Practical-1

Aim: Project Definition and objective of the specified module and Perform Requirement Engineering Process.

Project Definition

Project Title: Retail Store Management System

Objective:

To develop a web-based retail store management system that automates store operations, including inventory management, sales transactions, customer relationship management, and order return management. The system will provide a user-friendly interface for store managers, employees and customers.

Specific Objectives

1. **Admin (Product Management):**
    - Insert, update and delete product details such as:
        - Product name, SKU (Stock Keeping Unit), category, price, and quantity.
        - Supplier information (name, contact details).
        - Reorder level (for low-stock alerts).
    - Manage employee accounts and permissions.
    - Configure system settings, including tax rates, discounts and low-stock thresholds.
2. **Inventory Management:**
    - Add, edit and delete product details.
    - Track stock levels and update quantities in real time.
    - Search for products by name, category or SKU.
    - Generate alerts for low stock levels and expiry of products.
    - **Generate Low-Stock Reports:**

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- Automatically generate a list of low-stock items for reordering
- Include item details like SKU, name, current stock, and reorder level.

3. **Sales Management:**
   - Process sales transactions, including adding products to the cart, generating bills and accepting payments.
   - Maintain a record of daily sales and transaction history.
   - Track discounts, taxes and payment methods.
4. **Customer Management:**
   - Add, edit and delete customer details.
   - Track purchase history and preferences.
   - Implement loyalty programs to reward frequent customers.
5. **Order Return Management:**
   - **For Customers:**
     - Allow customers to initiate product returns through an online portal.
     - View return status updates.
   - **For Store Owners (Admin):**
     - Process and approve return requests.
     - Update inventory levels for returned products.
     - Generate reports to track return trends and reasons.
6. **Reports and Analytics:**
   - Generate real-time sales and inventory reports.
   - Analyze sales trends, profit/loss, and return patterns.
   - Generate downloadable low-stock reports for reordering.
7. **User Interface:**
   - Provide role-based dashboards:
     - **Admin Portal** for store owners and managers to manage the system.
     - **Employee Dashboard** for inventory and sales handling.
     - **Customer Portal** for purchase and return requests.
   - Implement secure login, password protection and role-based access control.

Requirement Engineering Process

1. **Requirement Gathering**
   - **Interviews:**

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- Ask store owners about the need for an **Admin Portal** mange products, employees and system settings.
- Example: "How often do you update product details like prices or quantities?"

- **Questionnaires:**
  - Collect input form store managers about required admin functionalities (e.g., low-stock thresholds, adding new suppliers).
- **Observation:**
  - Observe manual methods for product insertion, updates, and data maintenance.

2. **Requirement Analysis**
   - **Functional Requirements:**
     - **Admin Portal:**
       - Add new product details (name, SKU, price, category, quantity and supplier details).
       - Edit or delete product information.
       - Manage reorder levels for low-stock alerts.
       - Add employee accounts with role-based access (Admin, Manager, Employee).
       - Configure system settings (tax, discounts rules, etc.).
   - **Non-Functional Requirements:**
     - **Performance:** Quick product insertion and update operations (<3 seconds).
     - **Usability:** Intuitive Admin Portal design for non-technical users.
     - **Security:** Admin access protected with secure login and role-based restrictions.
   - **User Interface Requirements:**
     - **Admin Portal:**
       - Forms to add/edit product details.
       - Dashboard to view product inventory summaries and low-stock alerts.
       - User management section to assign roles.
3. **Requirement Specification**

   Create and updated **Software Requirements Specification (SRS)** document, including:
   - **Introduction:** Purpose and scope of the Retail Store Management System.
   - **Overall Description:** Key modules, including the **Admin Portal** for product and employee management.
   - **Specific Requirements:**
     - Product insertion, updates, and deletions.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

    - Role-based permissions (Admin, Employee, Customer).
    - Low-stock threshold configuration.
  - **External Interface Requirements:** Integration with barcode scanners and supplier databases.
4. **Requirement Validation**
  - **Stakeholder Reviews:** Validate the Admin Portal features with store owners.
  - **Prototyping:**
    - Create a mockup of the Admin Portal showcasing product insertion and inventory management screens.
  - **Testing Scenarios:**
    - Test product addition, edits and deletions.
    - Verify low-stock alerts and stock-level updates.
5. **Requirement Management**
  - Track any changes to admin functionalities and product management workflows.
  - Use tools like **JIRA** or **Git** to manage changes.

