

**UE22CS352B - Object Oriented Analysis & Design**

**Mini Project Report**

**Supply Chain Management System**

***Submitted by:***

***PES2UG22CS442 Rishi Gupta***

***PES2UG22CS449 Rithul Rakesh***

***PES2UG22CS459 Rohit Maddaly***

Semester VI - Section H

Facultly Name

**Dr. Mannar Mannan J**

**January - May 2025**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100ft Ring Road, Bengaluru – 560 085, Karnataka, India

**Problem Statement**

This project aims to develop a **modular, Java-based Supply Chain Management System** that allows suppliers to manage products, retailers to place orders, and administrators to process and monitor order fulfillment.

**Key Features**

Supplier Management

* Add and store suppliers with unique IDs and names.
* Each supplier can manage multiple products.

Product Inventory Management

* Add products to specific suppliers.
* Track stock quantity, price, and product details.
* Automatically reduce stock on order processing.

Retailer Management

* Register retailers with their details.
* View and manage retailer-specific order histories.

Order Lifecycle

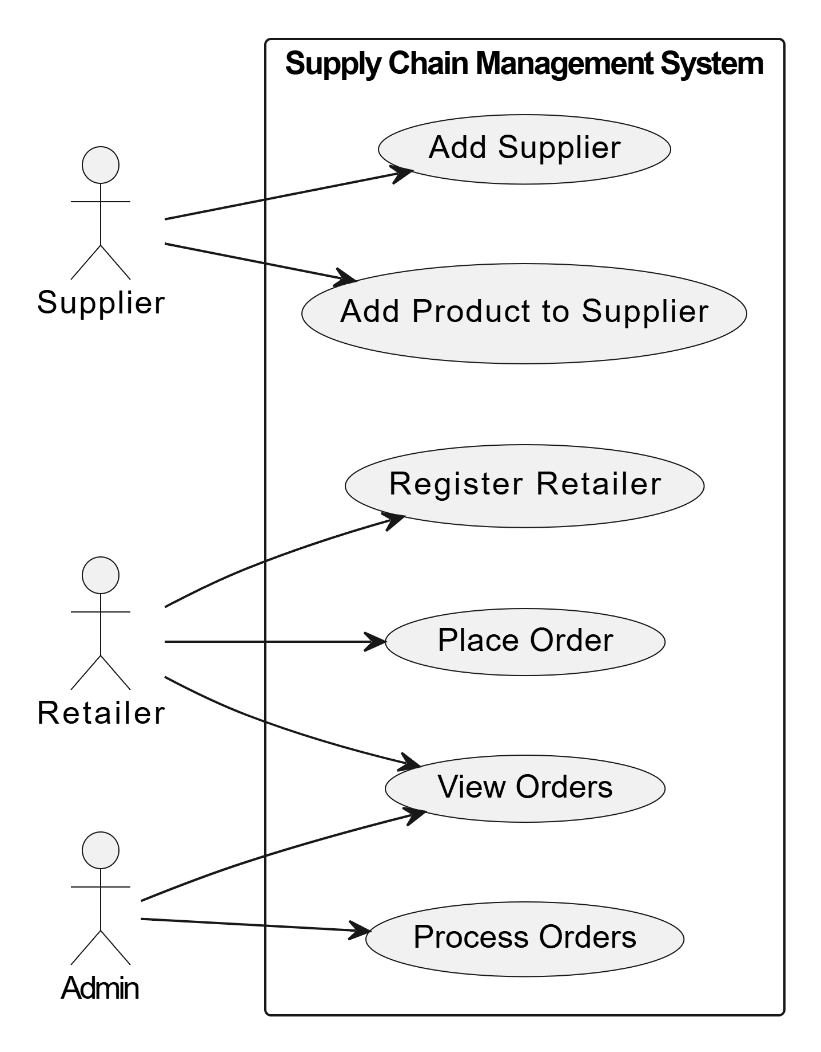
* Place orders by retailers for specific products.
* Automatically check stock availability and update product quantity.
* Orders are marked as “Shipped” if fulfilled, or “Failed” if stock is insufficient.

