LONDON METROPOLITAN UNIVERSITY

ITAHARI INTERNATIONAL COLLEGE

ing

Module Code & Module Title
CS5002NT– Software Engineering

Assessment Type
20% Group Coursework

Semester
2024-2025 Autumn

Group Name: D
Group members

London Met ID	Student Name
23049395	Safal Bhattarai
23049984	Sanjita Chaudhary
23049144	Aashish Khadka
23049236	Diwakar Yadav
23049237	Diya Rai

Assignment Due Date: 24 December 2024
Assignment Submission Date: 24 December 2024
Word Count: 3936

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Acknowledgements

We would like to take this opportunity as to express our heartfelt gratitude to our module leader, Mr. Amit Shrestha, for his exceptional assistance and support during these group project work and the successful completion of this course work. He has been able to dedicate his time in providing constructive criticisms, patiently tutoring us, for encouraging critical thinking in handling this contingency of this course work. From his passion in helping us learn to his friendly nature he has made the whole process not only workable but also kind of enjoyable.

we also like to extend our gratitude to our group members, whose teamwork and collaboration played an important role in this project. This enthusiasm of each of us to contribute, to share our unique ideas, to support and respect one another during challenging times its truly embodies the spirit of teamwork. This short and sweet Journey helped us not only to grow our self in terms of technical and analytical expertise but also highlighted the importance of open communication, mutual cooperation, and perseverance towards achieving a common goal.

It is greatly amplified that we got a chance to do this journey with Mr. Shrestha and for our colleagues; this was quite a meaningful and memorable experience. It is also important to underline that all the insights and all the experience which have been collected during the period of this work are invaluable, with the team is expected to use all the knowledge received during the process in the future work for the new task and new opportunities.

Table of Contents

1 Introduction

This course is part of the Software Engineering module, dealing with practical applications of the principles of Structured Software Engineering, special consideration is given to the Yourdon methodology. The project aims at showing the ability to apply structured software engineering methods competently, produce essential project management artifacts, and work collaboratively in a small team to develop a software system within a predetermined time frame.

The project focuses on developing a competitive Inventory Management System (IMS) for Global Tech Corporation. The aim of the system is to improve warehouse operations through better stock monitoring, automation of report generation, and order processing. One of the major challenges is to define the scope of the project, allocate resources in an efficient manner, and implement effective communication strategies in order to avoid the problems that occurred in the previous IMS project such as unclear scope, inadequate communication, and mismanagement of resources.

The core activities of this course involve developing key project management documents: business cases, progress logs, and detailed system specifications. It also includes designing Entity-Relationship Diagrams (ERD), Data Flow Diagrams (DFD), and Process Specifications (Pspecs) to realize a solid, scalable, and effective Information Management System (IMS) that meets both business requirements and engineering standards.

The final deliverables will include the detailed analysis and design documentation that charts a clear road map on how to put in place the IMS, hence affording hands-on experience of applying structured software engineering methodologies and project management techniques within real-world scenarios.

2 Business Case: Global Tech Corporation

2.1 Introduction

Global Tech Corporation is launching a project to develop a next-generation Inventory Management System (IMS) to optimize warehouse operations in Nepal. This initiative addresses the challenges faced in a previous IMS project, where unclear scope, poor resource allocation, and lack of structured management resulted in inefficiencies and customer dissatisfaction. The new IMS aims to streamline inventory tracking, order management, and reporting, with a focus on improving operational efficiency and decision-making. The project will follow a structured approach, incorporating best practices in project management, risk mitigation, and user training to ensure successful deployment. By implementing this system, Global Tech Corporation seeks to reduce operational costs, boost productivity, and enhance customer satisfaction, positioning itself for long-term success and growth.

2.2 Problem Statement

- Lack of clear project scope and documentation of features, . integration points, or end goals.
- Poor resource allocation, leading to delays and budget overruns.
- Unstructured communication and absence of regular updates.
- Inadequate risk management, resulting in system downtime and inefficiencies.
- No post-implementation review to assess outcomes or identify areas for improvement.

2.3 Objectives

- Develop and implement a next-generation, automated IMS for warehouse operations in Nepal.
- Streamline inventory tracking, purchase orders, sales management, and reporting processes.
- Address shortcomings of the previous project by adopting structured analysis, design, and project management practices.
- Ensure effective resource allocation, robust communication, and regular progress reviews.

- Enhance operational efficiency, reduce financial losses, and improve customer satisfaction.
- Mitigate risks such as system downtime and budget overruns through proactive planning.

2.4 Scope

Component	In Scope	Out of Scope
User Access Management	1. User registration and login for Admin 2. Customers with role-based access	Advanced user analytics and behaviour tracking
Purchase Order Management	3. Adding and viewing purchase orders, including supplier details	Automated supplier selection or bidding system
Report Generation	4. Generating purchase and sales reports for decision-making	Advanced predictive analytics or AI-based reporting
Sales Management	5. Managing sales orders, delivery methods, addresses, and dispatch details	Integration with external delivery/logistics systems
Payment	6. Secure payment for customers. 7. Record transaction and price comparison.	International payment support.

Table 1: Scope

2.5 Benefits

Quantitative Benefits are as follow

- Cost Savings: 25% reduction in operational costs through streamlined inventory and order management.
- Time Efficiency: 35% decrease in time spent on manual inventory tracking and reporting.
- Increased Revenue: 20% boost in sales due to better stock availability and improved customer satisfaction
  .
- Productivity Gains: 30% improvement in employee productivity by automating repetitive tasks.
- Improved Cash Flow: 25% increase in inventory turnover, reducing holding costs and improving liquidity.
- Error Reduction: 50% fewer errors in purchase orders, sales tracking, and inventory records.

Qualitative Benefits are as follow:

- Improved Decision-Making: Actionable insights through detailed reports will enable admins and managers to make informed, data-driven decisions.
- Enhanced Customer Experience: Streamlined operations, accurate inventory tracking, and secure payment processing will foster trust and satisfaction among customers.
- Increased Employee Productivity: Automation of repetitive tasks allows employees to focus on higher-value and strategic responsibilities.
- Greater System Reliability: Risk mitigation strategies and proper resource management ensure minimal downtime and smoother operations.
- Future Scalability: The system is designed to adapt to evolving business needs, providing a sustainable and scalable solution for long-term growth.

2.6 Costs

The project is estimated to cost Rs 160,000 in the first year, including Rs 100,000 for development, Rs 30,000 for implementation, and Rs 30,000 for maintenance and risk management. It is presented in tabular form below:

Category	Estimated Cost
Software Design and Development	Rs 65,000
Hardware Procurement	Rs 20,000
Testing & Quality Assurance	Rs 15,000
Deployment & Training	Rs 30,000
Annual Maintenance	Rs 20,000
Risk Mitigation	Rs 10,000
Total	Rs 160,000 /-

Table 2: Project Cost

2.7 Risks and Mitigation

Risks include system downtime, integration challenges, budget overruns, and staff resistance to change. These will be mitigated through robust planning, regular progress monitoring, and comprehensive training.

- **System Downtime**:
  Impact: Disrupts warehouse operations, causing delays and financial losses.
  Mitigation: Implement backup systems, failover mechanisms, and real-time monitoring.

- **Integration Challenges**:
  Impact: Compatibility issues with existing systems could delay implementation.
  Mitigation: Conduct comprehensive system analysis and thorough integration testing.

- **Data Security Breaches**:
  Impact: Exposure of sensitive data could harm reputation and operations.
  Mitigation: Use encryption, firewalls, secure authentication, and periodic audits.

- **Budget Overruns**:
  Impact: Increased costs due to scope creep or unforeseen issues.
  Mitigation: Define a clear scope, regularly review the budget, and use agile management.

- **Project Delays**:
  Impact: Missed deadlines affecting warehouse operations and project success.
  Mitigation: Set realistic timelines, enforce progress reviews, and escalate issues quickly.

- **Staff Resistance**:
  Impact: Resistance to adopting the new system may slow its effectiveness.
  Mitigation: Provide comprehensive training, support, and demonstrate system benefits.

- **Errors in Reporting**:
  Impact: Inaccurate data in purchase or sales reports could lead to poor decisions.
  Mitigation: Conduct regular testing and validation of the reporting system.

- **Customer Dissatisfaction**:
  Impact: Failure to meet expectations may harm trust and reputation.
  Mitigation: Focus on user feedback, ensure key features meet customer needs, and test thoroughly.

2.8 Timeline

The project will follow a 12-month schedule:

Phase	Duration (in months)	Activities
Phase 1: Planning and Design	1.5	○ Define scope and objectives ○ Assemble team ○ Allocate resources
Phase 2: Requirements Analysis	1.5	○ Gather system requirements ○ Document requirements ○ Identify risks
Phase 3: System Design	1.5	○ Create architecture ○ Design database and UI ○ Finalize workflows

Phase 4: Development	5	○ Build core modules ○ Integrate with systems ○ Conduct iterative testing
Phase 5: Testing	1.5	○ Perform end-to-end testing ○ Conduct performance testing ○ Address critical issues
Phase 6: Deployment & Training	1	○ Launch the system ○ Train staff ○ Provide user support

Table 3: Timeline

2.9 Conclusion

The successful implementation of the new Inventory Management System (IMS) by Global Tech Corporation is critical for overcoming the challenges faced in the previous project. By focusing on clear project scope, structured planning, and effective resource allocation, the IMS will streamline warehouse operations, improve inventory tracking, and enhance customer satisfaction. With a dedicated approach to risk management, testing, and training, the project aims to address past inefficiencies, reduce operational costs, and foster long-term business growth. The implementation of this system will not only optimize processes but also position Global Tech Corporation as a leader in automated warehouse management., reduce costs, and enhance customer satisfaction.

3 SRS

3.1 Description of SRS

A Software Requirements Specification (SRS) outlines all the essential requirements for a system to succeed. These requirements provide a detailed explanation of the system's

features during its development phase. They encompass not only the functional requirements (FRs) but also the non-functional requirements (NFRs). While functional requirements explicitly describe the system's intended features and operations, non-functional requirements are equally vital for the system's overall success. (Muhammad asif Asif, 2024)

3.2 Intended Audience

The SRS is intended for the following stakeholders:

- Project Managers: Make sure the business goals and resources are used well.
- Developers: To know the system function how does
- it works and other technical limits.
- Testers: To check the function is working correctly or not.
- End Users (Admins and Buyers): To make sure the system works for their daily tasks.
- System Designers: To plan and build the system’s structure.
- UI/UX Designers: The design should be easy to use and look attractive for users.
- Support Team: To communicate by each other and helping to solve the problem after the system was launched.
- Database Administrators: To make sure the data is stored safely.
- Stakeholders: To make sure the project track and its successful.
- Business Analysts: Make the system what users’ needs and requirements.

3.3 Definitions and Acronyms

Terms	Definition

IMS	Inventory Management System
SRS	Software Requirements Specification
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
UI	User Interface
Admin	System administrator with access to all functionalities
FR	Functional Requirements
NFR	Non-Functional Requirements
CPU	Central Processing Unit
SQL	Structured Query Language
RAM	Random Access Memory
GHz	Gigahertz
GB	Gigabyte
SSD	Solid-State Drive
Mbps	Megabits per second

Table 1: Definitions and Acronyms

4 Functional Requirements

“Functional requirements specify what a system should do. They define the actions, tasks, and operations that a system must perform to satisfy user needs and achieve its goals.

These requirements are concerned with the functionality of the system and describe the input, behaviour, and output of the system." (Modern Analyst Media LLC, 2024)

4.1 User Management

ID	Functions	Description	Priority
FR1.1	User Registration	Allows new users (Admin and Customers) to register with required credentials (e.g., name, email, role).	High
FR1.2	User Login	Enables registered users to log in securely with a valid username and password.	High
FR1.3	View User Profile	Provides users access to view their profile information, such as name, email, and role.	Medium
FR1.4	Edit User Profile	Allows users to update their profile details, such as email, contact number, or password.	Medium
FR1.5	User Role Management	Enables admins to manage user roles, including creating, updating, or removing roles as required.	High

Table 2: User management

4.2 Purchase Order

ID	Functions	Description	Priority

FR1.1	Create Purchase	Users can create new purchase order.	High
FR1.2	Edit Purchase	Users can update a purchase order	High
FR1.3	View Purchase	Displays the information of purchase order	Medium
FR1.4	Cancel Purchase	Allowing users to cancel purchase order	High
FR1.5	Generate Purchase Report	Users can have their reports for purchase orders.	Low
FR1.6	Approve Purchase	Allowing authorized people to approve the purchase order.	High

Table 3: Purchase Order

4.3 Report Preparation

ID	Functions	Description	Priority
FR2.1	Generate Purchase Report	The system must generate purchase reports detailing new inventory acquisitions, including product quantities, costs, and supplier information.	High
FR2.2	Generate Sales Report	Generate sales reports for the admin to track total sales, quantities sold, and profit/loss over specific timeframes.	High
FR2.3	Filter Report Data	Allow users to filter report data by specific criteria such as date range, product category, supplier, or customer for more targeted reports.	High
FR2.4	Schedule Automated Reports	Allow scheduling of automated reports to be generated and delivered (e.g., daily, weekly, monthly) to specified users via email or system notifications.	Low
FR2.5	Visual Representation of Reports	Include visual representations like charts or graphs (e.g., bar charts, pie charts) to make report data easier to understand and analyse.	Medium
FR2.6	Approve Purchase	Allowing authorized people to approve the purchase order.	High

Table 4: Report Preparations

4.4 Real Time Stock Updates

ID	Functions	Description	Priority
FR3.1	Real-Time Stock Updates	The system must instantly update inventory levels when products are added, removed, or sold, ensuring accurate and up-to-date stock information.	High
FR3.2	Automatic Synchronisation	Inventory changes must be automatically synchronised across all relevant modules and databases.	High
FR3.3	Threshold Notifications	The system should notify users when stock levels reach pre-defined thresholds for reordering.	Medium
FR3.4	Error Logging	Any inconsistencies or errors in stock updates must be logged for auditing and troubleshooting.	Medium

Table 5: Real Time Stock Updates

4.5 Dispatch Order

ID	Functions	Description	Priority
FR 4.1	Create Dispatch Order	The system shall allow Users to create a dispatch order by selecting items from inventory and specifying destination.	High
FR 4.2	Update Dispatch Order	The System Shall enable Users to modify existing dispatch orders, such as adjusting quantities or Changing destination.	Medium
FR 4.3	Track Dispatch Order	The System Shall allow Users to track the status of dispatched orders, including in-transit and delivered statuses.	High
FR 4.4	Cancel Dispatch Order	The system shall provide functionality to cancel a dispatch order before it is marked as completed.	Low
FR 4.5	Generate Dispatch Report	The System Shall generate report summarizing dispatched orders, including date, time, item, and delivery status.	Medium

Table 6: Dispatch Order

4.6 Payment

ID	Functions	Description	Priority
FR5.1	Payment Setup	The system should define product amount	High
FR5.2	Payment processing	Warehouse should secure online payment and offline options and processes the transaction properly.	High
FR5.3	Invoice management	Warehouse must auto-generate invoice after payment and for additional charges.	High
FR5.4	Refund and adjustment	Warehouse should support partial/full payments.	Medium
FR5.5	Payment tracking	The system should maintain detailed payment records and display payment history.	High
FR5.6	Notifications	The system should notify users about the deadline of payment, confirmations and overdue balance.	Low

Table 7: Payment

5 Non-Functional Requirements

“Non-functional requirements define the quality attributes, constraints, and system behaviours that are not related to specific functionalities but are crucial to the overall performance, usability, and reliability of a system. They describe how the system should operate rather than what it should do.” (Sommerville, 2011)

5.1 External Interface Required

External Interfaces Required refer to the interactions between the IMS and external systems or devices that enable the system to perform specific functions. These interfaces may involve communication with hardware devices, other software systems, or external services. The requirements to use our system is listed below:

5.1.1 Hardware Specifications

Hardware Component	Specification
Server	A dedicated server to host the IMS system. Specifications: 16 GB RAM, 8-core CPU, 500 GB SSD storage.
Client Devices	Desktops, laptops, or tablets with a minimum of 4 GB RAM and 2 GHz processor for accessing the system.
Networking	High-speed internet connection with a minimum bandwidth of 100 Mbps for real-time updates and seamless communication.
Printer	Label printers for generating product and inventory labels. Should support both thermal and inkjet printing.
Power Supply	Uninterrupted Power Supply (UPS) with 10 hours of battery backup for critical systems in case of power failure.

Table 8: Hardware Specifications

5.1.2 Software Specifications

Software Component	Specification
Operating System	The system will be hosted on a server with Windows Server 2019 or a Linux-based system (Ubuntu or CentOS) for stability and security. Client devices will require Windows 10 or macOS for accessing the system.
Database Management System (DBMS)	A relational DBMS like MySQL to store and manage data related to inventory, sales, users, and orders.
Web Server	Apache HTTP Server or Nginx for hosting the IMS web application, ensuring fast response times, scalability, and security for web-based interactions.
Backend Framework	React.js, a JavaScript framework for building a dynamic, interactive, and responsive user interface for the IMS. Vite will be used for fast bundling and module loading during development.
Frontend Framework	Uninterrupted Power Supply (UPS) with 10 hours of battery backup for critical systems in case of power failure.
Version Control	Git will be used for version control and collaborative development. GitHub will serve as the platform for code sharing, collaboration, and version tracking.
Cloud Hosting Platform	Cloud hosting services like Amazon Web Services (AWS) or Microsoft Azure will be used to host the IMS, providing scalable resources, high availability, and disaster recovery capabilities.

Table 9: Software Specifications

6 Other Non-Functional Requirements

ID	Category	Description
NFR5.1	Performance	The system should process stock updates within 1 second under normal load.
NRF5.2	Scalability	Must handle up to 1,000 simultaneous transactions without performance
NRF5.3	Reliability	Ensure 99.9% uptime for the stock update feature.
NFR5.4	Security	All stock data must be encrypted during storage and transmission.
NFR5.5	Maintainability	The system must allow easy updates and debugging with modular code design.
NFR5.6	Usability	The stock update interface must be user-friendly, offering intuitive workflows for all user roles.
NFR5.7	Auditability	The system must maintain detailed logs of all stock update activities for compliance and tracking purposes.
NFR5.8	Compliance	Ensure adherence to local and international inventory management standards (e.g., ISO 9001).
NFR5.9	Availability	The stock update feature must be accessible 24/7, with downtime limited to scheduled maintenance.

