# Entity Relationship (ER) Diagram Documentation

## What is an ER Diagram?

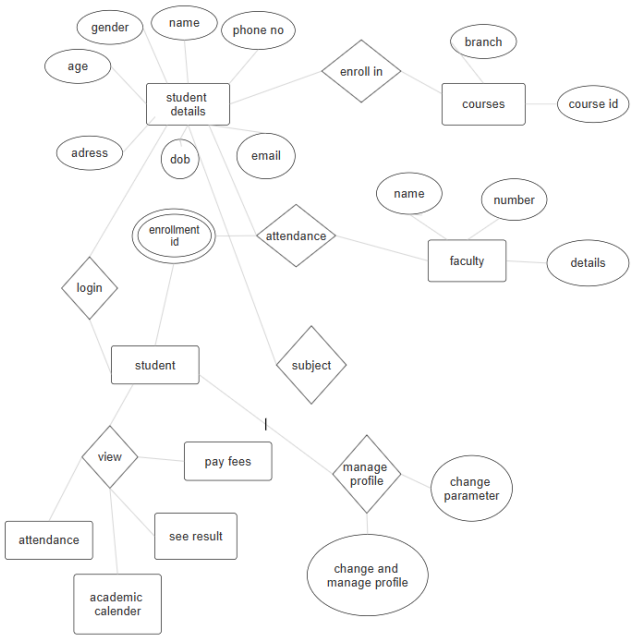
An Entity-Relationship (ER) Diagram is a type of flowchart that illustrates how entities such as people, objects, or concepts relate to each other within a system. ER diagrams are widely used in database design to visually represent the structure of data and the relationships between different entities.

## Use Cases of ER Diagrams

- - Database Design: Helps in designing relational databases by mapping out entities and relationships.
- - System Analysis: Used to analyze system requirements and clarify the data requirements of a system.
- - Communication Tool: Assists in communication between developers, database designers, and stakeholders.
- - Documentation: Provides a visual representation of data flow and structure for documentation purposes.
- - Problem Solving: Helps in identifying redundant data and improving system efficiency.

## ER Diagram of Student Information Management System

The following ER diagram represents the Student Information Management System:




## Entities and Attributes

### Student Details

Attributes:

- - Name
- - Gender
- - Age
- - Phone No
- - Address
- - DOB
- - Email
- - Enrollment ID

### Student

Attributes:

- - Login
- - View
- - Pay Fees
- - Manage Profile

### Courses

Attributes:

- - Course ID
- - Branch

### Faculty

Attributes:

- - Name
- - Number
- - Details

### Attendance

Attributes:

- - Subject

### Academic Calendar

Attributes:

- - Events, holidays, schedules

## Relationships

### Student Details

Relationships:

- - Enroll in Courses
- - Attendance

### Student

Relationships:

- - Login to access profile
- - View attendance & results
- - Pay Fees

### Courses

Relationships:

- - Students Enroll
- - Associated with Faculty

### Faculty

Relationships:

- - Handles Subjects
- - Manages Attendance

# E-Commerce Order Management System - Sequence Diagrams & Use Cases

## 1. Introduction

This document explains the sequence diagrams related to an E-Commerce Order Management System. The diagrams showcase two main processes: order placement with payment and order cancellation with refund handling. Additionally, corresponding use cases are described to explain the system's behavior.

## 2. Sequence Diagram: Order Placement and Payment

The following sequence diagram explains how a customer browses products, adds them to the cart, places an order, and proceeds with payment. Depending on the payment status, the system either confirms the order and schedules delivery or cancels the process and releases the reserved stock.

Key Steps:

1. 1. Customer browses products on the website.
2. 2. Website checks availability with inventory and provides product info.
3. 3. Customer adds items to cart and places the order.
4. 4. Website reserves stock and initiates payment processing.
5. 5. If payment succeeds, the order is confirmed and delivery is scheduled.
6. 6. If payment fails, reserved stock is released.

## 3. Sequence Diagram: Order Cancellation and Refund

This sequence diagram shows how the system handles order cancellation. The process depends on whether the order has already been shipped or not.

Scenarios:

7. 1. If the order is NOT shipped:

- - Reserved stock is released back to inventory.
- - Payment refund is processed immediately.

