ANALYSIS AND DESIGN

Part 2 – Completion of SRS Document

(Rana Mohammad Abaan Noon)

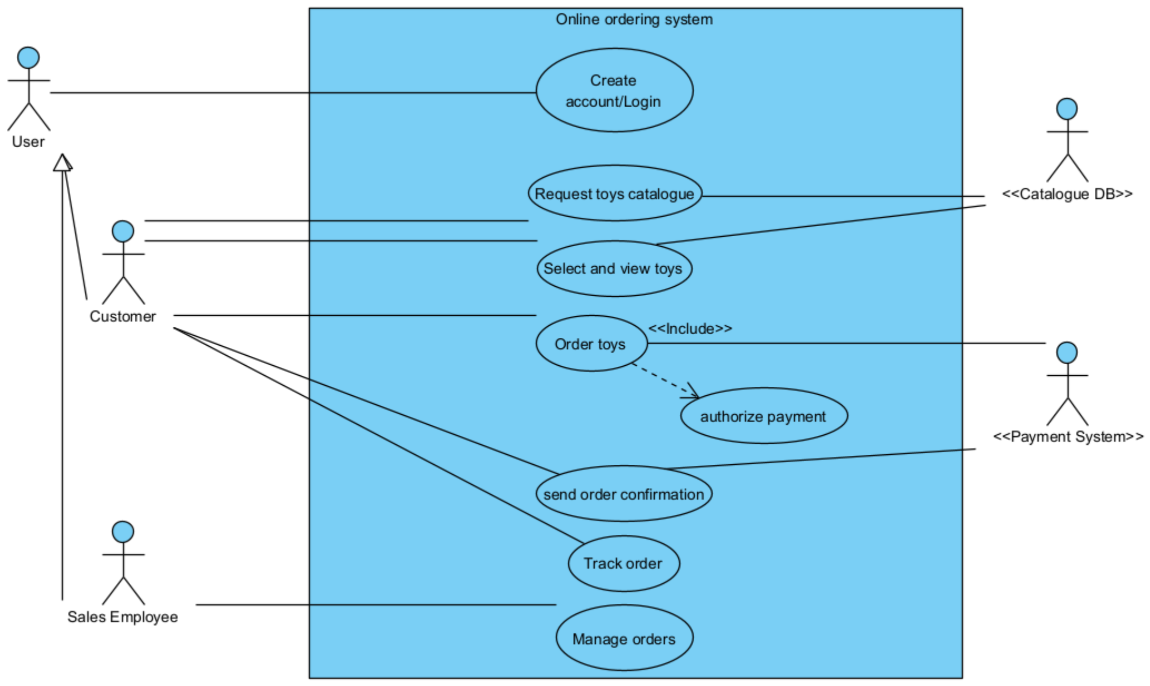
TABLE OF CONTENTS

Task 1. Use Case Diagrams (20 marks)

1.1. Create Use Case Diagram(s)

1.2. Justify the use of "extends" and "includes" relationships in your use case diagram and provide examples from your functional requirements.

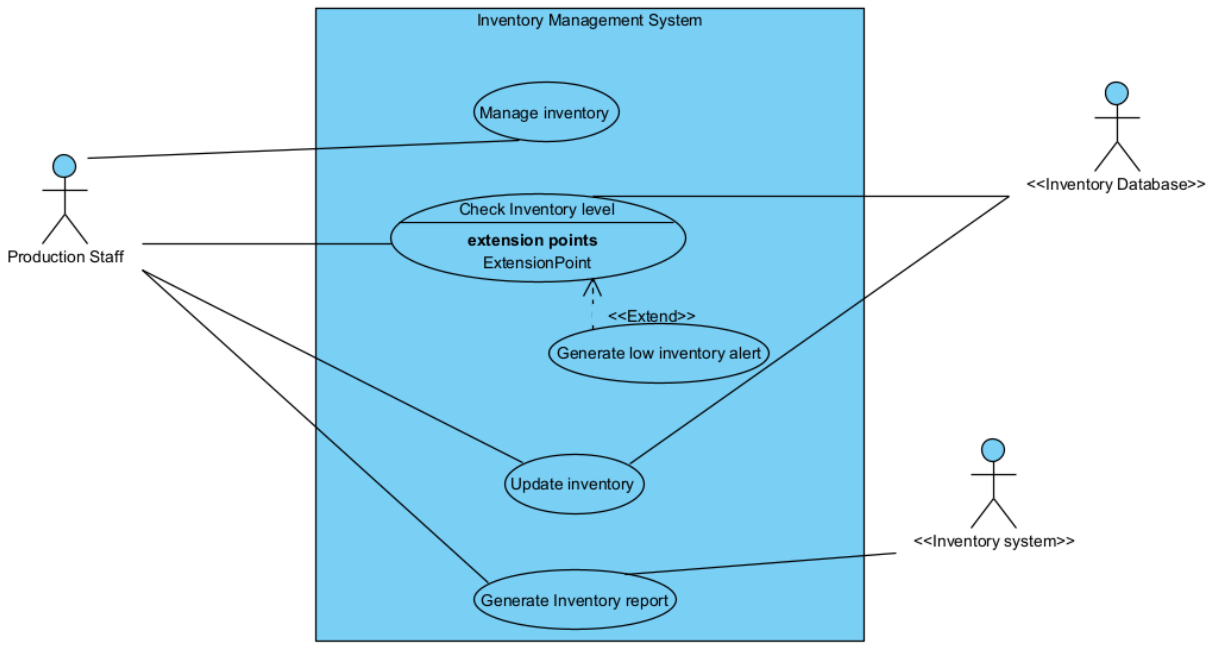
As previously in my first assignment, I have made diagrams for each of the smaller systems I broke the analysis and design into instead of one large system to combat the efficiency issues in the current toys system.



Online Ordering System

- **FR1**: The system shall allow customers to create an account and log in securely.
- **FR2**: The system shall provide a catalogue of products with detailed descriptions, images, and prices.
- **FR3**: The system shall allow customers to place orders online by selecting products from the catalogue.
- **FR4**: The system shall process online payments securely, supporting multiple payment methods (credit card, PayPal, etc.).

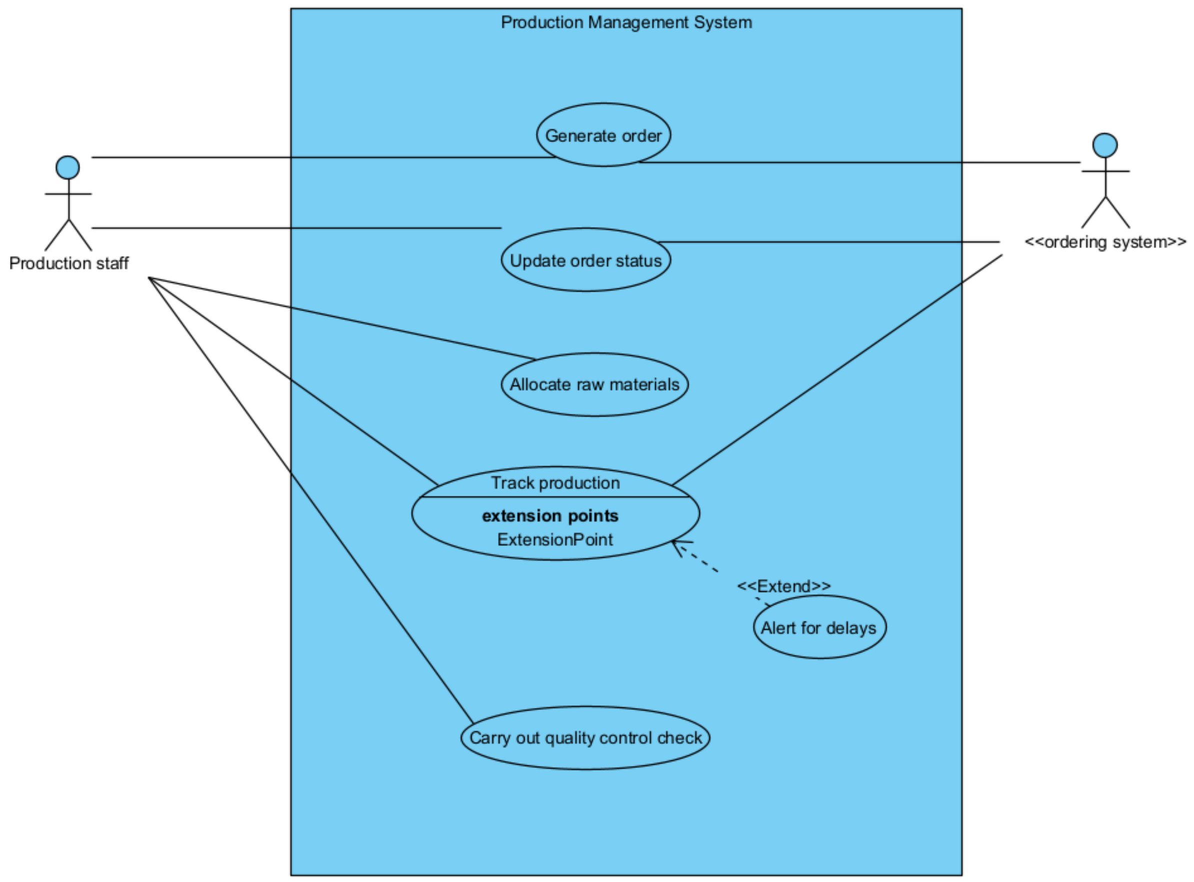
- **FR5**: The system shall generate and send order confirmation emails to customers after a successful order.
- **FR6**: The system shall provide order tracking functionality, allowing customers to view the status of their orders in real-time.
- **FR7**: the system shall allow sales staff to log in and manage the orders and the system platform
- **Include**: Authorize payment – when customers order toys they must also pay on the platform which is authorized by the system



Inventory Management System

- **FR8**: The system shall track inventory levels of raw materials and finished products in real-time.
- **FR9**: The system shall automatically generate alerts and orders when inventory levels fall below predefined thresholds.
- **FR10**: The system shall update inventory levels automatically when products are sold, or raw materials are consumed in production.