Outcome:

The **Retail Store Management** System will include a comprehensive **Admin Portal** that enables store owners to:

1. Manage product data efficiently (insert, update and delete).
2. Configure low-stock thresholds and receive timely notifications.
3. Manage employee roles and permissions.

This addition ensures better control over inventory, streamlined operations and improved productivity.

By performing all the above steps, the **Retail Store Management System** will automate key store operations, optimize inventory management and improve customer service, resulting in greater efficiency and profitability for retail businesses.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Practical - 2

Aim: To Identify Suitable design and Implementation model from the different software engineering models for **Retail Store Management System.**

When selecting a software engineering model for the **Retail Store Management System**, it's essential the system's complexity, requirements, timeline, and Scalability. Below are key factors for the decision:

1. **The project involves well-defined features** (inventory management, sales, order returns, admin portal, etc.) but can expand overtime.
2. **The requirements are fairly clear** , but minor changes may arise based on stakeholder feedback.
3. **The system must be developed iteratively** to allow incremental delivery and frequent testing for quality assurance.

Based on these considerations, the Incremental Development Model and Agile Model are the most suitable.

➢ Incremental Development Model

The Incremental Model focuses on developing the system, manageable parts (increments). Each increment adds specific features, building upon previously implemented functionality.

Why Incremental Model?

a. Suitable for systems where requirements are clear but development needs to be phased.
b. Ensures **early delivery** of core functionalities (e.g., Inventory Management) before moving to advanced features (e.g., Low-Stock Reports, Order Return Management).
c. Allows feedback from stakeholders after each increment to improve the system.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Implementation Strategy:

The system can be broken down into the following increments:

Increment	Features Implemented
Increment 1	Admin portal: Product insertion, updates, and deletions. Inventory tracking (stock levels).
Increment 2	Sales Management: Cart handling, billing and Payments.
Increment 3	Customer Management: Add/edit customer details, track purchase history.
Increment 4	Order Return Management: Customer return request and approval workflows.
Increment 5	Reports and Analytics: Generate sales, inventory, and low-stock reports.

➢ Advantages:

i. Prioritizes delivery of core features, ensuring stakeholders can start using the system early.
ii. Easier to manage risk as testing is performed after each increment.
iii. Flexibility to adapt and add features in later increment based on feedback.

➢ Agile Development Model

The **Agile Model** emphasizes flexibility, iterative development, and collaboration with stakeholder. It involves frequent releases of working software and continuous feedback to refine the product.

Why Agile Model?

a. Ideal for systems where minor requirements might change during development.
b. Supports rapid iteration and delivery of working modules (e.g., Inventory and Admin Portal first).
c. Encourages stakeholder involvement to ensure the system aligns with business needs.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- **Implementation Strategy (Scrum Approach):**
  i) **Sprint Planning :** Break down features into smaller tasks to be implemented in short cycles (2-4 weeks).
  ii) **Sprints:** Each sprint delivers a working module:
    (1) **Sprint 1:** Admin Portal for product management and basic inventory tracking.
    (2) **Sprint 2:** Sales Management with transaction processing.
    (3) **Sprint 3:** Customer Management and loyalty program.
    (4) **Sprint 4:** Order Return Management workflows for customers and store owners.
    (5) **Sprint 5:** Reports, analytics, and low-stock alerts.
  iii) **Daily Standups:** Regular team meetings to track progress and resolve challenges.
  iv) **Sprint Review:** Demonstrate the completed work to stakeholders for feedback.
  v) **Sprint Retrospective:** Discuss what worked well and identify areas for improvement.
- **Advantages:**
  - Allows for continuous improvement based on stakeholder feedback.
  - Delivers a usable product early, with new features added incrementally.
  - Handles changes in requirements effectively without affecting overall progress.