Table 10: Other non-functional requirements

Group Task

7 Overall Context Diagram

The Level 0 DFD (Context Diagram) represents the highest-level view of the Inventory Management System, showing how it interacts with four main external entities:

1. Supplier:

- Receives purchase orders from the system
- Sends order status back to the system
- Represents suppliers who provide products to the inventory

2. Customer:

- Sends order requests to the system
- Receives stock availability information
- Gets generated invoices from the system
- Receives order status updates
- Represents end-users who purchase products

3. Admin:

- Requests reports from the system
- Receives stock status information
- Gets generated reports
- Provides stock updates
- Manages and oversees the system operations
- Makes purchases from suppliers.

4. Inventory:

- Provides stock status to the system
- Receives payment verification status
- Represents the physical storage of products

Data Flows:

1. Input Flows:

- Purchase Request (from Customer)
- Stock Updates (from Admin)
- Order Status (from Vendor)
- Stock Status (from Inventory)

2. Output Flows:

- Generate Invoice (to Customer)
- Stock Availability (to Customer)
- Order Status (to Customer)
- Receives Reports (to Admin)
- Stock Status (to Admin)
- Receive Purchase Orders (to Vendor)
- Payment Verification Status (to Inventory)

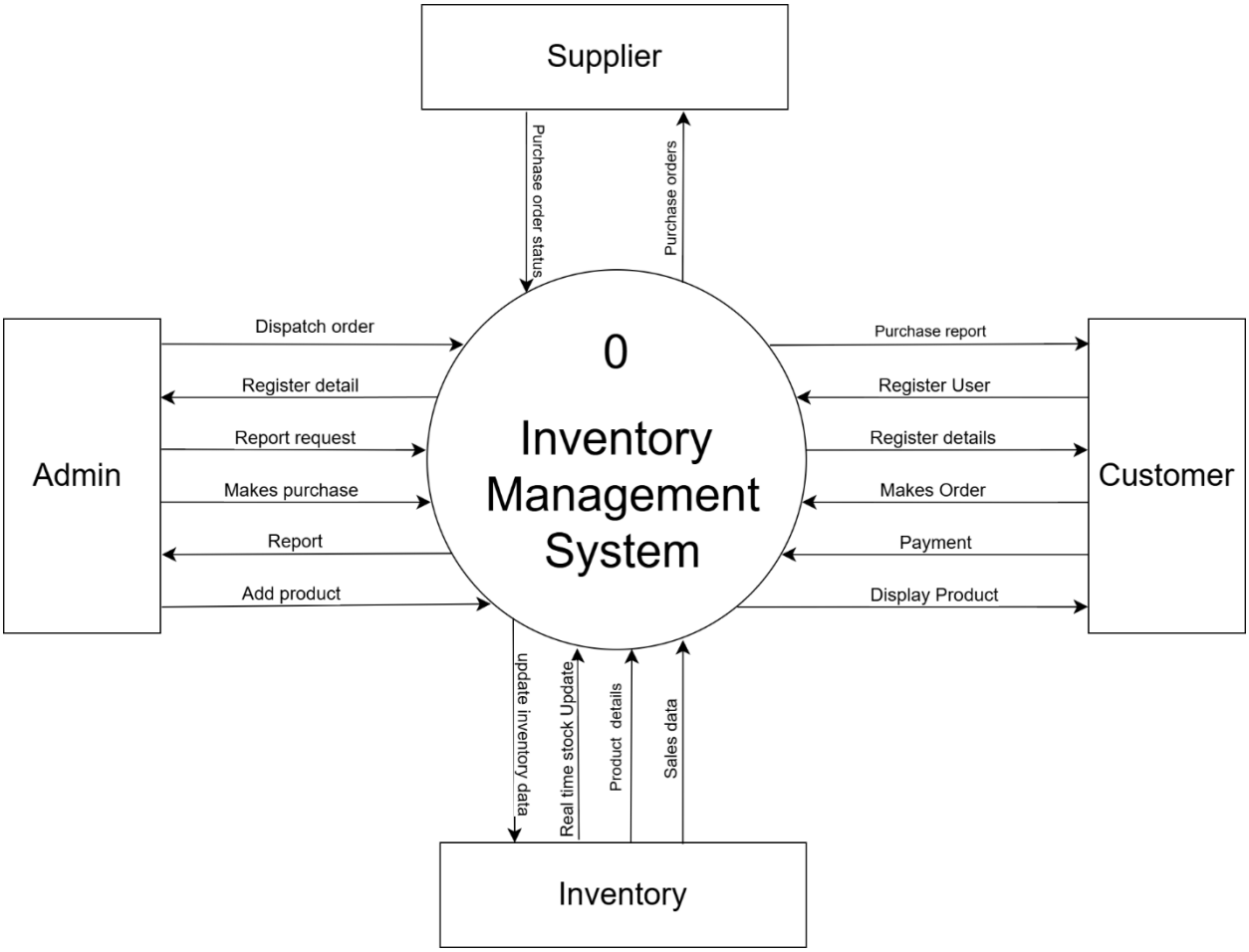
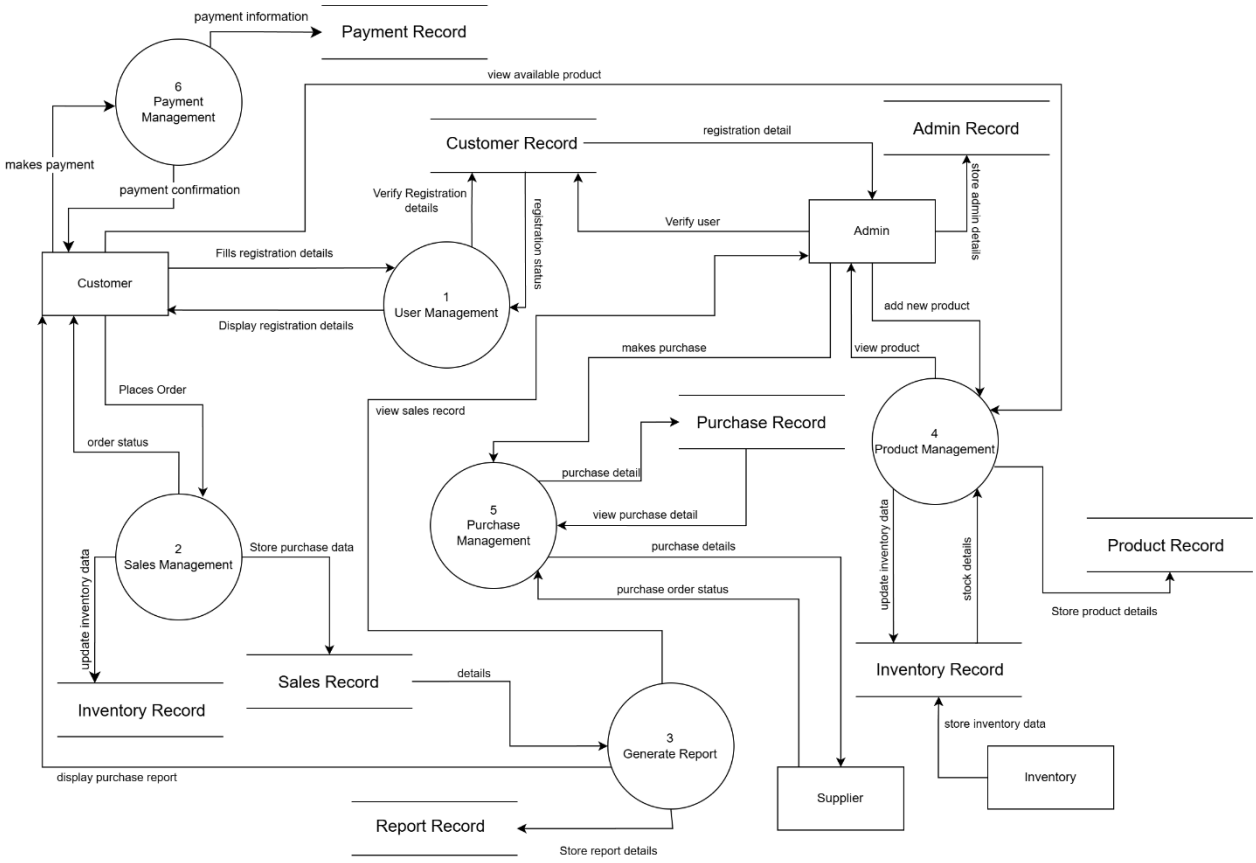


Figure 1: Context Diagram of Overall System

This context diagram effectively shows the system's boundaries and its interactions with external entities, providing a clear overview of the main data flows in and out of the Inventory Management System.



8 Level 1 DFD: Overall

Figure 2: Level 1 DFD of Overall system

This Level 1 DFD represents an inventory management system with several key processes and data flows:

1. User Management (Process 1):

- Handles customer registration
- Verifies registration details
- Communicates with Customer Record
- Interfaces with Admin for user verification

2. Sales Management (Process 2):

- Processes customer orders
- Updates Sales Record
- Provides order status to customers

- Stores order data
- Update inventory data

3. Generate Report (Process 3):

- Creates reports from sales and purchase data
- Stores report details in Report Record
- Provides purchase reports to relevant stakeholders

4. Product Management (Process 4):

- Handles product inventory
- Allows admin to add new products
- Updates Product Record
- Manages stock details with suppliers
- Updates inventory data

5. Purchase Management (Process 5):

- Manages purchase details
- Handles purchase orders
- Communicates with suppliers
- Updates Purchase Record

6. Payment Management (Process 6):

- Processes customer payments
- Records payment information
- Sends payment confirmation to customers
- Updates Payment Record

Data Stores in the system:

- Customer Record: Stores customer registration and details
- Sales Record: Maintains sales transaction data
- Purchase Record: Stores purchase order information
- Product Record: Contains product inventory data
- Payment Record: Tracks payment transactions
- Report Record: Stores generated reports

The diagram shows how data flows between these processes, external entities (Customer, Admin, Supplier), and data stores, creating a complete picture of the system's data movement and processing.

External Entities:

- Inventory: Manages inventory information
- Vendor: Handles order fulfilment and availability
- Customer: Initiates purchases and receives services
- Admin: Oversees system operations and receives reports

The diagram shows how these components interact through various data flows, creating a comprehensive system for managing inventory, sales, and purchases. Each process is numbered (1 through 6) indicating their sequence and relationship in the overall system flow.

9 Level 2 DFD

9.1 User Management

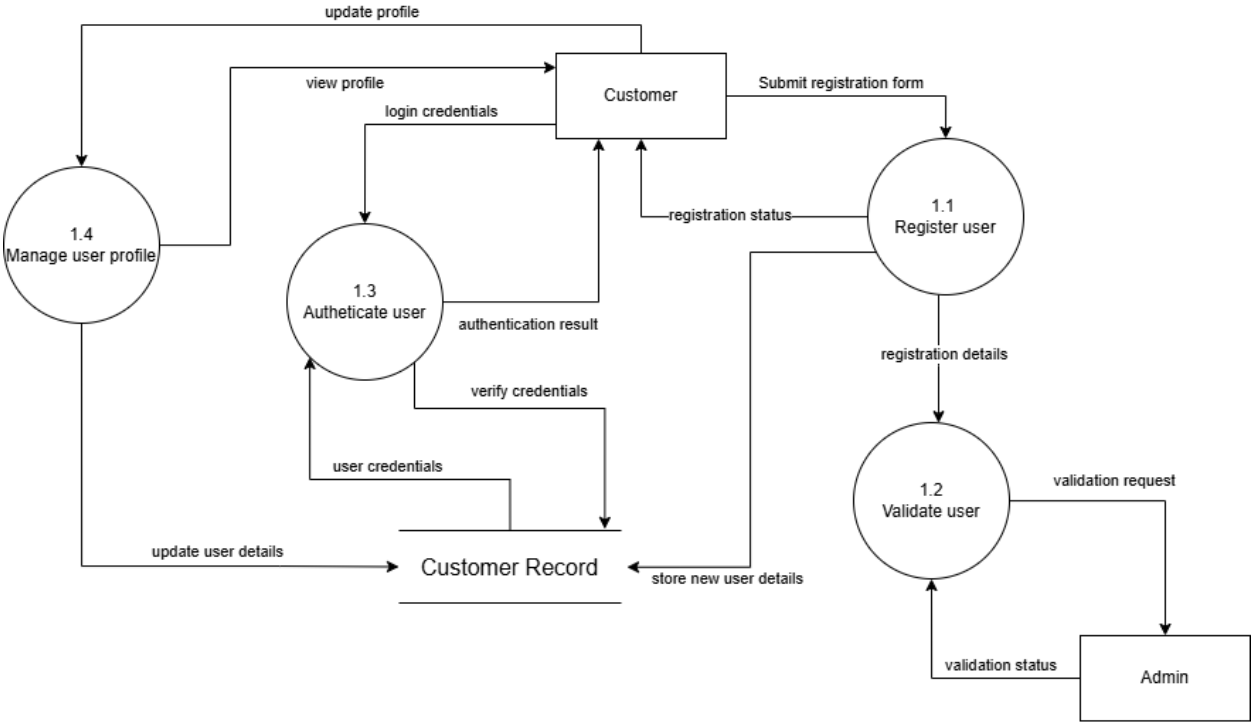


Figure 3: Level 2 DFD of User Management

Let's break down each component and its interactions:

1. Process 1.1: Register user

- Receives registration form from Customer
- Processes initial registration
- Sends registration status back to Customer
- Forwards registration details to validation process

2. Process 1.2: Validate user

- Receives registration details from Process 1.1
- Sends validation request to Admin
- Receives validation status from Admin
- Stores new user details in Customer Record

3. Process 1.3: Authenticate user

- Receives login credentials from Customer
- Verifies credentials against Customer Record
- Returns authentication result to Customer

4. Process 1.4: Manage user profile

- Handles profile updates from Customer
- Updates user details in Customer Record
- Allows customers to view their profile

Data Flows:

1. Customer → System:
   - Submit registration form
   - Login credentials
   - Update profile requests
   - View profile requests

2. System → Customer:
   - Registration status
   - Authentication result
   - Profile view data

3. System ↔ Admin:
    - Validation requests
    - Validation status

4. System ↔ Customer Record:
    - Store new user details
    - Update user details
    - Verify credentials
    - Retrieve profile information

This Level 2 DFD shows how the user management system handles the complete lifecycle of user interactions, from initial registration through to ongoing profile management, while maintaining appropriate validation and authentication steps.

9.2 Level 2 DFD of Sales management

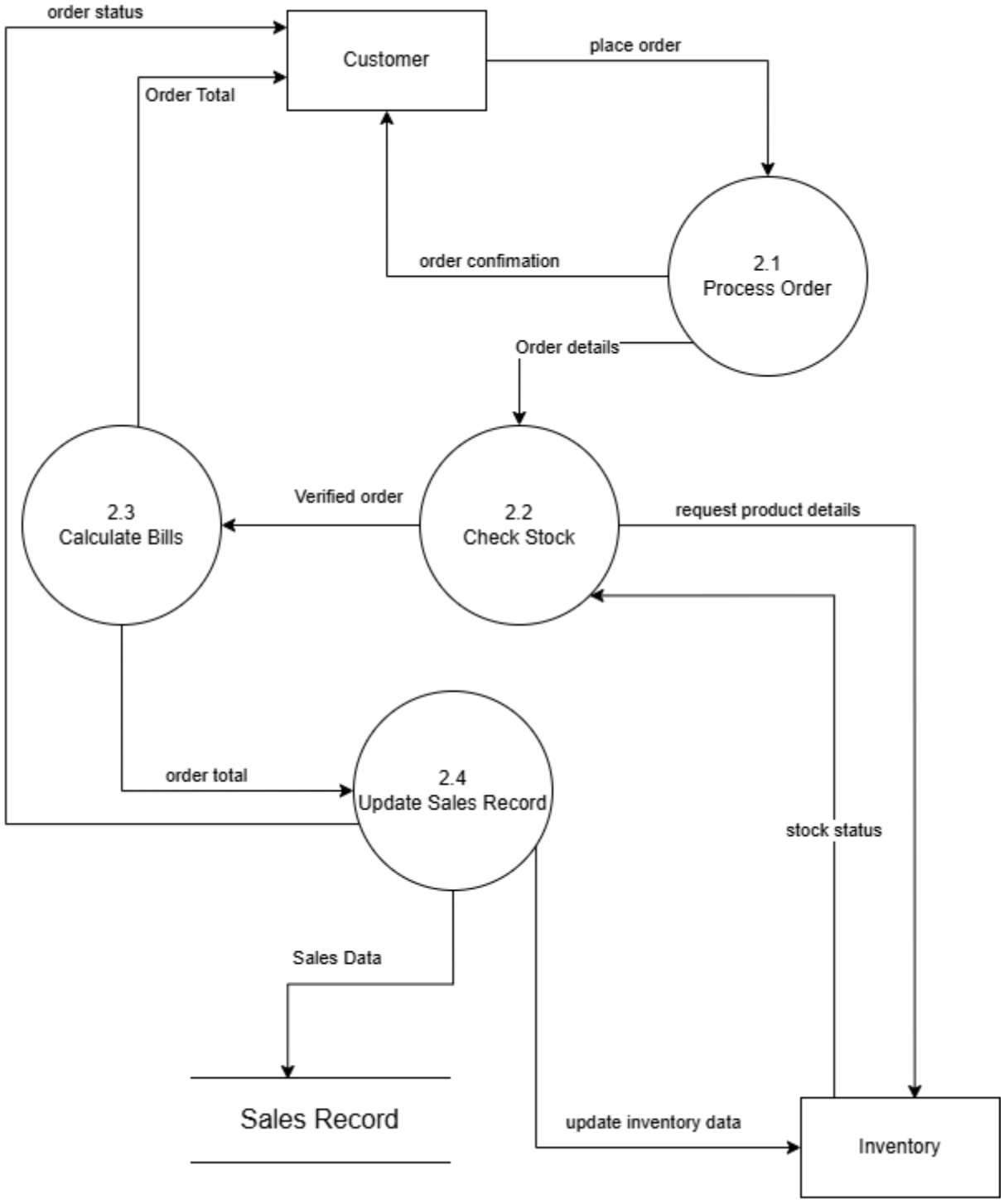


Figure 4: Level 2 DFD of sales management

Let's break down each component and its interactions:

1. Process 2.1: Process Order

- Receives initial order from Customer
- Sends order confirmation back to Customer
- Forwards order details to Check Stock process
- Primary function: Initial order processing and validation

2. Process 2.2: Check Stock

- Receives order details from Process Order
- Checks product availability with Inventory
- Gets stock status from Inventory
- Sends verified order to Calculate Bills
- Primary function: Inventory verification

3. Process 2.3: Calculate Bills

- Receives verified order from Check Stock
- Calculates total amount
- Sends order total to Customer
- Forwards order total to Update Sales Record
- Primary function: Bill calculation

4. Process 2.4: Update Sales Record

- Receives order total from Calculate Bills
- Updates Sales Record with transaction data
- Updates Inventory with new stock levels
- Sends order status to Customer
- Primary function: Record keeping

Data Flows:

1. Customer Interactions:
    - Places order (input)
    - Receives order confirmation
    - Receives order total
    - Receives order status

2. Inventory Interactions:
    - Provides stock status
    - Receives inventory updates

3. Sales Record Interactions:
    - Stores sales data
    - Maintains transaction records

This Level 2 DFD shows the complete cycle of a sales transaction, from order placement through stock checking, billing, and record updating. Each process has a specific responsibility in the sales management workflow, ensuring proper order processing, inventory management, and record keeping.

