Project Specification: Logistics Pro Shipment System

1. Overview

Global Cargo Solutions is a logistics firm that requires a C# module to manage international shipping costs. The system must validate shipment identifiers and calculate costs based on transport mode, weight, and storage duration.

---

2. Functional Requirements

2.1 Data Models

Implement the following class structure:

Class: Shipment

Property Name	Datatype	Access Modifier
`ShipmentCode`	`string`	public
`TransportMode`	`string`	public
`Weight`	`double`	public
`StorageDays`	`int`	public

Class: ShipmentDetails

- **Inheritance:** Must inherit from the `Shipment` class.
- **Method:** **`ValidateShipmentCode()`**
    - **Return Type:** `bool`
    - **Logic:**
        1. Length must be exactly **7 characters**.
        2. Prefix must be **"GC#"**.

        3. Characters after the prefix must be **digits**.
- **Method:** **`CalculateTotalCost()`**
    - **Return Type:** `double` (Return value rounded to 2 decimal places).
    - **Formula:** $TotalCost = (Weight \times RatePerKg) + \sqrt{StorageDays}$

---

3. Business Rules

Transport Mode	Rate per Kg (USD)
Sea	15.00
Air	50.00
Land	25.00

Note: The `TransportMode` input is case-sensitive.

---

4. Execution Logic (Program Class)

1. **Input Phase:** Prompt the user for the `ShipmentCode`.
2. **Validation Phase:** Call the validation method.
    - If **False**: Display `"Invalid shipment code"` and terminate gracefully.
    - If **True**: Proceed to collect `TransportMode`, `Weight`, and `StorageDays`.
3. **Calculation Phase:** Invoke the cost calculation and display the result.

---

5. Sample Test Cases

Test Case 1: Success

- **Input ID:** `GC#1001`
- **Mode:** `Air`
- **Weight:** `10`
- **Storage:** `16`

- **Expected Output:** `The total shipping cost is 504.00`

Test Case 2: Validation Failure

- **Input ID:** `BK#5555`
- **Expected Output:** `Invalid shipment code`

Part 1: The Solution Code

C#
```
using System;

namespace LogisticsApp
{
    // Base Class
    public class Shipment
    {
        public string ShipmentCode { get; set; }
        public string TransportMode { get; set; }
        public double Weight { get; set; }
        public int StorageDays { get; set; }
    }

    // Derived Class with Business Logic
    public class ShipmentDetails : Shipment
    {
        public bool ValidateShipmentCode()
        {
            // Rule: Length 6, starts with GC#, followed by 3 digits
            if (string.IsNullOrEmpty(ShipmentCode) || ShipmentCode.Length !=
6)
                return false;

            if (!ShipmentCode.StartsWith("GC#"))
                return false;

            string numericPart = ShipmentCode.Substring(3);
            return int.TryParse(numericPart, out _);
        }

        public double CalculateTotalCost()
        {
            double ratePerKg = 0;

            switch (TransportMode)
            {
                case "Sea": ratePerKg = 15; break;
                case "Air": ratePerKg = 50; break;
                case "Land": ratePerKg = 25; break;
                default: return 0.00;
            }

            // Formula: (Weight * Rate) + Sqrt(StorageDays)
            double cost = (Weight * ratePerKg) + Math.Sqrt(StorageDays);
            return Math.Round(cost, 2);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            ShipmentDetails ship = new ShipmentDetails();

            Console.WriteLine("Enter the shipment code");
            ship.ShipmentCode = Console.ReadLine();

            if (ship.ValidateShipmentCode())
            {
                Console.WriteLine("Enter transport mode");
                ship.TransportMode = Console.ReadLine();

                Console.WriteLine("Enter weight");
                ship.Weight = double.Parse(Console.ReadLine());

                Console.WriteLine("Enter storage days");
                ship.StorageDays = int.Parse(Console.ReadLine());

                Console.WriteLine($"The total shipping cost is
{ship.CalculateTotalCost():F2}");
            }
            else
            {
                Console.WriteLine("Invalid shipment code");
            }
        }
    }
}
```

Part 2: Unit Test Suite (NUnit)

If you are using a testing framework like NUnit or MSTest, these tests will ensure the code meets all the requirements specified in your Word template.