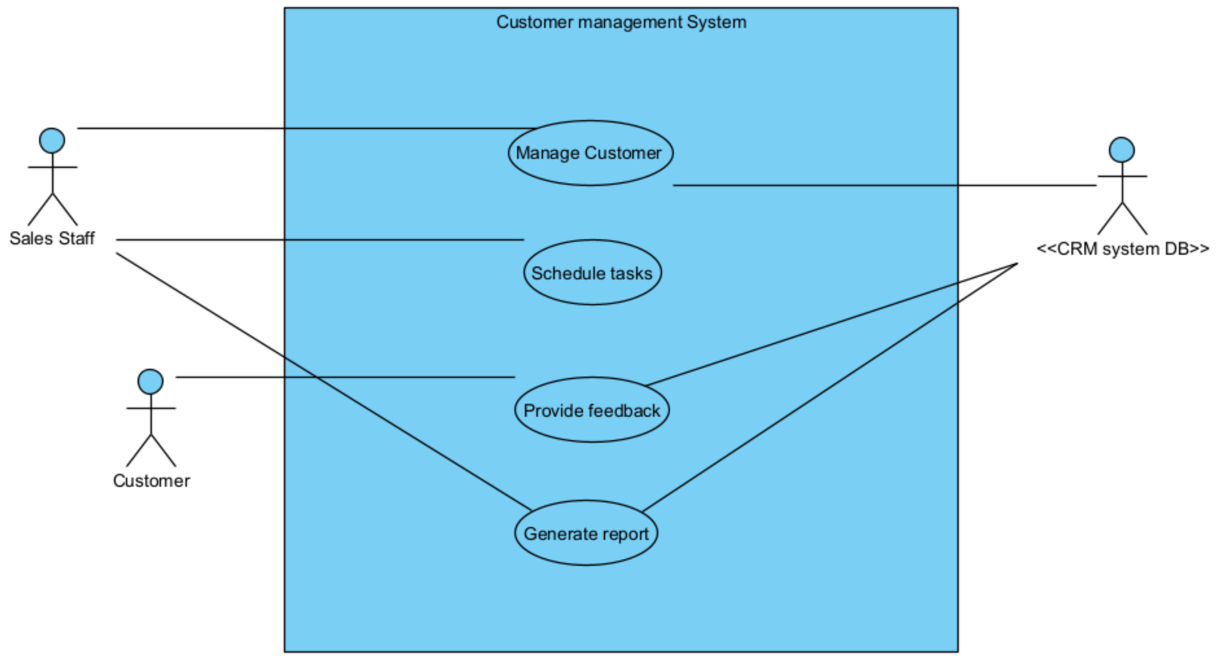
- **FR11**: The system shall allow manual adjustments to inventory levels by authorized personnel.
- **FR12**: The system shall generate reports on inventory turnover, stock levels, and reordering requirements.
- **Extend**: Generate low inventory alert – if the inventory levels are below a certain threshold then the system will send an alert



Production Management System

- **FR13**: The system shall generate work orders automatically when a product is out of stock and needs to be manufactured.
- **FR14**: The system shall allow production staff to update the status of work orders (e.g., in progress, completed).

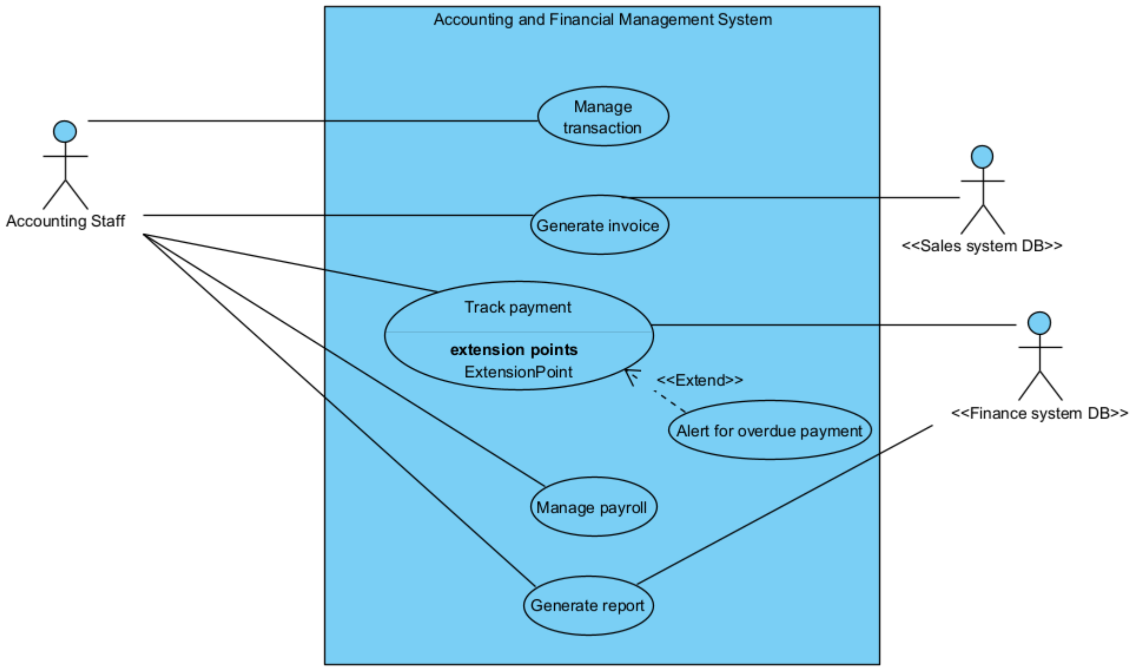
- **FR15**: The system shall integrate with inventory management to allocate raw materials to specific work orders.
- **FR16**: The system shall track production schedules and alert managers if there are delays in meeting deadlines.
- **FR17**: The system shall provide quality control checkpoints where raw materials and finished products can be marked as approved or rejected.
- **Extend**: Alert for delays – if there is any delays in the production an alert message will be sent to the staff to let them know.



Sales and Customer Relationship Management

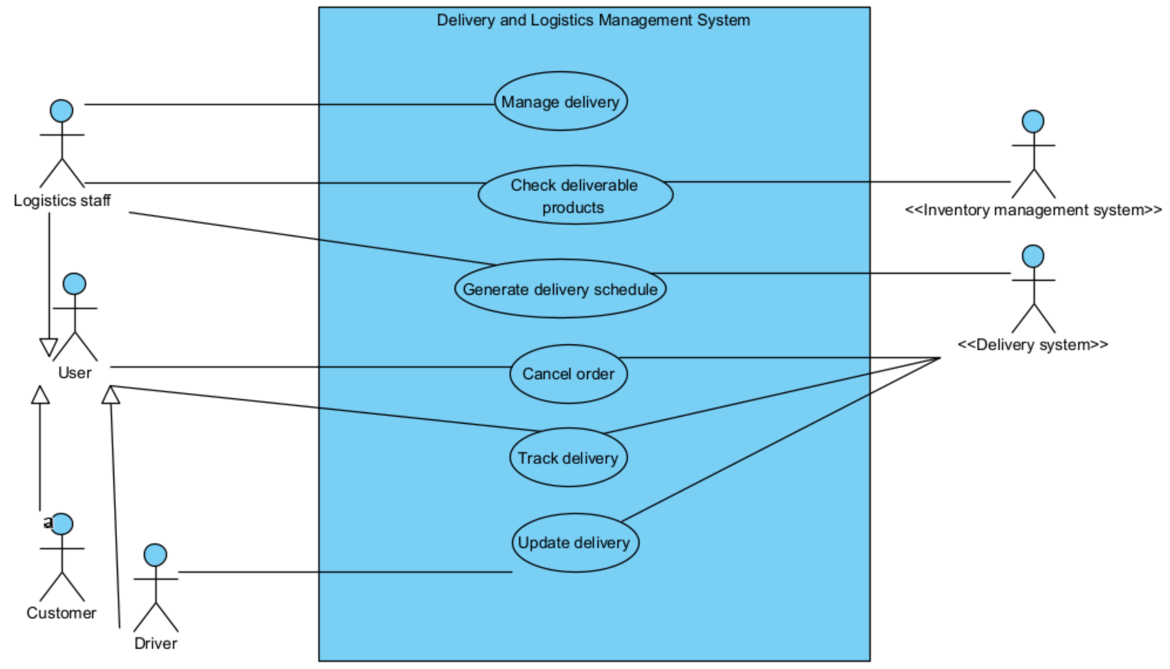
- **FR18**: The system shall store and manage customer information, including contact details, order history, and preferences.
- **FR19**: The system shall provide tools for the sales team to follow up on leads, manage customer communications, and schedule follow-up tasks.
- **FR20**: The system shall generate sales reports, including order volumes, customer demographics, and sales trends.

- **FR21**: The system shall allow customers to submit feedback and inquiries through the online platform, which will be logged and tracked by the sales team.



