SAP Sales and Distribution Automation Framework

A major project report submitted in partial fulfillment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

Submitted by

Ishita Verma (211126)

Under the guidance & supervision of

Prof. Dr. Pardeep Kumar

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

May 2025

Candidate's Declaration

We hereby declare that the work presented in this major project report entitled **'SAP Sales and Distribution Automation Framework'**, submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is an authentic record of our own work carried out during the period from July 2024 to May 2025 under the supervision of **Prof. Dr. Pardeep Kumar**.

We further declare that the matter embodied in this report has not been submitted for the award of any other degree or diploma at any other university or institution.

(Student Signature)
Name: Ishita Verma
Roll No.: 211126
Date: 09-05-2025

This is to certify that the above statement made by the candidates is true to the best of my knowledge.

Date: 09-05-2025
Place: Waknaghat

(Supervisor Signature)	(External Supervisor Signature)
Supervisor Name: Prof. Dr. Pardeep Kumar	Supervisor Name: Surya Suresh
Department: CSE & IT	Designation : Senior Software Engineer
Designation: Professor, SM-ACM	Department: Customer Success Team

Supervisor's Certificate

This is to certify that the major project report entitled **'SAP Sales and Distribution Automation Framework'**, submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, is a bonafide project work carried out under my supervision during the period from July 2024 to May 2025.

I have personally supervised the research work and confirm that it meets the standards required for submission. The project work has been conducted in accordance with ethical guidelines, and the matter embodied in the report has not been submitted elsewhere for the award of any other degree or diploma.

Date: 09-05-2025
Place: Waknaghat

(Supervisor Signature)	(External Supervisor Signature)
Supervisor Name: Prof. Dr. Pardeep Kumar	Supervisor Name: Surya Suresh
Department: CSE & IT	Designation : Senior Software Engineer
Designation: Professor, SM-ACM	Department: Customer Success Team

ACKNOWLEDGEMENT

I sincerely thank all who extended their assistance and valuable guidance to me throughout my project, ‘SAP Sales and Distribution Automation Framework’ This project has been a positive learning journey, and I would not have been able to reach this stage without the enthusiasm and support of a number of individuals.

My sincere thanks are extended to my mentor, Surya Suresh for his constant guidance, valuable advice, and patient encouragement throughout the development of this project, and for his deep knowledge in SAP and enterprise application development that regularly got me over the technical challenges of the project and kept me thinking clearly.

I also want to mention my academic supervisor, Prof. Dr. Pardeep Kumar, Department of Computer Science and Engineering, Jaypee University of Information Technology (JUIT), Waknaghat, for providing ongoing support and feedback, and excellent supervision and advice. He encouraged and helped me substantially from conceptualization until completion.

I would like to acknowledge my friends and support system who provided moral support, brainstormed ideas, and helped out at pivotal moments of this project.

Finally, I am immensely appreciative of my family for their consistent love, encouragement, and faith in me during this time. Their constant support has been a true source of strength.

Ishita Verma
(211126)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Meaning
SAP	System Application Product
ABAP	Advanced Business Application Programming
ERP	Enterprise Resource Planning
ECC	ERP Central Component
HANA	High Performance Analytic Appliance
CDS	Core Data Services
HCM	Human Capital Management
BAPI	Business Application Programming Interface
BADI	Business Add-Ins
PAI	Process After Input
PBO	Process Before output
OOP	Object Oriented Programming
MM	Material Management
GUI	Graphic User Interface
RFC	Remote Function Call
MVC	Model View Controller
ALV	ABAP List Viewer
CRM	Customer Relationship Management
SD	Sales Distribution
BDC	Batch Data Communication
LSMW	Legacy System Migration Wrokbench

LIST OF FIGURES

ABSTARCT

I would like to express my heartfelt gratitude to all those who helped and guided me through the "SAP Smart Sale Workflow System" project. Without the help and support of many many people, this project would not have reached its successful completion. It has been a wonderful educational experience.

I would first like to sincerely thank my mentor, Surya Suresh for his support, wise counsel, and understanding suggestions during this project. Surya was essential in helping me overcome technical hurdles and keeping me focused because of his vast knowledge of SAP and enterprise application development.

I would also like to extend my sincere appreciation to my academic supervisor, Prof. Dr. Pardeep Kumar of the Department of Computer Science and Engineering, Jaypee University of Information Technology (JUIT), Waknaghat. His unqualified support, constructive criticism, and insightful advice were invaluable. His support and suggestions guided the project from the initial stages to completion.

I would like to acknowledge my peers and goodwill members who also helped out during some critical stages of the project, shared ideas, and provided necessary moral support.

Finally, I want to express my heartfelt thanks to my family for all of the support, the encouragement, and the faith in me along the way. Their support has fostered my greatest strength!

Ishita Verma
(211126)

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Systems, Applications, and Products in Data Processing (SAP) is a well-known enterprise resource planning (ERP) application that helps firms manage and integrate their core functions. The suite of SAP modules encompasses several functional modules, including Financial Accounting (FI), Materials Management (MM), Human Capital Management (HCM), Production Planning (PP), and Sales and Distribution (SD). Each module is tightly integrated to allow information to flow, be transparent in operations, and keep data centrally with various departments.

The SD module is important for managing client contacts, delivery of products, sales actions, and billing activities. It represents of the order-to-cash cycle; from client inquiry, quotation, sales order, delivery, invoicing, and finally payment. The interlinking SD module's with MM and FI modules make possible for an accurate invoicing process, timely and efficient synchronization of department wide data, and in-transit stock investigations in real-time.

Organizations will often find scenarios where functionality does not 100 percent match their business processes or UI needs, often because the SD module can be built out in a flexible way but based on basic processes. SAP allows some development with their proprietary programming language, ABAP (Advanced Business Application Programming). Developers using ABAP can develop customized applications that introduce a different level of user experience, and allow for custom processes at the front end with specific company validations.

At the core of business apps built using ABAP is the module pool programming technique, which allows a developer to create screen-based transactions that have more control over user input and navigation. Using module pool programming, a developer can create business applications for an organization that lead users to take action in a more streamlined way. This type of application can initiate user actions in the next logical step of the process each and every time, as compared to retaining knowledge of the last entry the user made within a specific application transaction concept. Depending on the form of this particular application, users can navigate using enhanced elements such as push buttons, drop down menus, data entry forms, email alerts and notifications, various ALV (ABAP List Viewer) grids, as well as controlled pages or login screens.

Employing a full software module pool to optimize the sales and distribution process is the distinct solution provided as part of this act. write the response. The solution is made up of many features that include: A customer dashboard to manage orders, an admin dashboard to view and monitor orders, a secure registration system with customer sign up, and standard SAP system functionality such as raising inquiries, generating invoices to SAP Smart Forms.

Key features include: registration interface that gives the customer a drop-down choice for security question responses, customer IDs automatically generated, password validation, and secure data handling. Once signed in, customers can view their orders on an ALV Grid, raise queries, request issue follow-up, accept/reject quotes and cancel orders. However, admin users can complete orders, validate stock availability, filter order data by order status, email quotes, and produce bills using Smart Forms.

Custom SAP tables are designed to hold information about clients, supplies, orders, and quotes. The tables are populated using data migration and batch data transfer (BDC) processes, rather than collected and entered by hand, which creates better data integrity and a more realistic testing environment by closely aligning to a real-time corporate data condition. During the implementation period, SAP best practices were followed, including modularized code, standardized naming conventions, unified screen layout, and SAP UI presentation guidelines. The application has enhanced functionality and user experience by utilizing ALV Grids, smart forms, and triggers to automate email sending.

This project demonstrates how ABAP can be used to enhance SAP functionality and develop a complete and easy-to-use solution for managing sales operations. In doing so, efficiencies can be created when capitalizing on standard functionality with custom designed enhancements that reduce manual activity and provide users control and visibility of the important business transactions.

1.2 PROBLEM STATEMENT

The Sales and Distribution (SD) process is a key component of the organizational context providing revenue generation and a relationship with customers for your business through order handling, quote handling, and physical or digital product delivery. SAP has a robust SD module, but for internal users first and foremost, lack of flexibility by design from dynamic workflows, specific user experience environments, and external customer interaction. This becomes clear when speaking to clients looking to just allow interaction where the client has access and can interact directly. There are multiple conversations I was a part of mentioning what would be standard by leveraging integrations but reality was otherwise based on standard transactions in SAP.

Even though we have the goal to build a single, SAP-based system to manage every part of the sales process including customer registration, order entry, quote processing, reviews, cancellations, final invoice delivery, the current SAP SD environment does not fit well with regards for custom login screens, customer registration processes, secure authentication, and delivery of direct interaction like being able to view open orders and requests for quotations as well as the ability to accept and reject proposals - all of which are considered standard requirements that contribute to customer experience and transparency of process.

The administrators had problems with the status order process that prevented them from accurately managing order status, stock availability, alerting the customer through automated means, and generating quotes as required. In addition, the lack of intended functionality such as GUI attributes/processes that are dynamic, real-time email communication with the customer, ALV Grid order display, and making use of smart forms to make customized invoices, created a disjointed workflow which was based upon manual process, and indirect communication, which increased errors, delays, and frustrated customers.

Also, another challenge stemmed from the failure to provide a controlled testing environment to analyze the business processes utilizing real-time data. It has been difficult to test the end-to-end cycle, in an integrated manner, as manual data entry or TMG usage could never simulate the real-world business context. Wherever possible, to ensure data is loaded correctly, as intended by the settings, from actual settings to need to rely on data migration type procedures, or BDCS.

A new SAP application was built and implemented using ABAP and module pool programming to satisfy these gaps. The solution generates a unique transaction code that serves as the only point of entry for administrators and clients. It has a dynamic registration process, secure login capabilities, displays that are made to the specifications of SAP standard GUI elements, and a dashboard specifically made for clients to manage orders and inquiries. Administrators can do everything via one interface by filtering orders by status, checking stock, sending quotes with attachments by email, processing authorized orders, and generating bills by using Smart Forms.

To achieve consistency, as well as simple navigation through standard SAP SD displays, DDIC tables are also utilized by the program. It makes convenient use of ALV Grids for structured data and aligns with SAP naming and code standards, iteration and reusability are implemented correctly by using OOP concepts, and it works within the confines of the application. The unique solution definitively addresses the restraints of the agreed-upon SD module and creates a dynamic, interactive, and automated way for both administrator and users to work in a unique manner that suits the specific business needs.

1.3 OBJECTIVE

The objective of this project is to develop a fully customized SAP ABAP-based application that utilizes module pool programming, SAP Smart Forms, email communication, and custom database tables to optimize and digitize the Sales and Distribution (SD) process. The solution's main goals are to lessen reliance on human interventions, increase process transparency, and offer administrators and consumers a seamless user interface for effective order management. In order to provide an end-to-end order lifecycle management system, the project places a strong emphasis on automation, user-specific workflows, and interaction with conventional SAP capabilities. The following are the project's main goals:

- **Streamlining the SD Process with Custom Screens:** Custom-designed module pool screens were created to streamline customer-related tasks such creating inquiries, responding to quotations, tracking orders, and canceling. To guarantee precise and instantaneous transaction handling, every feature was integrated with the required field validations and logical flow.

- **Creating a Centralized and Secure Entry Point:** To operate as a single point of entry to the program, a unique transaction code was developed. Both administrators and consumers can use its dual login interface. To preserve data security and confidentiality, the customer registration module makes sure that each ID is generated uniquely and enforces password and security question validation requirements.

- **Automating Notifications and Improving Communication:** The application includes automated email notifications that are triggered at important business actions like inquiry submissions, quotation updates, cancellation requests, and order follow-ups. This allows for smooth communication between customers and the administrative team. This feature minimizes communication delays and keeps stakeholders informed.

- **Using Standard SAP Features to Improve Output**: Where appropriate, standard SAP Sales and Distribution screens were utilized, especially for the creation of inquiries. In order to ensure that output papers adhere to SAP compliance procedures and professional standards, Smart Forms were also used to create and transmit system-generated bills upon order fulfillment.

- **Organizing and Managing Data With Custom Tables:** A number of unique DDIC tables were made to hold information on clients, supplies, orders, and quotes. In order to maintain a steady and precise data flow, these tables are updated often in response to user activities. Order information is shown via interactive ALV grids, which provide the option to filter, choose, and perform further actions on client orders.

- **Enabling Admin Operations with Order Filtering and Fulfillment:** The admin panel offers options to check product stock, send quotes, and fulfill accepted orders in addition to allowing order filtering by state (Open, Cancelled, or Completed). After completion, Smart Forms are used to generate invoices. Better process monitoring and decision-making for administrative users are made possible by this modular architecture.

- **Adhering to SAP Coding and User Interface Guidelines:** To ensure code readability and reusability, all modules were designed using object-oriented ABAP principles and modularization techniques. All custom panels have a disciplined and consistent user experience, as all GUI components are implemented with SAP standards.

1.4 SIGNIFICANCE AND MOTIVATION OF THE PROJECT WORK

This project represented a great opportunity to work with SAP technology in a hands-on manner and addressed real-life business problems. This presented an opportunity to discover how SAP could be personalized to allow the automation of business procedures, particularly in terms of distribution and sales. Therefore, I was able to close the gap between the theoretical worlds of higher education and the practical world of corporate usage. The primary drivers of this project were to investigate the technical nature of SAP ABAP, more specifically, the area of module pool programming. Additionally, I was able to research how graphical user interfaces are developed to achieve specific business objectives. I was able to develop customized displays, encapsulate login portions of the project, and automate presentations of data to provide more clarity along the whole SAP programming lifecycle, starting from data modeling with Data Dictionary to real-time data validation and screen navigation logic. All of this provided opportunities to simulate business processes (client registration, completion of inquiry, managing an order) resulted in a sharpened focus on how interdependent all of the system components must be for the system as a whole.

Reflecting on the processes for handling cancellations, generating invoices and connecting queries to quotes highlighted how varied operations within SAP SD are all interrelated. The added level of complexity with the automated email alerts based on consumer interactions added a further layer to the system. We learned quite a bit about the importance of responsiveness and communication in corporate systems and how workflows can be scripted to create event-triggered triggers which reflect real communication flows.

I learned insight regarding transactions through the cancelling, invoicing processes as well as connecting queries to quotes, these highlighted how diverse operations within SAP SD were interrelated. The system was brought to another level with the incorporation of automatic email alerts that were based on consumer engagements.

The project allowed participants a chance to learn about SAP Smartforms and to develop and produce invoices through SAP Smartforms. The project, in addition to allowing participants a broad understanding of SAP's output management functions, evolved their understanding of processing, formatting, and sending documents in customer-facing operations.

All things considered, this project improved my technical foundation and gave me more confidence to modify SAP apps. It provided a consultant-like viewpoint where all technical decisions are informed by business needs and emphasized the need of clear, maintainable code methods. This training has given me information that is both academically and professionally beneficial, and it will surely help me advance in my future SAP development positions.

1.5 ORGANISATION OF PROJECT REPORT

This project report is organized into six in-depth chapters, each of which focuses on a distinct stage of the development process and offers a thorough understanding of a range of system design and implementation-related topics.

- **Chapter 1: Overview** : This chapter presents the project's central idea and outlines the background against which the SAP-based solution was created. It explains the history, the issues with manual sales and distribution procedures, and why SAP was chosen for automation. Along with summarizing the report's general structure, it also describes the application's main goals and anticipated results.

- **Chapter 2: Literature Review** : An extensive analysis of the technical elements pertinent to the project is presented in this part. It examines the ideas of process automation, Smartforms, module pool programming, and SAP ABAP. In order to choose custom SAP development as a solution, the chapter also examines the capabilities of the SD module and the drawbacks of traditional systems.

- **Chapter 3: System Design and Implementation** : From requirement analysis to interface design, data modeling using custom tables, and screen creation with module pools, it outlines the whole system development process in depth. It goes into detail on how transaction codes, validations, data transfer via BDC, and the reasoning for workflow-based email alerts are all integrated.

- **Chapter 4: System Testing:** It explains the many testing stages used to guarantee the built application's performance, data consistency, and functional accuracy. Key test scenarios, test data, observed findings, and any discrepancies between predicted and actual results are presented.

- **Chapter 5: Output Analysis and Discussion** : This chapter presents the outcomes that were attained following effective implementation. It describes how each component worked in real-world scenarios and how the system as a whole resolved the problems with the Sales and Distribution process that had been previously found. It also reflects on the advances obtained in process automation, user engagement, and communication efficiency.

- **Chapter 6: Conclusion and Future Enhancements** : By summarizing the project results and offering reflections on the experience gained throughout development, the last chapter brings the report to a close. It highlights potential improvements that might increase the solution's value and scalability, including adding more SAP modules, enhancing reporting features, or using analytics to aid in decision-making.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

Business process automation, integration, and customisation are becoming more necessary as a result of the development of enterprise resource planning (ERP) systems. SAP is still the most widely used ERP software for managing operations, particularly in the Sales and Distribution (SD) modules. Numerous research studies and technological contributions have examined the ways in which SAP solutions like as workflow automation, Module Pool Programming, and Smart Forms might improve the effectiveness of certain business processes.

Early studies concentrated on the fundamental application of Smart Forms for document production, offering approachable substitutes for more antiquated technologies like SAPscript. The development of layout-driven forms and their integration with ABAP driver programs to produce documents such as quotes and bills were the main focus of developer guides and practical tutorials. This prepared the way for more complex use cases, including using the CL_BCS class to send dynamic emails with attachments straight from SAP systems.

Concurrently, the area of ABAP module pool programming attracted interest as it made it possible to create interactive user interfaces in SAP GUI. The ability of screen painter tools and transaction code creation to provide bespoke applications like login pages, user dashboards, and real-time data processing has been covered in great detail in literature and SAP practitioner documentation.

Another area that has been thoroughly studied is the integration of event-based triggers and processes. Numerous case studies have illustrated the use of flexible processes to automate customer contacts, alerts, and approval chains. MyInbox and SAPconnect, two of SAP's native technologies, have been essential in establishing smooth SD operating experiences.

Additionally, role-based access control (RBAC) in SAP systems has been the subject of an expanding corpus of study. Developers have ensured safe and pertinent access by limiting functions according to user roles (e.g., admin vs. customer) through the usage of authorization objects.

In order to enhance functionality and user experience, recent research has switched its attention to integrating SAP systems with real-time ALV Grids, data validation methods, and contemporary UX principles (such as SAP Fiori). There are still limitations in the products' customization for small and mid-sized enterprises with limited resources, though.

In conclusion, the body of current literature offers solid basic understanding of every system component created in this research. Yet, for businesses that need customized SD management without fully depending on SAP S/4HANA or Fiori, a unified, role-based, workflow-integrated solution must be created utilizing traditional SAP capabilities, particularly module pool programming and Smart Forms.