9.3 Level 2 DFD of Product Management

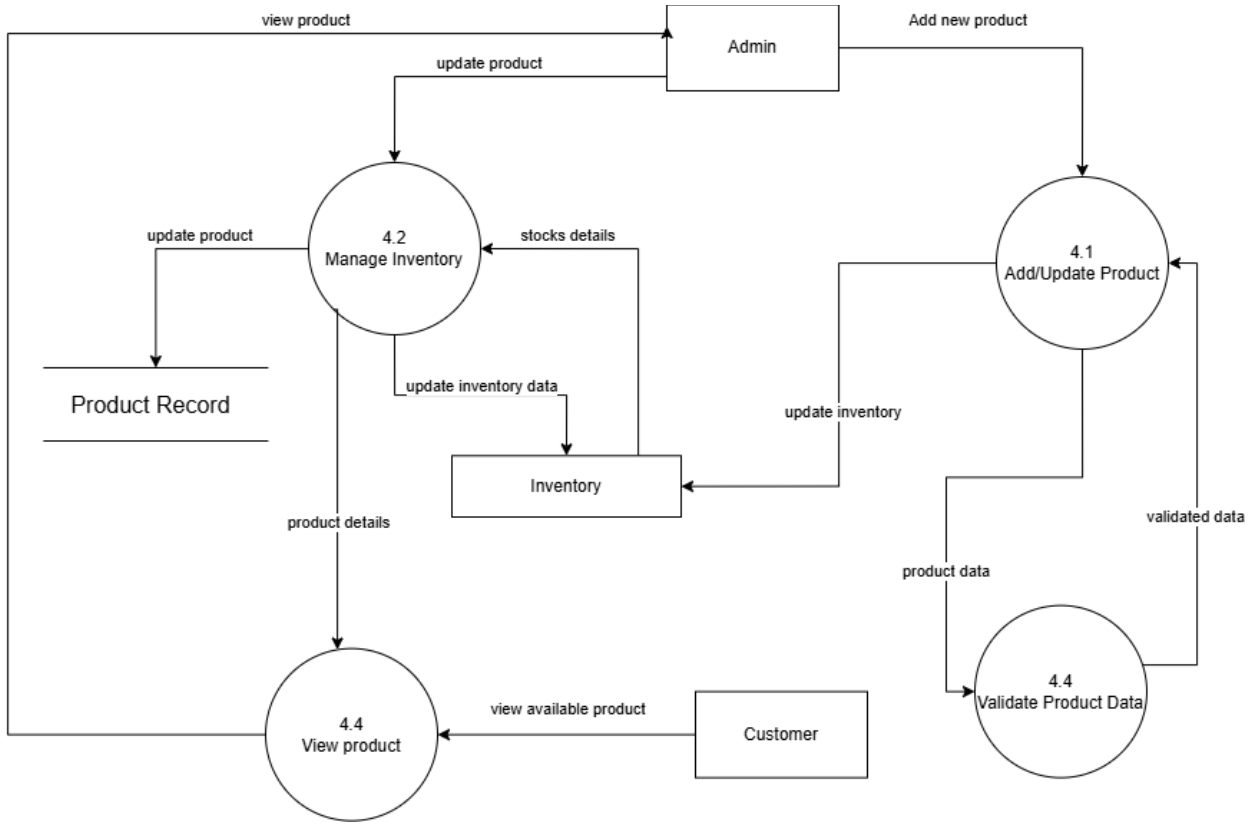


Figure 5: Level 2 DFD of product management

Let's break down and explain each component and its interactions:

1. Process 4.1: Add/Update Product

- Receives new product information from Admin
- Sends product data for validation
- Updates inventory with new product information
- Primary function: Product creation and modification

2. Process 4.2: Manage Inventory

- Receives stock details from Inventory
- Updates inventory data
- Updates Product Record
- Primary function: Inventory management

3. Process 4.3: View Product

- Gets product details from Product Record
- Shows available products to Customer
- Provides product information to Admin
- Primary function: Product information display

4. Process 4.4: Validate Product Data

- Receives product data from Add/Update Product
- Validates the information
- Returns validated data
- Primary function: Data validation

Data Stores:

1. Product Record: Stores all product information
2. Inventory: Maintains current stock levels

Data Flows:

1. Admin Interactions:
   - Adding new products
   - Updating existing products
   - Viewing product details

2. Customer Interactions:
   - Viewing available products

3. Internal Data Flows:
    - Product data validation
    - Inventory updates
    - Stock management
    - Product information retrieval

This Level 2 DFD demonstrates how the product management system handles:

- Product addition and updates
- Inventory management
- Product viewing for both admin and customers
- Data validation

The diagram shows the complete cycle of product management from creation through to customer viewing, with appropriate validation and record-keeping steps throughout the process.

10 E-R Diagram

An Entity-Relationship Diagram (ERD) is a visual representation of the data model for a system. It illustrates how entities (such as objects, people, or concepts) relate to each other within a database. ERDs are commonly used in database design to structure data efficiently and ensure logical relationships among entities.

Entities and Attributes

10.1 ERD

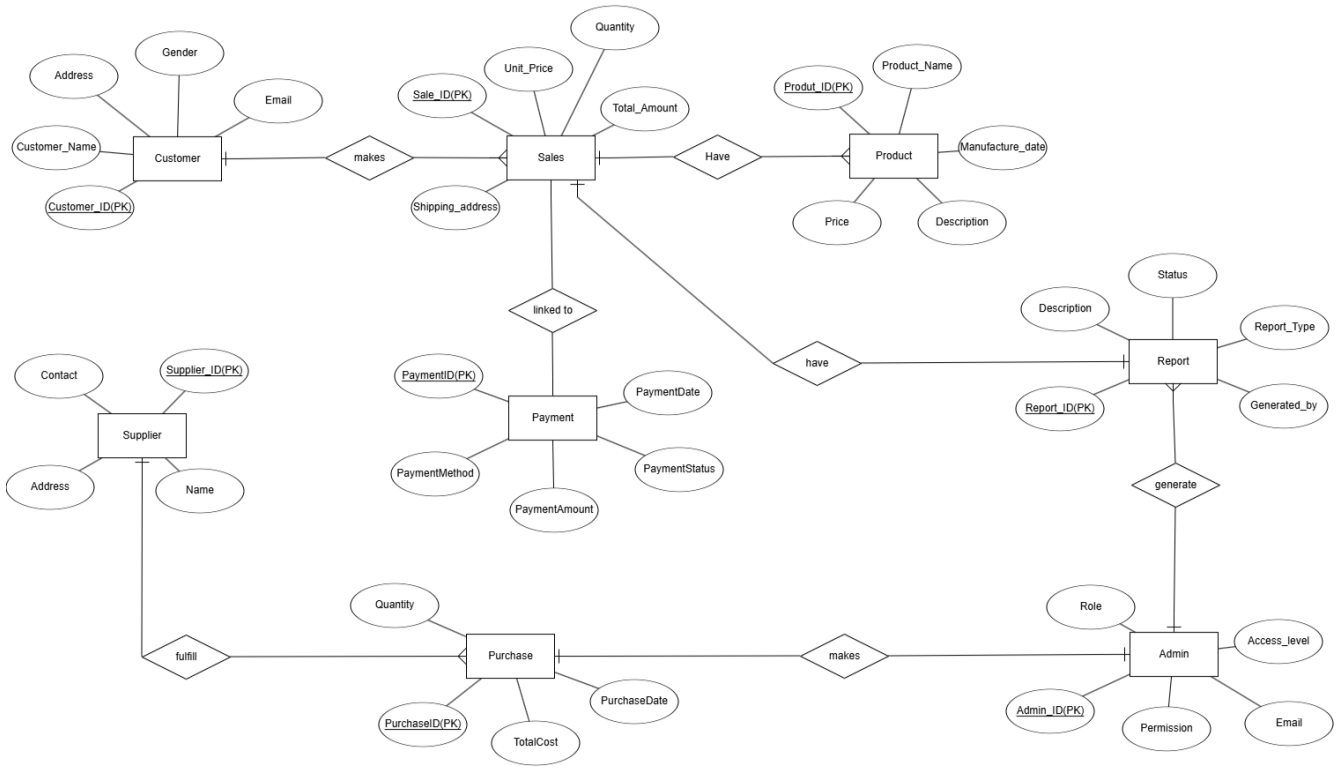


Figure 6: ERD diagram

11 Data Dictionary

A data dictionary can be described as a document or a set of files that hold metadata on the related database. Such containers contain information about other objects in the database, for example, data on ownership of data, data on the relationship of data to other objects, and other data. Data dictionary is one of the most important qualitative factors that would exist in any website that used relational database model. Actually, since it goes right to the core of what a database does, most users are – for all intents and purposes – unaware of it. It is mostly only the database administrator who communicates with the data dictionary.

Customer

Customer = Customer_details + Customer record /* data stores in database */

Customer_record = {Customer_details}*

Customer_details = Customer_ID + Customer_Name + (Address) + [Email, Phone_Number]

Customer_Name = Full_Name + (Middle_Name) + Last_Name

Gender = [“Male”, “Female”, “Rather]

Customer_ID = Number

Customer_Name = String

Address = String

Email = String

Phone_Number = Number

Report

Report = Report_details + Report_record /*data stores*/

Report_record = {Report_details}*

Report_details = Report_id + report_type + generated_by + report_status + description

Report_type = [“user_report”, “sales_report”, “purchase_report”]

Report_id = number

Report_name = String

Generated_by = String

Description = String

Purchase

Purchase = Purchase_details + Purchase_record

/* data stores */

Purchase_record = {Purchase_details}*

Purchase_details = Purchase_ID + Purchase_date + Customer_ID + Product_ID + Quantity + Total_amount

Purchase_ID = Number

Purchase_date = Date

Customer_ID = Number (Foreign key referencing from Customer table)

Product_ID = Number (Foreign key referencing from Product table)

Quantity = Number

Total_amount = Number

Sales

Sales = Sales_details + Sales_record

/* data stores */

Sales_record = {Sales_details}*

Sales_details = Sales_ID + Quantity + Unit_price + Total_amount + Shipping_address

Sales_ID = Number

Quantity = Number

Unit_price = Number

Total_amount = Number

Shipping_address = String

Payment

Payment = Payment_details + Payment_record

/* data stores */

Payment_record = {Payment_details}*

Payment_details = Payment_ID + Payment_method + Payment_date + Payment_status + Payment_Amount

Payment_ID = Number

Payment_Amount = Number

Payment_method = [“Credit Card”, “Debit Card”, “Cash”]

Payment_date = Date

Payment_status = [“Pending”, “Approved”, “Rejected”]

Product

Product = Product_details + Product_record

/* data stores */

Product_record = {Product_details}*

Product_details = Product_ID + Product_name + Description + Price + Manufacture_date

Product_ID = Number

Product_name = String

Description = String

Price = Number

Manufacture_date = Date

Admin

Admin = Admin_details + Admin_record

/* data stores */

Admin_record = {Admin_details}*

Admin_details = Admin_id + Admin_name + Email + Role + Access_level + Permissions

Admin_id = Number

Admin_name = String

Email = String

Role = [“Super Admin”, “Manager”, “User”]

Access_level = Number (e.g., 1, 2, 3 representing different access levels)

Permissions = [“View Reports”, “Add Products”, “Edit Customers”]

Supplier

Supplier = Supplier_details + Supplier_record /* data stores */

Supplier_record = {Supplier_details}*

Supplier_details = Supplier_ID + Name + Address + Contact

Supplier_ID = Number (Primary Key)

Name = String

Address = String

Contact = String

12 Structured Chart

A structured chart is a visible record of a system which portrays the system's structure in terms of its most atomic functional components or elements. This makes it less complex because most functions and dependencies of systems are put in a well-ordered manner in the format of a tree. Another nice thing about the structured chart is that each of the modules is described methodically and both the functional and non-functional character of the system is described. These charts are often used to coordinate and plan complex systems because it greatly aids in tracking. The type of symbols is employed to help layout well-organized charts. Below are the key elements and their descriptions:

1. **Module**

   A module is a particular segment of the system that might be detailed into other segments. In the structured chart, it is illustrated using a rectangular box ▭ Types of modules are listed below;

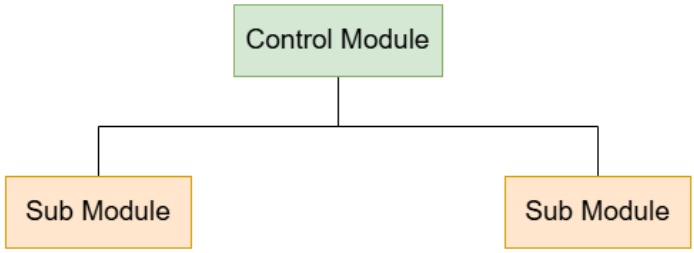


Figure 7: Module diagram

- Control Module:

  This is the main or original module that controls the general functionality and that serves to divide into submodules. This can be regarded as an organizational hub that oversees the exercise of control within the system and directs it properly.

- Submodule :

  Submodules are subtypes generated from control module. They perform specific operations that are in the big picture of the system. Each submodule is connected with the parent module and relies on the control module to provide tasks.

2. Conditional call

A conditional call shows that a control module can choose to factually call one or more submodules if some conditions are met. This feature is helpful when decisions have to be made on the fly in one or more of the interacting systems.

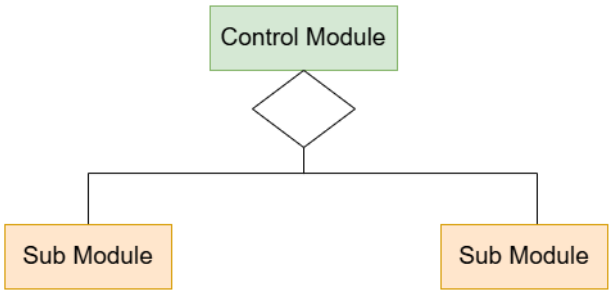


Figure 8: Conditional call diagram

3. Data Flow

Data flow is an actual movement of data from one module to another within the system. This maps several steps and processes ranging from input, through the output and the processed data in the system architecture.

Purpose:

By putting these into practice there is clear indication on how information is disseminated and managed in the different modules.

Representation:

Data flow is vicariously depicted using arrow:

4. Control Flow

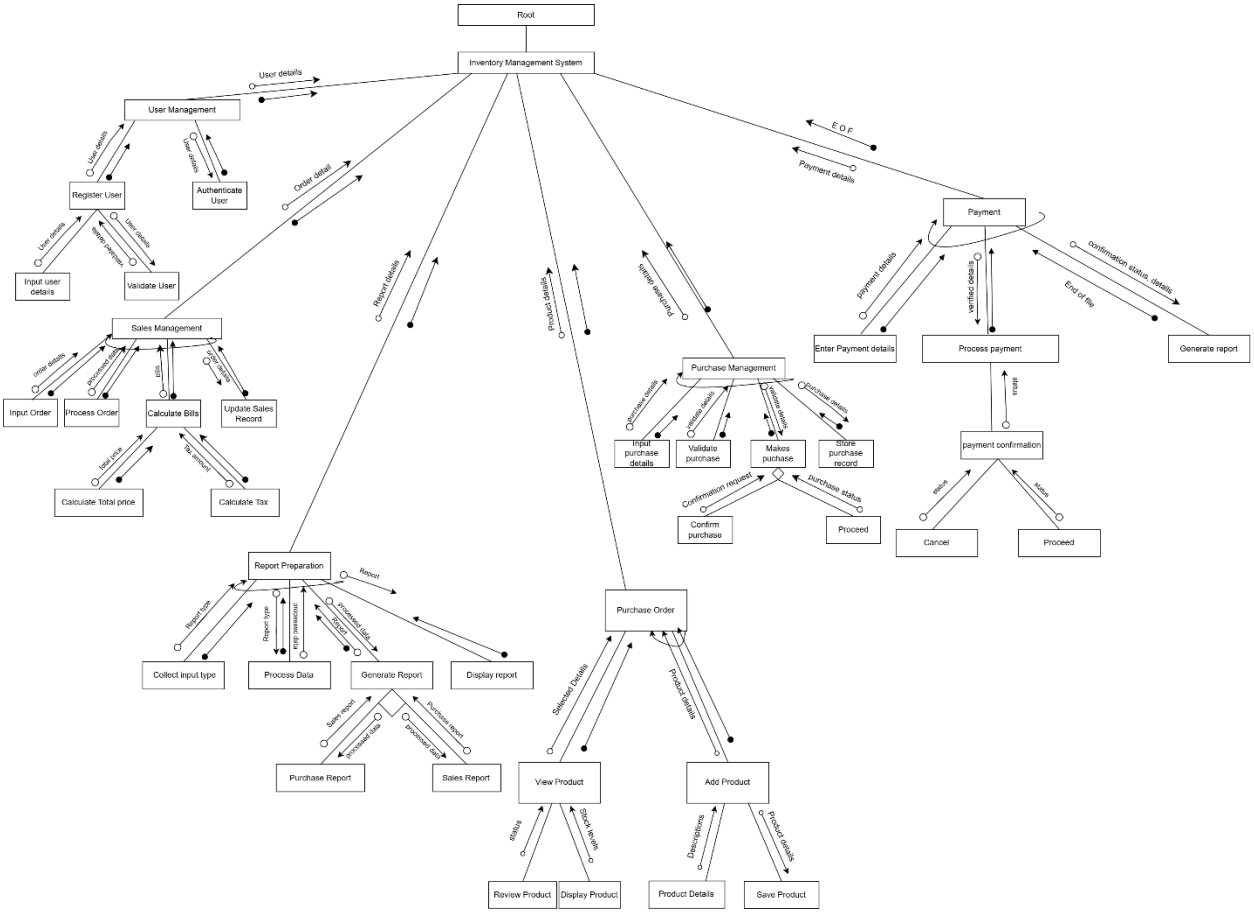
Control flow is the transfer of control, commands or verification signal between modules. It describes logical ordering, and dependencies, that execute programs.

