Faculty of Computing

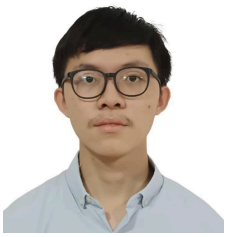
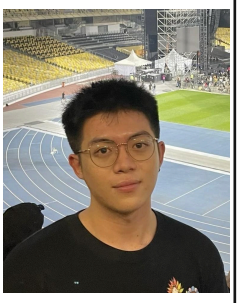
UTM Johor Bahru

Semester I 2024/2025

SEC3744: Enterprise Systems Design and Modeling

System Documentation

Group Member:

				
TIEW CHUAN RONG	LOO JIA CHANG	TIEW CHUAN SHEN	LEE YIK HONG	NIK ZULAIKHAA BINTI ZURAIDI AFANDI
A22EC0112	A22EC0074	A22EC0113	A21BE0376	A22EC0232

Section : 01
Group : 06

Lecturer : Dr. Aryati Binti Bakri
Date : 16th December 2024

Table of Content

Chapter 1: Introduction

1.1 Introduction

The Sales Management System is a newly developed solution created to make the sales process in AK Maju resources faster and easier to use. As AK Maju is a growing company, it has identified some problems and challenges in its current system. Although it is a good system, it does not focus on the specific needs of sales management, which causes issues like delays in checking stock, takes a lot of time in manual sales processing and no real-time updates for stock, delivery and billing status.

The new sales management system using SAP focuses on sales management and solving the problem above. It simplifies the whole sales process, the process begins with stock availability checks to ensure accurate and up-to-date inventory data. Next, it will create sales orders efficiently, followed by the generation and picking of outbound delivery orders. Lastly, it ends up by sending customer billing or receipts, ensuring all in one smooth process.

By replacing the old system, this system not only automates sales tasks but also improves operational efficiency and customer satisfaction. This solution has helped organizations to save time, reduce error and improve productivity. It also helps AK Maju work more efficiently and ensure customer satisfaction. With this system or solution, the company can handle its growing need and provide better service.

1.2 Problem Background

The current system used by AK Maju Resources Management System is a management system that has different parts to run the business. However, it lacks focus on sales management, resulting in several problems and challenges. For example, no real-time updating of stock causes stock checking to consume more time, sales orders and deliveries are done manually and no automation in generating billing and invoices. Handling these tasks separately has led to more mistakes and errors, which can make customers dissatisfied.

As AK Maju keeps growing, it’s clear that they need to have one centralised system to handle the entire sales process automatically. This can help companies to reduce mistakes and errors, speed up the sales management process and improve customer satisfaction.

1.3 Project Aim

The goal of this project is to create a sales management system that improves the sales process in AK Maju, becoming more efficient and productive by developing a sales management system that can automate and streamline the sales process in Ak Maju. This system will fix current problem by integrating important function like checking stocks, manage sales order, handling deliveries and creating bill.

The goal of this project also is to allow the sales management system to integrate smoothly with other sub-system in AK Maju like human resource, financial management, procurement and warehouse management. This is to improve communication between the sub-system, as with all the systems connected, data and information can flow easily between the systems.

1.4 Project Objectives

The primary objectives of this project are as follows:

1. To develop a Sales Management sub-system for Ak Maju that has to manage stock, create sales orders, create outbound delivery orders, pick outbound delivery orders and generate billing reports.
2. Develop a sub-system that can integrate with other sub-systems using SAP S/4HANA and SAP BTP.
3. To design a Sales Management System that is tailored with the needs of Ak Maju.
4. To study on how to create databases in SAP HANA Cloud.
5. Design a database that can be used in SAP HANA Cloud.
6. To build a system using SAP BTP tools that is scalable, ensuring that it can grow with the business and adapt to future needs.
7. To design a modern and intuitive user interface using SAP Fiori

1.5 Project Scope

The scope of this project is to:

1. **Real-time Stock Availability Check:** Real-time update and validation of the stock of a product for decision-making.
2. **Sales Order Creation:** Write down all details of the order and incorporate these into an inventory system.
3. **Auto-generate of quotation and invoice:** After the sales order is created it will automate the generation of quotation by filling the information from sales order and generating invoice when the customer confirms to make a purchase.
4. **Outbound Delivery:** The system automatically creates a delivery order and supports efficient picking processes, hence offering a faster way of shipping.
5. **Billing Integration:** This would ensure the customer's timely, precise, and automated invoices.
6. **System Deployment:** The system should be scalable to support business growth in the future and should handle more sales tasks as the company expands.

1.6 Project Importance

….pThe implementation of this sales management system using SAP S/4HANA and SAP BTP will offer a lot of benefits to AK Maju Resources, including:

1. **Increased operational efficiency:** Automating sales will save time and effort by cutting out the need for manual work.
2. **Improve accuracy:** It helps ensure that stock checks, order processing, and billing are accurate. This can reduce errors and mistakes made by humans.
3. **Enhanced Customer Satisfaction:** Faster service and correct billing will improve customer satisfaction and build stronger relationships with them.
4. **Scalability:** The system is designed to grow with the company and handle more sales in the future.
5. **Centralised management:** All sales activities will be managed in one system, making it easier to keep track and make better decisions.

Chapter 2: Literature Review

2.1 Introduction

This chapter serves as a foundational component of AK Maju Resources' development of its sales management system, including the discussion of major concepts and technologies. This literature review synthesises research, theories, and technologies applied in this work to establish the study's theoretical framework.

This literature review aims to determine whether there is any research or best practice that can be applied to help understand how a system may be developed to meet AK Maju's needs in the most beneficial way possible. This may include examining the subsystems' performance and integration into the sales management structure, as well as the role of modern technologies in improving efficiency. The review also notes the gaps in current knowledge by outlining new approaches that can be incorporated into the project.

This chapter aims to provide knowledge based on academic literature, case studies, and technical reports that explain a strong conceptual groundwork for tactical planning and implementing a sales management system targeting the AK Maju system.

2.2 Enterprise architecture

Enterprise architecture (EA) is a strategic framework that defines the fundamental organisation of a system, including its components, their relationships, and the principles guiding its design and evolution. Key concepts in enterprise architecture include concerns, principles, models, views, and frameworks. While the term "architecture" is broadly understood as the organisation of a system, in the context of enterprise architecture, a precise definition of principles and the mechanisms to turn them into effective regulatory tools is still evolving.

Figure 2.2.1: Togaf Enterprise architecture

One of enterprise architecture's most widely adopted frameworks is The Open Group Architecture Framework (TOGAF). TOGAF provides the Architecture Development Method (ADM), a systematic approach to designing, planning, implementing, governing, analysing, and building an enterprise architecture. This method allows organisations to align IT strategies with business goals, ensuring cohesive and scalable solutions. This TOGAF Enterprise architecture will show how a company runs its operations and technology.

The ADM is central to TOGAF which describes a method for developing the lifecycle of an enterprise architecture. It provides a structured, systematic approach to planning, managing and designing an organization's technology and processes. This TOGAF ADM provides a tested and repeatable process for developing architecture. There are a few phases in ADM cycle which is the preliminary phase, phase A: Architecture Vision, phase B: Business Architecture, phase C: Information Architecture, phase D: Technology Architecture, phase E: Opportunities and solution, phase F: Migration Planning, phase G: Implementation Governance, phase H: Architecture change management and Requirement Management. This ensures that the technology and business processes align with the goals.

A notable case study in enterprise architecture involves an IT strategy for a large manufacturing company in North America. The company, a leading equipment supplier to the oil and gas sector, faced challenges due to increasing business complexity and a lack of a formalized IT strategy. Info-Tech conducted a workshop with key stakeholders to identify areas of opportunity and co-design a three-year IT strategy alongside a tactical implementation plan. The new strategy enabled the company to modernise its business systems, align IT projects with core business needs, and optimise organisational performance. This case study illustrates the transformative power of enterprise architecture in fostering sustainable business growth and IT alignment with strategic goals.

In summary, enterprise architecture provides organisations with the tools and frameworks necessary to achieve operational excellence and align their technological capabilities with business objectives. Organisations can address complexities and drive innovation effectively by leveraging frameworks like TOGAF and applying lessons from real-world case studies.

FIgure2.2.2: Enterprise Architecture

Link clear (EA):
https://drive.google.com/file/d/11uEExW-iEL_84FOMbuSSB02KPc3kOpSR/view?usp=sharing

2.3 Related Sub-System

By using SAP S/4HANA to develop the sales management subsystem for AK Maju, we divided the sales management subsystem from SAP S/4HANA into 5 different modules that related to the sales process. Each different module plays an important role in managing and handling the sales process. The 5 different modules are managed stock (Quotation), create sales order (Quotation and invoice), create outbound delivery (Delivery Order), pick outbound delivery (Delivery Order) and post goods issue (Receipt). These 5 different parts in the system will work together efficiently to ensure the subsystem manages and handles the sales process completely.

Here will explain the 5 different modules of the sales management system. Manage Stock (Quotation) module will focus on managing stock availability and generate quotations when a customer requested. It will check the stock information is accurate and updated in inventory from time to time so that the company can provide reliable and accurate quotes to customers. This can avoid problems like insufficient stock or inaccurate information when selling products.

Create sales order (quotation and invoice) module is to create sales order. Once the customer receives the quotation and agrees with it, the system will automatically generate the sales order and an invoice. The sales order will contain the information of the purchase, such as description, quantities, and pricing. The invoice will provide the customer with a formal billing document. This ensures a smooth transition from quotation to sales order and invoice.

The Create Outbound Delivery (Delivery Order) module will be created when the sales order is confirmed and customer requested. This delivery order is essential for preparing and delivering the product to the customer. This module streamlines the process of delivery preparation process, to ensure that the orders are delivered on time and following the schedule.

Pick Outbound Delivery (Delivery Order) module will focus on the picking process in the warehouse. It will check for the item being delivered is correct and in good condition for

delivery according to the outbound delivery order. This is to ensure that the customer receives the correct item as this system can help minimize the errors in picking delivery.

The Post Goods Issue (Receipts) module is the final stage in the sales process. This module will record the item being delivered to the customer and send it to the customer together with the item. So when the items leave the warehouse, the system will automatically update the inventory to decrease the item in the stock. And it will generate a receipt to confirm the item has been delivered. This has shown transparency in the sales process.

From managing stock to creating orders, creating quotations, creating invoices, preparing deliveries, and issuing receipts, these subsystems work together to ensure that AK Maju can handle sales efficiently, minimise errors, and provide good service to customers. By using SAP S/4HANA, the entire process becomes automated and transparent, allowing the company to focus on growth and customer satisfaction.

2.4 Technology Used

Two technologies have been used to implement the Sales Management system for AK Maju Resources: SAP HANA and SAP Business Technology Platform (SAP BTP). These tools are crucial for seamless operations, real-time insights, and increased system scalability.

SAP Hana

The SAP S/4HANA Cloud Public Edition is a complete, out-of-the-box enterprise management solution and the foundation of the GROW with SAP program to help AK Maju adopt cloud ERP. This is significantly improving the sales management processes with its advanced features:

1. **AI-Based Autocomplete for Sales Order:** Simplify the process of filling out incomplete sales order documents with AI-powered capability that suggests missing fields.
2. **Bundled Forwarding Accounting and Billing:** Streamline invoicing by integrating billing and accounting. As a result, AK Maju can automate invoicing, improve the accuracy of cash flow and revenue recognition, simplify accounts receivable, and meet local requirements.

3. **On-Time Delivery Assurance:** Help ensure on-time delivery by finding and resolving the issues in real time from a detailed list of delayed sales orders and built-in analytics.

SAP Business Technology Platform (SAP BTP)

SAP BTP is an impressive tool for extending and adapting SAP applications while allowing an organisation to implement them within different parts of the business. This advances the working class's ability to create and adjust the business to the necessary changes quickly.

1. **Personalised Experiences**: This feature allows for the customisation of the applications to improve the user experience and make the processes fit the needs of AK Maju.
2. **Trusted Enterprise-Grade Platform**: This gives the ability to support and ensure reliability and compliance for AK Maju operations in a secure, strong, and flexible enterprise-grade platform.
3. **Thriving Partner Ecosystem**: Assist in collaboration and in enhancing new platform features and expanding the platform via third-party components integrations.
4. **Continuous Business Innovation**: This allows users to combine processes and experiences to help people make the right choices and promote growth in all directions.

SAP HANA and SAP BTP allow AK Maju to achieve its objectives confidently in sales management systems. In this manner, the company is equipped with best-in-class technologies that will drive optimisation, enhance the quality of decision-making, and promote growth.

2.5 Summary

This chapter has presented a comprehensive account of the general principles and technologies essential to creating the sales management system for AK Maju Resources. The discussion on enterprise architecture focused on its importance in ensuring IT initiatives are consistent with business goals, stressing such frameworks as TOGAF and their working in organisations. One of the case studies provided evidence of how the ideal implementation of enterprise architecture can improve IT alignment and business processes regarding efficiency.

Also, the technologies used for this project, SAP HANA and SAP BTP, were examined in detail. These tools provide great functionalities, including real-time insights into data, the automation of vital business processes and the ability to easily integrate various business applications critical for enhancing sales operations and scalability.

In addition, by combining theoretical concepts, practical experiences, and technologies, this literature review provides a solid basis for developing and deploying a bespoke sales management system that meets the needs and objectives of any organisation, particularly AK Maju Resources.

Chapter 3: Methodology

3.1 Introduction

The methodology chosen for developing the Sales Management System for AK Maju Resources is the Waterfall Model, a linear and sequential approach to system development. This model ensures that each phase of the project is completed thoroughly before moving to the next, minimising the risks of overlooking critical components of the system.

The Waterfall methodology is particularly suitable for this project due to its structured nature. It allows for comprehensive requirement gathering, well-documented design, and systematic implementation. Given the importance of accuracy, reliability, and seamless integration with AK Maju's existing systems, the Waterfall model provides a clear framework to ensure quality and precision throughout the development lifecycle.

This chapter outlines the selected methodology and describes its phases, including requirements gathering and analysis, system design, implementation, integration and testing, deployment, and maintenance. Each phase is systematically addressed to ensure the successful delivery of the Sales Management System tailored to AK Maju's operational needs.

3.2 The Chosen Methodology

For the development of the Sales Management System for AK Maju Resources, the Waterfall methodology has been selected. This methodology follows a structured, linear approach, where each phase of the project must be completed before proceeding to the next. The Waterfall model ensures a disciplined and well-documented process, which is essential for a system that will be built using SAP technologies like SAP HANA and SAP Business Technology Platform (SAP BTP).

The decision to use the Waterfall methodology, combined with SAP technologies, is justified by the following factors:

1. Clarity in Requirements:

All necessary information and system requirements will be gathered at the start of the project. This eliminates the need for frequent meetings or discussions with stakeholders during later phases, as the SAP-based implementation will strictly follow the predefined specifications.

2. Alignment with SAP's Structured Framework:

SAP systems require a clear and structured design due to their complex architecture and integration capabilities. The Waterfall methodology ensures the project follows a systematic approach, which aligns well with SAP's best practices.

3. Focus on Integration and Scalability:

The SAP ecosystem is designed for seamless integration with existing modules like human resources, procurement, and financial management. The Waterfall model's structured design phase ensures that these integrations are planned and implemented effectively.

4. Emphasis on Quality:

SAP systems demand high-quality implementation to leverage their real-time capabilities, scalability, and robust performance. The Waterfall model emphasizes thorough testing and documentation, ensuring a reliable and efficient system.

5. Predictable and Secure Development Process:

The linear nature of the Waterfall methodology is particularly suitable for SAP implementations, as it minimizes the risks of unexpected changes and ensures strict adherence to project timelines.

3.3 Phases of the Chosen Methodology

The development of the Sales Management System for AK Maju Resources will adhere to the sequential phases of the **Waterfall methodology**. Each phase is completed and thoroughly documented before proceeding to the next, ensuring a structured and efficient development process. The phases are described below:

3.3.1. Requirements Gathering and Analysis

- **Objective**: To collect and define all system requirements before development begins.
- **Activities**:
    - Conduct interviews with stakeholders to understand the business processes and challenges.
    - Analyze the existing manual or semi-automated system to identify pain points.
    - Document functional requirements (e.g., real-time stock updates, automated billing) and non-functional requirements (e.g., scalability, integration with SAP HANA).
- **Deliverables**:
    - System Requirements Specification (SRS) document.
    - A finalized list of all functional and non-functional requirements.

3.3.2. System Design

- **Objective**: To create a detailed blueprint for the system, focusing on architecture, database design, use case, activity diagram, sequence diagram and user interfaces.
- **Activities**:
    - Develop system architecture diagrams detailing data flows and module interactions.
    - Create use case diagrams, sequence diagrams and activity diagrams of each module.
    - Design database schemas (e.g., ER diagrams) for efficient data management.
    - Create user interface (UI) prototypes for features such as stock tracking, sales order processing, and billing.
    - Plan workflows for SAP module integrations using SAP HANA and SAP BTP.
- **Deliverables**:
    - System architecture diagrams.
    - use case, sequence diagram and activity diagram
    - Database schema and ER diagrams.
    - UI/UX prototypes and workflows.

3.3.3. Implementation

- **Objective**: To translate the system design into a fully functional application.
- **Activities**:
    - Develop individual system modules, including stock management, order processing, delivery tracking, and billing.
    - Configure and integrate SAP tools like SAP HANA and SAP BTP for real-time operations and scalability.
    - Conduct unit testing to ensure that each module functions independently.
- **Deliverables**:
    - Fully implemented modules.
    - Unit test reports.

3.3.4. Integration and Testing

- **Objective**: To ensure all components of the system work together seamlessly and meet the specified requirements.
- **Activities**:
    - Integrate all system modules to enable smooth communication and data flow.
    - Perform system testing to validate functionality and identify bugs.
    - Conduct performance testing to assess system scalability and speed under load.
    - Resolve any issues or inconsistencies identified during testing.
- **Deliverables**:
    - A fully integrated system.
    - System and performance testing reports.

3.3.5. Deployment

- **Objective**: To deploy the system in a live environment for real-world use.
- **Activities**:
    - Set up the system in the production environment using SAP infrastructure.
    - Train AK Maju staff on how to use the system effectively.
    - Provide user manuals and documentation.
    - Monitor the system during initial operations to address any critical issues.
- **Deliverables**:
    - Fully operational system in the production environment.
    - Training materials and user manuals.

3.3.6. Maintenance

- **Objective**: To ensure the system continues to meet user needs and adapts to future business requirements.
- **Activities**:

- Monitor system performance and resolve any issues encountered.
- Provide updates and enhancements to improve functionality or adapt to new requirements.
- Ensure compatibility with updates to SAP HANA and SAP BTP platforms.

- **Deliverables**:
  - Maintenance logs.
  - System updates and enhancements.

3.4 Project Planning Schedule (* include in phases of the methodology) - Gantt Chart

Figure 3.4.1: Gantt Chat

Clear Gantt Chart Link:

https://drive.google.com/file/d/12qj_Cnj9IFvZjdOrxzoRFUcxTu96phMw/view?usp=sharing

3.5 Summary

In conclusion, the methodology chosen for this project is the waterfall model. We need to identify and plan all necessary activities and key functionalities before we proceed to the next step or action. The ultimate goal is to ensure seamless connectivity and smooth process during the development of the project. Waterfall methodology prioritizes completing processes before reaching another process, no dynamic demands can be requested to add into the development process.

To summarize, the waterfall methodology covers six phases which are requirements gathering and analysis, system design, implementation, integration and testing, deployment and maintenance. One of the important points of using the waterfall model is adhering to the timelines of the pre-planned project schedule, this may be the biggest challenges in the project.

Just to mention, the technologies used in this project such as SAP HANA and SAP BTP will be an additional plus point to assist in our project development. It gave us extra ideas and sometimes undiscovered or ignored points in doing the project.

Chapter 4: Analysis and Design

4.1 Introduction

Analysis and design of the system is crucial for the success of maintenance and developing a comprehensive enterprise system. First of all, analysis must start from the company organization structure and understand the internal behaviour of the company, the operational workflow. Then, identify and analyze the issues of the current system. The analysis is the comparison between existing systems and proposed systems to make future improvements and identify the pros and cons of each system. Before the system gathering analysis,some interview questions are prepared to gather the functional requirements and non-functional requirements.

Next, system design is formulated based on the previous analysis to figure out the enterprise architecture. The Open Group Architecture Framework(TOGAF) is used as a concept to design, plan, implement and govern an enterprise information technology architecture. System architecture that integrates seamlessly with SAP tools like SAP HANA and SAP BTP is proposed to automate and streamline the operations. User-friendly interfaces and robust database structures are designed to support real-time data management and reporting. Detailed explanation diagrams of the proposed system like use case diagram, sequence diagram and activity diagram are also provided in this chapter. This chapter examines the organization’s current operations, explores potential system improvements, and details the system’s design to meet the outlined requirements.

4.2 Company Organization Structure (AK Maju)

AK Maju Resources is a Malaysian company incorporated on April 9, 2014. The company's main business is in the field of printing and publication. It involved printing and dealing with printing material. For example, The company's principal activity is printing and distribution of various products, including books, magazines, catalogs, brochures, and packaging materials. The company is located at no 39 & 41 Jalan Utama 3/2, Pusat Komersial Sri Utama, 85000 Segamat, Johor. It provides the services related to printing and publication.

Figure 4.2.1: organizational structure

4.3 Current System Analysis (Manual/any improvement) AK Maju

The existing AK Maju Resources Management System operates primarily through manual processes. This approach involves various tasks, such as order management, inventory tracking, and customer data handling, which are performed without the aid of an automated system. The current system presents several challenges:

- **Inefficiency**: Manual entry of data increases the likelihood of errors and delays in processing orders and managing inventory.
- **Time-Consuming**: Staff spend a significant amount of time on repetitive tasks, which could be streamlined through automation.
- **Lack of Real-Time Data**: The absence of real-time updates means that staff may not have the latest information on inventory levels or order statuses, leading to potential stock shortages or overstocking.
- **Limited Reporting Capabilities**: Generating reports is a manual process that can be cumbersome and slow, making it difficult to analyze business performance effectively.

4.5 Comparison between existing system and proposed system

For the current existing system for sales management at AK Maju Resources relies heavily on manual processes for task order management, inventory tracking, delivery order tracking and handling all the business data. Manual entry data will exist many errors, which can affect the orders and inventory process. The process is also time-consuming, as the staff will spend a lot of time on repetitive tasks like paperwork and stock checks. Additionally, the lack of real time tracking updates makes the staff does not get the current updates of the inventory levels or the status of the order, which will lead to shortages and overstocking. Generating a billing report will be very inefficient as it requires a manual process, making it difficult for analysis.

While with the use of the proposed sales management system it aims to solve this problem by automating the sales order management process and introducing new features. With this system, stock management will become more efficient by allowing real-time tracking of inventory levels and auto generated quotations as the customer requests. Creating a sales order will become a faster, easier and more accurate process as it just gets information from the database. Besides that, this system can reduce human error as it does not require manual input. The system will also streamline the creation of delivery orders and simplify the process of picking outbound deliveries. The system also allows users to track the process of delivery in real time. Additionally, it will automate the generation of invoices and receipt once the sales order is created, ensuring accurate and timely billing to the customer.

Overall, the proposed system will bring significant improvements to efficiency, accuracy, and customer experience. Real-time updates will ensure that staff always have access to the latest information about stock levels and orders, while automated processes will free up time for other important tasks. By addressing the limitations of the current system, the new automated solution will help AK Maju Resources operate more smoothly and effectively, providing a better experience for both employees and customers.

4.6 System Requirements Gathering Techniques, Interview, list of the interview questions

In this project, we had an interview for a system requirement gathering. Interviews were conducted in a lab to help us collect important information about the organisation and the system requirements. Before the interview session, we had to prepare a list of interview questions to gather information about the system requirement to ensure that all the information we need is listed. By collecting information through interviews we can know the direction of the clients towards the system.

Here are the question that we had prepare for system requirement gathering before the interview session start:

1) What specific features or functionality are you looking for in this system?
2) What is the current workflow of the company in sales management?
3) What are the formatting requirements for generating a document?
4) What are the challenges you faced when dealing with sales and tracking systems?
5) Do you have any existing software systems or databases that the new system needs to integrate with?
6) Can we have the documentation of the sales report or other documentation related to sales?
7) Do you have any specific additional function or feature that you want to add into the new system?
8) Do you need real-time monitoring for stock and sales management?
9) Can I have some examples of information about your customer and supplier?
10) Do you have problems dealing with the data or information in the old system?

4.7 System Requirements

4.7.1. Functional Requirements

Sales Order Processing

- The system must allow the creation, modification, and cancellation of sales orders.
- The system will automate the cost calculation process.
- The system must generate quotations for clients whenever they request.
- The system can view the history of sales orders.
- The system shall automatically generate invoices upon sales order confirmation.
- Once the system generates the invoice it will automatically update the quantity of the inventory.

Quotation Processing

- The system must allow the creation, modification, and deletion of quotations.
- The system will automatically calculate costs, including taxes and discounts.
- The system must generate professional, client-specific quotations upon request.
- The system can track and view the history of all quotations.
- Accepted quotations will automatically convert into sales orders.
- The system can send quotations directly to clients via email or link.

Billing and Invoicing

- The system must generate accurate billing documents
- The system shall provide a record of all invoices generated for each customer.
- The system can send the invoice or billing documents to the customer.

Delivery Management

- The system must facilitate the creation and management of outbound delivery orders.
- The system shall track the status of deliveries until they are completed.
- The system can monitor the delivery process until it is complete.
- The system can view the history of delivery order.
- If any problem happens during the delivery process it can be notified or reported through the system.

4.7.2. Non Functional Requirements

Performance

- The system shall retrieve sales order details from the database within 3 seconds.
- The system response time for common actions (e.g., creating an order) should not exceed 2 seconds.

Adaptability

- The system shall allow for easy modifications to sales workflows and business rules to accommodate changing business needs.

Maintainability

- The system code must be well-organized and documented to facilitate straightforward maintenance activities.

Reliability

- The system shall ensure high availability, minimizing downtime during normal operating conditions.

Usability

- The user interfaces must be intuitive and user-friendly for the staff of AK Maju, requiring minimal training.

Scalability

- The system shall be designed to handle an increasing number of sales orders without degradation in performance.

4.8. System Design

4.8.1 Enterprise architecture

Figure 4.8.1.1: Enterprise Architecture

Link clear (EA):

https://drive.google.com/file/d/11uEExW-iEL_84FOMbuSSB02KPc3kOpSR/view?usp=sharing

4.8.2 Explain each component in enterprise architecture (one per group) (such as mission, application, technology etc)

1. Vision

The vision emphasizes empowering businesses through seamless sales processes and data-driven insights. This aligns with the goal of enhancing decision-making by providing real-time sales data and automating sales processes, ultimately improving customer satisfaction.

2. Mission

The mission focuses on delivering a robust sales module that streamlines order management and analytics. By automating these processes, the sales management system supports AK Maju's objectives of operational efficiency and growth.

3. Architecture Vision

The architecture vision outlines strategic goals for optimizing operations, emphasizing real-time tracking and automation. This ensures that the sales management system automates tasks like order creation and invoice generation, enhancing service delivery.

4. Architecture Principles

Architecture principles guide the development of the sales management system to ensure it is user-centric and adaptable to business needs. This alignment enhances user experience and ensures that the system effectively supports sales processes.

5. Architecture Requirements

The architecture requirements specify that the system must integrate seamlessly with ERP for order and financial management and with CRM for customer data management. This integration provides a holistic view of sales and customer interactions.

6. Business Architecture

The business architecture defines core processes like inventory management and auto-generation of documents, such as quotations and invoices. By automating these processes, the sales management system reduces manual errors and improves efficiency.

Organization

The organization structure includes roles such as admin and operational staff, benefiting from tools that facilitate sales order management, delivery tracking, and invoice generation.

Function

The sales management system enhances the sales management function by streamlining order processing and ensuring accurate billing through integration with financial management systems.

Process

Key processes like order management and document generation are automated within the sales management system, facilitating quicker order fulfillment and reducing administrative burdens.

7. Application Architecture

The application architecture details the integration between the sales management system, ERP, and CRM applications. This integration ensures a cohesive workflow for order processing and customer management.

8. Data Architecture

Data architecture focuses on structuring and managing sales-related data, including customers, products, and orders. This organization enables quick access to information and accurate reporting, essential for effective sales management.

9. Technology Architecture

The technology architecture provides the necessary infrastructure, such as SAP HANA and SAP BTP, supporting real-time data processing and application integration for efficient operation and scalability.

10. Implementation Governance

Implementation governance involves monitoring the rollout of the sales management system using SAP tools and gathering user feedback, facilitating continuous improvement and ensuring the system meets user needs.

11. Opportunities and Solutions

This component identifies potential enhancements that align with business objectives, encouraging exploration of new features in the sales management system to drive efficiency and better serve customers.

12. Migration Planning

Migration planning outlines the strategy for transitioning to the new system, including data migration and testing, ensuring that historical sales data is accurately transferred to maintain continuity and support informed decision-making.

By connecting these components to the sales management system, AK Maju can ensure that the architecture effectively supports its operational needs and strategic goals.

4.8.3 System Architecture (submodule) (one per group)

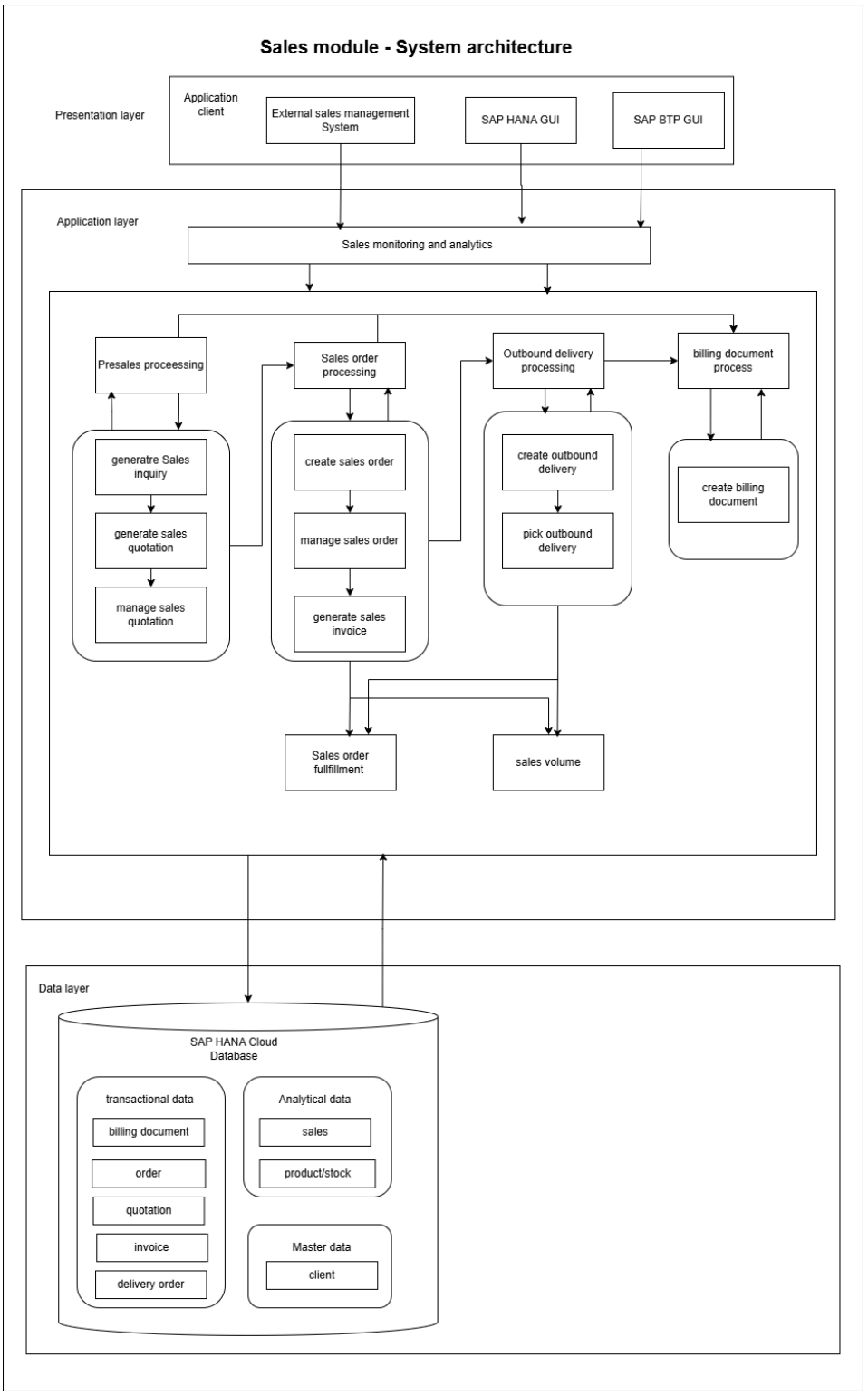


Figure 4.8.3.1: System Architecture (Sales Management System)

4.8.4 Explain each component in system architecture (such as data, presentation etc)

1. **Presentation Layer:**

   The presentation layer is the part where the user will interact with the system. It is the user interface and communication between the user and applications. It is to display information and collect information from the user. For example, the presentation layer in this system architecture consists of an External sales management system, SAP HANA GUI and SAP BTP GUI.

   a. External sales management system - This external sales management system will display teature and functionality like managing stock (quotation), creating sales order (quotation and invoice), creating outbound delivery (delivery order), pick outbound delivery (delivery order) and post gi (receipts).
   b. SAP HANA GUI - It is a user interface provided by SAP to help users with things like running reports and checking sales processes.
   c. SAP BTP GUI - It is a user interface provided by SAP. It is the interface where users interact with the sales system.