Comparison of Models

Aspect	Incremental Model	Agile Model
Flexibility	Medium: Features are pre-planned but can adapt incrementally.	High: Changes can be incorporated during any sprint.
Stakeholder Involvement	Moderate: Feedback after each increment.	High: Continuous feedback throughout development.
Delivery of Features	Gradual delivery with clear increments.	Frequent delivery of working software.
Risk Management	Medium: Risk management incrementally.	High: Frequent testing reduces risk.
Project Complexity	Suitable for medium-complexity systems.	Suitable for medium to high complexity systems.

- **Conclusion: Final Recommendation**

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- **Primary Choice: Agile Development Model**
  - It aligns with the need for flexibility, frequent testing, and incremental feature delivery.
  - Agile ensures active involvement of stakeholders (store owners, employees) to refine requirements, particularly for features like **Low-Stock Alerts** and **Order Return Management.**
- **Secondary Choice: Incremental Development Model**
  - If the project scope is well-defined with minimal changes expected, the Incremental Model is an effective alternative.

The Agile Model is particularly recommended because of its adaptability and efficiency in delivering a robust and scalable **Retail Store Management System** that meets stakeholder expectations.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Practical – 3

Aim : Prepare Software Requirement Specifications(SRS) for the Retail Store Management System.

1. Introduction

1.1 Purpose

The purpose of this document is to define the functional, non-functional, and technical requirements for the development of a **Retail Store Management System**. The system will serve as a centralized platform to automate retail store operations, including inventory management, sales transactions, customer and return management, and reporting.

1.2 Scope

The **Retail Store Management System** is a web-based application designed to streamline operations for store owners, employees, and customers. It will provide:

- An **Admin Portal** to manage product data, system configurations, and employee accounts.
- Real-time inventory tracking with low-stock alerts and reports.
- Transaction processing for sales and order returns.
- A reporting module for sales, inventory, and returns analysis.

1.3 Definitions, Acronyms, and Abbreviations

- **Admin Portal:** A feature that allows store owners to manage inventory, employee roles, and system configurations.
- **SKU:** Stock Keeping Unit, a unique identifier for products.
- **Low-Stock Report:** A report listing items below their defined reorder threshold.

1.4 References

- Agile Development Model Documentation.
- Retail Management System Design Patterns Guide.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

2. Overall Description

2.1 Product Perspective

The system will act as an integrated solution replacing manual retail processes. It will enable users to manage inventory, sales, and customer interactions effectively. The platform will be accessible via desktop and mobile devices.

2.2 Product Functions

1. **Admin Portal:**
   - Add, update, and delete products.
   - Manage employee accounts and permissions.
   - Configure system settings, including taxes, discounts, and reorder levels.
2. **Inventory Management:**
   - Track stock levels in real time.
   - Generate low-stock alerts and reports for reorder planning.
3. **Sales Management:**
   - Process transactions, calculate taxes and discounts, and generate invoices.
   - Track daily sales and transaction history.
4. **Customer Management:**
   - Maintain customer records and purchase history.
   - Implement a loyalty program to reward frequent buyers.
5. **Order Return Management:**
   - **Customer Portal:** Allow customers to request refunds or replacements.
   - **Admin Portal:** Process and approve return requests, update inventory accordingly.
6. **Reports and Analytics:**
   - Generate and export reports for sales, inventory, and order returns.
   - Provide actionable insights to improve business decisions.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

2.3 User Characteristics

- **Store Owners/Admins:** Manage inventory, employees, and system settings.
- **Employees:** Handle sales and order returns.
- **Customers:** View product details, make purchases, and initiate return requests.