2.1 Table for literature Review

S. No.	**Paper Title [Cite]**	**Journal/ Conference (Year)**	**Tools/ Techniques/ Dataset**	**Results**	**Limitations**
1.	Interactive Process Identification and Selection from SAP ERP	arXiv (2022)	Interactive SAP Explorer tool; Labeled property graph	Facilitates process identification within SAP ERP for process mining	Tool focuses on process identification; does not address customization or automation
2.	Smartforms–Hands-on Exercises	SpringerLink (2017)	SAP Smartforms; Form Painter; ABAP driver programs	Offers practical exercises for designing business documents using Smartforms	Focused on form design; does not address integration with workflows or SD module.
3.	Migrating SAP Smart Forms to Adobe Forms	SAP Community Blog (2024)	SAP Smart Forms, Adobe LiveCycle Designer	Provide a step-by-step guide for migrating Smart Forms to Adobe Forms, enhancing user experience and integration with	Focuses on migration process; does not address performance benchmarking or user

				modern SAP systems	feedback
4.	Exploring Integrational Aspects of SAP Access Control [	SAP Learning Journey (n.d.)	SAP Access Control, SAP SuccessFactors, SAP Cloud Identity Access Governance	Explains integration of SAP Access Control with various SAP systems to manage user access and compliance	Educational content; lacks empirical data or case studies
5.	The Evolution of SAP Security, Access Control, and IAM	InsideSAP (2020)	SAP Composite Roles, Business Roles, Identity and Access Management	Discusses the progression of SAP security models and the implementation of IAM solutions for efficient access provisioning	Descriptive analysis; does not provide implementation guidelines or performance metrics
6.	Securing the Future: Safeguarding Fiori Development with Role-Based Access Control (RBAC) and Authentication	Goerz-IT (2024)	SAP Fiori, Role-Based Access Control (RBAC), Authentication Mechanisms	Highlights the importance of implementing RBAC and authentication in SAP Fiori applications to enhance security	Opinion-based article; lacks empirical validation or case studies
7.	Send HTML Mail using Smartforms	SAP Community Blog (2011)	SAP Smart Forms, ABAP, HTML Email Templates	Demonstrates sending HTML emails using Smart Forms, enhancing document aesthetics and	Focuses on email formatting; does not cover integration with broader workflows

				communication	
8.	Triggering a Workflow on Receiving Inbound Email with Interactive Offline Adobe Form Attachment and Error Handling	SAP Community Blog (2021)	SAP Interactive Forms by Adobe, Email Integration	Demonstrates how to initiate workflows upon receiving emails with interactive forms, enhancing automation and error handling.	Focused on email-triggered workflows; additional configuration may be needed for SD-specific scenarios.
9.	Add Mail Notifications to a Process	SAP Tutorials (2022)	SAP Build Process Automation, Mail Notifications	Guides on adding email notifications within business processes, ensuring timely updates	Tutorial-based; may require adaptation for complex SD scenarios
10.	Email Sending Functionality for Different Business Processes in SAP	SAP Community Blog (2012)	ABAP, SAP Business Workflow, Email Integration	Explores methods to send emails across various business processes, improving communication	General overview; lacks specific focus on SD module

2.2 KEY GAPS IN THE LITERATURE

Even while SAP ERP systems and its component parts have been the subject of a substantial amount of study, several important topics have not received enough attention, especially when it comes to real-time, integrated solutions designed for small to mid-sized businesses. The lack of emphasis on the real-world implementation of end-to-end SAP custom development utilizing native technologies like module pool programming, Smart Forms, and workflow automation is a significant flaw in the literature currently in publication. The majority of academic research focuses on general implementations or theoretical models, but it doesn't show how these tools may be used in concert to address actual business problems in the Sales and Distribution (SD) space.

The inconsistent handling of SAP tools in scholarly and commercial publications is another drawback. Without examining how these elements work together as a cohesive system, studies frequently separate aspects like form generation, UI design, or backend processing. There isn't much research showing a whole workflow that is controlled within a specially designed SAP module that also supports user roles and access control, from client registration and quotation creation to order processing and invoice delivery. The creation of systems that accurately represent operational demands is hampered by the lack of such comprehensive investigations.

Furthermore, the significance of user-specific customisation is frequently overlooked in the research now in publication. Standard SAP features are extensively documented, but little research has been done on developing role-based user interfaces, dynamic validation systems, or interactive dashboards that serve different business stakeholders, such as administrators and clients. This disparity severely restricts the usefulness of existing research in fields where user experience, security, and adaptability are essential, as in the instance of SmartSale Ltd.

Furthermore, most educational and training materials on SAP development focus on syntax and functional overviews rather than delving into real-time project issues like email integration, modular programming, performance optimization, or error management in complicated transactions. There is frequently a gap between what developers have learnt and the difficulties they encounter while deploying and maintaining systems in business settings.

Last but not least, despite the growing popularity of process mining and automation, its use in fundamental SAP modules like SD is still underrepresented. Process mining studies now in existence usually concentrate on financial or procurement processes, offering little insight into how these methods might be used to map and optimize sales-related workflows like quotation approvals, cancellations, and invoicing.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENT AND ANALYSIS

Using a role-based SAP ABAP application, SmartSale Ltd.'s Sales and Distribution Management System was created to expedite order processing and client interactions. By integrating a simple user interface with backend bespoke SAP tables and function modules, the main goal was to streamline processes such as creating inquiries, tracking orders, and processing deliveries. Customers and administrators are the two main user roles that the program provides. Following a secure login, each position has access to particular features. Registration, authentication, client inquiry/order management, and administrator approval and delivery processing are core functional areas.

3.1.1 Functional Requirements

- **Login and Authentication:** The login interface functions as a safe entry point to various modules according to user roles, benefiting both administrators and consumers. While the admin login employs a predetermined internal check, customers log in using credentials kept in the special table ZCUSTOMER. The system checks the credentials entered upon submission, and if any are missing or wrong, the relevant error messages are shown. After authentication, the admin is transported to the order processing area, and clients are sent to their customized dashboard with purchase history and inquiry options. For new clients, there is a "Register Here" link that takes them to the registration page.

- **Customer Registration:** By providing personal information including their name, password, email address, security question, and response, new users can create an account using the customer registration module. The system reads the most recent item in the ZCUSTOMER table and increments it to guarantee that every new client is given a unique ID. Throughout the process, strong input validation procedures are used: all mandatory fields must be filled out, passwords must match in both fields, and they must fulfil minimal security requirements. The consumer is prompted to fix the inputs before continuing via user-friendly error messages in the event that any validation fails. The approved data is inserted into the ZCUSTOMER table using a specific function module, guaranteeing database consistency and modularity. Password recovery is made possible by the addition of a security question and response. After successfully registering, the user receives a confirmation message and is sent to the login screen, where they may start utilizing the order and inquiry functions. This guarantees that the system is only accessed by securely registered, authorized users.

- **Customer Dashboard and Order Management:** Customers are shown a dashboard with an ALV Grid display that provides a consolidated view of their past queries and order statuses after successfully logging in. They can use this interface to submit new questions, stop open ones, and do follow-up tasks like accepting or rejecting orders that have been confirmed. Using backend SELECT queries, the system dynamically retrieves pertinent data for the logged-in client and presents it in real-time. Function modules that oversee the creation, cancellation, and updating of orders are activated via buttons on the dashboard. Customer navigation is smooth and easy because to the interface's clear and simple design.

- **Admin Dashboard and Order Processing:** Through an ALV-based report, the admin dashboard offers a summary of every customer order. Admins can click a button to evaluate, accept, or reject outstanding orders. Only authorized orders are able to move on to the delivery phase. Additionally, the administrator may create delivery papers that mimic the actual SAP SD procedure. Backend function modules that update custom tables and monitor changes in order status are called by each action. From inquiry to delivery, the system is made to guarantee that each stage of the order lifecycle is tracked, managed, and handled effectively.

- **Inquiry and Delivery Workflow:** Customers choose a product and submit a request to start the inquiry process. Every query is traced through the system and recorded with a distinct inquiry ID. Administrators review these requests and, if accepted, start processing sales orders and deliveries. Custom logic handles each status change, such as "Pending," "Accepted," "Rejected," or "Delivered," and the customer's dashboard updates in real time. Structured ABAP code is used to manage these modifications, guaranteeing data integrity and effective status communication between admin and customer roles.

3.1.2 Non-Functional Requirements

The Sales and Distribution Management System's quality qualities are defined by the non-functional criteria, which guarantee that the application operates effectively, safely, and dependably for administrators and consumers alike.

1. Reliability and Availability: The system is made to function reliably in a variety of situations, giving administrators and customers unbroken access. Operations like client registration, login, order generation, and approval are carried out dependably without system crashes or data loss by utilizing bespoke function modules and error handling logic. Incorrect inputs are guided by error messages, which guarantee

2. Usability: To ensure use, a simple interface has been built using SAP GUI panels and ALV reports. Simple and easy design is the login page which is also used by both the administrator and clients. While the administrator can accept or reject orders, manage delivery, and look at what the customers are doing, the customers can only place purchases, view the status of their orders, and update their accounts. Input areas are labeled appropriately, clear instructions on each action are visible, and the overall configuration is basic, all of which reduces potential confusion for uses and allows for minimal training effort.

3. Performance Efficiency: The performance efficiency of the application relies on the usage of internal tables, arranged function modules, and also their effective SELECT statements to complete backend work as fast as possible. The model embraces developability, allowing events e.g. placing an order, creating a customer's ID, and updating ZCUSTOMER and ZORDER tables, to be processed quickly in the background. The ALV reports and form submission screen load times were reduced which provides fluid function even with numerous users in the application or on a single screen.

4. Security: Security is an omnipresent priority throughout the entire application. Customer registration allows simple password verification, reconfirmation of re-entered passwords, and even a security question for eventual help recovering accounts. Every authentication credential is secured in ZCUSTOMER database. At customer login, consumers, only see their orders; however, when the appropriate administrator roles have been authorized, support functions for approval and delivery become available. This is achieved as evidence and optional playable moments because, for a given user role, the system checks to see that the role for action is valid. Replaced role-checks (validating roles)

5. . Data Integrity and Error Handling: To ensure data consistency, there are specific function modules for each operation that can utilize custom tables. These ensure that transactions are committed correctly; and if not, rollback techniques are implemented to safeguard against incomplete or corrupted data insertions. Each user interface that utilizes function modules employs input validations against the same validations in other modules and ensures that no invalid or harmful data can be inserted into the system. When a validation fails, the system displays a user-friendly error message that enables the user(s) to amend their input as necessary without impacting their workflow.

3.1.3 Custom Tables Used

Custom tables were built utilizing SAP's Data Dictionary (DDIC) to create data structures that could accommodate the management and storage of data in multiple modules of the system. Tables form the basis for managing all product information, order workflows, and engagement with customers.

The **ZCUSTOMER** table is essential for managing users. It contains important client information including a unique customer ID, the customers full name, encrypted password, a security question, and the answer. The ZCUSTOMER table provides secure and authenticated user management during user logins and password re-set.

The **ZPRODUCTS** table has all the relevant product information, and is kept current in all aspects. It may contain fields for product ID and name, price per unit, and real time stock levels, so that the application can check for availability in real-time when create order.

All customer orders' header-level data is managed by the **ZORDERS** table. The order ID, order placement date, and current status (such as open, completed, or canceled) are all recorded in each entry. In order to guarantee precise traceability and user-specific tracking, it also creates a connection to the specific customer who placed the order.

The **ZINQUIRY** table is used to handle pre-sale and customer support inquiries. It records each customer's unique question about a particular product or order and records the pertinent customer and product information for the administrator's use.

Last but not least, the administrator's price quotations are managed by the **ZQUOTATION** table. Information like the quoted amount and the status indicating whether the customer has accepted or rejected the proposal are included in these quotations, which are linked to the corresponding orders.

3.1.4 User Requirements

Customers and administrators are the two main user categories, and the system has been developed to meet their basic needs.

Customers may register, securely log in, make orders, ask questions, and read or reply to quotations using the system's user-friendly interface. Key personal information, including name, password (with complexity validation), and a security question-answer combination to facilitate password recovery, must be entered by the user upon registration. Customers may place new orders by choosing items and quantities, log in, explore the products that are available, and monitor the status of their purchases after registering. They may also check quotes provided by the administrator before accepting or rejecting them, and they can ask questions to get answers to questions about orders or items.

For **Administrators** , the system provides the administrators with the tools needed to manage and monitor all customer interactions and information. Simply log in securely, and you will be presented with a dashboard that shows you all incoming orders, customer requests, and details about the products. You as an administrator will need to respond to all client requests, provide quotes, accept or decline orders, and keep the product inventory up to date. In the role of administrator, you will have privileges that allow your access to all important data tables and it is your obligation to keep the integrity of the business transactions safe by providing timely updates and confirmations.

All users must be authorized and possess a reliable internet connection before using the various functions of the system. Problem-free processes for all users require secure data-handling, error-handling and valid input validation to take place.

3.1.7 Use Case Descriptions

Use Case 1: Customer Registration

Consumers provide their name, password, security question, and response while registering. By comparing re-entered passwords, necessary fields, and password complexity, the system guarantees input validation. The last entry in the ZCUSTOMER database is automatically increased to provide a new, distinct customer ID. Following successful data entry and validation, the client is sent to the login screen. This procedure guarantees that the system is only accessible to verified users.

Use Case 2: Customer Login

Users who have registered use their login information to log in. The information in the ZCUSTOMER database is compared to the user ID and password by the system. The user's dashboard is accessible upon successful authentication. Should the information be inaccurate, the system either displays pertinent error messages or enables the user to retrieve their password by answering a security question.

Use Case 3: Placing Orders

Consumers may order, define quantity, and peruse and choose things from the list. The ZPRODUCTS table is used by the system to determine stock availability. An entry relating the order to the customer ID, order date, and order status is made in the ZORDERS database if the product is in stock. The system notifies the user as necessary if there is not enough stock or if the input is incorrect.

Use Case 4: Raising Inquiries

Customers can use an inquiry form to ask questions about orders or products. These questions are associated with the appropriate client and product and are kept in the ZINQUIRY database. This makes it possible for the administrator to monitor and address inquiries.

Use Case 5: Admin Quotation Management

Incoming orders and queries are visible to administrators. After evaluating, they are able to produce quotes that include terms and prices, which are then saved in the ZQUOTATION table.

These quotes are tagged for customer evaluation and are connected to certain orders.

Use Case 6: Quotation Response by Customer

After reviewing the quotes they received, customers will communicate whether they accept or reject them. The ZQUOTATION table gets updated with their response. If they accept, they may initiate order processing, if they reject it, they may need to modify or to close the quotation.

3.2 PROJECT DESIGN AND ARCHITECTURE

The architecture of this project was designed with modularity, scalability, and maintainability in mind. We utilize a systematic tiering of data management, application logic, and user interface. By separating concerns, it allows each component to perform a specific function and to be modified independently of the system as a whole.

3.2.1 Overall Design Approach

Easy maintenance and separation of concerns is at the forefront of the layered, modular design developed for this project. Each module encapsulates various business functions, such as order management, login, registration, inquiry and quote management.

The design includes both front end and back end capabilities: the front end is comprised of SAP GUI panels that the user can see, and the back end is comprised of ABAP-based custom function modules and custom tables that govern how data will be processed and stored.

This design allows both users and administrators to have access to the presented features of the system e.g. registration, placing orders, responding to inquiries, keeping quotes, and logging in all from a single entry point. The modular design also makes it easier to allow for scale and re-use.

3.2.2 System Architecture

The project is structured using a two-tier architecture, dividing responsibilities between:

1. Presentation Layer

This layer consists of SAP GUI-based screens and forms. It serves as the interaction point for end-users—both customers and admins. Key interfaces include:

- Login and registration screens
- Product catalog view
- Order placement form
- Inquiry and quotation screens

These screens are user-friendly, integrated with input validations to ensure that data entered (like passwords, product quantities, or inquiry text) meets the required formats and constraints.

2. Application and Data Layer

This layer contains:

- Custom function modules that perform logic-based tasks such as authentication, data retrieval, ID generation, and order processing.
- Custom DDIC tables (ZCUSTOMER, ZORDERS, ZPRODUCTS, ZINQUIRY, ZQUOTATION) used for storing persistent business data.

Function modules act as the bridge between the user interface and the database, ensuring that all interactions are validated, secure, and modular. This structure supports clean code practices and centralized logic handling.

