Information Technology University of the Punjab

Department of Computer and Software-Engineering

BSSE-23 B

Submitted by: MANAL RANA

Roll Number: BSSE23099

Assignment-1: UML Diagrams

Software Design and Architecture
SE203-T (Spring-25)

Information Technology University (ITU) – Lahore

Title: Inventory and order management system

SYSTEM DESCRIPTION:

This enables a user to place an order and successfully receive it.

Architecture used:

Layered (N-Tier) Architecture

This architecture follows a structured approach with multiple layers for separation of concerns.

Functional Requirements:

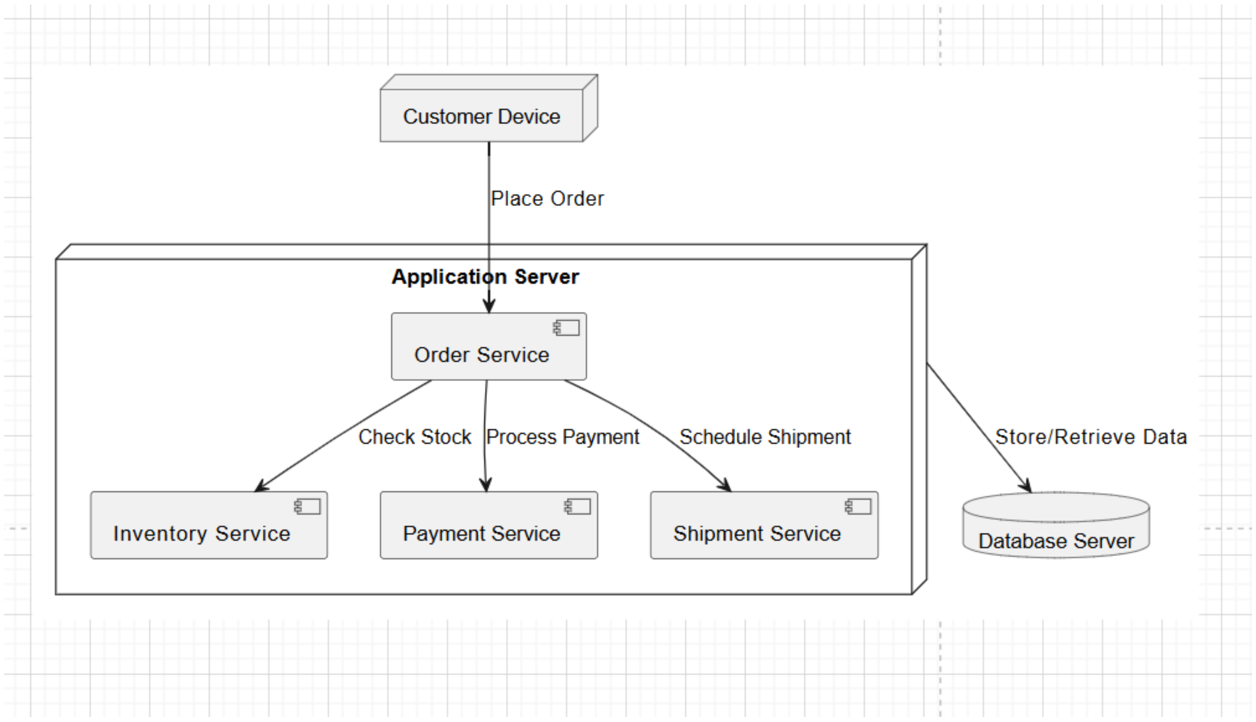
1. Admins, managers, and employees must log in with valid credentials.
2. Add, update, and remove inventory items.
3. Track stock levels and notify when stock is low.
4. Categorize items based on type, supplier, and location.
5. Customers can place, modify, and cancel orders.
6. System validates stock availability before confirming orders.
7. Supports multiple payment methods (credit card, PayPal, etc.).
8. Notify customers about delivery status updates.

NON- FUNCTIONAL REQUIREMENTS:

- The system should handle a growing number of orders and inventory items without performance issues.
- Inventory updates should reflect in real time.
- Order processing should take no more than 2 seconds.
- Data encryption for sensitive information (customer details, payments).
- Secure login with multi-factor authentication (MFA).
- System uptime should be **99.9%** to ensure smooth operations.