Purpose:

To show how the system receives commands and how control is passed from one module to another.

Representation:

Control flow is depicted as arrow: 


Root
Inventory Management System
User details
User Management
User details
Register User
User details
Authenticate User
User details
Input user details
Validate User
Order detail
Sales Management
Input Order
Process Order
Calculate Bills
Update Sales Record
Calculate Total price
Calculate Tax
Report details
Product details
Purchase details
Purchase Management
Input purchase details
Validate purchase
Makes puchase
Store purchase record
Confirm purchase
Proceed
E O F
Payment details
Payment
Enter Payment details
Process payment
Generate report
payment confirmation
Cancel
Proceed
Report Preparation
Collect input type
Process Data
Generate Report
Display report
Purchase Report
Sales Report
Purchase Order
View Product
Add Product
Review Product
Display Product
Product Details
Save Product

13 Process Specification

A process specification is a method used in identifying describing and justifying the decision making procedures and formula which are used in mapping input data into output data of a process. Its purpose is downward distribution and to inform on regulatory and/or engineering standards and procedures. Therefore, the most effective means of guaranteeing near perfect quality and consistence in the data collected is the definition of precise and unambiguous process requirements.

13.1 User management

Process ID: 1.1

Process Name: Register User Process

Description: This process generates a customer to register by submitting their details through a registration form. The system captures the details and forwards them for validation.

Process Input:

- Registration form submitted by customer.

Process Output:

- The Registration form submitted by the customer is validated.

Process flow.

1. The Customer submits the registration form and provides their details, such as name, email, password, and any other required information , through a registration form on the system's interfaces.
2. The system captures the provided information and temporarily stores it in the database.
3. Once the form is processed successfully , the system provides immediate feedback to the customer.
4. Now , this registration details are sent to the validate user process.

13.2 Generate Report

Process ID : 3.1

process name: Validate process data

Process description : This process generate that the input data for generating reports meets quality standards, format specification .

Process input : Retrieve transaction records

Process output : 1 Generates properly formatted report

2 Stores and share the report with authorized users and admin.

Process Flow:

This process takes the finalized and refined data from the “preview data” process and retrives previous reports from the “sales managements” data store.

Now this process then create the report in required format and then compiles the final report based on given data.

This process then securely store the final generated report in report record data store for future access and auditing.

It then share the generated report via email to admin for stakeholders.

13.3 Payment

Process ID: 3.1

Process name : Collect Payment detail

Process description : Collect payment information provided by the customer

Process inputs : 1. Receive payment information

Process Output : Stores collected information from customers

Process flow:

1.This process takes the initial payment details from the customer and stores in payment records.

2. Then this process checks the provided information is completed and in proper format.

3. Now, it securely stores the collected payment information for further validation and processing.

4.After that, it sends the stored validated details for the next authentication and authorization.

13.4 Product Management

Process ID: 4.2

Process Name: Purchase Inventory

Process Description:

This makes and ensures that data from the transactions could be used when monitoring inventory levels and even when making changes on the inventory data. They include, pulling information of the stock in the inventories, updating the figures of the stock and monitoring the status of inventories. It also generates a notification of low inventory and safely sends out the changed inventories.

Process Input:

- Retrieve Inventory Transaction Records

Process Output:

- Updated Inventory Data
- Inventory Alerts
- Secure Storage of Inventory Data

Process Flow:

1. Retrieve Transaction Data:

The first process of this automation entails pulling of current data involving inventories including stock movements, sales and returns.

2. Access Existing Inventory Data:

This one compares and update the data with the inventory management system by selecting the existing inventory data.

3. Update Inventory Levels:

The data information from the specific transaction also causes the system to adjust the current inventory stock quantity too.

4. Generate Inventory Alerts:

This process involves a comparison between the value on the existing stock and the value fixed so as to give the required standard. If there is any stock lower than the necessary minimum quantity a notification is sent to the inventory manager as well as to the stakeholders.

13.5 Sales Management

Sales management

Update Sales Record

Process ID: 2.4

Process Name: Update Sales Record

Process description: This process updates the sales record with the latest transaction data and updates the inventory system to reflect changes based on the verified order.

Process input:

It Verified order details, order total and stock status to ensure accurate updates.

Process output:

It Updated sales records and inventory levels to reflect the completed transaction.

Process flow:

This process Receive the verified order details from Preview data calculate bills .

Now , it update the sales record with the transaction details.

Then This process send sales data to the sales Record storage.

It then Update the inventory database to reflect the changes based on stock status.

14 Progress Logs

14.1 Meeting 01

Meeting Logbook

Meeting No: 01 **Date: 24/11/2024**

Start Time: 8:00 **End Time: 10:00**

Items Discussed:

1. Business Case Structure.
2. Coursework Breakdown
3. Team Assignments

Achievements:

- We finalized the structure and scope of the Business Case.
- We divide the coursework task for each member.

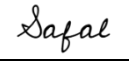
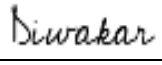
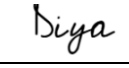
Problems:

We faced some problem regarding coursework tasks and Business case. There were some issues about estimating costs for software and estimating Quantitative benefits for the project.

Task for next meeting:

1. Completion of Business Case

Student signature:

Student Name:	Signature:
Safal Bhattarai	
Diwakar Yadav	
Sanjita Chaudhary	
Aashish Khadka	
Diya Rai	

14.2 Meeting 02

Meeting Logbook

Meeting No: 02 **Date: 25/11/2024**

Start Time: 9:00 PM **End Time: 10:00 PM**

Items Discussed:

1. Finalise the Business Case
2. Content Refinements
3. Formatting and Submission

Achievements:

- Completed and Finalized all sections of Business Case.

Problems:

Limited time to ensure the business case is totally erorr-free.

Task for next meeting:

1. SRS

Student signature:

Student Name:	Signature:
Safal Bhattarai	Safal
Diwakar Yadav	Diwakar
Sanjita Chaudhary	Sanjita
Aashish Khadka	Aashish
Diya Rai	Diya

14.3 Meeting 03

Meeting Logbook

Meeting No: 03 **Date: 10/12/2024**

Start Time: 8:00 PM **End Time: 11:00 PM**

Items Discussed:

1. SRS content
2. SRS format
3. Group and Individual DFD

Achievements:

- Completion of SRS

Problems:

Time management: Difficult in balancing individual and group task together.

Unclear processes: Individual DFD need further clarification

Task for next meeting:

1. Completion of Group and Individual DFD
2. ER-Diagram

Student signature:

Student Name:	**Signature:**
Safal Bhattarai	Safal
Diwakar Yadav	Diwakar
Sanjita Chaudhary	Sanjita
Aashish Khadka	Aashish
Diya Rai	Diya

14.4 Meeting 04

Meeting Logbook

Meeting No: 04 **Date: 14/12/2024**

Start Time: 7:00 PM **End Time: 10:00 PM**

Items Discussed:

1. DFD both group and individual
2. ER-Diagram

Achievements:

- Completion of DFD both group and individual
- Completion of ER-Diagram
- Final formatting for milestone 2

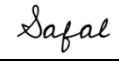
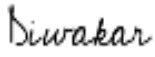
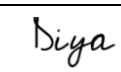
Problems:

Minor mismatches in data flow in group dfd were identified and resolved. Ensuring alignment between the DFD processes and the ERD was time consuming.

Task for next meeting:

1. Completion of Group and Individual DFD
2. ER-Diagram

Student signature:

Student Name:	Signature:
Safal Bhattarai	
Diwakar Yadav	
Sanjita Chaudhary	
Aashish Khadka	
Diya Rai	

14.5 Meeting 05

Meeting Logbook

Meeting No: 05 **Date: 20/12/2024**

Start Time: 8:00 PM **End Time: 11:00 PM**

Items Discussed:

1. Data Dictionary
2. Structure Chart of both group and individual
3. Module Specification of individual

Achievements:

- Completion of data dictionary
- Draft completion of structure chart of individual
- Completion of group structure chart

Problems:

There was specification Depth problem as some module specifications lacked detailed input/output descriptions and needed additional work. Due to lack of time individual structure chart is pending.

Task for next meeting:

1. Finalise the document for milestone
2. Completion of process specification and individual structure chart

Student signature:

Student Name:	Signature:
Safal Bhattarai	Safal
Diwakar Yadav	Diwakar
Sanjita Chaudhary	Sanjita
Aashish Khadka	Aashish
Diya Rai	Diya

14.6 Meeting 06

Meeting Logbook

Meeting No: 05 **Date: 20/12/2024**

Start Time: 8:00 PM **End Time: 11:00 PM**

Items Discussed:

4. Process specification
5. Final completion of document

Achievements:

- Completed the final draft of the document
- Finalized the process specifications

Problems:

Minor adjustments to the DFDs and ERD required additional time. Ensuring the specification alignment perfectly aligned with the DFDs required extra discussion and time.

Task for next meeting:

3. Final Review of document

Student signature:

Student Name:	**Signature:**
Safal Bhattarai	Safal
Diwakar Yadav	Diwakar
Sanjita Chaudhary	Sanjita
Aashish Khadka	Aashish
Diya Rai	Diya

Individual Task

15 Diwakar Yadav: Purchase Order

London-met id: 23049236

College id: np05cp4a230017

15.1 Context Level Diagram

Using a Level 0 DFD of the Purchase Order System, it is noted how the system interacts with other external systems and preliminary activities. It paints the system in a separate entity and does not consider the smaller aspects of the system.

Processes:

1. Receive Purchase Request:

This process is receiving info from Customer Order and that contains details of what is required, the quantity of each product and where it should be taken.

2. Validate Purchase Request:

Informs the buyer that all the details that are needed for the order and delivery are correct and performs a check if each of the ordered products is in stock.

3. Generate Purchase Order:

It also results in the creation of the official purchased order document which has to be presented to the Vendor on validation of this process.

4. Send Purchase Order to Vendor:

This process ensure that the Purchase order is communicated to the Vendor to enable him to process it.

5. Receive Vendor Confirmation:

Receives acknowledgement from the Vendor as to the status of that order that may be accepted, rejected or modified.

External Entities:

1. Customer:

Once the order is created the prices are locked and the order request is when ordering process starts in the system.

2. Vendor:

Eventually, pays for the order and sends an acknowledgement of that order and the purchase order.

3. Management:

Is involved in the purchase process and checks what is going on by looking for himself in the vendor's confirmations

Figure 9: Level 0 DFD

15.2 Level 1 DFD

The Level 1 of the DFD reduces the complexity of the Purchase Order System and presents how the latter functions, step-by-step.

Processes:

Receive Purchase Request:

The Fig below shows the customer enters a purchase order to the system which comes with information about the product the customer wants to purchase.

Validate Purchase Request:

The identification of whether the products ordered were in store or not the next step is followed by the system while performing the order-processing of a wrong order detail.

Generate Purchase Order:

For the last of the validation process, the system also produces an actual purchase order form normally used in the company.

Send Purchase Order to Vendor:

It then forwards the purchase order which it has created to the Vendor.

Receive Vendor Confirmation:

The system receives response from the Vendor, whether he accepts the order or not and the status of the delivery.

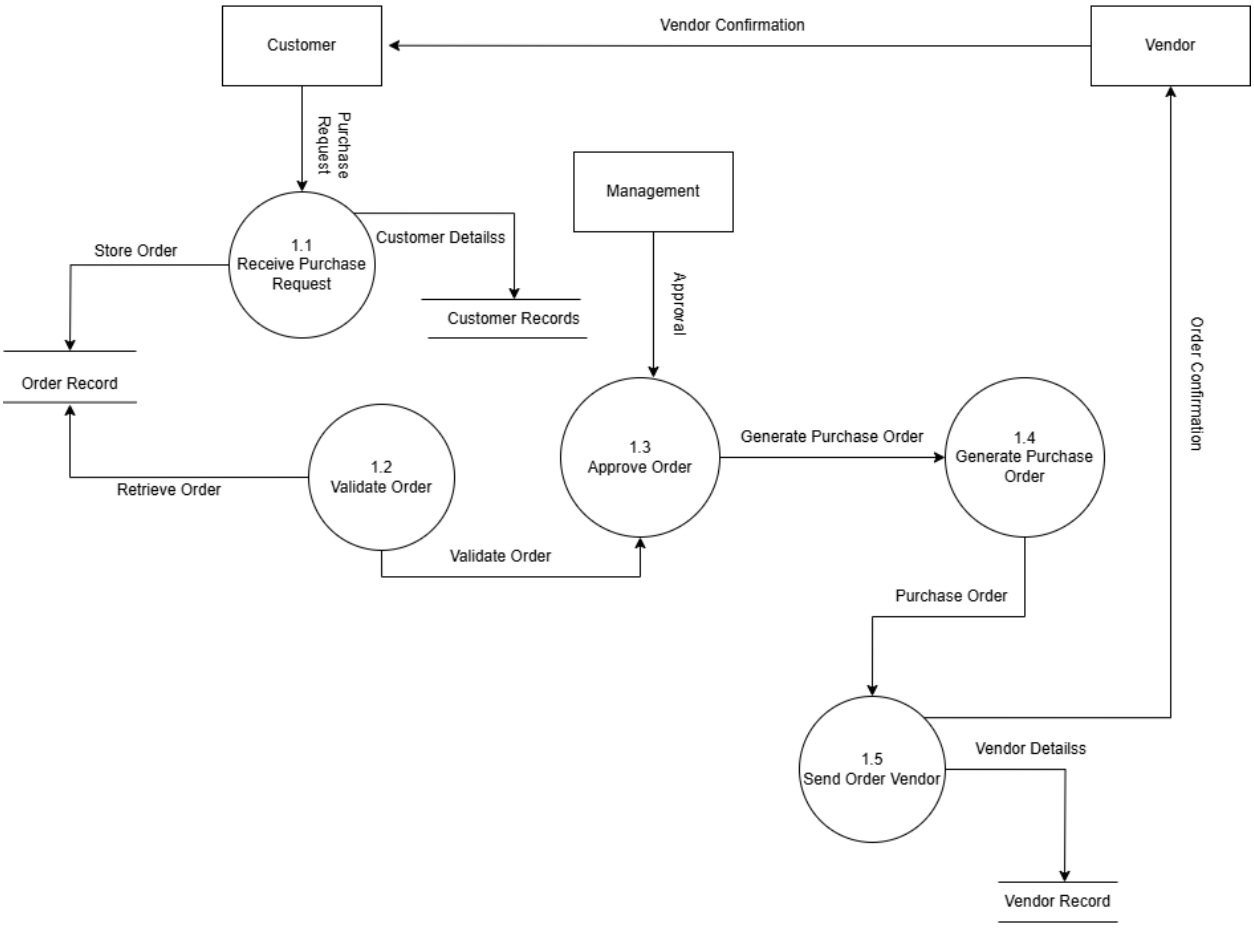


Figure 10: Level 1 DFD of purchase order

15.3 Level 2 DFD

In the context of a Purchase Order System a high level Data Flow Diagram or DFD will represent a complex activity. This explains how the system generates purchase orders to the Customer, Vendor and the Management.

This process starts with the Customer initiating a Purchase Request to The system. This is a request that the system receives and responds to, asking it whether it is true that all the details provided are accurate; as well as whether the items the user is asking for are in stock. After that the system generates an actual Purchase Order after an order has been made and confirmed. It is a business document that requires the Buyer to prepare

and forward it to the Vendor for the Vendor to prepare the Vendor Confirmation showing the position of the business purchase order regarding several products and services. Then it sends this confirmation to Management, to check processes in order to know if it's correct.

As we explore from the Level 0 DFD, this only includes passing of data from the system to the external entities and within the system, more of the processes involved are not well represented to a certain extent. And goes to the important activities and the communication that occur during the purchase order process.