2. **Application Layer:**

   Application layer is the heart or the brain of the application. In this part the information collected like sales order information, customer information, product information is processed here. The application layer can also CRUD (Create, Remove, Update and Delete) data in the data layer. This layer will handle the sales process step by step.

   a. Presales processing - Deals with the early stage of the sales process like create sales inquiries and quotations that customers requested.
   b. Sales order processing - It helps to create and manage sales orders, keep track of the order and generate invoices for customers.
   c. Outbound delivery processing - This will create and manage the delivery order and ensure the correct item is delivered to the customer. It also help to keep track of the condition of the item shipped to the customer.

d. Billing document process - This will create and manage the final billing documents for the customer when the item is delivered to the customer. It helps to handle payments.
e. Sales order fulfillment and sales volume - It keeps track of how well the sales are doing and provides insight to help manage sales performance.

3. Data Layer:

Data layer is where all the information and data is stored and managed.

a. SAP HANA Cloud Database - SAP HANA Cloud is a powerful database system. It is hosted in the cloud, where the data is stored online. This allows for real-time processing, fast performance and scalability. This store all the data like transactional data, master data and analytical data in this SAP HANA Cloud Database.

4.8.5 Project Design

Use Case Diagram for enterprise system (one per group)

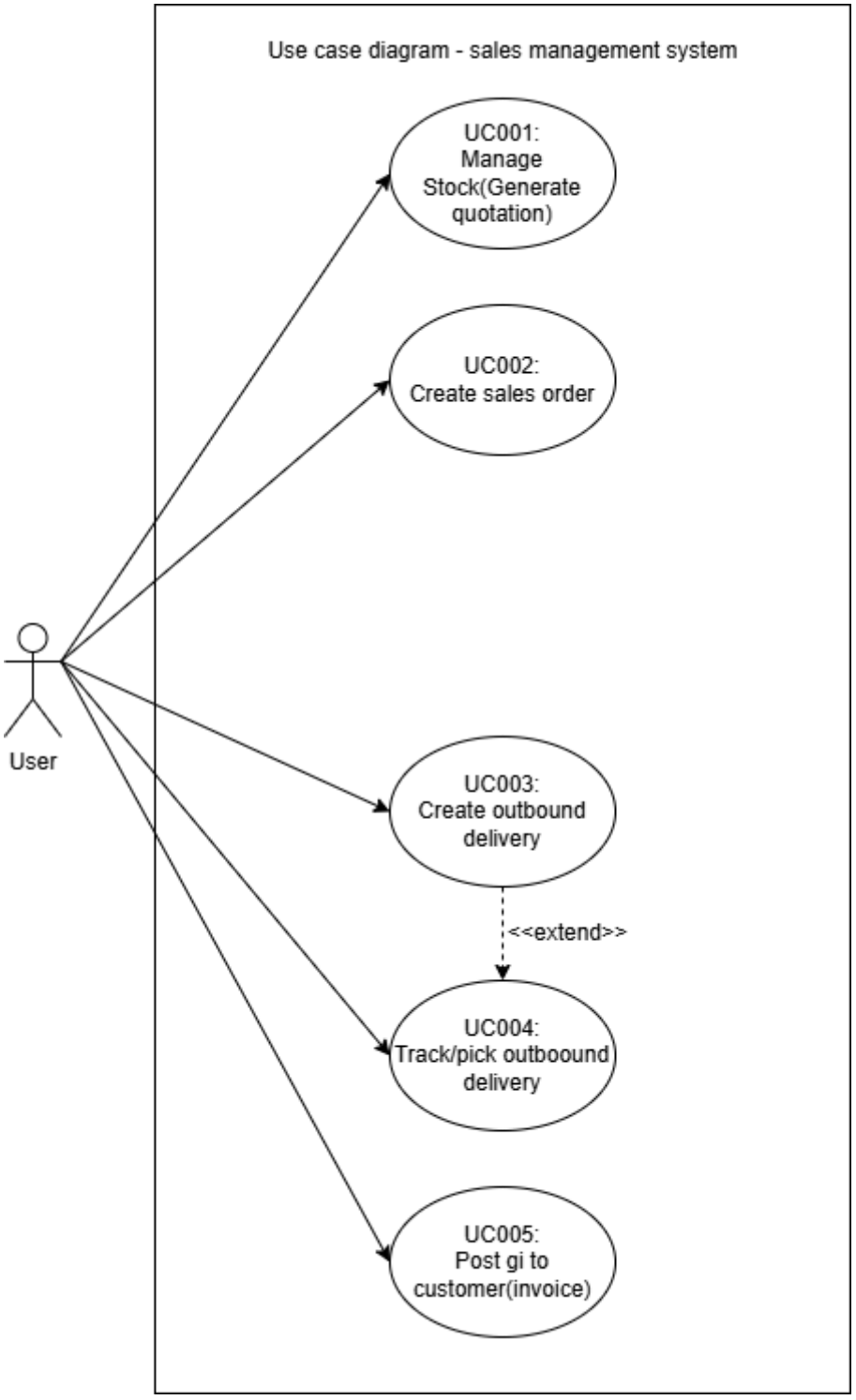


Figure 4.8.5.1: use case diagram - sales management system

Use Case Diagram for sub system (one per group)

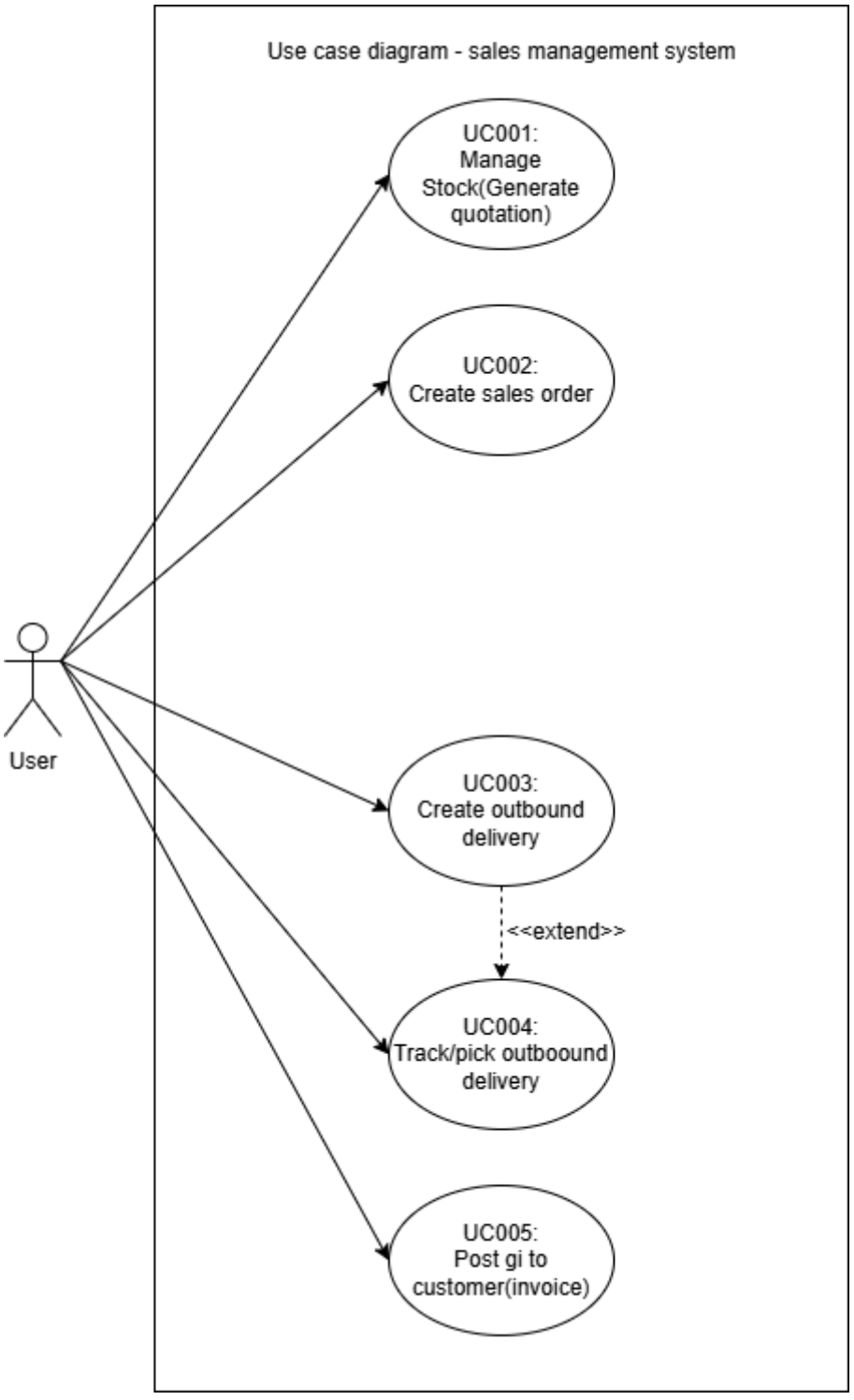


Figure 4.8.5.2: use case diagram - sales management system

Use Case Description (individual)

Use Case Description for Create Quotation (Loo Jia Chang)

Use case: Create Quotation
ID: UC001
Actors: 1. Admin 2. Staff
Preconditions: 1. The user must have valid login credentials to access the Sales Management System. 2. The Sales Management System interface must be operational and connected to the database (SAP HANA).
Flow of events: Scenario: Create a Quotationz 1. **Admin/Staff** opens the Sales Management System through SAP BTP. 2. The system displays the Sales Management interface. 3. The user selects the "Create" option for a new quotation. 4. The system prompts the user to enter quotation details (e.g., customer name, product details, pricing, etc.). 5. The user fills in the necessary information and clicks "Create Order." 6. The system processes the quotation and saves the information into the database. Scenario: Edit a Quotation 1. **Admin/Staff** selects an existing quotation from the system interface. 2. The system retrieves and displays the saved quotation details from the database. 3. The user updates the required information in the quotation. 4. The user clicks "Save." 5. The system processes the changes and updates the database with the new quotation information. Scenario: Delete a Quotation 1. **Admin/Staff** selects a quotation they want to delete from the system interface. 2. The system confirms the action with the user. 3. The user confirms deletion. 4. The system processes the deletion request and removes the quotation from the database.
Postconditions: 1. Quotation data is accurately saved, updated, or removed from the database. 2. Confirmation messages are displayed after each action (create, edit, delete). 3. The quotation is processed without errors, and the system updates its records accordingly.

Alternative flow: 1. -
Exception flow (if any): 1. Database Connection Failure: The system notifies the user of a connection issue and disables further actions until the issue is resolved. 2. Unauthorized Access: If the user does not have permissions, the system denies access to the quotation module.

Use Case Description for Create Sales Order (Tiew Chuan Rong)

Use case: Create Sales Order
ID: UC002
Actors: 3. Admin 4. Staff
Preconditions: 3. Admin/Staff is logged into the system 4. The product is available
Flow of events: 1. Admin/Staff opens the Sales Management System interface. 2. Admin/Staff navigates to the Create Sales Order pages. 3. Click “Create Sales Order” Button. 4. Admin/Staff enter the sales order information by filling in the following details: 4.1. input buyer Information 4.2. input product detail 4.3. Order Date 4.4. status 4.5. quantity 5. Click “Create” Button. 6. Sales Order is created and saved in the database. 7. System will displays a list of sales order created in a table form.
Postconditions: 4. Sales Order is successfully generated and stored in the database. 5. Sales Order can be viewed and accessed. 6. Sales order can be printed out.
Alternative flow: 2. -
Exception flow (if any): 3. If the Admin/Staff clicks the “Cancel” button, the system will cancels the sales order creation process.

Use Case Description for Create Outbound Delivery (Nik Zulaikhaa)

Use case: Create Outbound Delivery
ID: UC003
Actors: 5. Admin 6. Staff
Preconditions: 5. Admin/Staff is logged into the system 6. Sales order and invoice have been generated
Flow of events: 8. Admin/Staff opens the Sales Management System interface. 9. Admin/Staff navigates to the Outbound Delivery Page. 10. Click “Create Outbound Delivery” Button. 11. Fill in the following details: 11.1. Sales Order ID 11.2. Customer Name 11.3. Order Date 12. Click “Create” Button. 13. Delivery Order is generated and saved in the database. 14. System displays a list of outbound deliveries, including the newly created delivery order.
Postconditions: 7. Delivery Order is successfully generated and stored in the database. 8. Delivery Order can be viewed and accessed.
Alternative flow: 3. -
Exception flow (if any): 4. If the user clicks the “Cancel” button, the system cancels the delivery order creation process.

Use Case Description for Pick Delivery Order (Tiew Chuan Shen)

Use case: Pick Outbound Delivery
ID: UC004
Actors: 5. Admin 6. Staff
Preconditions: 1. Admin/Staff is logged into the system. 2. Outbound Delivery has been created and confirmed. 3. Products are available and ready for delivery.
Flow of events: 1. Admin/Staff navigates to the “Pick Outbound Delivery” section of the Sales Management System interface. 2. Click the “Create” Button. 3. Admin/Staff enter the delivery order information by filling in the following details: 14.1. input delivery information 14.2. input product detail, retrieves the corresponding product, price, and unit details automatically based on the delivery unit ID. 14.3. Order Date 14.4. status 5. Users enter the delivery number to search for an existing delivery order. 6. The system processes the request by searching the database for the delivery information. 7. If found, the system will display the delivery order created in a search list. 8. Users can change the status of the delivery order to be picked. 9. The system processes the delivery header and generates the output for the selected delivery.
Postconditions: 1. Delivery Order is successfully generated and stored in the database. 2. Delivery Order can be viewed and accessed to be processed. 3. Delivery Order can be picked.
Alternative flow: 1. If the Admin/Staff decides to cancel the search, the system redirects them back to the "Pick Outbound Delivery" section without processing.
Exception flow (if any): 2. If the Admin/Staff clicks the “Cancel” button, the system will cancel the delivery order creation process. 3. If the quantity specified exceeds the available stock, the delivery order cannot be picked up.

Use Case Description for Post Goods Issue (Lee Yik Hong)

Use case: Post Goods Issue
ID: UC005
Actors: 1. Admin 2. Staff
Preconditions: 1. Admin/Staff is logged into the system. 2. Outbound Delivery has been created and confirmed. 3. Products are available and ready for delivery.
Flow of events: 1. Admin/Staff opens the Sales Management System interface. 2. Admin/Staff navigates to the "Post Goods Issue" page. 3. Click the “Post Goods Issue” button. 4. Admin/Staff selects an Outbound Delivery ID from the available list. 5. Enter the following details: ● Outbound Delivery ID ● Customer Name ● Goods Issue Date ● Confirmation of product details (e.g., quantity, condition) 6. Click the “Post” button to confirm the goods issue. 7. The system processes the information and updates the database to reflect: ● Inventory reduction. ● Status update for the delivery as "Goods Issued". ● Generation of a Goods Issue document. 8. The system displays a confirmation message indicating the successful posting of the goods issue. 9. A printable Goods Issue document is generated and saved.
Postconditions: 1. Goods Issue is successfully posted in the database. 2. Inventory records are updated to reflect the delivered items. 3. Goods Issue document is generated and can be printed.
Alternative flow: 1. If Admin/Staff decides not to proceed, they can click the "Cancel" button, and the system redirects them to the previous page without saving any changes.
Exception flow (if any): 1. If the system detects insufficient stock, the Goods Issue process cannot proceed, and an error message is displayed. 2. If the entered Outbound Delivery ID is invalid, the system prompts the user to select a valid ID. 3. If there is a system error, the process is halted, and an appropriate error message is shown.

Activity Diagram (individual)

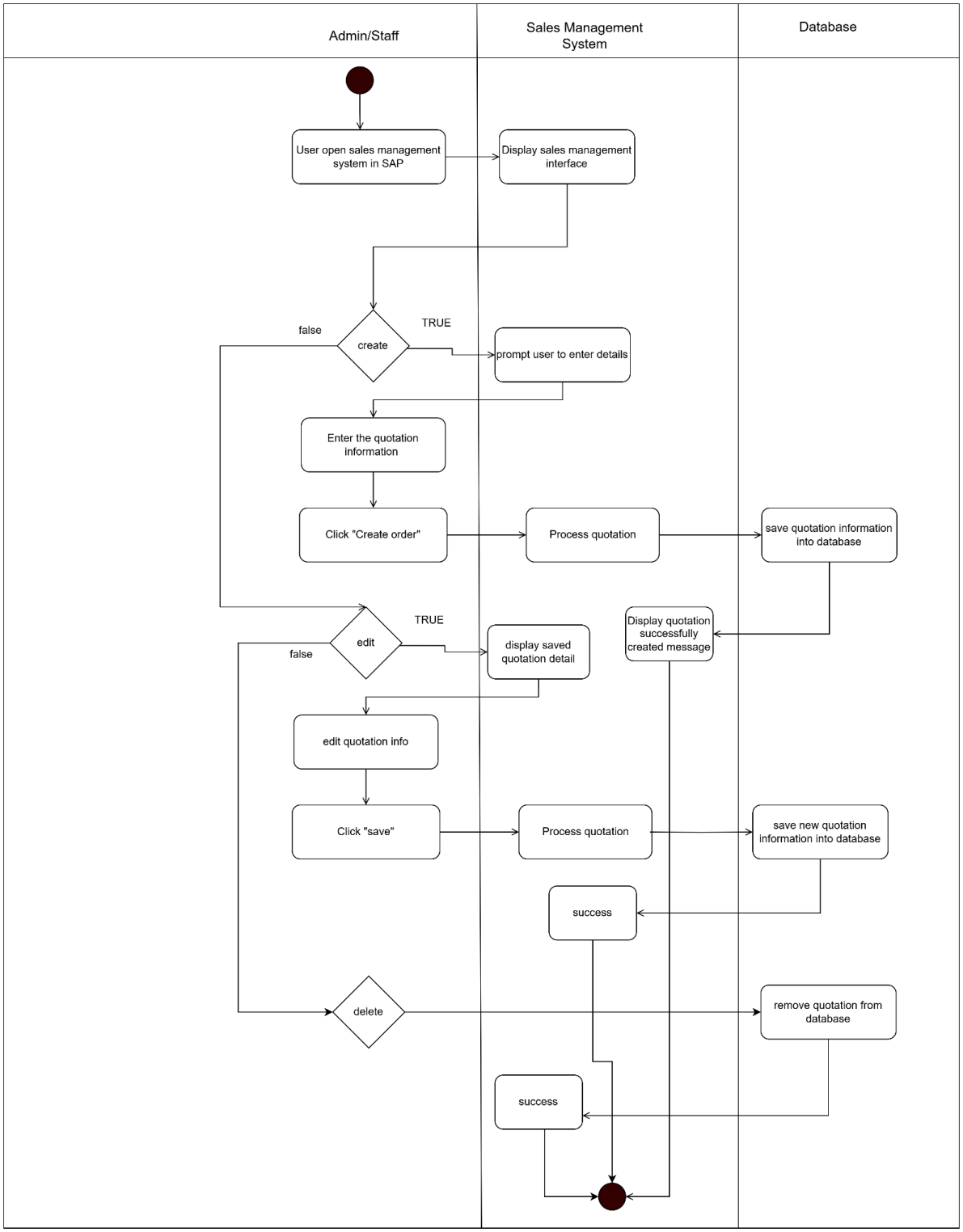


Figure 4.8.5.2:Activity diagram - create quotation (Loo Jia Chang)

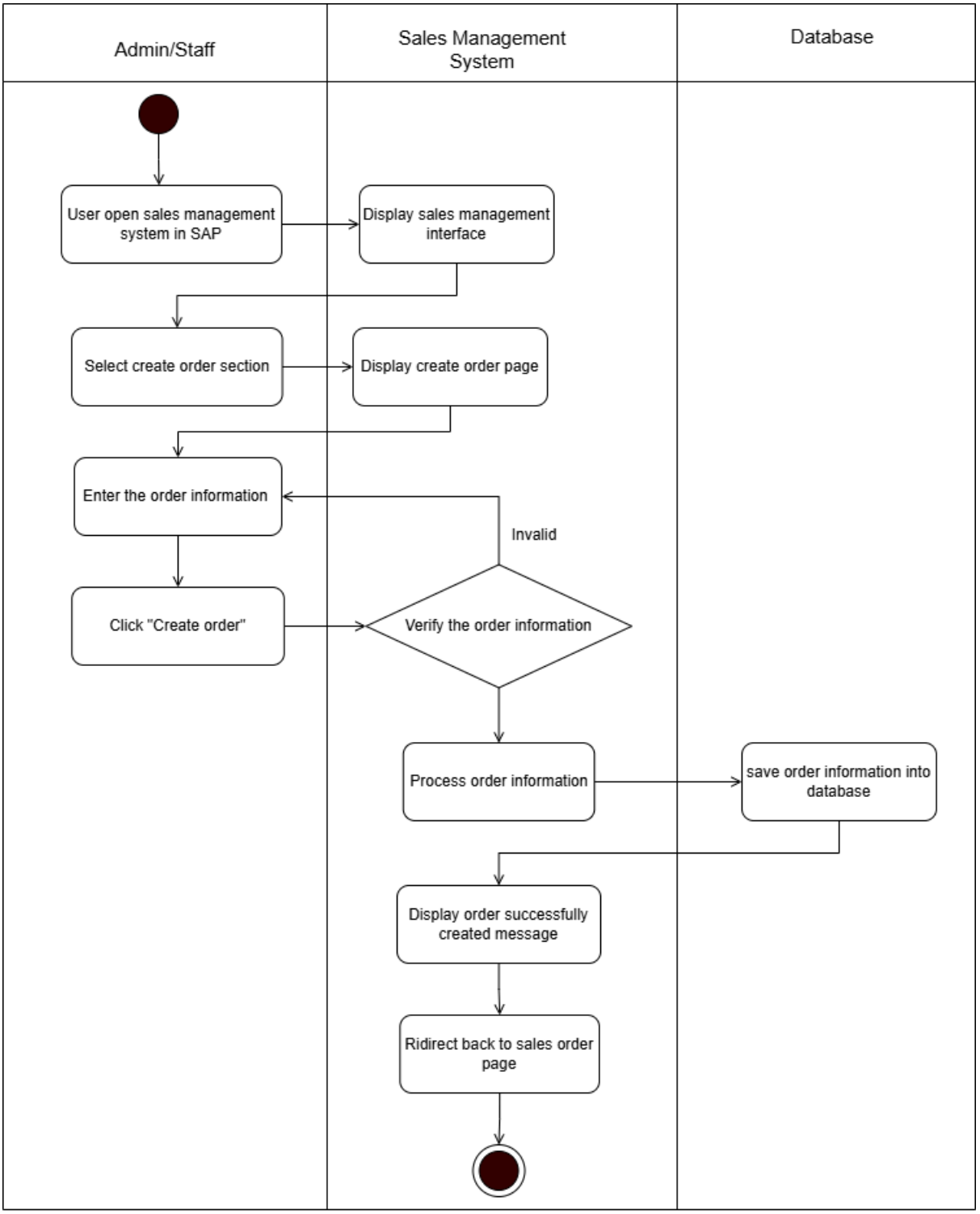


Figure 4.8.5.2:Activity diagram - create sales order (Tiew Chuan Rong)

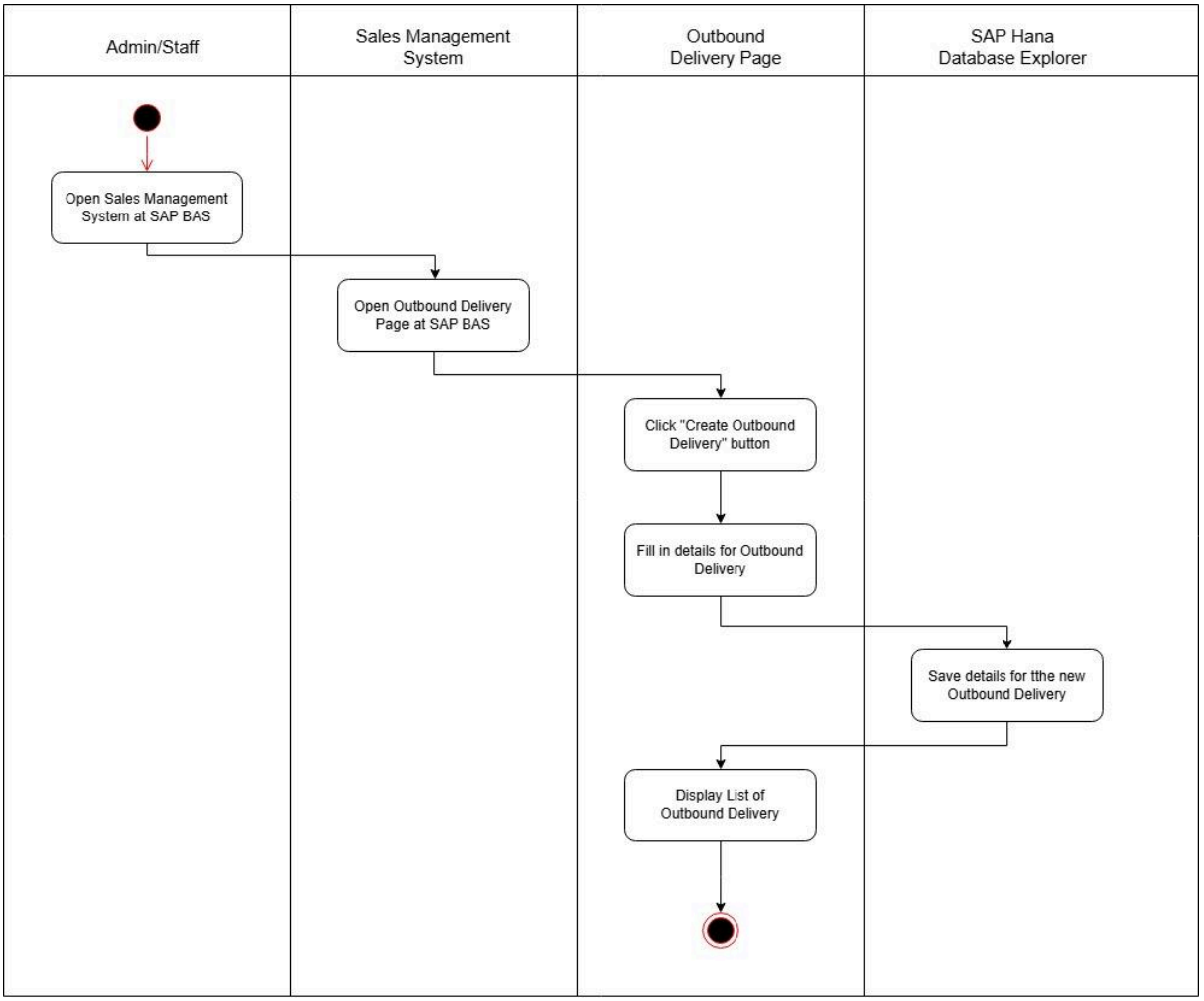


Figure 4.8.5.3: activity diagram - create outbound delivery (NIK ZULAIKHAA)

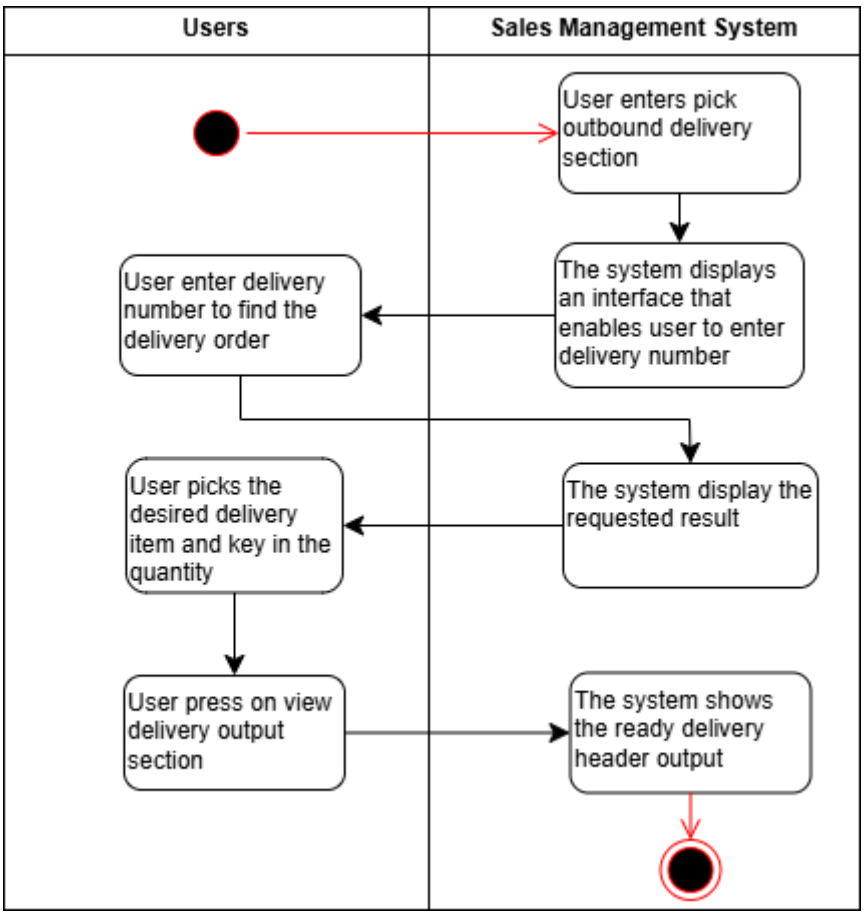


Figure 4.8.5.4: activity diagram - pick outbound delivery (Tiew Chuan Shen)

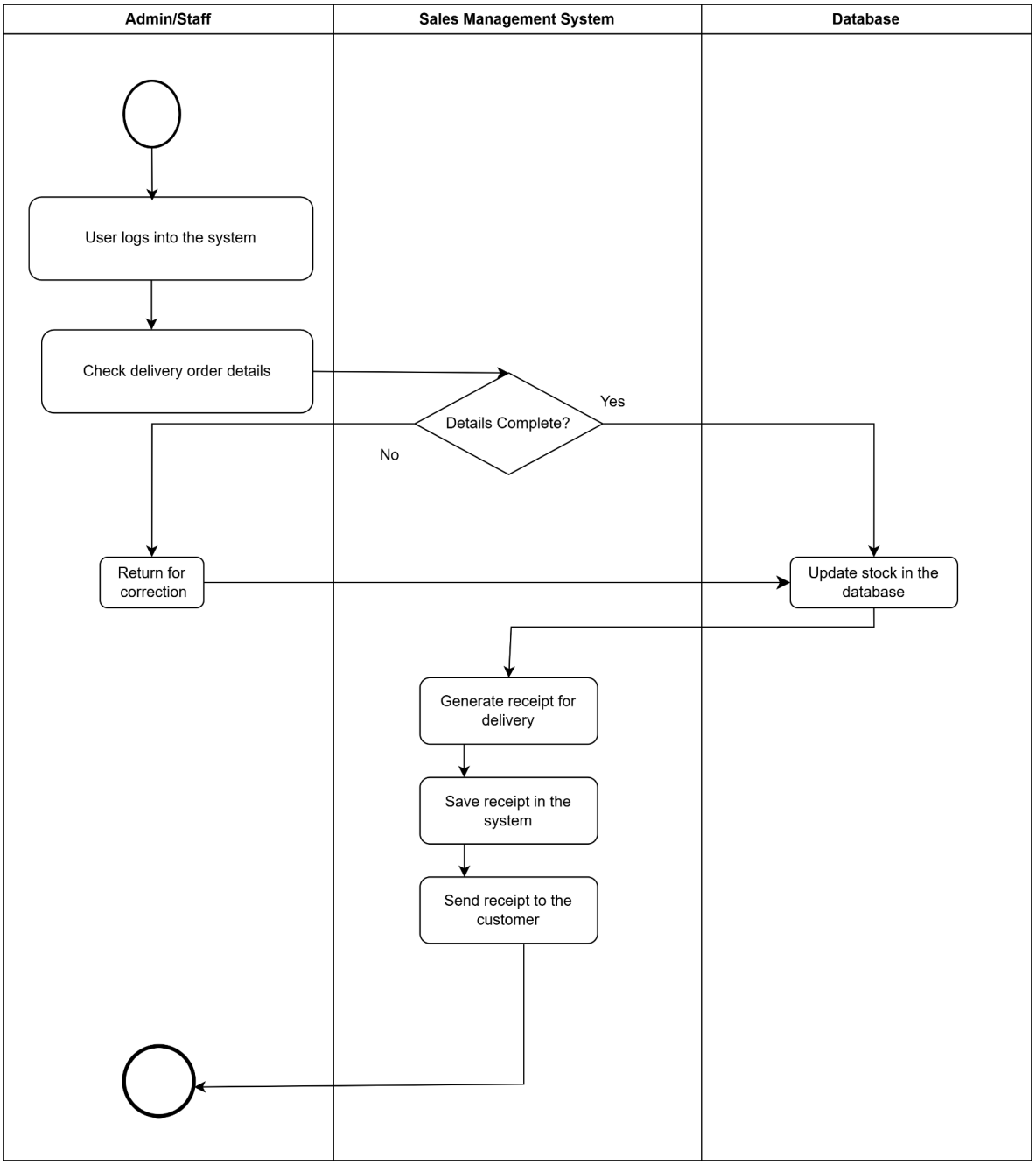


Figure 4.8.5.4: activity diagram - Post Goods Issue (Lee Yik Hong)