UML DIAGRAMS

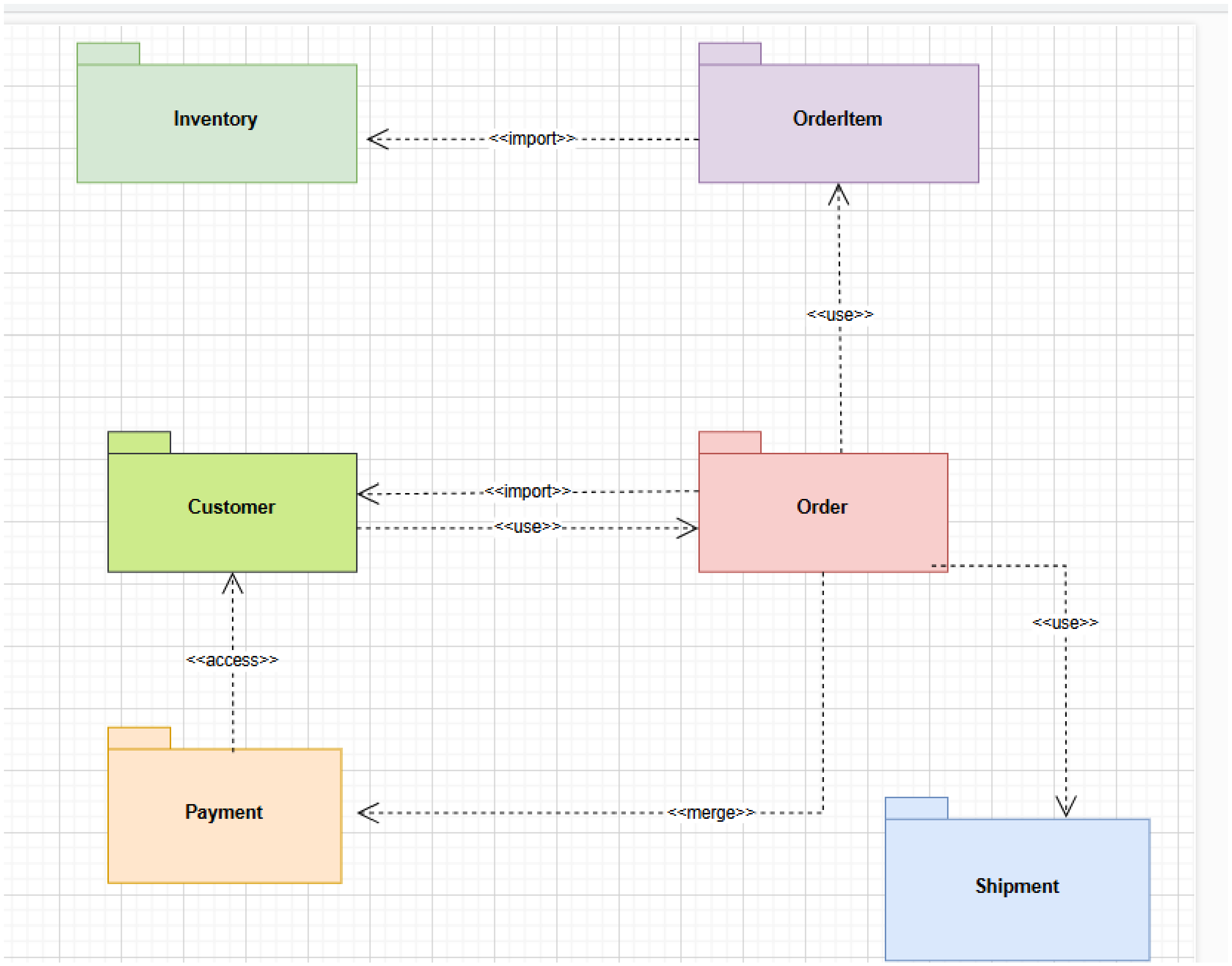
DEPLOYMENT DIAGRAM



Description:

This shows the state of the system when it is being deployed.

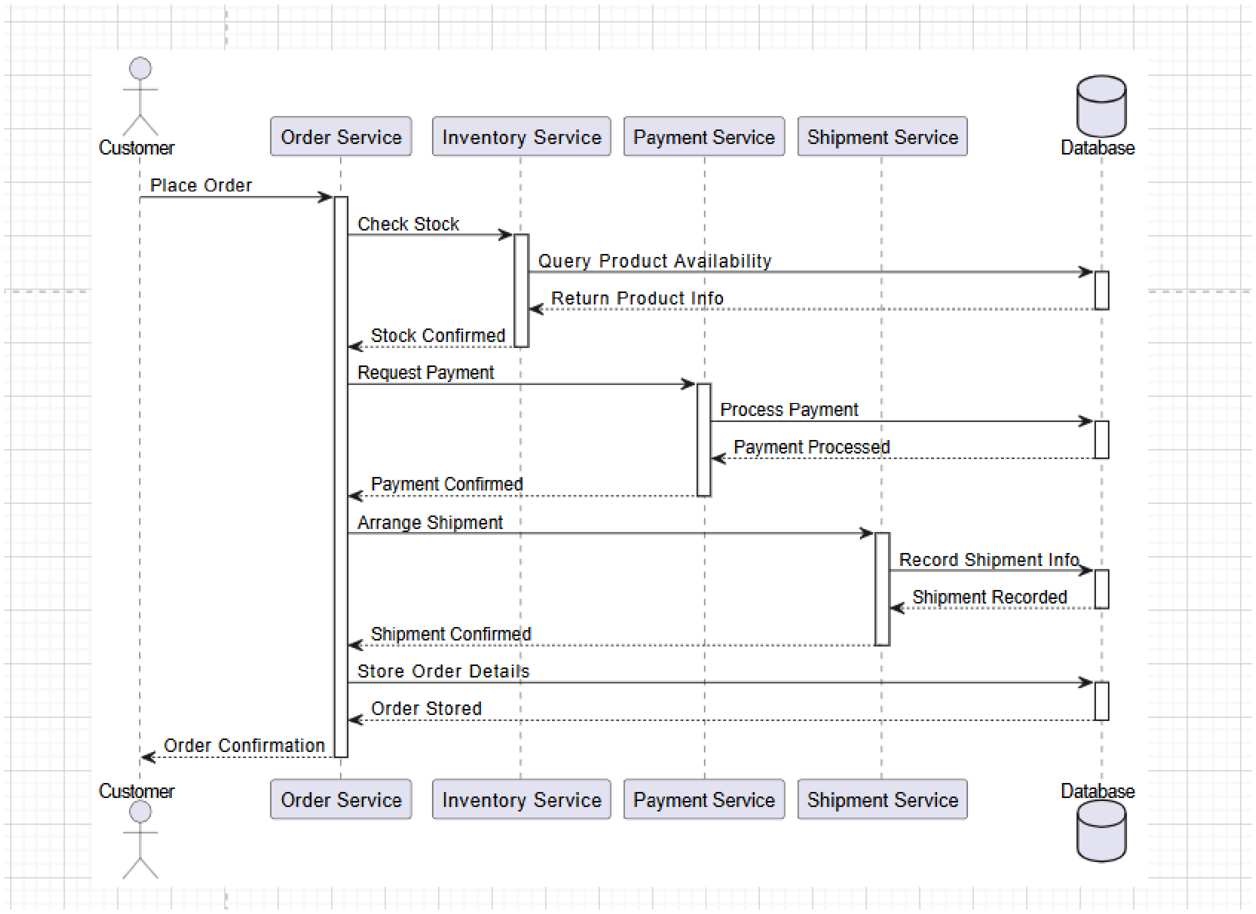
Package diagram:



Description:

This diagram shows how the customer places an order here the <<use>> arrow means that the customer simply uses the other package without owning it, then the order uses order item package in the same way, so the customer places an order and that order has a few items that have the references of the products or you can say the properties of the products that are in the inventory as it <<imports>>from it, which means that it imported the public properties of the inventory package, the order is then shipped and the payment class is merged with the order package as when the order is cancelled the payment is cancelled too, lastly the payment also imports from the customer as it needs customer credentials to process.

