and has orders assigned to them..

**Main Flow:**

1. The driver navigates to the "Assigned Orders" section on their dashboard.
2. The system displays a list of orders assigned to the driver, including order ID, customer name, delivery address, and payment status.
3. The driver selects an order to view more details.
4. The system displays detailed information about the order, including user details, and delivery address.

**Subflow:**

- If no orders are assigned to the driver:
  1. The system displays a message indicating no orders have been assigned.

**Alternative Flow:**

- If the driver wishes to refresh the list of orders:

1. The driver clicks the "Refresh" button to view the latest assigned orders.
- 

### ♦ 3. View User Details for Delivery (Name, Phone, Address) Use Case

#### **Precondition:**

1. The driver is logged in and has assigned orders.

#### **Main Flow:**

1. The driver selects an order from their assigned orders list.
2. The system displays detailed information about the user, including the user's name, phone number, and delivery address.
3. The driver reviews the delivery details to ensure proper delivery.

#### **Subflow:**

- If the delivery address is incorrect or missing:
  1. The driver can request the admin to correct the address or contact the user for confirmation.

#### **Alternative Flow:**

- If the driver is unable to reach the user (e.g., phone number is unreachable):
    1. The system provides the option for the driver to report an issue to the admin.
- 

### ♦ 4. Check Payment Status (Online or COD) Use Case

#### **Precondition:**

- The driver is logged in and has an assigned order.

#### **Main Flow:**

1. The driver selects an order from the assigned list.
2. The system displays the order details, including the payment method (online or COD).
3. The driver checks if the payment status is "Online" or "Cash on Delivery (COD)."

#### **Subflow:**

- If the payment status is "Online":
  1. The system confirms that the payment has been received, and the driver does not need to collect cash on delivery.

**Alternative Flow:**

- If the payment status is "COD":
    1. The driver is notified that the payment needs to be collected at the time of delivery.
- 

**♦ 5. Mark Payment Received (if COD) Use Case****Precondition:**

1. The driver is logged in and has a COD order assigned.

**Main Flow:**

1. The driver selects a COD order from their assigned orders list.
2. The system displays the payment status as "Cash on Delivery."
3. The driver receives payment from the user upon delivery.
4. The driver marks the payment as "Received" in the system.
5. The system updates the order status to "Payment Received" and confirms the action.

**Subflow:**

- If the user does not have the required amount or refuses to pay:
  1. The system allows the driver to cancel the order delivery and report the issue.

**Alternative Flow:**

- If the driver mistakenly marks the payment as "Received" for an online order:
    1. The system will display a warning or error message indicating that the payment has already been made.
- 

**♦ 6. Mark Order Delivered Use Case****Precondition:**

1. The driver is logged in and has an assigned order to deliver.

**Main Flow:**

1. The driver selects an order from the assigned orders list.
2. The driver delivers the order to the user at the specified address.
3. The driver marks the order as "Delivered" in the system once the delivery is successful.
4. The system updates the order status to "Delivered" and confirms the action.

### Subflow:

- If the order is not successfully delivered (e.g., user is unavailable):
  1. The driver can report the issue to the admin, and the order status remains as "Pending" or "Undelivered."

### Alternative Flow:

- If the driver accidentally marks the order as "Delivered" before the delivery is made:
    1. The system displays a warning and asks the driver to confirm that the order was delivered.
- 

## 4. Class Diagram:

### 4.1 Relations:

- Admin manages many Products (1 to many)
- Admin manages many Drivers(1 to many)
- User places many Orders (1 to many)
- Order contains many OrderItems (1 to many)
- OrderItem linked to a single Product (many to 1)
- User has many CartItems (1 to many)
- Driver delivers many Orders (1 to many)

OrderDetails, DriverOrderViewModel, DriverProfileViewModel, and ProductSales are view models used to display aggregated info

#### 1. Admin → Product

**Type:** Association

**Reason:** Admin manages products, but Product can exist independently of Admin (not a part-whole relationship).

**Arrow:** Solid line, no diamond.

---

#### 2. Admin → Driver

**Type:** Association

**Reason:** Admin assigns/adds Drivers, but they are independent entities.

**Arrow:** Solid line, no diamond.

---

### 3. User → Order

**Type:** Aggregation

**Reason:** An Order is associated with a User, but Orders can exist even if the User is deleted (for logs/history).

**Arrow:** Hollow diamond on **User** side.

---

### 4. Order → OrderItems

**Type:** Composition

**Reason:** **OrderProduct** is tightly bound to Order—if an Order is deleted, its OrderProduct list should also be deleted.

**Arrow:** Solid diamond on **Order** side.

---

### 5. Driver → Order

**Type:** Association

**Reason:** A Driver is assigned to deliver an order, but Orders can exist before Drivers are assigned.

**Arrow:** Solid line, no diamond.

---

### 6. Product → OrderItems

**Type:** Dependency

**Reason:** OrderProduct needs Product details (name, price) at the time of order but doesn't directly link to or modify Product.