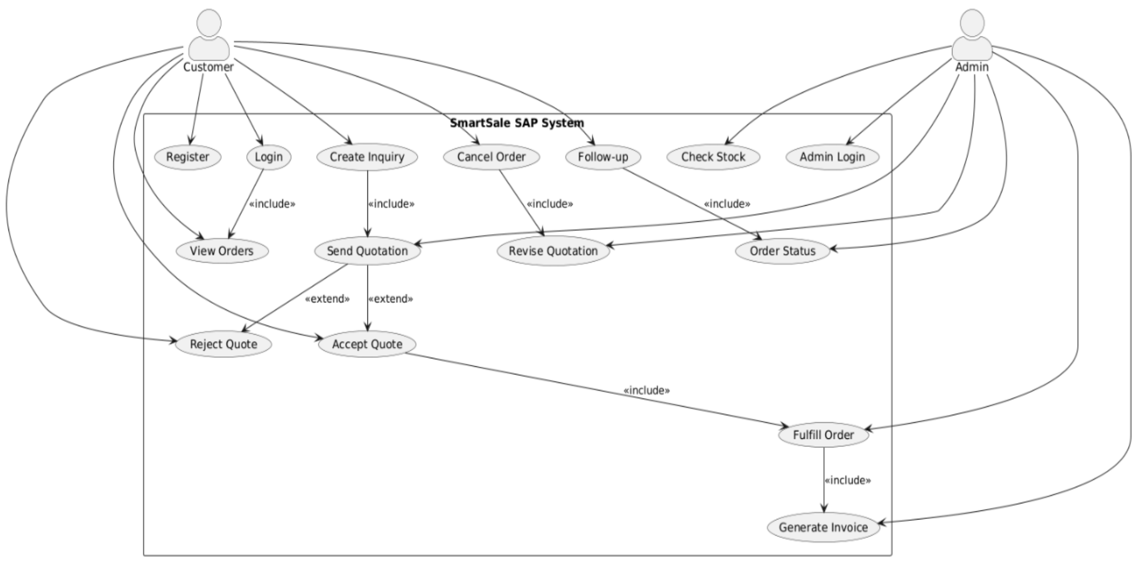


Figure 3.2.1 Use Case Diagram

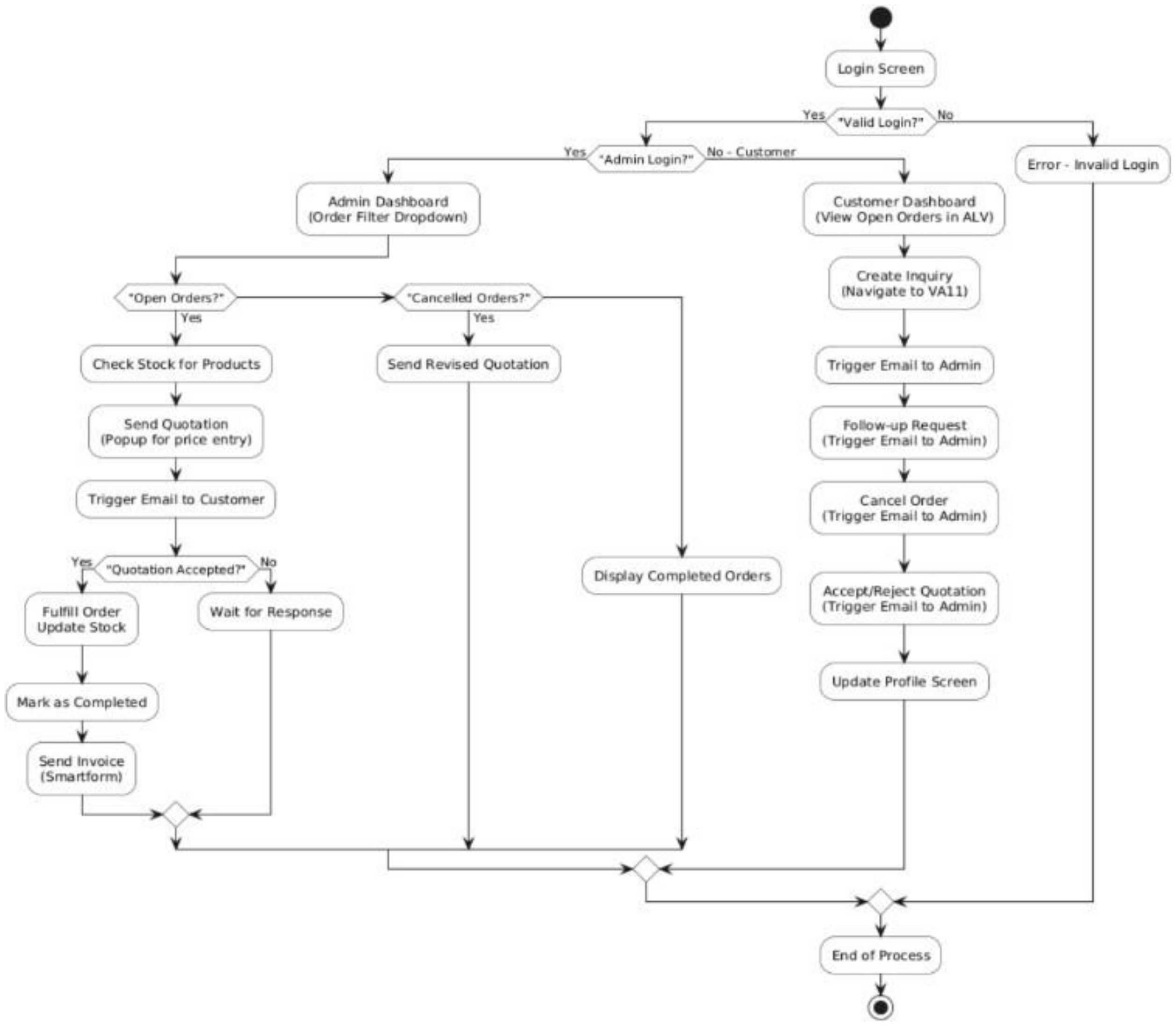

Login Screen
"Valid Login?"
Yes
No
Error - Invalid Login
"Admin Login?"
Yes
No - Customer
Admin Dashboard (Order Filter Dropdown)
Customer Dashboard (View Open Orders in ALV)
Create Inquiry (Navigate to VA11)
"Open Orders?"
Yes
"Cancelled Orders?"
Yes
Check Stock for Products
Send Revised Quotation
Trigger Email to Admin
Send Quotation (Popup for price entry)
Follow-up Request (Trigger Email to Admin)
Trigger Email to Customer
Cancel Order (Trigger Email to Admin)
"Quotation Accepted?"
Yes
No
Display Completed Orders
Fulfill Order Update Stock
Wait for Response
Accept/Reject Quotation (Trigger Email to Admin)
Update Profile Screen
Mark as Completed
Send Invoice (Smartform)
End of Process

Figure: 3.2.2 Flow Chart

3. Component-wise Design

- **Login and Authentication**

  A common login screen supports both customer and admin access. Upon submitting credentials, the system authenticates the user against the ZCUSTOMER table. Based on the role associated with the login, users are redirected to their respective dashboards. Passwords are stored in an encrypted format, and a security question mechanism supports account recovery.

- **Customer Registration**

  A dedicated registration page allows new users to sign up by entering their name, password, and selecting a security question. The system automatically generates a unique customer ID by fetching the latest entry from ZCUSTOMER and incrementing it. Inputs like password strength and field completeness are thoroughly validated before data is committed. Successful registration leads to a confirmation message and redirects the user to the login screen.

- **Order Management**

  Post-login, customers can browse available products (fetched from ZPRODUCTS) and place orders through a guided interface. Each order placed updates the ZORDERS table with a unique order ID and order details, while stock availability is checked and updated in ZPRODUCTS. The module ensures that orders cannot be placed for out-of-stock items, maintaining inventory integrity.

- **Inquiry Management**

  The system allows customers to raise inquiries about their orders through a structured form. These inquiries are recorded in the ZINQUIRY table, with clear mapping to the customer and the associated order/product. Admins access these records to respond effectively, enabling a two-way communication mechanism within the platform.

- **Quotation Processing**

  Admins can review submitted orders and inquiries and respond by generating quotations. The quotation form includes fields for specifying the quotation amount, deadline, and remarks. All quotation details are stored in the ZQUOTATION table and linked to the respective order. Customers can then log in to accept or reject the quotation, facilitating a transparent and trackable business process.

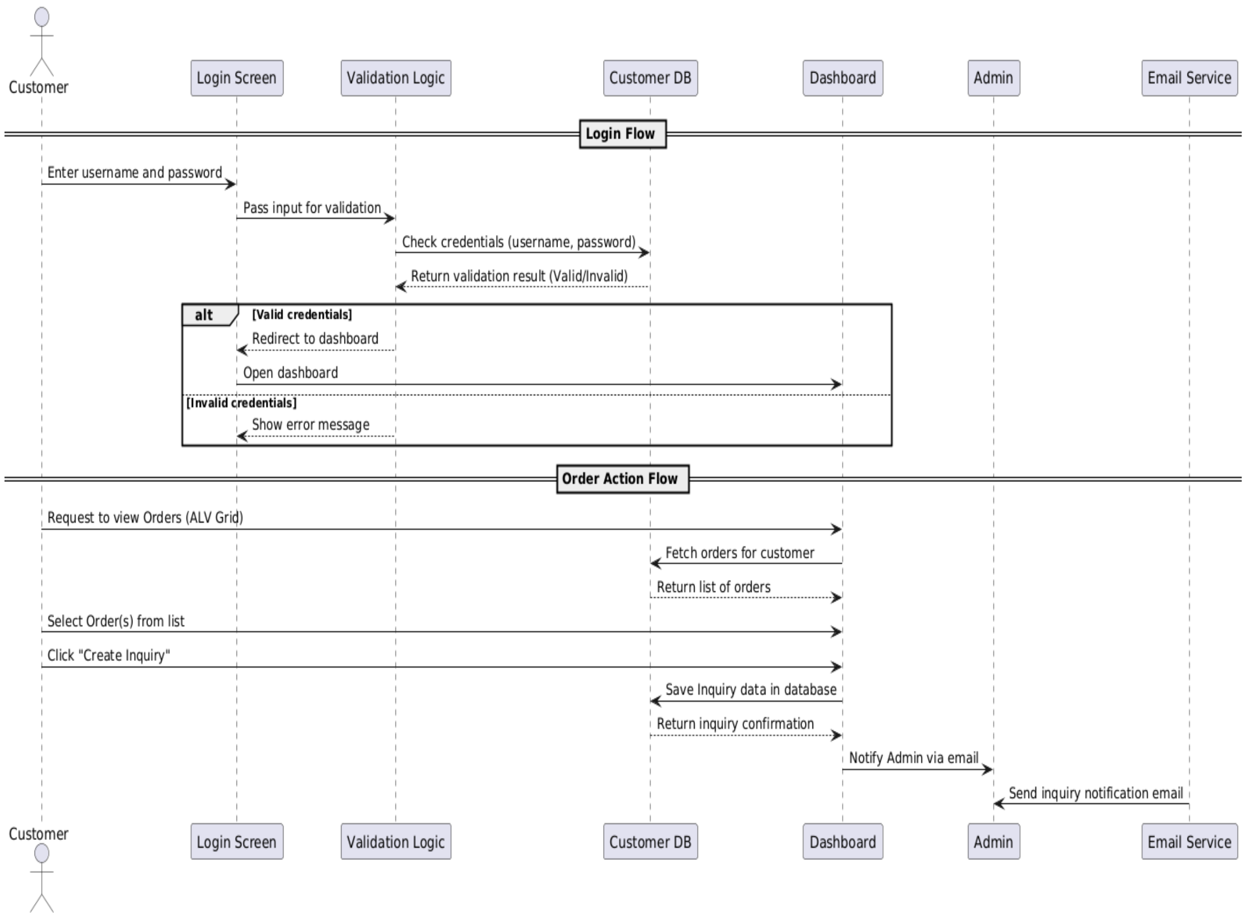

Customer
Login Screen
Validation Logic
Customer DB
Dashboard
Admin
Email Service
Login Flow
Enter username and password
Pass input for validation
Check credentials (username, password)
Return validation result (Valid/Invalid)
alt
[Valid credentials]
Redirect to dashboard
Open dashboard
[Invalid credentials]
Show error message
Order Action Flow
Request to view Orders (ALV Grid)
Fetch orders for customer
Return list of orders
Select Order(s) from list
Click "Create Inquiry"
Save Inquiry data in database
Return inquiry confirmation
Notify Admin via email
Send inquiry notification email

Figure 3.2.3 Sequence Diagram

3.2.3 Data Flow and Table Interaction

All functional components are underpinned by well-structured data flow between custom DDIC tables. Key interactions include:

- **Registration** → Inserts validated data into ZCUSTOMER.
- **Login** → Verifies credentials from ZCUSTOMER.
- **Order Placement** → Inserts data into ZORDERS, updates stock in ZPRODUCTS.
- **Inquiry Submission** → Records customer inquiries in ZINQUIRY.
- **Quotation Handling** → Admin-generated quotations are saved in ZQUOTATION and linked to respective orders.

Each user action triggers one or more backend operations, ensuring real-time consistency across the system. The use of foreign keys and ID linking ensures data traceability and smooth relational flows among entities.

3.2.4 Error Handling and Validation

Comprehensive validation mechanisms are embedded in both frontend input checks and backend logic modules:

- **Input-Level Validation:** Screens enforce rules like mandatory field checks, valid email format, password complexity, and matching re-entered passwords.
- **Backend Exception Handling:** Function modules include error-handling routines for scenarios like:
    - Invalid login credentials
    - Attempting to register with an existing username
    - Placing an order with insufficient stock
    - Accepting a quotation after expiration

In all such cases, clear error messages are displayed, and the system prevents faulty data from being processed. This ensures robustness, data integrity, and a better user experience.

3.3 DATA PREPARATION

The preparation of clean, structured, and consistent data was a foundational step in the successful implementation of the order management system. Data preparation ensured that all key functionalities—such as login authentication, product selection, order creation, inquiry management, and quotation handling—could be tested and validated in a realistic and controlled environment. It involved creating and populating master data and transactional records across multiple custom SAP tables, validating entries, and automating repeatable processes.

3.3.1 Master Data Creation

Master data refers to the static core data that remains relatively constant and is essential for driving various business processes. The following master data sets were prepared:

- **Customer Master Data (ZCUSTOMER):** A set of test customer profiles was created manually to simulate different login scenarios. Each record included:A system-generated unique customer ID, Customer name, Secure, encrypted password, Security question and answer for password recovery.These entries enabled rigorous testing of the login, registration, and authentication mechanisms and ensured that password validation and security protocols were functioning properly.

- **Product Master Data (ZPRODUCTS):** The product catalog was designed to replicate a real-world inventory. It included Product ID (unique identifier), Product name and description, Unit price, Available stock quantity.

  This data was essential for testing product availability during order placement, checking stock depletion after successful order processing, and validating order rejection in case of insufficient stock.

- **Admin Data:** For administrative functionalities, a separate set of credentials was created to access quotation and inquiry management features. Admin login testing ensured the redirection to the correct interface and access to order-level actions.

3.3.2 Transactional Test Data

To validate system operations and ensure the smooth flow of business processes, transactional data was also prepared and inserted into the corresponding custom tables:

- **Order Data (ZORDERS):** Dummy orders were inserted with various statuses (Open, Completed, Cancelled) to test order creation, viewing history, and admin-side processing. Each order was linked to a customer and referenced specific products, mimicking actual usage patterns.

- **Inquiry Data (ZINQUIRY):** Simulated customer inquiries were added, each associated with a customer and a product or order. These entries helped in testing how admins retrieve and respond to concerns raised by customers through the front-end screens.

- **Quotation Data (ZQUOTATION):** Admin-generated quotations were inserted for selected orders. These records included the proposed price, status of quotation (Accepted/Rejected), and were used to test the approval flow and system response after customer decision.

3.3.3 Data Validation and Consistency Checks

Once the master and transactional datasets were prepared, validation checks were conducted to ensure correctness and reliability:

- **ID Format and Uniqueness:** Each table entry was verified for unique and sequential identifiers to maintain data integrity and avoid conflicts during read/write operations.

- **Referential Integrity:** Dependencies between tables were carefully tested—for example, ensuring that all quotations referenced valid orders and that all inquiries originated from registered customers.

- **Logical and Business Rule Validation:** Orders were checked to ensure they did not exceed available stock, and invalid customer credentials were handled gracefully with appropriate error messages.

3.3.4 Automation and Reusability

To support repeated testing and improve development efficiency, reusable ABAP function modules and scripts were developed for automated data preparation:

- **Dynamic Data Insertion:** Functions were designed to dynamically insert new customers, products, or orders with minimal manual intervention. This was particularly useful during iterative testing phases.

- **Data Reset Utilities:** Modules were also written to clean up test records or reset table contents to a known baseline, ensuring consistency across multiple test cycles and team members.

- **Error Logging:** Any invalid inputs or failed insertions were captured using backend logging mechanisms, helping the development team identify and correct issues promptly.

3.4 IMPLEMENTATION

The implementation phase involved developing a modular and structured ABAP solution using SAP GUI, custom reports, function modules, and database tables. The backend logic and user interface were integrated through clearly defined modules and reusable components, ensuring maintainability and scalability.

3.4.1 Tools and Technologies Used

The project was developed using the following core SAP tools and technologies:

- **ABAP (Advanced Business Application Programming):**
  Used for writing backend logic, validations, and custom function modules.
- **SAP GUI:**
  Interface for executing and testing the developed programs.
- **SE11 (Data Dictionary):**
  For creating custom tables like ZCUSTOMER, ZORDERS, and ZQUOTATION.
- **SE80 (Object Navigator):**
  Central environment for managing programs, screens, and function modules.
- **SE37 (Function Builder):**
  To create reusable function modules such as login validation and order handling.

- **SE51 (Screen Painter):**
  Used for designing custom screens (login, registration, order forms).
- **SMARTFORMS:**
  For generating formatted documents such as quotations and invoices.
- **SHDB (BDC Recorder):**
  Used for automating data entry where applicable.

3.4.2 Implementation Overview

The project began with a clear understanding of what it was seeking to achieve – a SAP-based solution that would simplify the ability to place orders (positional, tarnational), register clients, generate inquiries, and control quotations. To achieve these objectives, we implemented a modular and tiered design that separated the user interface, business logic, and data storage.

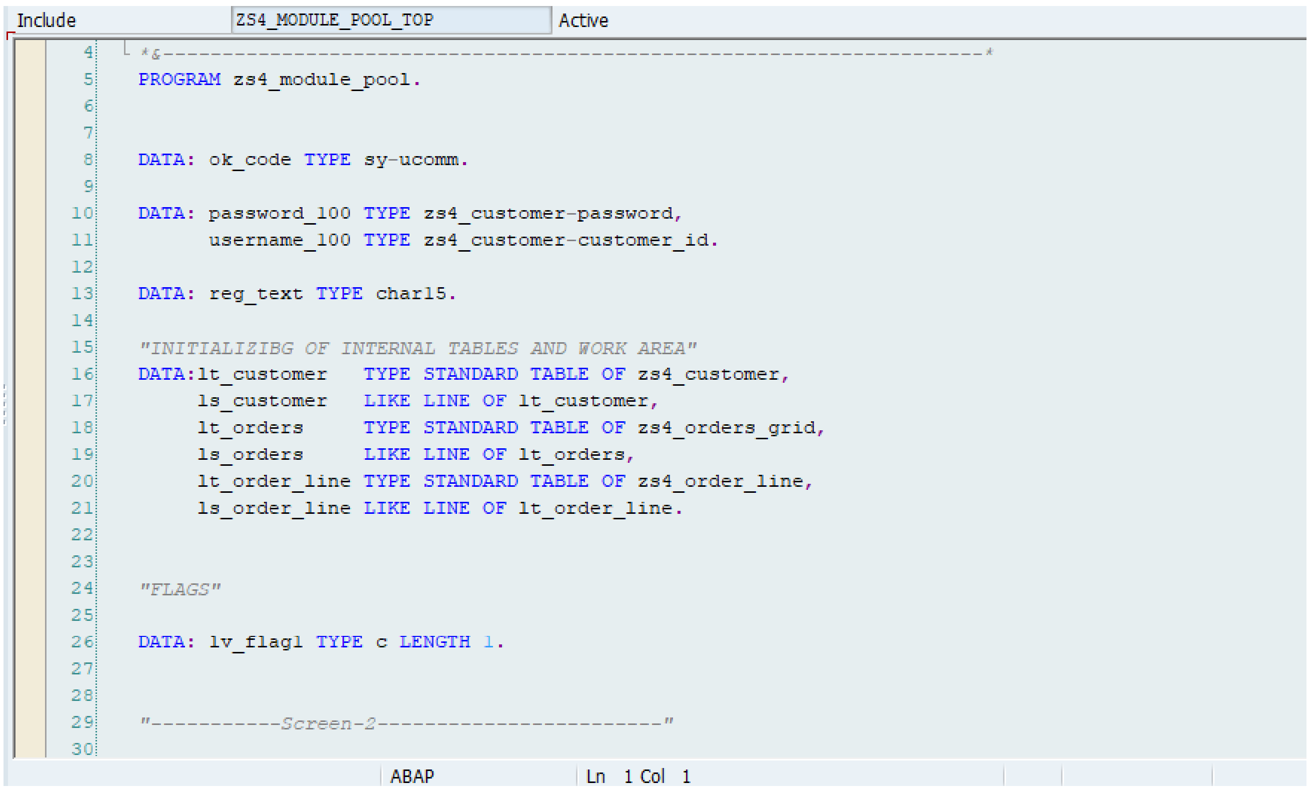
Initially, we conceived this project using a bottom-up development methodology. We created new tables (ZCUSTOMER, ZPRODUCTS, ZORDERS, ZINQUIRY, and ZQUOTATION) to create a realistic database design. Then we developed ABAP function modules that encompassed core functions such as generation of customer IDs, stock updates, order tracking, and client registration functions. Once we completed those modules, we began developing user interfaces that interfaced with the relevant back end components. Throughout the development process, we employed an iterative testing approach and strong error handling to deal with invalid inputs and inadequate supply.