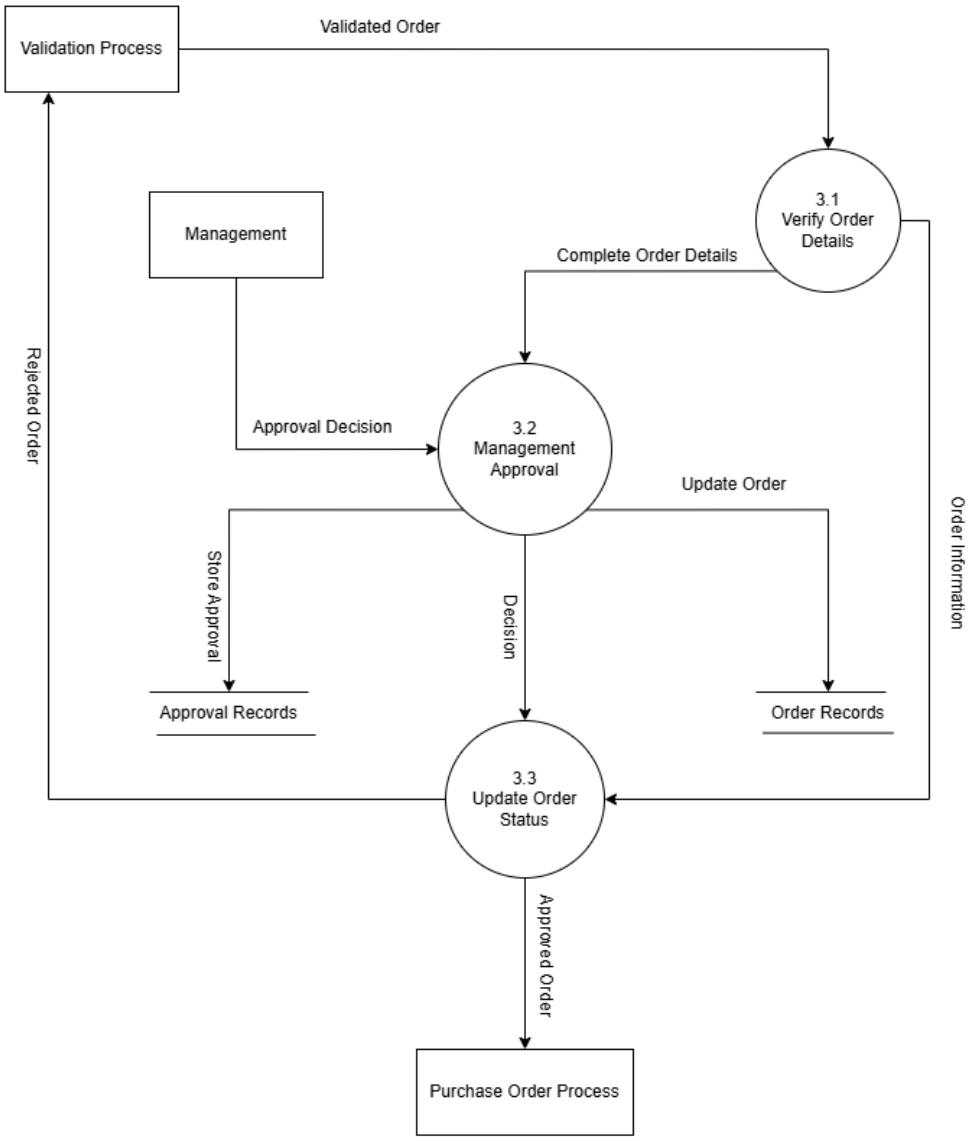


Figure 3: Level 2 DFD

15.4 Structured Chart

15.5 Module Specification

15.5.1 Module Name: Purchase Order

Purpose

15.5.2 Pseudocode

```
DO
    /* Step 1: Capture user input for Purchase Order details */
    VAR supplierID = INPUT("Enter Supplier ID:")
    VAR itemID = INPUT("Enter Item ID:")
    VAR quantity = INPUT("Enter Quantity:")
    VAR unitPrice = INPUT("Enter Unit Price:")
    VAR deliveryDate = INPUT("Enter Delivery Date:"
    /* Step 2: Validate Supplier */
    IF NOT ValidateSupplier(supplierID) THEN
        PRINT("Invalid Supplier ID. Please try again.")
        RETURN
    END IF
    /* Step 3: Validate Item */
    IF NOT ValidateItem(itemID) THEN
        PRINT("Invalid Item ID. Please try again.")
```

```
  RETURN

   END IF

   /* Step 4: Calculate total cost */

   VAR totalCost = quantity * unitPrice

   /* Step 5: Generate a unique Purchase Order ID */

   VAR purchaseOrderID = GenerateUniqueID()

   /* Step 6: Save Purchase Order to the database */

   SaveToDatabase(purchaseOrderID, supplierID, itemID, quantity, unitPrice, totalCost,
deliveryDate)

   /* Step 7: Confirm success */

   PRINT("Purchase Order Created Successfully with ID: " + purchaseOrderID)

END DO
```

15.5.3 Input Parameters:

- supplierID(String): An identifying number, which is assigned to each supplier with whom the organization deals.
- itemID(String): Unique identification number assigned to the item which is being ordered.
- quantity(Integer): Number of units to be purchased Each item ordered in large quantities.

15.5.4 Output Parameters:

- purchaseOrderID(String): Newly created purchase order is associated with an identification number.
- confirmationMessage(String): Generally, a status message that reflects the status of the message produced due to a purchase order.

15.5.5 Local Variables:

- supplierID(String): The place holder where the user gets to enter the identified supplier.
- itemID(String): Saves the ID number that is entered by the user.

- quantity(Integer): Storing the number of substances that the user entered in a certain form.

15.5.6 Global Variables:

- PurchaseOrderDB(Data Store): Consists of all the documents of the purchase order he has created.
- SupplierDB (Data Store): Contains all the details about any supplier who ever registered on the platform.

15.5.7 Calls:

- ValidateSupplier(supplierID)`: Verifies the supplier recorded by the customer exists in SupplierDB.
- ValidateItem(itemID): It determines first if the Item exists and second if it exists in a local database called Itemdb.

15.5.8 Called By:

Procurement Integrated Management

- Inventory Management System

- Order Processing System

16 Safal Bhattarai: Report Preparation

London-met id: 23049395

College id: np0cp4a230039

16.1 Context Level Diagram

This context diagram shows how the Report Preparation system interacts with three main stakeholders:

1. Customers who can request and view purchase reports

2. Administrators who can request specific reports (like sales and profit/loss) and view sales reports

3. The Inventory System which provides the necessary inventory data for report generation

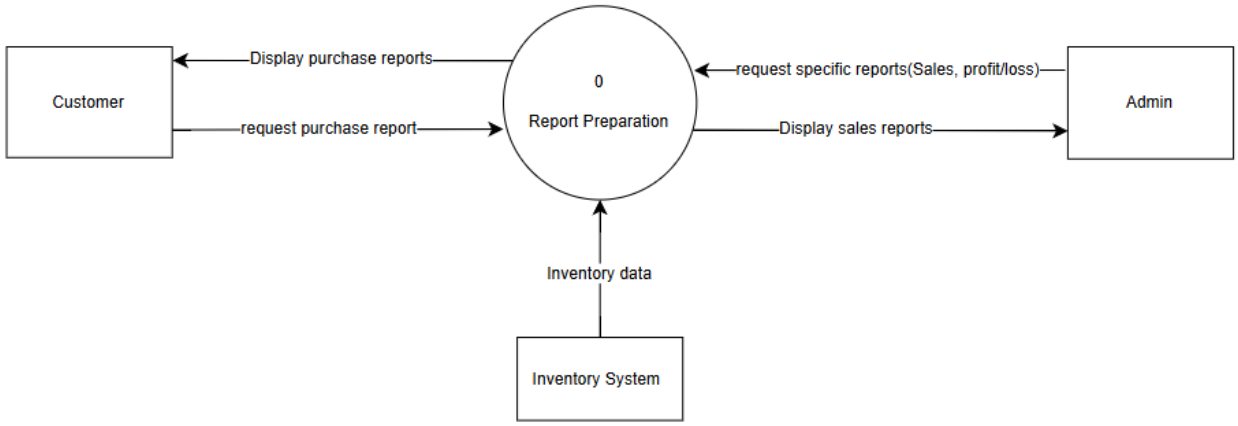


Figure 11: Context Diagram of report preparations

The diagram illustrates the basic data flows and interactions between these entities and the central Report Preparation process, providing a high-level view of the system's scope and boundaries.

16.2 Level 1 DFD

This Level 1 DFD breaks down the report generation system into five main processes:

1. Process 1: Collect Input Data

- Receives report requests from both customers and admin
- Handles initial data collection requirements
- Forwards requests to data retrieval process

2. Process 2: Retrieve Data

- Gets sales data from Sales Data store
- Organizes raw data for processing
- Sends collected data to Process Data

3. Process 3: Process Data

- Receives raw data from Retrieve Data
- Performs data analysis and formatting
- Prepares processed data for report generation

4. Process 4: Generate Report

- Creates formatted reports from processed data
- Stores report details in Report Record
- Forwards completed reports for delivery

5. Process 5: Deliver Reports

- Receives generated reports
- Delivers appropriate reports to requesters:
    - Purchase reports to customers
    - Sales reports to admin

Data Flows:

- Customer Interactions:
    - Sends purchase report requests
    - Receives purchase reports

- Admin Interactions:
    - Requests sales reports
    - Receives sales reports

Data Store Interactions:

- Sales Data: Provides historical sales information
- Report Record: Stores generated reports

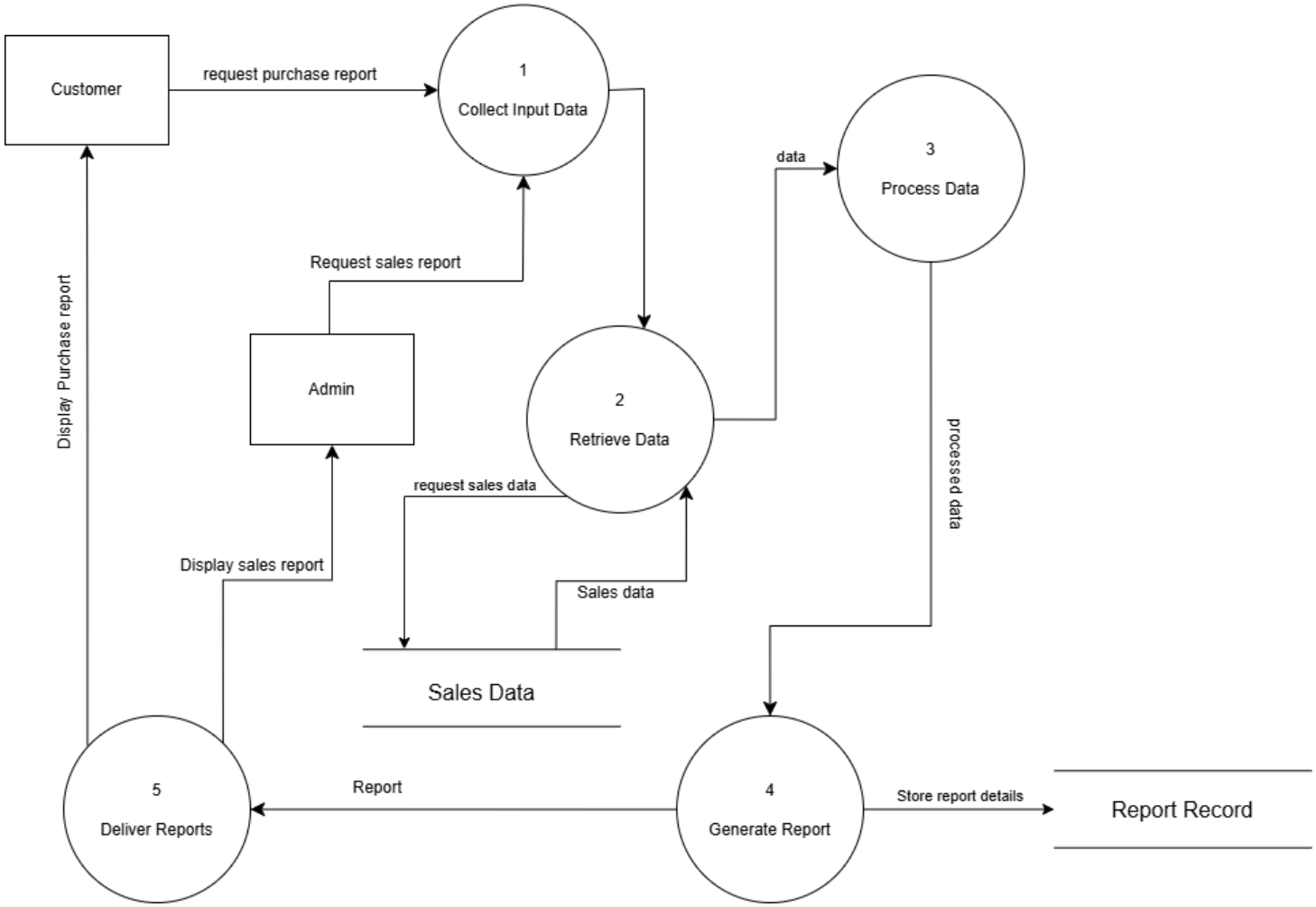


Figure 12: Level 1 DFD of Report Preparations

The diagram shows how data flows through these processes, starting from user requests and ending with report delivery, while interacting with various data stores along the way.

16.3 Level 2 DFD for Generate report

This Level 2 DFD shows 5 sub process:

1. Process 4.1: Validate Process Data

- Input: Receives processed data from Process 3 (Process Data)
- Function: Validates the accuracy and completeness of data
- Output: Sends validated data to Populate Report

2. Process 4.2: Create Report Format

- Function: Defines the structure and layout of the report
- Output: Provides report structure to Populate Report
- Determines format based on report type (sales/purchase)

3. Process 4.3: Populate Report

- Inputs:
    - Validated data from Process 4.1
    - Report structure from Process 4.2
- Function: Fills the report template with validated data
- Output: Creates draft report for verification

Process 4.4: Verify Report

- Input: Receives draft report
- Function: Reviews for accuracy and completeness
- Outputs:
    - Sends final report to Deliver Reports (Process 5)
    - Forwards final report for storage

5. Process 4.5: Store Report Details

- Input: Receives final verified report
- Function: Manages report storage
- Output: Stores report details in Report Record

Data Flows:

1. Input Flow: Processed data enters from Process 3
2. Internal Flows:
   - Validated data flows to report population
   - Report structure guides population
   - Draft report moves to verification
   - Final report goes to storage and delivery
3. Output Flows:
   - Final report sent to Process 5 (Deliver Reports)
   - Report details stored in Report Record

This Level 2 DFD shows the detailed steps involved in generating a report, ensuring data validation, proper formatting, accurate population, verification, and secure storage of the final report.

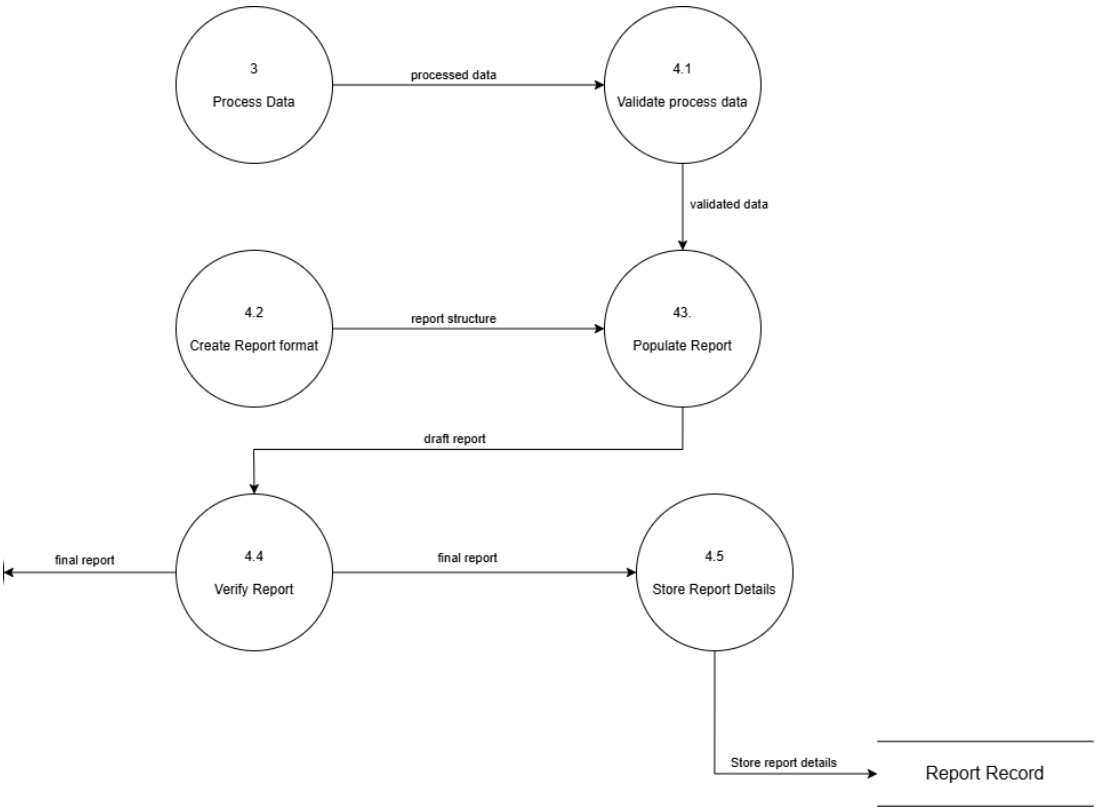
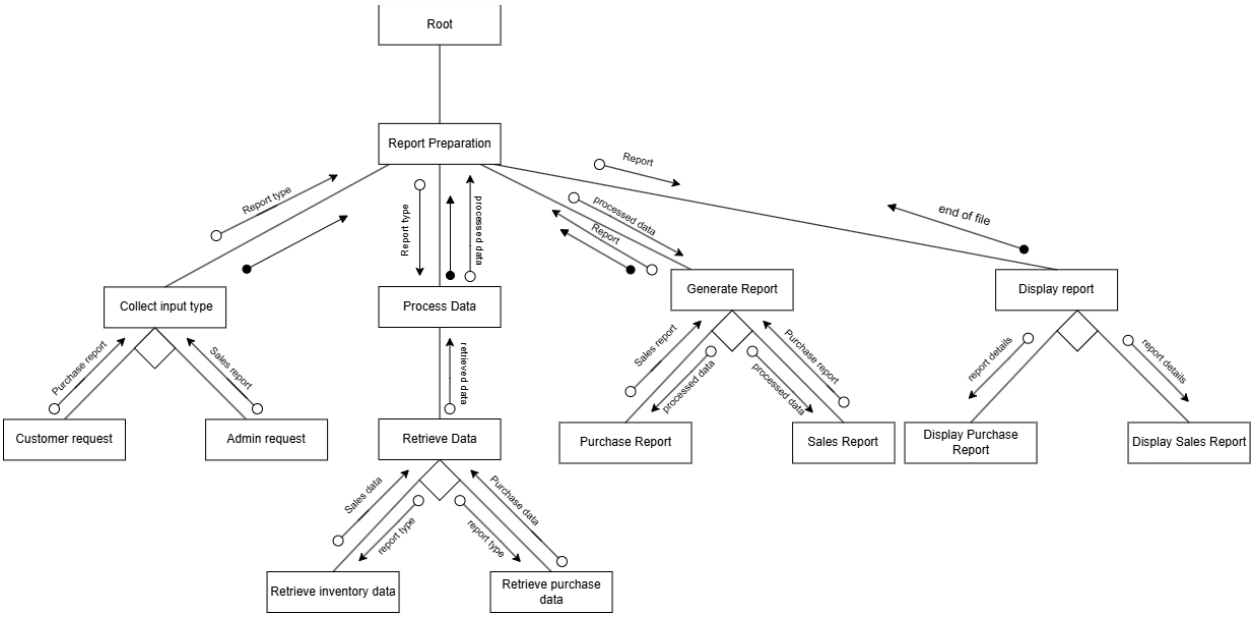


Figure 13: Level 2 DFD of Report Preparation

The diagram shows detailed interactions between processes, highlighting how data moves through the system from initial input to final report generation and display.