Accounting and Financial Management System

- **FR22**: The system shall record all sales transactions automatically and update the financial records in real-time.
- **FR23**: The system shall generate invoices and send them to customers electronically after an order is placed.
- **FR24**: The system shall track payments received and generate reminders for overdue payments.
- **FR25**: The system shall manage payroll, including calculating wages, taxes, and deductions, and generating payslips for employees.
- **FR26**: The system shall generate financial reports, including profit and loss statements, balance sheets, and cash flow statements.
- **Extend**: Alert for overdue payments – if a consumer has not paid on time, the system will check and send a message to the staff letting them know which payments are overdue.


Delivery and Logistics Management System
Manage delivery
Check deliverable products
Generate delivery schedule
Cancel order
Track delivery
Update delivery
Logistics staff
User
Customer
Driver
<<Inventory management system>>
<<Delivery system>>

Delivery and Logistics Management System

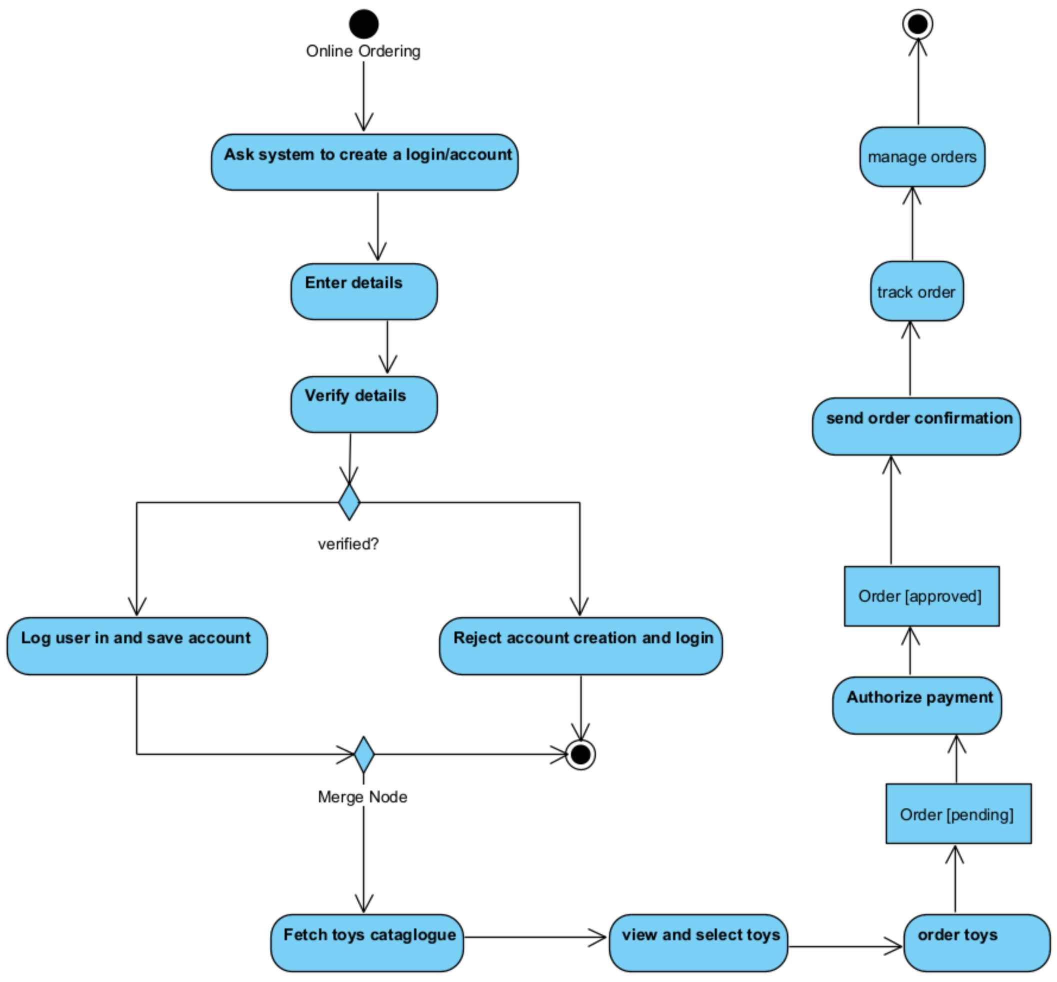
- **FR27**: The system shall generate delivery schedules based on order locations and available delivery resources.
- **FR28**: The system shall track the status of deliveries in real-time, allowing customers and staff to view estimated delivery times.
- **FR29**: The system shall send automated notifications to customers when their order is out for delivery or if there are any delays and or cancelled orders.
- **FR30**: The system shall allow delivery drivers to update the status of deliveries through a mobile interface (e.g., delivered, failed attempt).
- **FR31**: The system shall integrate with inventory management to ensure that only available products are scheduled for delivery.

Task 2. Activity Diagrams (30 marks)

2.1. Create Activity Diagram(s)

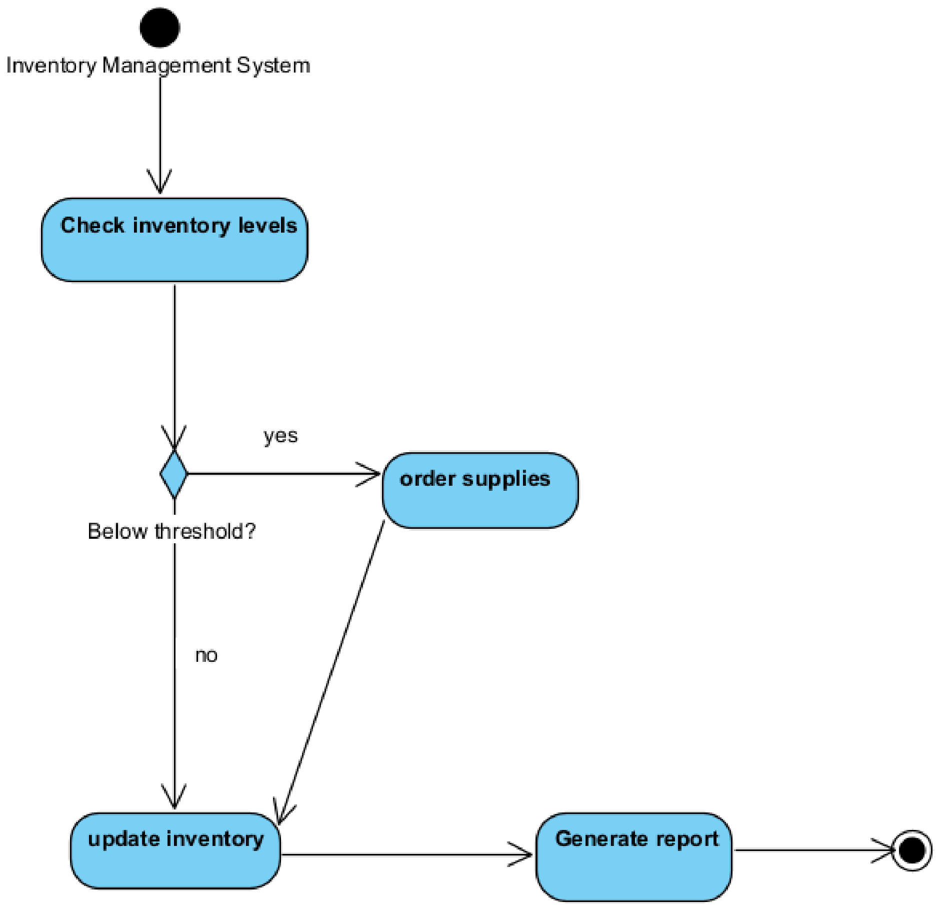
2.2. How did you determine the sequence and dependencies of activities in your diagram based on the functional requirements?

2.3. Justify the use of decision points and forks nodes in your diagram.



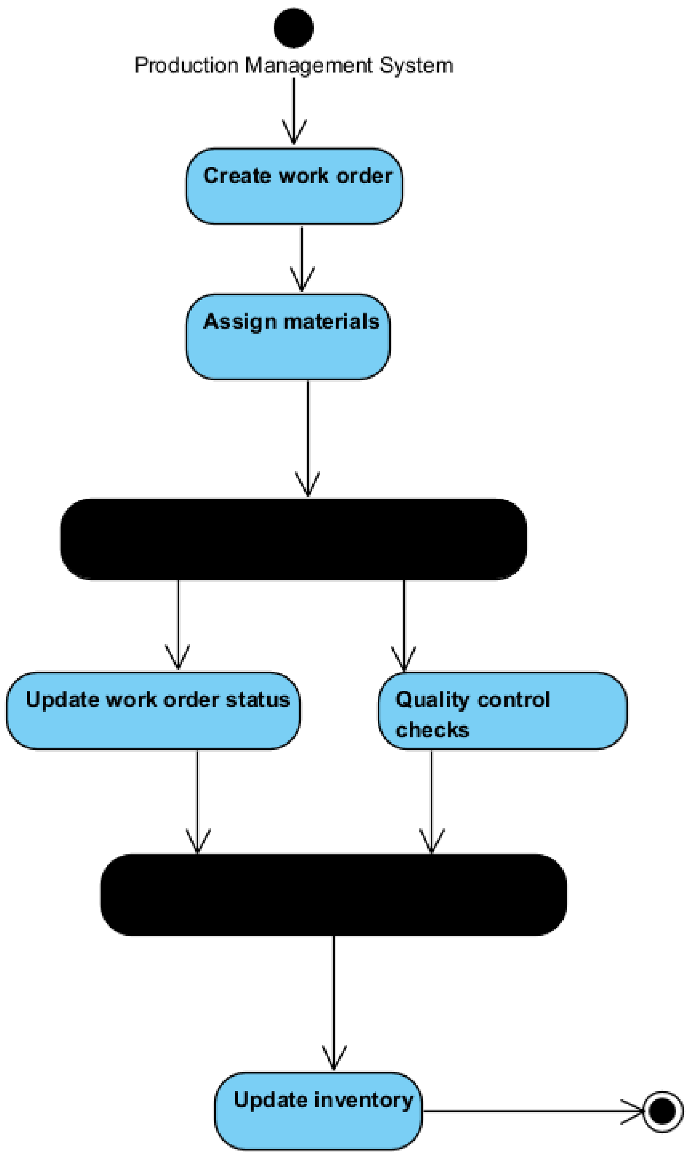
Sequence flows logically from logging in, browsing, and placing an order to processing payment and sending confirmation. Each step is sequentially dependent on the success of the previous step, especially around payment processing.