**Arrow:** Dotted line (dependency).

---

### 7. User → CartItem

**Type:** Composition

**Reason:** A Cart exists only for a specific User and dies with the User.

**Arrow:** Solid diamond.

---



## 8. CartItem → Product

### Type: Dependency

**Reason:** CartItem refers to Product (for name, price, etc.). But it doesn't control or own the Product.

**Arrow:** Dotted line.

---

## 4.2 Class Descriptions

### 1. Admin

- Represents the admin user of the system. Contains login credentials and possibly other information needed to authenticate and authorize admin access.

### 2. AdminOrderItemViewModel

- A ViewModel used to display detailed information about individual items in an order from the admin's perspective, including product info, quantity, price, and possibly user and driver details.

### 3. AdminOrderViewModel

- A ViewModel used by the admin to view all orders placed by users. May include user details, order total, order status, and assigned driver.

### 4. CartItem

- Represents an item added to a user's shopping cart. Includes product ID, user ID, quantity, and subtotal. Temporary until the order is placed.

### 5. Driver

- Represents a delivery driver. Contains login credentials and profile information. Used to assign and track deliveries.

### 6. DriverOrderViewModel

- A ViewModel for drivers to see orders assigned to them. Includes user delivery details, payment method, and product list for delivery.

### 7. DriverProfileViewModel

- A ViewModel used to display the profile information of a logged-in driver, such as name, assigned area, and contact information.

#### 8. **Order**

- Represents the main order entity placed by a user. Contains user ID, order date, total amount, payment method, delivery address, and assigned driver ID.

#### 9. **OrderDetails**

- Contains specific delivery and payment-related information for an order, such as address, phone number, payment status, and delivery status.

#### 10. **OrderProduct**

- Represents the many-to-many relationship between an order and the products within that order. Stores the order ID, product ID, quantity, and price per item.

#### 11. **OrderItem**

- Represents individual items within an order (alternative to or used alongside OrderProduct). Contains product details, quantity, and price.

#### 12. **OrderViewModel**

- A ViewModel for displaying user-specific order history and order tracking. Includes order summary and details like status, total, and assigned driver.

#### 13. **Product**

- Represents a grocery item available in the system. Contains product name, category, price, quantity in stock, and image.