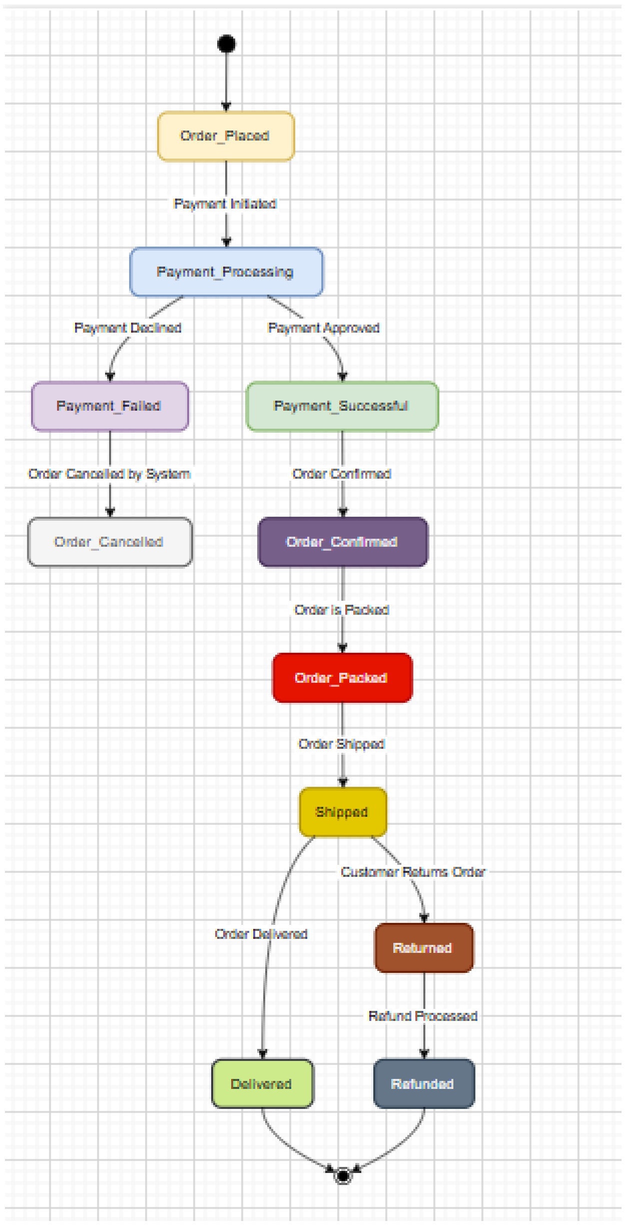
Sequence Diagram:



Description:

This shows the process of the over system, along with the timeline that is how long the process lasts or how much the program is used, here order service has the longest timeline as the order service acts as a coordinator between all the modules, that when an order is placed the service asks the inventory to check for the availability of the stock , then inventory works if the stock is there then the order service asks the payment service to do its job, then it asks the shipment service to deliver the package, and finally when the order is confirmed and is on its way then the function of order service ends therefore it has the longest timeline.

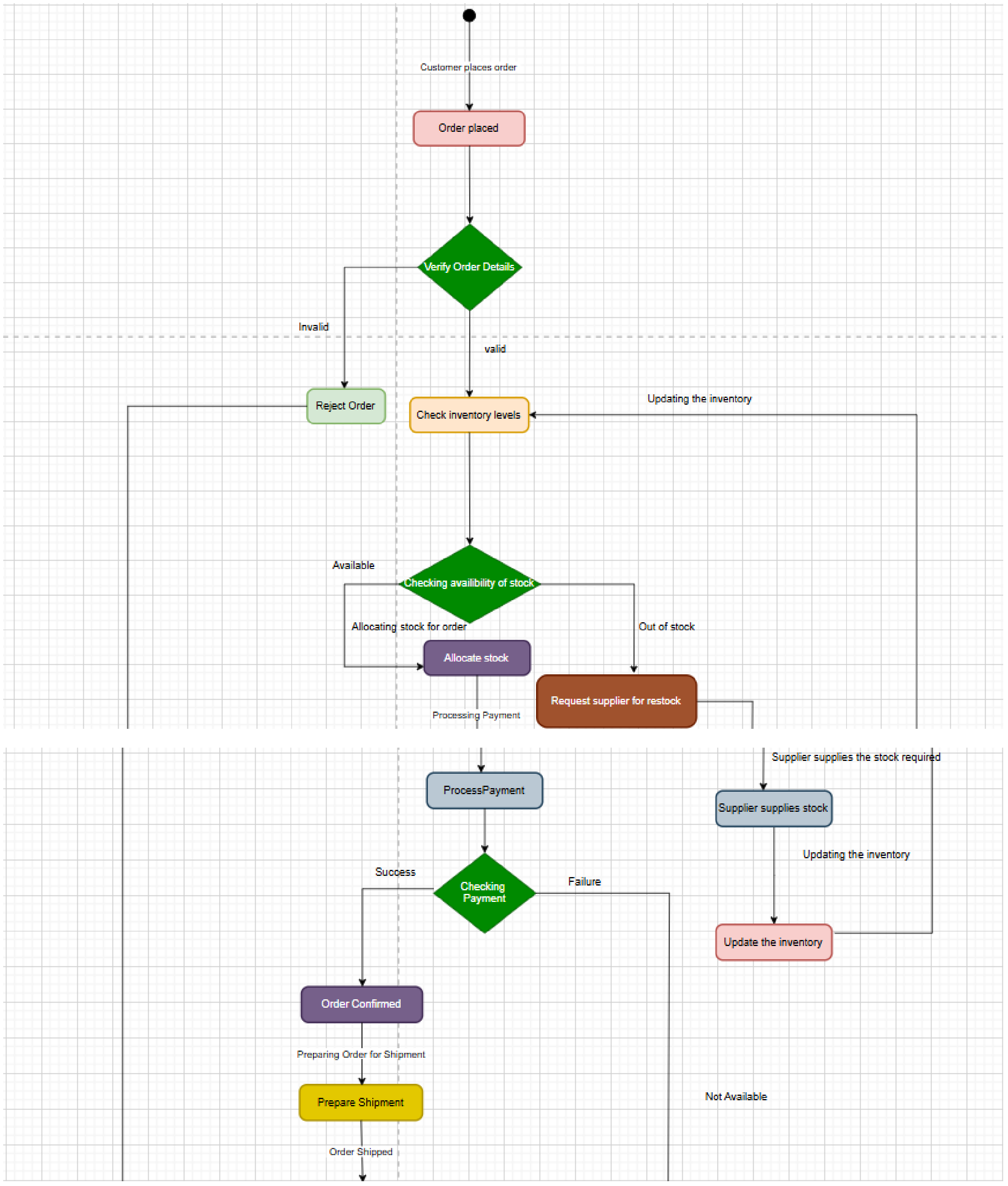
State diagram:

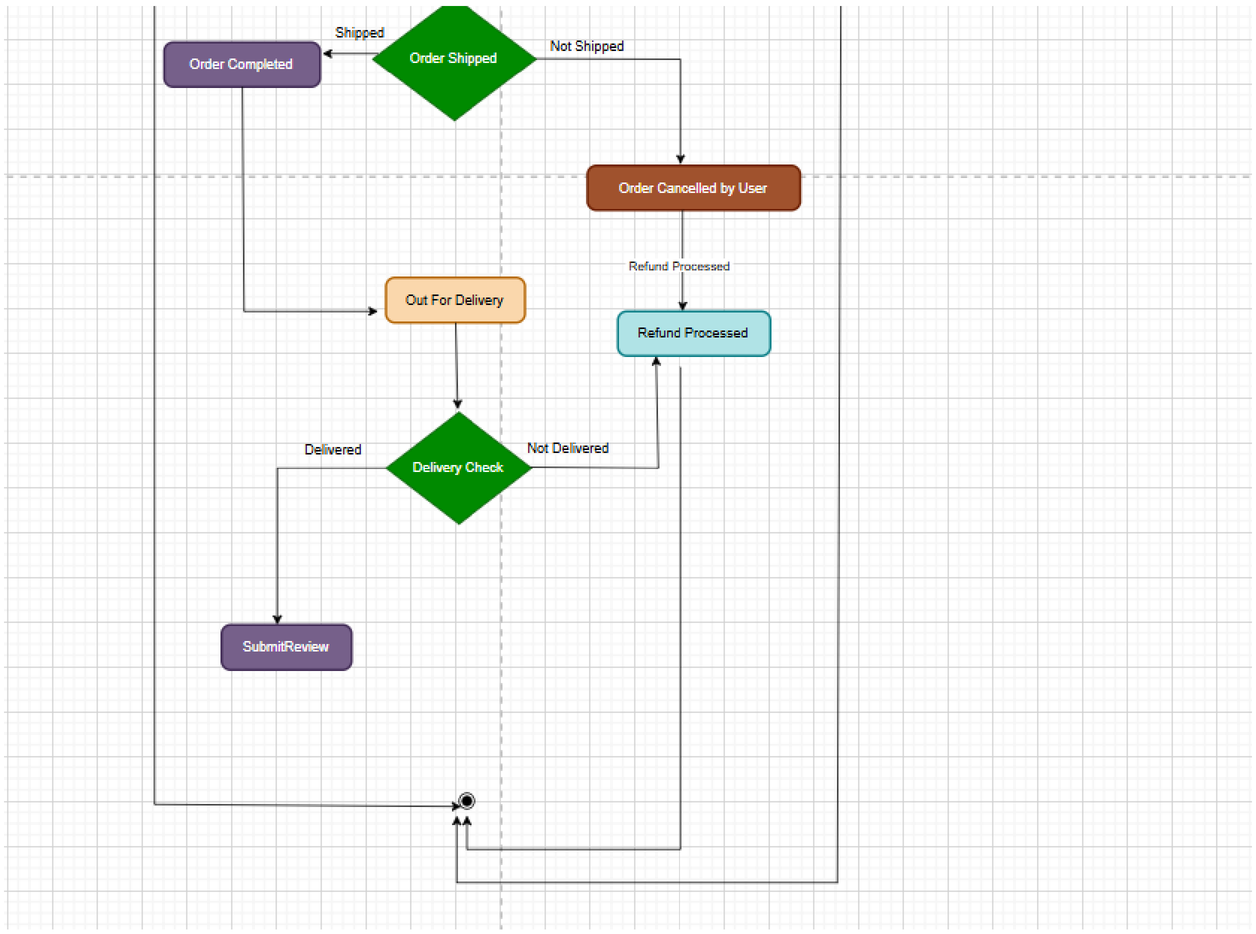


Description:

This shows various states achieved during the order is placed from it being confirmed to its payment and then shipment then finally the parcel is either delivered or refunded.