- **Backend Logic and Function Modules:** ABAP function modules, which served as the foundation for the business logic, were used to manage the backend implementation. Data validation, record insertion, updates, and retrieval from bespoke DDIC tables were handled by these modules. The customer registration function, for instance, verifies the difficulty of the password, reads the most recent record in the ZCUSTOMER database to automatically produce a new customer ID, and safely inserts the verified information. To guarantee that all interactions were data-driven and consistent with regulations, comparable modules were developed for order processing, quotation generation, and login verification.
- **Frontend Interface Design**: To provide a user-friendly interface, screens were

created using SAP's screen painter. The layout of each screen, including the login, registration, product selection, order summary, inquiry, and quote screens, was intended to be user-friendly and consistent. Before invoking backend logic, users were prompted for accurate input by means of screen-level input validations. From login to customer dashboard to transactional modules like order or inquiry, the screen navigation made sense.

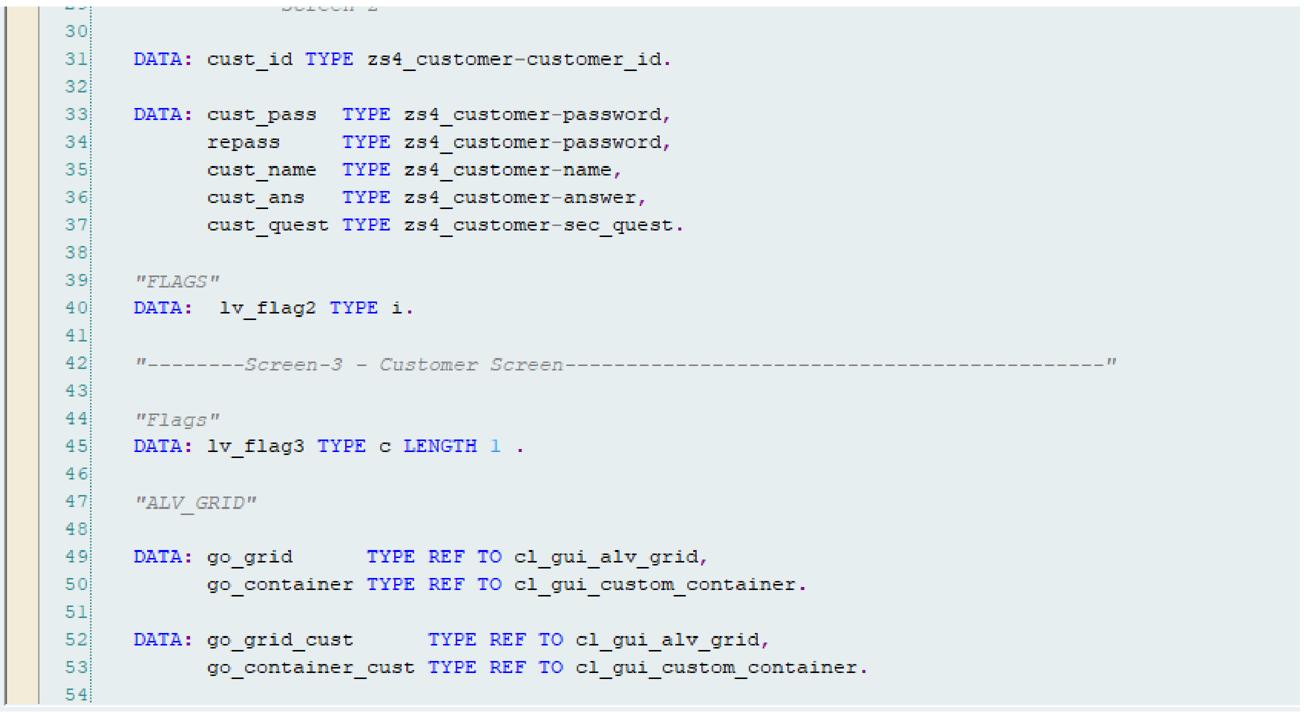
- **Validation and Error Handling:** Two tiers of validation checks were constructed. Form-level validations on the frontend made sure that inputs like erroneous data types, mismatched passwords, and empty fields were detected before processing. To handle failures on the backend, exception handling blocks were included into function modules. Users received clear error notifications, making it easy for them to comprehend and fix problems.

- **Testing and Integration:** In the development stage, every component was tested separately. The components were combined and tested end-to-end to guarantee data consistency and smooth user interaction after unit testing verified the modules were operating as intended. To confirm the system's complete operation, fake orders were conducted, sample client registrations were made, and questions were recorded. The accuracy and thoroughness of administrative contacts, such as providing quotes and monitoring order changes, were also examined.



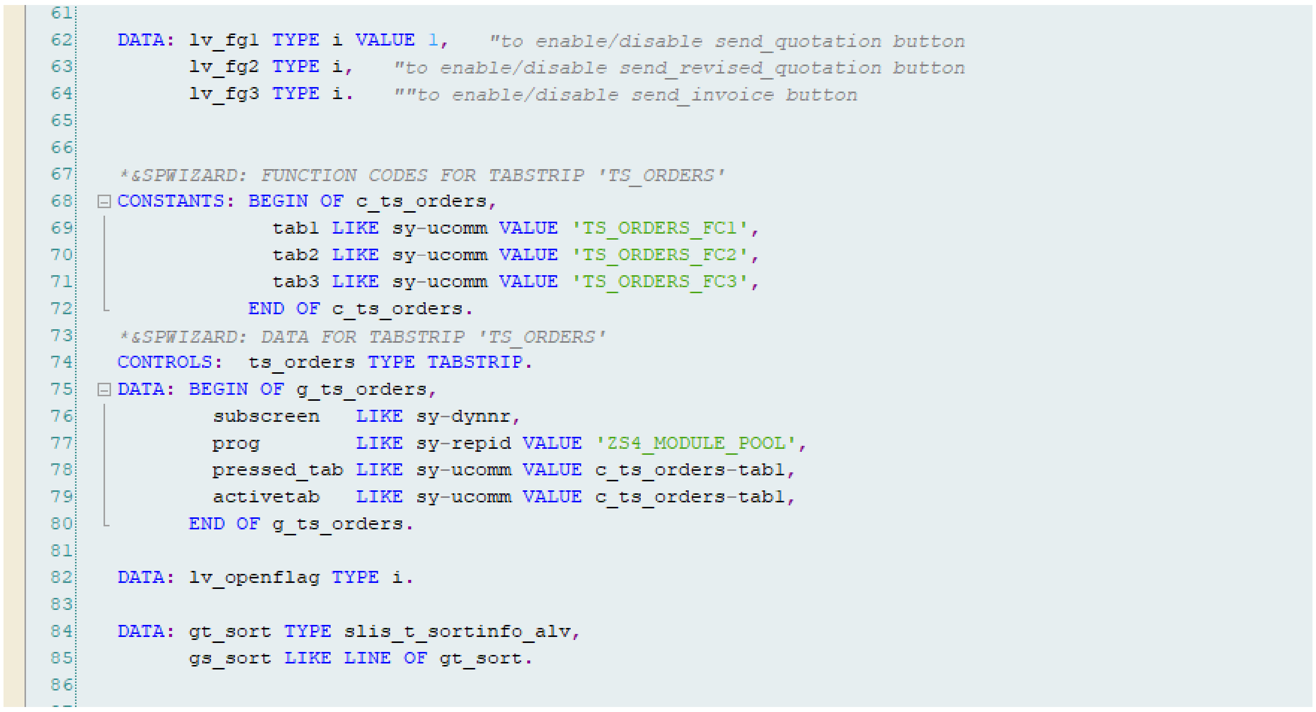
```
*&---------------------------------------------------------------------*
PROGRAM zs4_module_pool.


DATA: ok_code TYPE sy-ucomm.

DATA: password_100 TYPE zs4_customer-password,
      username_100 TYPE zs4_customer-customer_id.

DATA: reg_text TYPE char15.

"INITIALIZIBG OF INTERNAL TABLES AND WORK AREA"
DATA:lt_customer   TYPE STANDARD TABLE OF zs4_customer,
     ls_customer   LIKE LINE OF lt_customer,
     lt_orders     TYPE STANDARD TABLE OF zs4_orders_grid,
     ls_orders     LIKE LINE OF lt_orders,
     lt_order_line TYPE STANDARD TABLE OF zs4_order_line,
     ls_order_line LIKE LINE OF lt_order_line.


"FLAGS"

DATA: lv_flag1 TYPE c LENGTH 1.


"-----------Screen-2------------------------"

```

Figure 3.4.2.1 Data Declaration in TOP for Screen 1.



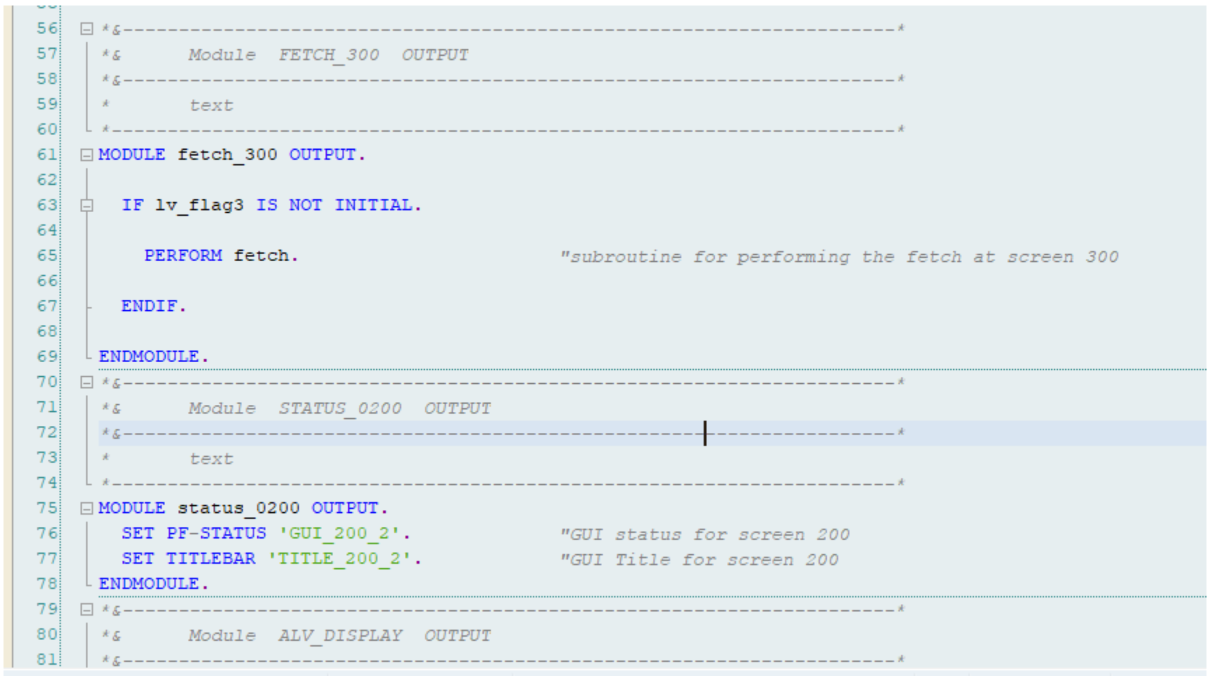
```

DATA: cust_id TYPE zs4_customer-customer_id.

DATA: cust_pass  TYPE zs4_customer-password,
      repass     TYPE zs4_customer-password,
      cust_name  TYPE zs4_customer-name,
      cust_ans   TYPE zs4_customer-answer,
      cust_quest TYPE zs4_customer-sec_quest.

"FLAGS"
DATA:  lv_flag2 TYPE i.

"--------Screen-3 - Customer Screen---------------------------------------------"

"Flags"
DATA: lv_flag3 TYPE c LENGTH 1 .

"ALV_GRID"

DATA: go_grid      TYPE REF TO cl_gui_alv_grid,
      go_container TYPE REF TO cl_gui_custom_container.

DATA: go_grid_cust      TYPE REF TO cl_gui_alv_grid,
      go_container_cust TYPE REF TO cl_gui_custom_container.

```

Figure 3.4.2.2 Data Declaration in TOP for Screen 2.



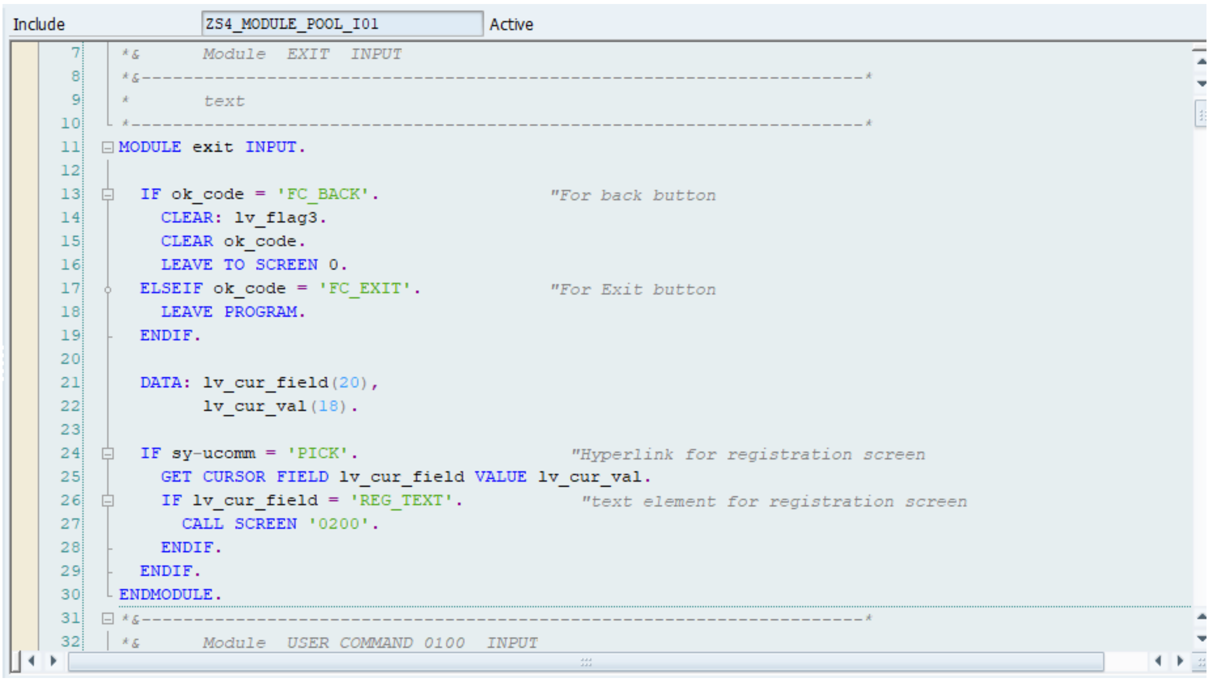
```
DATA: lv_fg1 TYPE i VALUE 1,    "to enable/disable send_quotation button
      lv_fg2 TYPE i,    "to enable/disable send_revised_quotation button
      lv_fg3 TYPE i.    ""to enable/disable send_invoice button


*&SPWIZARD: FUNCTION CODES FOR TABSTRIP 'TS_ORDERS'
CONSTANTS: BEGIN OF c_ts_orders,
             tab1 LIKE sy-ucomm VALUE 'TS_ORDERS_FC1',
             tab2 LIKE sy-ucomm VALUE 'TS_ORDERS_FC2',
             tab3 LIKE sy-ucomm VALUE 'TS_ORDERS_FC3',
           END OF c_ts_orders.
*&SPWIZARD: DATA FOR TABSTRIP 'TS_ORDERS'
CONTROLS:  ts_orders TYPE TABSTRIP.
DATA: BEGIN OF g_ts_orders,
        subscreen   LIKE sy-dynnr,
        prog        LIKE sy-repid VALUE 'ZS4_MODULE_POOL',
        pressed_tab LIKE sy-ucomm VALUE c_ts_orders-tab1,
        activetab   LIKE sy-ucomm VALUE c_ts_orders-tab1,
      END OF g_ts_orders.

DATA: lv_openflag TYPE i.

DATA: gt_sort TYPE slis_t_sortinfo_alv,
      gs_sort LIKE LINE OF gt_sort.

```

Figure 3.4.2.3: Data Declaration in TOP for Screen 3.