2.4 Constraints

- The system must comply with data privacy regulations (e.g., GDPR).
- Performance: The system should handle up to 10,000 transactions per day without performance degradation.

2.5 Assumptions and Dependencies

- Users will have access to stable internet connections.
- Integration with third-party payment gateways will be supported.

3. Specific Requirements

3.1 Functional Requirements

1. **Admin Portal:**
    - Ability to add, update, and delete products with details (e.g., SKU, price, category, supplier info).
    - Role-based access control for employees and admins.
    - Configuration options for tax rates, discounts, and low-stock thresholds.
2. **Inventory Management:**
    - Real-time stock level updates.
    - Automatic generation of low-stock alerts and downloadable reports.
3. **Sales Management:**
    - Process sales transactions with options for multiple payment methods.
    - Generate itemized bills with taxes and discounts applied.
4. **Customer Management:**

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- Maintain a database of customer information and purchase history.
- Track loyalty points and notify customers of rewards.

5. **Order Return Management:**
   - Customers can request refunds, replacements, or store credits.
   - Store owners can approve or reject requests and update inventory accordingly.
6. **Reports and Analytics:**
   - Generate daily, weekly, and monthly reports for inventory and sales.
   - Export reports in PDF and Excel formats.

3.2 Non-Functional Requirements

1. **Performance:**
   - The system should respond to user actions within 3 seconds.
   - Capable of handling up to 1,000 concurrent users.
2. **Usability:**
   - Intuitive interface for all user roles.
   - Mobile-friendly design for accessibility.
3. **Security:**
   - Role-based access control for secure data handling.
   - Encrypt sensitive data (e.g., customer and payment details).
4. **Scalability:**
   - System should support adding new features without major rework.
5. **Availability:**
   - Ensure 99.9% uptime.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

4. External Interface Requirements

4.1 User Interfaces

- **Admin Portal:** Dashboard with options to manage inventory, employees, and reports.
- **Customer Portal:** View product catalog, place orders, and request returns.

4.2 Hardware Interfaces

- Compatible with barcode scanners for inventory updates.
- Integrates with printers for invoice generation.

4.3 Software Interfaces

- Integrates with third-party payment gateways (e.g., Stripe, PayPal).
- Interfaces with supplier databases for reorder automation.

4.4 Communication Interfaces

- Uses HTTPS for secure communication.

5. Other Requirements

- Regular backups of data to prevent loss.
- Documentation for system usage and troubleshooting.

6. Appendices

- Glossary of terms used in the system.
- Links to relevant regulatory and compliance documents.

This SRS serves as a comprehensive guide to ensure the successful development and deployment of the **Retail Store Management System**.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Practical - 4

Aim : Develop Software project management planning(SPMP) for the Retail Store Management System.

1. Introduction

1.1 Purpose of the SPMP

This Software Project Management Plan outlines the framework for managing the development of the **Retail Store Management System**. It includes project objectives, organizational structure, risk management strategies, resource allocation, and timelines to ensure the successful delivery of the system.

1.2 Project Scope

The **Retail Store Management System** is a web-based platform designed to automate retail operations, including:

- Admin Portal for inventory and employee management.
- Sales management and transaction processing.
- Customer and return management.
- Reporting and analytics for sales, inventory, and returns.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

2. Project Organization

2.1 Organizational Structure

Role	Responsibility	Assigned To
Project Manager	Overall project planning, coordination, and execution.	[Project Manager Name]
Business Analyst	Requirement gathering and stakeholder communication.	[BA Name]
Lead Developer	Technical design and system architecture.	[Lead Developer Name]
Developers	Coding and implementation of system features.	[Developer Names]
QA Engineers	Testing and quality assurance of the system.	[QA Engineers Names]
UI/UX Designer	Designing the user interface and ensuring usability.	[UI/UX Designer Name]

2.2 Stakeholders

- **Primary Stakeholders:** Retail store owners, employees, and customers.
- **Secondary Stakeholders:** Project sponsors, system administrators, and regulatory bodies.

