

University of Asia Pacific

Inventory Management System

Fahim Shahriar Alam 23201170

Tahmidur Rahman Seam 23201175

Auntic Ayon 23201183

CSE 212

Section : D Group: 6

Database System Lab

Dr. Shubhra Prosun Paul

Assistant Professor

Department of CSE



November 16, 2025

Table of Contents

1.	Database Planning	
1.1.	Introduction	1
1.2.	The Role of Inventory Management System	1
1.3.	Benefits of Inventory Management System	2
2.	Background Study	
2.1.	Case Studies	4
2.2.	Limitation of the existing system	5
2.3.	Project Objectives	6
3.	Database Design	
3.1.	Block Diagram	7
3.2.	Block Diagram with their description	7
3.3.	Schema Diagram	8
3.4.	Schema Diagram with their description	8
3.5.	ER- Diagram	11
3.6.	ER- Diagram with their description	12
4.	Implementation	
4.1.	Create Table	13
4.2.	Insert Data in Table	15
4.3.	SQL Query	17
5.	Analysis & Discussion	
5.1.	Website login page Picture with their description	21
5.2.	Entity Picture with their description	21
5.3.	Attribute Picture with their description	22
6.	Complex Engineering Problem (CEP)	
6.1.	CEP Mapping	24
6.2.	K's, P's attribute	24
6.3.	A's attribute	25
7.	Finalization & Improvement	
7.1.	Future Work	26
7.2.	Conclusion	26
8.	Reference	
8.1.	linking	27

Chapter 1

Database Planning

1.1 Introduction

Inventory management is a critical aspect of every business, whether small or large, that deals with the production, storage, and distribution of goods. It involves the careful tracking of raw materials, work-in-progress items, and finished products to ensure smooth business operations. Without effective inventory control, businesses may face challenges such as stockouts, excess inventory, inefficient resource utilization, and poor customer satisfaction. Proper inventory management, on the other hand, can lead to reduced operational costs, optimized stock levels, and improved profitability.

An efficient **Inventory Management System (IMS)** ensures that the right quantities of items are available at the right time, minimizing both excess stock and shortages. This requires real-time visibility into inventory levels, sales trends, and demand forecasting. An IMS also tracks product movement from suppliers to warehouses and ultimately to customers, helping businesses stay on top of stock replenishment and order fulfillment. Poor inventory management can not only lead to financial losses but can also impact the reputation of the company, as customers may experience delayed deliveries or unavailability of products.

At the heart of a modern **Inventory Management System** is a **Database Management System (DBMS)**. A DBMS provides the structure and organization necessary to efficiently store, manage, and retrieve vast amounts of inventory-related data. By ensuring that information is structured and accessible, the DBMS enables businesses to perform essential tasks such as tracking stock levels, managing suppliers, processing sales transactions, and updating inventory quantities in real-time. The use of a DBMS ensures data consistency, avoids duplication, and supports multiple users working concurrently. It also simplifies complex queries, allowing businesses to generate valuable reports and perform in-depth analysis.

1.2 The Role of Database Management System (DBMS) in Inventory Management

A **DBMS** forms the backbone of any modern inventory management system. By leveraging a relational database, businesses can maintain a well-structured and organized database where information about products, transactions, and suppliers is stored and can be easily updated or queried. This structured approach allows businesses to track stock levels in real-time, ensuring that inventory is replenished and managed effectively without relying on manual processes.



1.3 Benefits of Inventory Management Systems

An efficient **Inventory Management System** offers several key benefits that directly impact a business's bottom line. Some of these benefits include:

1. **Cost Reduction:** Proper inventory management minimizes the costs associated with overstocking, understocking, and wastage. For example, by analyzing sales data and stock levels, the system can automatically trigger reorders only when needed, ensuring that businesses maintain lean inventories.
2. **Improved Customer Satisfaction:** By ensuring that products are available when customers need them, businesses can avoid stockouts and delays in fulfilling orders. This leads to increased customer satisfaction and loyalty, which are crucial for business success in competitive markets.
3. **Enhanced Decision Making:** With real-time inventory data at their fingertips, managers and business owners can make informed decisions regarding procurement, pricing, and promotional strategies. By understanding product performance and sales trends, businesses can adjust their strategies accordingly.
4. **Streamlined Operations:** An automated IMS reduces manual intervention, simplifying tasks such as stock counting, order processing, and inventory tracking. Employees can focus on higher-value tasks rather than spending time on manual data entry or inventory checks.
5. **Better Cash Flow Management:** Effective inventory control helps businesses avoid over-investing in inventory. With an efficient system in place, businesses can free up cash that would otherwise be tied up in excess stock and reinvest it into other areas of the business.

This project focuses on the development of an **Inventory Management System (IMS)** using DBMS principles, leveraging a relational database to streamline inventory control. The database will store crucial information about products, suppliers, customers, and transactions. It will allow for the seamless addition, update, and retrieval of data related to inventory items, and support business operations such as stock updates after sales or purchases. The system will also include functionality to generate reports and queries that can help businesses track product performance, evaluate sales trends, and make data-driven decisions.

The goal of this project is to demonstrate how a well-designed **DBMS-based IMS** can significantly enhance operational efficiency, improve decision-making, and reduce errors in inventory management. By automating various tasks such as stock level adjustments, reorder point calculations, and transaction logging the system will not only save time but also improve the accuracy and reliability of the inventory data. Furthermore, it will provide businesses with actionable insights, enabling them to optimize stock levels, forecast demand more accurately, and improve their overall supply chain processes.