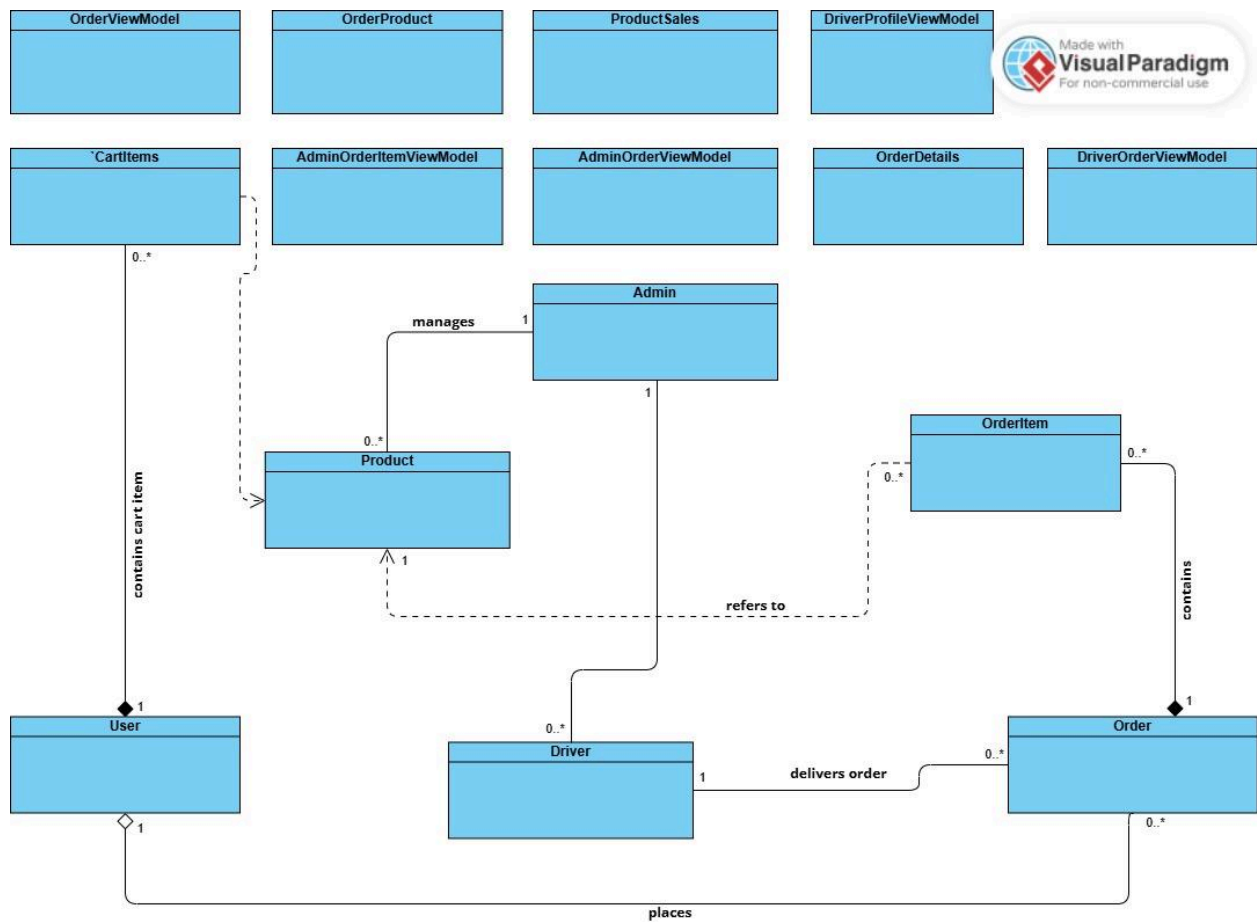
#### 14. **ProductSales**

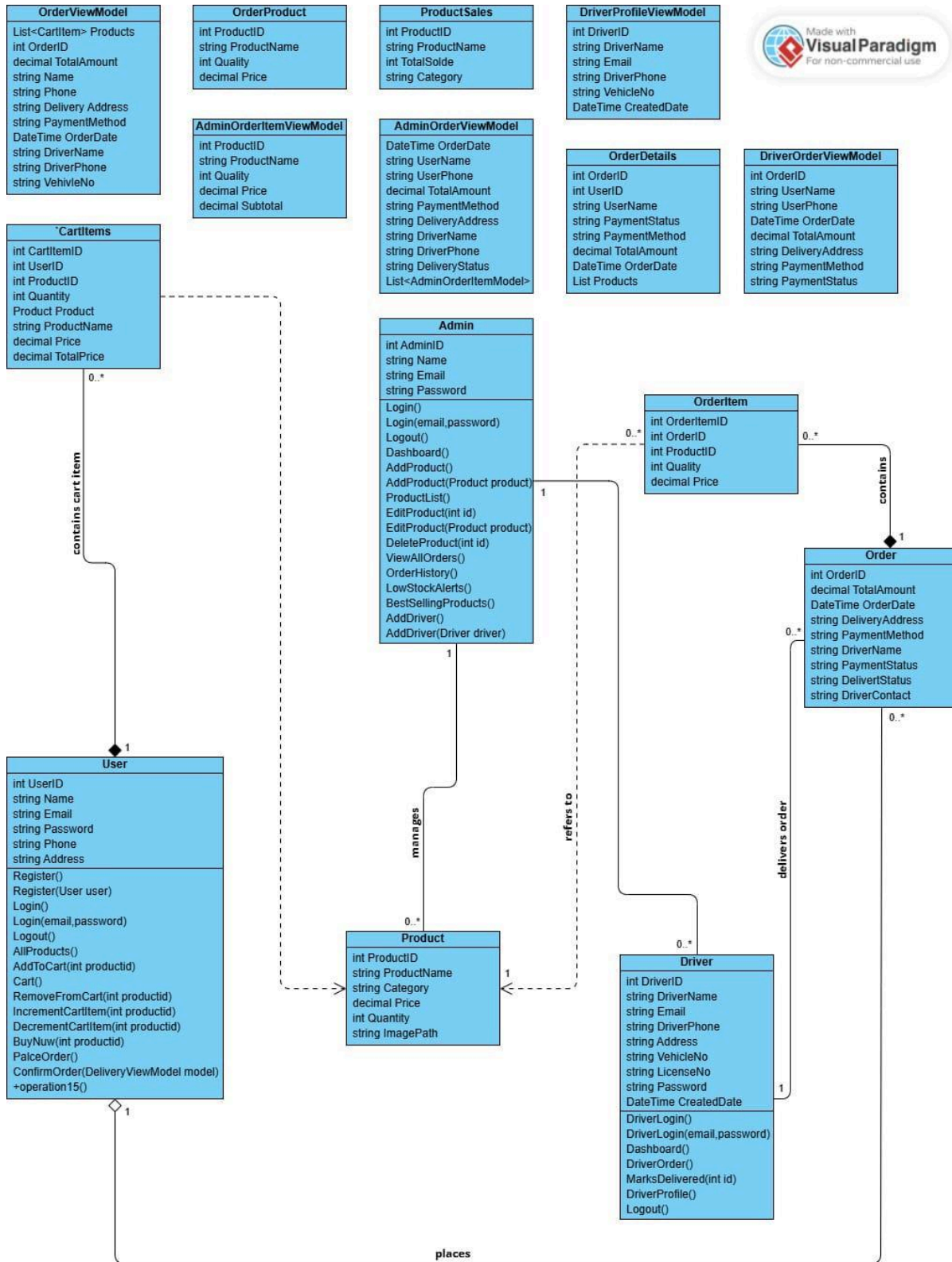
- Tracks the number of times a product has been sold. Used for generating sales reports and identifying best-selling products.