3. Management Process

3.1 Project Lifecycle

The project will follow the **Agile Development Model**, incorporating iterative sprints for continuous feedback and delivery.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

3.2 Milestones and Deliverables

Milestone	Deliverable	Timeline
Project Planning	Finalized requirements, SRS, and SPMP.	Week 1–2
System Design	System architecture, database design, and UI mockups.	Week 3–4
Development Sprint 1	Admin Portal: Product and employee management module.	Week 5–6
Development Sprint 2	Inventory management and low-stock alerts.	Week 7–8
Development Sprint 3	Sales processing and transaction management.	Week 9–10
Development Sprint 4	Order return management workflows.	Week 11–12
Testing Phase	System testing, bug fixing, and quality assurance.	Week 13–14
Deployment	System launch and user training.	Week 15

3.3 Tools and Technologies

- **Development Tools:** Visual Studio Code, Git, Docker.
- **Programming Languages:** JavaScript (Node.js, React), Python (for backend services).
- **Database:** PostgreSQL.
- **Testing Tools:** Selenium, JUnit, and Postman.
- **Project Management:** Jira, Trello.

4. Technical Process

4.1 Development Approach

- Agile sprints with defined goals and deliverables.
- Continuous integration and deployment (CI/CD).

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- Use of version control (Git).

4.2 Testing Approach

- Unit Testing: Verifying individual modules.
- Integration Testing: Ensuring modules work seamlessly together.
- User Acceptance Testing (UAT): Validating the system meets business needs.

5. Work Plan

5.1 Work Breakdown Structure (WBS)

Task ID	Task Name	Duration	Assigned To
1.1	Requirement Analysis	2 weeks	Business Analyst
2.1	System Design	2 weeks	Lead Developer, UI/UX Designer
3.1	Admin Portal Development	2 weeks	Development Team
3.2	Inventory Management Development	2 weeks	Development Team
3.3	Sales Module Development	2 weeks	Development Team
3.4	Return Management Development	2 weeks	Development Team
4.1	Testing and QA	2 weeks	QA Team
5.1	Deployment	1 week	DevOps Team

5.2 Resource Allocation

- **Human Resources:** 1 Project Manager, 1 Business Analyst, 2 Developers, 1 QA Engineer, 1 UI/UX Designer.
- **Hardware Resources:** Servers, workstations, barcode scanners, and printers.
- **Software Resources:** Development tools, database licenses, testing tools.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

6. Risk Management

6.1 Risk Identification

- Requirement changes during development.
- Technical challenges with system integration.
- Resource availability and delays.

6.2 Risk Mitigation

- Frequent stakeholder meetings for requirement validation.
- Use of modular design to minimize integration issues.
- Buffer time in the schedule to accommodate unexpected delays.

7. Monitoring and Control

7.1 Progress Tracking

- **Daily Standups:** Track task progress and address roadblocks.
- **Weekly Sprint Reviews:** Review completed work and gather feedback.
- **Milestone Checkpoints:** Evaluate deliverables at each milestone.

7.2 Performance Metrics

- On-time delivery of milestones.
- Number of bugs found and resolved per sprint.
- User satisfaction during UAT.

8. Closure Plan

- Final system delivery and deployment.
- Post-deployment support for two months.
- Handover of documentation and source code to the client.

This SPMP provides a detailed roadmap to ensure the successful completion of the **Retail Store Management System** project.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Practical - 5

Aim: To do Cost and Effort Estimation using different Software Cost Estimation models.