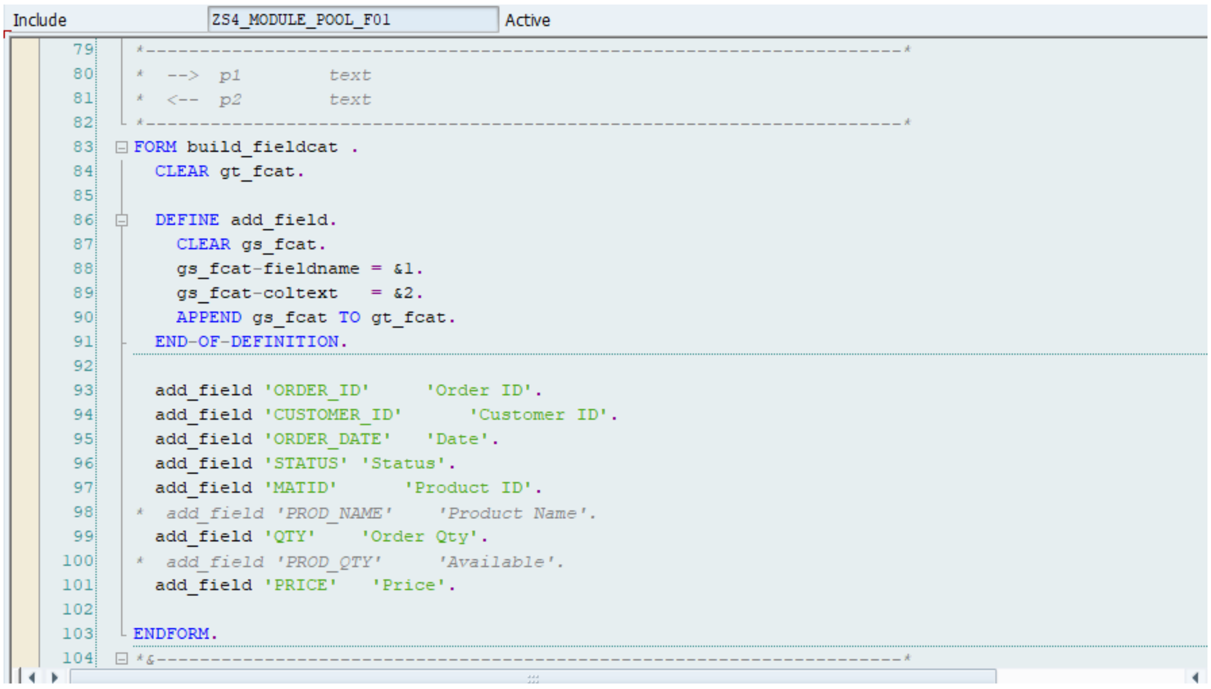
```
*&---------------------------------------------------------------------*
*&      Module  FETCH_300  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE fetch_300 OUTPUT.

  IF lv_flag3 IS NOT INITIAL.

    PERFORM fetch.                        "subroutine for performing the fetch at screen 300

  ENDIF.

ENDMODULE.
*&---------------------------------------------------------------------*
*&      Module  STATUS_0200  OUTPUT
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
MODULE status_0200 OUTPUT.
  SET PF-STATUS 'GUI_200_2'.              "GUI status for screen 200
  SET TITLEBAR 'TITLE_200_2'.             "GUI Title for screen 200
ENDMODULE.
*&---------------------------------------------------------------------*
*&      Module  ALV_DISPLAY  OUTPUT
*&---------------------------------------------------------------------*
```

Figure 3.4.2.4: PBO for Screen 1.


Include ZS4_MODULE_POOL_I01 Active

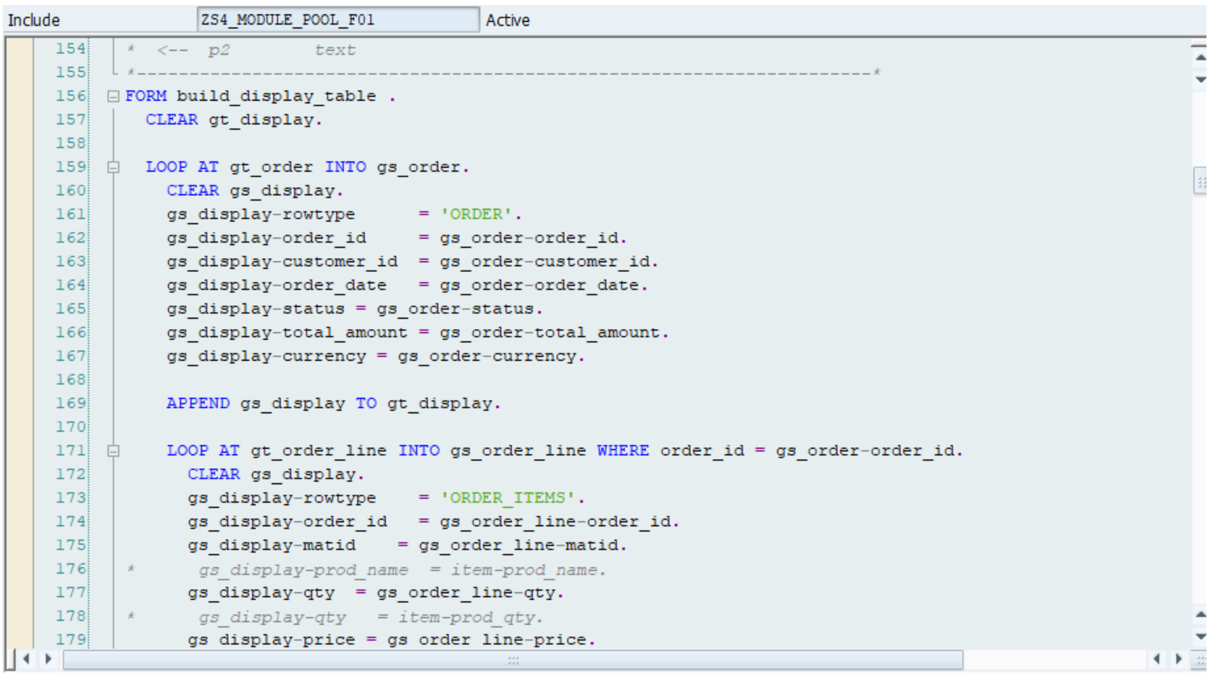
```
*&       Module  EXIT  INPUT
*&---------------------------------------------------------------------*
*        text
*----------------------------------------------------------------------*
MODULE exit INPUT.

  IF ok_code = 'FC_BACK'.                   "For back button
    CLEAR: lv_flag3.
    CLEAR ok_code.
    LEAVE TO SCREEN 0.
  ELSEIF ok_code = 'FC_EXIT'.               "For Exit button
    LEAVE PROGRAM.
  ENDIF.

  DATA: lv_cur_field(20),
        lv_cur_val(18).

  IF sy-ucomm = 'PICK'.                       "Hyperlink for registration screen
    GET CURSOR FIELD lv_cur_field VALUE lv_cur_val.
    IF lv_cur_field = 'REG_TEXT'.              "text element for registration screen
      CALL SCREEN '0200'.
    ENDIF.
  ENDIF.
ENDMODULE.
*&---------------------------------------------------------------------*
*&       Module  USER COMMAND 0100  INPUT
```

Figure 3.4.2.5: PAI for Screen 1.


Include ZS4_MODULE_POOL_F01 Active

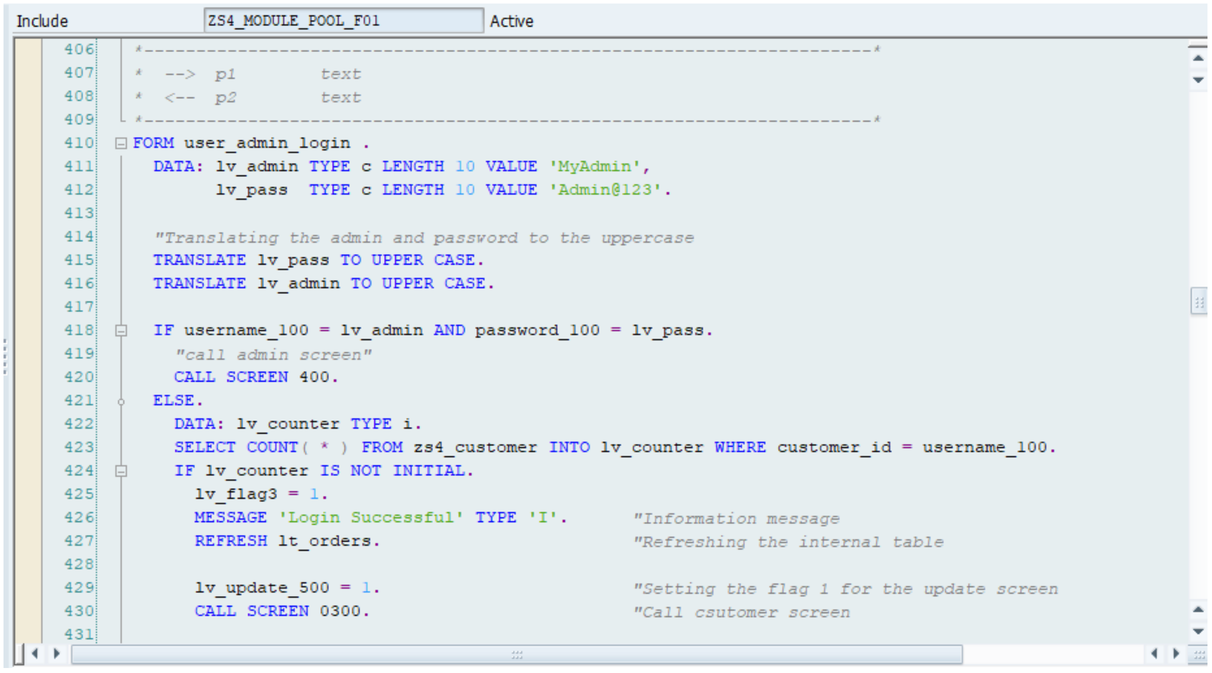
```
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM build_fieldcat .
  CLEAR gt_fcat.

  DEFINE add_field.
    CLEAR gs_fcat.
    gs_fcat-fieldname = &1.
    gs_fcat-coltext   = &2.
    APPEND gs_fcat TO gt_fcat.
  END-OF-DEFINITION.

  add_field 'ORDER_ID'      'Order ID'.
  add_field 'CUSTOMER_ID'       'Customer ID'.
  add_field 'ORDER_DATE'    'Date'.
  add_field 'STATUS'  'Status'.
  add_field 'MATID'       'Product ID'.
*  add_field 'PROD_NAME'     'Product Name'.
  add_field 'QTY'     'Order Qty'.
*  add_field 'PROD_QTY'      'Available'.
  add_field 'PRICE'    'Price'.

ENDFORM.
*&---------------------------------------------------------------------*
```

Figure 3.4.2.6: Form-Routines Build_FieldCatalog.



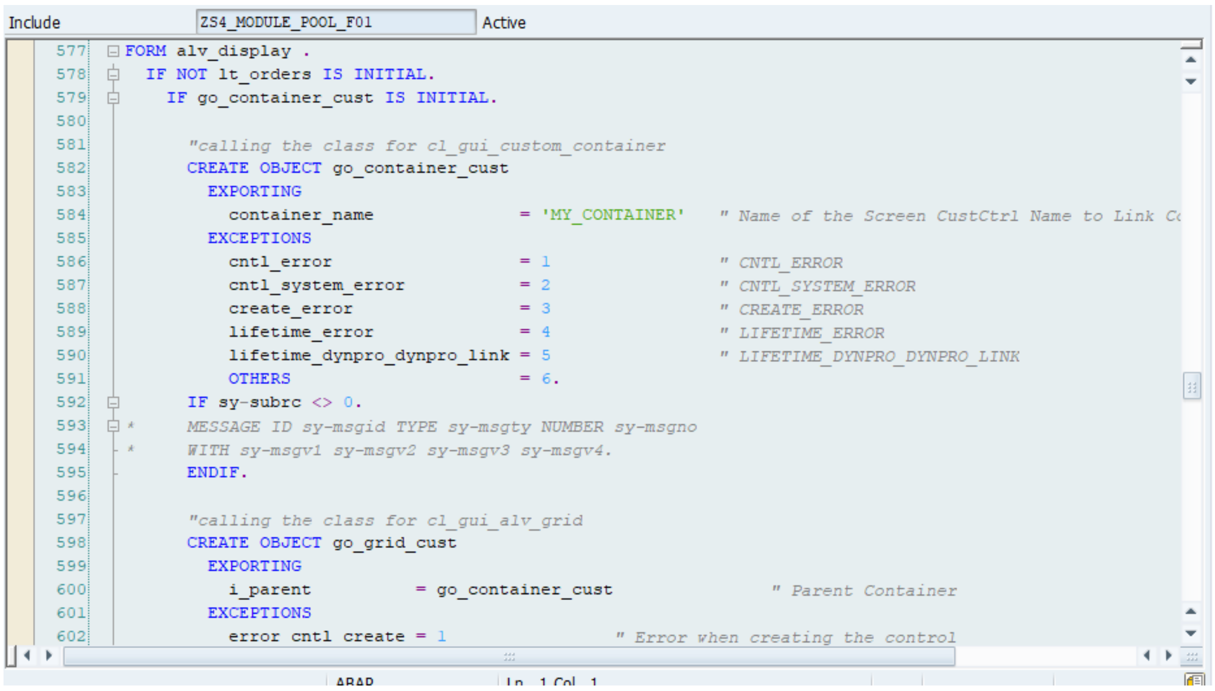
```
Include    ZS4_MODULE_POOL_F01    Active
*  <--  p2        text
*----------------------------------------------------------------------*
FORM build_display_table .
  CLEAR gt_display.

  LOOP AT gt_order INTO gs_order.
    CLEAR gs_display.
    gs_display-rowtype      = 'ORDER'.
    gs_display-order_id     = gs_order-order_id.
    gs_display-customer_id  = gs_order-customer_id.
    gs_display-order_date   = gs_order-order_date.
    gs_display-status = gs_order-status.
    gs_display-total_amount = gs_order-total_amount.
    gs_display-currency = gs_order-currency.

    APPEND gs_display TO gt_display.

    LOOP AT gt_order_line INTO gs_order_line WHERE order_id = gs_order-order_id.
      CLEAR gs_display.
      gs_display-rowtype     = 'ORDER_ITEMS'.
      gs_display-order_id    = gs_order_line-order_id.
      gs_display-matid     = gs_order_line-matid.
*      gs_display-prod_name   = item-prod_name.
      gs_display-qty   = gs_order_line-qty.
*      gs_display-qty    = item-prod_qty.
      gs_display-price = gs_order_line-price.
```

Figure 3.4.2.7: Form-Routines Build_Display_table.



```
Include    ZS4_MODULE_POOL_F01    Active
*----------------------------------------------------------------------*
*  -->  p1        text
*  <--  p2        text
*----------------------------------------------------------------------*
FORM user_admin_login .
  DATA: lv_admin TYPE c LENGTH 10 VALUE 'MyAdmin',
        lv_pass  TYPE c LENGTH 10 VALUE 'Admin@123'.

  "Translating the admin and password to the uppercase
  TRANSLATE lv_pass TO UPPER CASE.
  TRANSLATE lv_admin TO UPPER CASE.

  IF username_100 = lv_admin AND password_100 = lv_pass.
    "call admin screen"
    CALL SCREEN 400.
  ELSE.
    DATA: lv_counter TYPE i.
    SELECT COUNT( * ) FROM zs4_customer INTO lv_counter WHERE customer_id = username_100.
    IF lv_counter IS NOT INITIAL.
      lv_flag3 = 1.
      MESSAGE 'Login Successful' TYPE 'I'.       "Information message
      REFRESH lt_orders.                         "Refreshing the internal table

      lv_update_500 = 1.                         "Setting the flag 1 for the update screen
      CALL SCREEN 0300.                          "Call csutomer screen
```

Figure 3.4.2.8: Form-Routines Module for User_Admin_Login.

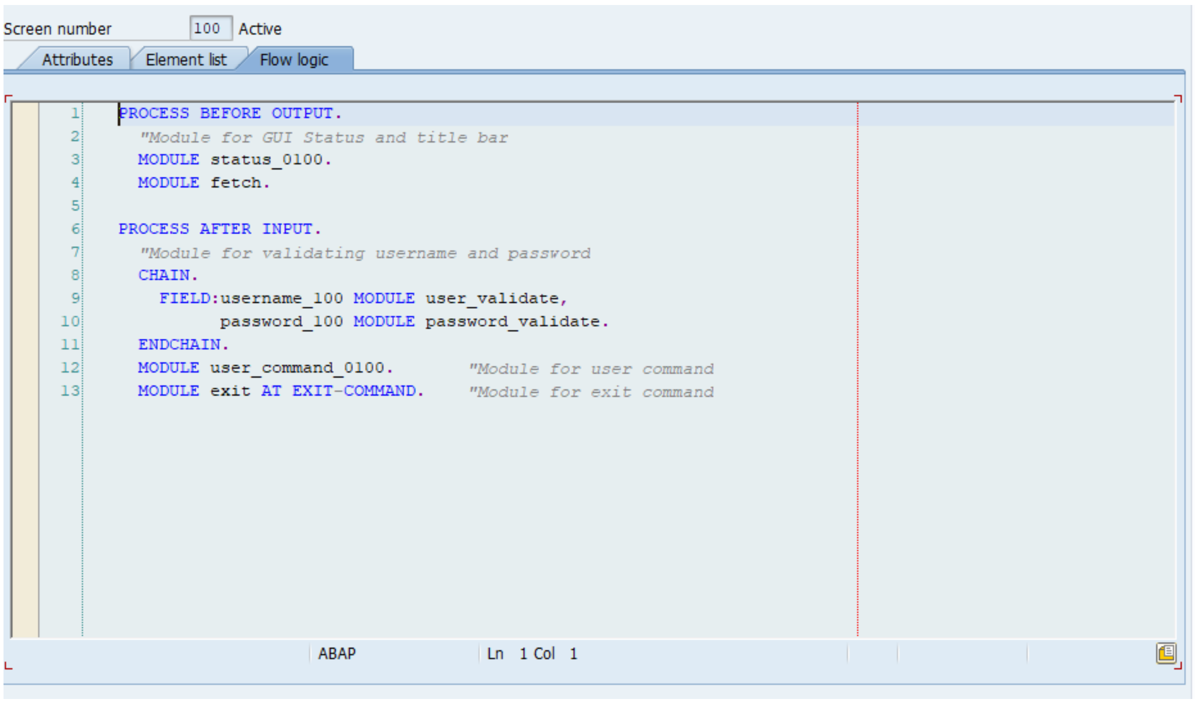


Include ZS4_MODULE_POOL_F01 Active

```
FORM alv_display .
  IF NOT lt_orders IS INITIAL.
    IF go_container_cust IS INITIAL.

      "calling the class for cl_gui_custom_container
      CREATE OBJECT go_container_cust
        EXPORTING
          container_name              = 'MY_CONTAINER'   " Name of the Screen CustCtrl Name to Link Co
        EXCEPTIONS
          cntl_error                  = 1                " CNTL_ERROR
          cntl_system_error           = 2                " CNTL_SYSTEM_ERROR
          create_error                = 3                " CREATE_ERROR
          lifetime_error              = 4                " LIFETIME_ERROR
          lifetime_dynpro_dynpro_link = 5                " LIFETIME_DYNPRO_DYNPRO_LINK
          OTHERS                      = 6.
      IF sy-subrc <> 0.
*       MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
*       WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
      ENDIF.

      "calling the class for cl_gui_alv_grid
      CREATE OBJECT go_grid_cust
        EXPORTING
          i_parent          = go_container_cust              " Parent Container
        EXCEPTIONS
          error_cntl_create = 1                   " Error when creating the control
```

ABAP Ln 1 Col 1

Figure 3.4.2.9: Form-Routines Module for ALV Display.