8. 2. If the order is ALREADY shipped:

- - Website requests return or delivery stop with the shipping partner.
- - Once the item is returned, refund is processed.

## 4. Use Cases

### Use Case 1: Place Order

Actors: Customer, Website, Inventory, Payment, Shipping

Precondition: Customer has selected items to purchase.

Steps:

9. 1. Customer browses and selects products.
10. 2. Customer places the order.
11. 3. Website reserves stock from inventory.
12. 4. Website processes payment.

13. 5. If successful, order is confirmed and delivery scheduled.

Postcondition: Order placed successfully or stock released if payment fails.

### Use Case 2: Cancel Order

Actors: Customer, Website, Inventory, Payment, Shipping

Precondition: Customer has placed an order.

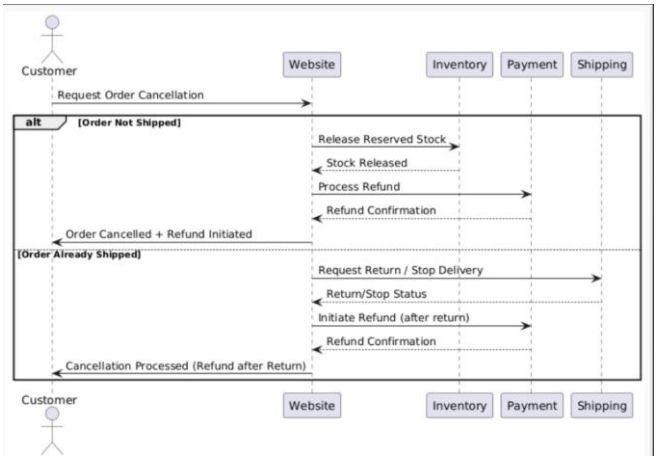
Steps:

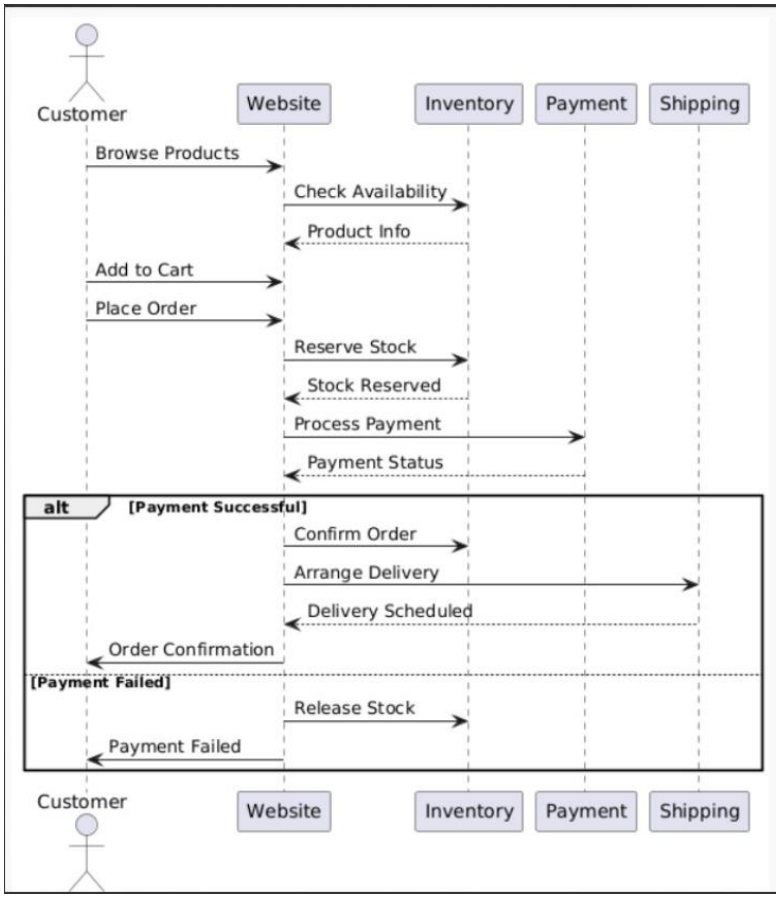
14. 1. Customer requests order cancellation.
15. 2. If the order is not shipped: stock is released and refund is initiated.
16. 3. If the order is shipped: website initiates return or delivery stop.
17. 4. Refund is processed after confirmation.