Sequence (individual)

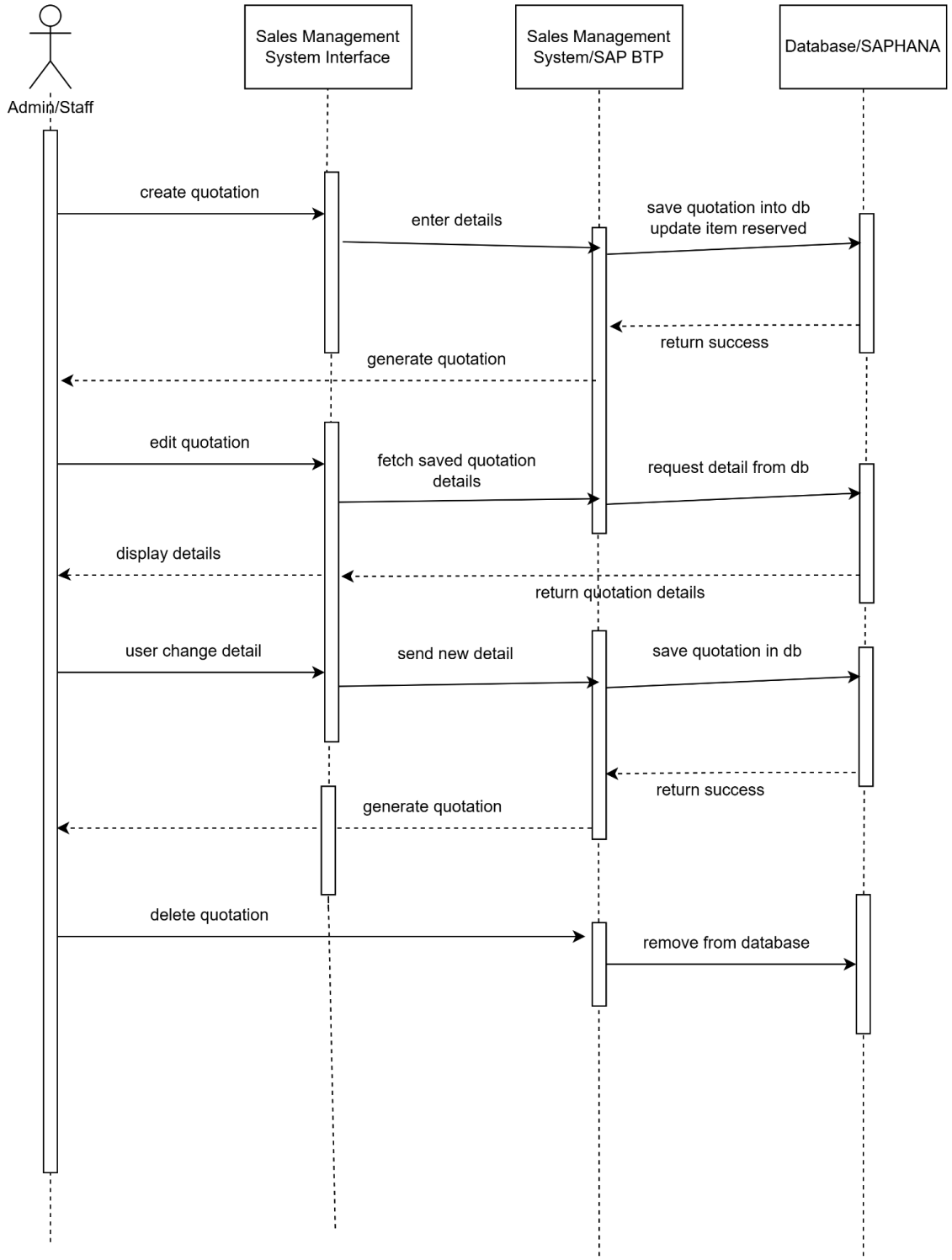


Figure **:** ***sequence diagram - create quotation (Loo Jia Chang)***

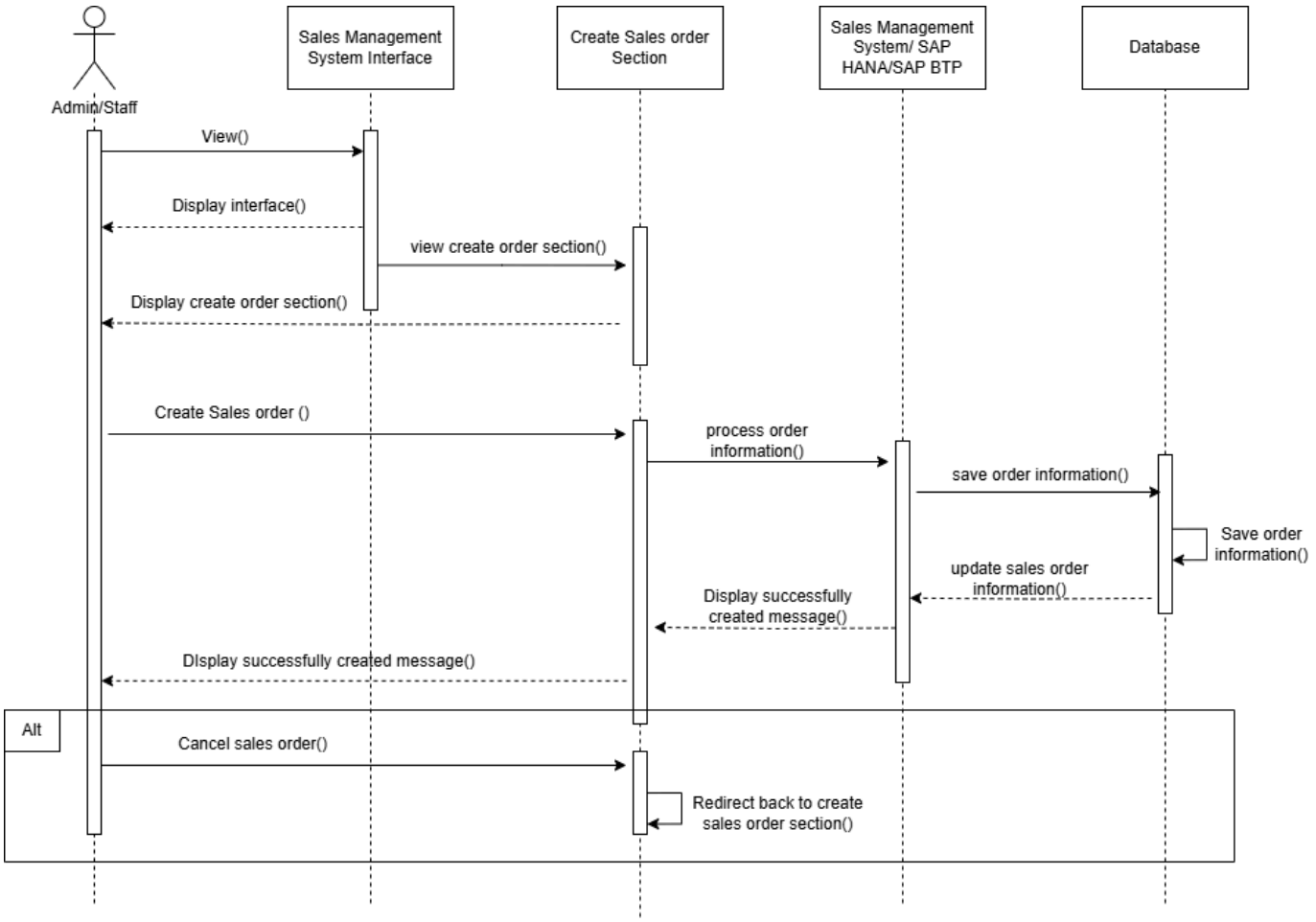


Figure **:** ***sequence diagram - create sales order (Tiew Chuan Rong)***

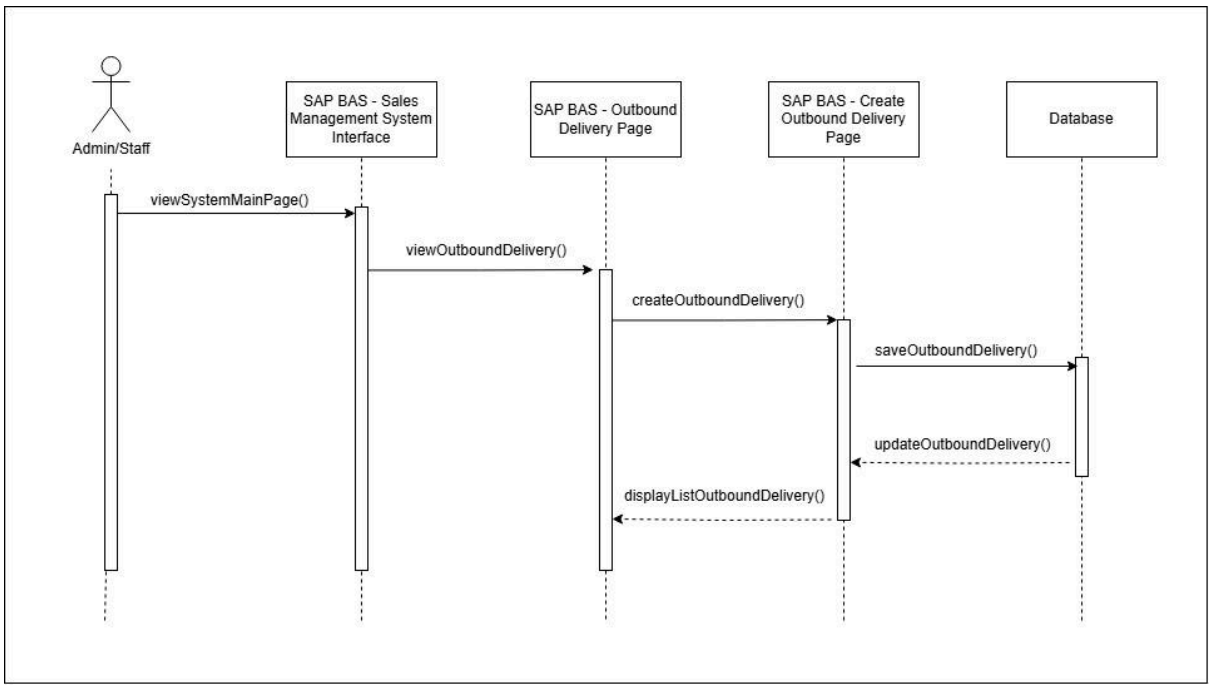


Figure: sequence diagram - create outbound delivery (NIK ZULAIKHAA)

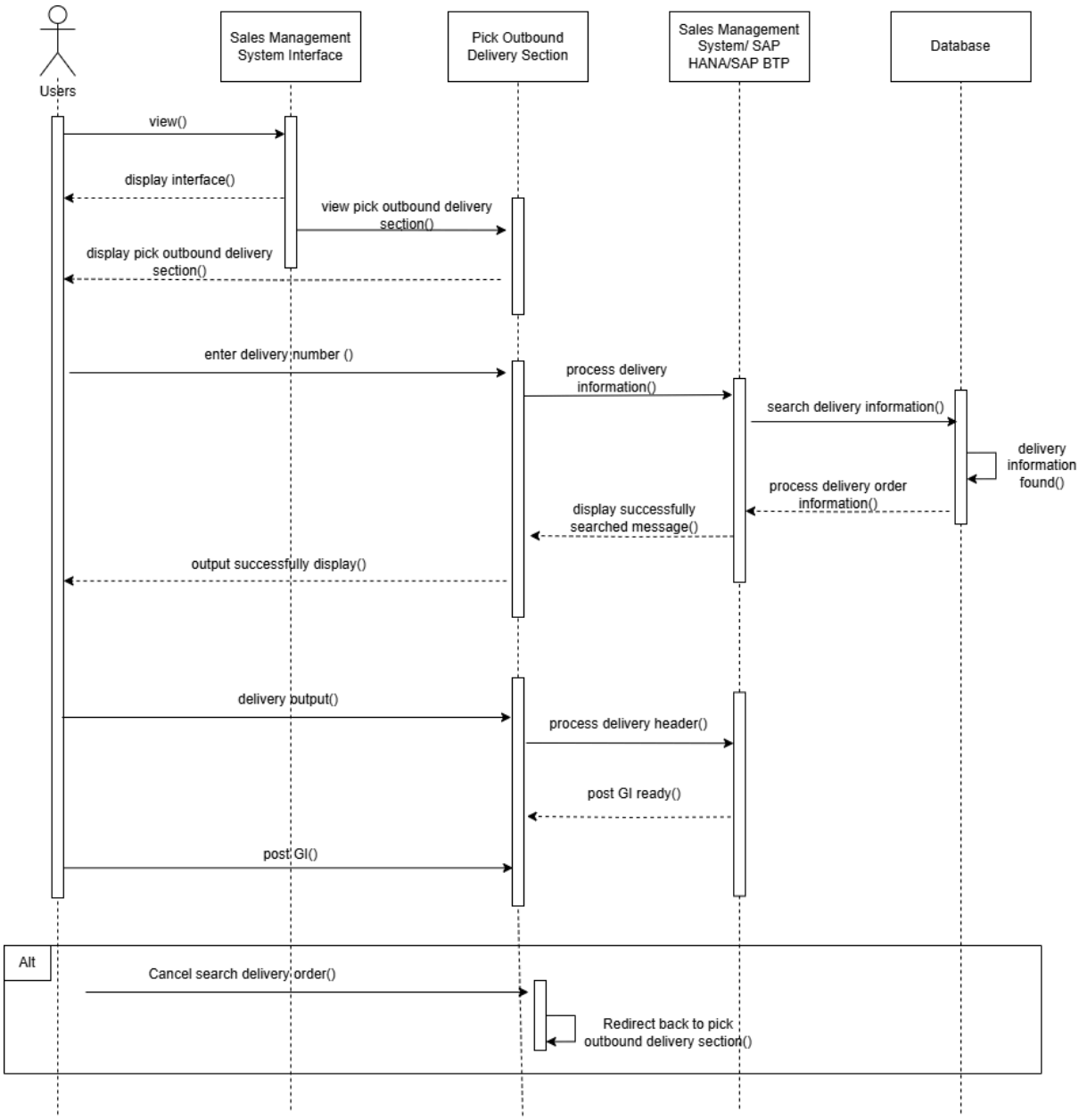


Figure: sequence diagram - pick outbound delivery (Tiew Chuan Shen)

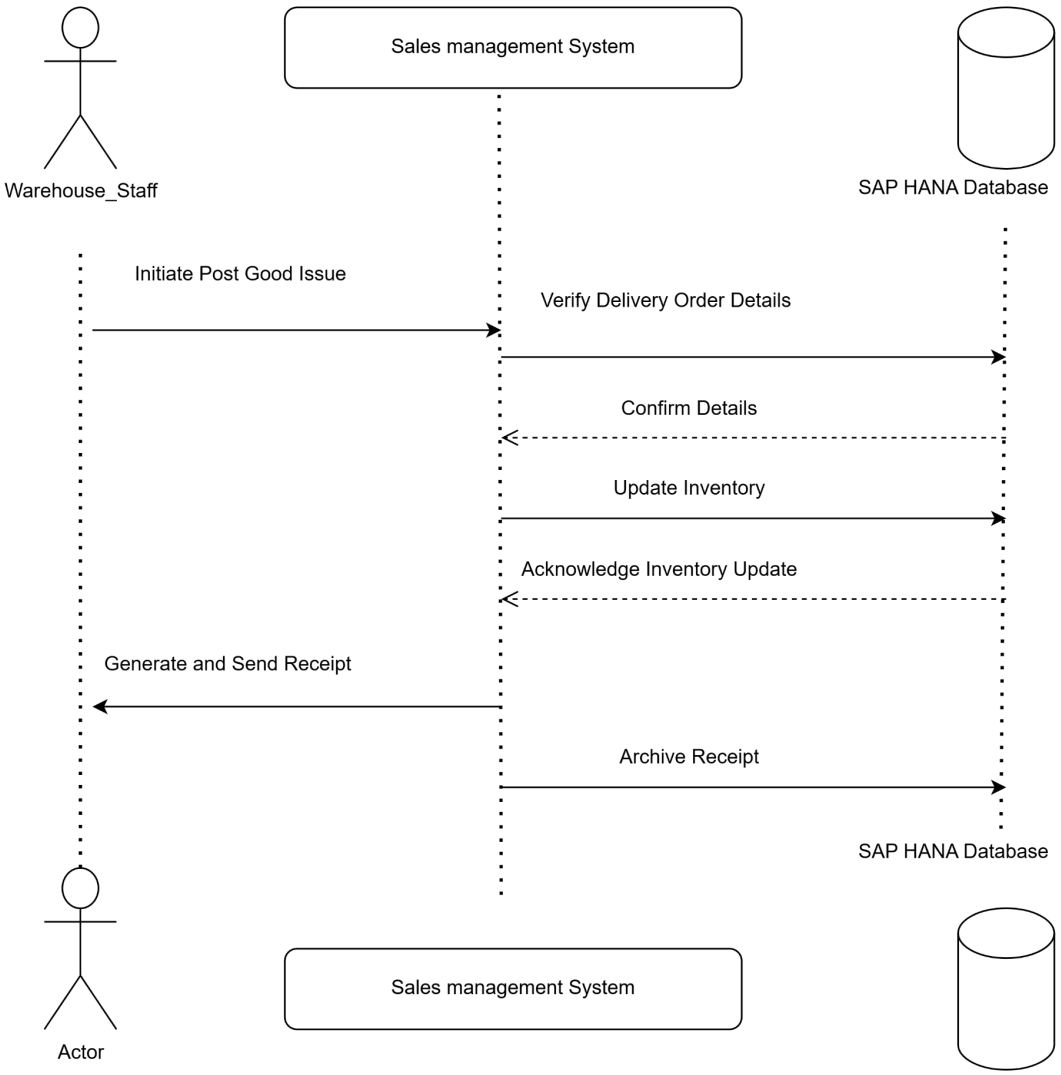


Figure: sequence diagram - Post Goods Issue (Lee Yik Hong)

4.8.6 Database Design - one design database for enterprise system

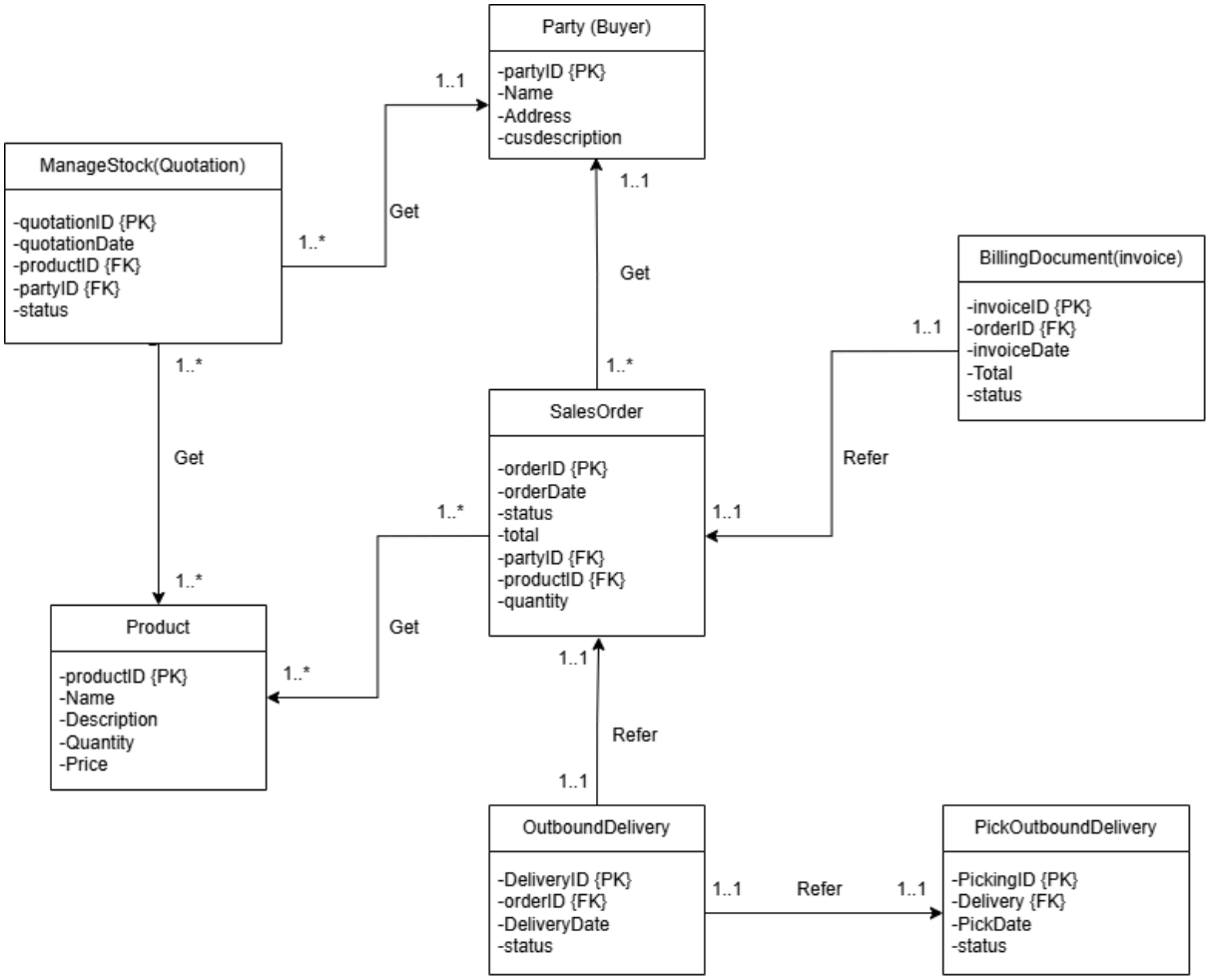


Figure 4.8.6.1: Database Design

4.8.7 Interface Design (individual)

Create Sales order Interface Design - (Tiew Chuan Rong)

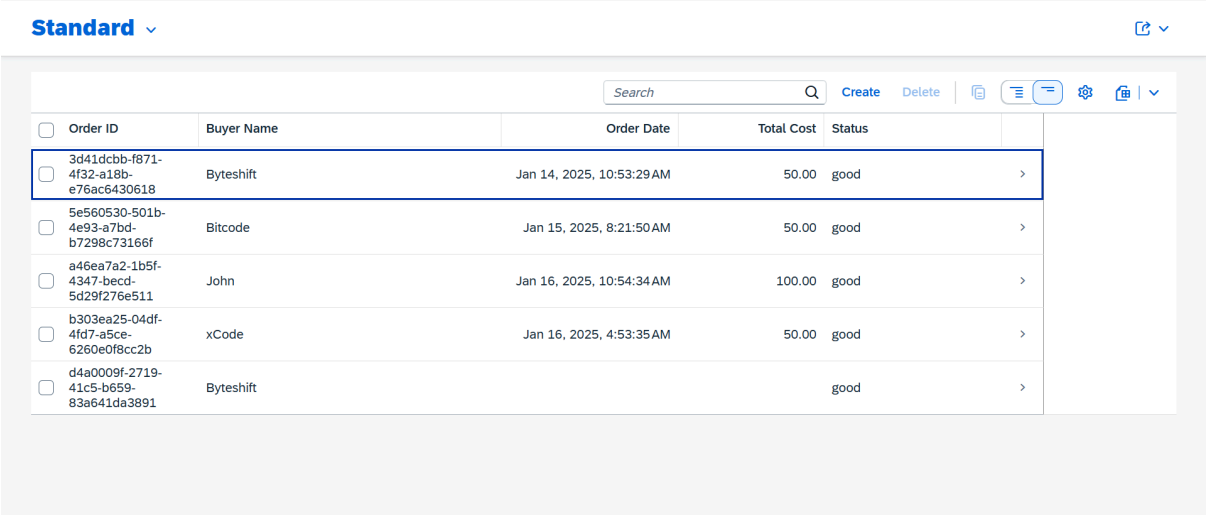


Figure 4.8.7.1: Create sales order interface design 1

The above Figure 4.8.7.1, is the simple and well organised interface for a system of creating sales orders. The interface design consists of a table with rows and columns displaying information about the sales order created.Each row represents one order and the columns show details like Order ID, Buyer Name, Order Date, Total Cost and status. On the top right, there are buttons for actions like “Create” to create new sales orders and “Delete” to remove the selected orders that are created. There is a search bar for users to quickly find a specific order that is created by typing the keyword. There are also settings for adjusting the view of the table.

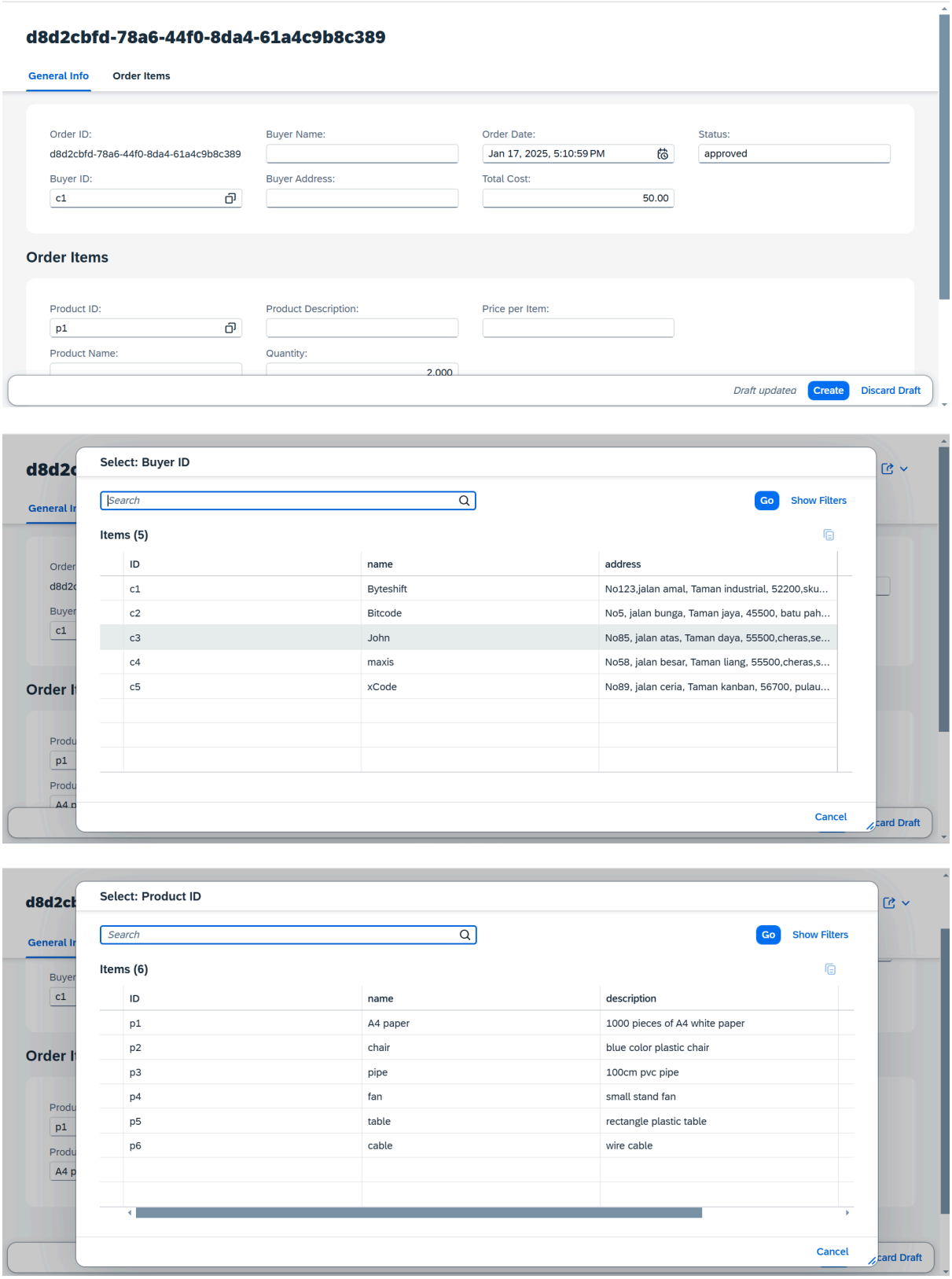


Figure 4.8.7.2: create sales order interface design 2

The figure 4.8.7.2 shows that the place for input buyer information and order item information. The user can just select the buyer id that is in the list then it will auto fill the buyer information in the column. Same for the product/item the user can select it from the list that they want to add then it will auto-fill the product information in the column. The user has to input the order date, quantity and status of the order itself. The order ID will be auto generated.

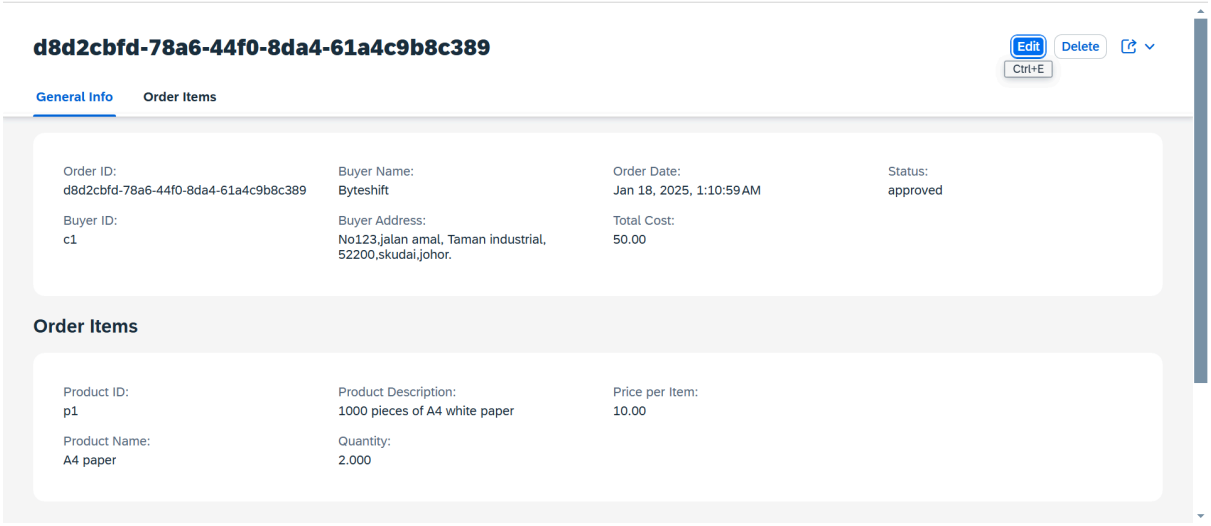
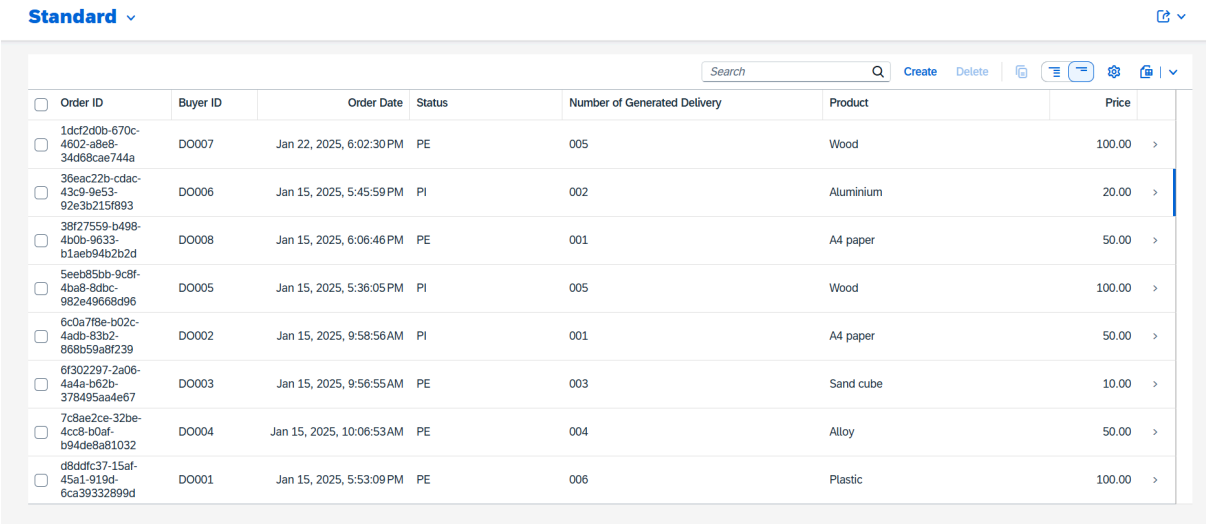


Figure 4.8.7.3: display created sales order information interface design 3

The figure 4.8.7.3 will show the information of the order after it click the create button.