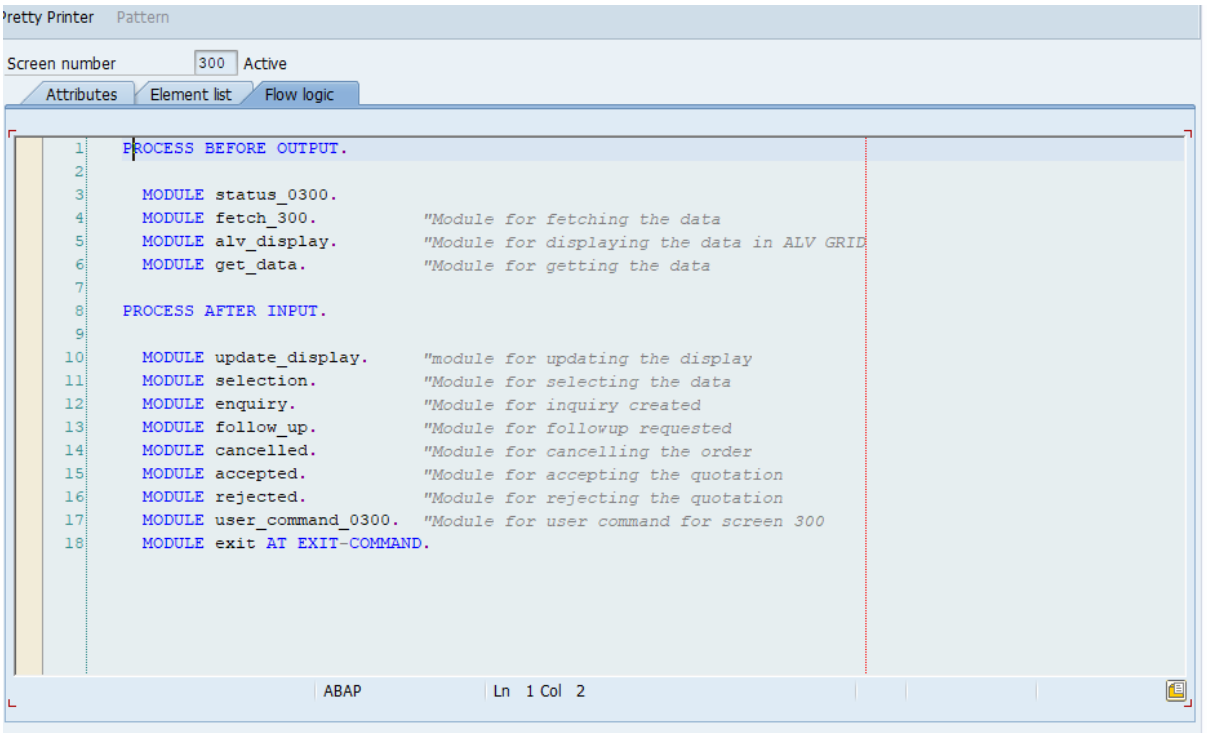
Screen number 100 Active

Attributes | Element list | Flow logic

```
PROCESS BEFORE OUTPUT.
  "Module for GUI Status and title bar
  MODULE status_0100.
  MODULE fetch.

PROCESS AFTER INPUT.
  "Module for validating username and password
  CHAIN.
    FIELD:username_100 MODULE user_validate,
          password_100 MODULE password_validate.
  ENDCHAIN.
  MODULE user_command_0100.     "Module for user command
  MODULE exit AT EXIT-COMMAND.  "Module for exit command
```

ABAP Ln 1 Col 1

Figure 3.4.2.10: Screen 100.



```
PROCESS BEFORE OUTPUT.

  MODULE status_0300.
  MODULE fetch_300.            "Module for fetching the data
  MODULE alv_display.          "Module for displaying the data in ALV GRID
  MODULE get_data.             "Module for getting the data

PROCESS AFTER INPUT.

  MODULE update_display.       "module for updating the display
  MODULE selection.            "Module for selecting the data
  MODULE enquiry.              "Module for inquiry created
  MODULE follow_up.            "Module for followup requested
  MODULE cancelled.            "Module for cancelling the order
  MODULE accepted.             "Module for accepting the quotation
  MODULE rejected.             "Module for rejecting the quotation
  MODULE user_command_0300.    "Module for user command for screen 300
  MODULE exit AT EXIT-COMMAND.
```

Figure 3.4.2.11: Screen 300.

3.5 KEY CHALLENGES

We ran across a number of technical and design-related issues when creating the SAP Sales & Distribution platform for SmartSale Ltd. Effectively addressing these improved the system's resilience and made a substantial contribution to our comprehension of SAP development methodologies.

1. Specifying a Scalable Database Structure: One of the first challenges we took to solve was to create a unique set of tables that could persuasively store data about customers, products, orders, inquiries, and quotes whilst maintaining relationship integrity. To rectify the situation, we took to SE11 to define the normalized tables and we created relationships between the primary and foreign keys to ensure data integrity and accessibility.

2. Managing Multi-Screen Navigation: Creating seamless transitions between the module pool's various user interfaces was a major challenge. Consistent session handling was necessary for every action, including accessing the inquiry creation panel from the order list or navigating from the login screen to a customized dashboard. We had to manage temporary variables across screens, gracefully handle exceptions, and maintain user context. Conditional validations in PBO/PAI, state management across program modules, and meticulous screen call sequencing using CALL SCREEN were used to achieve this.

3. Dynamic Form Generation: Creating SmartForms that could handle custom headers, varying order sizes, and a variety of customer data presented another difficulty. Clear, expertly formatted forms that could be exported as PDFs were required. They also needed to be connected to the email dispatch system. This required exact control over output handling, page formatting, and data retrieval.

4. Generating SmartForms (Quotations and Invoices): It was new to learn and use SMARTFORMS for dynamic document generation. Formatting problems and data binding errors were among the difficulties. We were able to develop well-organized templates for quotes and invoices that precisely retrieved data from custom tables with the aid of SAP documentation and practice.

5. Email Integration and Notification Logic: Using the CL_BCS framework, we integrated automated email functionality to improve workflow communication. Complexity was added, though, by dynamically creating customized messages, attaching documents created by SmartForm, and making sure that messages were sent only when necessary. Robust error-handling procedures and fallback logic were required to handle exceptions like missing email addresses, invalid content, or server connectivity problems.

6. Capturing User Selections in ALV Grids: Orders and quotations were among the many records that were displayed using interactive ALV grids. Enabling users to select multiple rows and take action only on those particular records was one of the major challenges here. This necessitated precisely executing backend logic, extracting identifiers, and capturing selected rows—all without affecting unselected entries. The GET_SELECTED_ROWS method and ALV event handling were used in our implementation, along with validation procedures to ensure that the selections were accurate.

CHAPTER 4: TESTING

We implemented a systematic and comprehensive manual testing approach to guarantee the seamless operation of smartSale Ltd.'s SAP-based Sales & Distribution system. Both the admin and customer modules underwent testing, with a particular emphasis on confirming data integrity, user interface behavior, and functionality. Prior to deployment, logical mistakes, inconsistent user interfaces, and possible data flow breakdowns were to be found and fixed.

4.1 TESTING STRATEGY

- **Customer Module Manual Testing:** By resembling actual user situations like new registrations, logging in with both valid and invalid credentials, and placing product orders, the customer module was manually tested. The dynamic generation and duplication-free storage of customer IDs in the ZCUSTOMER table were confirmed by the testers. The login module was validated to guarantee correct redirection and access controls, and the registration process was examined for input validations. We also tested the feature that allowed customers to browse products, add items to their carts, and complete orders. In order to make sure that the right error messages and responses were triggered, testing included scenarios such as stock unavailability, placing multiple orders, and invalid product IDs.

- **Admin Module Manual Testing:** To make sure admin users could view, manage, and reply to customer inquiries, manual testing was done for the admin module. By choosing pending orders and creating quotations using SAP SmartForms, the quotation generation feature was evaluated. We confirmed that the generated quotations were saved in the ZQUOTATION table and included accurate dynamic content, including order IDs, customer names, and pricing information. The CL_BCS class was used to simulate successful and unsuccessful delivery scenarios in order to validate email delivery. In order to verify that changes were reflected in both the user interface and backend databases, admin users were also tested for their capacity to reject or accept orders and update statuses appropriately.

- **Customer Dashboard Functional Testing:** Every interactive feature of the customer dashboard, including the ability to track order status, access inquiry forms, review quotations, and log out, was thoroughly tested. To make sure that the session state was maintained and pertinent data was correctly passed, the navigation between these screens was verified. Multiple back-to-back operations and screen refreshes were among the conditions used to test the dashboard's responsiveness and input handling.

- **Admin Dashboard Functional Testing:** The functional testing of the admin dashboard included confirming the inquiry listing, search filters, workflows for approving quotes, and the creation of SmartForms. By choosing particular entries and verifying corresponding backend changes, the dynamic selection and updating of rows in ALV grids were tested. Functional tests verified that the system stopped invalid actions on orders that had already been processed and duplicate quotations.

- **SmartForm Output Testing**: To verify the dynamic rendering of content, SmartForms were tested by simulating orders of different sizes and complexity. The forms' alignment, internal table data mapping, and PDF formatting were all examined. We also tested the integrated logic's capacity to turn SmartForms into spool requests and send them as email attachments.

- **Database Table Validation**: In transaction SE16, manual inspections were used to verify all custom tables, including ZCUSTOMER, ZPRODUCTS, ZORDERS, ZINQUIRY, and ZQUOTATION. To verify that frontend operations and backend persistence were properly integrated, insert, update, and delete operations were tested through the application interface, and changes were recorded in the corresponding database entries.

4.1.1 Tools Used

To ensure the functional accuracy, stability, and user readiness of the Sales and Distribution portal developed in SAP for smartSale Ltd., a variety of SAP-native tools were employed during testing. These tools enabled precise verification of module logic, user interface behavior, data integrity, and backend workflows. The following tools were central to the testing process:

- **SE80 – Object Navigator:** SE80 was essential for managing and testing all development objects in one place. Programs, Smartforms, function modules, and screen elements were all accessible to testers through SE80. It facilitated screen navigation, allowed testers to replicate real-world development environment processes, such as order submission, quotation handling, and login, and verified screen logic (PBO and PAI). It also simplified screen-to-screen navigation troubleshooting.
  This transaction code was particularly important for testing and separating different function modules. For example, the modules responsible for order processing, dynamic ID generation during registration, customer login validation, and quotation creation were independently tested using SE37. By passing test input values and evaluating the output, errors could be found and corrected prior to integrating these functions into the larger system.

- **SE38 / SA38 – Report Execution and Output Testing:** Custom ABAP reports generated as part of the backend processing were tested using SE38 and SA38. By using these transaction codes, testers were able to run the reports independently from the GUI screens and confirm that the data was being retrieved, processed, and presented accurately. By validating output formatting, such as order summaries and error logs, these were also used to confirm that generated outputs adhered to business requirements.
- **SAP Debugger (/h):** The SAP Debugger was a crucial tool for tracking logic step-by-step, especially for the PBO (Process Before Output) and PAI (Process After Input) modules. Every time there was a logical error, an unexpected value, or strange screen behavior, the debugger was launched using the command /h. This allowed testers to look at variable values, conditional paths, and field states while the test was running. It was especially useful in resolving issues with navigation, data transfer between screens, and validation logic.

- **ALV Grid Debugging and Display Tools**: Testing was primarily concerned with confirming selection and interactivity because the project made extensive use of ALV (ABAP List Viewer) Grids to display order lists, quotations, and inquiries. The display output was checked to make sure all rows, fields, and selections behaved as intended using ALV event handlers. In order to prevent inaccurate data from being transmitted or carried out in backend operations like cancellation or quotation acceptance, special attention was paid to how multi-row selections were captured and processed.

- **Smartform Preview and Testing Tools** : Smartform testing tools were used to verify the correctness and layout of dynamically generated documents, like invoices and quotes. These made it possible to preview the PDF output produced for every test order and confirm conditional sections (like item lines or totals) and dynamic field mapping and layout consistency. Additionally, the correct Smartform was attached as a PDF to every triggered message in order to test the integration with email modules.

- **Workflow-Builder**
  Custom business workflows related to the order lifecycle were modeled and tested using the Workflow Builder (SWDD). For example, a workflow event was set off to send an email to the customer or the administrator informing them when a customer placed an order or received a quote. In order to make sure that the right procedures were followed and no steps were skipped, testing scenarios with SWDD included assigning tasks to users. Additionally, it assisted in confirming message customization for various workflow branches and escalation handling. This made sure that every step of the process—from creating the order to confirming it—was timely, traceable, and compliant with company policies.

4.2 TEST CASE AND OUTCOMES

Thorough testing was conducted to evaluate the system’s robustness and correctness across different modules. Below is a detailed description of key test scenarios along with their intended objectives, input conditions, and observed results.

1. Login Functionality Validation

Test Case 1: Admin Login with Valid Credentials

- **Objective:** Ensure that the system correctly authenticates the admin user.
- **Input:** Username: MyAdmin, Password: Admin@123
- **Expected Outcome:** Admin is successfully logged in and redirected to the admin interface with a welcome message.
- **Actual Outcome:** Admin login succeeded; dashboard loaded as expected.

Test Case 2: Customer Login with Valid Credentials

- **Objective:** Confirm that customers with registered credentials can access their account.
- **Input:** Valid customer username and password stored in the database.
- **Expected Outcome:** Login is successful, and the customer screen displaying their orders in ALV format is shown.
- **Actual Outcome:** Customer successfully logged in; order view displayed correctly.

Test Case 3: Invalid Username Attempt

- **Objective:** Verify system response when a non-existent username is used.
- **Input:** Random username with any password
- **Expected Outcome:** Login fails with an "Invalid username or password" message.
- **Actual Outcome:** Authentication blocked; appropriate error message shown.

Test Case 4: Incorrect Password Attempt

- **Objective:** Ensure correct handling of incorrect password for valid users.
- **Input:** Valid username with incorrect password
- **Expected Outcome:** System denies access with an error message.
- **Actual Outcome:** Login prevented; error displayed.

Test Case 5: Submission with Empty Fields

- **Objective:** Validate input field checks for empty credentials.
- **Input:** Blank username and/or password fields
- **Expected Outcome:** Prompt shown asking the user to fill in both fields.
- **Actual Outcome:** Input validation triggered; login restricted.

2. Registration Screen Field Validations

Test Case 1: Complete and Valid Input

- **Objective:** Ensure successful registration when all fields are correctly filled.
- **Input:** Customer name, matching passwords, selected security question and answer
- **Expected Outcome:** New customer entry is stored, and confirmation message appears.
- **Actual Outcome:** Registration completed and user redirected to login.

Test Case 2: Missing Customer Name

- **Objective:** Validate input checking when the name field is left empty.
- **Input:** Blank name with other fields correctly filled
- **Expected Outcome:** Registration blocked; message prompts for customer name.
- **Actual Outcome:** Error message displayed; registration halted.

Test Case 3: Missing Password Fields

- **Objective:** Check for empty password input handling.
- **Input:** Password and confirmation fields left empty
- **Expected Outcome:** System alerts user to fill in password fields.
- **Actual Outcome:** Warning shown; registration not allowed.

Test Case 4: Passwords Do Not Match

- **Objective:** Ensure matching of password fields is enforced.
- **Input:** Password and confirm password fields with different values
- **Expected Outcome:** System shows mismatch error and clears the password fields.
- **Actual Outcome:** Validation triggered; registration stopped.

Test Case 5: Security Question Not Selected

- **Objective:** Validate whether the user selects a security question during registration.
- **Input:** All fields filled except security question
- **Expected Outcome:** Error message prompts selection of a security question.
- **Actual Outcome:** Prompt displayed; registration paused.

3. Inquiry Creation and Email Notification

Test Case 1: Create Inquiry with Valid Details

- **Objective:** Validate successful inquiry creation and email notification.
- **Input:** Logged-in customer selects order and submits an inquiry with valid data
- **Expected Outcome:** Inquiry saved in SAP; email sent to admin.
- **Actual Outcome:** Inquiry recorded successfully; email delivered.

Test Case 2: Inquiry Without Order Selection

- **Objective:** Ensure order selection is required for inquiry.
- **Input:** No order selected; user clicks Create Inquiry
- **Expected Outcome:** Warning message: "Please select an order"
- **Actual Outcome:** Message displayed; inquiry creation prevented.

Test Case 3: Email Sending Failure (Simulated)

- **Objective:** Observe behavior when email server is not reachable.
- **Input:** Inquiry submitted with email service intentionally disabled
- **Expected Outcome:** Inquiry saved; error message shown for failed email
- **Actual Outcome:** Email failed gracefully; inquiry saved

Test Case 4: Missing Mandatory Fields on Inquiry Screen

- **Objective:** Ensure SAP field-level validations work during inquiry.
- **Input:** Fields like quantity or material left blank
- **Expected Outcome:** SAP standard error prompts user to complete mandatory data
- **Actual Outcome:** Fields highlighted; inquiry not submitted

Test Case 5: Multiple Inquiries in One Session

- **Objective:** Check system stability during continuous inquiry submissions
- **Input:** User creates 3–4 inquiries in a single session
- **Expected Outcome:** Each inquiry submitted separately; separate emails sent

- **Actual Outcome:** All inquiries processed; emails sent successfully

Test Case 6: Missing Admin Email Configuration

- **Objective:** Ensure proper handling when admin email is not configured
- **Input:** Remove admin email from settings and submit inquiry
- **Expected Outcome:** Inquiry saved, but email not sent; proper message displayed
- **Actual Outcome:** System warned about missing email; no crash occurred

4. Admin Operations: Quotation & Stock Check

Test Case 1: Valid Stock Check

- **Objective:** Confirm stock is checked for a selected order correctly
- **Input:** Admin selects an order with available stock for items
- **Expected Outcome:** Message confirms all items are in stock
- **Actual Outcome:** Stock availability confirmed; process allowed to proceed

Test Case 2: Stock Shortage Identified

- **Objective:** Test system response when requested quantity exceeds availability
- **Input:** Admin selects an order with insufficient stock
- **Expected Outcome:** Message lists out-of-stock items; action halted
- **Actual Outcome:** Shortage flagged; fulfillment blocked

Test Case 3: Quotation Generation

- **Objective:** Validate quotation creation and customer notification
- **Input:** Admin selects an order and inputs prices
- **Expected Outcome:** Quotation created and email sent to customer
- **Actual Outcome:** Quotation processed; notification received by customer

CHAPTER 5. RESULTS AND EVALUATION

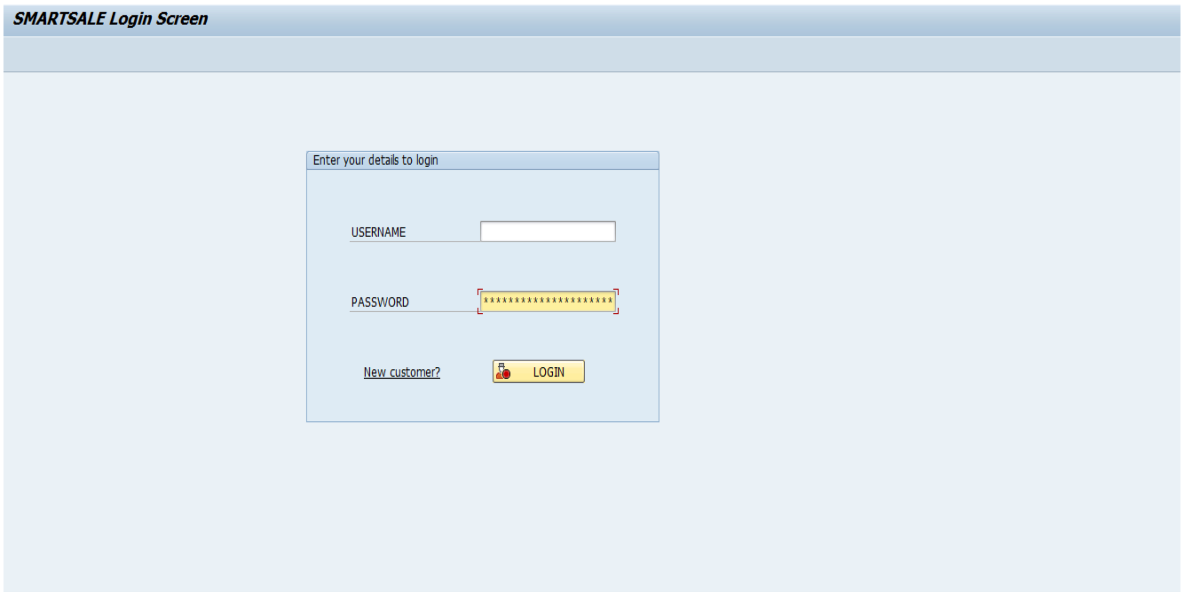
5.1 RESULTS

Using ABAP, the developed SmartSale Sales & Distribution solution was successfully deployed in the SAP environment, meeting all project goals as specified in the problem statement. ALV reports, custom SAP tables, Smart Forms, and the custom module pool program all functioned flawlessly across a range of business processes. With appropriate field-level validations, role-specific navigation, and secure access, the login and registration modules operated effectively. By accurately identifying valid and invalid inputs, the system made sure that only authorized users—either customers or administrators—could continue. No system failure occurred as invalid login attempts were handled politely through message prompts.