Graphical User Interface (GUI)

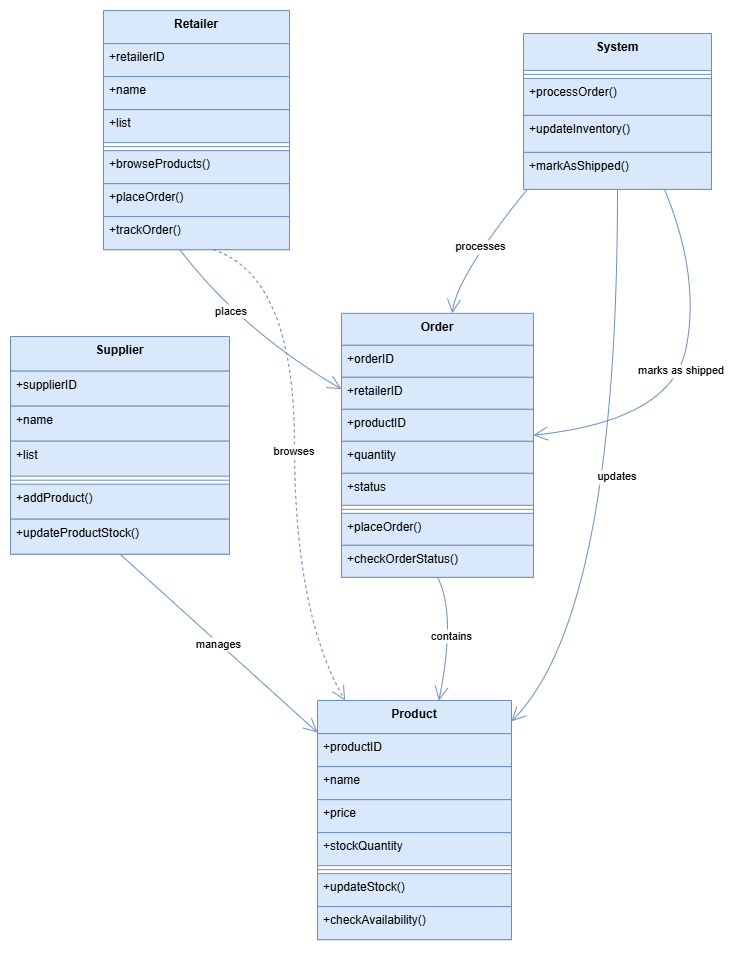
* Java Swing-based interface with:
  + Tabs for suppliers, products, retailers, and orders.
  + Dialog-driven forms for adding entities and placing orders.
  + Real-time table updates reflecting inventory and order status.