Pick Outbound Delivery Interface Design - (Tiew Chuan Shen)



Order ID	Buyer ID	Order Date	Status	Number of Generated Delivery	Product	Price
1dcf2d0b-670c-4602-a8e8-34d68cae744a	DO007	Jan 22, 2025, 6:02:30 PM	PE	005	Wood	100.00
36eac22b-cdac-43c9-9e53-92e3b215f893	DO006	Jan 15, 2025, 5:45:59 PM	PI	002	Aluminium	20.00
38f27559-b498-4b0b-9633-b1aeb94b2b2d	DO008	Jan 15, 2025, 6:06:46 PM	PE	001	A4 paper	50.00
5eeb85bb-9c8f-4ba8-8dbc-982e49668d96	DO005	Jan 15, 2025, 5:36:05 PM	PI	005	Wood	100.00
6c0a7f8e-b02c-4adb-83b2-868b59a8f239	DO002	Jan 15, 2025, 9:58:56 AM	PI	001	A4 paper	50.00
6f302297-2a06-4a4a-b62b-378495aa4e67	DO003	Jan 15, 2025, 9:56:55 AM	PE	003	Sand cube	10.00
7c8ae2ce-32be-4cc8-b0af-b94de8a81032	DO004	Jan 15, 2025, 10:06:53 AM	PE	004	Alloy	50.00
d8ddfc37-15af-45a1-919d-6ca39332899d	DO001	Jan 15, 2025, 5:53:09 PM	PE	006	Plastic	100.00

Figure 4.8.7.4: Pick Outbound Delivery interface design 1

The above figure shows the interface of pick outbound delivery after clicking into the pick section. As you can see, the interface design consists of a table with rows and columns displaying information about the delivery order created. Each row represents one order, and the columns show details like Order ID, Buyer ID, Order Date, the number of generated deliveries, and status. On the top right, there are buttons for actions like “Create” to create new delivery orders and “Delete” to remove the selected orders that are created. There is a search bar for users to quickly find a specific order that is created by typing the matching keyword. There are also settings for adjusting the view of the table. To see the status of the delivery order, it is either picked or pending status, which means the product has not yet been delivered.

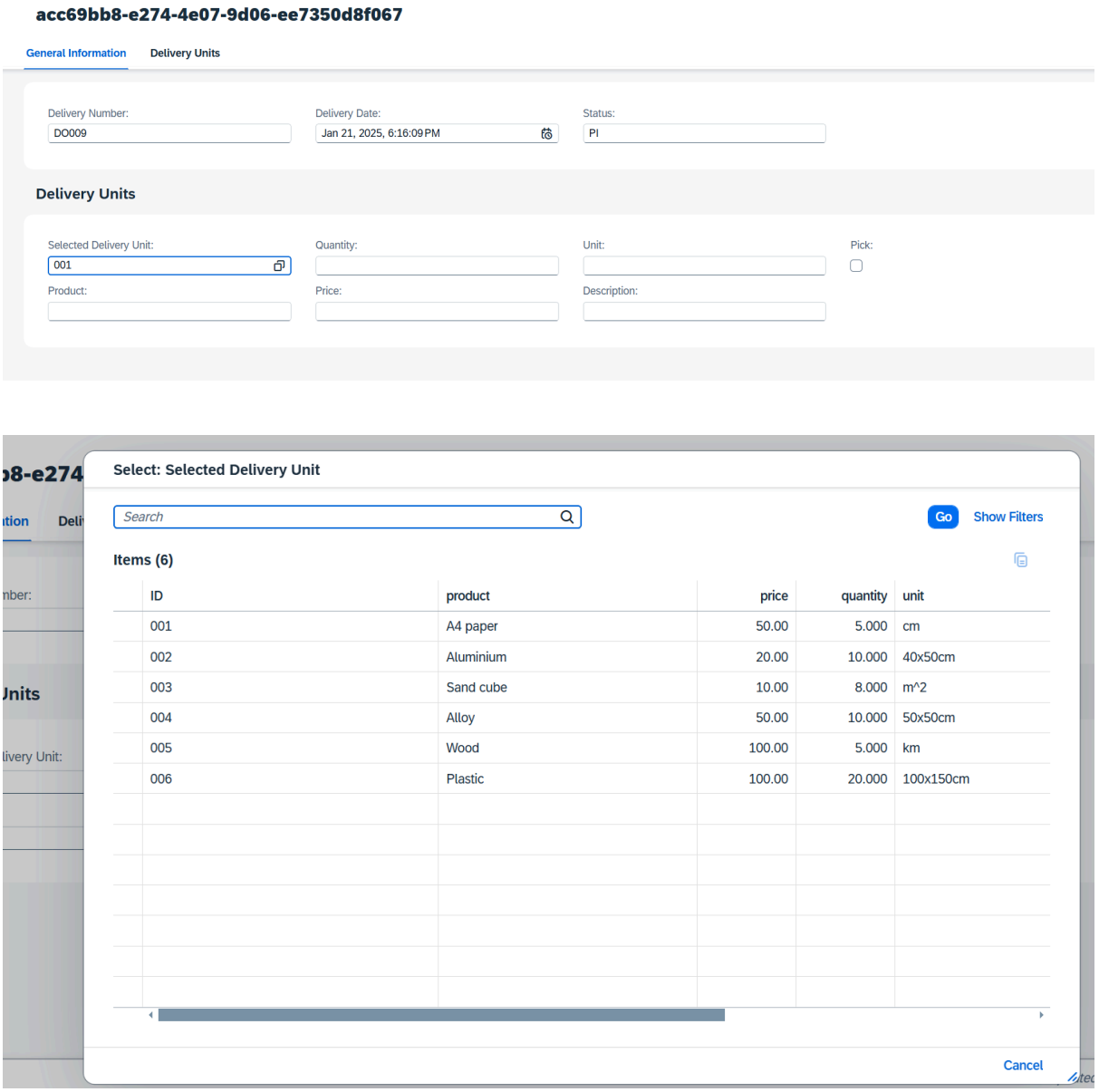


Figure 4.8.7.5: Pick Outbound Delivery Interface Design 2

The figure 4.8.7.5 shows that the place for input delivery order information and delivery order product information. The user can just select the product id that is in the list then it will auto fill the product information in the column. The user has to input the delivery number, delivery date and changing status of the order itself. The delivery order ID will be auto generated.

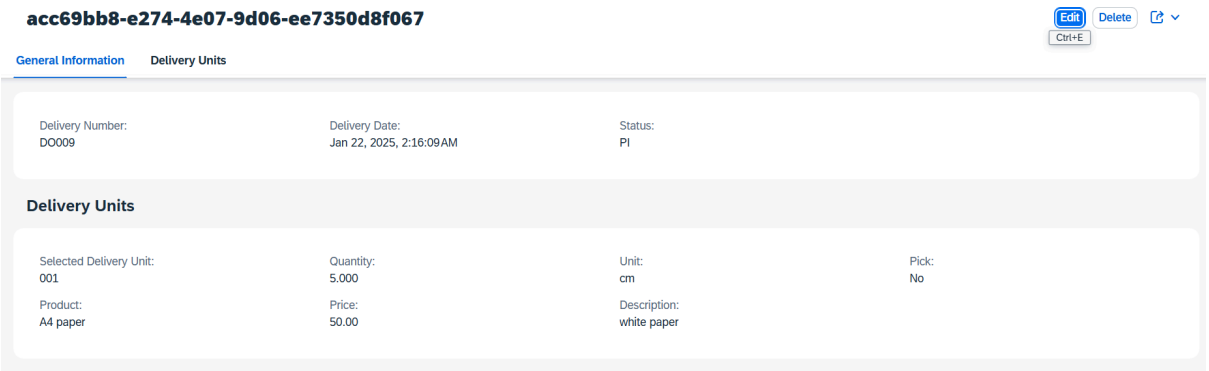


Figure 4.8.7.6: Pick Outbound Delivery Interface Design 3

Figure 4.8.7.6 will show the output information of the delivery order after clicking the create button. You can edit it or delete it if the buyer has changed the decision.

Post Good Issue Interface Design - (Lee Yik Hong)

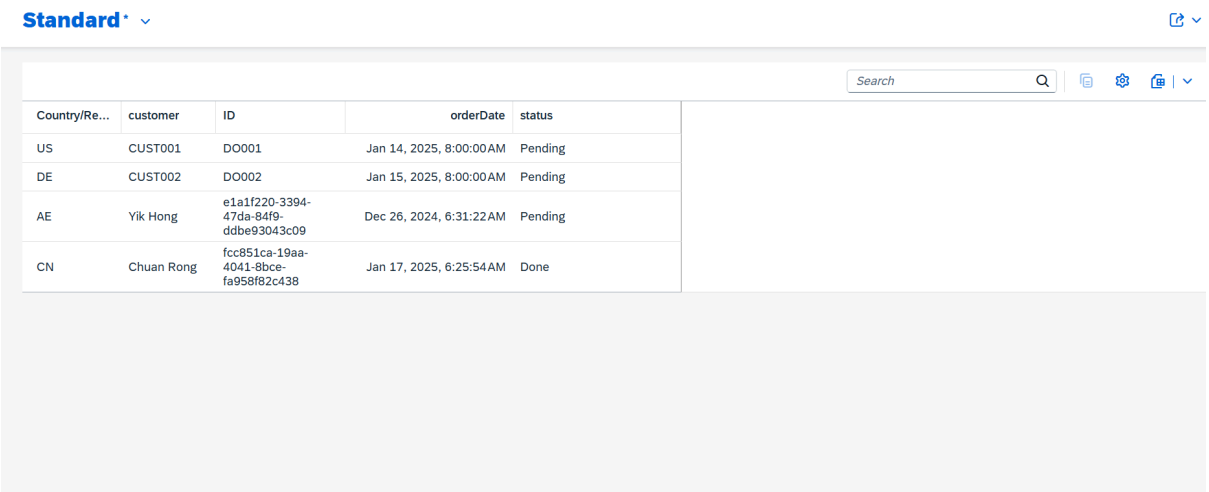


Figure 4.8.7.7: Initiate Post Good Issue interface design

The figure 4.8.7.7 shows the interface for initializing the item list in the Post Good Issue process. This interface is designed to extract and display data from both the **DELIVERYORDER** and **DELIVERYORDERITEMS** databases. It consists of a table with rows and columns that present detailed information about the delivery orders and their associated items. Each row represents a single delivery order or item, while the columns display attributes such as Country/Region, Customer Name or ID, a unique Order ID, the

Order Date and Time, and the current Status of the order or item. The **Status** column indicates whether the item is still pending or has been completed.

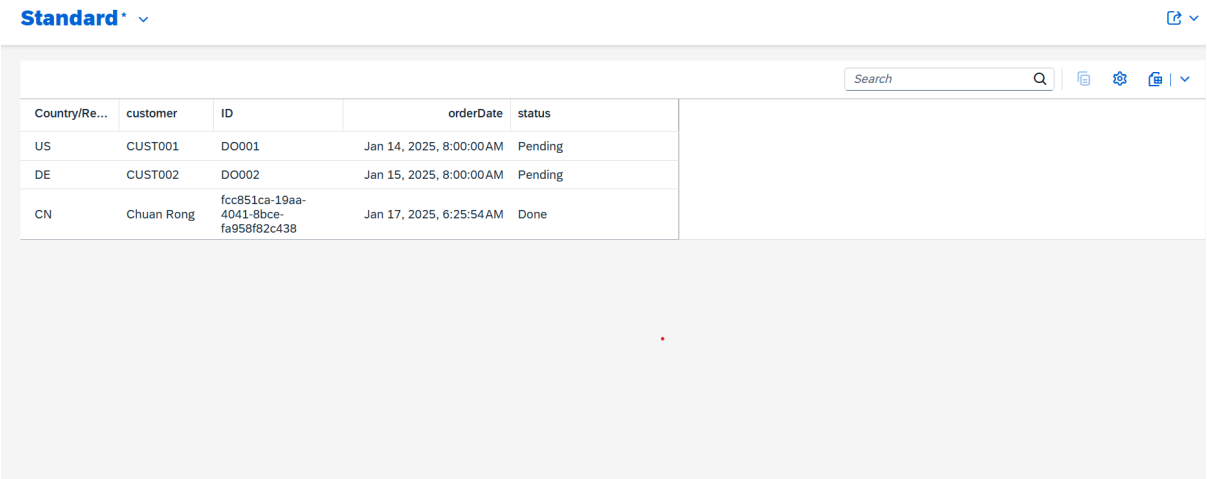


Figure 4.8.7.8: Update Inventory interface design

The figure 4.8.7.8 shows the interface for the **Update Inventory** process. This interface is designed to provide users with a streamlined view of delivery orders to manage and update inventory efficiently. The main design element is a table that displays information retrieved from the **DELIVERYORDER** and **DELIVERYORDERITEMS** databases. Each row in the table represents a specific delivery order, and the columns display critical details such as the Country/Region, Customer Name or ID, a unique Order ID, the Order Date and Time, and the current Status of the order. The **Status** column helps users easily track whether an order is still pending or has already been completed and updated from the database will immediately shown. For example, customer “Yik Hong” has been removed from the database ,due to done sending the invoice and it will update directly in the list.

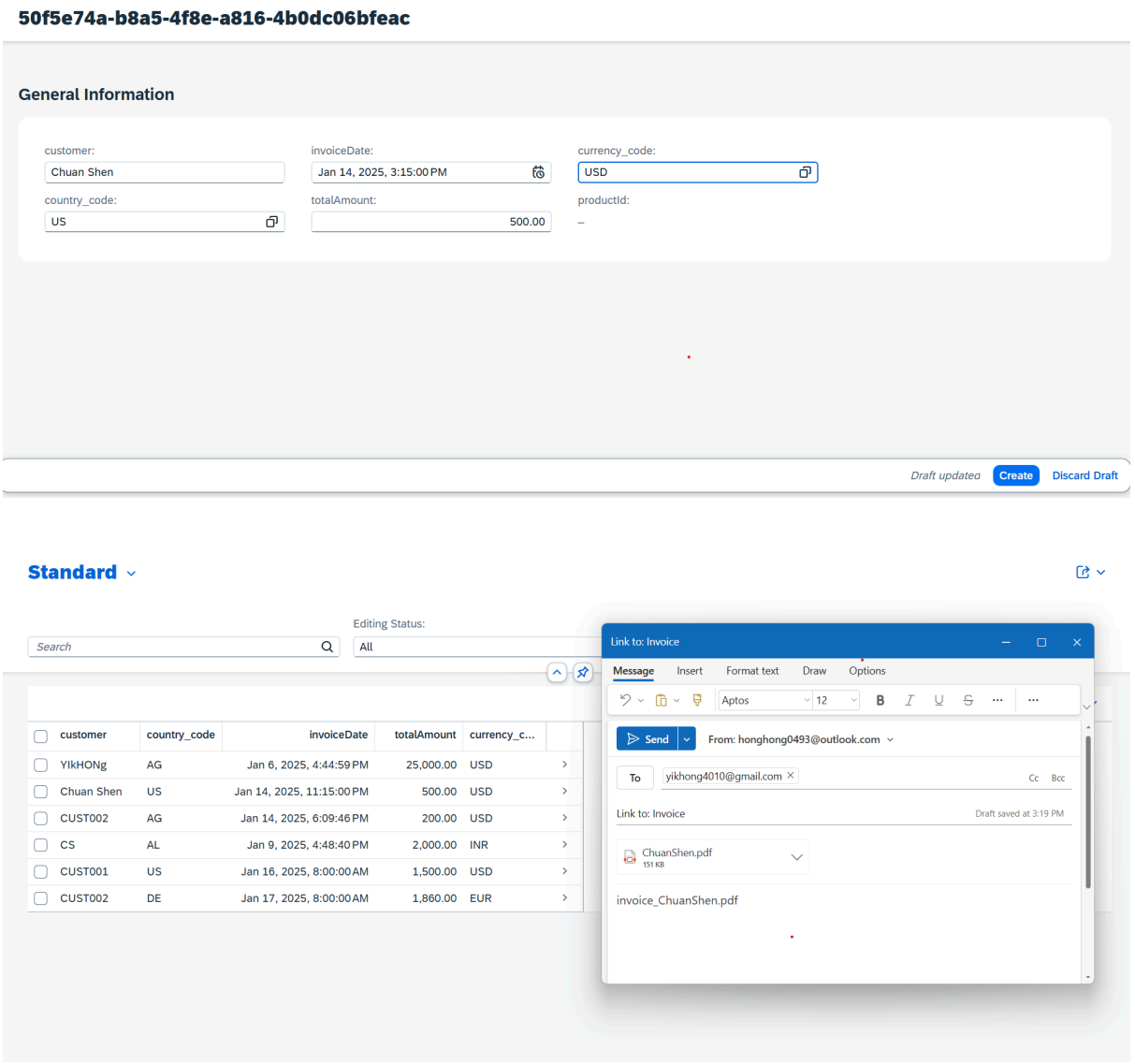


Figure 4.8.7.9: Generate and Send Receipt interface design

The above figures illustrate the **Generate and Send Receipt** interface design, which facilitates the creation and sharing of invoices. This interface retrieves and displays data from the **INVOICE** and **INVOICEITEMS** databases to ensure accuracy and consistency in managing invoice records.

In the **General Information** section, users can view and input key details about the invoice. Fields include the **Customer Name**, **Country Code**, **Invoice Date**, **Total Amount**, **Currency Code**, and **Product ID**. These fields are designed for both display and data entry, ensuring that all necessary invoice information is captured. The layout is simple and user-friendly, with a "Create" button to finalize the invoice and a "Discard Draft" option to

cancel the operation if needed. The section provides a structured and efficient way to input or update invoice details before they are processed.

The second figure highlights the interface for viewing and sending invoices. It includes a table listing all available invoices, with columns such as **Customer**, **Country Code**, **Invoice Date**, **Total Amount**, and **Currency Code**. Each row represents a specific invoice record, allowing users to quickly locate and manage invoices. Additional actions, such as selecting an invoice to email, are streamlined through integration with the email client. The email composition window appears when a user chooses to send a receipt, pre-filled with the relevant invoice file (e.g., a PDF attachment). This ensures the sharing process is fast, efficient, and reduces the risk of errors.

Overall, this interface design integrates seamlessly with the **INVOICE** and **INVOICEITEMS** databases to simplify generating and sending receipts. Its clear and intuitive layout ensures users can handle invoice-related tasks with ease, from creating new records to sharing finalized receipts with customers.

Quotation Interface Design - (Loo Jia Chang)

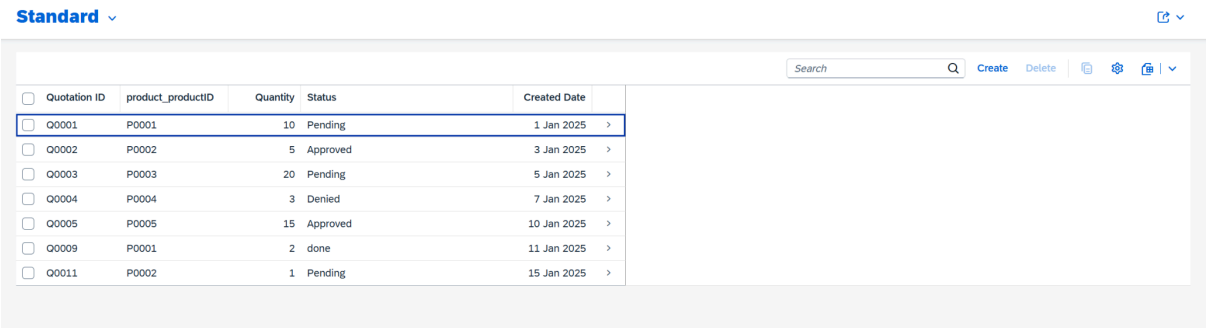


Figure 4.8.7.10:

All Quotation

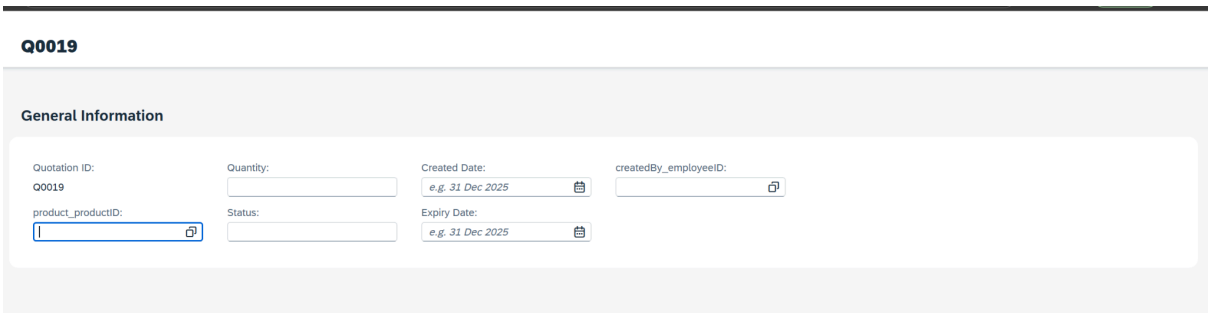


Figure 4.8.7.11:

Create Quotation

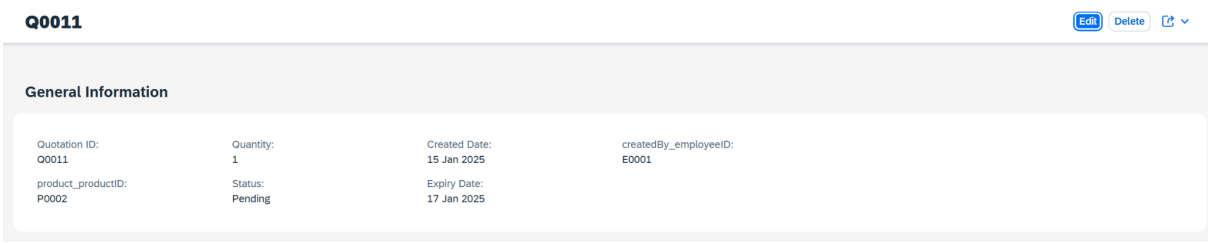


Figure 4.8.7.12:

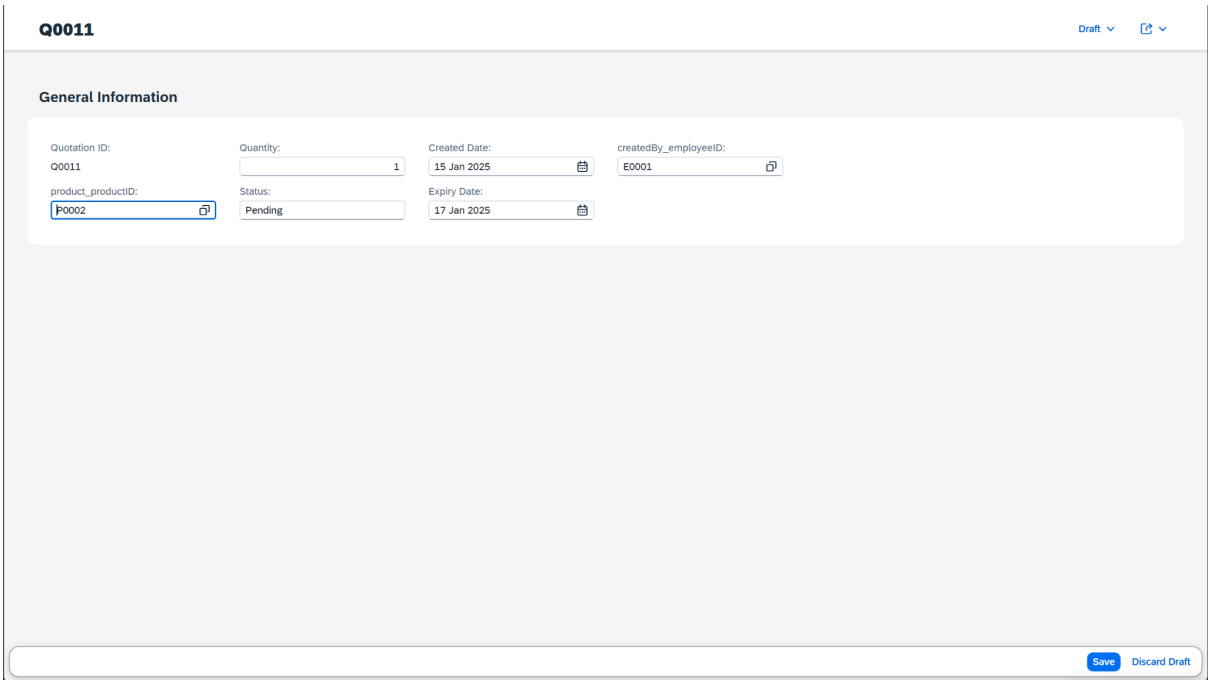


Figure 4.8.7.13:

Edit quotation

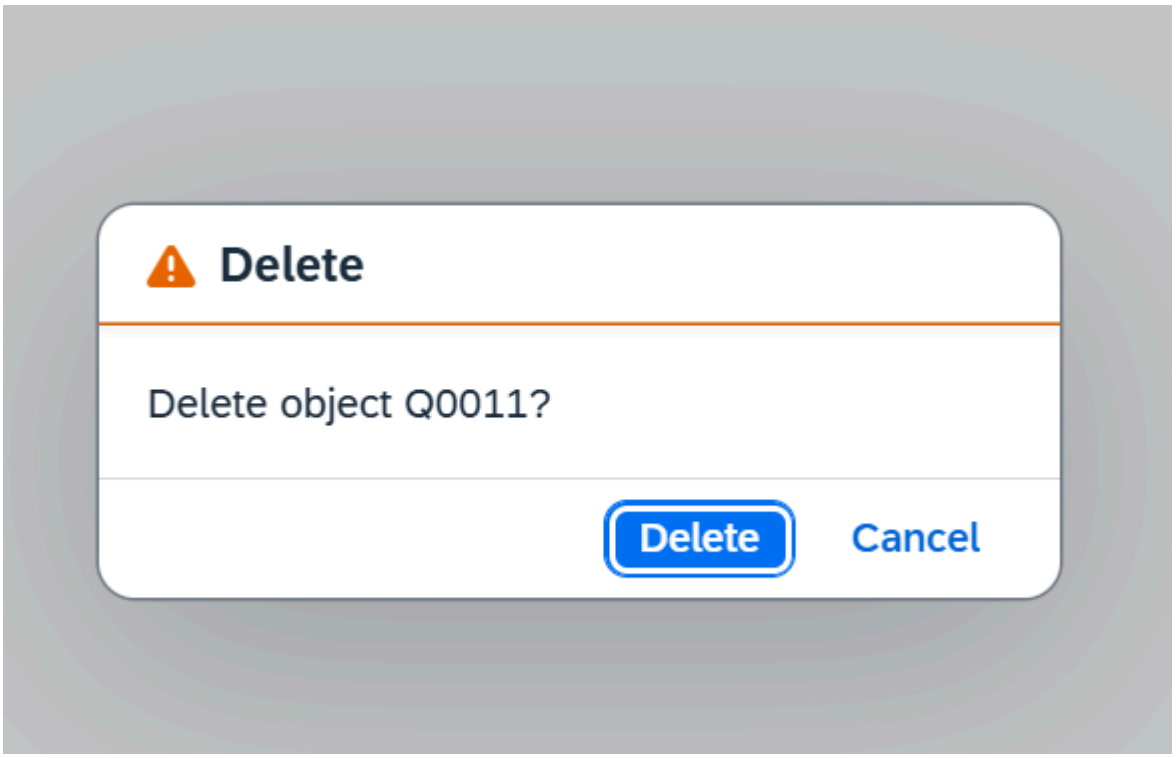


Figure 4.8.7.14:

Delete quotation

Create Outbound Delivery Interface Design - (Nik Zulaikhaa)

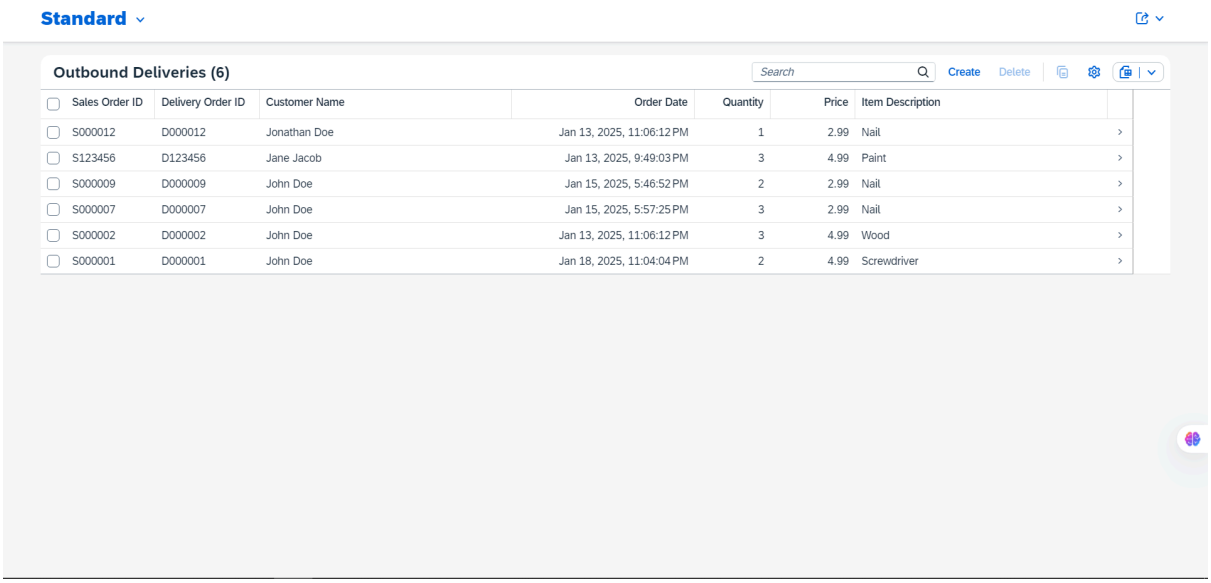


Figure 4.8.7.15: Outbound Delivery Page

The figure 4.8.7.15 shows the main page for the Create Outbound Delivery Interface. The user is required to click on the Create button to create a new outbound delivery order.

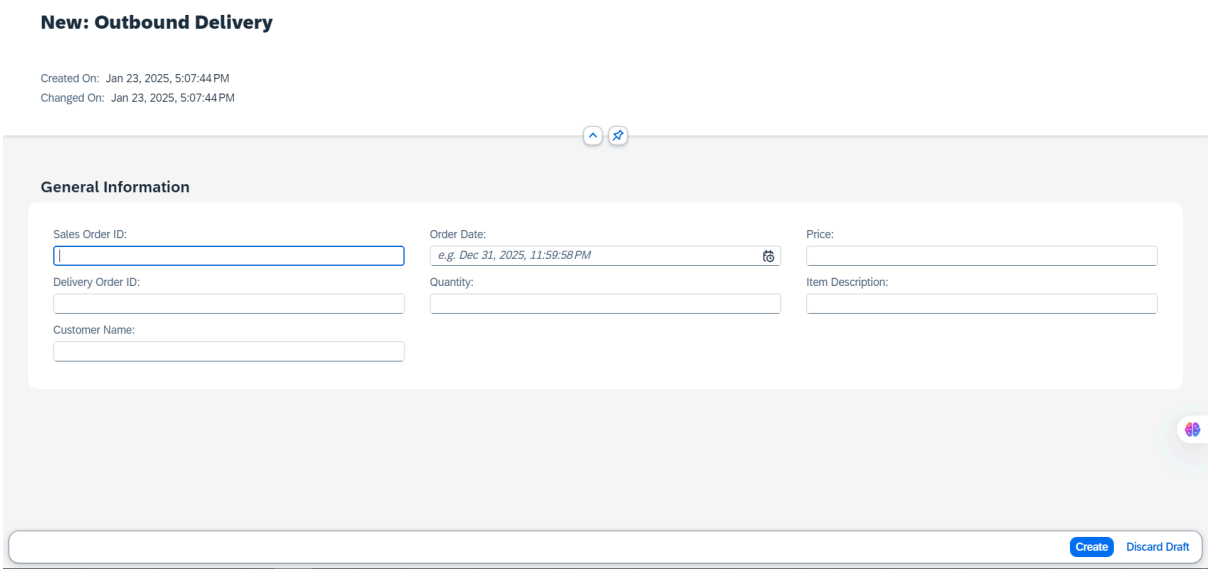


Figure 4.8.7.16: Create Outbound Delivery Form

The figure 4.8.7.16 shows the information that the user is required to fill in order to generate the new Outbound Delivery.