C#
```
using NUnit.Framework;

namespace LogisticsTests
{
    [TestFixture]
    public class ShipmentTests
    {
        private ShipmentDetails _details;
        [SetUp]
        public void Setup() => _details = new ShipmentDetails();

        [Test]
        public void ValidateShipmentCode_ValidCode_ReturnsTrue()
        {
            _details.ShipmentCode = "GC#123";
            Assert.IsTrue(_details.ValidateShipmentCode());
        }

        [Test]
        [TestCase("AX#123")]   // Wrong Prefix
        [TestCase("GC#12")]    // Too short
        [TestCase("GC#1234")]  // Too long
        [TestCase("GC#ABC")]   // Not numeric
        public void ValidateShipmentCode_InvalidCodes_ReturnsFalse(string 
code)
        {
            _details.ShipmentCode = code;
            Assert.IsFalse(_details.ValidateShipmentCode());
        }

        [Test]
        public void CalculateTotalCost_AirMode_ReturnsCorrectValue()
        {
            _details.TransportMode = "Air";
            _details.Weight = 10;
            _details.StorageDays = 16; // Sqrt is 4
            // (10 * 50) + 4 = 504.00
            Assert.AreEqual(504.00, _details.CalculateTotalCost());
        }

        [Test]
        public void 
CalculateTotalCost_CaseSensitivity_ReturnsZeroForLowercase()
        {
            _details.TransportMode = "air"; // Should fail because it's not 
"Air"
            _details.Weight = 10;
            _details.StorageDays = 16;
            Assert.AreEqual(0.00, _details.CalculateTotalCost());
        }
    }
}
```

Summary of what you've created:

1. **A Domain Model:** Using inheritance for clean data separation.
2. **Validation Logic:** Using string manipulation and parsing.
3. **Mathematical Logic:** Using `Math.Sqrt` and `Math.Round`.
4. **Verification:** A test suite to prevent future bugs

C# Boilerplate Implementation

C#

```
using System;

namespace LogisticsApp
{
    // --- DATA MODEL LAYER ---
    public class Shipment
    {
        // Public properties as per requirements
        public string ShipmentCode { get; set; }
        public string TransportMode { get; set; }
        public double Weight { get; set; }
        public int StorageDays { get; set; }
    }

    // --- BUSINESS LOGIC LAYER ---
    public class ShipmentDetails : Shipment
    {
        /// <summary>
        /// Validates if the ShipmentCode follows the "GC#NNN" format.
        /// </summary>
        public bool ValidateShipmentCode()
        {
            // TODO: Implement length check (7), prefix check (GC#), and
numeric suffix check
            return false;
        }

        /// <summary>
        /// Calculates the total cost based on TransportMode, Weight, and
StorageDays.
        /// </summary>
        public double CalculateTotalCost()
        {
            double pricePerKg = 0;

            // TODO: Implement case-sensitive switch for Sea (15), Air (50),
Land (25)

            // TODO: Calculate: (Weight * pricePerKg) +
Math.Sqrt(StorageDays)

            return 0.00;
        }
    }

    // --- PRESENTATION LAYER ---
    class Program
    {
        static void Main(string[] args)
        {
            ShipmentDetails shipment = new ShipmentDetails();
            // 1. Get and Validate Shipment ID
            Console.WriteLine("Enter the shipment code:");
            shipment.ShipmentCode = Console.ReadLine();

            if (shipment.ValidateShipmentCode())
            {
                // 2. Get additional inputs
                Console.WriteLine("Enter transport mode (Sea/Air/Land):");
                shipment.TransportMode = Console.ReadLine();

                Console.WriteLine("Enter weight (kg):");
                shipment.Weight = double.Parse(Console.ReadLine());

                Console.WriteLine("Enter storage days:");
                shipment.StorageDays = int.Parse(Console.ReadLine());

                // 3. Output Result
                double finalCost = shipment.CalculateTotalCost();
                Console.WriteLine($"The total shipping cost is
{finalCost:F2}");
            }
            else
            {
                Console.WriteLine("Invalid shipment code");
            }

            // Keep console open
            Console.WriteLine("\nPress any key to exit...");
            Console.ReadKey();
        }
    }
}
```