1. Use Case Points (UCP) Method

Steps for UCP Method:

1. **Identify the Actors**: 5 simple actors (e.g., Customer, Admin, Store Employee).
2. **Identify Use Cases**: 10 simple, 5 average, 3 complex.
3. **Calculate Unadjusted Use Case Points (UCP)**:
    - Actor UCP = (5 × 1) = 5.
    - Use Case UCP = (10 × 5) + (5 × 10) + (3 × 15) = 50 + 50 + 45 = 145.
    - **Total Unadjusted UCP** = 5 + 145 = 150.
4. **Adjusted UCP** = 150 × 1 (complexity factor) = 150.
5. **Effort** = 150 UCP × 15 person-hours/UCP = 2250 person-hours.
6. **Cost** = 2250 person-hours × ₹4150/hour = **₹93,75,000**.

2. Function Point Analysis (FPA)

Steps for FPA Method:

1. **Identify Function Types**:
    - **External Inputs (EI)** = 20 (e.g., adding products, customer registration).
    - **External Outputs (EO)** = 15 (e.g., sales transactions, reports).
    - **External Inquiries (EQ)** = 10 (e.g., inventory lookup, order tracking).
    - **Internal Logical Files (ILF)** = 8 (e.g., product database, sales history).
    - **External Interface Files (EIF)** = 4 (e.g., payment gateway, supplier interface).

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

2. **Assign Complexity Weights**:
    - Simple = 4, Average = 5, Complex = 7 (assigned to different functions).

3. **Calculate Unadjusted Function Points (UFP)**:
   UFP=(20×5)+(15×5)+(10×4)+(8×7)+(4×5)=100+75+40+56+20=291\text{UFP} = (20 \times 5) + (15 \times 5) + (10 \times 4) + (8 \times 7) + (4 \times 5) = 100 + 75 + 40 + 56 + 20 = 291
4. **Adjustment Factor**: Assume 1.2.
5. **Effort** = 291 UFP × 20 hours/UFP = 5820 person-hours.
6. **Cost** = 5820 person-hours × ₹4150/hour = **₹2,41,53,000**.

3. COCOMO (Constructive Cost Model)

Steps for COCOMO Model:

1. **Size**: 30 KLOC (30,000 lines of code).
2. **Mode**: Semi-Detached (moderate complexity).
3. **Constants**: For Semi-Detached mode, a = 3.0 and b = 1.12.
4. **Effort**: Effort=3.0×(30)1.12=3.0×38.3=114.9 person-months\text{Effort} = 3.0 \times (30)^{1.12} = 3.0 \times 38.3 = 114.9 \text{ person-months}
5. **Cost** = 114.9 person-months × ₹5,83,000/month = **₹6,67,57,900**.

Summary of Cost Estimation

Estimation Model	Effort (Person-Hours/Person-Months)	Cost Estimate (INR)
Use Case Points (UCP)	2250 person-hours	**₹93,75,000**
Function Point Analysis (FPA)	5820 person-hours	**₹2,41,53,000**
COCOMO	114.9 person-months	**₹6,67,57,900**

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Conclusion:

- The **Use Case Points (UCP)** method gives the **lowest cost estimate** at **₹93,75,000**, suitable for a smaller or moderately complex project.
- The **Function Point Analysis (FPA)** method gives a **higher estimate** of **₹2,41,53,000**, which is ideal for larger, more detailed systems.
- The **COCOMO model** provides the **highest estimate** at **₹6,67,57,900**, indicating a large and complex system with substantial effort and resource requirements.

These estimations provide a range of potential costs depending on the size and complexity of the project.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Practical – 6

Aim : To prepare System Analysis and System Design of identified Requirement Specification using structure design as DFD with data dictionary and Structure chart for the Retail Store Management System.

System Analysis

1. Requirements Identification for the Order Return Management Module:

The **Order Return Management** module handles the return of purchased goods from customers and ensures that the returned products are processed properly in the system, updating inventory, managing refunds, and generating reports.

The primary functions of this module include:

- **Product Return Request**: A customer can request to return a product based on predefined conditions (e.g., damaged product, wrong product, etc.).
- **Approval Process**: A store employee or admin will validate the return request.
- **Inventory Update**: If the return is approved, the product is returned to the stock.
- **Refund Processing**: The customer is refunded as per the return policy.
- **Notifications**: The system notifies the customer and the store team about the return status.