Decision Point ("verified?"): This node has two decisions where if the user is not verified it will simply reject the account creation/login and end the activity there or if the user is verified, it will let them continue to the system and or logout if they wish to.



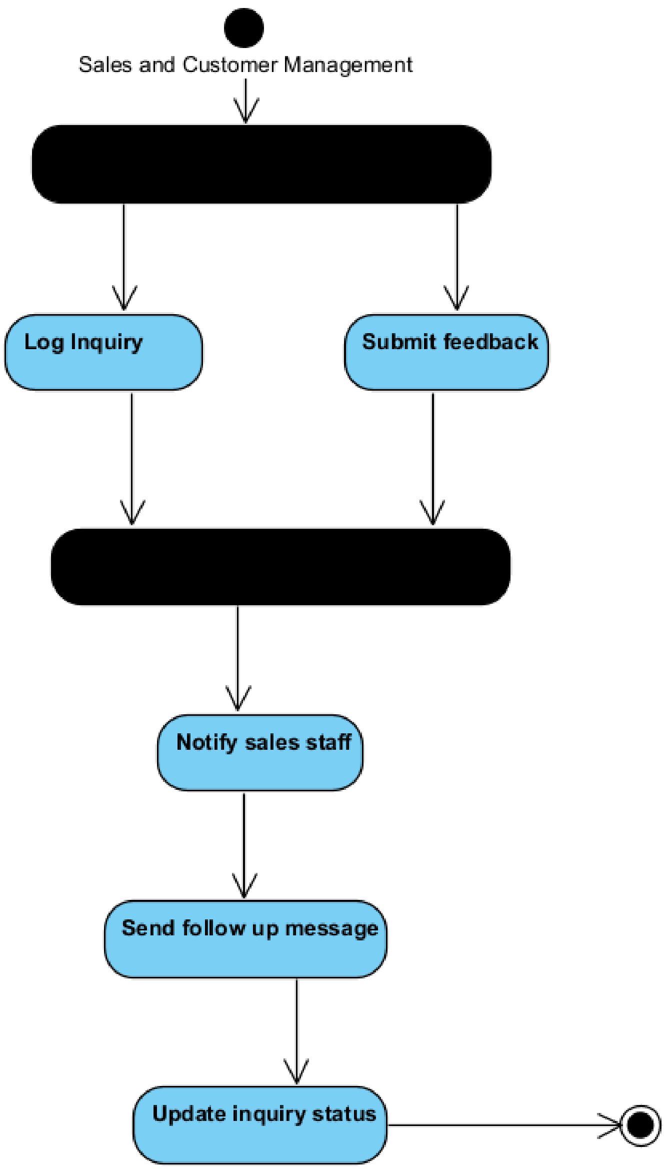
Sequence is determined by the need for periodic checks on inventory levels, which lead to reordering if stocks fall below the threshold.

Decision Point ("Below Threshold?"): Used to decide whether to place an order based on inventory levels. If the inventory level is above the threshold, no action is taken, allowing the process to end.



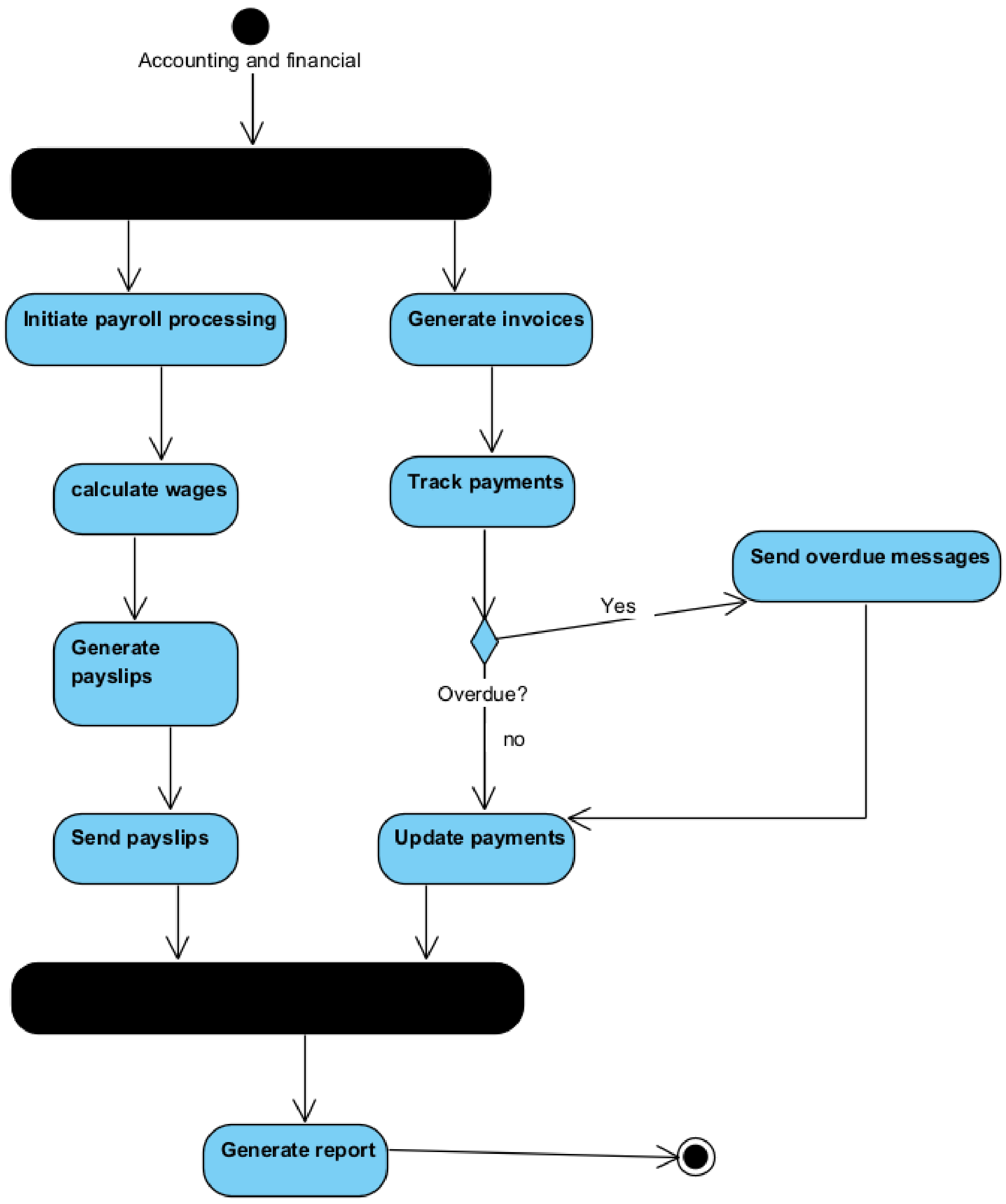
Sequence starts with creating a work order, assigning materials, and enabling staff to update work order status. Quality control checks and inventory updates depend on the completion of the production.

Fork Node: Used to allow for parallel activities, such as updating the status and performing quality control checks. This parallelism allows production status updates and quality control checks to occur simultaneously.



Inquiry submission by the customer initiates a process where the system logs and notifies sales staff, with each step building upon the prior.

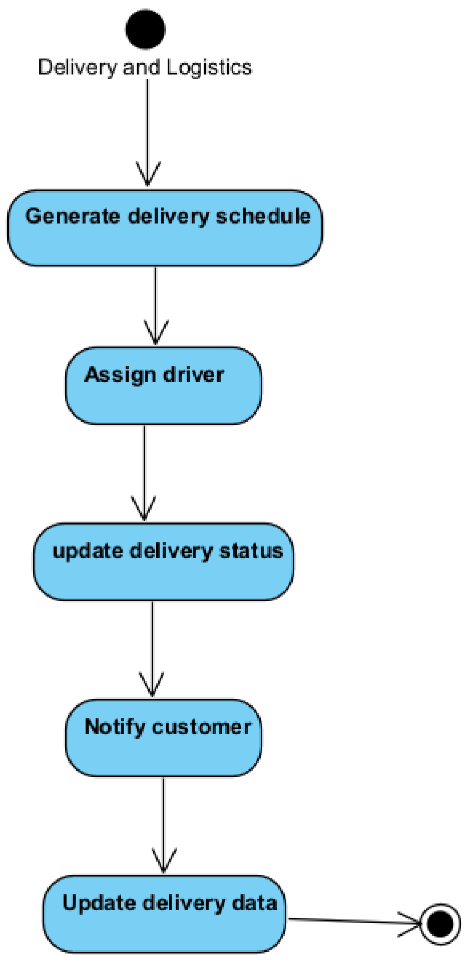
Fork Node: Customers can submit inquiries and feedback which are then notified to the staff.



Each activity in payroll processing flows sequentially, with wages calculation and payslip generation being dependent on each preceding activity's success and simultaneously invoice and payments can be processed as well leading up to the generation of the final report.

Decision Point ("Overdue?"): If payments are overdue, messages are sent out to the consumer and then payments data is updated.