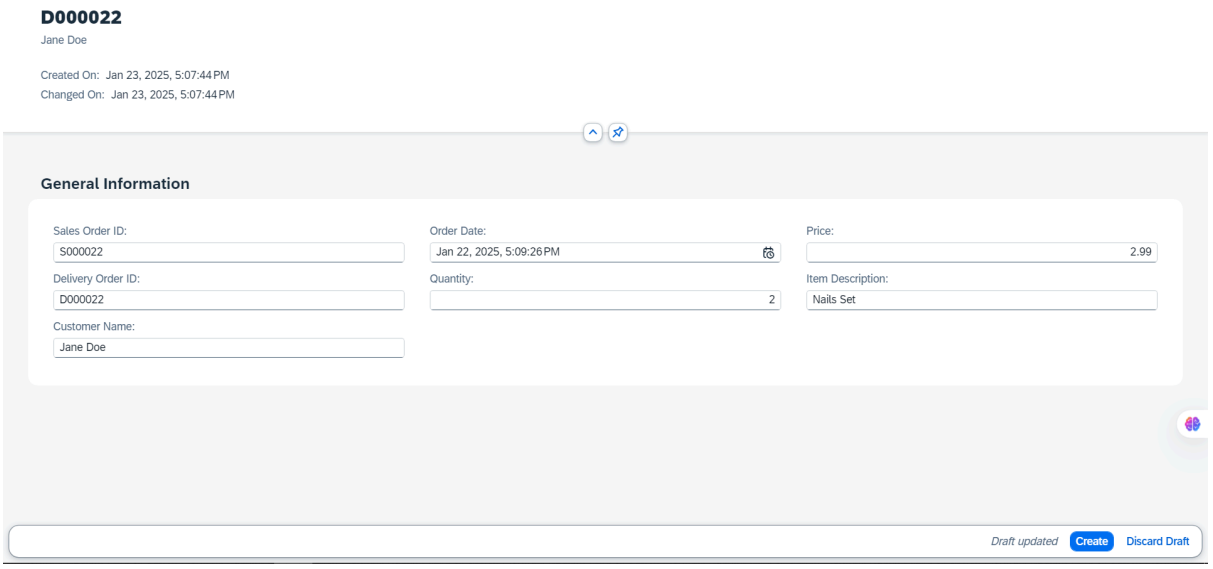


Figure 4.8.7.17: Create Outbound Delivery Form

The figure 4.8.7.17 displays the filled in form and the user is now required to click on the Create button in order to generate the new Outbound Delivery.

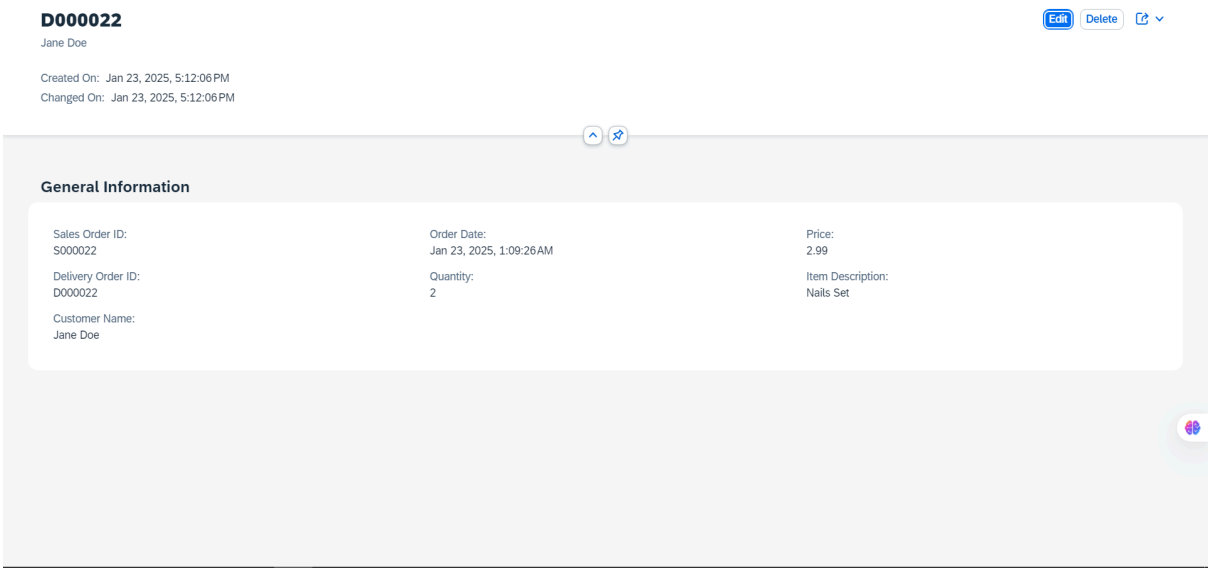


Figure 4.8.7.18: Generated Outbound Delivery Display

The figure 4.8.7.18 displays the generated Outbound Delivery form and is now ready for the Outbound Delivery document to be sent to the customer.

4.9 Summary

The Analysis and Design chapter provided a comprehensive approach to transforming AK Maju Resources' manual sales management system into an automated, SAP-integrated solution. Through systematic analysis of the organisation's current processes and challenges, the chapter outlined the requirements and the proposed system's design. The challengers are the existing manual system's inefficiencies, including delayed order processing, lack of real-time stock updates, and error-prone manual data entry. The new system incorporates automation in sales order processing, delivery tracking, and billing while ensuring integration with SAP HANA and SAP BTP for scalability and real-time operations. Detailed architectural diagrams, database designs, and user interface concepts to ensure usability, accuracy, and seamless data flow. To conclude, this chapter forms the foundation for implementing the system, ensuring alignment with organisational goals and addressing the limitations of the current setup.

Chapter 5: System Implementation

5.1 Introduction

In this chapter, we will discuss the process of implementing the system, which involves system development, creating the database, and coding the main functions and methods of the system. This system implementation phase will show how the system is designed into a working application.

To develop the system, we will use SAP BTP tools and platforms. In SAP BTP, we will be using SAP BTP Cockpit as our development environment. There are many functionalities and modules in SAP BTP Cockpit, and we will select SAP Business Application Studio for full-stack application development and SAP HANA Cloud for creating databases. SAP HANA Cloud operates on the Cloud Foundry platform. This database offers high performance and flexibility for handling large amounts of data while ensuring that it integrates seamlessly with our application. In the SAP Business Application Studio, we will select full-stack cloud applications to develop and implement our system. As we are using a trial account, there are many limitations, so this full-stack cloud application is the only function that provides a connection to the SAP HANA Cloud database. Besides that, full stack cloud application leverages SAP Fiori as its user interface framework, offering a modern and intuitive interface.

By using the SAP BTP platform, it provides a user-friendly interface and a comprehensive set of tools for creating, testing, and deploying applications. These tools provide a powerful and efficient way to develop and deploy the system in a cloud environment.

5.2 System Development

System development is the process of turning our design into an actual working system. It includes creating the database, coding the system's main functions, and ensuring everything works together smoothly. For this project, we used SAP Business Technology Platform (SAP BTP) because it has all the tools needed to create, test, and deploy our system in a cloud environment.

The tools that we used are

1. SAP Business Application Studio:
- It allows us to develop full-stack cloud applications, meaning we can create both the backend and the frontend .
- It's a powerful, cloud-based development environment specifically designed for SAP applications.
2. SAP HANA Cloud:
- This is the "Storage" of the system, where all the data is stored.
- It works on a platform called Cloud Foundry and is great for handling large amounts of data.
- It integrates smoothly with other SAP tools like BAS, which makes the development process more efficient.
3. SAP Fiori:
- This is what gives the system a modern and easy-to-use interface.
- It ensures that users can navigate and use the system effortlessly.

Steps in Developing the System

1. **Planning:**
    - Before we started developing, we had designed the system properly like drawing system architecture, enterprise architecture, use case, sequence diagram, activity diagram and database.

- We listed all the features we wanted, such as creating sales orders, managing stock(quotation), creating outbound delivery, picking outbound delivery and generating billing reports(invoice).

2. **Database Creation:**
    - We used SAP HANA Cloud to set up the database. This involved creating tables to store data for things like products, customers, orders, and deliveries.
3. **Coding the Main Functions:**
    - We code to make the system work. For example:
        - A function to create a sales order.
        - A function to manage stock (invoice).
        - A function to create a delivery order.
        - A function for picking delivery orders.
        - A function to generate a billing report (invoice).

5.2. Create Database (individual)

Create Sales Order Database - (Tiew Chuan Rong)

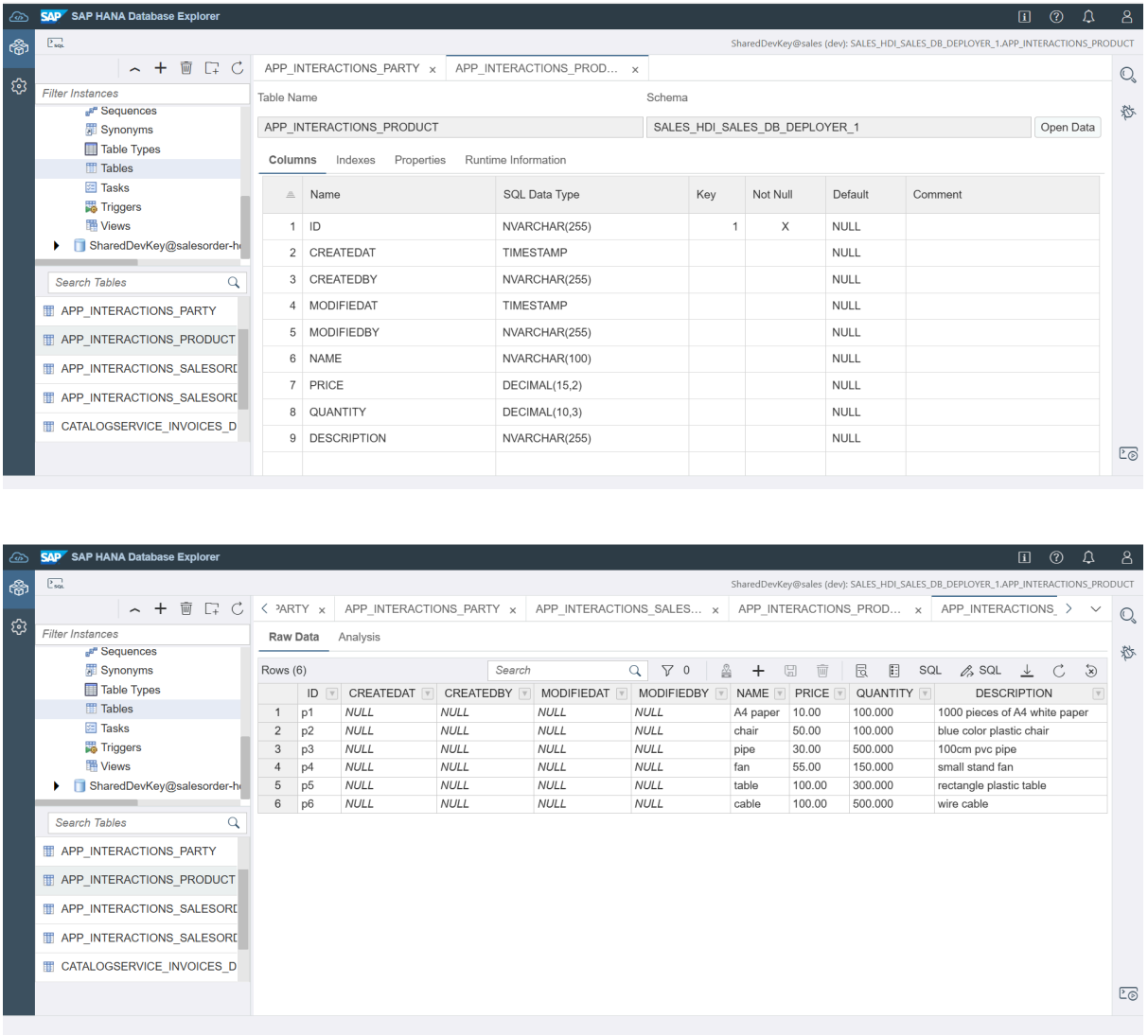
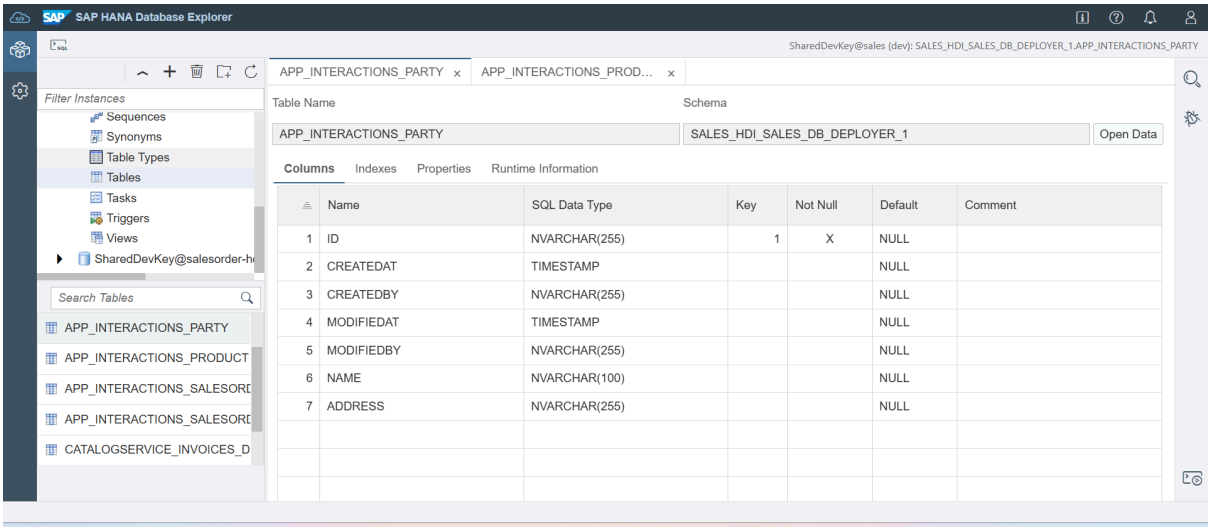


Figure 5.2.1: create product database

Figure 5.2.1 shows the database created for the product entity. In this product entity it will consist of product information attributes like product ID, product name, product price, product quantity, product description, created, create by, modifydat and modify attributes. So that all product data will be stored in this database.



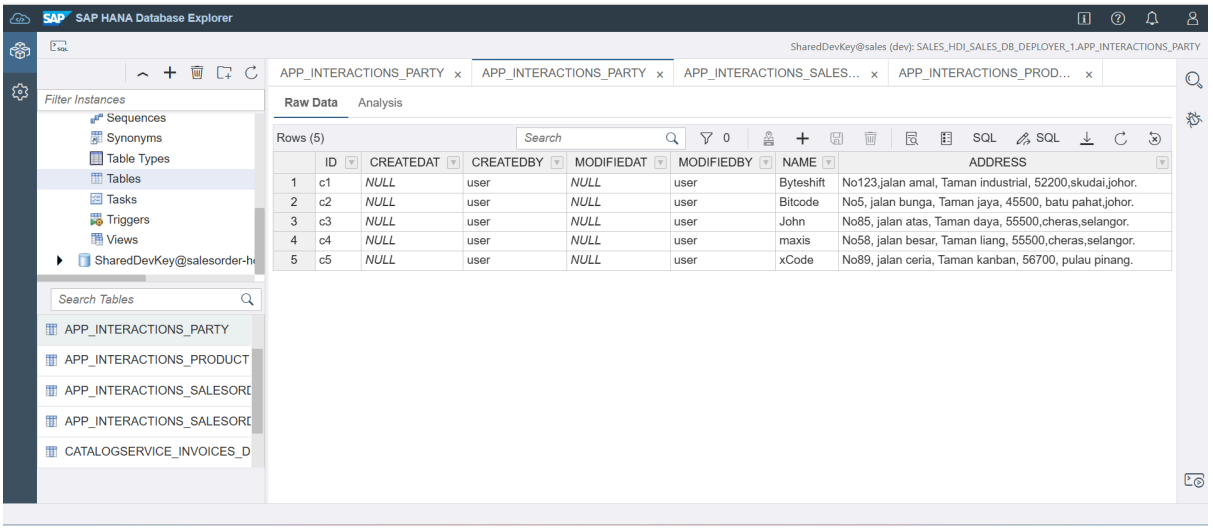


Figure 5.2.2: create party(buyer) database

Figure 5.2.2 shows the database created for the party entity. In the party entity, it will consist of the buyer information attribute like party ID, party name, party address, created at, createby, modified and modified by attributes. So, every party's data will be stored in this database.

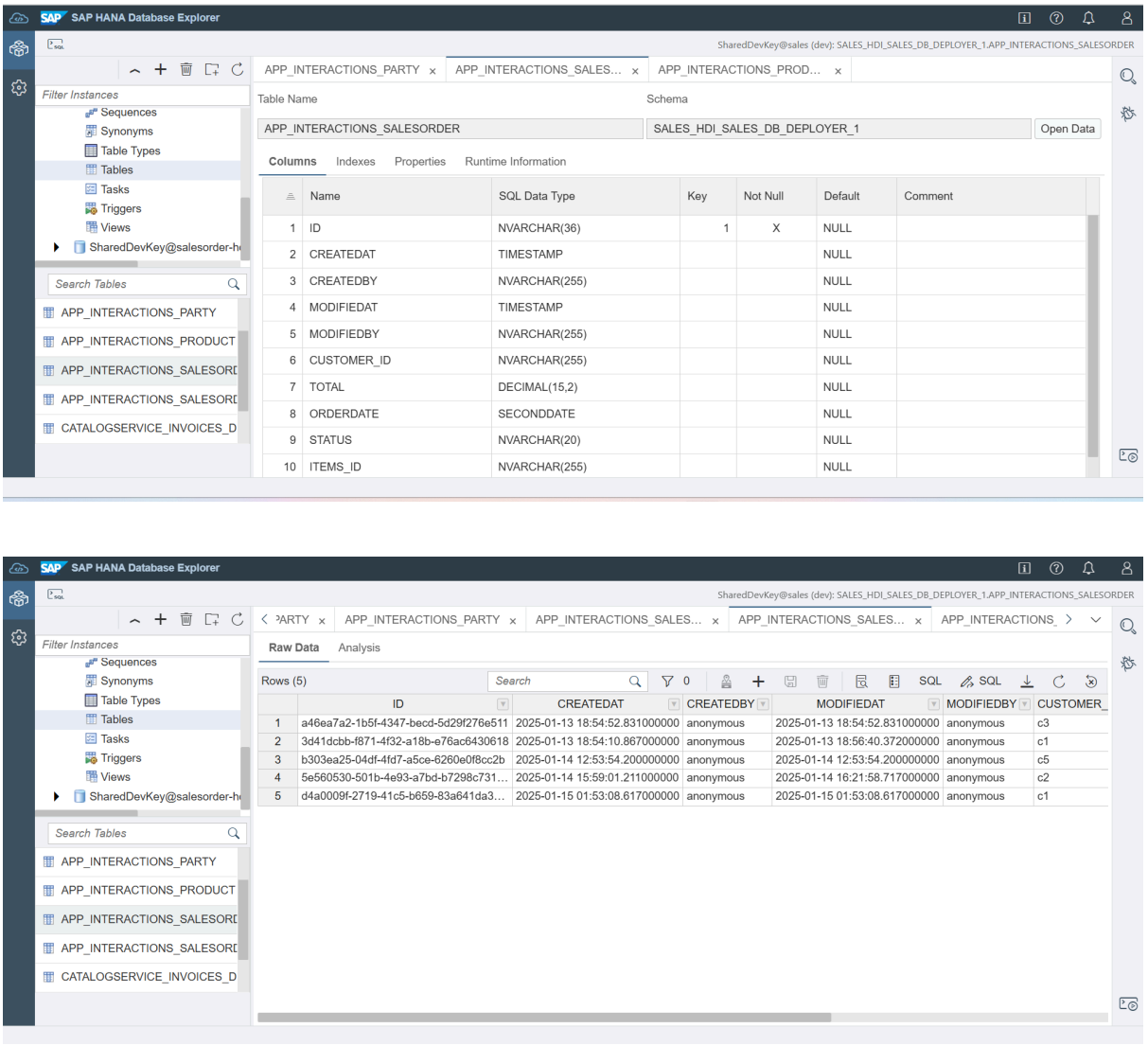


Figure 5.2.3: create sales order database

Figure 5.2.3 shows the database created for the sales order entity. It consists of the sales order information attributes like sales order ID, customer ID for sales order, total of the sales order, status of the sales order, item ID for sales order, the quantity of the item for the sales order attributes and order date for the sales order. So, data on every sales order will be stored in this database.

Quotation Database

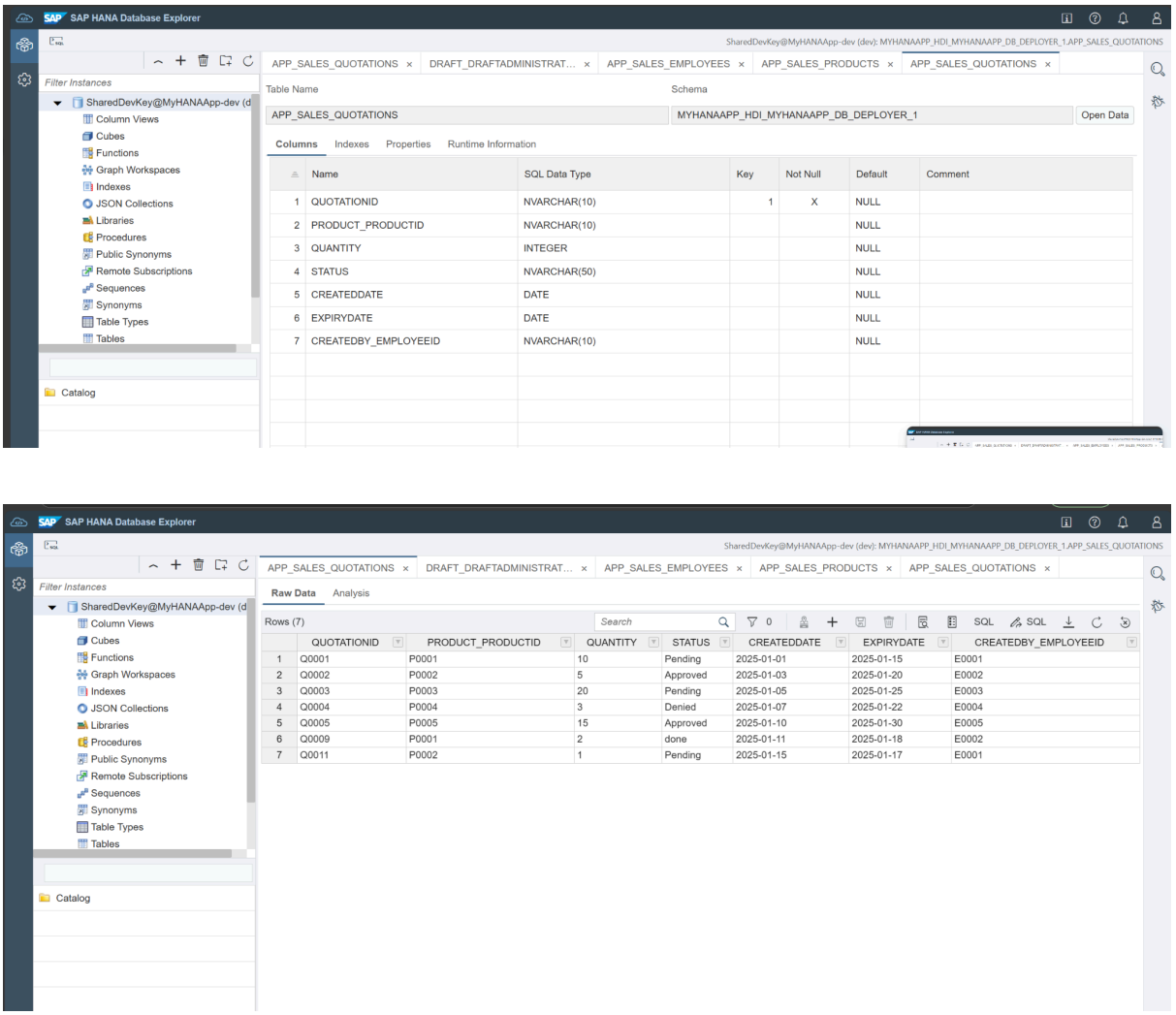


Figure 5.2.4

Figure 5.2.4 shows the database created for the quotation entity. In this quotation entity, it consists of quotation information attributes such as quotation ID, product ID, quantity, status, creation date, expiry date, and the employee ID of the person who created the quotation. Each attribute plays a significant role in storing and managing all the essential details related to sales quotations in this database.

This ensures that every piece of information regarding a sales quotation, including its association with products, employees, and its validity period, is efficiently stored and accessible.

product database

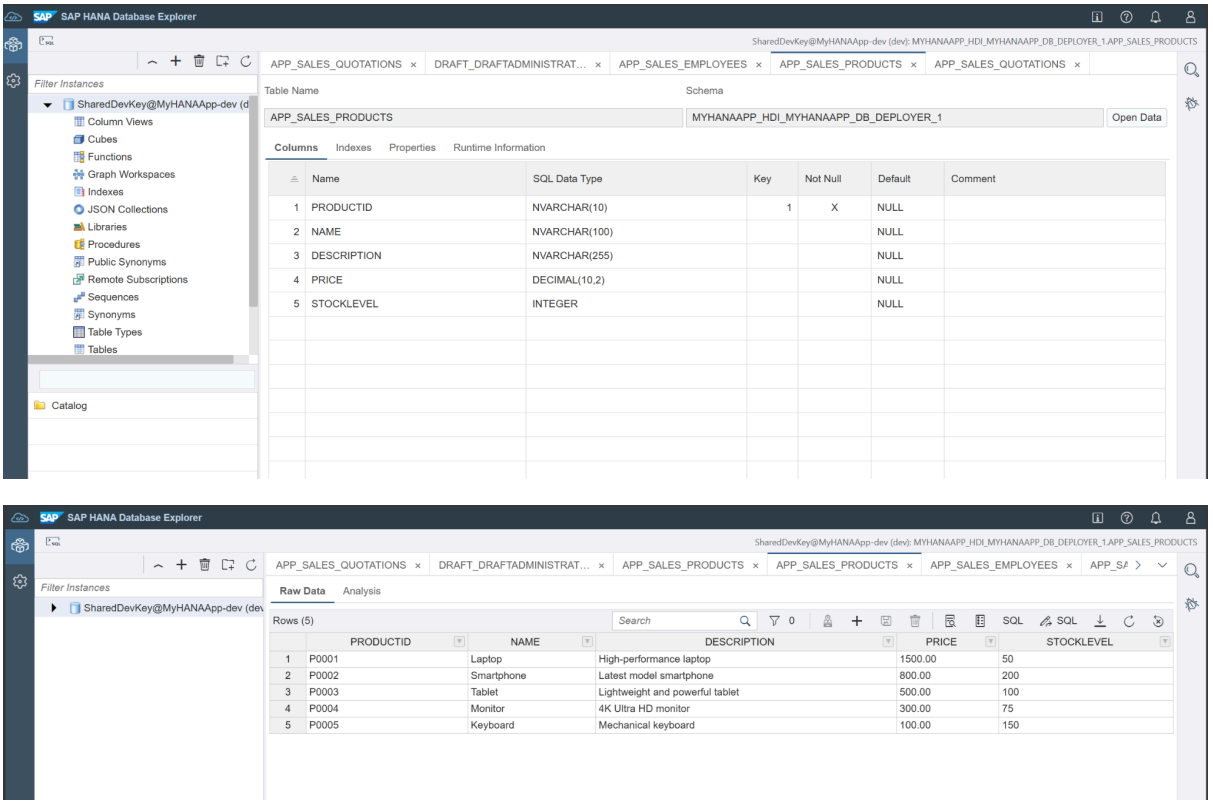


Figure 5.2.4

Figure 5.2.4 shows the database created for the product entity. In this product entity, it consists of product information attributes such as product ID, product name, product description, product price, and stock level. Each attribute is essential to store the necessary data for managing product details in this database.

This ensures that every piece of information regarding a product, including its unique identifier, description, pricing, and inventory levels, is efficiently stored and accessible for business processes.

Employee database

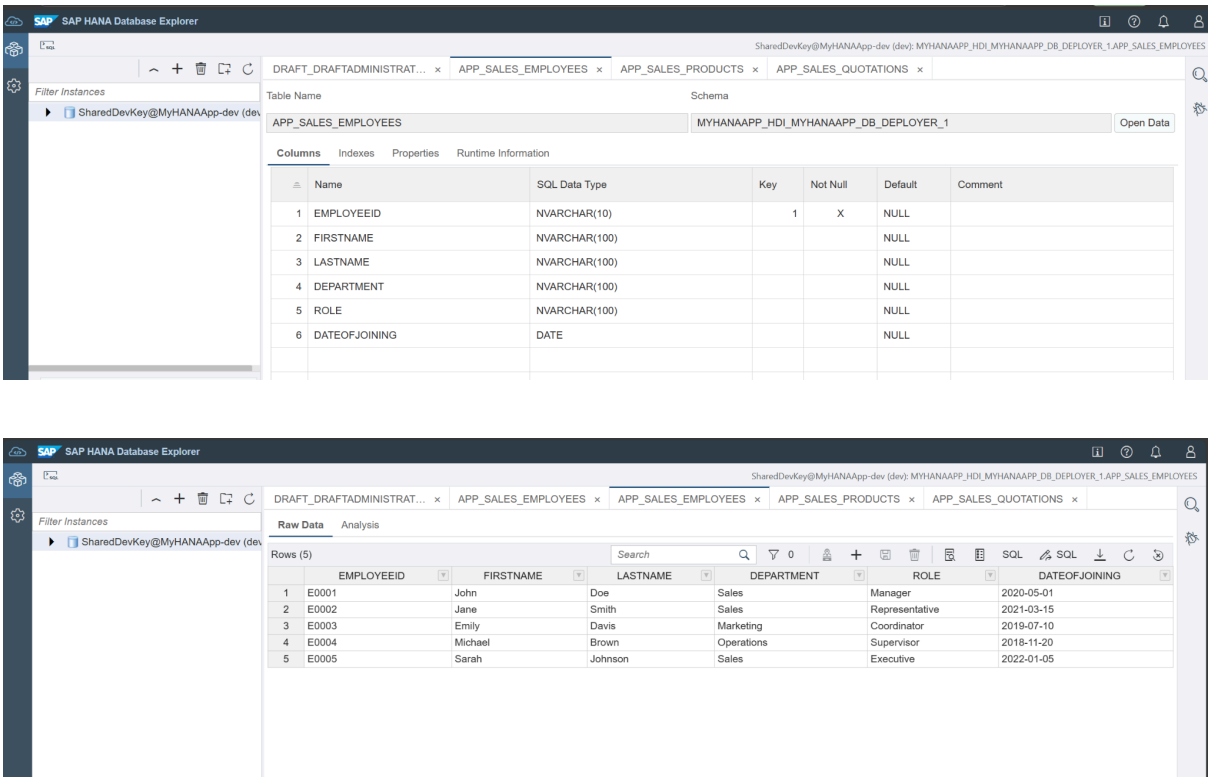


Figure 5.2.5

Figure 5.2.5 shows the database created for the employee entity. In this employee entity, it consists of employee information attributes such as employee ID, first name, last name, department, role, and date of joining. Each attribute ensures that the data regarding the employees is efficiently stored in this database.

This database is essential for managing employee details, tracking their departmental roles, and maintaining a record of their joining date, which is crucial for organisational processes.

Draft Database

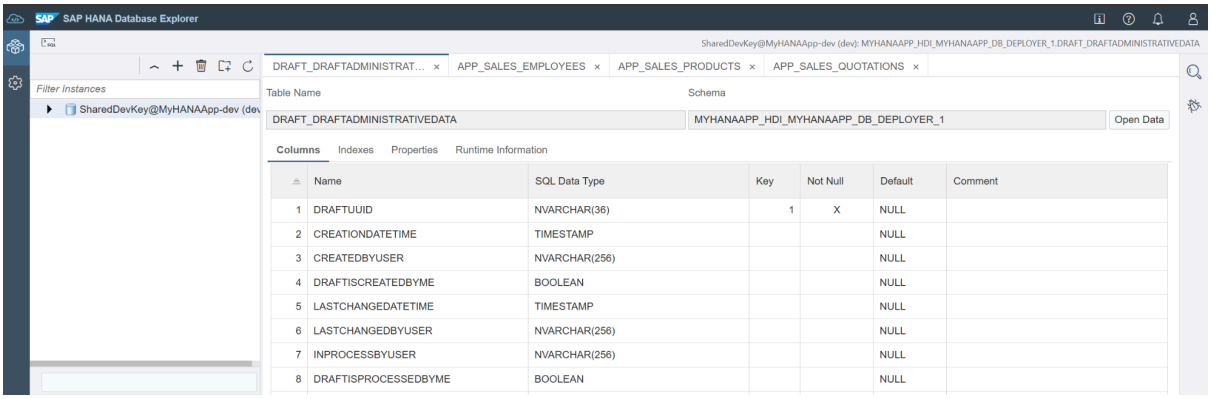
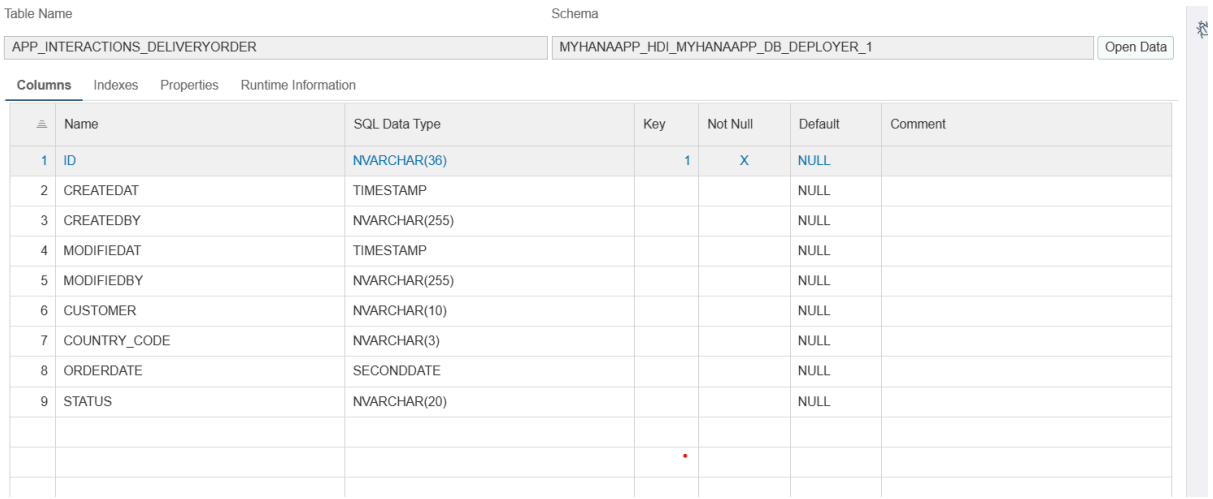


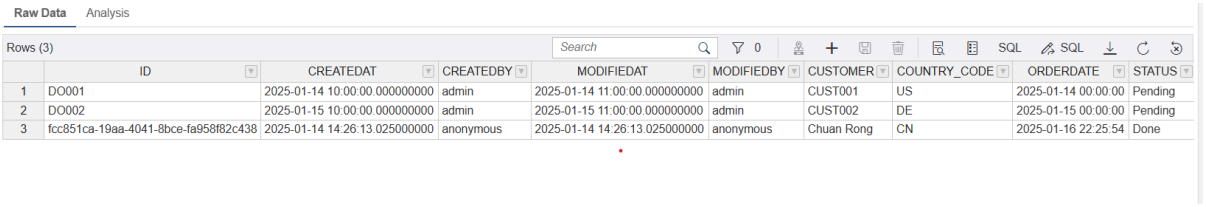
Figure 5.2.5

Figure 5.2.5 shows the database created for the draft administrative data entity. In this draft administrative entity, it consists of attributes such as draft UUID, creation date and time, created by user, draft is created by me, last change date and time, last changed by user, in process by user, and draft is processed by me.