Through a dynamic ALV Grid that supported row-wise selection and real-time interaction, the customer dashboard gave users access to their active orders. From this screen, customers could ask questions. The system used the standard SAP email class CL_BCS to automatically send the administrator an email after an inquiry was submitted. After testing all related validations, such as missing data, multiple inquiry submissions, and missing admin email configuration, each instance showed proper system behavior.

On the administrative end, functions like quotation generation and stock validation operated as intended in a variety of situations. The backend table values were used by the system to accurately display stock availability, and quotations could only be generated when there was enough stock. Meaningful alerts were displayed in the event of a shortage, avoiding improper processing. To ensure a smooth communication flow, automated emails were sent to the relevant customers as soon as quotations were submitted.

The system performed as anticipated for every use case that was specified during the testing phase. Important user interface elements like pop-up messages, ALV Grids, and Smart Forms improved user experience and provided clarity in transactional procedures. Error handling was implemented robustly, and navigating between various custom screens was seamless.


SMARTSALE Login Screen

Enter your details to login

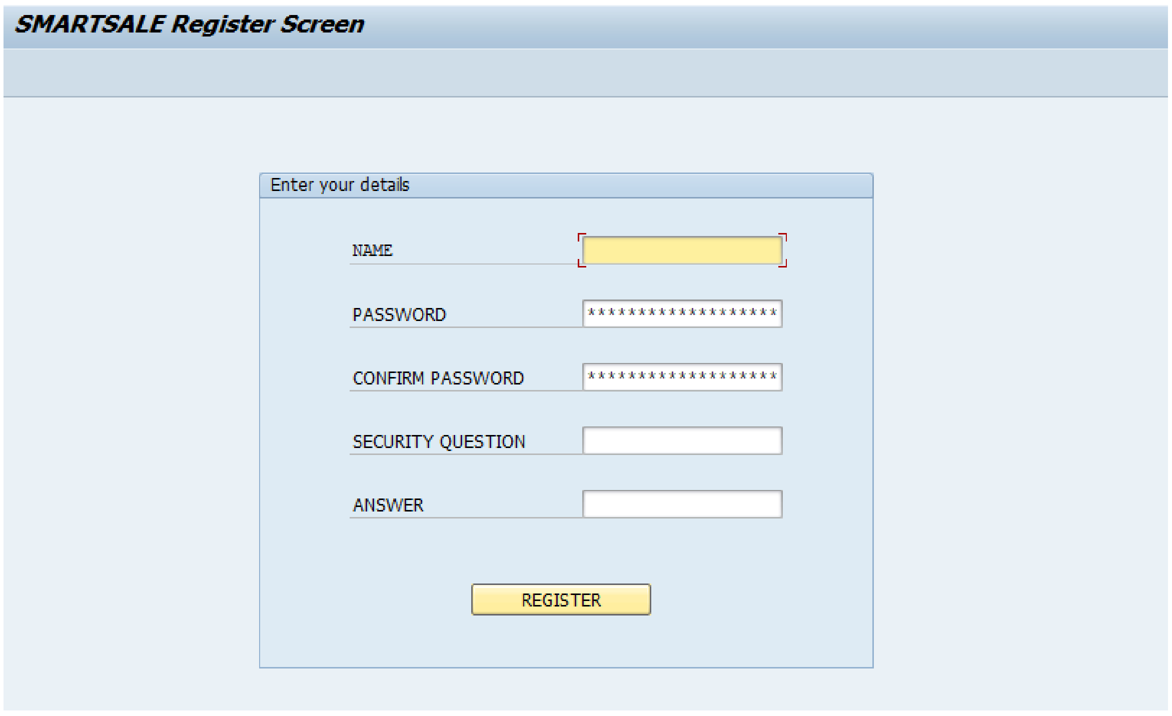
USERNAME

PASSWORD ******************

New customer?

LOGIN

Figure 5.1: Login Screen


SMARTSALE Register Screen

Enter your details

NAME

PASSWORD ******************

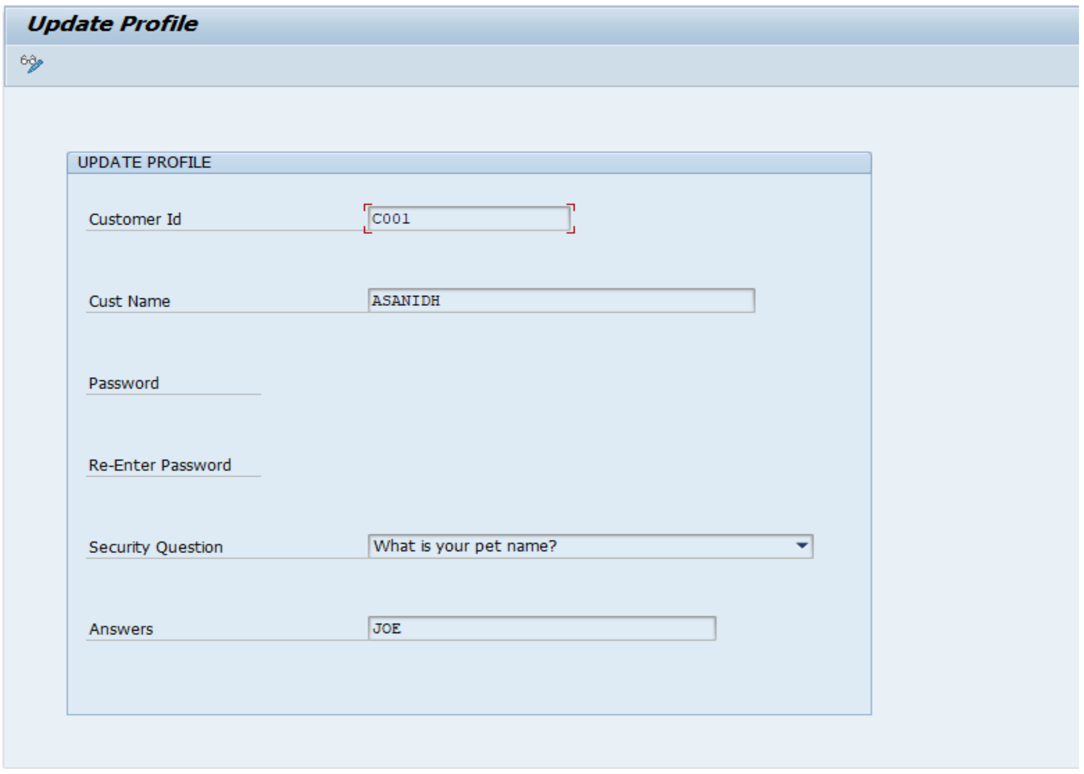
CONFIRM PASSWORD ******************

SECURITY QUESTION

ANSWER

REGISTER

Figure 5.2: Register Screen

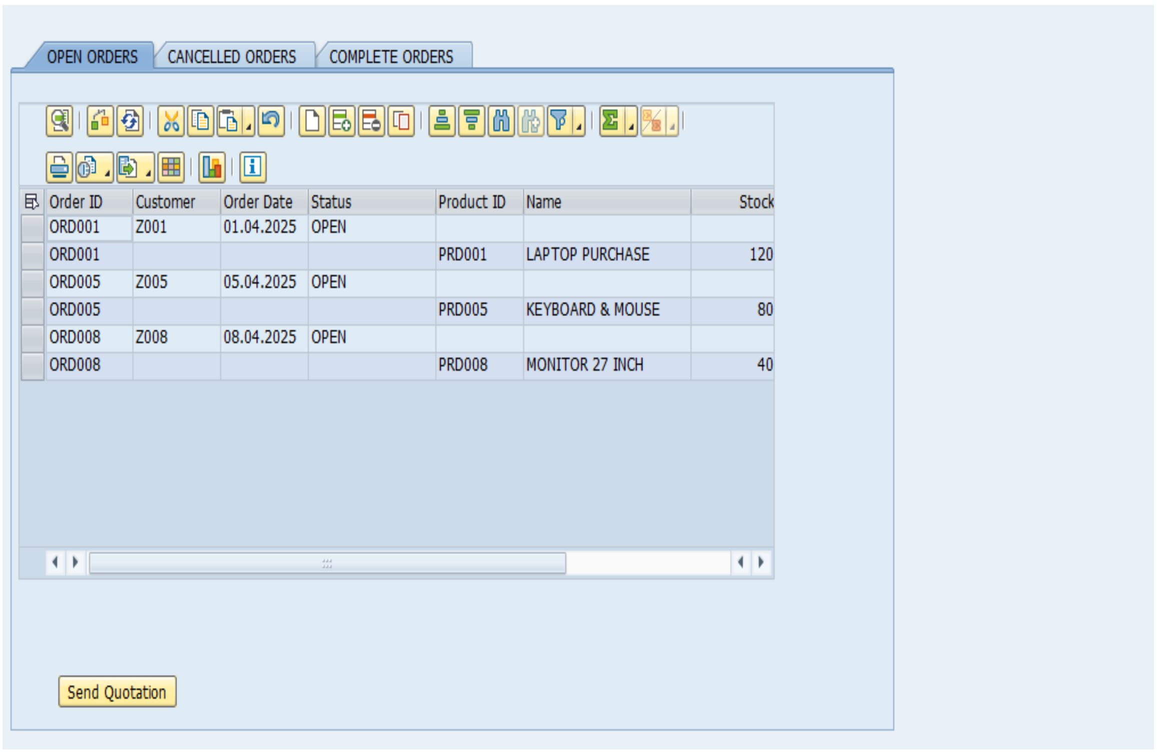


Update Profile

UPDATE PROFILE

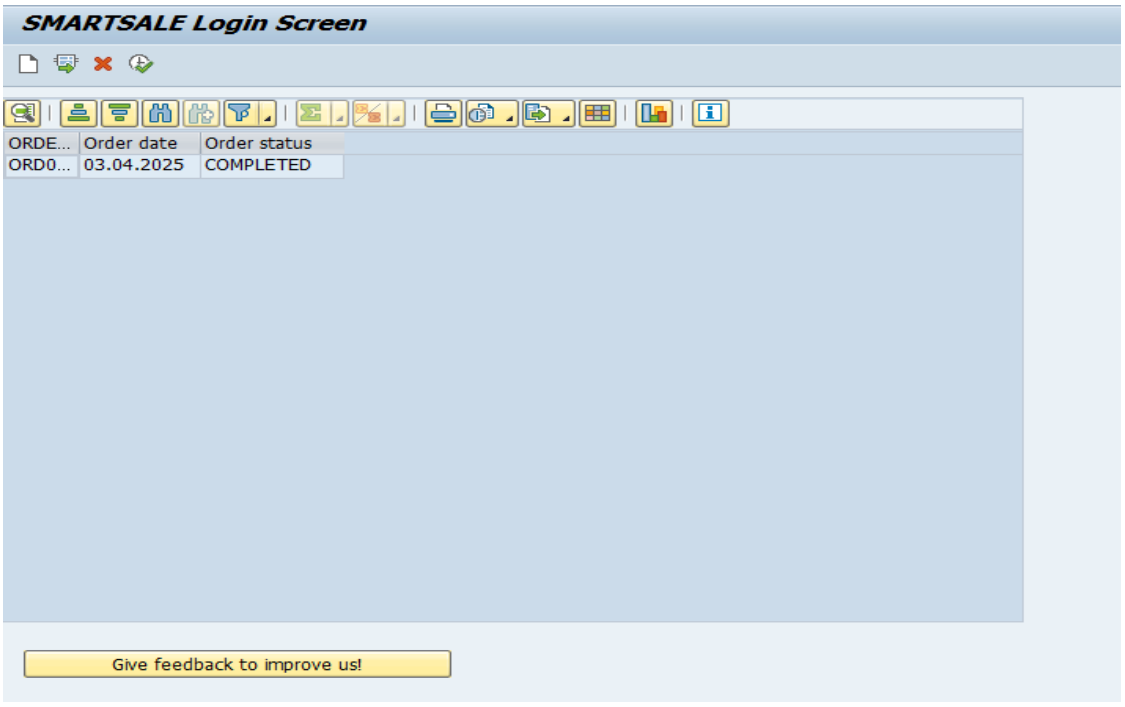
Customer Id	C001
Cust Name	ASANIDH
Password	
Re-Enter Password	
Security Question	What is your pet name?
Answers	JOE

Figure 5.3 Update Profile Screen.



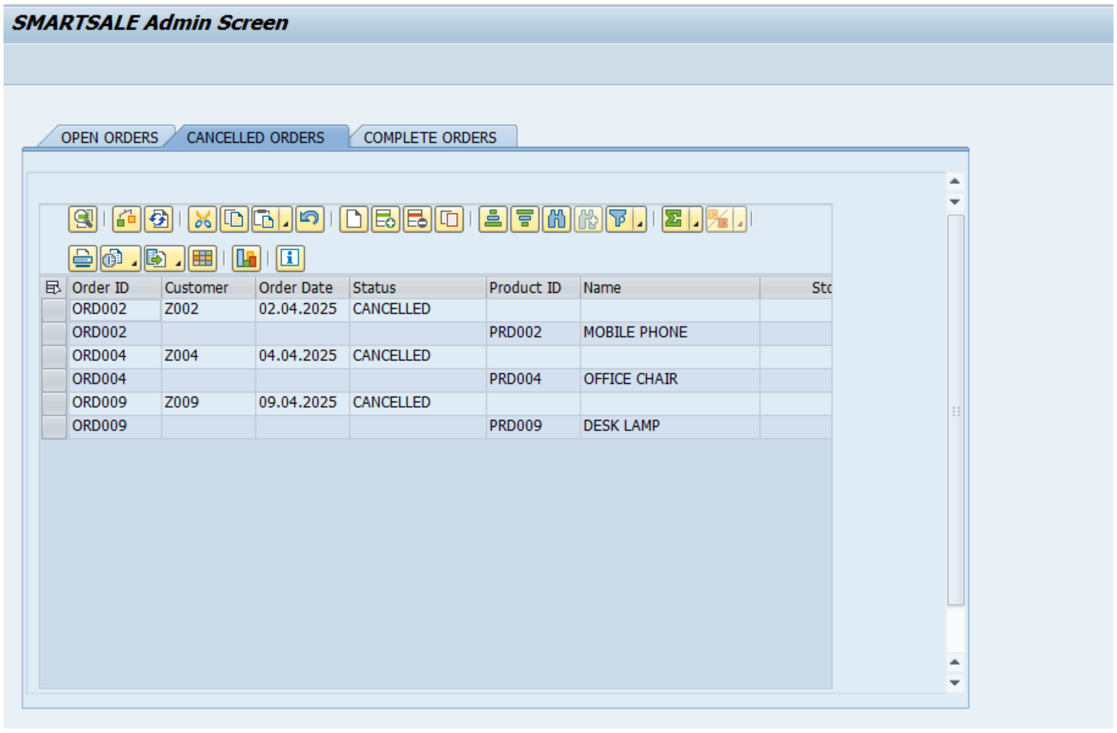
Order ID	Customer	Order Date	Status	Product ID	Name	Stock
ORD001	Z001	01.04.2025	OPEN			
ORD001				PRD001	LAPTOP PURCHASE	120
ORD005	Z005	05.04.2025	OPEN			
ORD005				PRD005	KEYBOARD & MOUSE	80
ORD008	Z008	08.04.2025	OPEN			
ORD008				PRD008	MONITOR 27 INCH	40

Figure 5.4 Admin Screen with Open orders



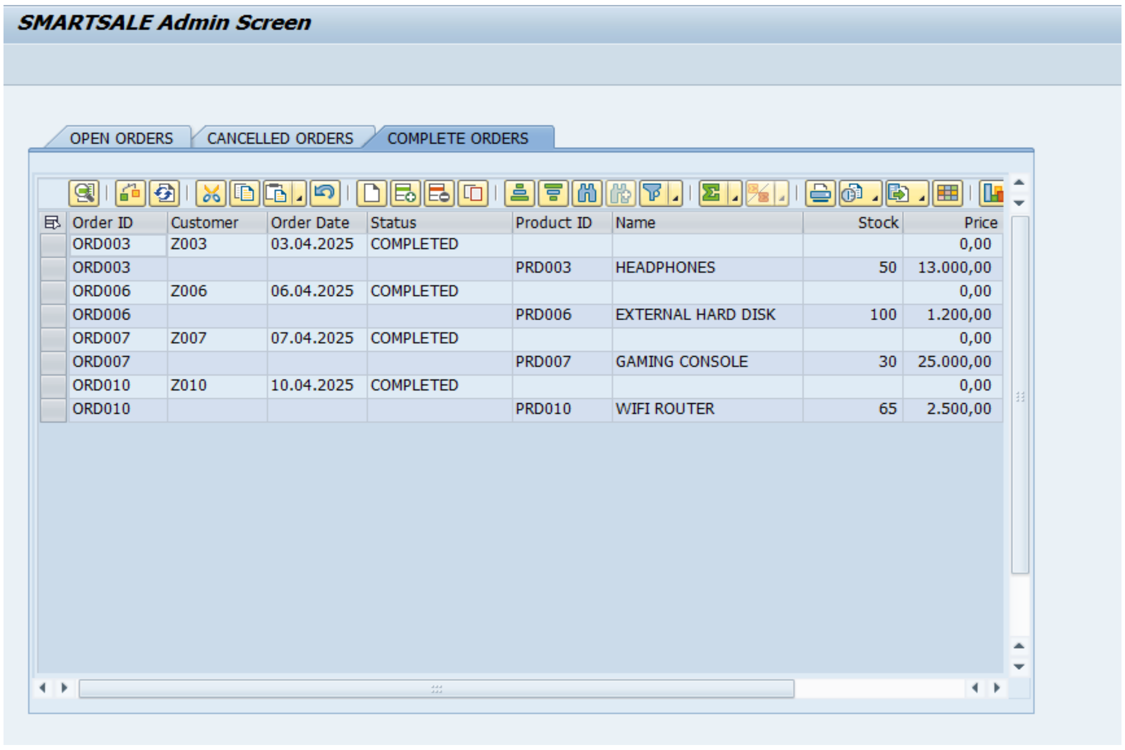
ORDE...	Order date	Order status
ORD0...	03.04.2025	COMPLETED

Figure 5.5: Customer Screen showing orders.