Activity Diagram:

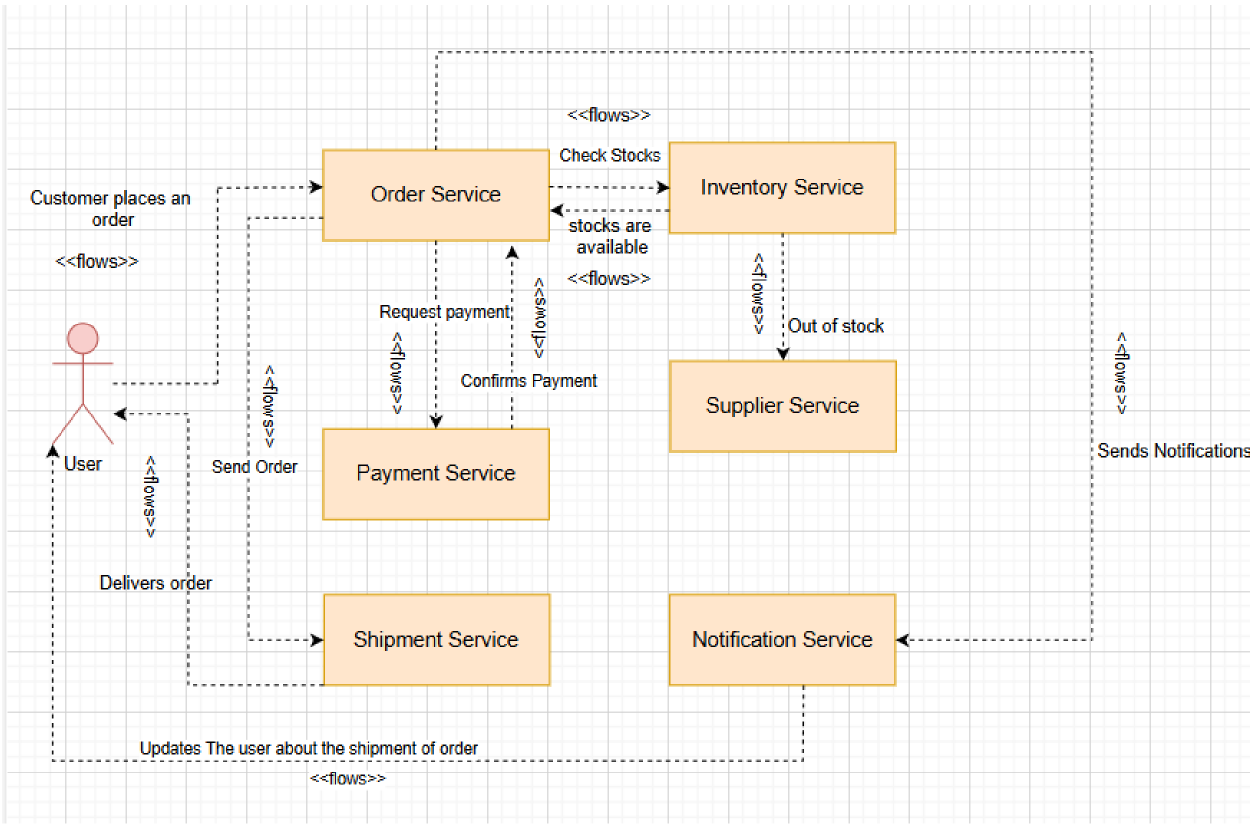

Customer places order
Order placed
Verify Order Details
Invalid
valid
Reject Order
Check inventory levels
Updating the inventory
Available
Checking availibility of stock
Allocating stock for order
Out of stock
Allocate stock
Request supplier for restock
Processing Payment
Supplier supplies the stock required
ProcessPayment
Supplier supplies stock
Updating the inventory
Success
Checking Payment
Failure
Update the inventory
Order Confirmed
Preparing Order for Shipment
Prepare Shipment
Not Available
Order Shipped



Description:

This shows the flow of events occurring throughout the system, it has some decisions showing the alternate paths taken, this shows how the user places and order, how it is processed, how it is shipped and how it is refunded in averse situations.