#### 15. **User**

- Represents a customer using the system. Contains personal details such as name, email, password, and contact information for placing orders.

## CLASS DIAGRAM





**5.**

# **ACTIVITY DIAGRAMS**

# **6.**

# **COLLABORATION**

# **&**

# **OBJECT DIAGRAM**

# **7.**

# **SEQUENCE DIAGRAMS**

# **8.**

# **STATE**

# **DIAGRAMS**



# **9.**

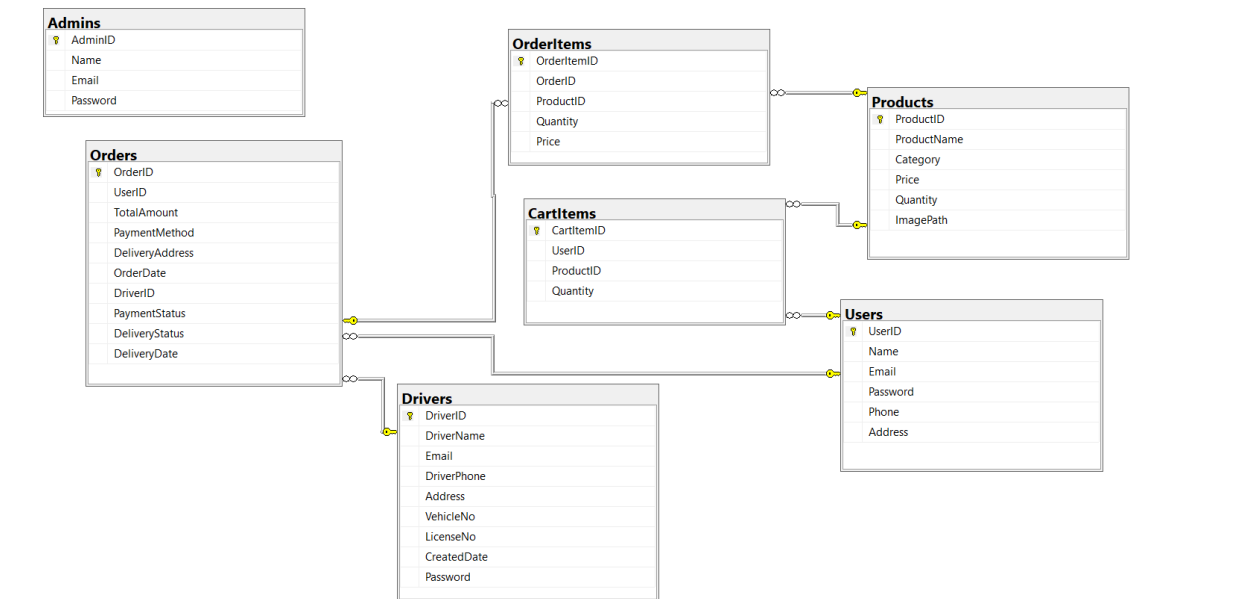
# **COMPONENT DIAGRAM**

# **10.**

# **DEPLOYMENT DIAGRAM**

# 11. System Architecture:

## 11.1 ERD of the System



## 11.2 System code and output:

- **SYSTEM CODE**

### -- Admins Table

```
CREATE TABLE Admins (  
    AdminID INT PRIMARY KEY IDENTITY,  
    Name NVARCHAR(100),  
    Email NVARCHAR(100),  
    Password NVARCHAR(100)  
);
```

### -- Products Table

```
CREATE TABLE Products (  
    ProductID INT PRIMARY KEY IDENTITY,  
    ProductName NVARCHAR(100),  
    Category NVARCHAR(50),  
    Price DECIMAL(10, 2),  
    Quantity INT,  
    ImagePath NVARCHAR(MAX)  
);
```

### -- Orders Table

```

CREATE TABLE Orders (
    OrderID      INT PRIMARY KEY IDENTITY,
    UserID       INT,
    OrderDate    DATETIME,
    TotalAmount  DECIMAL(10, 2),
    PaymentMethod NVARCHAR(20),
    PaymentStatus NVARCHAR(20),
    DriverID     INT,
    DeliveryDate  DATETIME,
    DeliveryStatus NVARCHAR(50),
    DeliveryAddress NVARCHAR(200)
);

```

#### **-- OrderItems Table**

```

CREATE TABLE OrderItems (
    OrderItemID INT PRIMARY KEY IDENTITY,
    OrderID     INT,
    ProductID   INT,
    Quantity    INT,
    Price       DECIMAL(10, 2) -- Fixed typo from (10.2) to (10, 2)
);