Order ID	Customer	Order Date	Status	Product ID	Name	Sto
ORD002	Z002	02.04.2025	CANCELLED			
ORD002				PRD002	MOBILE PHONE	
ORD004	Z004	04.04.2025	CANCELLED			
ORD004				PRD004	OFFICE CHAIR	
ORD009	Z009	09.04.2025	CANCELLED			
ORD009				PRD009	DESK LAMP	

Figure 5.6: Admin Screen with Cancelled Orders

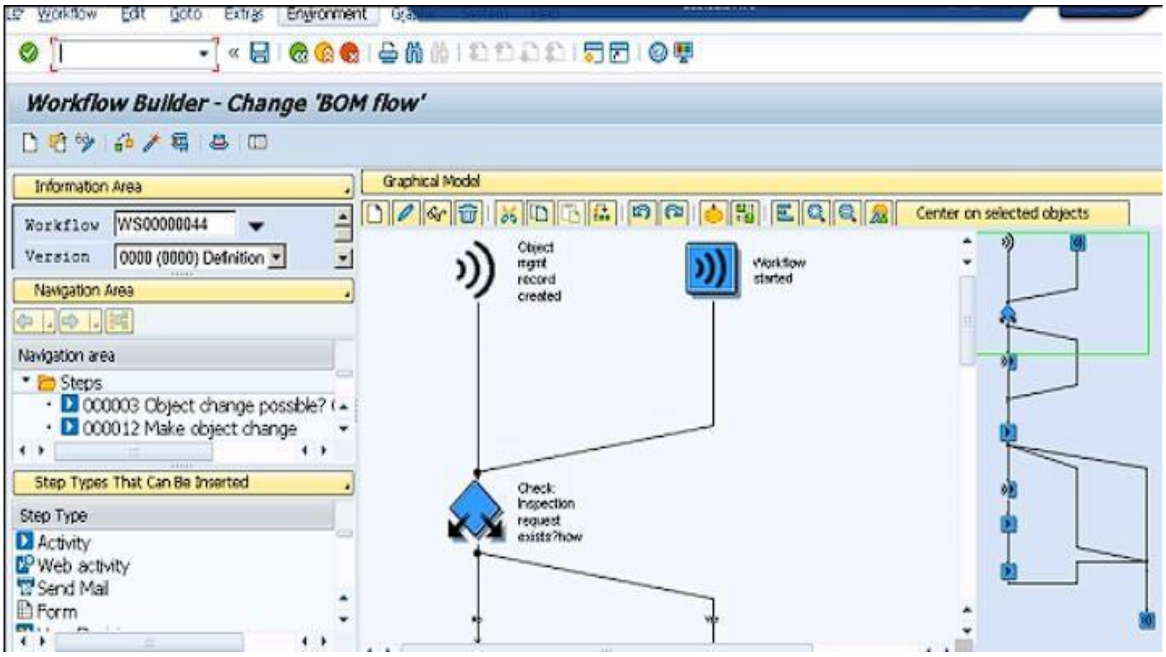


SMARTSALE Admin Screen

OPEN ORDERS | CANCELLED ORDERS | COMPLETE ORDERS

Order ID	Customer	Order Date	Status	Product ID	Name	Stock	Price
ORD003	Z003	03.04.2025	COMPLETED				0,00
ORD003				PRD003	HEADPHONES	50	13.000,00
ORD006	Z006	06.04.2025	COMPLETED				0,00
ORD006				PRD006	EXTERNAL HARD DISK	100	1.200,00
ORD007	Z007	07.04.2025	COMPLETED				0,00
ORD007				PRD007	GAMING CONSOLE	30	25.000,00
ORD010	Z010	10.04.2025	COMPLETED				0,00
ORD010				PRD010	WIFI ROUTER	65	2.500,00

Figure 5.7: Admin Screen with Completed Orders



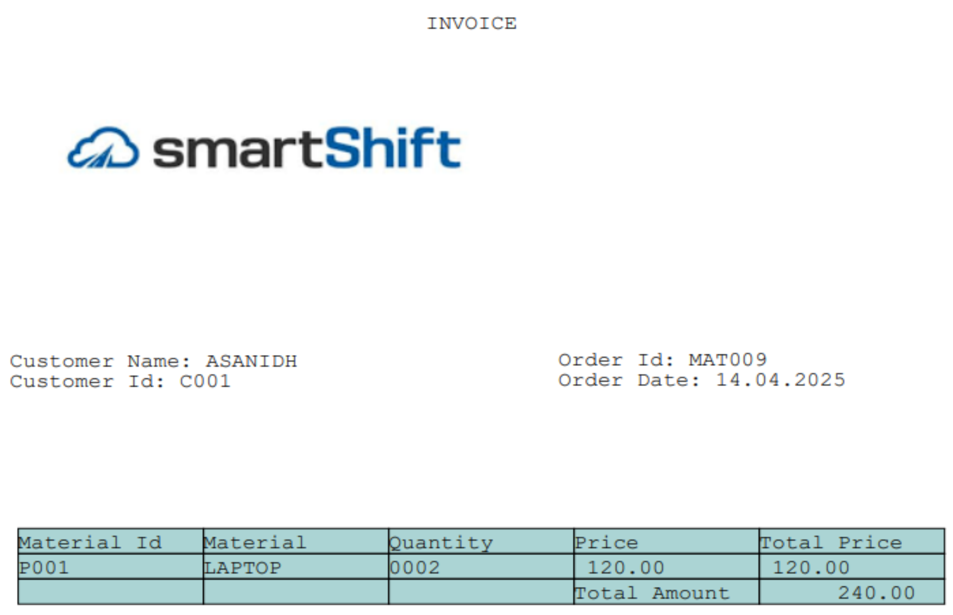
Workflow Builder - Change 'BOM flow'

Workflow WS00000044

Version 0000 (0000) Definition

Steps
- 000003 Object change possible?
- 000012 Make object change

Figure 5.8: Workflows



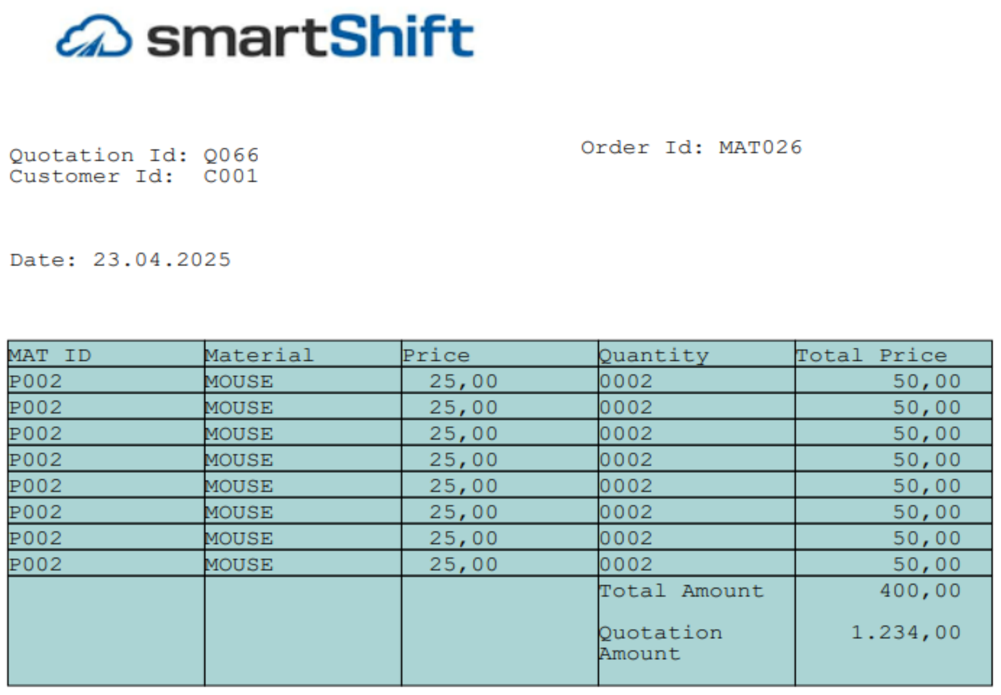
INVOICE

Customer Name: ASANIDH
Customer Id: C001

Order Id: MAT009
Order Date: 14.04.2025

Material Id	Material	Quantity	Price	Total Price
P001	LAPTOP	0002	120.00	120.00
			Total Amount	240.00

Figure 5.9: Smart forms Invoice Detail



Quotation Id: Q066
Customer Id: C001

Order Id: MAT026

Date: 23.04.2025

MAT ID	Material	Price	Quantity	Total Price
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
P002	MOUSE	25,00	0002	50,00
			Total Amount	400,00
			Quotation Amount	1.234,00

Figure 5.10: Smart forms Quotation Details

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The primary goal of this project was to develop a customized SAP Sales and Distribution (SD) application for SmartSale Ltd. using ABAP, aimed at streamlining the sales workflow, enhancing user interaction, and improving overall operational efficiency. The need arose from the existing manual or partially automated systems which lacked user-specific interfaces, real-time communication, and integrated validation features — ultimately resulting in delays, inaccuracies, and limited control over order processing.

Through careful design and implementation, the project succeeded in building a robust, role-based application that distinguishes between customer and admin access. The solution incorporated custom login and registration modules, dynamic screen handling using module pool programming, and SAP-standard elements like ALV Grids and Smart Forms to ensure usability and consistency. By automating actions such as order inquiries, quotations, stock validation, and follow-ups, the application effectively reduced manual effort and potential errors in the SD cycle.

Workflows including automated email triggers made communication feasible between the administrative team and customers, enabling speedy communication and status updates. The system utilized validation logic to enforce data integrity by ensuring transactions could only occur based on all appropriate circumstances from login security to stock availability.

The deployment was a success, but the project experienced some challenges. For example, the email module currently uses static configuration rather than dynamic role-based distribution, and the application does not have real-time stock integration with external systems. Additionally, this version restricted the user experience to SAP GUI environments, ignoring cross-platform compatibility and mobile responsive design.

However, this project has made a substantial contribution. It illustrates how custom ABAP programming can be used to expand traditional SAP capabilities to satisfy particular business needs. It also emphasizes the efficient use of workflows and Smart Forms for documentation and communication in an efficient SD process. Most importantly, it offers a scalable framework for future development, incorporating contemporary features like dashboards, analytics, and external APIs.

In conclusion, the project not only addressed the core challenges faced by SmartSale Ltd. in their order and inquiry processing, but also set the stage for continuous improvement in their SAP-based operations. The outcomes prove the practical value of tailored SAP solutions in enhancing process automation, user satisfaction, and decision-making accuracy.

6.2 LIMITATIONS

While the developed SAP Sales and Distribution system has achieved its primary objectives and functions effectively across core modules, certain constraints were identified during implementation and testing that define the current boundaries of the application.

- **Dependence on SAPconnect for Email Notifications:** The current email communication feature is built on SAPconnect, which requires precise SMTP configuration to function correctly. In environments where the mail server settings are either misconfigured or inaccessible, workflow notifications may fail to deliver, leading to delays in important communications such as quotations or inquiry confirmations.

- **Limited UI/UX Design:** While the system is functional, the user interface is form-based and designed using classic module pool techniques. It lacks the modern look and responsiveness that can be achieved using SAP Fiori or Web Dynpro, which may affect user engagement and efficiency in the long term.

- **Admin Login Credentials Hardcoded:** Currently the admin login is hardcoded for demonstration purposes. While suitable for controlled scenarios, this isn't suitable for a real application since it lacks flexibility and security in authentication. A possible route to a more secure implementation could involve dynamic user authentication with roles and incorporating SAP role-based authorization management.

- **Input Validation Coverage is Minimal:** The UI has some standard validation for inputs (i.e., login, registration, sequential inquiry creation, etc.), though as there is limited handling of all edge cases around validation for inputs, (i.e. input with special characters, varying lengths of inputs). Some additional validation and sanitization at the field level will add more comprehensive integrity to the data as well as overall user experience.

- **Performance Considerations for Bigger Datasets:** To prove functional correctness, this version of application was built and tested with a very small amount of test data. We have not applied any performance tuning to responsiveness, memory, or screen load times around managing larger datasets. Tuning for handling bigger datasets with scale will need to be applied for enterprise rollout.

6.3 FUTURE SCOPE

The system can be further improved in a number of ways to increase its scalability, efficiency, and user experience while staying true to the initial goals of order and inquiry management:

- **Role-based access control (RBAC):** Improving security and performance by moving to a more advanced role-based access control, allowing for different permissions for administrators, support staff and clients.

- **Email Configuration Module:** This feature in the admin panel aims to provide flexibility in communication workflows through real-time email configuration (SMTP settings, recipients changes) without changing the code.

- **Dashboard Review:** To improve visibility into business operations, we are introducing a graphical dashboard for the admin interface, to display key statistics such as daily inquiries, total orders, and user activity over time.

- **Notification System:** Implementing real-time system notifications (email/SMS/pop-up alerts) to inform users about inquiry responses, order confirmations, or system updates.

- **Automated Stock Alerts:** Extending the stock checking functionality to include low-stock alerts, enabling proactive replenishment and avoiding last-minute shortages.

- **Secure Password Reset Mechanism**: To improve account recovery and user support, a user-friendly and secure password reset option utilizing email verification or OTP has been introduced.

REFERENCES

[1] SAP SE, "ALV Grid Control," SAP Help Portal. [Online]. Available: https://help.sap.com/doc/saphelp_nw73ehp1/7.31.19/enus/4e/b7a512999e0134e10000000a42189b/frameset.htm

[2] SAP SE, "Smart Forms Basics," SAP Help Portal. [Online]. Available: https://help.sap.com/docs/SUPPORT_CONTENT/abapoutput/3353523866.html

[3] M. M. Raj, ABAP Development for SAP S/4HANA, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2021.

[4] B. Goerlich, SAP Smart Forms: Creating and Managing Forms in SAP ERP, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2017.

[5] P. Buley, Workflow for SAP S/4HANA: Business Process Automation with SAP Workflow, Boston, MA, USA: SAP Press, 2020.

[6] C. C. Smith, Customizing SAP S/4HANA: Tools and Techniques for Tailoring the System to Meet Business Requirements, Boston, MA, USA: SAP Press, 2020.

[7] SAP SE, "Properties of the ALV Grid Control," SAP Help Portal. [Online]. Available: https://help.sap.com/doc/saphelp_snc70/7.0/enUS/4e/ba881e260e56a5e10000000a421937/content.htm

[8] M. T. Smith, "Improving Legacy SAP Applications with ALV Grid Control," International Journal of SAP Technologies, vol. 9, no. 3, pp. 45–51, 2021.

[9] J. T. Anderson, ABAP Objects: ABAP Programming in SAP NetWeaver, Boston, MA, USA: SAP Press, 2016.

[10] D. Obermeier, Efficient SAP Business Workflow: Practical Solutions for Workflow Automation, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2020.

[11] SAP SE, "Working with the ALV Grid Control," SAP Help Portal. [Online]. Available: https://help.sap.com/doc/saphelp_snc70/7.0/enUS/4e/bd16291041389ee10000000a421937/content.htm

[12] H. Frank and T. Allgaier, SAP S/4HANA Sales Certification Guide (C_TS462_2021):

Application and Configuration Preparation, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2021.

[13] C. Weber and M. Wegelin, SAP System Security Guide, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2019.

[14] SAP SE, "Module Pool Programming," SAP Help Portal. [Online]. Available: https://help.sap.com/docs/SUPPORT_CONTENT/wdabap/3362186650.html

[15] M. Bader, SAP ABAP Advanced Cookbook, Birmingham, UK: Packt Publishing, 2015.

[16] SAP SE, "Multiple Tabs in a Selection Screen in Module Pool Programming," SAP Help Portal.[Online].Available:
https://help.sap.com/docs/SUPPORT_CONTENT/abap/3353523998.html

[17] R. D. Winter, ABAP Development for SAP Business Workflow, Boston, MA, USA: SAP Press, 2011.

[18] K. Krishnamoorthy, SAP Integration Suite: Managing APIs, Workflows, and Event-Based Processes, Boston, MA, USA: SAP Press, 2022.

[19] M. Ahmed, ABAP Development for SAP HANA, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2020.

[20] N. Ghosh and S. Goon, Workflow for SAP S/4HANA, Boston, MA, USA: SAP Press, Rheinwerk Publishing, 2022.

[21] D. Mather, ABAP Development for SAP S/4HANA: ABAP Programming Model for SAP Fiori, Boston, MA, USA: SAP Press, 2020.

[22] M. Missbach, M. Fenz, and B. Mühling, Securing SAP S/4HANA: Security Guide, Boston, MA, USA: SAP Press, 2021.

[23] B. U. Weinshel, SAP S/4HANA Sales: Business User Guide, Boston, MA, USA: SAP Press, 2022.

[24] M. Keller, SAP Workflow Management: A Practical Guide, Rheinwerk Publishing, 2021.

[25] J. Heinrich, SAP Smart Forms: Creating Forms Without Scripts, Boston, MA, USA: SAP Press, 2020.