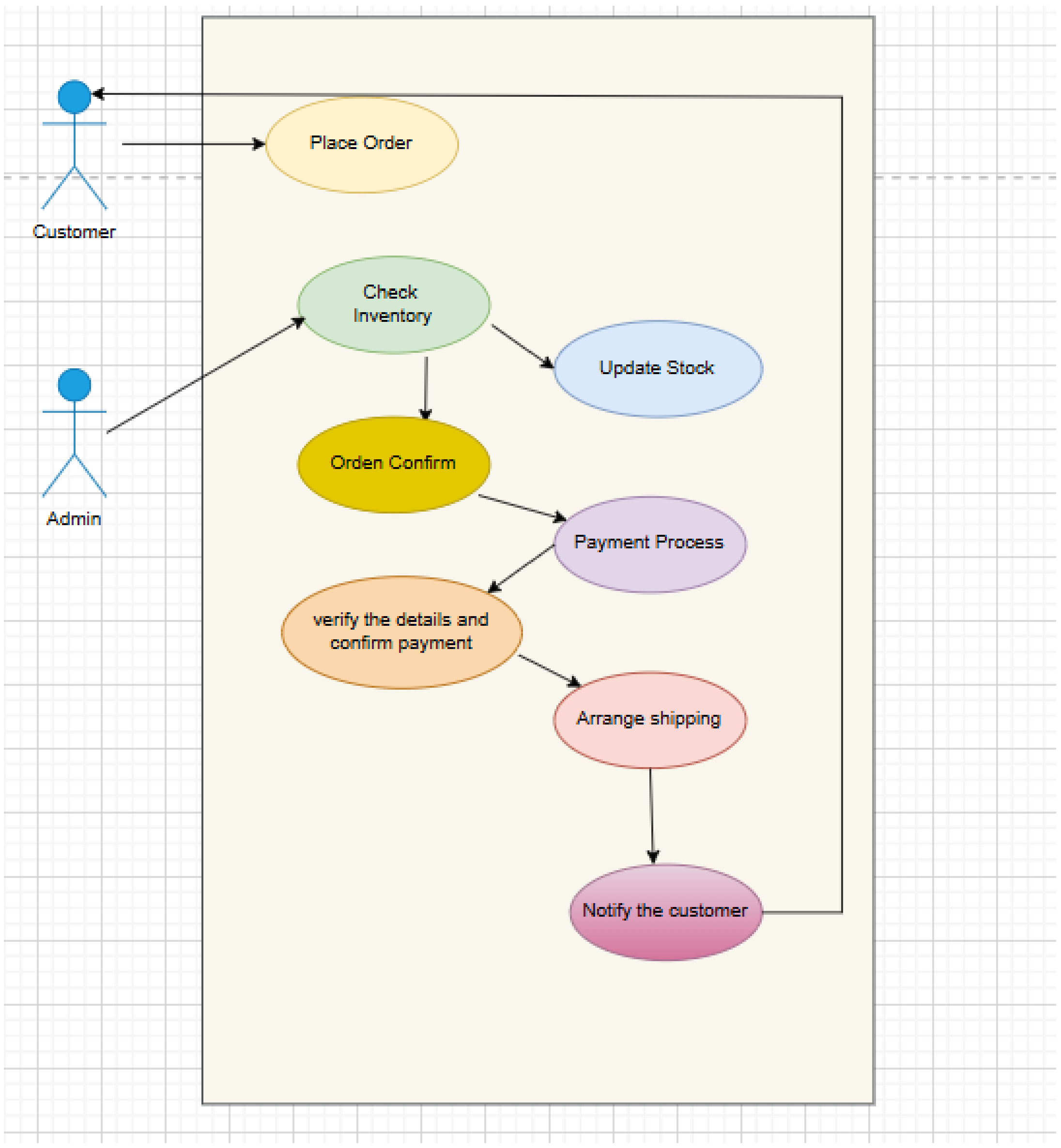
Information Flow Diagram:



Description:

This diagram shows how **data and information move** between different components, systems, or users.

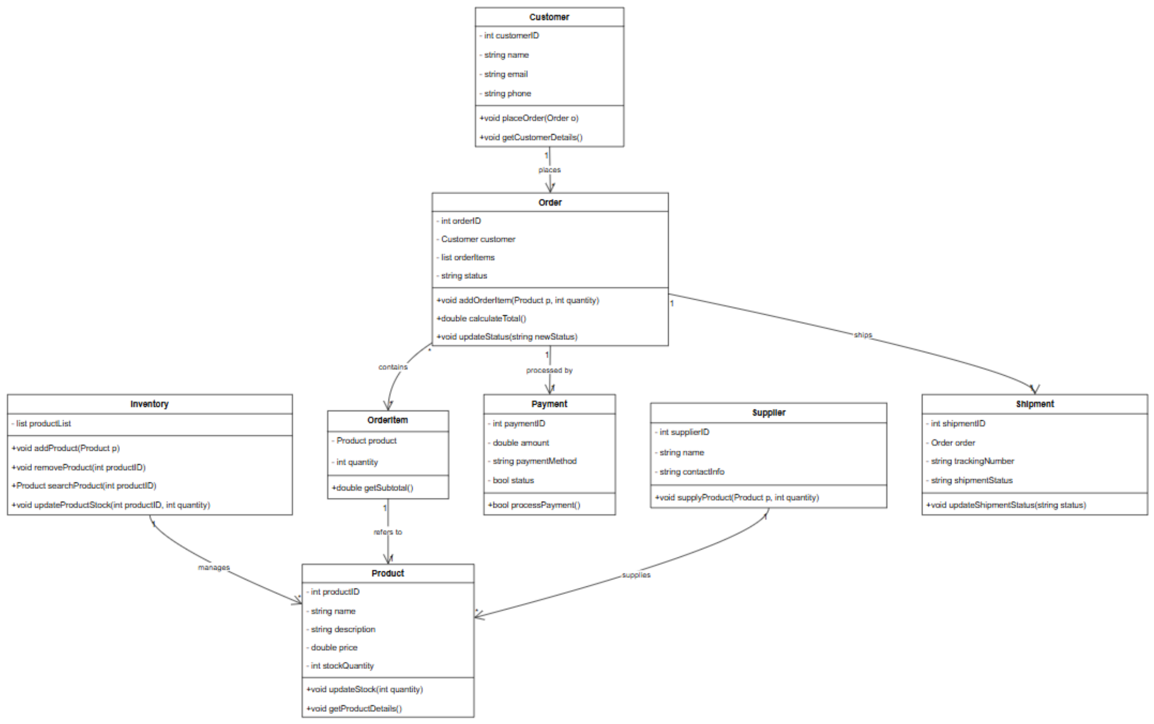
Use- Case diagram:



Description:

This shows the flow of events and the interaction between the customer and the processes plus the interaction between the admin and the process and ultimately between the admin and the customer too.