**Models**

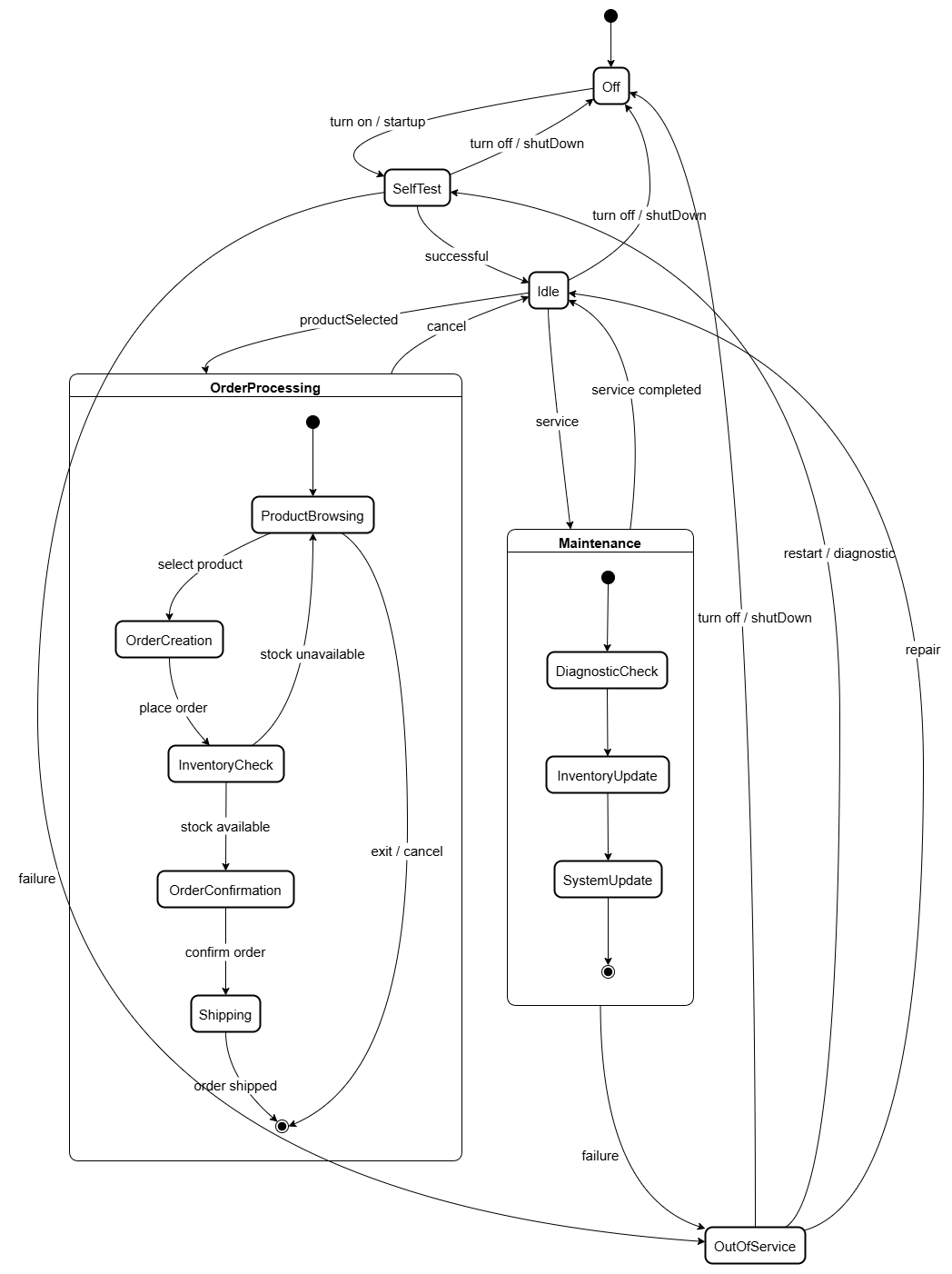
**Use Case Diagram:**



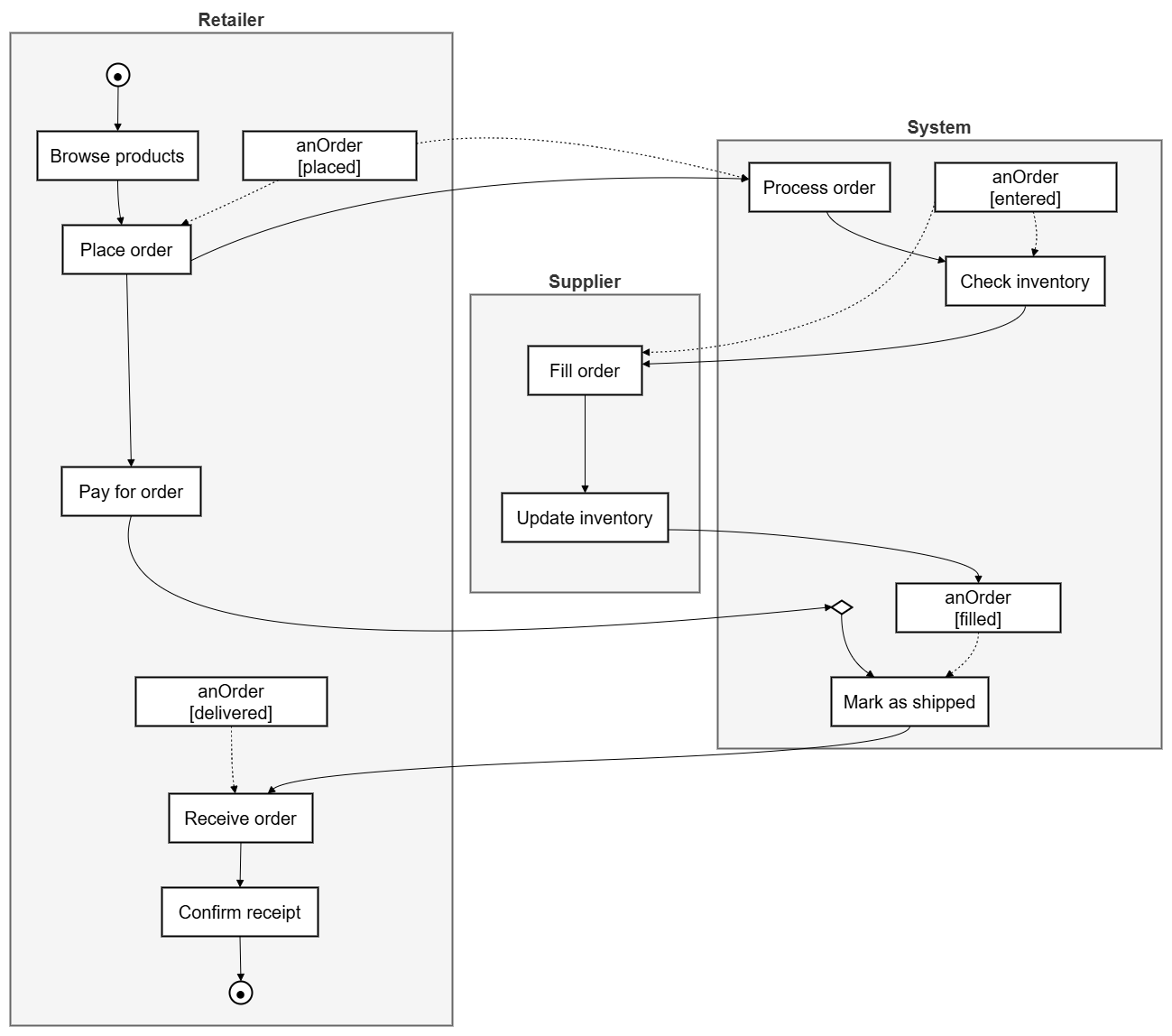
**Class Diagram:**



**State Diagram:**

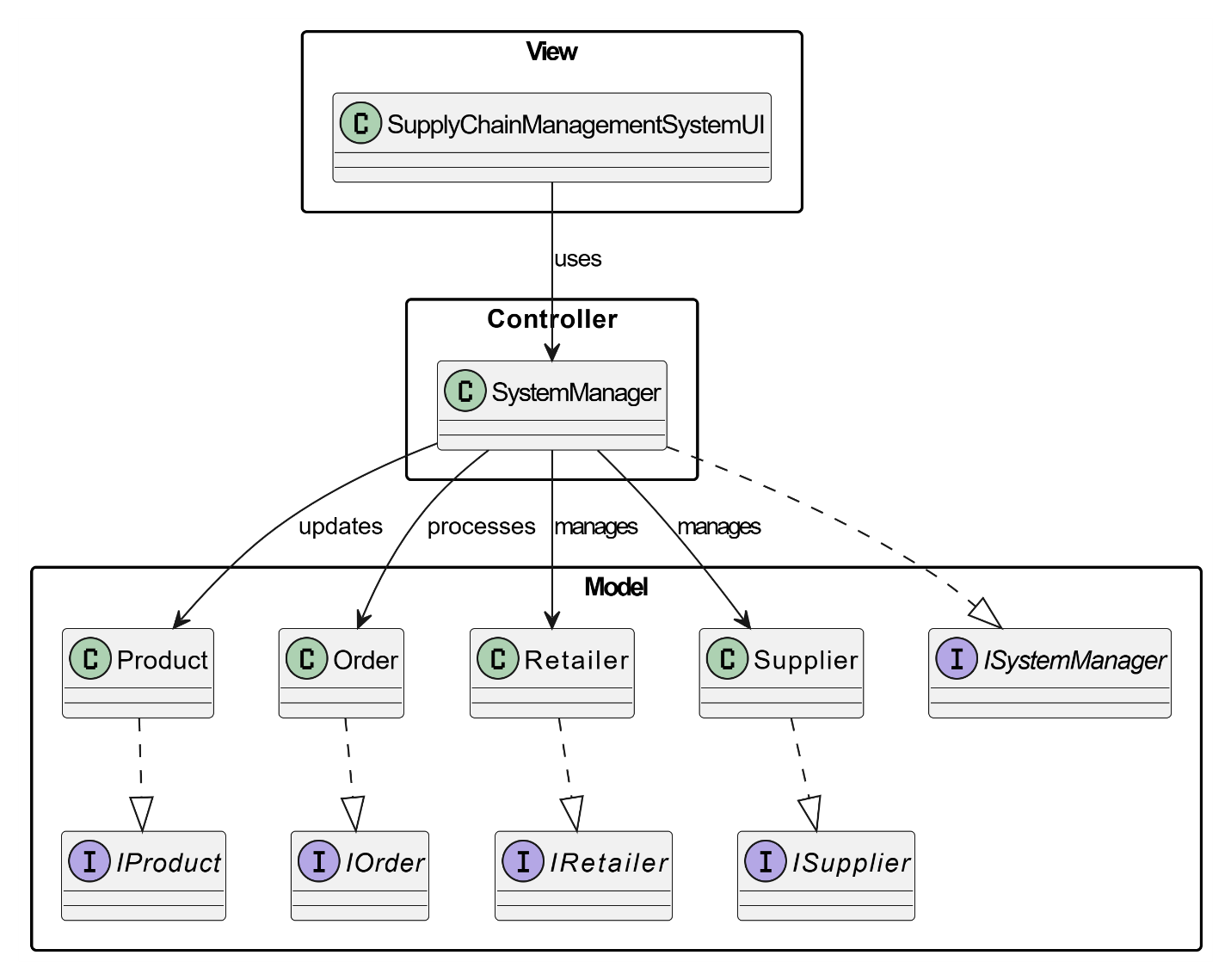


**Activity Diagram:**



**Architecture Patterns, Design Principles, and Design Patterns**

**Architecture Patterns: Model - View - Controller Pattern (MVC):**



**Design Principles:**

1. Single Responsibility Principle (SRP) – SOLID
   * Each class handles one responsibility
   * Easier to debug, test, and maintain.
2. Interface Segregation Principle (ISP) – SOLID
   * We have created separate interfaces for different roles - IProduct, IOrder, IRetailer, ISupplier, ISystemManager
   * This avoids fat interfaces and keeps each interface focused. Each entity only implements the methods it actually needs.
3. Dependency Inversion Principle (DIP) – SOLID
   * The SupplyChainManagementSystemUI use ISystemManager instead of the concrete SystemManager Similarly, Retailer holds List<IOrder> instead of List<Order>.
   * This allows for easier testing and decoupling of modules.
4. Separation of Concerns - UI logic (SupplyChainManagementSystemUI) is clearly separated from business logic (SystemManager, models).

**Design Patterns:**

MVC (Model–View–Controller)

* Model: Product, Order, etc.
* View: SupplyChainManagementSystemUI
* Controller: SystemManager

**Github link to the Codebase:**

**Screenshots:**

**Individual contributions of the team members:**

|  |  |
| --- | --- |
| Name | Module worked on |
| Rishi Gupta |  |
| Rithul Rakesh |  |
| Rohit Maddaly |  |