```

#### **-- Drivers Table**

```

CREATE TABLE Drivers (
    DriverID     INT PRIMARY KEY IDENTITY,
    DriverName   NVARCHAR(100),
    DriverPhone  NVARCHAR(20),
    Area        NVARCHAR(100),
    Email       NVARCHAR(100),
    Address     NVARCHAR(200),
    VehicleNo   NVARCHAR(50),
    LicenseNo   NVARCHAR(50),
    CreatedDate DATETIME,
    Password    NVARCHAR(256)
);

```

#### **-- CartItems Table**

```

CREATE TABLE CartItems (
    CartItemID INT PRIMARY KEY IDENTITY,
    UserID     INT,
    ProductID  INT,
    Quantity   INT
);

```

- **OUTPUT :**

**Admin Table:**

Results Messages				
	AdminID	Name	Email	Password
1	1	Emaz	emaz@gmail.com	123

**Users Table:**

	UserID	Name	Email	Password	Phone	Address
1	5	Emaz Ali Khan	emaz@gmail.com	65566	03358965471	North Nazimabad Karachi
2	6	Abdul Rafay	rafay@gmail.com	456	03358962369	Nazimabad, Karachi
3	7	Sumaika Asif	sumaika@gmail.com	789	03358963258	Gulshan, Karachi

**Drivers Table:**

	DriverID	DriverName	Email	DriverPhone	Address	VehicleNo	LicenseNo	CreatedDate	Password
1	12	Ali	ali@gmail.com	03352586941	Nazimabad, Karachi	12345	ab-123	2025-05-08 18:04:03.093	123
2	14	Khurram	khurram@gmail.com	03356247982	FB-area, Karachi	bv-896	c-258	2025-05-08 18:07:11.007	789

**Products Table:**

	ProductID	ProductName	Category	Price	Quantity	ImagePath
1	35	Dalda Banaspati	Dairy Product	700.00	50	2b884b31-1e0c-49e0-b74b-26bca0f35a46.jpg
2	36	Dalda Cooking Oil	Oil	600.00	25	6d5b15ce-ed24-4248-a80a-f430f8012ef8.png
3	37	Tullo Banaspati	Dairy Product	680.00	21	20250508182002249_tullogheejpg.jpg
4	38	Tullo Cooking Oil	Oil	570.00	33	925e3965-5c99-478e-ac6a-43600de70b83.jpg
5	39	Tapal Danedar	Tea	300.00	48	f3e92ec5-40b1-4c36-ae35-0dc38536b77d.jpg
6	40	Vital Tea	Tea	290.00	40	7079c102-4cc4-4416-816e-fd25c8196dc3.jpg
7	41	Lipton Danedar	Tea	320.00	30	40c7f72a-5209-4073-a9c1-484154d581c8.jpg
8	42	Sugar(2 KG)	Food	220.00	98	f14bb5c6-82f5-4211-aba9-4c226c1b5f0f.jpg
9	43	Ponam Biryani Rice (2 KG)	Rice	300.00	40	3626066a-a53a-4367-92eb-6bda61bc8a07.jpg
10	44	Ponam Supreme Rice (2 KG)	Rice	250.00	46	2c2e7803-a6f7-43f5-894c-24b5232cce2.jpg
11	45	Ponam Maida (1 KG)	Flour	150.00	19	b1f9e7dd-1c4d-4dfa-872c-f5b3f1223308.jpg
12	46	Ponam Flour (5 KG)	Flour	500.00	51	52d3cc8d-797a-483e-8af8-ff7453d71ee9.jpg
13	47	Ponam Dhanial Powder (0.5 KG)	Spices	120.00	10	368a4648-1ff8-4f9a-9078-4067093eb8fc.jpg
14	48	Ponam Daal (0.5 KG)	Daal	110.00	14	ebe1be18-87e7-4d5c-8c20-935a82e737f0.jpg
15	49	Dairy Omung (1 Litre)	Dairy Product	250.00	50	6ccee8a1-fbb9-493e-a993-c0ceecf94949.jpg
16	50	Olpers (1.5 Litre)	Dairy Product	320.00	17	79a9398e-d1a7-4c38-b34a-39c4201d422d.jpg
17	51	Nestle Milk Pak (1 Litre)	Dairy Product	260.00	25	b0567e41-f397-486e-81b9-ac83815c9322.jpg
18	52	Prince Biscuit (Pack)	Biscuit	190.00	21	68a42ba9-e477-4ff0-9786-5ba0f6e766cf.jpg
19	53	Chocolato (Pack)	Biscuit	220.00	20	da765a11-5685-4082-ab83-5c6c7e960146.jpg
20	54	Slanty (1 Pack)	Chips	290.00	19	4f38ab14-2c49-49c3-a60d-1b0c56883710.jpg
21	55	Dairy Milk (1 Pack)	Chocolate	320.00	11	c507a56b-eb87-4fdc-a413-ea4ef5284ca0.jpg
22	56	Cheer Polish (Small)	Polish	100.00	29	95566fd0-e854-4b60-96a2-17021279b81f.jpg
23	57	Harpic (Small)	Toilet Clean...	250.00	30	513afb14-f73c-4465-a3ab-2a3ff504b3fa.jpg
24	58	Ariel	Detergent	200.00	25	c705aafb-c4ad-4973-b85f-574b46dc8d47.jpg
25	59	Fogg (Single)	BodySpray	200.00	49	4bb2add0-ecd1-43c1-989f-fd29e3a143d4.jpg
26	60	Lifebuoy Soap (Pack)	Soap	250.00	48	6fc47632-23c7-4efa-9b9d-90db87ceadc6.jpg
27	61	Safeguard (Pack)	Soap	280.00	49	3067d259-8c6f-412a-b201-7186a1294d6a.jpg
28	62	Palmolive (Pack)	Soap	280.00	49	419a3e55-5b1f-48e2-ba96-0e29e239a0ce.jpg
29	63	Lifebuoy Shampoo	Shampoo	240.00	13	cefb78be-1e9e-4923-b2e5-a27e8891517c.jpg
30	64	TRESemme Shampoo	Shampoo	350.00	23	698e313c-165a-4c93-a8ca-bd9f2ad72b05.jpg