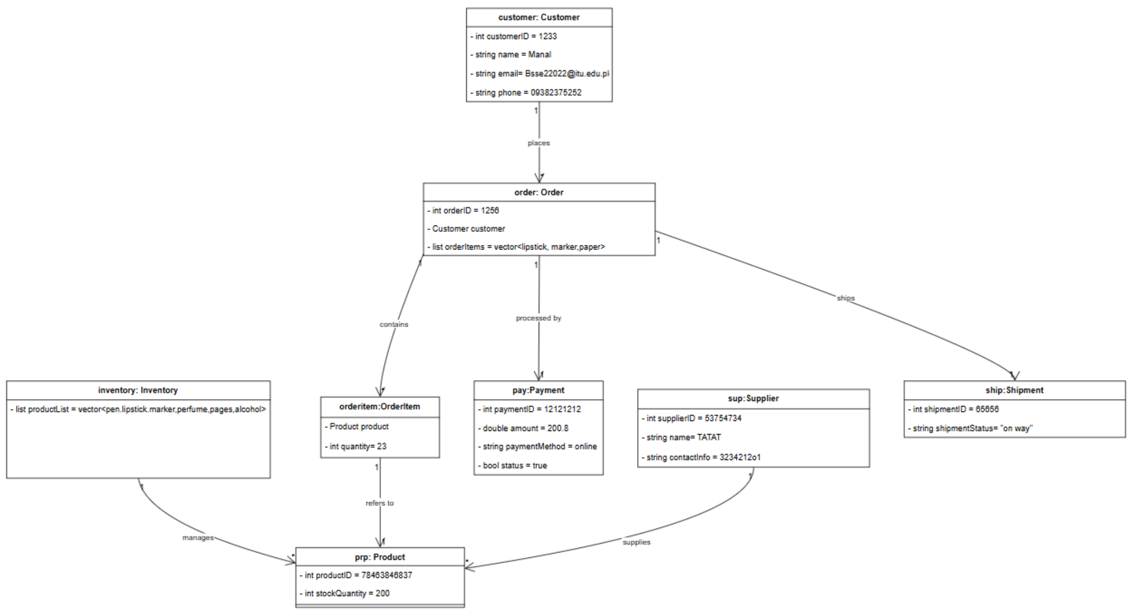
Class Diagram:



Description:

This shows the classes, the blueprint of the system.

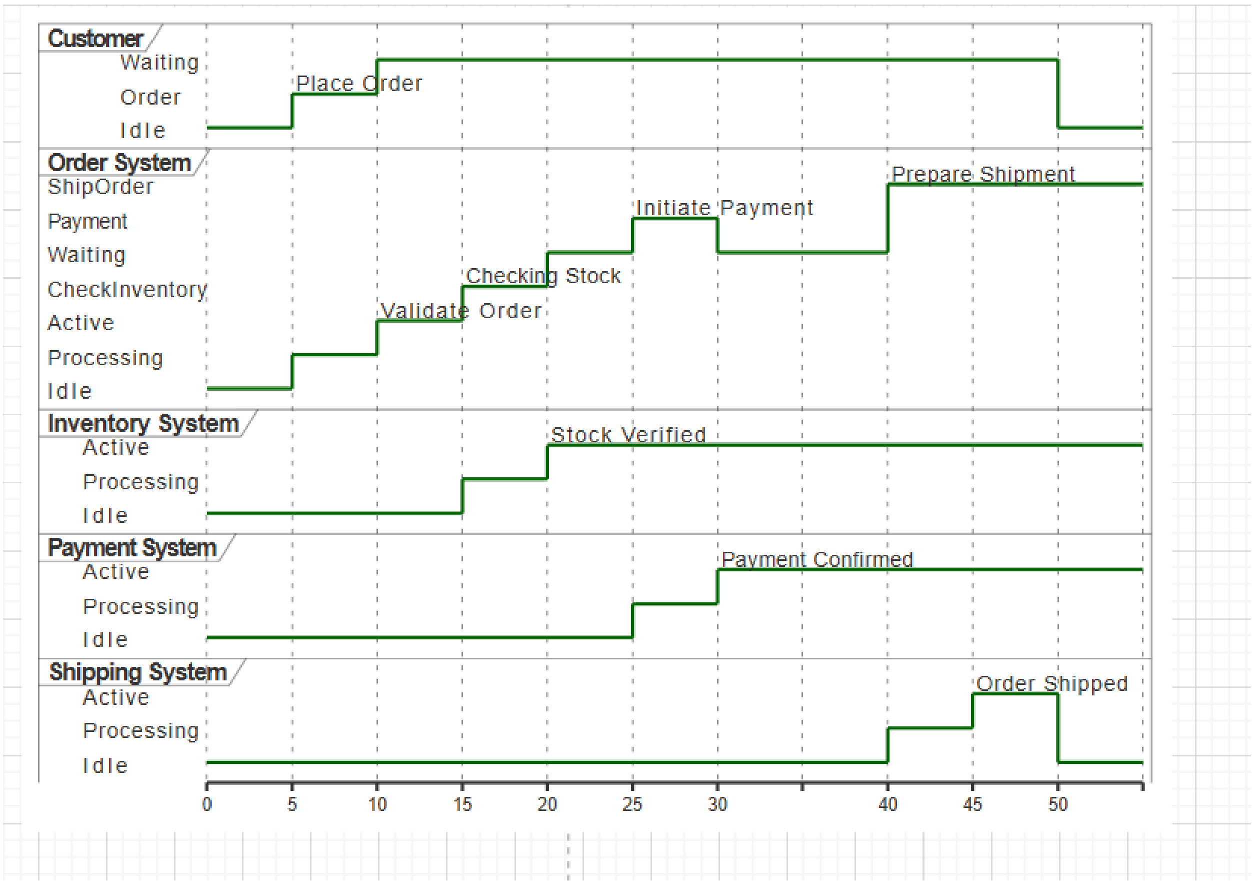
Object Diagram:


customer: Customer
- int customerID = 1233
- string name = Manal
- string email= Bsse22022@itu.edu.pk
- string phone = 09382375252
1
places
order: Order
- int orderID = 1256
- Customer customer
- list orderItems = vector<lipstick, marker,paper>
1
1
1
contains
processed by
ships
inventory: Inventory
- list productList = vector<pen.lipstick.marker,perfume,pages,alcohol>
orderitem:OrderItem
- Product product
- int quantity= 23
pay:Payment
- int paymentID = 12121212
- double amount = 200.8
- string paymentMethod = online
- bool status = true
sup:Supplier
- int supplierID = 53754734
- string name= TATAT
- string contactInfo = 3234212o1
ship:Shipment
- int shipmentID = 65656
- string shipmentStatus= "on way"
1
refers to
manages
supplies
prp: Product
- int productID = 78463846837
- int stockQuantity = 200

Description:

This shows the objects of classes, because in real- life execution the objects interact with one another, that have some real values.