Each attribute ensures that the system efficiently tracks and manages draft-related administrative data. This database is crucial for logging user actions, monitoring modifications, and maintaining the state of draft processes for better system functionality and accountability.

Post Good Issue Database - (Lee Yik Hong)





Raw Data | Analysis

Rows (3)

	ID	CREATEDAT	CREATEDBY	MODIFIEDAT	MODIFIEDBY	CUSTOMER	COUNTRY_CODE	ORDERDATE	STATUS
1	DO001	2025-01-14 10:00:00.000000000	admin	2025-01-14 11:00:00.000000000	admin	CUST001	US	2025-01-14 00:00:00	Pending
2	DO002	2025-01-15 10:00:00.000000000	admin	2025-01-15 11:00:00.000000000	admin	CUST002	DE	2025-01-15 00:00:00	Pending
3	fcc851ca-19aa-4041-8bce-fa958f82c438	2025-01-14 14:26:13.025000000	anonymous	2025-01-14 14:26:13.025000000	anonymous	Chuan Rong	CN	2025-01-16 22:25:54	Done

Figure 5.2.6 DELIVERYORDER Database

The **DELIVERYORDER** database manages delivery orders with key fields such as `ID` (unique identifier), `CUSTOMER`, `COUNTRY_CODE`, `ORDERDATE`, and `STATUS` (e.g., "Pending" or "Done"). Audit fields like `CREATEDAT`, `MODIFIEDAT`, `CREATEDBY`, and `MODIFIEDBY` track record creation and modifications.

The raw data shows examples such as order `DO001` (customer `CUST001`, US, "Pending") and a completed order for `Chuan Rong` (CN). This structure ensures efficient order tracking with detailed auditability.

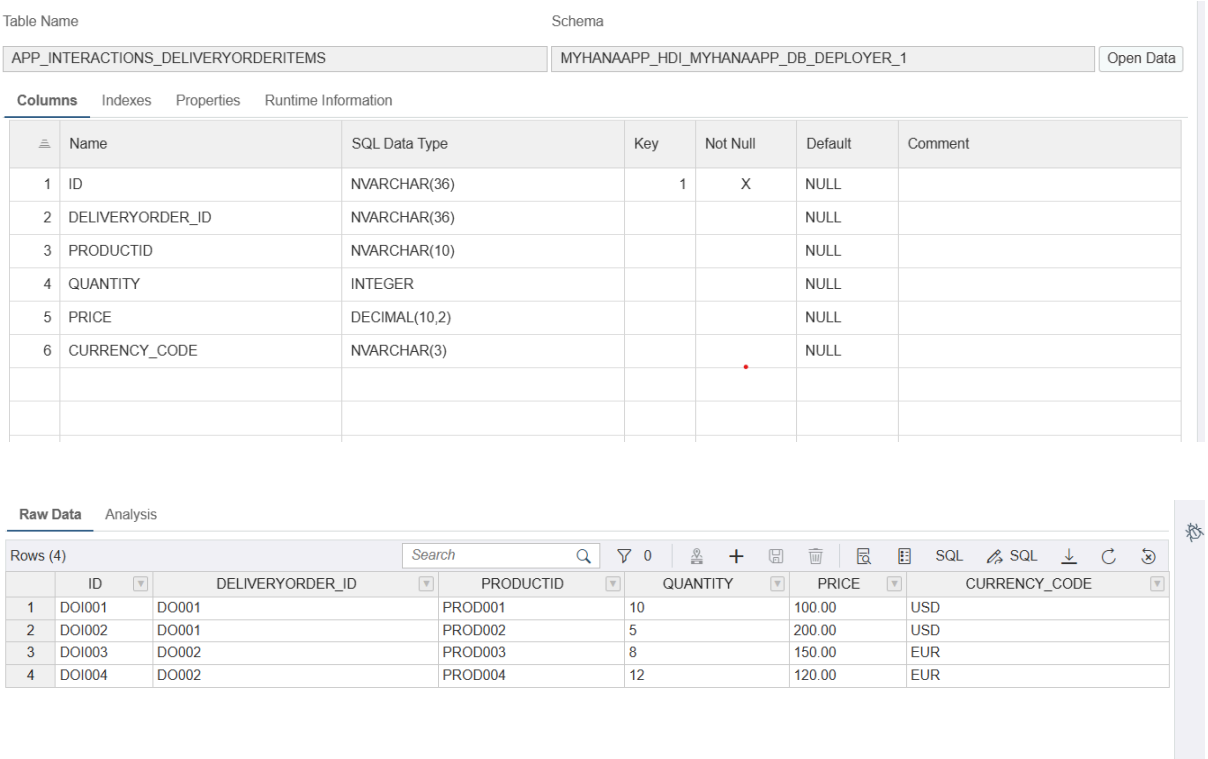


Table Name: APP_INTERACTIONS_DELIVERYORDERITEMS

Schema: MYHANAAPP_HDI_MYHANAAPP_DB_DEPLOYER_1

Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(36)	1	X	NULL	
2	DELIVERYORDER_ID	NVARCHAR(36)			NULL	
3	PRODUCTID	NVARCHAR(10)			NULL	
4	QUANTITY	INTEGER			NULL	
5	PRICE	DECIMAL(10,2)			NULL	
6	CURRENCY_CODE	NVARCHAR(3)			NULL	

Raw Data | Analysis

Rows (4)

	ID	DELIVERYORDER_ID	PRODUCTID	QUANTITY	PRICE	CURRENCY_CODE
1	DOI001	DO001	PROD001	10	100.00	USD
2	DOI002	DO001	PROD002	5	200.00	USD
3	DOI003	DO002	PROD003	8	150.00	EUR
4	DOI004	DO002	PROD004	12	120.00	EUR

Figure 5.2.7 DELIVERYORDERITEMS Database

The **DELIVERYORDERITEMS** database tracks items associated with delivery orders. Key fields include `ID` (unique item identifier), `DELIVERYORDER_ID` (link to the delivery order), `PRODUCTID` (product identifier), `QUANTITY` (number of units), `PRICE` (unit price), and `CURRENCY_CODE` (currency type).

The raw data shows examples such as item `DOI001`, linked to order `DO001`, for product `PROD001` with a quantity of 10 and a price of 100 USD. Another example is item `DOI003` linked to order `DO002` for product `PROD003` priced in EUR. This database ensures efficient tracking of delivery order items and their details.

Table Name: APP_INTERACTIONS_INVOICE
Schema: MYHANAAPP_HDI_MYHANAAPP_DB_DEPLOYER_1
Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(36)	1	X	NULL	
2	CREATEDAT	TIMESTAMP			NULL	
3	CREATEDBY	NVARCHAR(255)			NULL	
4	MODIFIEDAT	TIMESTAMP			NULL	
5	MODIFIEDBY	NVARCHAR(255)			NULL	
6	DELIVERYORDER_ID	NVARCHAR(36)			NULL	
7	CUSTOMER	NVARCHAR(10)			NULL	
8	COUNTRY_CODE	NVARCHAR(3)			NULL	
9	INVOICEDATE	SECONDDATE			NULL	
10	TOTALAMOUNT	DECIMAL(10,2)			NULL	
11	CURRENCY_CODE	NVARCHAR(3)			NULL	

Raw Data | Analysis

Rows (6)

	ID	CREATEDAT	CREATEDBY	MODIFIEDAT	MODIFIEDBY	DELIVERYORDER_ID	CUSTOMER	COUNTRY_CODE	INVOICEDATE	TOTALAMOUNT	CURREN
1		2025-01-16 09:00:00.000000000	admin	2025-01-16 10:00:00.000000000	admin	DO001	CUST001	US	2025-01-16 00:00:00	1500.00	USD
2		2025-01-17 09:00:00.000000000	admin	2025-01-17 10:00:00.000000000	admin	DO002	CUST002	DE	2025-01-17 00:00:00	1860.00	EUR
3	08-ad15-a2fe21c84164	2025-01-15 00:45:28.165000000	anonymous	2025-01-15 00:45:28.165000000	anonymous	NULL	YIkHONg	AG	2025-01-06 08:44:59	25000.00	USD
4	51-bd00-54c7b42100be	2025-01-15 00:48:55.914000000	anonymous	2025-01-15 00:48:55.914000000	anonymous	NULL	CS	AL	2025-01-09 08:48:40	2000.00	INR
5	d6-a4c5-26ee3580d56f	2025-01-15 02:10:03.575000000	anonymous	2025-01-15 02:10:03.575000000	anonymous	NULL	CUST002	AG	2025-01-14 10:09:46	200.00	USD
6	e-a816-4b0dc06bfeac	2025-01-22 07:15:49.562000000	anonymous	2025-01-22 07:15:49.562000000	anonymous	NULL	Chuan Shen	US	2025-01-14 15:15:00	500.00	USD

Figure 5.2.8 INVOICE Database

The **INVOICE** database tracks invoice records with key fields such as `ID` (unique identifier), `DELIVERYORDER_ID` (link to the related delivery order), `CUSTOMER` (customer identifier),

COUNTRY_CODE (region), INVOICEDATE (date of invoice), TOTALAMOUNT (invoice total), and CURRENCY_CODE (currency type). Audit fields like CREATEDAT, CREATEDBY, MODIFIEDAT, and MODIFIEDBY ensure comprehensive record tracking and traceability.

The raw data includes examples such as invoice DOI001 linked to delivery order DO001 for customer CUST001 (US), with a total amount of 1500 USD, and another invoice for CUST002 (DE), with a total of 1900 EUR. This database ensures seamless tracking of invoice details and their linkage to delivery orders for efficient management.

Table Name: APP_INTERACTIONS_INVOICEITEMS

Schema: MYHANAAPP_HDI_MYHANAAPP_DB_DEPLOYER_1

Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(36)	1	X	NULL	
2	INVOICE_ID	NVARCHAR(36)			NULL	
3	PRODUCTID	NVARCHAR(10)			NULL	
4	QUANTITY	INTEGER			NULL	
5	UNITPRICE	DECIMAL(10,2)			NULL	
6	TOTALPRICE	DECIMAL(10,2)			NULL	
7	CURRENCY_CODE	NVARCHAR(3)			NULL	

Raw Data | Analysis

Rows (4)

	ID	INVOICE_ID	PRODUCTID	QUANTITY	UNITPRICE	TOTALPRICE	CURRENCY_CODE
1	II001	INV001	PROD001	10	100.00	1000.00	USD
2	II002	INV001	PROD002	5	200.00	500.00	USD
3	II003	INV002	PROD003	8	150.00	1200.00	EUR
4	II004	INV002	PROD004	12	120.00	720.00	EUR

Figure 5.2.9 INVOICEITEMS Database

The **INVOICEITEMS** database tracks items associated with invoices. Key fields include `ID` (unique item identifier), `INVOICE_ID` (link to the related invoice), `PRODUCTID` (product identifier), `QUANTITY` (number of units), `UNITPRICE` (price per unit), `TOTALPRICE` (calculated total), and `CURRENCY_CODE` (currency type).

The raw data includes examples such as item `II001` linked to invoice `INV001`, representing product `PROD001` with a quantity of 10, unit price of 100 USD, and a total price of 1000 USD. Another example is item `II004` for invoice `INV002`, product `PROD004`, with a quantity of 12 and a total of 720 EUR. This database ensures efficient tracking of individual invoice items and their financial details.

Pick Outbound Delivery Database - (Tiew Chuan Shen)

Table Name: APP_INTERACTIONS_OUTBOUNDDELIVERY

Schema: NEWAKMAJU_HDI_NEWAKMAJU_DB_DEPLOYER_2

Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(36)	1	X	NULL	
2	CREATEDAT	TIMESTAMP			NULL	
3	CREATEDBY	NVARCHAR(255)			NULL	
4	MODIFIEDAT	TIMESTAMP			NULL	
5	MODIFIEDBY	NVARCHAR(255)			NULL	
6	DELIVERYNUMBER	NVARCHAR(10)			NULL	
7	DELIVERYDATE	SECONDDATE			NULL	
8	QUANTITY	DECIMAL(10,2)			NULL	
9	PRODUCT	NVARCHAR(100)			NULL	
10	STATUS	NVARCHAR(20)			PE	
11	ITEMS_ID	NVARCHAR(255)			NULL	
12	SELECTEDUNITS_ID	NVARCHAR(255)			NULL	

Raw Data | Analysis

Rows (9) Search 0 SQL SQL

	ID	CREATEDAT	CREATEDBY	MODIFIEDAT	MODIFIEDBY	DELIVERYNUMBER	D
1	6f302297-2a06-4a4a-b62b-378495aa4e67	2025-01-14 17:57:07.952000000	anonymous	2025-01-14 17:57:07.952000000	anonymous	DO003	20
2	7c8ae2ce-32be-4cc8-b0af-b94de8a81032	2025-01-14 18:07:27.198000000	anonymous	2025-01-14 18:07:27.198000000	anonymous	DO004	20
3	5eeb85bb-9c8f-4ba8-8dbc-982e49668d96	2025-01-15 01:37:13.755000000	anonymous	2025-01-15 01:45:36.765000000	anonymous	DO005	20
4	d8ddfc37-15af-45a1-919d-6ca39332899d	2025-01-15 01:53:20.337000000	anonymous	2025-01-15 01:53:37.067000000	anonymous	DO001	20
5	36eac22b-cdac-43c9-9e53-92e3b215f893	2025-01-15 01:46:18.744000000	anonymous	2025-01-15 01:56:16.142000000	anonymous	DO006	20
6	6c0a7f8e-b02c-4adb-83b2-868b59a8f239	2025-01-14 17:59:22.506000000	anonymous	2025-01-15 02:01:52.229000000	anonymous	DO002	20
7	1dcf2d0b-670c-4602-a8e8-34d68cae744a	2025-01-15 02:03:07.830000000	anonymous	2025-01-15 02:03:07.830000000	anonymous	DO007	20
8	38f27559-b498-4b0b-9633-b1aeb94b2b2d	2025-01-15 02:07:00.351000000	anonymous	2025-01-15 02:07:00.351000000	anonymous	DO008	20
9	acc69bb8-e274-4e07-9d06-ee7350d8f067	2025-01-21 10:17:21.268000000	anonymous	2025-01-21 10:17:21.268000000	anonymous	DO009	20

Figure 5.2.10 outbound delivery database

Figure 5.2.10 shows the database created for the outbound delivery database. The database consists of delivery order information attributes such as delivery order ID, delivery number, delivery date, status and product id. Each attribute ensures that the data regarding the delivery order is efficiently stored in this database. This database is essential for managing delivery order details, tracking their delivery status, and maintaining a record of their delivery date, which is crucial for organisational processes.

Table Name: APP_INTERACTIONS_DELIVERYUNITS

Schema: NEWAKMAJU_HDI_NEWAKMAJU_DB_DEPLOYER_2

Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(255)	1	X	NULL	
2	DELIVERY_ID	NVARCHAR(36)			NULL	
3	PRODUCT	NVARCHAR(50)			NULL	
4	QUANTITY	DECIMAL(10,3)			NULL	
5	UNIT	NVARCHAR(100)			NULL	
6	PRICE	DECIMAL(10,2)			NULL	
7	DESCRIPTION	NVARCHAR(500)			NULL	
8	APPROVETIME	SECONDDATE			NULL	
9	PICKED	BOOLEAN			0	
10	CREATEDAT	TIMESTAMP			NULL	
11	CREATEDBY	NVARCHAR(255)			NULL	
12	MODIFIEDAT	TIMESTAMP			NULL	
13	MODIFIEDBY	NVARCHAR(255)			NULL	

Raw Data | Analysis

Rows (6) Search 0 SQL SQL

	ID	DELIVERY_ID	PRODUCT	QUANTITY	UNIT	PRICE	DESCRIPTION	APPROVETIME	PICKED	CREATEDAT	CREATEDBY	MODI
1	002	D002	Aluminium	10.000	40x50cm	20.00	Silver	NULL	false	NULL	NULL	NULL
2	003	D003	Sand cube	8.000	m^2	10.00	Cubic square	NULL	false	NULL	NULL	NULL
3	004	D003	Alloy	10.000	50x50cm	50.00	Metal	NULL	false	NULL	NULL	NULL
4	005	D004	Wood	5.000	km	100.00	Natural	NULL	false	NULL	NULL	NULL
5	001	D001	A4 paper	5.000	cm	50.00	white paper	NULL	false	NULL	NULL	NULL
6	006	D005	Plastic	20.000	100x150cm	100.00	non-renewable	NULL	false	NULL	NULL	NULL

Figure 5.2.11 outbound delivery database

Figure 5.2.11 shows the database created for the product entity. In this product entity, it consists of product information attributes such as product ID, delivery ID, quantity, unit, product name, price, and the description of the product. Each attribute plays a significant role in storing and managing all the important details related to the product in this database. This ensures that every piece of information regarding a product is efficiently stored and accessible.

Create Outbound Delivery Database - (Nik Zulaikhaa)

Table Name: MY_OUTBOUNDDELIVERY | Schema: MYHANAAPP_HDI_MYHANAAPP_DB_DEPLOYER_4 | Open Data

Columns | Indexes | Properties | Runtime Information

	Name	SQL Data Type	Key	Not Null	Default	Comment
1	ID	NVARCHAR(36)	1	X	NULL	
2	CREATEDAT	TIMESTAMP			NULL	
3	CREATEDBY	NVARCHAR(255)			NULL	
4	MODIFIEDAT	TIMESTAMP			NULL	
5	MODIFIEDBY	NVARCHAR(255)			NULL	
6	SALESORDERID	NVARCHAR(10)			NULL	
7	DELIVERYORDERID	NVARCHAR(10)			NULL	
8	ORDERDATE	SECONDDATE			NULL	
9	CUSTOMERNAME	NVARCHAR(1024)			NULL	
10	CUSTOMERADDRESS	NVARCHAR(1024)			NULL	
11	ITEMDESCRIPTION	NVARCHAR(1024)			NULL	
12	QUANTITY	INTEGER			NULL	
13	PRICE	DECIMAL(10,2)			NULL	

Figure 5.2.12 Outbound Delivery Database

Rows (6) | Search | SQL | SQL

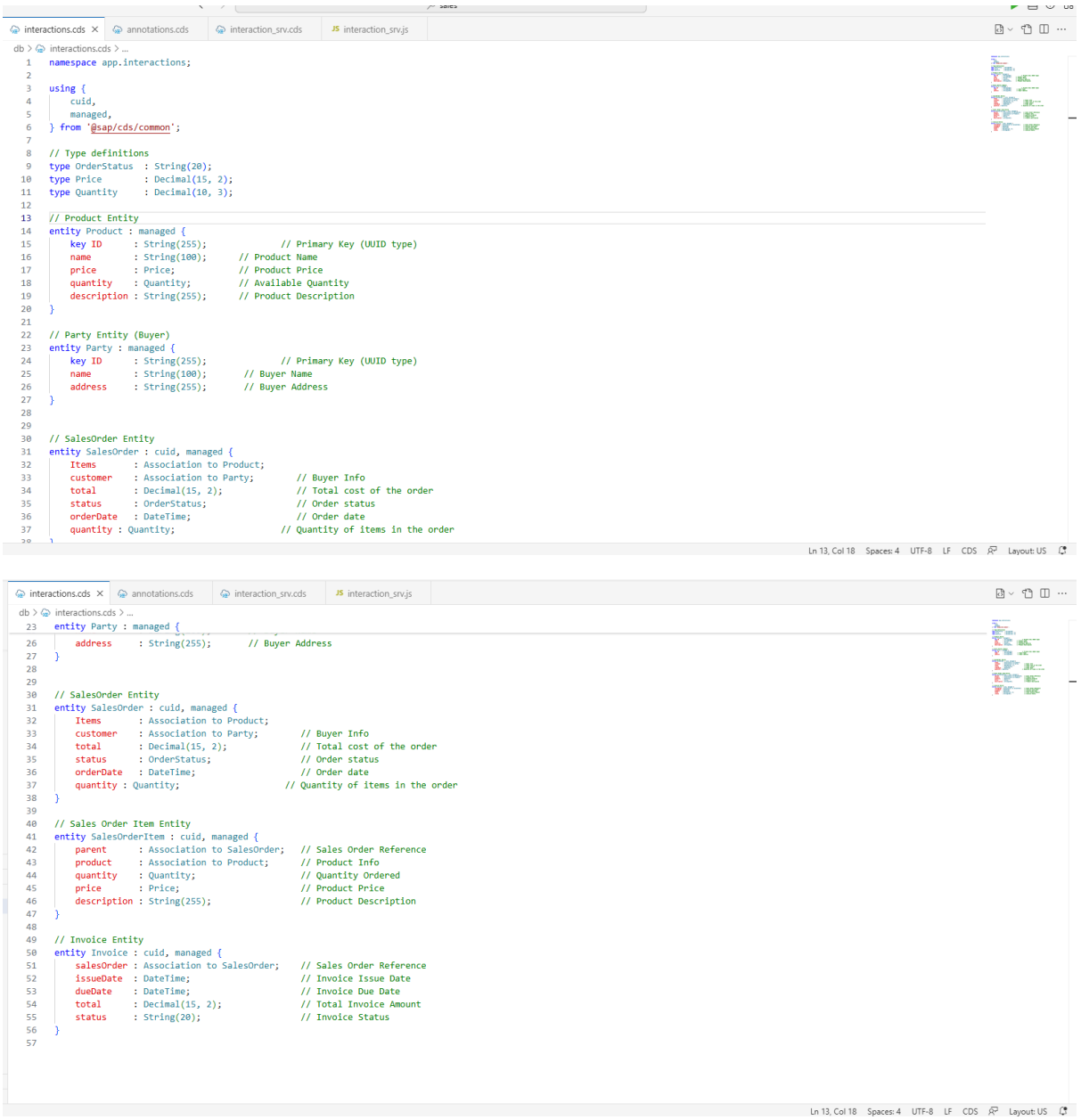
	ID	CREATE…	CREA…	MODIFIE…	MOD…	SALES…	DELIVE…	ORDERD…	CUSTOMERNAM…	L…	ITEMDESCRI…	QUANTITY	PRICE
1	bb24e…	2025-01-…	anony…	2025-01…	anon…	S000002	D000002	2025-01-…	John Doe	N.	Wood	3	4.99
2	8c1f8b…	2025-01-…	anony…	2025-01…	anon…	S123456	D123456	2025-01-…	Jane Jacob	N.	Paint	3	4.99
3	f0e523…	2025-01-…	anony…	2025-01…	anon…	S000001	D000001	2025-01-…	John Doe	N.	Screwdriver	2	4.99
4	6d3da…	2025-01-…	anony…	2025-01…	anon…	S000012	D000012	2025-01-…	Jonathan Doe	N.	Nail	1	2.99
5	a5fb90…	2025-01-…	anony…	2025-01…	anon…	S000009	D000009	2025-01-…	John Doe	N.	Nail	2	2.99
6	af47c2…	2025-01-…	anony…	2025-01…	anon…	S000007	D000007	2025-01-…	John Doe	N.	Nail	3	2.99

Figure 5.2.13 Outbound Delivery Data

Figure 5.2.12 shows the database created for the outbound delivery entity. In this outbound delivery entity, it consists of sales order information attributes such sales ID, delivery ID, quantity, product name and price. It also includes information about the customer like the customer's name and customer's address. Every attribute is crucial for storing and managing all key details related to outbound delivery within this database. This guarantees that all information about a customer and their order is effectively stored and easily accessible.

5.3 Coding of the system's main functions/Process (individual)

Create Sales Order coding of the system's main function - (Tiew Chuan Rong)



```
namespace app.interactions;

using {
    cuid,
    managed,
} from '@sap/cds/common';

// Type definitions
type OrderStatus  : String(20);
type Price        : Decimal(15, 2);
type Quantity     : Decimal(10, 3);

// Product Entity
entity Product : managed {
    key ID       : String(255);             // Primary Key (UUID type)
    name         : String(100);      // Product Name
    price        : Price;            // Product Price
    quantity     : Quantity;         // Available Quantity
    description  : String(255);      // Product Description
}

// Party Entity (Buyer)
entity Party : managed {
    key ID       : String(255);             // Primary Key (UUID type)
    name         : String(100);      // Buyer Name
    address      : String(255);      // Buyer Address
}


// SalesOrder Entity
entity SalesOrder : cuid, managed {
    Items       : Association to Product;
    customer    : Association to Party;        // Buyer Info
    total       : Decimal(15, 2);              // Total cost of the order
    status      : OrderStatus;                 // Order status
    orderDate   : DateTime;                    // Order date
    quantity : Quantity;                    // Quantity of items in the order
}
```

```
    address      : String(255);      // Buyer Address
}


// SalesOrder Entity
entity SalesOrder : cuid, managed {
    Items       : Association to Product;
    customer    : Association to Party;        // Buyer Info
    total       : Decimal(15, 2);              // Total cost of the order
    status      : OrderStatus;                 // Order status
    orderDate   : DateTime;                    // Order date
    quantity : Quantity;                    // Quantity of items in the order
}

// Sales Order Item Entity
entity SalesOrderItem : cuid, managed {
    parent      : Association to SalesOrder;   // Sales Order Reference
    product     : Association to Product;      // Product Info
    quantity    : Quantity;                    // Quantity Ordered
    price       : Price;                       // Product Price
    description : String(255);                 // Product Description
}

// Invoice Entity
entity Invoice : cuid, managed {
    salesOrder : Association to SalesOrder;    // Sales Order Reference
    issueDate  : DateTime;                     // Invoice Issue Date
    dueDate    : DateTime;                     // Invoice Due Date
    total      : Decimal(15, 2);               // Total Invoice Amount
    status     : String(20);                   // Invoice Status
}
```

Figure 5.3.1: Coding for interaction.cds (create sales order)

Figure 5.3.1 shows the coding of interaction.cds, and this is the part of creating entities for the database. This is a file used to define the underlying data structure and relationships for entities in the system. It is typically where the data model of the domain is

created and maintained, and it serves as the foundation upon which services, annotations, and business logic are built. This coding acts as a central place for defining database schema for generating database tables. For example, entities like sales orders, products, and parties are created and modeled here.

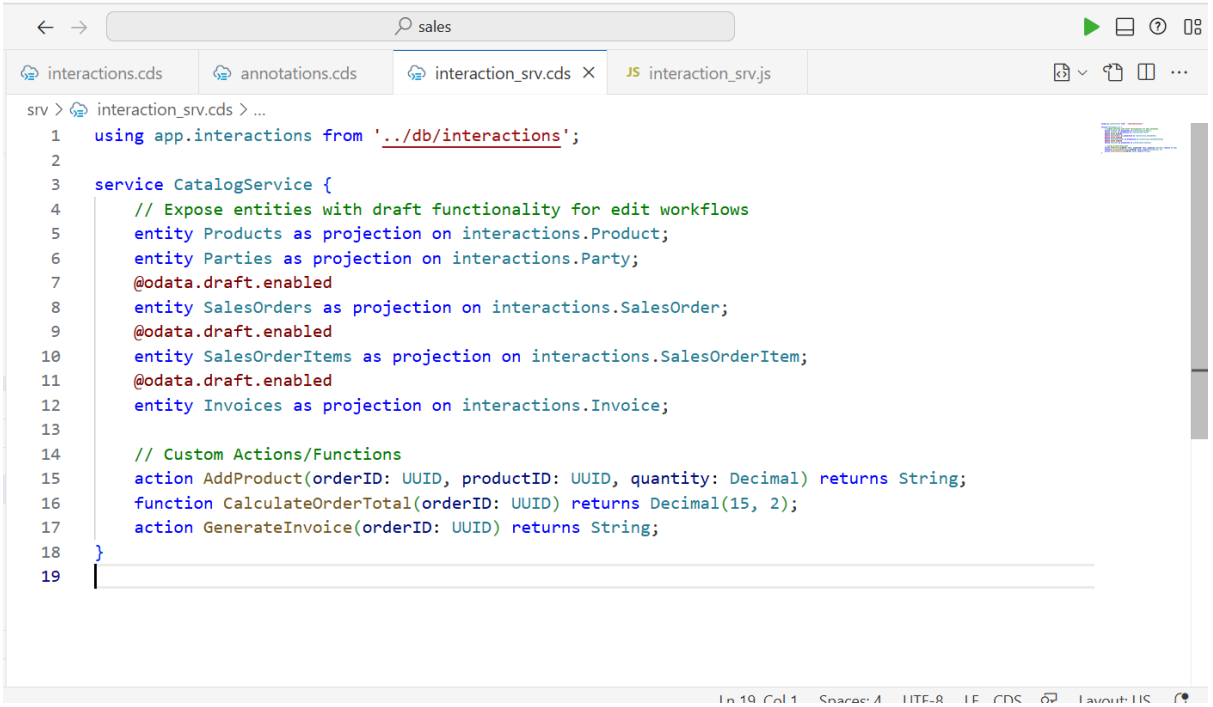
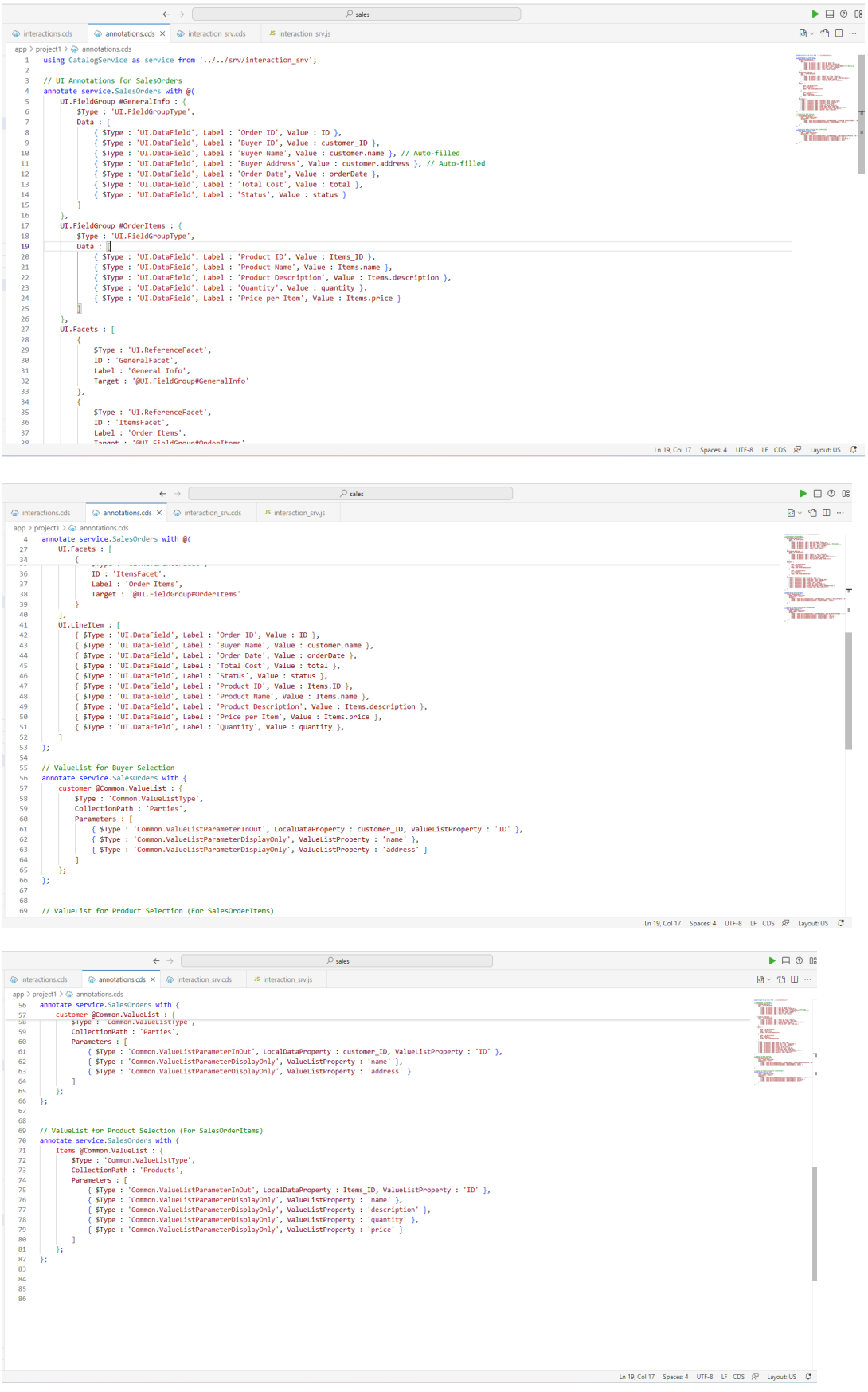