16.4 Structured Chart



16.5 Module Specification

16.5.1 Module Name: Report Preparation

16.5.2 Purpose

The Report Preparation module is responsible for collecting processed data, formatting it according to the requested report type (Sales or Purchase), and storing the finalized report. It delivers the reports to customers or administrators based on the request.

16.5.3 Pseudocode:

```
DO
    /* Step 1: Get user type */
    VAR userType = Database.getUserType()
    /* Step 2: Determine report type based on user */
    IF (userType == "Customer") THEN
        PRINT("Customer detected.")
```

```
    VAR reportType = "Purchase"
ELSE IF (userType == "Admin") THEN
    PRINT("Admin detected.")
    PRINT("Select Report Type:")
    PRINT("1. Sales")
    PRINT("2. Profit/Loss")
    VAR reportType = INPUT("Enter Report Type: (1 for Sales, 2 for Profit/Loss)")

    /* Map numeric input to report type */
    IF (reportType == "1") THEN
        reportType = "Sales"
    ELSE IF (reportType == "2") THEN
        reportType = "Profit/Loss"
    END IF
END IF

/* Step 3: Ask for date range */
PRINT("Please specify the date range for the report.")
VAR startDate = INPUT("Enter Start Date (YYYY-MM-DD):")
VAR endDate = INPUT("Enter End Date (YYYY-MM-DD):")

/* Step 4: Process report based on report type */
```

```
SWITCH (reportType)
    CASE "Sales":
        PRINT("Generating Sales Report...")

        /* Nested Case: Check if it's a Profit/Loss Report */
        PRINT("Do you want a detailed Profit and Loss Report?")
        PRINT("1. Yes")
        PRINT("2. No")
        VAR isProfitLoss = INPUT("Enter your choice (1 for Yes, 2 for No):")

        IF (isProfitLoss == "1") THEN
            PRINT("Generating Profit and Loss Report...")
            VAR dateRange = get.salesDate(startDate, endDate) /* Validate date range */
            VAR ProfitLossReport = GenerateReport("ProfitLossData", dateRange)
            PRINT(ProfitLossReport)
        ELSE
            PRINT("Generating standard Sales Report...")
            VAR dateRange = get.salesDate(startDate, endDate) /* Validate date range */
            VAR SalesReport = GenerateReport("SalesData", dateRange)
            PRINT(SalesReport)
        END IF
    BREAK
```

```
        CASE "Purchase":
            PRINT("Generating Purchase Report...")
            VAR dateRange = get.purchaseDate(startDate, endDate) /* Validate date range
*/
            VAR PurchaseReport = GenerateReport("PurchaseData", dateRange)
            PRINT(PurchaseReport)
        BREAK


        DEFAULT:
            PRINT("Invalid report type selected. Please try again.")
        BREAK
    END SWITCH
END DO
```

16.5.4 Input Parameters:

- userType (String): Type of the user (Customer or Admin).
- reportType (String): Type of report requested (Purchase, Sales, or Profit/Loss).
- startDate (Date): Start date for the report.
- endDate(Date): End date for the report.
- isProfitLoss (Boolean): Admin's choice to generate a detailed Profit/Loss report

16.5.5 Output Parameters:

- Generated Report (String): The completed report for delivery to the requester.

16.5.6 Local Variables:

userType (String): Stores the user type fetched from the database.

reportType (String): Stores the report type requested by the user.

SalesReport, PurchaseReport, ProfitLossReport (String): Temporarily holds the generated report.

16.5.7 Global Variables:

UserDatabase (Data Store): Maintains information on users (e.g., user type).

16.5.8 Calls:

Database.getUserType(): Fetches the type of user making the request.

get.salesDate(startDate, endDate)

get.purchaseDate(startDate, endDate)

16.5.9 Called By:

Sales Management, Admin

17 Aashish Khadka: Real-Time Stock Updates

London-met id: 23049144

College id: np05cp4a230004

17.1 Context Level Diagram

The Context Diagram gives an overall view of Real-time Stock Updates facility in the context of the IMS. It illustrates the system's interactions with two key external entities:

1. Admin: Being accountable for the compilation of stock variation and receiving stock status notification and threshold alert.

2. Buyer: Orders stock and gets information about the availability of the stock.

This diagram provides a conceptual illustration on how the Real-time Stock Updates system integrates with these entities and how data flows in and out of it, thus providing a good picture of the system’s interfaces and what is in and out of the scope of the project.

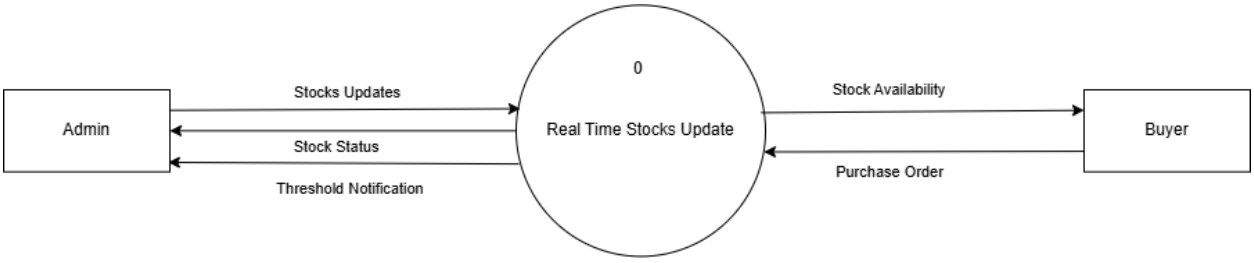


Figure 14: Level 0 DFD of Real time stocks update

17.2 Level 1 DFD

The Level 1 DFD breaks down the Inventory Management System into five main processes, showing how they interact with external entities and data stores.

Processes:

1. Process 1: Process Real time stock changes

- Gets stock updates from Admin
- Ensures the readiness of the data on stock information and processes the information.
- Fetches new stock data to the Stock Database

2. Process 2: Update Inventory Levels

- Incorporates purchase orders received from the Buyers
- Alters stocks as per the sales needed
- Causes appropriate changes to be made to the Stock Database

3. Process 3: Monitor Stock Levels

- Comparing current stock status to the status of the pre-inventory audits
- Sends threshold alerts of Admin
- Offers Buyers with Available Stock Status

4. Process 4: Synchronize Inventory

- It guarantees that data is consistent in the whole system.
- Controls the data exchange between Stock Database and Sync Database
- Ensures that data of different inventories is fully accurate in real time.

5. Process 5: Log Errors

- Records any discrepancies with the changes made in stock levels
- Saves each kind of error information into the Error Log database
- Makes every step more manageable and the system easier to regulate, maintain, and update.

Data Stores:

- Stock Database: Reference point for up to date status of existing stock.
- Sync Database: Ensures that the data corresponding to the inventory is in accord with the one in the system.
- Error Log: Contains logbooks of system failures and discrepancies

Data Flows:

- Admin Interactions:
  - Updates Process 1 on stock values
  - Processes receive threshold notifications from Process 3

- Buyer Interactions:
  - Encases purchase orders for procedure in Process 2
  - It also receives the stock availability information of certain products from Process 3

- Internal Data Flows:
  - Stock data moves from process to process and between databases
  - Error information sent to Error Log

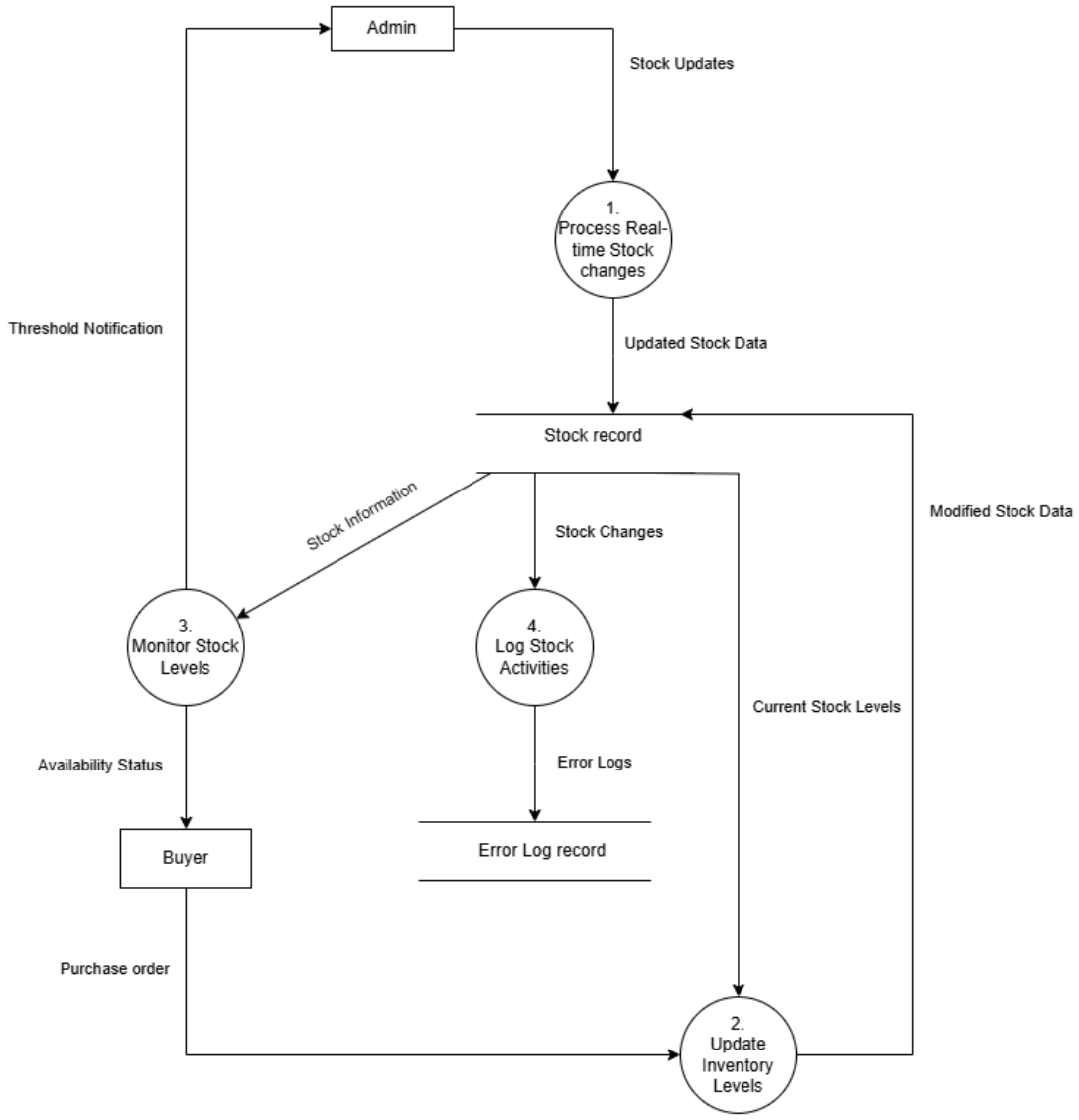


Figure 15: Level 1 DFD of Real time stock update

17.3 Level 2 DFD

This Level 2 DFD provides a detailed breakdown of the Real-time Stock Changes process, showing five interconnected sub-processes and their interactions with data stores:

Sub-processes:

1. Process 1.1: Validate Stock Input

- Input: Receives new stock data through the ‘Admin’ very Commodities.
- Function: Concerned with format and accuracy of the stock information entering its source system
- Output:

1. Forward validated data to Calculate Stock Changes
2. Transfers validation errors to Log Errors process
3. Issues failure data to Admin for further correction

2. Process 1.2: Calculate Stock Changes

- Input:

1. Obtain data from Process 1.1 validated
2. Pulled current quantities in stock from the Inventory record

- Function: Calculates required level corrections for stock
- Output:

3. Updates Available products record with stock received from an Inventory record
4. Forwarded to Log Errors the calculation of errors.
5. Produce change history related to inventory control

3. Process 1.3: Synchronize Inventory

- Input:

1. Accepts information of change log

2. Processes inventory updates
- Function: This guarantees that there is correct and unvarying direction of data associated with the inventory.
- Output: Sends sync data to active record Inventory

4. Process 1.4: Check Thresholds

- Input: Always gets new levels of clients from the Inventory record.
- Function: Compares the stocks with certain limits
- Output: Sends threshold notifications to Admin when they get to specific numerical values

5. Process 1.5: Log Errors

- Input:
1. Gets validation errors from Process 1.1
2. This receives calculation errors from Process 1.2
- Function: Manages data regarding error processes and formats
- Output: Saves error data in Error log record.

Data Stores:

1. Inventory record:
- Keeps up to date stock quantity
- Stores change logs
- Directs synchronization data.

2. Error log record:

- Stores all system errors
- Error history is kept to help resolve them

Key Data Flows:

1. Input Flows:

- Checkout to the new stock data submitted to you by Admin for validation of input data.
- Location Stock Position from Inventory record to Calculate Stock Changes

2. Process Flows:

- Exchanged data between processes
- Updates stock and change logs to the Inventory record
- Student errors to Log Errors process

3. Output Flows:

- Threshold alerts to Admin
- Error data to Error log record
- Generally, simultaneous with their inventory updates

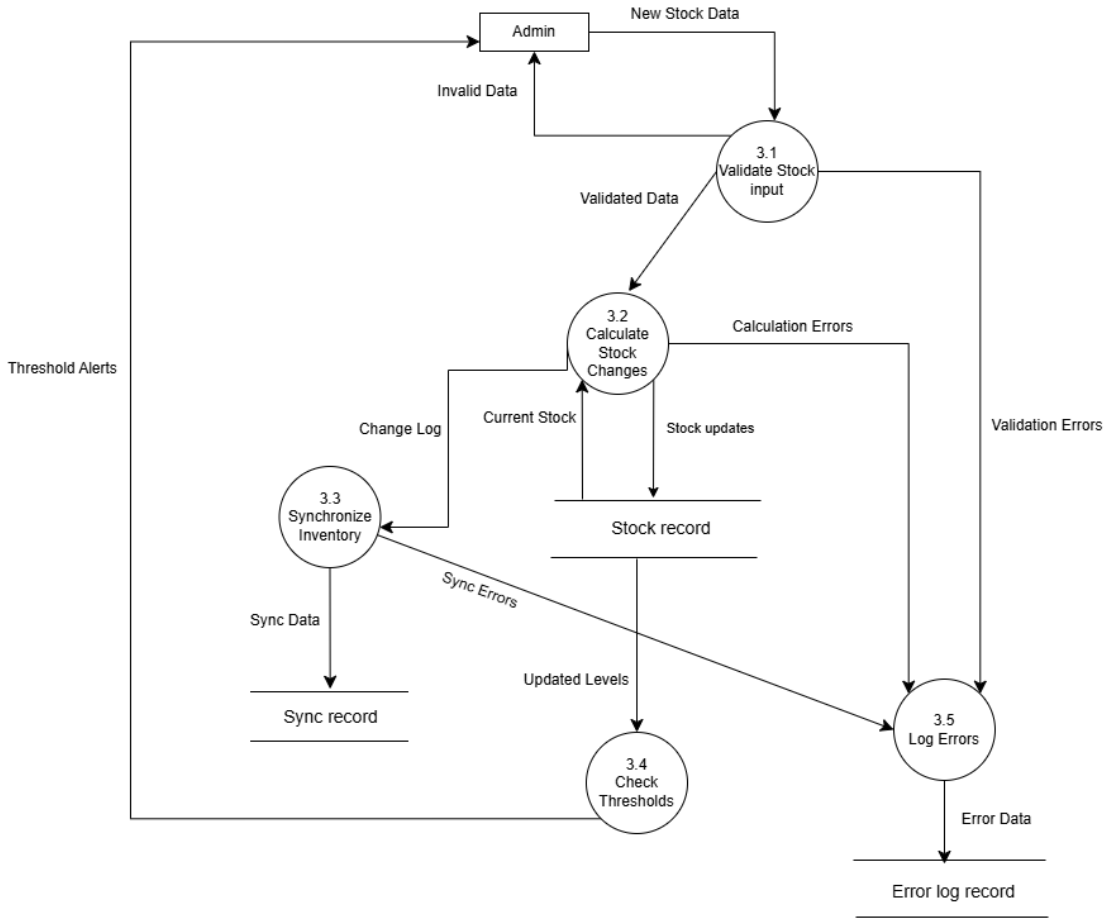
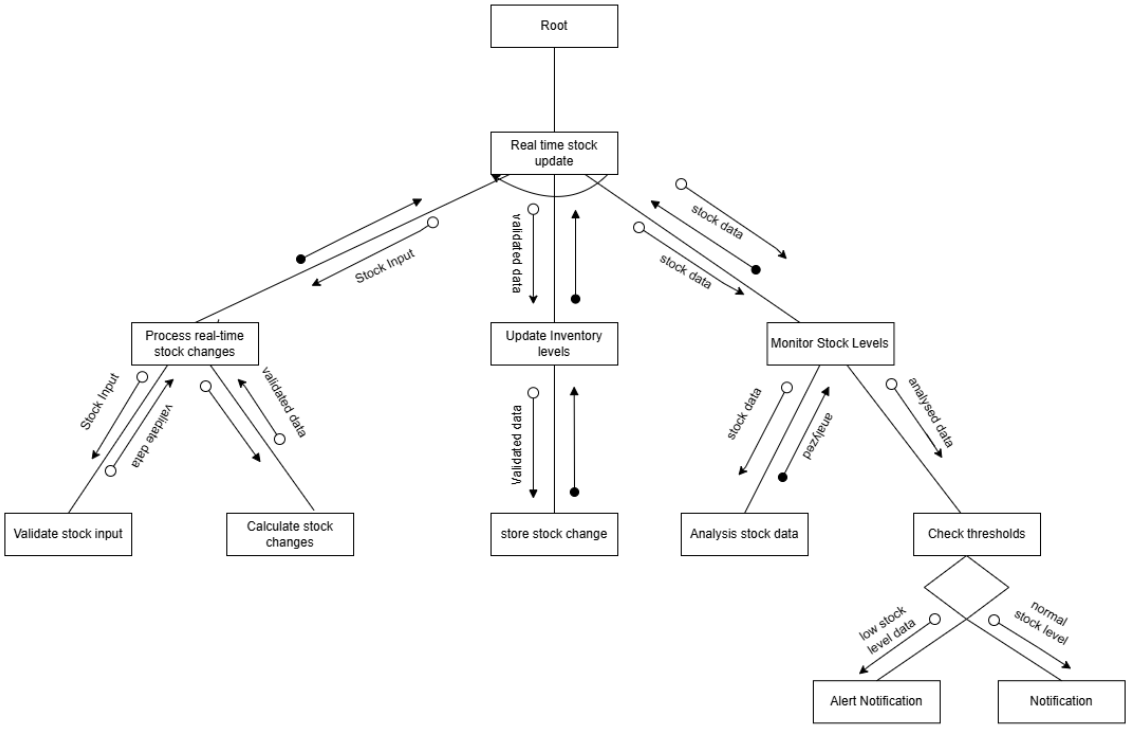


Figure 16: Level 2 DFD of Real time stock update

17.4 Structured Chart



The Structure chart above indicates how the automation of the Real-time Stock Update system is composed with emphasis to the Root node and the general flow of the system. The system branches into three main modules: Current stock fluctuation, Change in stock and, Stock tracking. Real time stock change deals with the input validation of stocks and the computations as well whereas Update inventory levels deals with the storage of stock changes. The Monitor Stock Levels module is then divided into an analytical level and a threshold check level; the latter further divides into a low stock Alert Notification level and a normal stock Notification level. Application connectivity is represented by dashed lines while end to end transaction is represented by solid line arrow where valid data stock input moves from one module to another. This gives clear data control in the hierarchy degradation and update till it issues the notification giving a complete stock control.