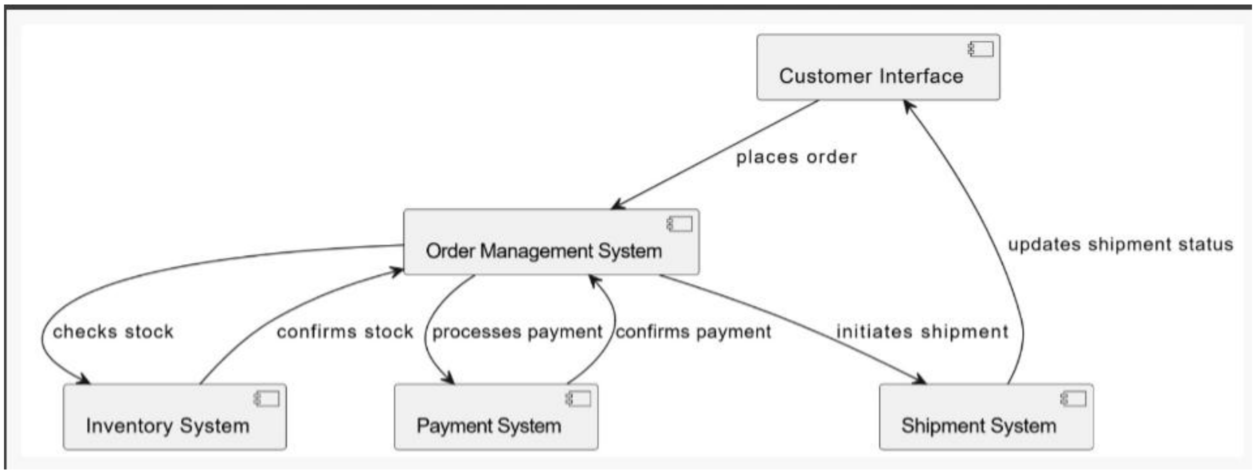
Timing Diagram:



Description:

This shows how the **objects change over time** in a system

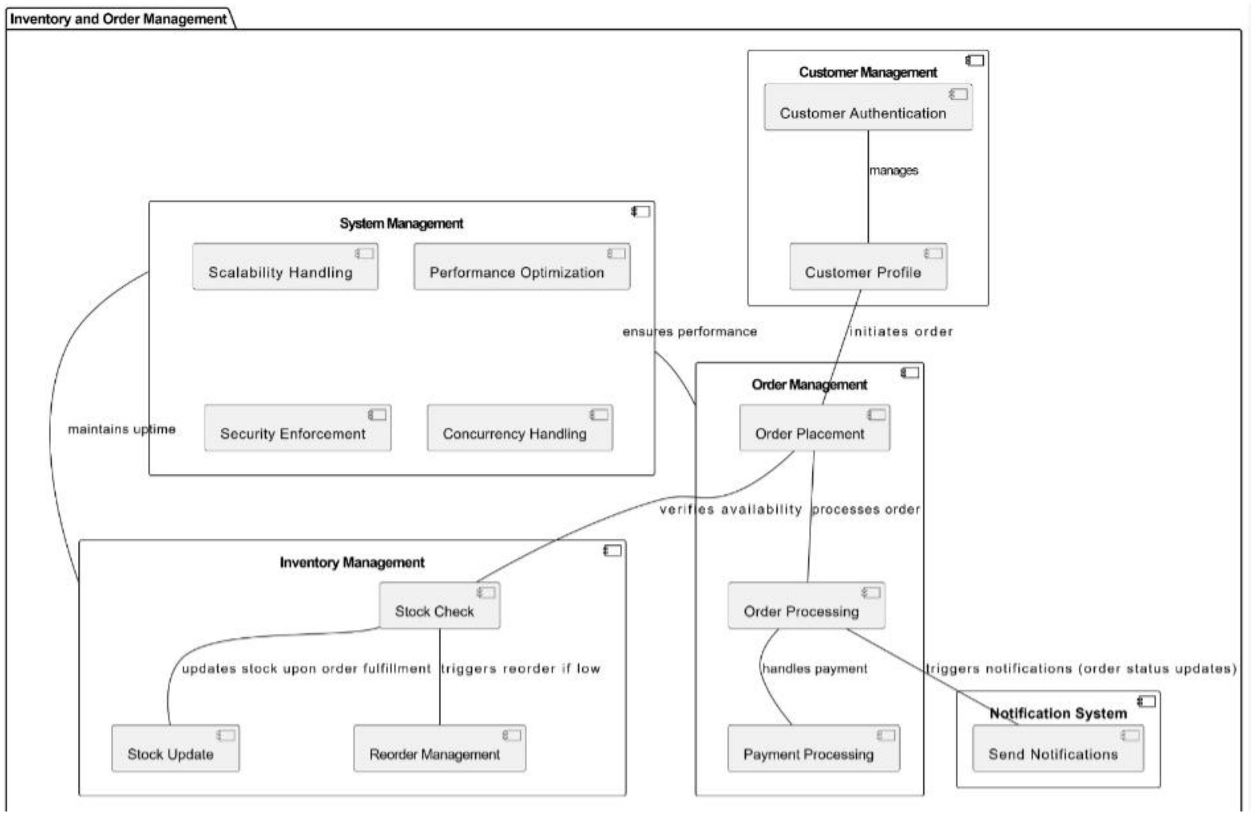
Component diagram:



Description:

A **Component Diagram** in UML shows the **high-level structure of a system** by depicting its components, their relationships, and dependencies.

Composite Diagram:



Description:

This shows the internal structure of a class, component, or system, revealing how its parts interact to achieve functionality.