## Orders Table:

	OrderID	UserID	TotalAmount	PaymentMethod	DeliveryAddress	OrderDate	DriverID	PaymentStatus	DeliveryStatus	DeliveryDate
1	79	5	2370.00	Cash on Delivery	North Nazimabad Karachi	2025-05-08 18:50:20.243	14	Pending	Pending	NULL
2	80	5	2190.00	Online	North Nazimabad Karachi	2025-05-08 18:52:28.517	14	Paid	Pending	NULL
3	81	6	3080.00	Cash on Delivery	Nazimabad Karachi	2025-05-08 18:55:05.303	14	Pending	Pending	NULL
4	82	6	2010.00	Cash on Delivery	FB-Area, Karachi	2025-05-08 18:56:01.057	12	Pending	Pending	NULL
5	83	7	5310.00	Online	Gulshan,Karachi	2025-05-08 18:58:12.063	12	Paid	Delivered	2025-05-08 19:03:04.860
6	84	7	2350.00	Cash on Delivery	Gulshan,Karachi	2025-05-08 18:58:48.390	12	Pending	Pending	NULL
7	85	7	3700.00	Cash on Delivery	Gulshan,Karachi	2025-05-08 18:59:12.070	12	Pending	Pending	NULL
8	86	5	3450.00	Cash on Delivery	Block	2025-05-08 18:59:56.793	12	Paid	Delivered	2025-05-08 19:03:06.390
9	87	5	2680.00	Cash on Delivery	Block D, North Nazimabad, Karachi	2025-05-08 19:00:42.653	14	Pending	Pending	NULL
10	88	5	2700.00	Cash on Delivery	Shadman town	2025-05-11 17:48:09.303	12	Paid	Delivered	2025-05-11 17:55:11.240

## OrderItems Table:

	OrderItemID	OrderID	ProductID	Quantity	Price
1	91	79	35	1	700.00
2	92	79	36	1	600.00
3	93	79	46	1	500.00
4	94	79	50	1	320.00
5	95	79	60	1	250.00
6	96	80	64	1	350.00
7	97	80	63	2	240.00
8	98	80	62	1	280.00
9	99	80	55	2	320.00
10	100	80	42	2	220.00

## CartItems Table:

	CartItemID	UserID	ProductID	Quantity
1	176	5	36	5
2	177	5	39	1
3	179	5	49	1
4	181	5	57	4

## 12. Conclusion:

### 12.1 Summary of OOAD Principles

Throughout the development of the SmartGrocery system, key Object-Oriented Analysis and Design (OOAD) principles were applied, including encapsulation, abstraction, inheritance, and modularity. These principles helped in creating a well-structured, maintainable, and scalable application by separating responsibilities among different classes and actors.

### 12.2 Learning Outcomes

We gained practical experience in applying OOAD concepts to a real-world system by working on this project. We learned how to identify actors, use cases, and system requirements, and translate them into functional modules. The project also enhanced our skills in database integration, system design, and user interface development using ASP.NET MVC.

### 12.3 Challenges Faced

Some of the major challenges included handling data consistency between modules, ensuring secure and smooth user authentication, and implementing dynamic features like inventory tracking and driver assignment. Time management was also a challenge, especially while balancing testing and feature development.

---

## 13. References:

1. Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education.
2. Larman, C. (2004). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Prentice Hall.
3. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson.
4. Microsoft Docs. (n.d.). [ASP.NET MVC Documentation](#).
5. W3Schools. (n.d.). ADO.NET Tutorial
6. TutorialsTeacher. (n.d.). Dapper ORM Tutorial

## 14. Appendices:

### Website Interfaces

### User Login

**Email**

**Password**

Login

Don't have an account? [Register here](#)

### Driver Login

**Email**

**Password**

Login

### Admin Login

**Email**

**Password**

Login

### Create Your Account

**Name**

**Email**

**Password**

**Phone**

**Address**

Register Now →

Already have an account? [Sign in here](#)




User:

Fresh Groceries Just For You

Handpicked quality products for your family

QSearch products...