17.5 Module Specification

17.5.1 Module Name: Real Time Stock Updates

17.5.2 Purpose

The Real-Time Stock Updates module guarantees real-time tracking and notification of change in the stock of inventory. It offers immediate update as far as stock additions, deletions, and modifications are concerned, and lets the users know when the stock is low, or if the products have synchronization problems. This module aimed at the enhancement of inventory management and ensuring a single source of the truth.

17.5.3 Pseudocode

```
DO

   /* Step 1: Fetch initial stock data */

   VAR currentStockData = Database.Stock.getStockData()



   /* Step 2: Monitor for stock updates */

   WHILE (True) DO

      VAR newStockData = Database.Stock.getStockData()



      /* Step 3: Compare new data with current data */

      VAR changes = DetectChanges(currentStockData, newStockData)
```

```
        /* Step 4: Process changes if any */
        IF changes != NULL THEN
            FOR EACH change IN changes DO
                UpdateStock(change.itemID, change.newQuantity)
                CheckThreshold(change.itemID, change.newQuantity)
                TriggerSynchronization(change.itemID,change.newQuantity,
change.updateType)
            END FOR
        END IF

        /* Step 5: Update current stock data */
        currentStockData = newStockData

        WAIT_FOR(RealTimeUpdateEvent)
    END WHILE
END DO
```

17.5.4 Input Parameters:

- Stock Data (Object): Got from the main input by Database.Stock.getStockData(), and used for figuring out when changes have happened.

17.5.5 Output Parameters:

- changes (Array): Below is the list of manufactures' stock items with updates.
- alerts (Array): Alarm like low stock message or message for synchronization failure.

17.5.6 Local Variables:

- currentStockData (Object): Saves the data of the stock obtained from the database at the start of the program.
- newStockData (Object): Suspended as a temporary container of updated stock data for comparison.
- changes (Array): One of its functional requirements that holds details of change in stock levels detected by the system.

17.5.7 Global Variables:

- InventoryDatabase (Data Store): Source of information concerning stocks.
- UserDatabase (Data Store): These are user details used in generating alerts.
- SyncLog (Data Store): Records synchronization operations for monitoring and subsequently auditing.

17.5.8 Calls:

- Database.Stock.getStockData(): Recover data which is stored in the inventory of stocks.
- DetectChanges(currentStockData, newStockData): To find updates, compares stock data.

- UpdateStock(itemID, newQuantity): Responsible for maintain and update the stocks levels in database
- CheckThreshold(itemID, newQuantity): It detects when the quantity of stocks significantly drops with relation to the necessary amount.
- TriggerSynchronization(itemID, newQuantity, updateType): There is the confirmation that data Integration leads to data consistency across different systems.
- Wait_For(RealTimeUpdateEvent):Pauses the process until a real-time update event occurs.

17.5.9 Called By:

The entire pseudocode logic is initiated by an external process that triggers this infinite loop (DO ... WHILE(True)).

18 Diya Rai: Dispatch Order

London-met id: 23049237

College id: NP05CP4A230237

18.1 Context Level Diagram

The level 0 DFD Show how users interact with the Dispatch order management System to create, update, track order, cancel order, and generate reports. It highlights the flow of information between users, the system and its data stores at a high level between the Customer and the Admin.

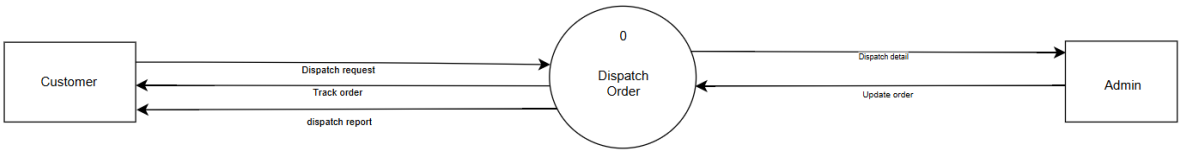


Figure 17: Context Level Diagram of Display Order

18.2 Level 1 DFD

The Level 1 DFD provides the user interacting with four main processes like creating/updating dispatch order, tracking order status, cancelling orders, and generating reports. These processes manage data flow between the user, order data store, inventory data store and report data store.

Here, The Level 1 DFD breaks down the Dispatch Order management system into four Main processes:

Process 1: Collect Dispatch Request

- It Receives Dispatch request From the customer and make ensure that the requested details are captured and stored for the further process.
- It then send the dispatch details to the next process.

Process 2 : Dispatch Cancellation

- It Handle the cancellation of requested orders from the customer .
- Then Verifies and processes for the cancellation.
- It Notifies the Admin of the cancellation of request from the customer.

Process 3 : Process Dispatch Request

- Processes the collection of dispatch request and Validates them for further process.
- It Updates the dispatched orders status and sends them to the Admin for further Action.
- Then Shared the Dispatch report process with for the next process.

Process 4 : Store Dispatch Orders

- This process Stores the dispatch orders for the furture references.
- It Then Updates the dispatched orders record in the system for tracking and reporting of the order.

Data Stores:

1. Dispatch Record
   - It Stores the finalized data and processed the dispatch order for furture reference.

Data Flows:

1. Customer Interactions:
   - It sends dispatch request to process 1.
   - It can be cancel orders by interacting with the process 2.

2. Admin Interactions:

   - Receives dispatch Orders request from process 3.

- Processes Cancellation requests from process 2.

3. Internal Data Flows:

- The Dispatched details flow between the processes.
- The processed Orders and their records are to updated the dispatch Record.

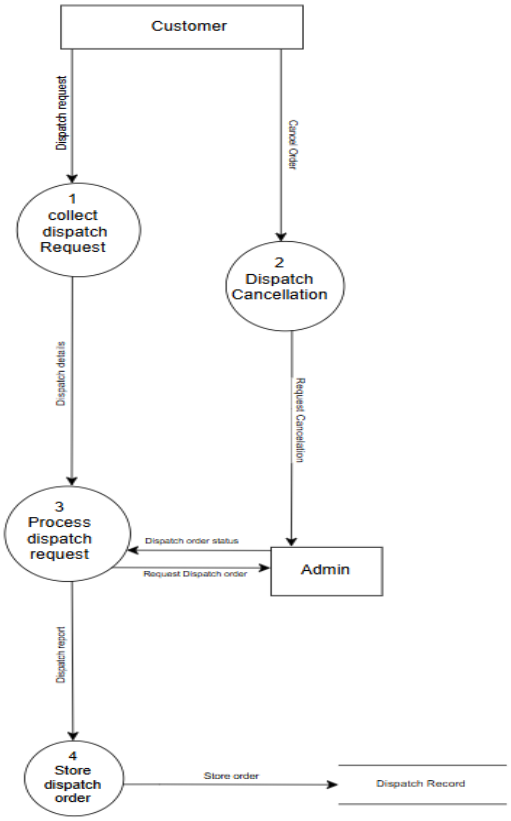


Figure 18: Level 1 DFD of Dispatch Order

18.3 Level 2 DFD

The level 2 DFD outlines the detailed steps of order processing with the user submits order details, which are validated. The system then checks inventory availability, creates

the order, and verifies the destination. If necessary, the inventory updates and the order is confirmed. It illustrates the flow of data between processes and the data stores in the System.

This Level 2 DFD Provides the detailed breakdown the Process 3 of “Process dispatch request” from the Level 1 DFD . It Shows 5 Sub Process :

1. Process 3.1 : Verify Request Details
    - Input : It Receives the dispatch details from the customer.
    - Function : It Ensure that the dispatched request from the customer is accuarate and complete.
    - Output: Sends Verified details of the customer for the next process , “To check Request Status”.

2. Process 3.2: Check Request Status
    - Input : It Receives The Verified details of the customer form process 3.1.
    - Function : Then Check the status of the dispatched request , Such as Whether it is approved , pending, or need to updates.
    - Output: It Send the status for check results to the “Process Order.”

3. Process 3.3 : Process Order
    - Inputs: Receives Status for the check result from the process 3.2.
    - Function : It processes the dispatch order based on the status , to ensuring that it is ready for the storage and futher action.
    - Output : Sends the order details to the “Update Dispatch Record.”

4. Process 3.4 : Update Dispatch Record
    - Input : Receives Order details from the process 3.3.
    - Function : Updates the Dispatch Record with the latest order for the information, and it ensure that the orders is accurately tracked.

- Output: It provides the updated records for the retrival or the confirmation of the order.

5. Process 3.5 : Confirm Dispatch
   - Inputs : It Updated records from the process 3.4.
     It confirm the dispatch from the customer.
   - Function : It confirm the dispatch order with the customers based on the records Updates.
   - Output: Finally , Finalizes the dispatch order and closes the process.

Data Stores:

1. Dispatch Record:
   - Stores the details of updated and confirmed it to the dispatch order for the retrival of future references.

Data Flows:

1. Customer Interactions:
   - It Send dispatch detaileds to the process 3.1.
   - Confirms the dispatch order with the process 3.5 for further details .

2. Internal Data Flows:
   - Verified The details flow of “Check Request Stataus.”
   - Check Status result flow of the “Process Order.”
   - Updated the records Which are Stored in the “Dispatch Record” and then retrives for the confirmation.
3. OutPut Data Flow :
   - It Finalize the final dispatch is then confirmed and shared it with the customer.

This Level 2 DFD, Provides the detailed breakdown of the dispatch order process.

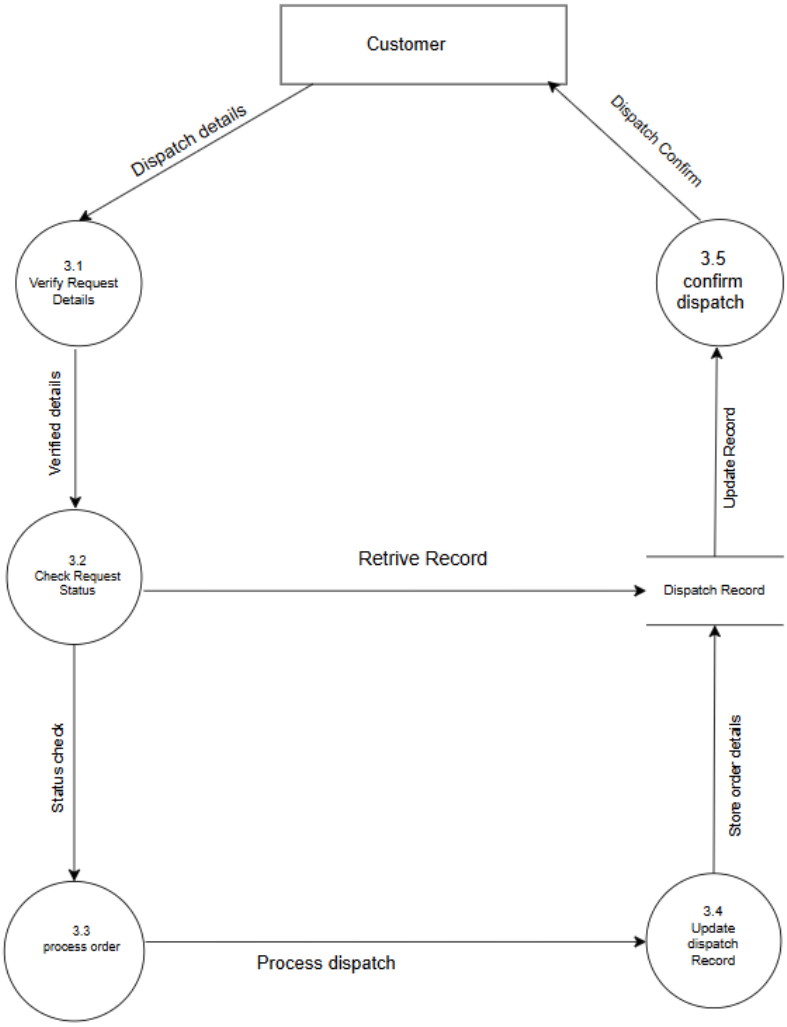
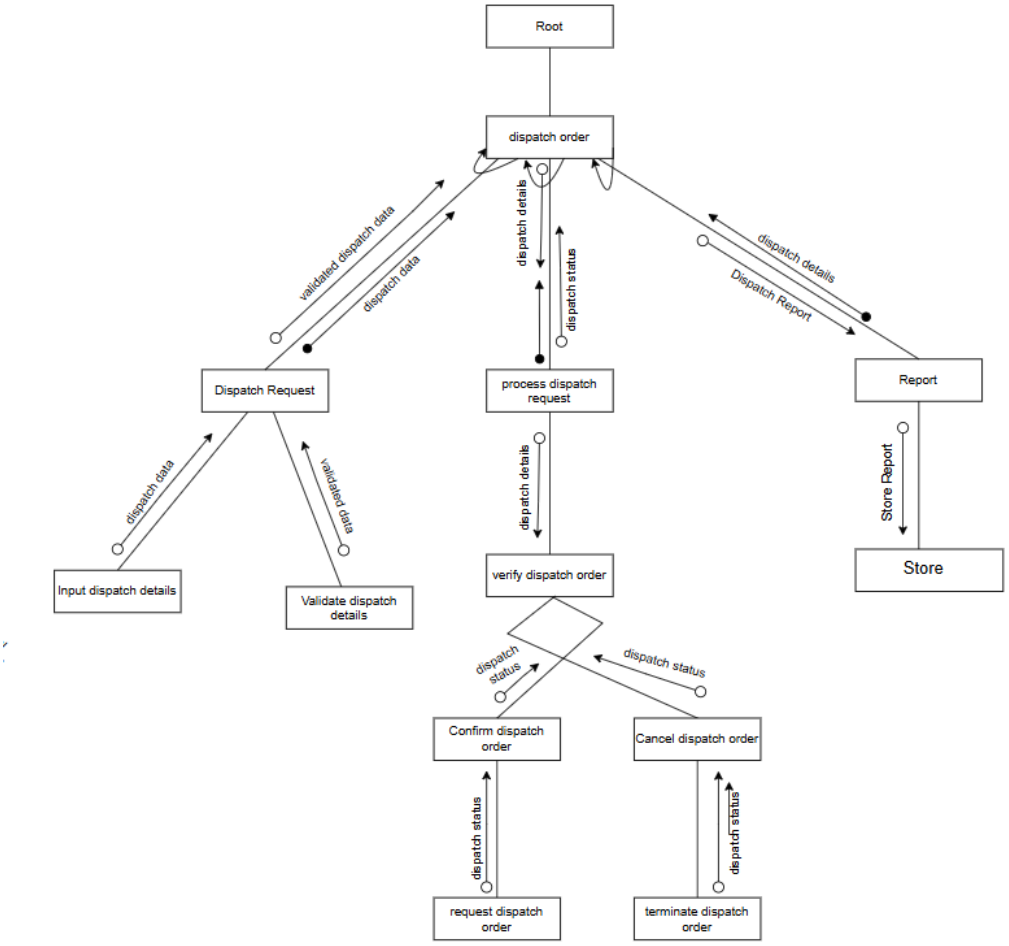


Figure 19: Level 2 DFD of dispatch order

18.4 Structured Chart



This Structure Chart Show the amount of detail and data flow of a dispatch order system from initial process of dispating an order to the Report finalization. This involves entering the dispatch details and checking whether the details entered are valid or not, and it involved with the dispatching of the request and checking the status of the order.

In this way, the desired result for dispatch to confirm or cancels the order on the result that produced by the system. Then the dispatch details are then sent to general report that will save the details in the system. Then the dispatch details are sent to the generate report which will saved as a record. All these steps contribute to make that the data is checked on their way for ensuring to the last stage of the dispatch order data.

18.5 Module Specification

18.5.1 Module Name: Dispatch order

18.5.2 Purpose:

A dispatch order is a document that tells the team what to send, where to send it , and how to handle it. it helps ensure the right items get delivered to the right place on time, keeping everything organized and on track.