Fork Nodes: The staff can work on both the payroll or the invoice/payments and in the end generate the final report.



Begins with schedule generation and flows through driver updates, customer notifications, and system record updates. Each step depends on successful completion of the prior.

Decision Points: Not required as there's no branching logic based on conditions.

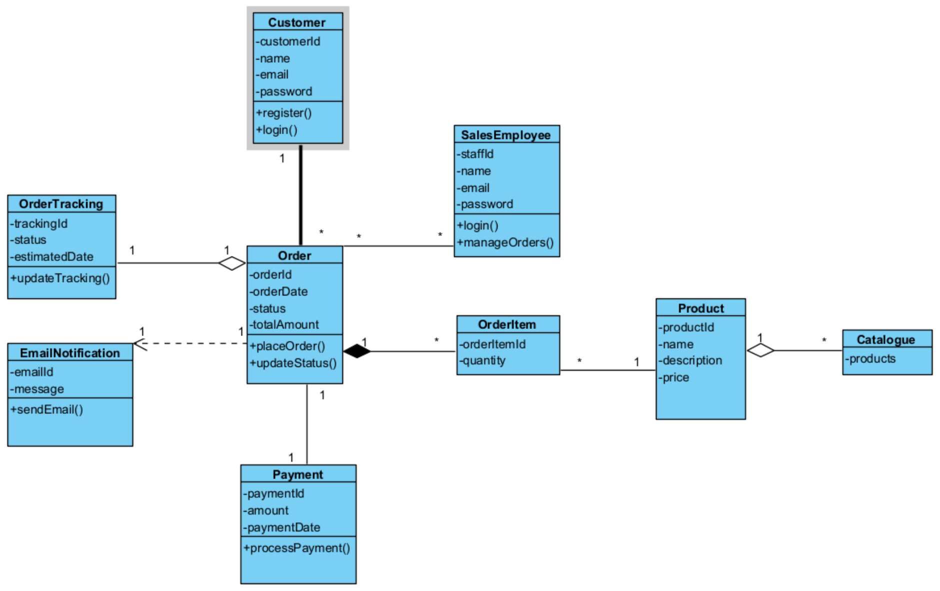
Fork Nodes: Not used, as each step must occur sequentially for consistent tracking.

Task 3. Logical Class Diagrams (30 marks)

3.1. Create Class Diagram(s)

3.2. Describe the steps you took to identify and define the classes and their attributes in your class diagram, refer to your functional requirements.

3.3. Discuss the role of Association, Generalization, and Aggregation relationships in your class diagram.



CMS

Customer – Order

Type: Association

Multiplicity: 1 to *

Description: A customer can place multiple orders, but each order is associated with only one customer.

Order – OrderItem

Type: Composition

Multiplicity: 1 to *

Description: An order is composed of multiple order items, meaning that an OrderItem cannot exist without an Order. If an Order is deleted, its associated OrderItems are also deleted.

OrderItem – Product

Type: Association

Multiplicity: * to 1

Description: Each OrderItem is linked to one Product, but a Product can be linked to multiple OrderItems across different orders.

Order – Payment

Type: Association

Multiplicity: 1 to 1 (optional)

Description: Each order may have one associated payment. This association is optional because the payment might be pending or handled through different means.

Order – OrderTracking

Type: Aggregation

Multiplicity: 1 to 1 (optional)

Description: An Order has a OrderTracking instance associated with it to manage tracking information. However, it is an aggregation because OrderTracking could exist independently of Order.

Order - EmailNotification

Type: Dependency

Multiplicity: 1 to 1 (optional)

Description: An Order depends on EmailNotification for sending confirmation emails upon completion. This dependency indicates that the email functionality might only be triggered during specific order events, such as order placement or status update.

SalesEmployee - Order

Type: Association

Multiplicity: * to *

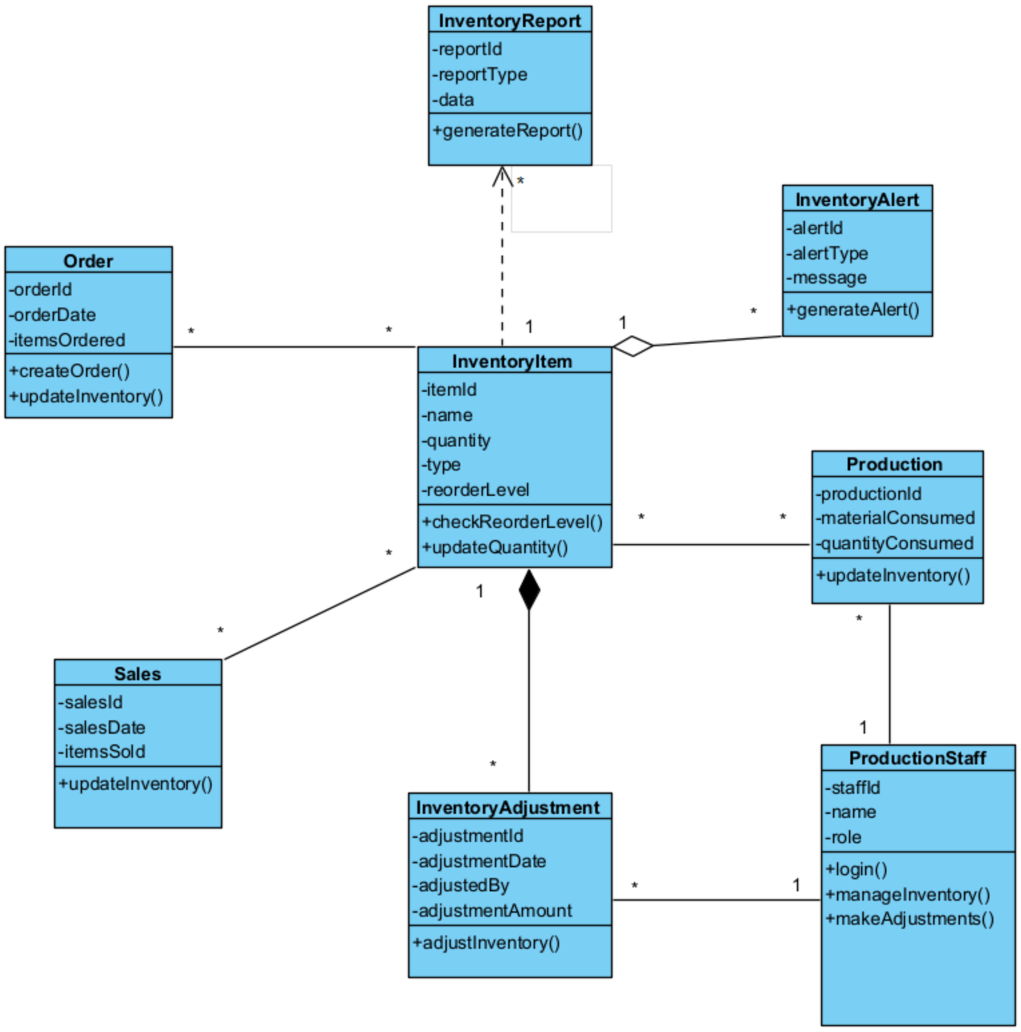
Description: Sales Employee can manage multiple orders, and each order can potentially be handled by multiple sales staff members (for updates, tracking, etc.).

Catalogue - Product

Type: Aggregation

Multiplicity: 1 to *

Description: A Catalogue is an aggregation of Products. Each product exists independently but is also part of a product catalogue that can be displayed or modified.



IMS

InventoryItem - InventoryAlert

Type: Aggregation

Multiplicity: 1 to *

Description: An InventoryItem can have multiple InventoryAlerts if levels are low, though the alert can exist independently of the item.

InventoryItem - Order

Type: Association

Multiplicity: * to *

Description: An InventoryItem may be linked to multiple Orders for reordering, while an Order can include multiple InventoryItems.

InventoryItem - Sales

Type: Association

Multiplicity: * to *

Description: An InventoryItem can be part of multiple sales transactions, and each Sales instance may contain multiple items.

InventoryItem - Production

Type: Association

Multiplicity: * to *

Description: An InventoryItem (raw material) can be part of multiple Production instances, representing its use in different production processes.

InventoryItem - InventoryAdjustment

Type: Composition

Multiplicity: 1 to *

Description: Each InventoryAdjustment directly modifies the InventoryItem's quantity. This is a composition as the adjustment is specific to an item and does not have relevance independently.

InventoryItem - InventoryReport

Type: Dependency

Multiplicity: 1 to *

Description: An InventoryItem depends on the InventoryReport for summarizing its details, but the report is generated on demand and does not permanently associate with the item.