All ItemsIn StockSpecial Offers




Dalda Banaspati

{ Rs. 700.00 }

★★★★☆ (42)

Add to Cart




Dalda Cooking Oil

{ Rs. 600.00 }

★★★★☆ (42)

Add to Cart




Tullo Banaspati

{ Rs. 680.00 }

★★★★☆ (42)

Add to Cart



Tullo Cooking Oil

{ Rs. 570.00 }

★★★★☆ (42)

Add to Cart

Smart Grocery

Home

Cart




My Orders

Logout

Your Cart

11 item(s) in your cart

[+Add More Items](#)

Product	Price	Quantity	Total
<div><div>Dalda Cooking Oil</div><div>SKU: 36 <a href="#">Remove</a></div></div>	Rs. 600.00	<div>- 5 +</div>	Rs. 3,000.00
<div><div>Tapal Danedar</div><div>SKU: 39 <a href="#">Remove</a></div></div>	Rs. 300.00	<div>- 1 +</div>	Rs. 300.00
<div><div>Dairy Omuna (1 Litre)</div></div>	Rs. 250.00	<div>- 1 +</div>	Rs. 250.00

Order Summary

Subtotal Rs. 4,550.00

Delivery Free

Total Rs. 4,550.00

[Proceed to Checkout](#)

Smart Grocery

Home

Cart

My Orders

Logout

Confirm Your Order

Name

Emaz Ali Khan

Phone

03358965471

Address

Enter delivery address

Payment Method

Cash on Delivery

Place Order

© 2025 - SmartGrocery Application

Your Order History

Continue Shopping

Order ID	Date	Amount	Payment	Delivery Info	Items
#88	11 May 2025	Rs. 2,700.00	Cash on Delivery	Address: Shadman town Driver: Ali Contact: 03352586941 Vehicle: 12345	1 × Dalda Banaspati (Rs. 700.00 each) 1 × Ponam Flour (5 KG) (Rs. 500.00 each) 1 × Olpers (1.5 Litre) (Rs. 320.00 each) 3 × Prince Biscuit (Pack) (Rs. 180.00 each) 1 × Choccolato (Pack) (Rs. 220.00 each) 1 × Slanty (1 Pack) (Rs. 290.00 each) 1 × Cheer Polish (Small) (Rs. 100.00 each)
#87	08 May 2025	Rs. 2,680.00	Cash on Delivery	Address: Block D, North Nazimabad, Karachi Driver: Khurram Contact: 03356247982 Vehicle: bv-896	2 × Dalda Banaspati (Rs. 700.00 each) 1 × Dalda Cooking Oil (Rs. 680.00 each) 1 × Tulo Banaspati (Rs. 680.00 each)
#86	08 May 2025	Rs. 3,450.00	Cash on Delivery	Address: Block Driver: Ali Contact: 03352586941 Vehicle: 12345	4 × Ponam Supreme Rice (2 KG) (Rs. 250.00 each) 1 × Ponam Maida (1 KG) (Rs. 150.00 each) 3 × Ponam Biryani Rice (2 KG) (Rs. 300.00 each) 2 × Dalda Banaspati (Rs. 700.00 each)
#80	08 May 2025	Rs. 2,190.00	Online	Address: North Nazimabad Karachi Driver: Khurram Contact: 03356247982 Vehicle: bv-896	1 × TRESemme Shampoo (Rs. 350.00 each) 2 × Lifebuoy Shampoo (Rs. 240.00 each) 1 × Palmolive (Pack) (Rs. 280.00 each) 2 × Dairy Milk (1 Pack) (Rs. 320.00 each)

Admin:

Welcome, Emaz!

Tuesday, May 20, 2025

Stock Status

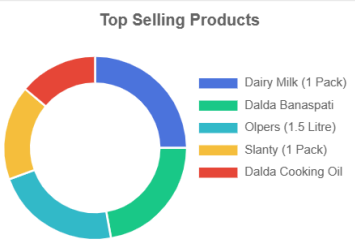
View All Products

PRODUCT ID	PRODUCT	CATEGORY	CURRENT STOCK	ACTION
64	TRESemme Shampoo	Shampoo	2	<a href="#">Restock</a>

Sales Analytics

View Sales Report

RANK	PRODUCT NAME	CATEGORY	TOTAL SOLD
1	Dairy Milk (1 Pack)	Chocolate	9
2	Dalda Banaspati	Dairy Product	8
3	Olpers (1.5 Litre)	Dairy Product	8
4	Slanty (1 Pack)	Chips	6
5	Dalda Cooking Oil	Oil	5



## Add New Product

Product Name \*

Product name is required

Quantity \*

Please enter a valid quantity

Category \*

Please enter a product category  
Example: Fruits, Vegetables, Dairy, etc.