18.5.3 Pseudo code:

```
DO

  /* Step 1: Get dispatch order details */
VAR order Details = GetDispatchOrderInfo()

/*Step2: Validate dispatch request */

IF (ValidateDispachRequest(orderDetails))
THEN
    VAR dispatch Type =INPUT (“Select Dispatch Type: 1. Standard 2. Express”)
 PRINT (“Invalid Dispatch Order Details”)
  EXIT

END IF
/* Step 3: Process dispatch based on type */

SWITCH (dispatch Type)
  CASE “Standard”:
    VAR standard status= ProcessStandardDispatch(orderDetails)
```

```
        IF (standardStatus == “Ready”) THEN UpdateDispatchStatus(“Preparing”)
         Assign Resources (“Standard Dispatch”)
        END IF
BREAK

CASE “Express”:
    VAR expressStatus= processExpressDispatch(orderDetails)
IF (expressStatus == “Ready”) THEN
      UpdateDispatchStatus(“Prioritized”)
       AssignResources(“ExpressDispatch”)
   END IF
BREAK

END SWITCH
/* step 4: confirm and initiate dispatch */
IF (dispatch Status == “preparing” OR dispatch Status == “Prioritized”) THEN

       VAR confirmation = INPUT (“Confirm Dispatch? (Y/N)”)
        IF (confirmation == “Y”) THEN

              InitiateDispatch(orderDetails)
   UpdateDispatchStatus(“In Transit”)
    GenerateDispatchRecort(dispatchType)
```

```
        PRINT (“Dispatch Initiated Successfully”)
      ELSE

    cancelDispatch(orderDetails)
   UpdateDispatchStatus(“Cancelled”)
     PRINT (“Dispatch Cancelled”)
        END IF


ELSE
            PRINT (“Unable to Initiate Dispatch”)

    UpdateDispatchStatus(“Failed”)
            PRINT (“Dispatch Cancelled”)
        END IF



      ELSE
          PRINT (“Unable to initiate Dispatch”)
   UpdateDispatchStatus (“Failed”)
     END IF
       /* Step 5: Store dispatch details */
StoreDispatchDetails (OrderDetails, dispatchStatus)
```

```
/* Step 6 : Generate and Store Dispatch Report */
PROCEDURE GenerateDispatchReport(dispatchType)
        VAR reportData= CreateDispatchReport(dispatchType,OrderDetails)
        PRINT ("Dispatch Report Generated:")
  PRINT (reportData)
          /*      Store Report    */
          StoreDispatchReport(reportData)
           PRINT(" Dispatch Report Stored Successfully ")
END PROCEDURE

END DO
```

18.5.4Input parameter:

- OrderDetails
- DispatchType
- confirmation

18.5.5 Output parameter:

- Dispatch Status.

18.5.6 Local Variables:

- OrderDetails
- DispatchType
- Confirmation

- Standard status
- dispatch order
- ExpressStatus

18.5.7 Global Variables:

- DispatchStatus
- OrderDetails

18.5.8 Calls:

- GetDispatchOrderInfo
- UpdatedispatchStatus
- InitiateDispatch.

18.5.9 Called By:

- Main.

19 Sanjita Chaudhary: Payment

London-met id: 23049984

College id: NP05CP4A230161

19.1 Context Level Diagram

-

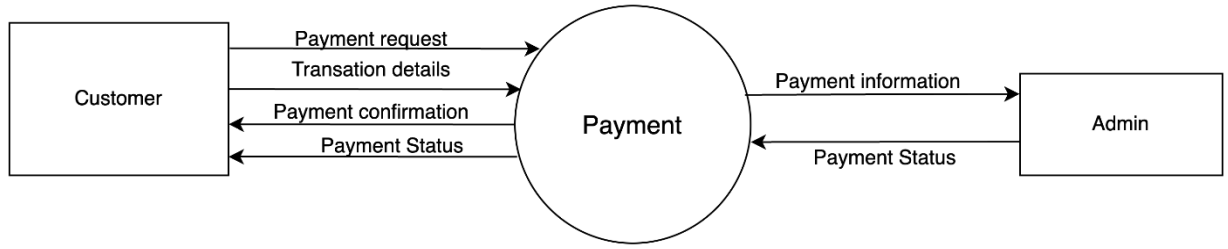


Figure 20: Context Diagram of Payment

19.2 Level 1 DFD

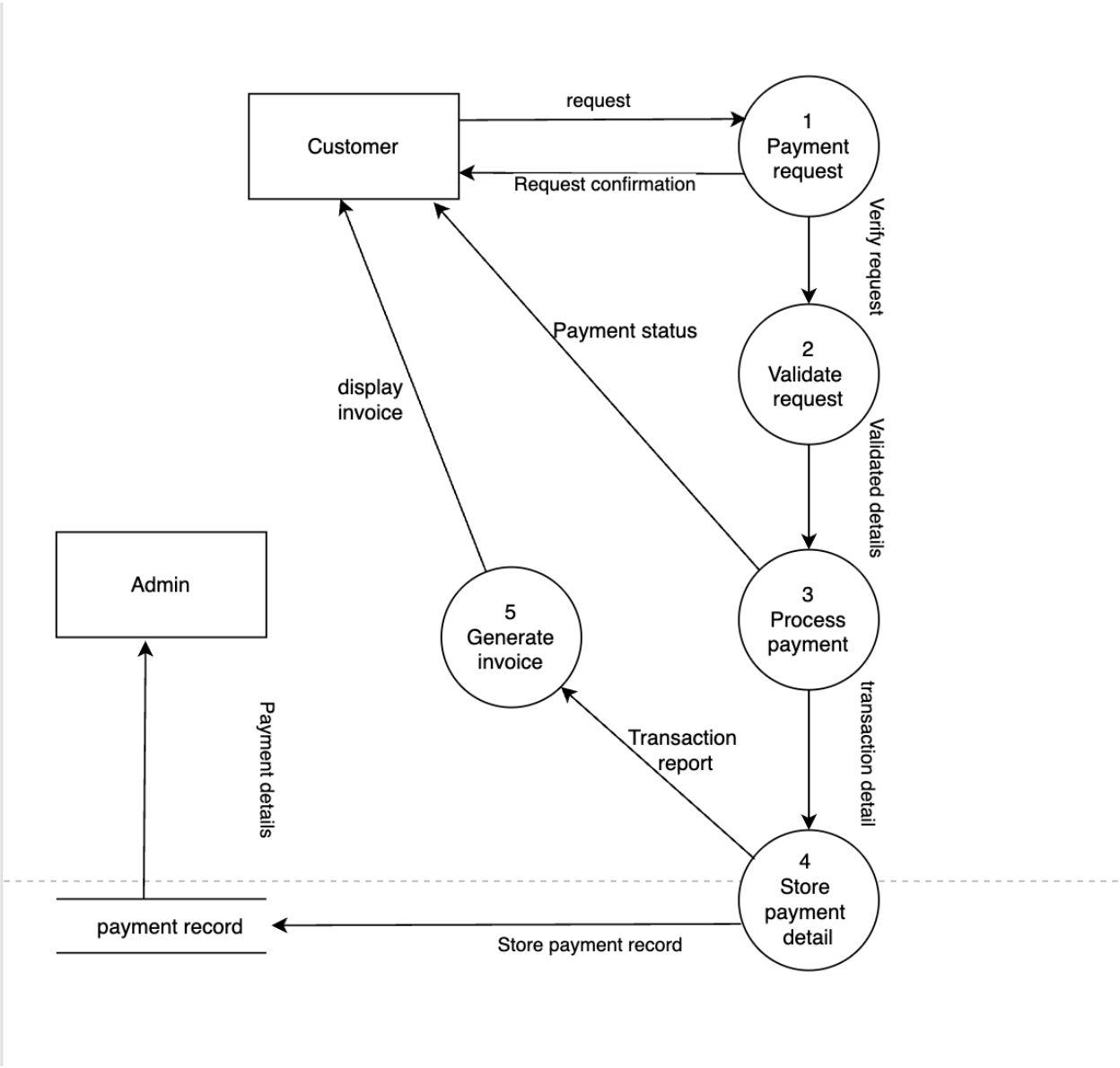
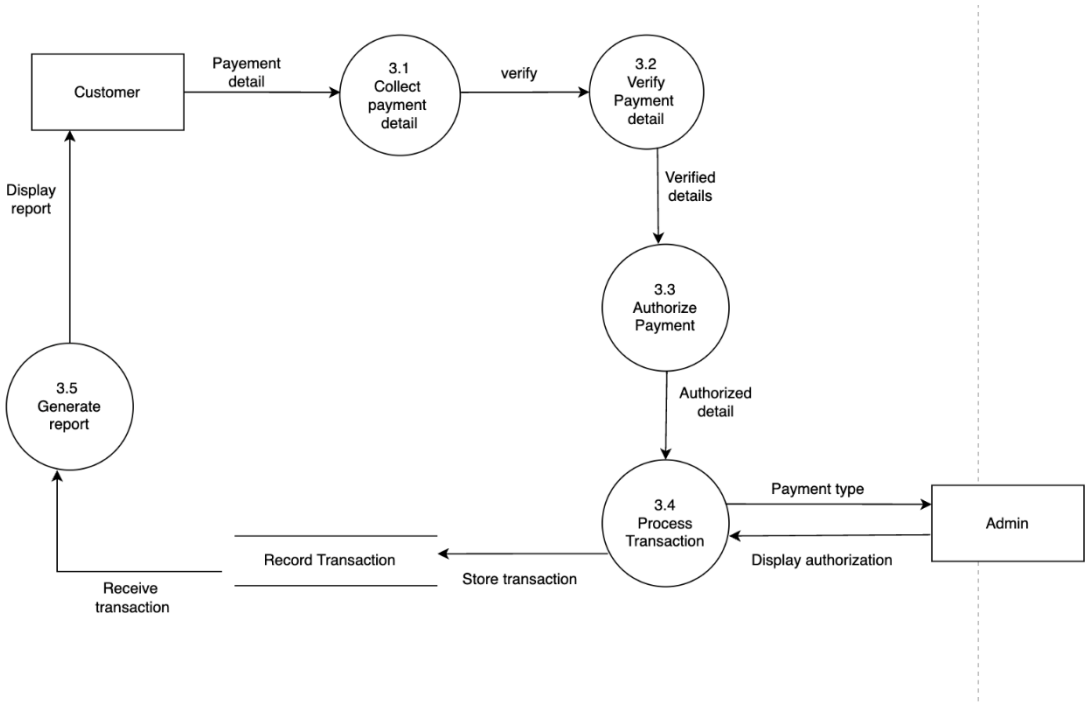
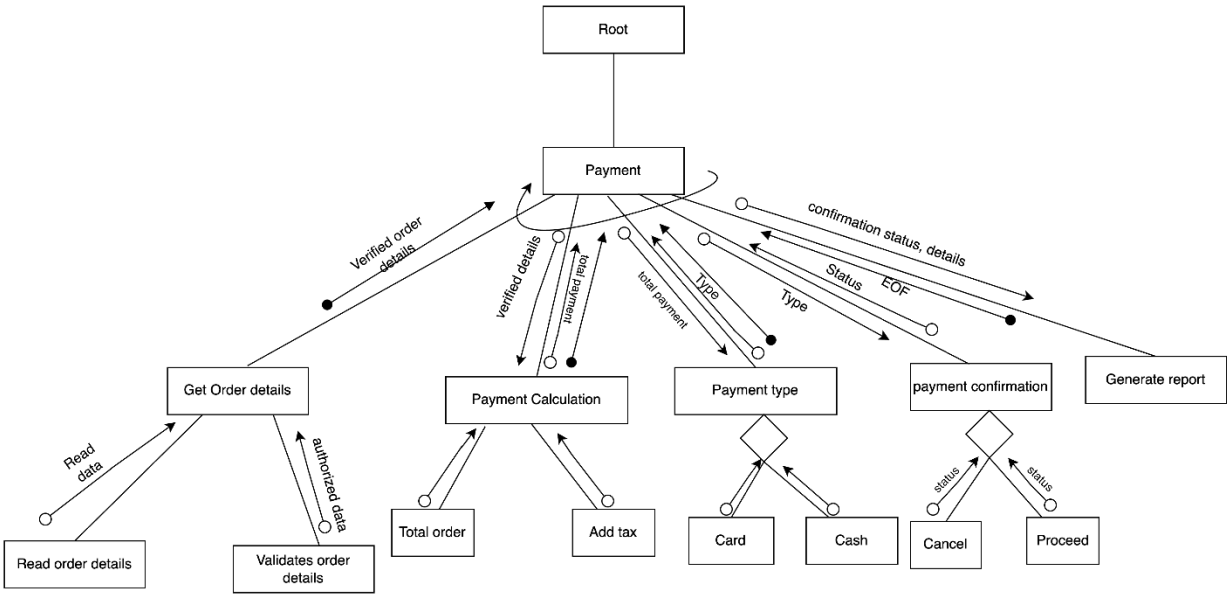


Figure 21: Level 1 DFD of payment

19.3 Level 2 DFD of process payment



19.4 Structured Chart



-This diagram shows a hierarchical payment processing system. It initializes with a root

node that leads to the payment modules which manages and control the flow of payment. In the payment modules, it starts by handling with Get Order details, where the information of order is retrieved through read order details and it validates order details. Verified order details are moved to the payment calculation, it calculates the total order and tax of the specific order then sends total payment of that order to the payment type. In payment type, user can choose between the option i.e. cash or card, after that payment type moved to the confirmation where the customer can also have two option ”Cancel or Processed”. If customer chooses the cancellation then the placed order will be cancelled immediately Or chooses processed then the status will be send for report generation and then customer receives their pay

19.5 Module Specification

19.5.1 Module name: Payment

19.5.2 Purpose:

The payment module facilitates secure and efficient financial transaction by collecting, validating and processing payment details. It supports multiple payment methods such as debit and credit to ensure authorization, generates invoice and securely stores transaction records in a database for further more reports.

19.5.3 Pseudocode

```
DO
/*get payment details*/
 VAR orderDetails = getOrderDetails()
   IF orderDetails.isValid() THEN

        //Calculate total payment amount
        VAR totalOrder = paymentCalculation(orderDetails)
        VAR tax = paymentCalculation(orderDetails)
        VAR totalAmount = totalOrder + tax
        RETURN totalAmount

// Select payment type
        VAR paymentType = INPUT(“Select payment type: 1. Card, 2. Cash”)
        IF paymentType.isValid THEN()
        SWITCH (paymentType)
        CASE “Card”:
        VAR cardStatus=processCardPayment(totalAmount)
        IF (cardStatus == Success) THEN
        PRINT(“”Total amount paid by card)
```

```
        END IF
        BREAK

        CASE “Cash”:
        VAR cashStatus = processCashPayment(totalAmount)
        IF(cashStatus == success) THEN
        PRINT(“”Total amount paid by cash)
        END IF

BREAK

        END SWITCH

        //Confirm payment status
        VAR paymentConfirmation = processConfirmation(totalAmount, paymentType)
        IF(paymentConfirmation == proceed) THEN
        PRINT(“Payment confirmed ”)
        ELSE
        PRINT(“Payment cancelled”)


        END IF

        VAR Report = generateReport(orderDetails, totalAmount, paymentStatus )
        Report.addOrderDetails(orderDetails)
        Report.addTotalAmount(totalAmount)
        Report.addPaymentStatus(paymentStatus)

RETURN Report

  ELSE
PRINT(“Order details are invalid to proceed ”)

  END IF
END DO
```

19.5.4 Input parameter:

Payment Type

19.5.5 Output parameter:

TotalAmount, Report

19.5.6 Local Variables:

- totalOrder
- tax
- totalAmount
- paymentType
- cardStatus
- cashStatus
- paymentConfirmation
- Report

19.5.7 Global Variables:

- getOrderDetails()
- paymentCalculation()
- processCardPayment()
- processCashPayment()
- processConfirmation()
- generateReport()

19.5.8 Calls:

- getOrderDetails()
- orderDetails.isValid()
- paymentCalculation(orderDetails)
- paymentType.isValid()
- processCardPayment(totalAmount)
- processCashPayment(totalAmount)
- processConfirmation(totalAmount, paymentType)
- generateReport(orderDetails, totalAmount, paymentStatus)
- Report.addOrderDetails(orderDetails)
- Report.addTotalAmount(totalAmount)
- Report.addPaymentStatus(paymentStatus)

19.5.9 Called By: Main

20 Conclusion

The suggested Inventory Management System (IMS) is designed to build up operational efficiency, accuracy and scalability by combining main characteristics for warehouse and inventory management.

Initially, the User Mangement System makes sure for secure access through role-based permissions, safe logins and protecting sensitive data, authorizing controlled user access. The purchase order module simplify ownership by enabling easy creation, modification, approvals and cancellation of orders by improving accountability and responsiveness.

Similarly, the Real-Time stock update system maintains correct inventory counts through automatic correspondence and alerts when stock reaches its critical levels by enabling dynamic management and avoiding shortage or overstocking. The Dispatch Order System upgrades planning by simplifying order creation, tracking deliveries and generating detailed order detailed reports for better prestige and communication.

Additionally, the Reporting and Analytics Module provides adjustable and pleasing reports by offering clear data-driven awareness to help track its performance and decision-making. The Payment System guarantee secure processing, refund handling, invoice generation and see-through payment tracking, encouraging trust and reliability.

More over, the IMS system was built on Cloud-based infrastructure with a workable design and delivers high performance, scalability and data security. It automatically processes and decreases errors by improving decision-making and making it reliable and future-ready solutions for well-run warehouse management and sustainable growth.

21 References

Asif, M. (2024, November) *(PDF) annotation of software requirements specification (SRS), extractions of nonfunctional requirements, and measurement of their tradeoff*. Available at: https://www.researchgate.net/publication/331537700_Annotation_of_Software_Require

ments_Specification_SRS_Extractions_of_Nonfunctional_Requirements_and_Measurement_of_Their_Tradeoff (Accessed: 13 December 2024).

Staff, M.A.M.L. (2024) *What are functional requirements?*, *Requirements.com*. Available at: https://requirements.com/Content/What-is/what-are-functional-requirements (Accessed: 15 December 2024).

Sommerville, I., 2011. *Software Engineering*. 9th ed. Pearson Education.