ProductionStaff - InventoryAdjustment

Type: Association

Multiplicity: 1 to *

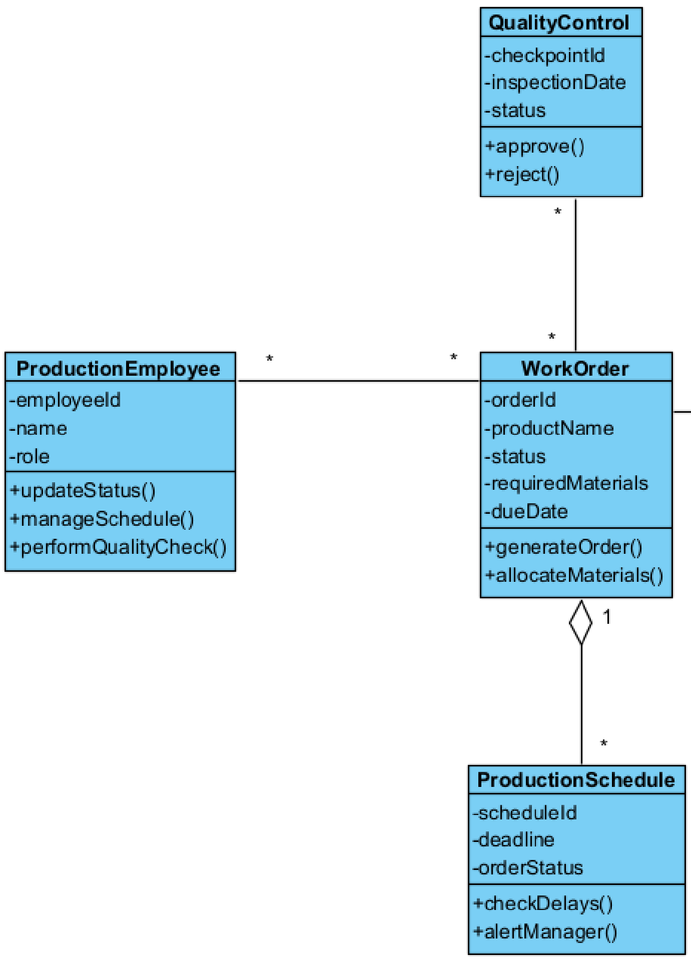
Description: Production staff can make multiple manual adjustments to inventory levels. Each adjustment records the staff member who made it for tracking purposes.

ProductionStaff - Production

Type: Association

Multiplicity: 1 to *

Description: A production staff member can be involved in multiple production tasks, such as tracking the consumption of raw materials, updating inventory based on production needs, and ensuring correct usage of resources.



PMS

WorkOrder - Inventory

Type: Association

Multiplicity: * to 1

Description: Each WorkOrder requires raw materials from Inventory, which allocates the materials as needed.

WorkOrder - ProductionEmployee

Type: Association

Multiplicity: * to *

Description: Production staff members can manage multiple WorkOrders and update their statuses. Multiple staff can be associated with a single work order.

WorkOrder - ProductionSchedule

Type: Aggregation

Multiplicity: 1 to *

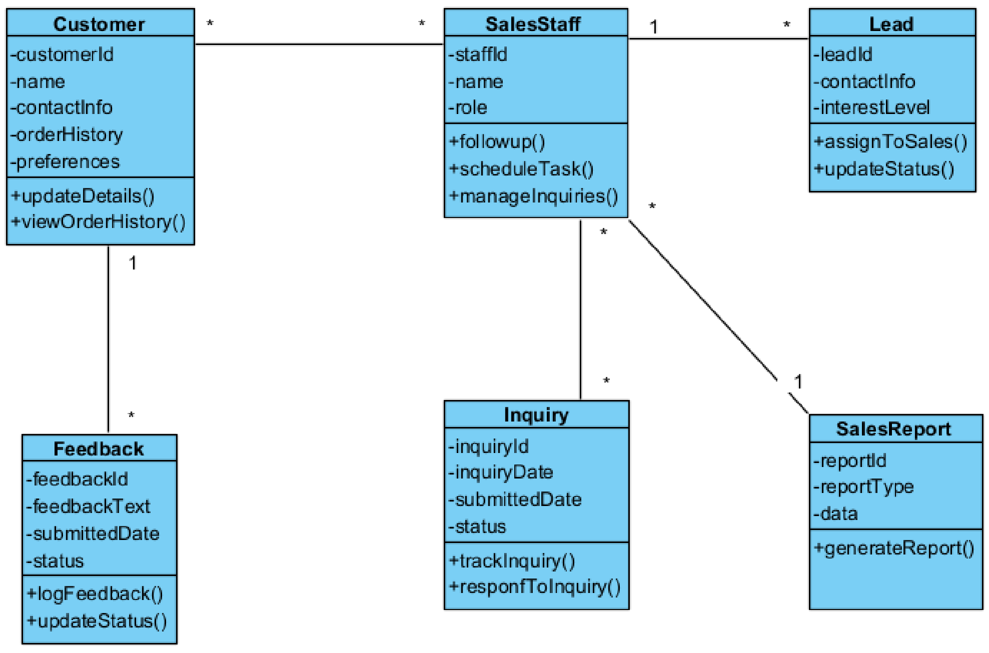
Description: A WorkOrder has an associated ProductionSchedule to track deadlines. The schedule alerts managers if deadlines aren't met.

WorkOrder - QualityControl

Type: Association

Multiplicity: * to *

Description: Each WorkOrder may pass through multiple QualityControl checkpoints. Each checkpoint marks items as approved or rejected.



SCRMS

Customer - SalesStaff

Type: Association

Multiplicity: * to *

Description: Sales staff members interact with multiple customers, managing communications and addressing inquiries. Each customer may be connected with multiple sales staff over time.

SalesStaff - Lead

Type: Association

Multiplicity: 1 to *

Description: Sales staff can manage multiple leads and follow up on potential customers, each representing potential sales opportunities.

Customer - Feedback

Type: Association

Multiplicity: 1 to *

Description: Each customer can submit multiple pieces of feedback. Feedback is logged and managed by sales staff.

SalesStaff - Inquiry

Type: Association

Multiplicity: * to *

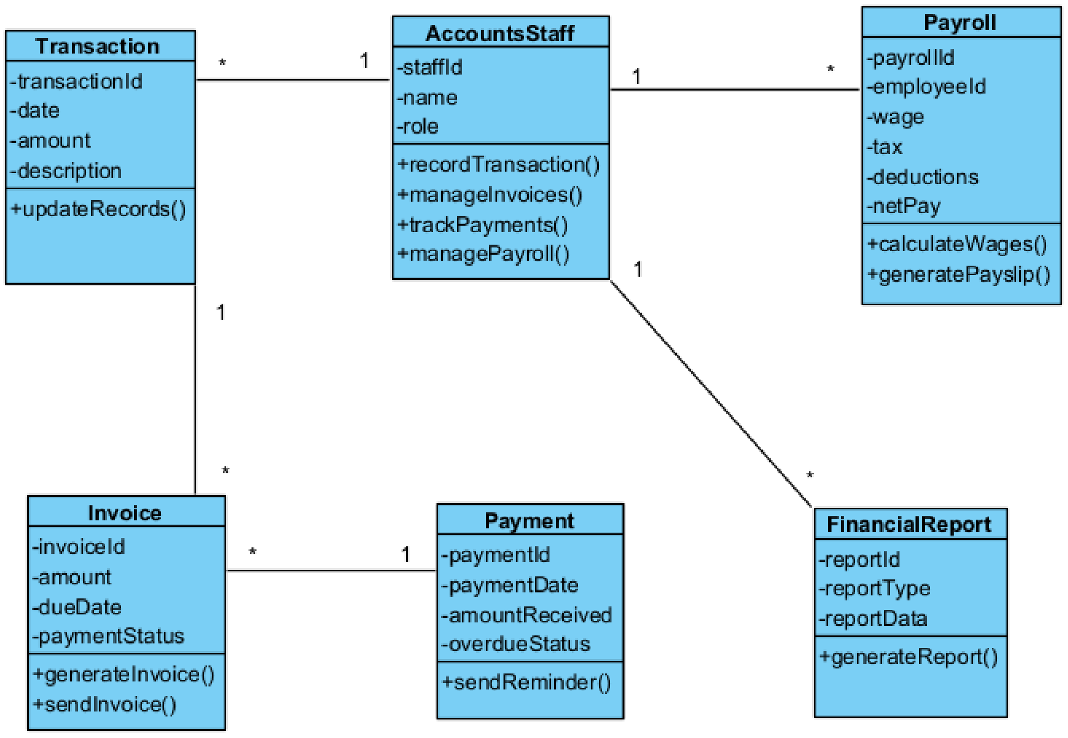
Description: Sales staff members are responsible for responding to multiple customer inquiries, which may cover different products or services.

SalesReport - SalesStaff

Type: Association

Multiplicity: 1 to *

Description: Sales reports are generated based on sales staff activities and customer data, and they provide insights for the sales team’s strategies.



AFMS

Transaction - AccountsStaff

Type: Association

Multiplicity: * to 1

Description: Accounts staff members oversee and manage recorded sales transactions in the system.

Transaction - Invoice

Type: Association

Multiplicity: 1 to *

Description: Each transaction generates one or more invoices for customer orders, which include payment details and due dates.

Invoice - Payment

Type: Association

Multiplicity: * to 1

Description: Payments correspond to invoices, and each payment is tracked to ensure timely processing. If overdue, reminders are sent.