Product Image \*

Choose File

No file chosen

Maximum file size: 5MB (JPEG, PNG)  
Product image is required

Price (Rs.) \*

Rs.

Please enter a valid price (minimum Rs. 0.01)





← Back to List

Add Product

© 2025 - SmartGrocery Application

## Product List

+Add New Product

ID	Name	Category	Price	Quantity	Image	Actions
35	Dalda Banaspati	Dairy Product	700.00	50		<div>EditDelete</div>
36	Dalda Cooking Oil	Oil	600.00	25		<div>EditDelete</div>
37	Tullo Banaspati	Dairy Product	680.00	21		<div>EditDelete</div>
38	Tullo Cooking Oil	Oil	570.00	33		<div>EditDelete</div>

## Edit Product

ProductName

Dalda Banaspati

Category

Dairy Product

Product Image

Choose File

No file chosen

Leave blank to keep current image (JPEG/PNG, max 5MB)

Price

Rs.

700.00

Quantity

50

← Back to List

Update Product

© 2025 - SmartGrocery Application

Assign Driver

DriverName

Enter driver's full name

Email

Enter email address

Password

Enter password

DriverPhone

Enter phone number

Address

Enter address

VehicleNo

Enter vehicle number

LicenseNo

Enter license number

+Assign Driver

Reset

© 2025 - SmartGrocery Application

### Driver List

+Add New Driver

ID	NAME	EMAIL	PHONE	PASSWORD	ADDRESS	VEHICLE NO.	LICENSE NO.	ASSIGNED ON
14	Khurram	khurram@gmail.com	03356247982	789	FB-area,Karachi	bv-896	c-258	May 08, 2025
12	Ali	ali@gmail.com	03352586941	123	Nazimabad, Karachi	12345	ab-123	May 08, 2025

© 2025 - SmartGrocery Application

### All Orders

Print All

Order #88 11 May 2025 05:48 pm

Delivered

Customer Details

Name: Emaz Ali Khan

Phone: 03358965471

Delivery Address: Shadman town

Order Information

Payment Method: Cash on Delivery

Payment Status: Paid

Delivery Date: 11 May 2025 05:55 pm

Driver: Ali (03352586941)

Vehicle No: 12345

PRODUCT ID	PRODUCT NAME	QTY	UNIT PRICE	SUBTOTAL
35	Dalda Banaspati	1	Rs. 700.00	Rs. 700.00
46	Ponam Flour (5 KG)	1	Rs. 500.00	Rs. 500.00
50	Olpers (1.5 Litre)	1	Rs. 320.00	Rs. 320.00
53	Prima Blend (Dark)	2	Rs. 100.00	Rs. 200.00

# Driver:

Smart Grocery

Home

My Deliveries

Logout

Welcome, Ali!

3

Delivered Orders

3

Pending Orders

📍 Delivery Progress

50% Delivered

Total Orders: 6 Completion Rate: 50%

⚡ Quick Actions

☰ View My Assigned Orders

👤 My Profile

Smart Grocery

Home

My Deliveries

Logout

👤 Driver Profile

Dashboard / Profile

Ali

Driver ID: 12

✉

📞

💬

✓

📄 Basic Information

Full Name	Ali
Email Address	ali@gmail.com
Phone Number	03352586941
Address	Nazimabad, Karachi
Vehicle Number	🚗 12345
License Number	ab-123
Member Since	May 08, 2025

© 2025 - SmartGrocery Application

## My Assigned Orders

Order ID	Customer Name	Customer Phone	Order Date	Total Amount	Delivery Address	Payment Method	Payment Status	Delivery Status	Action
88	Emaz Ali Khan	03358965471	11 May 2025 05:48 pm	Rs2,700.00	Shadman town	Cash on Delivery	Paid	Delivered	Delivered
86	Emaz Ali Khan	03358965471	08 May 2025 06:59 pm	Rs3,450.00	Block	Cash on Delivery	Paid	Delivered	Delivered
85	Sumaika Asif	03358963258	08 May 2025 06:59 pm	Rs3,700.00	Gulshan,Karachi	Cash on Delivery	Pending	Pending	Mark Delivered
84	Sumaika Asif	03358963258	08 May 2025 06:58 pm	Rs2,350.00	Gulshan,Karachi	Cash on Delivery	Pending	Pending	Mark Delivered
83	Sumaika Asif	03358963258	08 May 2025 06:58 pm	Rs5,310.00	Gulshan,Karachi	Online	Paid	Delivered	Delivered
82	Abdul Rafay	03358962369	08 May 2025 06:56 pm	Rs2,010.00	FB-Area, Karachi	Cash on Delivery	Pending	Pending	Mark Delivered