Postcondition: Customer receives refund after successful cancellation.

## 5. How These Diagrams Were Made

The diagrams are created using UML (Unified Modeling Language) sequence diagram notation. Actors such as Customer, Website, Inventory, Payment, and Shipping represent the entities involved in the process. The arrows indicate the flow of messages or actions between these entities. The 'alt' block is used to represent alternative scenarios, such as successful or failed payments, and order cancellation before or after shipment.



# Use Case Diagram

## 1. What is a Use Case Diagram?

A **Use Case Diagram** is a **UML (Unified Modeling Language)** diagram that represents the **interactions between actors (users or systems)** and the **functions (use cases)** of a system.

- It shows **what the system does** (functional requirements) instead of **how it is implemented**.
- It is widely used in **requirement analysis** and **system design** phases.

---

## 2. Why Do We Need a Use Case Diagram?

- To **visualize the functional scope** of the system in a simple diagram.
- To **communicate system behavior** to developers, testers, and clients.
- To **document requirements** in a structured way.
- To **identify actors and their roles**, ensuring nothing is missed.
- To serve as a **starting point for design, testing, and project planning**.
- To **bridge the gap** between technical and non-technical stakeholders.

---

## 3. Key Elements of a Use Case Diagram

### 🔹 Actors

- Represent **users or external systems** that interact with the system.
- Types:
    - **Primary Actor** → Initiates interaction (e.g., Patient books appointment).

---

    - **Secondary Actor** → Supports the process (e.g., Doctor updates records).

### 🔹 Use Cases

- Functionalities or actions performed by the system.
- Represented by **ellipses (ovals)**.
- Each use case should provide **value to an actor**.

### 🔹 System Boundary

- Represented by a **rectangle** enclosing all use cases.
- Defines the **scope** of the system – what is inside (system functions) vs outside (actors).

### 🔹 Relationships

1. **Association (—)** → Connects actors with use cases.
2. **Include (<>)** → Mandatory relationship; a use case always uses another.
    - Example: “Book Appointment” <> “Check Doctor Availability”.
3. **Extend (<>)** → Optional behavior extending a use case.
    - Example: “Generate Bill” <> “Book Appointment”.
4. **Generalization** → Inheritance between actors or use cases.
    - Example: “Admin” inherits permissions from “Receptionist”.

## 4. How to Create a Use Case Diagram (Step by Step)

1. **Define the system boundary** → Draw a rectangle and name it (e.g., *Hospital Appointment System*).
2. **Identify actors** → List all users and external systems that interact with the system.
3. **Identify use cases** → Write down major functionalities needed by each actor.
4. **Place use cases inside the system boundary**.
5. **Connect actors with use cases** using associations.
6. **Add relationships** (include, extend, generalization) where necessary.
7. **Review and validate** → Ensure no actor or functionality is missed.

---

## 5. Example – Hospital Appointment System

### Actors:

1. **Patient** – Books and manages appointments.
2. **Doctor** – Views appointments, updates medical records.
3. **Receptionist** – Assists with scheduling, generates reports.
4. **Administrator (optional)** – Manages staff, system settings.

### Use Cases:

- **Book Appointment**
- **Cancel Appointment**
- **View Appointment Details**
- **Check Doctor Availability**
- **Update Patient Records**
- **Generate Report**
- **Manage Users** (Admin)

### Relationships:

- **Patient** → **Book Appointment** (association).
- **Book Appointment** <> **Check Doctor Availability**.
- **Receptionist** → **Generate Report**.
- **Doctor** → **Update Patient Records**.
- **Administrator** → **Manage Users** (generalization of Receptionist).

---

## Detailed Example Walkthrough:

- A **Patient** can **book an appointment**. While doing this, the system must **check doctor availability** (<> relationship).


View Different Doctor
View Doctor Profiles & Description
Search for Available Appointment
Book an Appointment
Manage Appointment
Manage Appointment
Modify Appointment
View Patient records and Information
Send a Reminder to Patient
View all specific Appointment
Give Prescription and Chat
Update/Delete Records
Actor
Patient
Actor
Doctor