Figure 5.3.2: Coding for interaction_srv.cds (create sales order)

Figure 5.3.2 shows the coding of interaction_srv.cds to make the entities created in the interaction.cds available to the service definition. The catalog service is the name of the service being defined. Next, is the entity projection used to expose specific entities from the interaction namespace to the service. This makes the entities like product, party, and sales orders available to the service definition. The annotation @odata.draft.enabled enables draft functionality for these entities.



```
using CatalogService as service from '../../srv/interaction_srv';

// UI Annotations for SalesOrders
annotate service.SalesOrders with @(
    UI.FieldGroup #GeneralInfo : {
        $Type : 'UI.FieldGroupType',
        Data : [
            { $Type : 'UI.DataField', Label : 'Order ID', Value : ID },
            { $Type : 'UI.DataField', Label : 'Buyer ID', Value : customer_ID },
            { $Type : 'UI.DataField', Label : 'Buyer Name', Value : customer.name }, // Auto-filled
            { $Type : 'UI.DataField', Label : 'Buyer Address', Value : customer.address }, // Auto-filled
            { $Type : 'UI.DataField', Label : 'Order Date', Value : orderDate },
            { $Type : 'UI.DataField', Label : 'Total Cost', Value : total },
            { $Type : 'UI.DataField', Label : 'Status', Value : status }
        ]
    },
    UI.FieldGroup #OrderItems : {
        $Type : 'UI.FieldGroupType',
        Data : [
            { $Type : 'UI.DataField', Label : 'Product ID', Value : Items_ID },
            { $Type : 'UI.DataField', Label : 'Product Name', Value : Items.name },
            { $Type : 'UI.DataField', Label : 'Product Description', Value : Items.description },
            { $Type : 'UI.DataField', Label : 'Quantity', Value : quantity },
            { $Type : 'UI.DataField', Label : 'Price per Item', Value : Items.price }
        ]
    },
    UI.Facets : [
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'GeneralFacet',
            Label : 'General Info',
            Target : '@UI.FieldGroup#GeneralInfo'
        },
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'ItemsFacet',
            Label : 'Order Items',
```



```
annotate service.SalesOrders with @(
    UI.Facets : [
        {
            ID : 'ItemsFacet',
            Label : 'Order Items',
            Target : '@UI.FieldGroup#OrderItems'
        }
    ],
    UI.LineItem : [
        { $Type : 'UI.DataField', Label : 'Order ID', Value : ID },
        { $Type : 'UI.DataField', Label : 'Buyer Name', Value : customer.name },
        { $Type : 'UI.DataField', Label : 'Order Date', Value : orderDate },
        { $Type : 'UI.DataField', Label : 'Total Cost', Value : total },
        { $Type : 'UI.DataField', Label : 'Status', Value : status },
        { $Type : 'UI.DataField', Label : 'Product ID', Value : Items.ID },
        { $Type : 'UI.DataField', Label : 'Product Name', Value : Items.name },
        { $Type : 'UI.DataField', Label : 'Product Description', Value : Items.description },
        { $Type : 'UI.DataField', Label : 'Price per Item', Value : Items.price },
        { $Type : 'UI.DataField', Label : 'Quantity', Value : quantity },
    ]
);

// ValueList for Buyer Selection
annotate service.SalesOrders with {
    customer @Common.ValueList : {
        $Type : 'Common.ValueListType',
        CollectionPath : 'Parties',
        Parameters : [
            { $Type : 'Common.ValueListParameterInOut', LocalDataProperty : customer_ID, ValueListProperty : 'ID' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'name' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'address' }
        ]
    };
);

// ValueList for Product Selection (For SalesOrderItems)
```



```
annotate service.SalesOrders with {
    customer @Common.ValueList : {
        $Type : 'Common.ValueListType',
        CollectionPath : 'Parties',
        Parameters : [
            { $Type : 'Common.ValueListParameterInOut', LocalDataProperty : customer_ID, ValueListProperty : 'ID' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'name' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'address' }
        ]
    };
};

// ValueList for Product Selection (For SalesOrderItems)
annotate service.SalesOrders with {
    Items @Common.ValueList : {
        $Type : 'Common.ValueListType',
        CollectionPath : 'Products',
        Parameters : [
            { $Type : 'Common.ValueListParameterInOut', LocalDataProperty : Items_ID, ValueListProperty : 'ID' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'name' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'description' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'quantity' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'price' }
        ]
    };
};
```

Figure 5.3.3: Coding for annotations (create sales order)

Figure 5.3.3 shows the coding of annotation.cds, where it appears to define the UI annotations for the entities in the SAP CAP model. These annotations enrich the metadata of your entities, enabling SAP Fiori Elements to generate a UI automatically. The using statement imports catalogue service from the interctions_srv.cds file. This allows the annotations to reference entities. The field groups define logical field groupings for display purposes in a UI. Next is the facet, which defines sections or tabs in a UI, linking to field groups or other information. The line item defines fields to display in a table view, like the list of sales orders created. The value list defines the dropdown for fields to select the item from the related entity.

Create quotation - Loo Jia Chang.

annotations.cds | sales.cas

MyHANAApp > db > sales.cds

```
1  namespace app.sales;
2
3  type ProductID   : String(10);
4  type QuotationID : String(10);
5  type Status      : String(50);
6  type Quantity    : Integer;
7  type Description : String(255);
8  type TDate       : Date;
9  type EmployeeID  : String(10);
10
11 entity Products {
12     key productID : ProductID @UI.Identification: [{ type: #AS_PRIMARY, position: 10 }] @title: 'Product ID';
13     name          : String(100) @UI.LineItem: [{ position: 20 }] @title: 'Name';
14     description   : Description @UI.LineItem: [{ position: 30 }] @title: 'Description';
15     price         : Decimal(10,2) @UI.LineItem: [{ position: 40 }] @title: 'Price';
16     stockLevel    : Integer @UI.LineItem: [{ position: 50 }] @title: 'Stock Level';
17 }
18
19 entity Quotations {
20     key quotationID : QuotationID @UI.Identification: [{ type: #AS_PRIMARY, position: 10 }] @title: 'Quotation ID';
21     product         : Association to Products @UI.LineItem: [{ position: 20 }] @title: 'Product';
22     quantity        : Quantity @UI.LineItem: [{ position: 30 }] @UI.DataPoint: { type: #Number } @title: 'Quantity';
23     status          : Status @UI.LineItem: [{ position: 40 }] @title: 'Status'; // E.g., Pending, Approved, Denied
24     createdDate     : TDate @UI.LineItem: [{ position: 50 }] @UI.DataPoint: { type: #Date } @title: 'Created Date';
25     expiryDate      : TDate @UI.LineItem: [{ position: 60 }] @UI.DataPoint: { type: #Date } @title: 'Expiry Date';
26     createdBy       : Association to Employees @UI.LineItem: [{ position: 70 }] @title: 'Created By';
27 }
28
29 entity Employees {
30     key employeeID : EmployeeID @UI.Identification: [{ type: #AS_PRIMARY, position: 10 }] @title: 'Employee ID';
31     firstName      : String(100) @UI.LineItem: [{ position: 20 }] @title: 'First Name';
32     lastName       : String(100) @UI.LineItem: [{ position: 30 }] @title: 'Last Name';
33     department     : String(100) @UI.LineItem: [{ position: 40 }] @title: 'Department';
34     role           : String(100) @UI.LineItem: [{ position: 50 }] @title: 'Role';
35     dateOfJoining  : TDate @UI.LineItem: [{ position: 60 }] @UI.DataPoint: { type: #Date } @title: 'Date of Joining';
36     tasks          : Composition of many Quotations on tasks.createdBy = $self @title: 'Quotations';
37 }
38
```

Figure 5.3.4

Figure 5.3.4 shows the coding of db/sales.cds, and this is the part of creating entities for the database. This file is used to define the underlying data structure and relationships for entities

in the system. It is typically where the data model of the domain is created and maintained, and it serves as the foundation upon which services, annotations, and business logic are built.

This coding acts as a central place for defining the database schema for generating database tables. For example, entities like Products, Quotations, and Employees are created and modeled here, with attributes such as product details, quotation data, and employee information clearly defined to maintain the structure and relationships within the system.

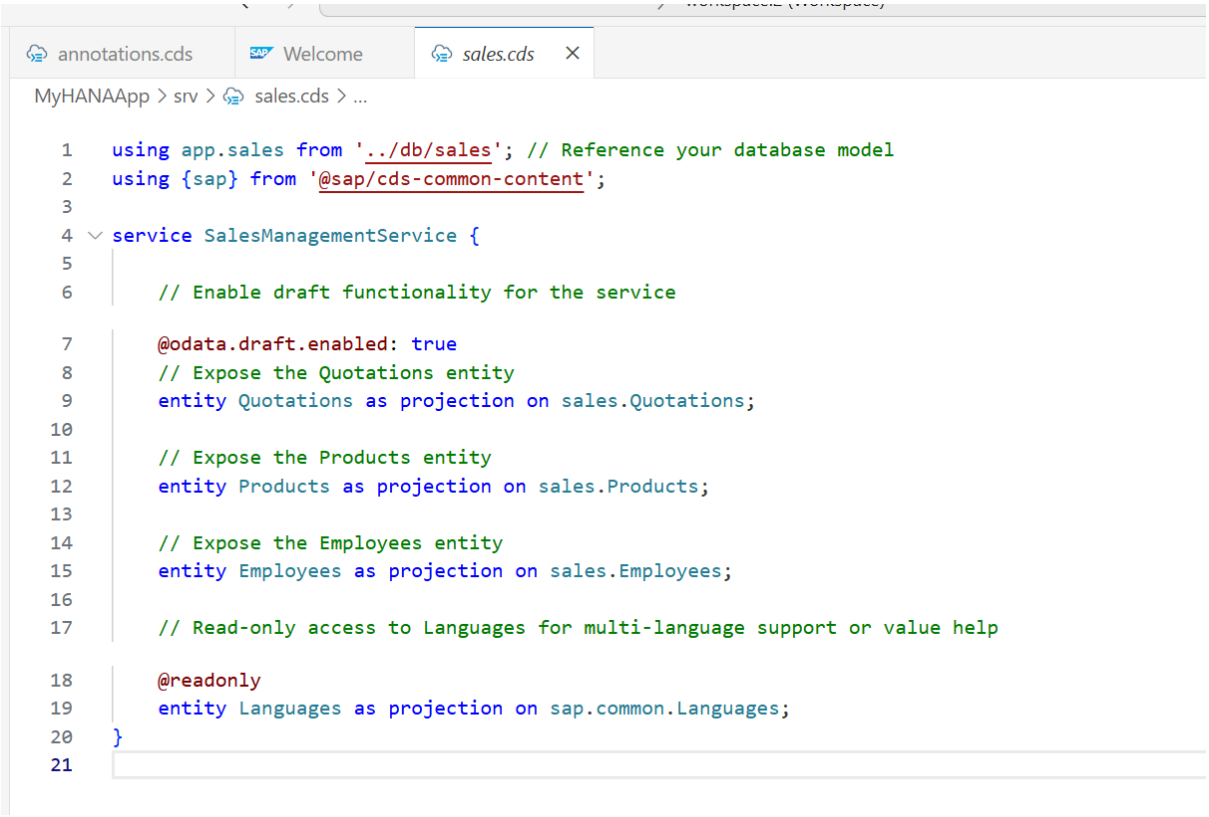


Figure 5.3.5: Coding for db/sales.cds (Sales Management Service)

Figure 5.3.5 shows the coding of sales.cds to make the entities created in the db/sales.cds database definition available to the service definition. The SalesManagementService is the name of the service being defined. Within this service, entity projections are used to expose specific entities from the sales namespace to the service.

This makes entities like Quotations, Products, and Employees available to the service definition. The annotation @odata.draft.enabled enables draft functionality for these entities, allowing them to support draft-based data handling. Additionally, a read-only entity projection for Languages is included to provide multi-language support or value help.

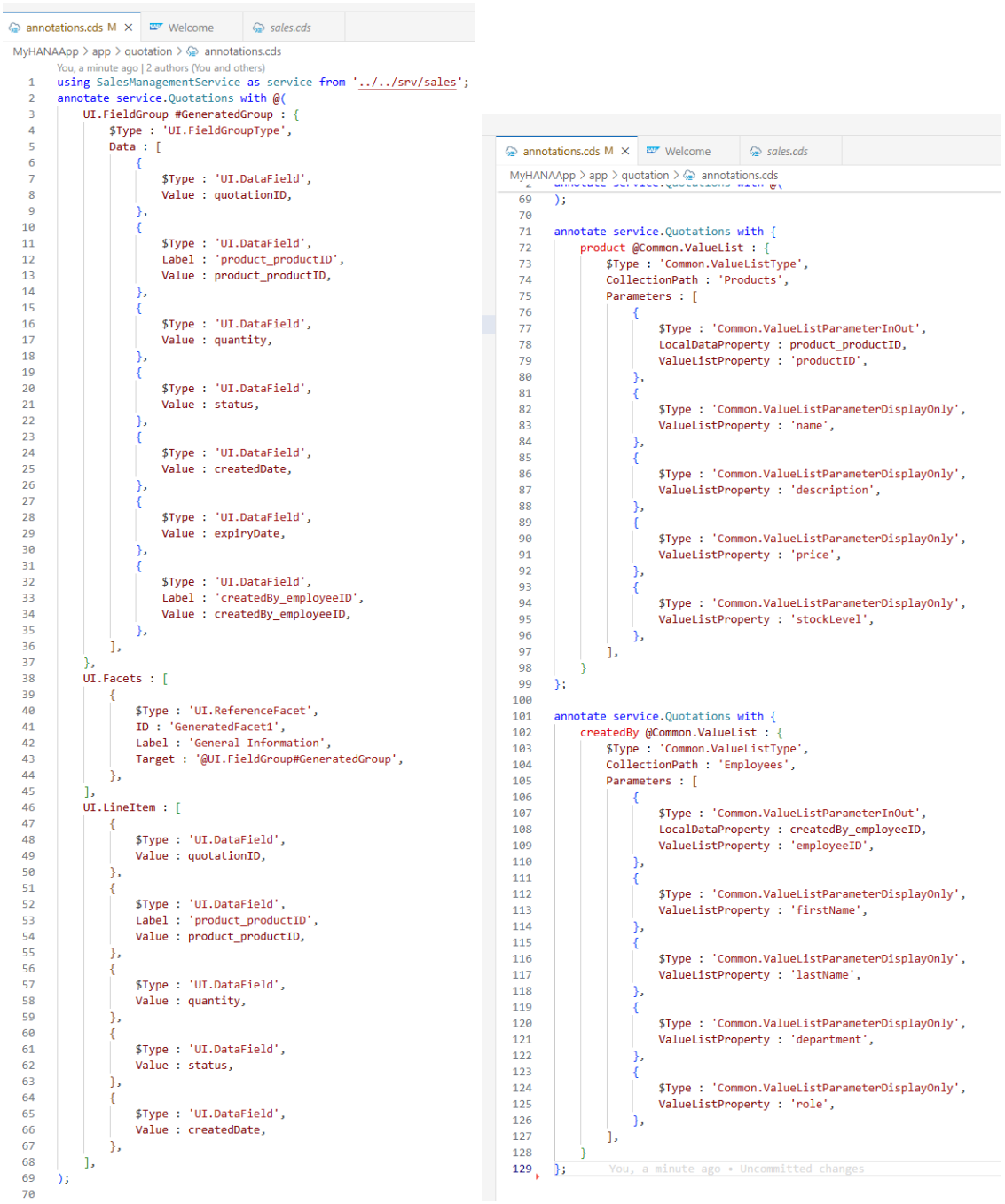


Figure 5.3.6 shows the coding of annotations.cds where it appears to define the UI annotations for the entities in the SAP CAP model. These annotations enrich the metadata of your entities, enabling SAP Fiori Elements to generate a UI automatically. The using statement imports the SalesManagementService from the sales.cds file. This allows the annotations to reference entities.

The Field Groups define logical groupings of fields for display purposes in a UI. For example, fields such as quotationID, product_productID, quantity, status, and createdBy_employeeID are organised for structured display.

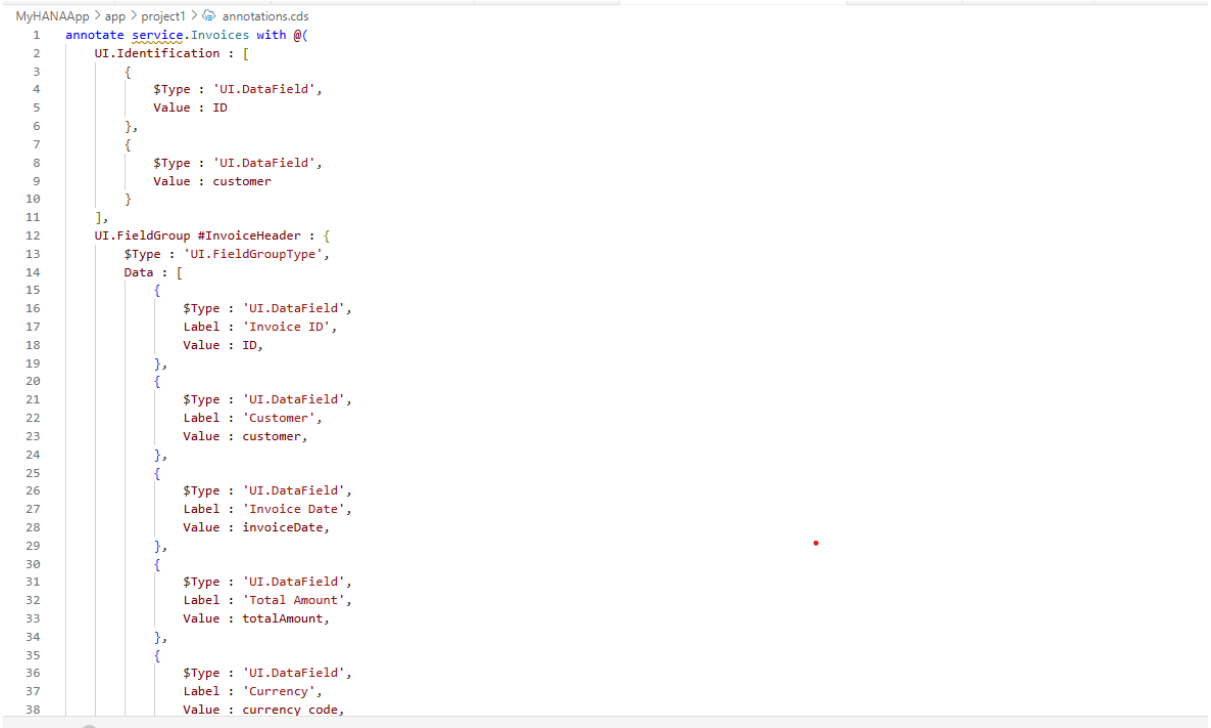
Next, the Facets define sections or tabs in the UI, linking to field groups or other related data and organising the layout into meaningful sections.

The Line Items define fields to display in a table view, such as a list of quotations or products, providing users with a concise overview of relevant data.

Finally, the Value Lists define dropdown menus for fields to select items from related entities. For instance, the product field references the Products entity and is created by referencing the Employees entity, offering contextual value suggestions.

This coding serves as a foundation for building intuitive and efficient UIs in SAP Fiori applications.

Create Post Good Issue coding of the system's main function - (Lee Yik Hong)



```
    UI.Facets : [
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'InvoiceHeader',
            Label : 'Invoice Details',
            Target : '@UI.FieldGroup#InvoiceHeader',
        },
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'LineItems',
            Label : 'Line Items',
            Target : 'items',
        }
    ],
    UI.Create : true, // Enables the "Create" button for the entity
    Common.Label : 'Invoices',
    Common.DraftRoot : { DraftNode : true } // Enables draft handling
);

annotate service.InvoiceItems with @(
    UI.LineItem : [
        {
            $Type : 'UI.DataField',
            Label : 'Product ID',
            Value : productId,
        },
        {
            $Type : 'UI.DataField',
            Label : 'Quantity',
            Value : quantity,
        },
        {
            $Type : 'UI.DataField',
            Label : 'Unit Price',
            Value : unitPrice,
        }
```

Figure 5.3.7: Coding for annotations (Initiate and update inventory)

```
MyHANAApp > app > project2 > annotations.cds
using CatalogService as service from '../../srv/interaction_srv';
annotate service.Invoices with @(
    UI.FieldGroup #GeneratedGroup : {
        $Type : 'UI.FieldGroupType',
        Data : [
            {
                $Type : 'UI.DataField',
                Label : 'customer',
                Value : customer,
            },
            {
                $Type : 'UI.DataField',
                Label : 'country_code',
                Value : country_code,
            },
            {
                $Type : 'UI.DataField',
                Label : 'invoiceDate',
                Value : invoiceDate,
            },
            {
                $Type : 'UI.DataField',
                Label : 'totalAmount',
                Value : totalAmount,
            },
            {
                $Type : 'UI.DataField',
                Label : 'currency_code',
                Value : currency_code,
            },
            {
                $Type : 'UI.DataField',
                Label : 'productId',
                Value : productId,
            },
        ],
    },
    UI.Facets : [
```

MyHANAApp > app > project2 > annotations.cds

```
annotate service.Invoices with @(
    UI.Facets : [
        {
            Target : '@UI.FieldGroup#GeneratedGroup',
        },
    ],
    UI.LineItem : [
        {
            $Type : 'UI.DataField',
            Label : 'customer',
            Value : customer,
        },
        {
            $Type : 'UI.DataField',
            Label : 'country_code',
            Value : country_code,
        },
        {
            $Type : 'UI.DataField',
            Label : 'invoiceDate',
            Value : invoiceDate,
        },
        {
            $Type : 'UI.DataField',
            Label : 'totalAmount',
            Value : totalAmount,
        },
        {
            $Type : 'UI.DataField',
            Label : 'currency_code',
            Value : currency_code,
        },
        {
            $Type : 'UI.DataField',
            Label : 'productId',
            Value : productId,
        },
    ]
```

Figure 5.3.8: Coding for annotations (Create and sent invoice(receipt))

MyHANAApp > db > interactions.cds > Text

```
namespace app.interactions;

using {
    Country,
    Currency,
    cuid,
    managed
} from '@sap/cds/common';

type BusinessKey : String(10);
type Price       : Decimal(10, 2);
type Text        : String(1024);

entity DeliveryOrder : cuid, managed {
    items      : Composition of many DeliveryOrderItems
                     on items.deliveryOrder = $self;
    customer   : BusinessKey;
    country    : Country;
    orderDate  : DateTime;
    status     : String(20); // e.g., "Pending", "Invoiced"
};

entity DeliveryOrderItems : cuid {
    deliveryOrder : Association to DeliveryOrder;
    productId     : BusinessKey;
    quantity      : Integer;
    price         : Price;
    currency      : Currency;
};

entity Invoice : cuid, managed {
    items        : Composition of many InvoiceItems
                       on items.invoice = $self;
    deliveryOrder: Association to DeliveryOrder; // Link to DeliveryOrder
    customer     : BusinessKey;
    country      : Country;
    invoiceDate  : DateTime;
    totalAmount  : Price;
```

Figure 5.3.9: Coding for interaction.cds (Initiate, update inventory and create and sent invoice(receipt))

MyHANAApp > srv > interaction_srv.cds > { } CatalogService > DeliveryOrders

```
using app.interactions from '../db/interactions';
using {sap} from '@sap/cds-common-content';

service CatalogService {

    @odata.draft.enabled: true
    entity DeliveryOrders as projection on interactions.DeliveryOrder {
        ID,         // Include the primary key
        items,      // Include composition
        customer,
        country,
        orderDate,
        status
    };

    entity DeliveryOrderItems as projection on interactions.DeliveryOrderItems {
        ID,             // Include the primary key
        deliveryOrder,  // Include foreign key association to DeliveryOrders
        productId,
        quantity,
        price,
        currency
    };

    @odata.draft.enabled: true
    entity Invoices as projection on interactions.Invoice {
        ID,         // Include the primary key
        items,      // Include composition
        deliveryOrder, // Include foreign key association to DeliveryOrders
        customer,
        country,
        invoiceDate,
        totalAmount,
        currency
    };

    entity InvoiceItems as projection on interactions.InvoiceItems {
        ID,          // Include the primary key
```

Figure 5.3.10: Coding for interaction_srv.cds (Initiate ,update inventory and create and sent invoice(receipt))

Pick Outbound Delivery - (Tiew Chuan Shen)

app > project2 > annotations.cds

You, a few seconds ago | 1 author (You)

```
using CatalogService as service from '../../srv/interaction_srv';

annotate service.OutboundDelivery with @(
    UI.FieldGroup #General : {
        $Type : 'UI.FieldGroupType',
        Data : [
            {
                $Type : 'UI.DataField',
                Label : 'Delivery Number',
                Value : deliveryNumber
            },
            {
                $Type : 'UI.DataField',
                Label : 'Delivery Date',
                Value : deliveryDate
            },
            {
                $Type : 'UI.DataField',
                Label : 'Status',
                Value : status
            }
        ]
    },
    UI.Facets : [
        {
            $Type : 'UI.ReferenceFacet',
            Label : 'General Information',
            Target : '@UI.FieldGroup#General'
```

app > project2 > annotations.cds

```
annotate service.OutboundDelivery with @(
    UI.LineItem : [
        { $Type : 'UI.DataField', Label : 'Product', Value : items.product },
        { $Type : 'UI.DataField', Label : 'Price', Value : items.price },
        { $Type : 'UI.DataField', Label : 'Quantity', Value : items.quantity },
        { $Type : 'UI.DataField', Label : 'Unit', Value : items.unit },
        { $Type : 'UI.DataField', Label : 'Description', Value : items.description },
        { $Type : 'UI.DataField', Label : 'Pick', Value : items.picked },


    ]
);


annotate service.OutboundDelivery with {
    items @Common.ValueList : {
        $Type : 'Common.ValueListType',
        CollectionPath : 'DeliveryUnits',
        Parameters : [
            { $Type : 'Common.ValueListParameterInOut', LocalDataProperty : items_ID, ValueListProperty : 'ID' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'product' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'price' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'quantity' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'unit' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'description' },
            { $Type : 'Common.ValueListParameterDisplayOnly', ValueListProperty : 'picked' },
```

Figure 5.3.11: Coding for annotations.cds (Pick Outbound Delivery)

The figure 5.3.11 shows the coding part for the interface of the application, all features in the application can work and perform depending on this cds file.

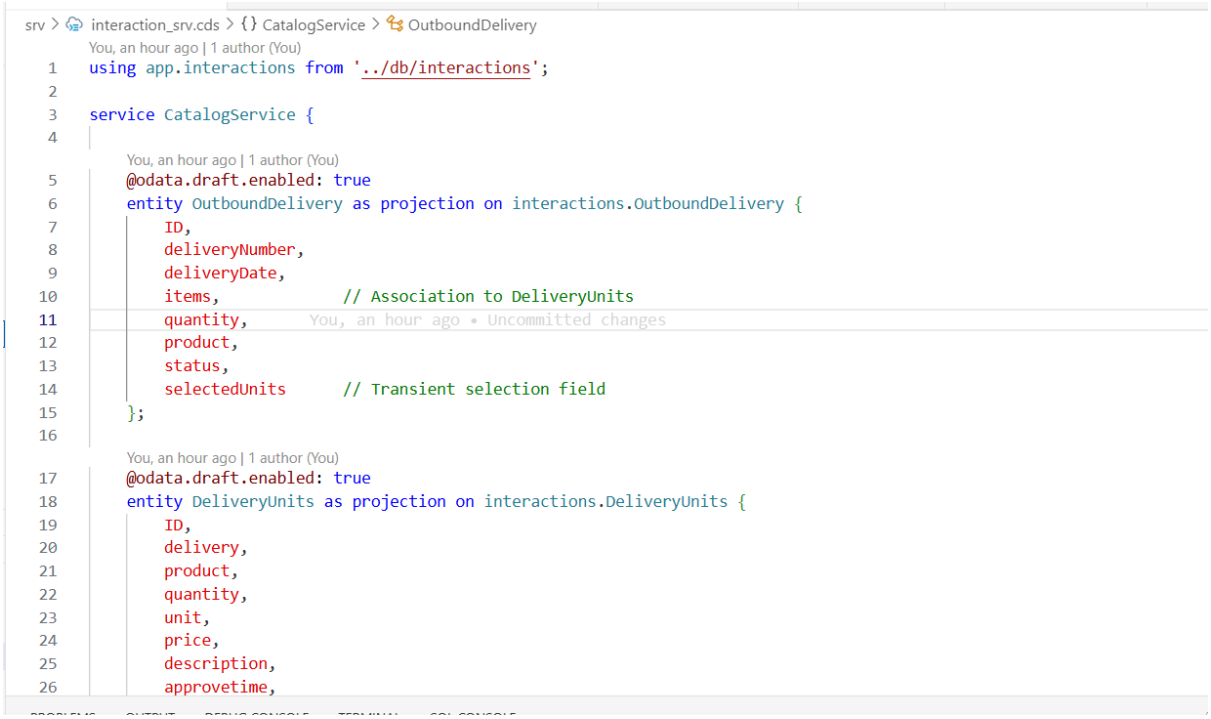


Figure 5.3.13: Coding for interaction_srv.cds (Pick Outbound Delivery)

The figure 5.3.13 shows how the entity database can respond to the presentation layer by figuring out the attributes of the delivery order information.

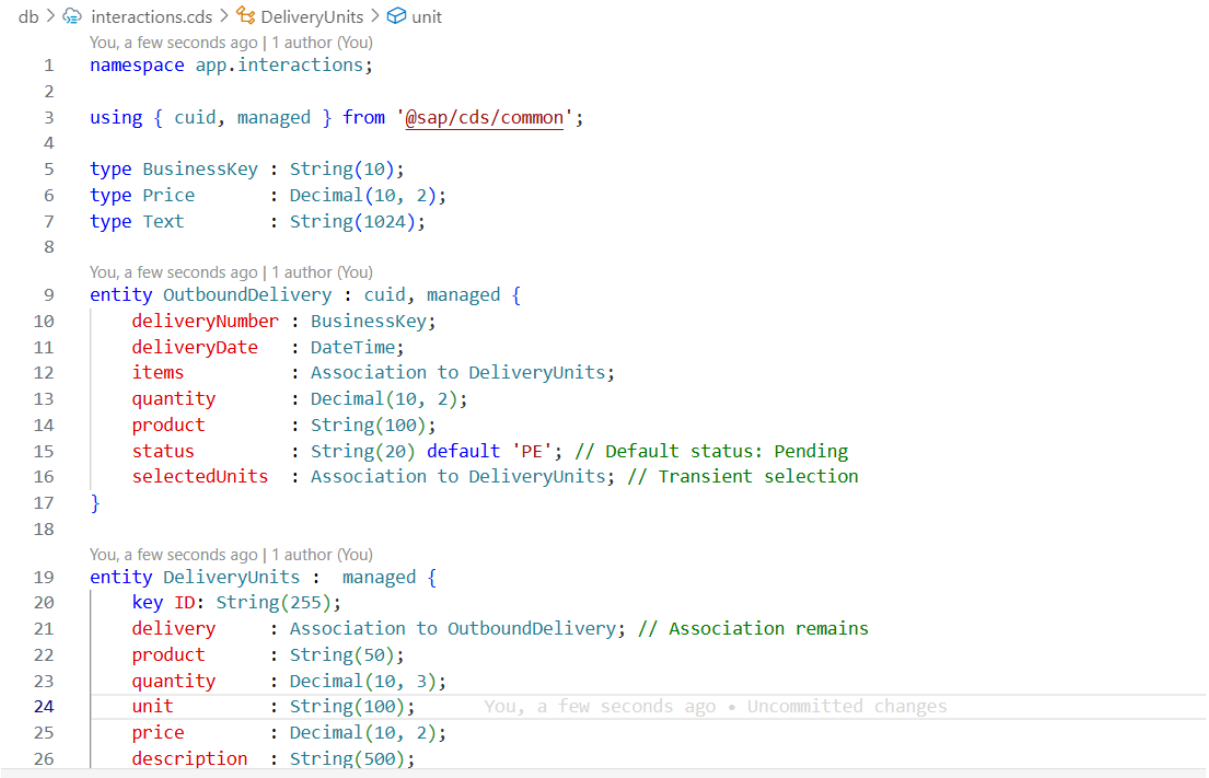


Figure 5.3.12: Coding for interactions.cds (Pick Outbound Delivery)

The figure 5.3.12 shows the database coding of each entity inside the pick outbound delivery module. It is likely the same as sql language database. This ensures that the database storing method is more efficient, clean and well-organized.