Payroll - AccountsStaff

Type: Association

Multiplicity: 1 to *

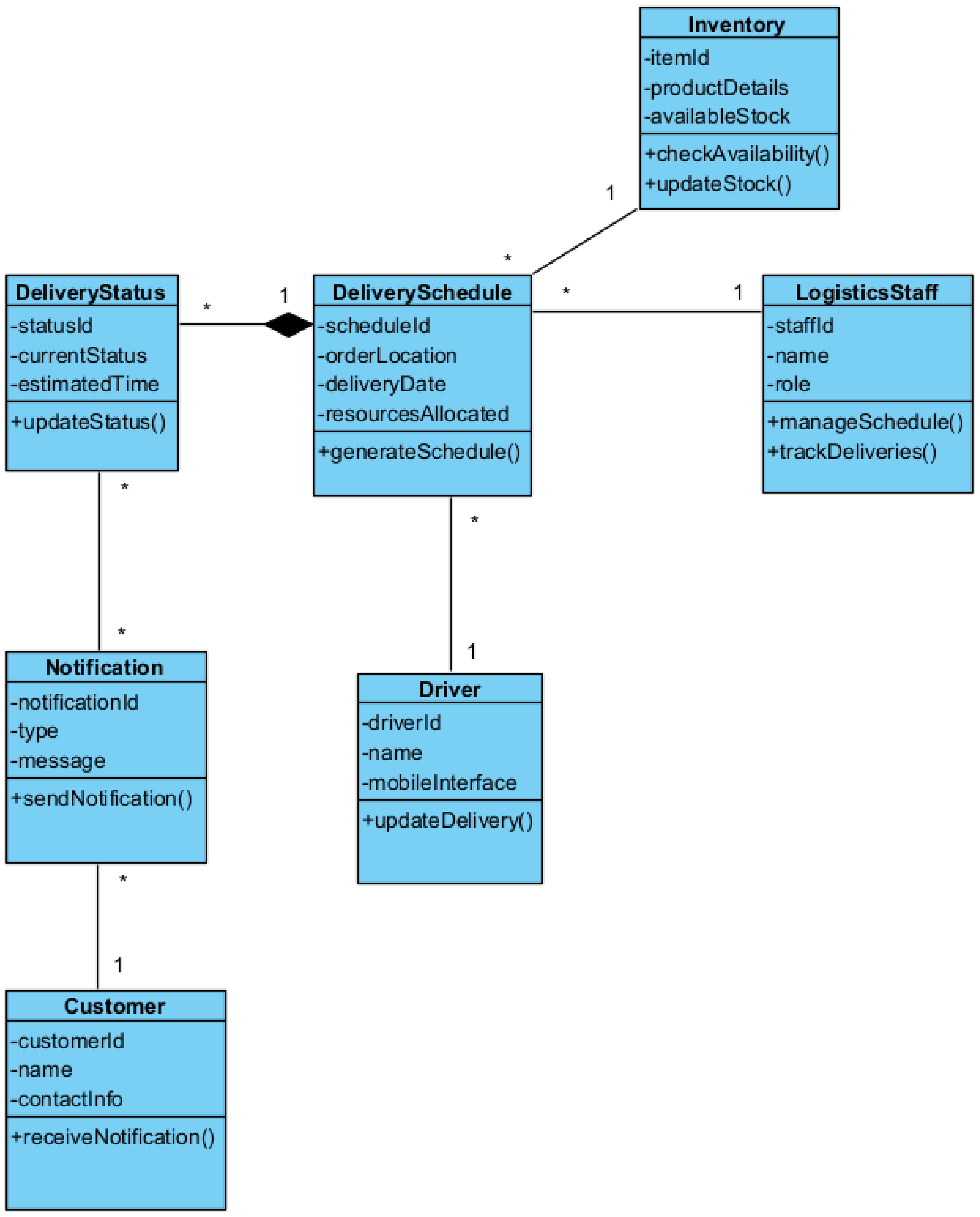
Description: Accounts staff members manage payroll tasks, calculating wages, taxes, and generating payslips.

FinancialReport - AccountsStaff

Type: Association

Multiplicity: * to 1

Description: Financial reports are generated by accounts staff to provide insights into the organization’s financial health.



DLS

DeliverySchedule - LogisticsStaff

Type: Association

Multiplicity: * to 1

Description: Logistics staff members manage and monitor multiple delivery schedules, adjusting them as necessary.

DeliverySchedule - Driver

Type: Association

Multiplicity: * to 1

Description: Each delivery schedule is assigned to a driver, who is responsible for executing and updating the delivery status through their mobile interface.

DeliveryStatus - DeliverySchedule

Type: Composition

Multiplicity: * to 1

Description: Each delivery has DeliveryStatus changes, which provides real-time tracking updates on the order.

DeliveryStatus - Notification

Type: Association

Multiplicity: * to *

Description: Notifications are sent based on updates in delivery status, such as when an order is out for delivery, delayed, or canceled.

Inventory - DeliverySchedule

Type: Association

Multiplicity: 1 to *

Description: The system integrates with Inventory to verify product availability before scheduling deliveries.

Notification - Customer

Type: Association

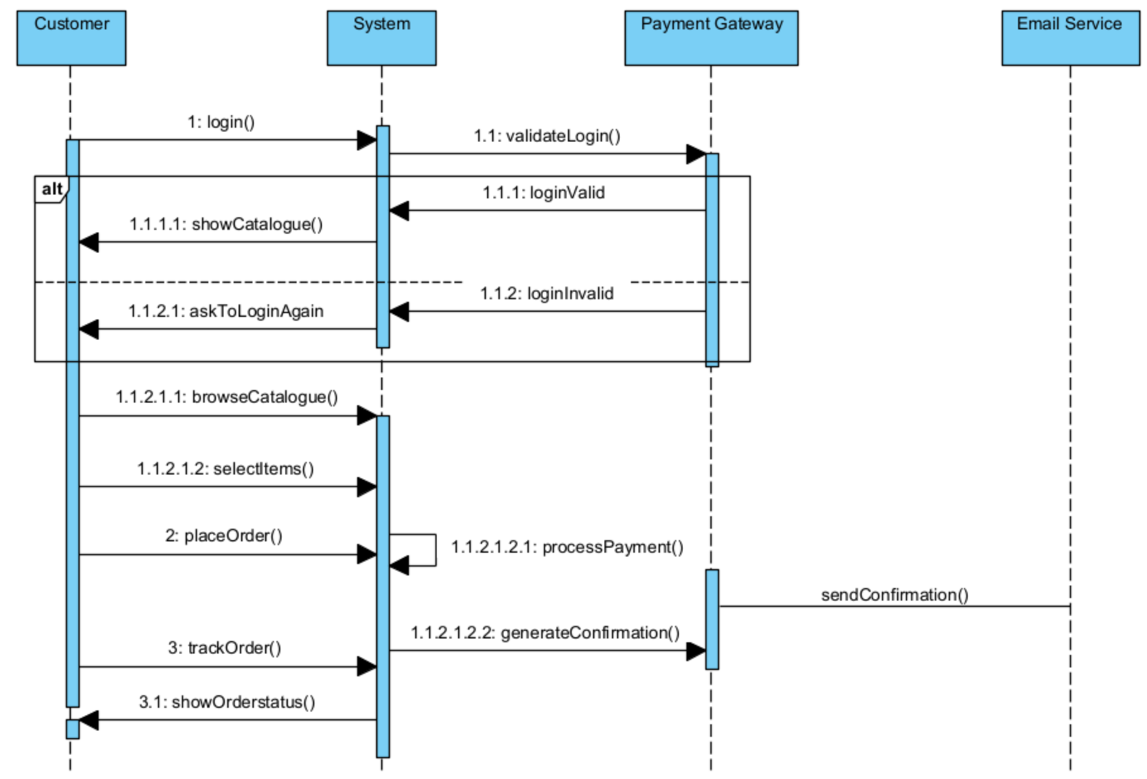
Multiplicity: * to 1

Description: Each customer can receive multiple notifications about their delivery's progress, status updates, or any delays.

Task 4. Sequence Diagrams (20 marks)

4.1. Create Sequence Diagram(s)

4.2. Justify the use of different message types (synchronous, asynchronous, return) and any reflexive messages in your diagram.

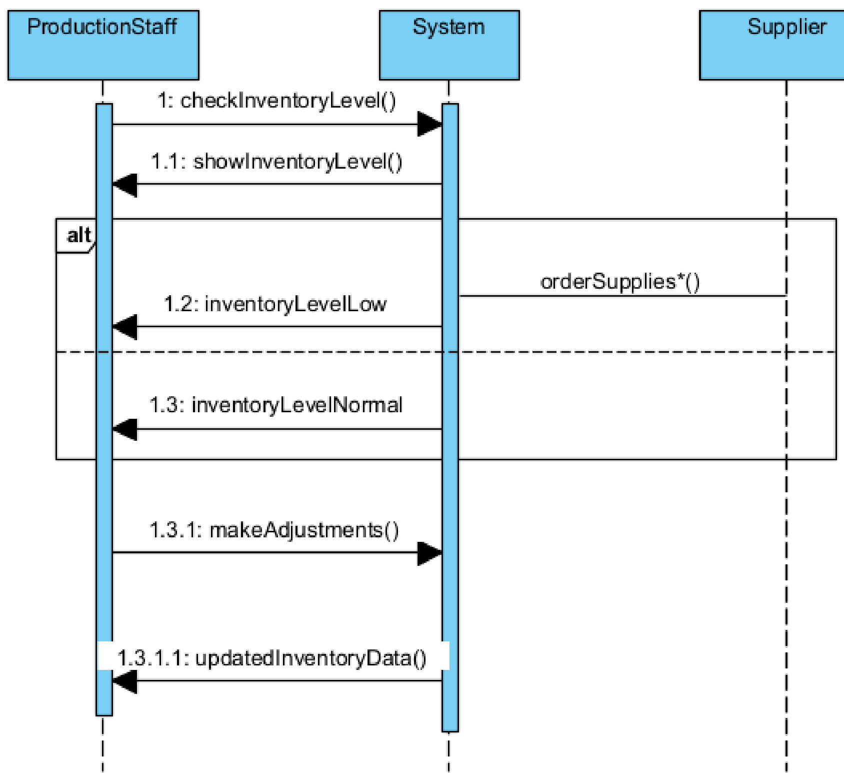


Synchronous Messages (e.g., "login," "processPayment"): Used for requests that must complete before the sequence can proceed. Here, both login and payment processing must succeed before continuing.

Self Message ("processPayment"): The system does a completion of the processing, essential for continuing to the next step and the confirmation is sent on email.

Asynchronous Message ("sendConfirmation"): Sending the order confirmation email can be asynchronous, as it doesn't block the main process flow.