System Design

2. Data Flow Diagram (DFD)

A **Data Flow Diagram (DFD)** represents the flow of data in the system. We will create a context-level DFD for the **Order Return Management** module and then decompose it into lower-level diagrams.

Context-Level DFD (Level 0)

At this level, we capture the overall interaction between external entities and the system.

- **External Entities**: Customer, Store Admin

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- **Process**: Order Return Management
- **Data Store**: Order Details, Product Inventory, Customer Account Information
- **Data Flows**:
    - **Return Request** (from Customer to System)
    - **Return Approval/Denial** (from System to Customer)
    - **Inventory Update** (from System to Product Inventory)
    - **Refund Processed** (from System to Customer)
    - **Return Report** (from System to Store Admin)

Level 1 DFD (Decomposing the Order Return Management Process)

Here, we break down the main process into more detailed sub-processes:

1. **Receive Return Request**:
    - **Input**: Return details from the customer.
    - **Output**: Return request data.
2. **Validate Return Request**:
    - **Input**: Return details from the customer, Order and Product Information.
    - **Output**: Validated return request or return denial.
3. **Update Inventory**:
    - **Input**: Validated return request, Product information.
    - **Output**: Updated product inventory.
4. **Process Refund**:
    - **Input**: Validated return request, Payment information.
    - **Output**: Processed refund to the customer.
5. **Generate Return Report**:
    - **Input**: Return transaction details.
    - **Output**: Return report for the store admin.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

Data Dictionary

The **Data Dictionary** is used to define and describe data elements in the system.

Data Element	Description	Format/Type	Example
Return Request	Customer's request to return a product, including reason for return.	String, Date, Reason	"Defective"
Order Details	Details about the customer's order (order ID, date, items, etc.).	Integer, Date, List	Order #12345
Product Inventory	List of products in the store's inventory.	Integer, String, Float	Product ID, Stock
Refund Amount	Amount to be refunded to the customer based on the return.	Float	₹500
Return Report	Report containing the status and summary of all processed returns.	String, Date, List	"Returns Report"
Customer Account	Information about the customer (name, address, contact details, etc.).	String, Integer	Name: "John Doe"

Structure Chart

A **Structure Chart** represents the hierarchical relationship between different processes. It provides a clear overview of the system's functionality and how components are related.

Order Return Management Structure Chart:

- **Order Return Management (Top-Level Process)**:
  - **Receive Return Request**:
    - Sub-process: Capture product details, return reason, and customer information.
  - **Validate Return Request**:
    - Sub-process: Verify product condition, check return policy, and order history.
  - **Update Inventory**:

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1

- Sub-process: Update stock levels, mark product as returned.

- **Process Refund**:
  - Sub-process: Verify payment method, calculate refund amount, and initiate refund.
- **Generate Return Report**:
  - Sub-process: Generate a summary report for the store admin.

Each of these processes can be broken down further into their respective functions and sub-functions.

Example System Design Flow

1. **Customer Interaction**:
   - The customer submits a **Return Request**.
2. **System Validation**:
   - The system validates the return against predefined criteria (such as the reason for the return, the product condition, and the return policy).
3. **Inventory Update**:
   - Once validated, the system updates the product's stock level in the inventory.
4. **Refund**:
   - The customer is refunded based on the store's return policy.
5. **Report Generation**:
   - A return report is generated, summarizing the return activities for the admin.

Conclusion

In the **Order Return Management Module**, the **System Design** (using **DFD**, **Data Dictionary**, and **Structure Chart**) provides a clear outline of how data flows and how the system processes the return request. The structured design ensures that each step in the process is well-defined and can be easily translated into a functional system.

Enrollment No. : 2403031307008
Jayesh Kartarsingh Yadav
Div : 4 CC1