Create Outbound Delivery - (Nik Zulaikhaa)

MyHANAApp > srv > outboundDelivery.cds > { } CatalogService

```
using { my } from '../db/schema'; // Adjust the path if necessary
using { sap } from '@sap/cds-common-content';

service CatalogService {

    @odata.draft.enabled: true
    entity OutboundDeliveries as projection on my.OutboundDelivery;

    @odata.draft.enabled: true
    entity OutboundDeliveryItem as projection on my.OutboundDeliveryItem; // Ensure this entity is defined in your schema

    @readonly
    entity Languages as projection on sap.common.Languages;
}
```

Figure 5.3.13: Coding for outboundDelivery.cds (Create Outbound Delivery)

Figure 5.3.13 illustrates how the entity database interacts with the presentation layer by identifying the attributes of the delivery order information.

MyHANAApp > app > outbounddeliveryui > annotations.cds

```
using CatalogService as service from '../../srv/outboundDelivery'; // Adjust the path as necessary

annotate service.OutboundDeliveries with @(
    UI.HeaderInfo: {
        Title: {
            $Type: 'UI.DataField',
            Value: deliveryOrderId
        },
        TypeName: 'Outbound Delivery',
        TypeNamePlural: 'Outbound Deliveries',
        Description: { Value: customerName }
    },
    UI.HeaderFacets: [{
        $Type: 'UI.ReferenceFacet',
        Target: '@UI.FieldGroup#Admin'
    }],
    UI.FieldGroup #GeneratedGroup: {
        $Type: 'UI.FieldGroupType',
        Data: [
            {
                $Type: 'UI.DataField',
                Label: 'Sales Order ID',
                Value: salesOrderId,
                // Add Value Help for Sales Order ID
                @UI.ValueList: {
                    entity: 'SalesOrders', // Adjust this to your actual Sales Orders entity
                    label: 'Sales Order ID',
                    description: 'Sales Order Description' // Adjust as necessary
                }
            },
            {
                $Type: 'UI.DataField',
                Label: 'Delivery Order ID',
                Value: deliveryOrderId
            },
            {
                $Type: 'UI.DataField',
                Label: 'Customer Name',
                Value: customerName
            },
            {
                $Type: 'UI.DataField',
                Label: 'Order Date',
```

Figure 5.3.14: Coding for annotations.cds (Create Outbound Delivery)

```
MyHANAApp > app > outbounddeliveryui > annotations.cds
annotate service.OutboundDeliveries with @(
    UI.FieldGroup #GeneratedGroup: {
        Data: [
            {
            },
            {
                $Type: 'UI.DataField',
                Label: 'Quantity',
                Value: quantity
            },
            {
                $Type: 'UI.DataField',
                Label: 'Price',
                Value: price
            },
            {
            $Type: 'UI.DataField',
            Label: 'Item Description',
            Value: itemDescription
            }
        ]
    },
    UI.FieldGroup #Admin: {
        Data: [
            {
                $Type: 'UI.DataField',
                Value: createdAt
            },
            {
                $Type: 'UI.DataField',
                Value: modifiedAt
            }
        ]
    },
    UI.Facets: [
        {
            $Type: 'UI.ReferenceFacet',
            ID: 'GeneratedFacet1',
            Label: 'General Information',
            Target: '@UI.FieldGroup#GeneratedGroup'
        }
    ],
```

```
MyHANAApp > app > outbounddeliveryui > annotations.cds
annotate service.OutboundDeliveries with @(
    ],
    UI.LineItem: [
        {
            $Type: 'UI.DataField',
            Label: 'Sales Order ID',
            Value: salesOrderId
        },
        {
            $Type: 'UI.DataField',
            Label: 'Delivery Order ID',
            Value: deliveryOrderId
        },
        {
            $Type: 'UI.DataField',
            Label: 'Customer Name',
            Value: customerName
        },
        {
            $Type: 'UI.DataField',
            Label: 'Order Date',
            Value: orderDate
        },
        {
            $Type: 'UI.DataField',
            Label: 'Quantity',
            Value: quantity
        },
        {
            $Type: 'UI.DataField',
            Label: 'Price',
            Value: price
        },
        {
            $Type: 'UI.DataField',
            Label: 'Item Description',
            Value: itemDescription
        }
    ],
```

Figure 5.3.15: Coding for annotations.cds (Create Outbound Delivery)

MyHANAApp > app > outbounddeliveryui > annotations.cds

```
3   annotate service.OutboundDeliveries with @(
83      UI.LineItem: [
119     ],
120     UI.Action: {
121         $Type: 'UI.Action',
122         Label: 'Create New Outbound Delivery',
123         Target: 'create'
124     }
125 );
126
127 annotate service.OutboundDeliveryItems with @(
128     UI.HeaderInfo: {
129         Title: {
130             $Type: 'UI.DataField',
131             Value: itemDescription
132         },
133         TypeName: 'Outbound Delivery Item',
134         TypeNamePlural: 'Outbound Delivery Items'
135     },
136     UI.FieldGroup #GeneratedGroup: {
137         $Type: 'UI.FieldGroupType',
138         Data: [
139             {
140                 $Type: 'UI.DataField',
141                 Label: 'Item Description',
142                 Value: itemDescription
143             },
144             {
145                 $Type: 'UI.DataField',
146                 Label: 'Quantity',
147                 Value: quantity
148             },
149             {
150                 $Type: 'UI.DataField',
151                 Label: 'Price',
152                 Value: price
153             },
154             {
155                 $Type: 'UI.DataField',
156                 Label: 'Currency',
157                 Value: currency_code
158             }
159         ]
```

MyHANAApp > app > outbounddeliveryui > annotations.cds

```
127 annotate service.OutboundDeliveryItems with @(
136     UI.FieldGroup #GeneratedGroup: {
159         ]
160     },
161     UI.Facets: [
162         {
163             $Type: 'UI.ReferenceFacet',
164             ID: 'GeneratedFacet1',
165             Label: 'General Information',
166             Target: '@UI.FieldGroup#GeneratedGroup'
167         }
168     ],
169     UI.LineItem: [
170         {
171             $Type: 'UI.DataField',
172             Label: 'Item Description',
173             Value: itemDescription
174         },
175             {
176         $Type: 'UI.DataField',
177         Label: 'Quantity',
178         Value: quantity
179         },
180         {
181             $Type: 'UI.DataField',
182             Label: 'Price',
183             Value: price
184         },
185         {
186             $Type: 'UI.DataField',
187             Label: 'Currency',
188             Value: currency_code
189         }
190     ]
191 );
192
193 // Annotations for texts related to OutboundDeliveryItems
194 annotate service.OutboundDeliveryItems.texts with @(UI: {
195     Identification: [{ Value: itemDescription }],
196     SelectionFields: [
197         locale,
198         itemDescription
199     ],
```

Figure 5.3.16: Coding for annotations.cds (Create Outbound Delivery)

Figure 5.3.17: Coding for annotations.cds (Create Outbound Delivery)

Figure 5.3.14 - 5.3.17 displays the coding component of the application's interface. All the application's features operate and function based on this CDS file.

Post Good Issue - (Lee Yik Hong)

MyHANAApp > app > project1 > annotations.cds

```
annotate service.Invoices with @(
    UI.Identification : [
        {
            $Type : 'UI.DataField',
            Value : ID
        },
        {
            $Type : 'UI.DataField',
            Value : customer
        }
    ],
    UI.FieldGroup #InvoiceHeader : {
        $Type : 'UI.FieldGroupType',
        Data : [
            {
                $Type : 'UI.DataField',
                Label : 'Invoice ID',
                Value : ID,
            },
            {
                $Type : 'UI.DataField',
                Label : 'Customer',
                Value : customer,
            },
            {
                $Type : 'UI.DataField',
                Label : 'Invoice Date',
                Value : invoiceDate,
            },
            {
                $Type : 'UI.DataField',
                Label : 'Total Amount',
                Value : totalAmount,
            },
            {
                $Type : 'UI.DataField',
                Label : 'Currency',
                Value : currency_code,
            }
        ],
    },
    UI.Facets : [
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'InvoiceHeader',
            Label : 'Invoice Details',
            Target : '@UI.FieldGroup#InvoiceHeader',
        },
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'LineItems',
            Label : 'Line Items',
            Target : 'items',
        }
    ],
    UI.Create : true, // Enables the "Create" button for the entity
    Common.Label : 'Invoices',
    Common.DraftRoot : { DraftNode : true } // Enables draft handling
);

annotate service.InvoiceItems with @(
    UI.LineItem : [
```

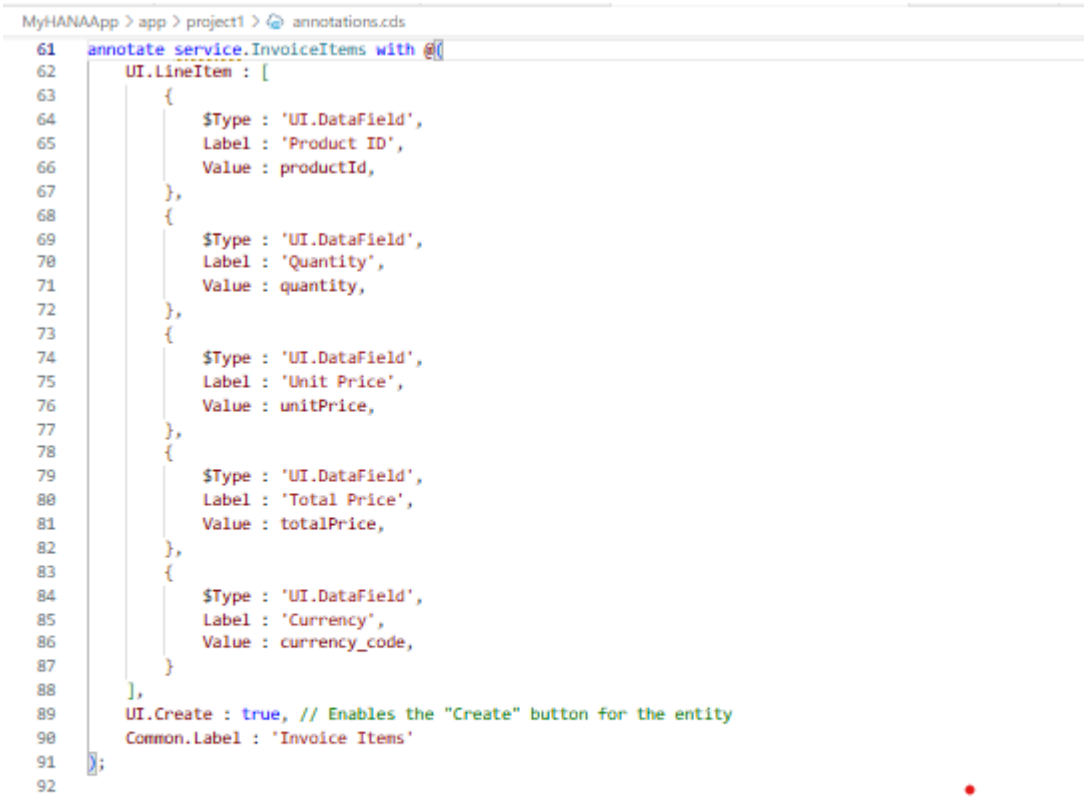


Figure 5.3.18: Coding for annotations.cds (Delivery Order)(Post Good Issue)

The provided **annotations.cds** code enhances the UI for the **Invoices** and **InvoiceItems** entities in the **Delivery Order** service using SAP Fiori elements. For **Invoices**, it defines key identification fields (`ID`, `customer`) and a structured **FieldGroup** (**InvoiceHeader**) to display details like `invoiceDate`, `totalAmount`, and `currency_code`. Facets link the **InvoiceHeader** and related **Line Items**, creating a hierarchical and intuitive display. Features such as a "Create" button and draft handling are enabled for seamless management.

For **InvoiceItems**, fields like `productId`, `quantity`, `unitPrice`, and `totalPrice` are annotated for display in the UI. A "Create" button allows adding items to invoices, and user-friendly labels ensure clarity. This annotation framework provides a structured, functional, and user-friendly interface for managing invoices and their items.

MyHANAApp > app > project2 > annotations.cds

```
using CatalogService as service from '../../srv/interaction_srv';
annotate service.Invoices with @(
    UI.FieldGroup #GeneratedGroup : {
        $Type : 'UI.FieldGroupType',
        Data : [
            {
                $Type : 'UI.DataField',
                Label : 'customer',
                Value : customer,
            },
            {
                $Type : 'UI.DataField',
                Label : 'country_code',
                Value : country_code,
            },
            {
                $Type : 'UI.DataField',
                Label : 'invoiceDate',
                Value : invoiceDate,
            },
            {
                $Type : 'UI.DataField',
                Label : 'totalAmount',
                Value : totalAmount,
            },
            {
                $Type : 'UI.DataField',
                Label : 'currency_code',
                Value : currency_code,
            },
            {
                $Type : 'UI.DataField',
                Label : 'productId',
                Value : productId,
            },
        ],
    },
    UI.Facets : [
        {
            $Type : 'UI.ReferenceFacet',
            ID : 'GeneratedFacet1',
            Label : 'General Information',
            Target : '@UI.FieldGroup#GeneratedGroup',
        },
    ],
    UI.LineItem : [
        {
            $Type : 'UI.DataField',
            Label : 'customer',
            Value : customer,
        },
        {
            $Type : 'UI.DataField',
            Label : 'country_code',
            Value : country_code,
        },
        {
            $Type : 'UI.DataField',
            Label : 'invoiceDate',
            Value : invoiceDate,
        },
```

```
        },
        {
            $Type : 'UI.DataField',
            Label : 'totalAmount',
            Value : totalAmount,
        },
        {
            $Type : 'UI.DataField',
            Label : 'currency_code',
            Value : currency_code,
        },
        {
            $Type : 'UI.DataField',
            Label : 'productId',
            Value : productId,
        },
    ],
);

annotate service.Invoices with {
    deliveryOrder @Common.ValueList : {
        $Type : 'Common.ValueListType',
        CollectionPath : 'DeliveryOrders',
        Parameters : [
            {
                $Type : 'Common.ValueListParameterInOut',
                LocalDataProperty : deliveryOrder_ID,
                ValueListProperty : 'ID',
            },
            {
                $Type : 'Common.ValueListParameterDisplayOnly',
                ValueListProperty : 'customer',
            },
            {
                $Type : 'Common.ValueListParameterDisplayOnly',
                ValueListProperty : 'country_code',
            },
            {
                $Type : 'Common.ValueListParameterDisplayOnly',
                ValueListProperty : 'orderDate',
            },
            {
                $Type : 'Common.ValueListParameterDisplayOnly',
                ValueListProperty : 'status',
            },
        ],
    }
};
```

Figure 5.3.19: Coding for annotations.cds (Invoice)(Post Good Issue)

The **annotations.cds** file enhances the **Invoices** entity's UI with a structured **FieldGroup** (`GeneratedGroup`) displaying key fields like `customer`, `country_code`, `invoiceDate`, `totalAmount`, `currency_code`, and `productId`. These fields are organized under a **Facet** labeled "General Information" and included in the **LineItem** view for consistency.

Additionally, a **Value List** links `deliveryOrder` to the **DeliveryOrders** collection, mapping `deliveryOrder_ID` and displaying related fields (`customer`, `country_code`, `orderDate`, `status`). This setup ensures a user-friendly, efficient interface for managing invoices and associated delivery orders.

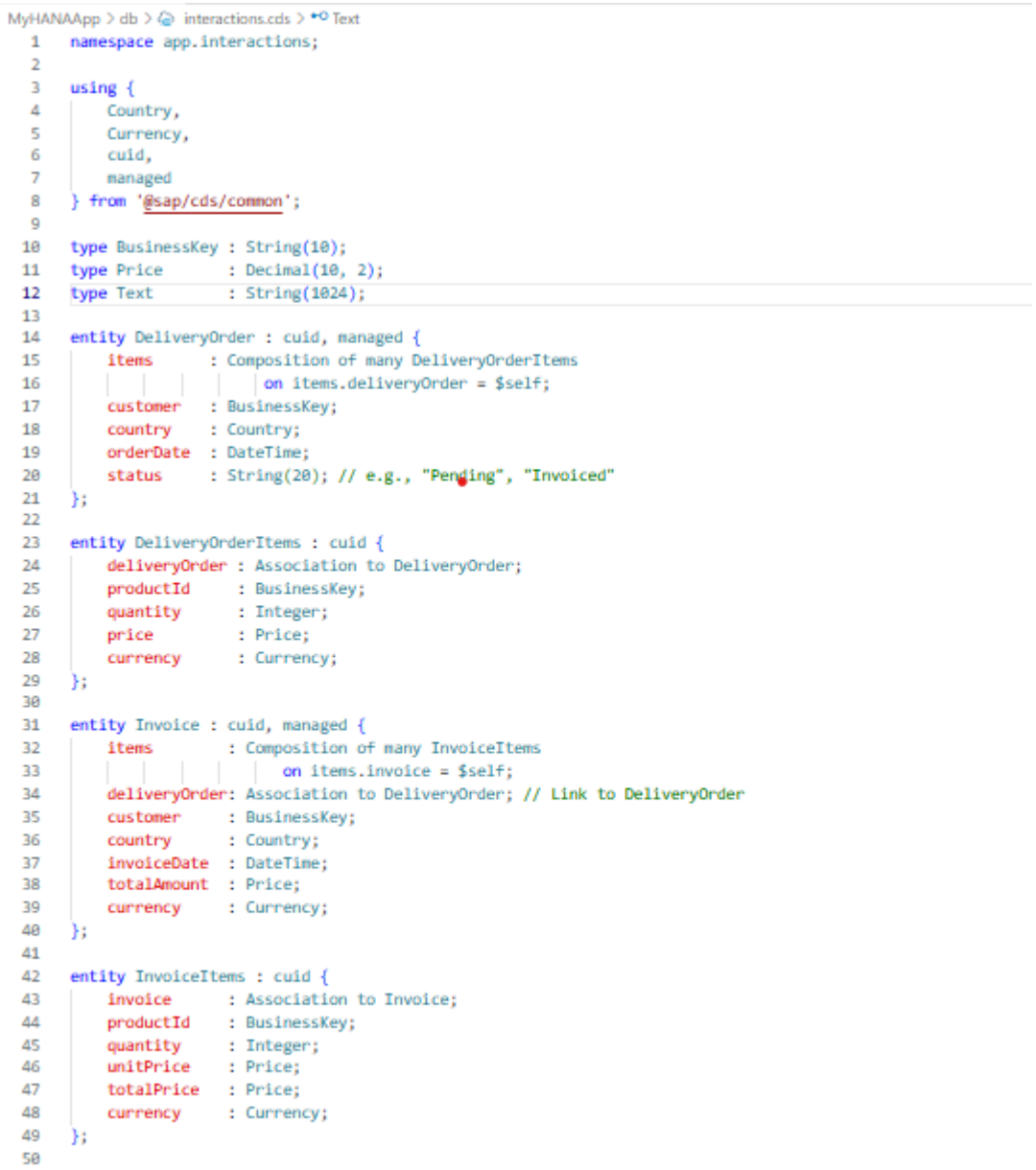


Figure 5.3.19: Coding for db/interaction.cds (DeliveryOrder+Invoice)(Post Good Issue)

The **db/interaction.cds** file defines the data model for managing delivery orders and invoices in the **Post Good Issue** process. The **DeliveryOrder** entity includes attributes such as `customer`, `country`, `orderDate`, and `status`, with a composition linking to **DeliveryOrderItems**. Each item is detailed with fields like `productId`, `quantity`, `price`, and `currency`.

The **Invoice** entity manages invoices with fields such as `customer`, `country`, `invoiceDate`, `totalAmount`, and `currency`. It includes a composition to **InvoiceItems**, capturing details like

`productId`, `quantity`, `unitPrice`, and `totalPrice`. The **Invoice** entity also establishes an association with the related **DeliveryOrder**, ensuring traceability.

This model provides a robust framework for linking delivery orders to invoices, ensuring data consistency and supporting efficient operations.

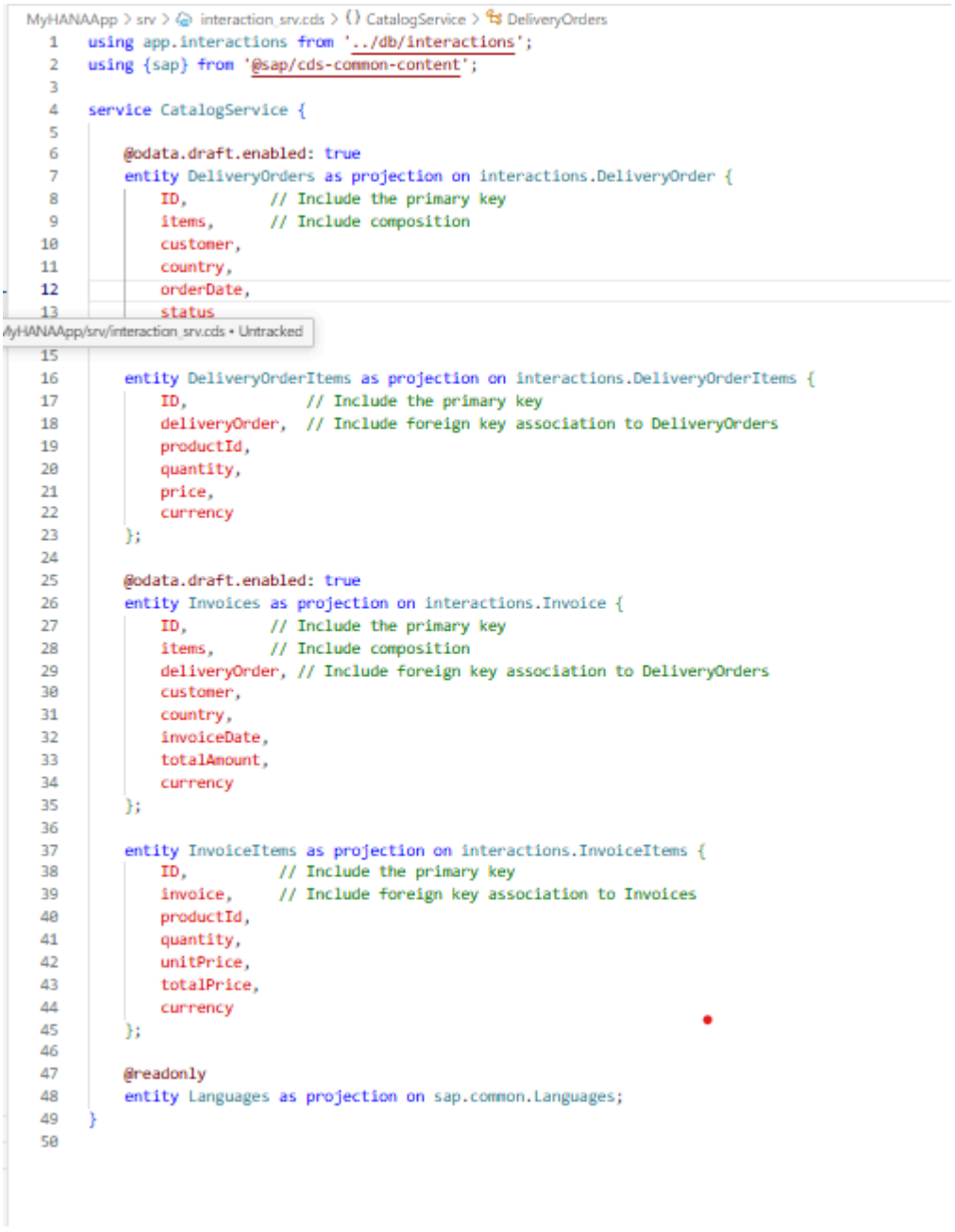


Figure 5.3.20: Coding for interaction.srv.cds (DeliveryOrder+Invoice)(Post Good Issue)

The **interaction.srv.cds** defines the service layer for managing **DeliveryOrders** and **Invoices** in the **Post Good Issue** process. It includes draft-enabled projections for entities like `DeliveryOrders`, `DeliveryOrderItems`, `Invoices`, and `InvoiceItems`, linking delivery orders to invoices via foreign key associations. Key fields such as `customer`, `orderDate`, `invoiceDate`, `totalAmount`, and item details like `productId`, `quantity`, and `price` ensure comprehensive data handling. The inclusion of a read-only `Languages` entity supports standardized language references. This setup enables efficient, draft-enabled workflows for seamless delivery and invoice management.

5.4 Summary

The system implementation phase has successfully translated the system design into a functional application, bringing the sales management system to life for AK Maju Resources. Leveraging the capabilities of SAP Business Technology Platform (SAP BTP) and SAP HANA Cloud, the system integrates advanced technologies to handle large volumes of data, ensure scalability, and maintain seamless operations. This phase is aimed at the process of realization of the AKMaju's sales management system.

By implementing key modules such as stock management, sales order processing, outbound delivery management, and billing, the system achieves its primary objective of automating and streamlining sales processes. The adoption of SAP Fiori has further enhanced the user experience with intuitive and modern interfaces, promoting easy navigation and effective system use.

The system implementation marks a significant step in improving operational efficiency, reducing human errors, and ensuring real-time data accessibility. With thorough testing and integration, the sales management system is well-positioned to meet the current and future needs of AK Maju Resources, setting a robust foundation for operational excellence.

Chapter 6: Conclusion

6.1 Introduction

This chapter summarizes the journey undertaken to develop and implement a comprehensive Sales Management System for AK Maju Resources. The project aimed to address the inefficiencies of the existing manual system by automating critical sales processes, improving accuracy and scalability, and enhancing customer satisfaction.

Through the integration of SAP technologies such as SAP HANA Cloud and SAP BTP, the system has been tailored to meet the unique needs of AK Maju while ensuring scalability and flexibility for future growth. The following sections will discuss the achievements, constraints, and future suggestions for the system, emphasizing its role in transforming AK Maju's sales operations into a streamlined, automated, and efficient process.

6.2 System Contribution/Achievement

Operational Efficiency:

- Data entry automation helps prevent mistakes and saves AK Maju staff time throughout the sales order process.
- The system helps speed up delivery work by generating orders automatically and showing current delivery status.

Improved Accuracy:

- Live inventory updates help the business maintain precise stock levels and prevent both excess inventory and stockouts.
- Automated billing and invoicing work better than people at handling money statements in your records.

Customer Satisfaction:

- Faster order processing and delivery increase customer satisfaction and loyalty.
- Transparent and reliable billing fosters trust with customers.

Scalability:

- The system handles more orders while connecting to all important business functions like people management and product buying.

Centralized Management:

- Preserving sales information in a single SAP S/4HANA and SAP BTP platform gives AK Maju better ways to run their business.

Enhanced Decision Support:

- Built-in analytics for sales order fulfillment and performance provide insights for better strategic planning.

6.3 System Constraint

Dependency on Technology:

- The system depends primarily on SAP S/4HANA platforms and SAP BTP infrastructure. Damage to SAP S/4HANA and SAP BTP infrastructure impacts AK Maju operations.

Initial Implementation Cost:

- Costly in terms technology licenses, staff training, and platform integration before using SAP.

Learning Curve:

- New team members of AK Maju require learning this system before they can use it effectively.

Scalability Limits in Trial Version:

- Since the system operates through a temporary account, these advanced features or expanded usage might have limited performance.

Complexity of Integration:

- Linking the system with current business processes may bring unexpected compatibility problems that need special system modifications.

Real-Time Data Dependency:

- The system functions best when AK Maju staffs enter correct data at the moment they collect it. Problems with input timing or quality information will spread throughout the entire system.

6.4 Future Suggestion

To further support the efficiency and scalability of AK Maju's operations, the following enhancements for the Sales Management System are suggested. These suggestions leverage SAP's Enterprise Information System (EIS) framework and integration capabilities to maximize system performance and business process alignment:

1. Improved Sub-System Integration
   Enhance the integration between the Sales Management System and other core enterprise sub-systems such as procurement, inventory, and finance using SAP S/4HANA. This will facilitate automated data synchronization across departments, improving order-to-cash processes, procurement planning, and financial reconciliation.
2. Real-Time Inventory Optimization
   Expand the real-time inventory management function to integrate with procurement planning. The system can trigger purchase orders automatically when stock reaches a defined threshold, utilizing SAP's Material Requirements Planning (MRP) capabilities to ensure uninterrupted operations and timely replenishment.
3. Automated Billing and Payment Reconciliation
   Enhance the billing system by integrating it with financial management modules. This will enable the system to automate payment reconciliation, ensuring that customer payments are tracked and invoices marked as paid without manual intervention, improving cash flow visibility.
4. Workflow Automation for Sales Approvals
   Utilize SAP Business Workflow to implement approval processes for large or high-priority sales orders. This will ensure that any significant transactions are reviewed by relevant departments (e.g., finance or senior management) before confirmation, reducing the risk of errors or financial discrepancies.

5. Extended Reporting and KPI Dashboards
   Leverage SAP Analytics Cloud (SAC) to provide enhanced reporting capabilities. Real-time dashboards can display key performance indicators (KPIs), such as sales performance, outstanding orders, and delivery timelines, allowing management to make data-driven decisions more efficiently.
6. Customer Data Integration
   Integrate SAP's Customer Relationship Management (CRM) functionality with the sales system. This enhancement would allow for comprehensive tracking of customer preferences, order history, and communications, enabling sales staff to offer personalized services and build stronger relationships.
7. User Experience Improvements with SAP Fiori Enhancements
   Expand the SAP Fiori interface by incorporating additional interactive features, such as drag-and-drop functionality for sales order management, customizable views, and enhanced notifications for pending tasks, to improve usability for both administrative and operational staff.

These enhancements focus on strengthening the integration of enterprise sub-systems, reducing manual workloads, and enabling smoother, automated business processes. By leveraging SAP's comprehensive capabilities, AK Maju can achieve greater operational efficiency, data consistency, and customer satisfaction.

6.5 Summary

In summary, the development and implementation of the Sales Management System for AK Maju Resources mark a significant milestone in addressing the inefficiencies of their current manual processes. The proposed system leverages the capabilities of SAP HANA and SAP BTP to automate critical functions such as sales order processing, inventory management, delivery tracking, and billing. These enhancements not only streamline operations but also minimize human errors, improve real-time data accessibility, and enhance customer satisfaction.

Through the integration of various enterprise subsystems, the system addresses many inefficiencies that existed in the manual approach, such as delayed order processing, lack of real-time inventory updates, and errors caused by manual data handling. These improvements have streamlined operations, reduced errors, and increased customer satisfaction by speeding up service delivery and ensuring accurate billing and order tracking.

However, challenges remain in fully realizing the potential of enterprise integration and automation. The scalability of the system, as well as the integration of complex business processes across departments like procurement, finance, and human resources, requires further refinement. SAP's ecosystem offers several solutions to address these challenges. By enhancing the connection between subsystems within AK Maju through SAP's centralized architecture, the company can achieve better data consistency and workflow automation, crucial for long-term growth and operational efficiency.

Future recommendations included in this chapter align with SAP's EIS model, which emphasizes seamless communication between organizational units through interconnected applications. For instance, integrating real-time inventory control with procurement and sales forecasting can help prevent stock shortages and ensure timely deliveries. Additionally, automated billing integrated with SAP's financial module would support better cash flow management by synchronizing payment processing and reconciliation in real time.

Another critical aspect of the system's evolution is the expansion of data-driven decision-making. Leveraging SAP Analytics Cloud (SAC) for comprehensive reporting and performance monitoring will allow management to track key metrics like sales performance,

outstanding orders, and customer behavior. This aligns with the EIS principle of delivering timely and actionable business intelligence to support strategic planning.

Furthermore, improvements to the SAP Fiori user interface are proposed to enhance usability and operational efficiency for system users. With features such as task notifications, dynamic reporting views, and interactive sales order management, staff will experience a more intuitive interface that reduces the time spent on administrative tasks.

The system's architecture and design focus on scalability, usability, and seamless integration with existing subsystems, ensuring alignment with AK Maju's business goals. By addressing current constraints and setting the foundation for future growth, this system offers a robust platform for operational efficiency and long-term success. As AK Maju continues to expand, the system is well-positioned to evolve with additional features and technologies to meet future demands.

References

1. Greefhorst, Danny & Proper, Henderik. (2011). The Role of Enterprise Architecture. 10.1007/978-3-642-20279-7_2.
2. Stelzer, D. (2010). Enterprise architecture principles: literature review and research directions. In Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops: International Workshops, ICSOC/ServiceWave 2009, Stockholm, Sweden, November 23-27, 2009, Revised Selected Papers 2 (pp. 12-21). Springer Berlin Heidelberg.
3. Elena Kornyshova, Judith Barrios,
4. Process-oriented Knowledge Representation of the Requirement Management Phase of TOGAF-ADM: an Empirical Evaluation, Procedia Computer Science, Volume 192, 2021, Pages 2239-2248, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2021.08.237.
5. Estrach, P. (2024, November 27). Key benefits of Enterprise Architecture. MEGA. https://www.mega.com/blog/key-benefits-of-enterprise-architecture#:~:text=Enterprise%20architecture%20(EA)%20is%20the,to%20improve%20their%20business%20operations.
6. Info-Tech Research Group. (n.d.). *IT strategy case study of a large manufacturing company*. https://www.infotech.com/research/it-it-strategy-case-study-of-a-large-manufacturing-company