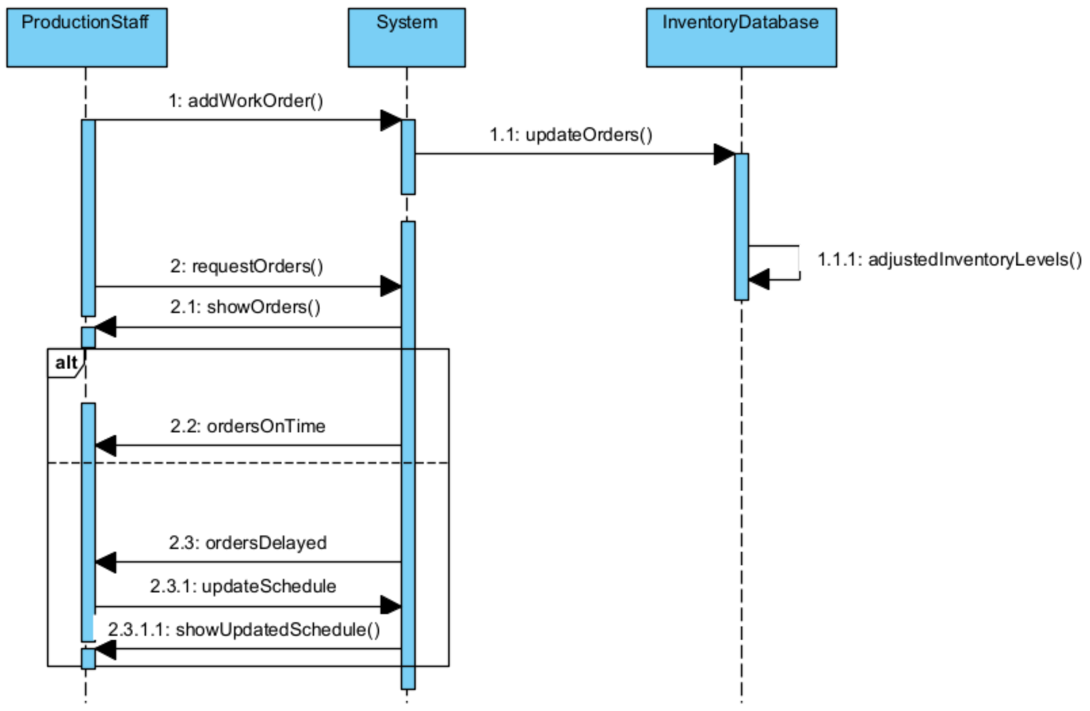
Return Messages (e.g., “showOrderStatus”): returns the status of the order placed after requesting to track it.



Synchronous Messages (e.g., "checkInventoryLevel"): Necessary for the system to obtain the latest data before decision-making.

Asynchronous Message ("orderSupplies"): Ordering supplies can be initiated without blocking the system, allowing it to perform other tasks.

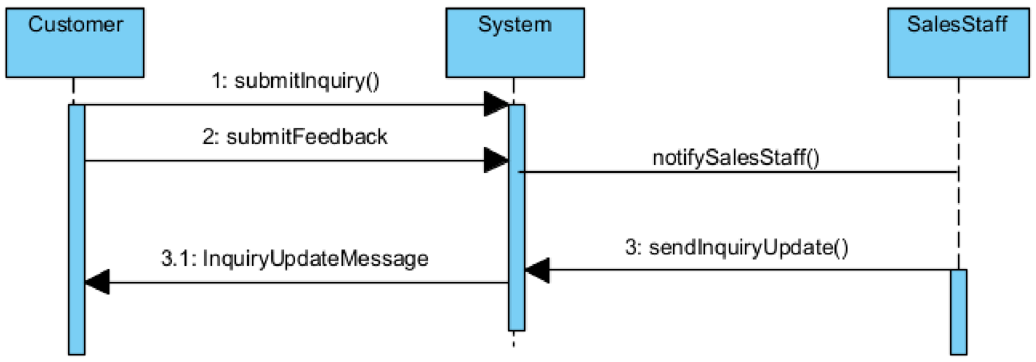
Return Messages (e.g., “showInventoryLevel”): returns the Inventory levels after the checkInventoryLevel request by the staff.



Synchronous Messages (e.g., "addWorkOrder"): Staff makes a work order.

Reflexive Message (“adjustedInventoryLevels”): Inventory Database automatically updates the data.

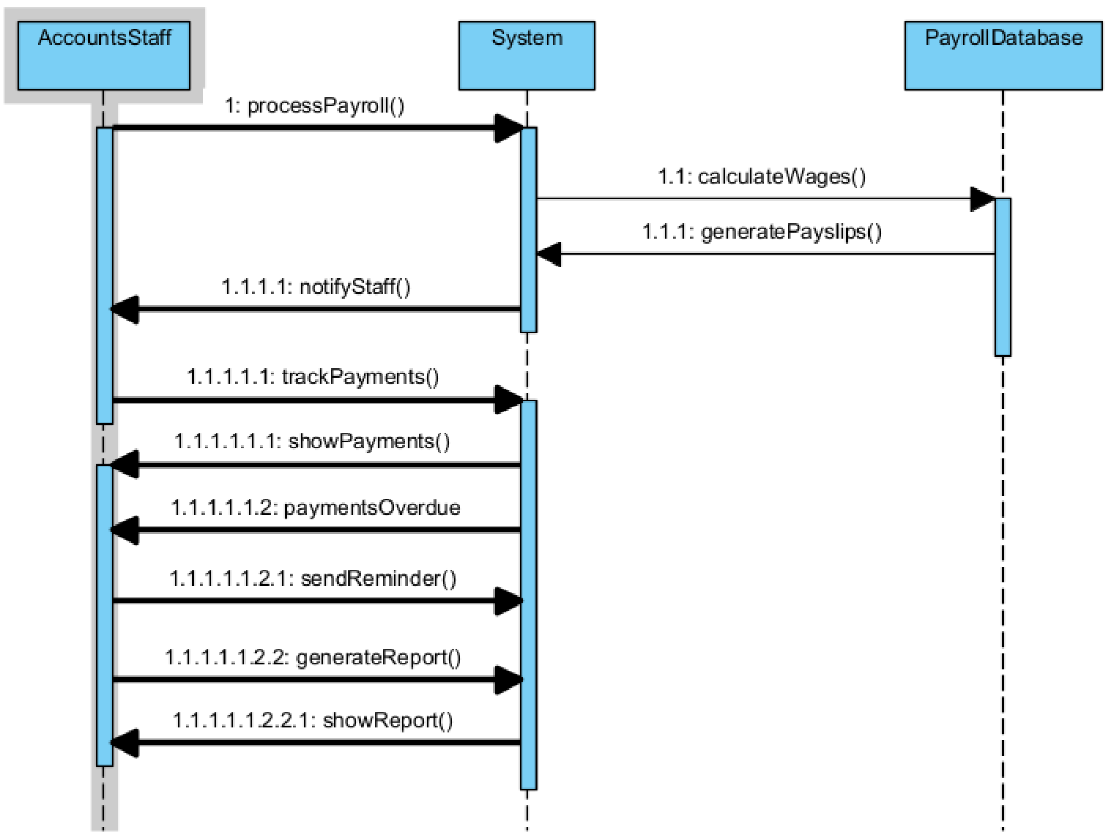
Return Messages (e.g., from the "showOrders"): Shows the current work orders after the system is updated.



Synchronous Messages (e.g., "Submit Inquiry"): Needed to capture and log the inquiry immediately.

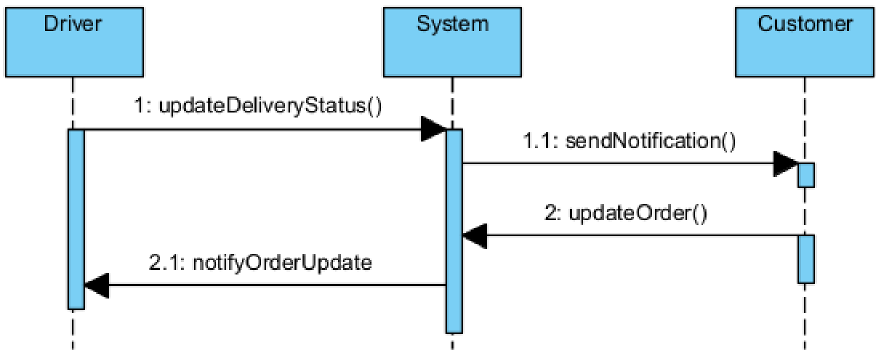
Asynchronous Messages (e.g., "Notify Sales Staff"): Not blocking, allowing the system to manage other tasks concurrently.

Return Message (e.g., “InquiryUpdateMessage”): Lets the customer know about their feedback/inquiries.


AccountsStaff
System
PayrollDatabase
1: processPayroll()
1.1: calculateWages()
1.1.1: generatePayslips()
1.1.1.1: notifyStaff()
1.1.1.1.1: trackPayments()
1.1.1.1.1.1: showPayments()
1.1.1.1.1.2: paymentsOverdue
1.1.1.1.1.2.1: sendReminder()
1.1.1.1.1.2.2: generateReport()
1.1.1.1.1.2.2.1: showReport()

Synchronous Messages (e.g., "processPayroll"): Ensures data is up-to-date before generating payslips.

Return Messages ("Generate Payslips" or "showReport"): Confirms payroll completion and report generation requested by the staff.



Synchronous Messages (e.g., "Update Status"): Ensures real-time updates for delivery status.

Return Messages (e.g., "updateOrder"): Customer makes changes according to the order status such as cancelling or location/time changes after seeing the order status notification.