In conclusion, an efficient **Inventory Management System (IMS)** is essential for businesses looking to optimize their supply chain, improve operational efficiency, and deliver better customer experiences. By incorporating a **DBMS**, this system will provide businesses with accurate, real-time data to make informed decisions. This project aims to develop a fully functional IMS that showcases the power of **Database Management Systems** in addressing real-world inventory challenges, improving business outcomes, and enabling growth.

Chapter 2

Background Study

2.1 Case Studies

1. harishy0406 / Inventory-Management-System (GitHub Project)

The Inventory Management System developed by *harishy0406* is a Java-based desktop application created using NetBeans IDE and connected to a MySQL database through XAMPP. The system offers real-time tracking of inventory levels, allowing users to add, update, and delete inventory items efficiently. It also includes features such as report generation for both inventory and sales data, user authentication to ensure secure access, and options for database backup and restoration. This project stands out for its user-friendly graphical interface and implementation of standard CRUD operations, making it a practical and reliable tool for managing inventory. A Java-based desktop application developed using NetBeans IDE, with MySQL via XAMPP. Features include: real-time tracking of inventory levels, add/update/delete of inventory items, report generation for inventory and sales, user authentication/authorization, and backup/restore of database. Highlight: emphasizes usability (GUI), standard CRUD operations, and reporting. [GitHub](#)

2. nishant0820 / Inventory-Management-System (GitHub Project)

The Inventory Management System created by *nishant0820* is a Python-based application that uses the tkinter library for its graphical user interface and sqlite3 for a lightweight database solution. The project also integrates modules such as PIL for image handling and datetime for managing timestamps. Designed primarily for educational purposes, this system has a simpler scope and limited functionality, making it ideal for demonstration and learning. It offers basic inventory management features and serves as a practical tool for understanding core concepts of inventory control and system design. Built in Python, using tkinter for GUI, sqlite3 for lightweight database, and other modules like PIL (images), datetime. Purpose more educational or demonstration-based; simpler scope, more limited in scale. [GitHub](#)

3. InvenTree (GitHub / Open-Source Project)

InvenTree is an open-source inventory management system developed by the InvenTree community, built with a Python backend using the Django framework. It features a web-based admin interface and provides a REST API, making it accessible and flexible for users. The system is designed specifically for low-level stock control and detailed parts tracking. Additionally, its plugin-based architecture allows for easy customization and extension, enabling users to tailor the system to their specific inventory management needs. An open-source system by the InvenTree community. The backend is Python + Django. Provides a web-admin interface and REST API. Designed for low-level stock control and parts tracking. Also supports plugin-architecture so it can be extended. [GitHub](#)

4. Case Study: Amazon Fulfillment Centers

The case study on Amazon Fulfillment Centers, conducted by Bahareh Aghababaei in 2024, explores how Amazon optimizes inventory levels across multiple fulfillment centers. It focuses on balancing the costs of shipping, holding, and receiving inventory to improve overall efficiency. The study employs large-scale optimization models to tackle the complex decision-making process, addressing the computational challenges associated with NP-hard problems by using techniques such as aggregation, hierarchical disaggregation, and greedy algorithms. This approach highlights the sophisticated strategies needed to manage inventory effectively in a large and dynamic supply chain environment. Examines effective decision making about how much inventory to store at different fulfillment centers, balancing shipping, holding, and receiving costs. Uses large-scale optimization models; deals with NP-hardness, uses aggregation + hierarchical disaggregation + greedy methods for computational tractability. [Open Collections](#)

5. Case Study: Quality of Inventory Management System at BARMM-Ministry of Public Works, Basilan

The case study by Lesley Ann Atilano-Tang and Kurt Damsani in 2023 investigates the inventory management practices at the BARMM Ministry of Public Works in Basilan. Using a mixed-method approach, combining both primary and secondary data, the study reveals several shortcomings in the existing system, including poor documentation, lack of standardization, and weak tracking of inventory levels. The authors suggest several improvements, including the implementation of a centralized inventory system, the establishment of standard operating procedures (SOPs), and enhanced staff training to address these issues and improve overall inventory management efficiency in the public sector. Mixed-method case study (primary + secondary data) investigating existing local public-sector inventory practices. Findings: poor documentation, lack of standardization, weak tracking of inventory levels. Suggested improvements: centralized inventory system, standard operating procedures (SOPs), staff training. [SSRN](#)

2.2 Limitation of the existing system

Although several inventory management systems have been developed by different users, most of them face common drawbacks:

- 1. Limited Scalability** – Many GitHub projects (e.g., Java Swing or Tkinter-based systems) are designed for small shops, making it difficult to handle large volumes of data or multi-branch operations.
- 2. Single-User or Local Usage** – Several desktop-based systems (such as those created with Java or Python) work only on a single machine, lacking proper support for multi-user access and real-time synchronization.
- 3. Weak Security Features** – While some projects include basic login modules, advanced features like encryption, audit trails, or role-based privilege control are often missing.

4. **Lack of Forecasting and Analytics** – Most existing systems focus only on CRUD operations (Create, Read, Update, Delete) without offering demand forecasting, sales trend analysis, or low-stock alerts.
5. **No Integration with Other Systems** – Many community projects operate as standalone tools, without the ability to integrate with accounting software, e-commerce platforms, or supplier databases.
6. **Limited Reporting Capabilities** – Reports generated in several GitHub projects are basic and static, with little support for customizable dashboards or advanced data visualization.
7. **Poor Documentation and Maintenance** – A number of open-source projects are not actively maintained, and incomplete documentation makes them difficult to adapt or extend.
8. **Insufficient Error Handling** – In some systems, invalid inputs or unexpected operations (e.g., negative stock, duplicate records) are not properly managed, leading to data inconsistencies.

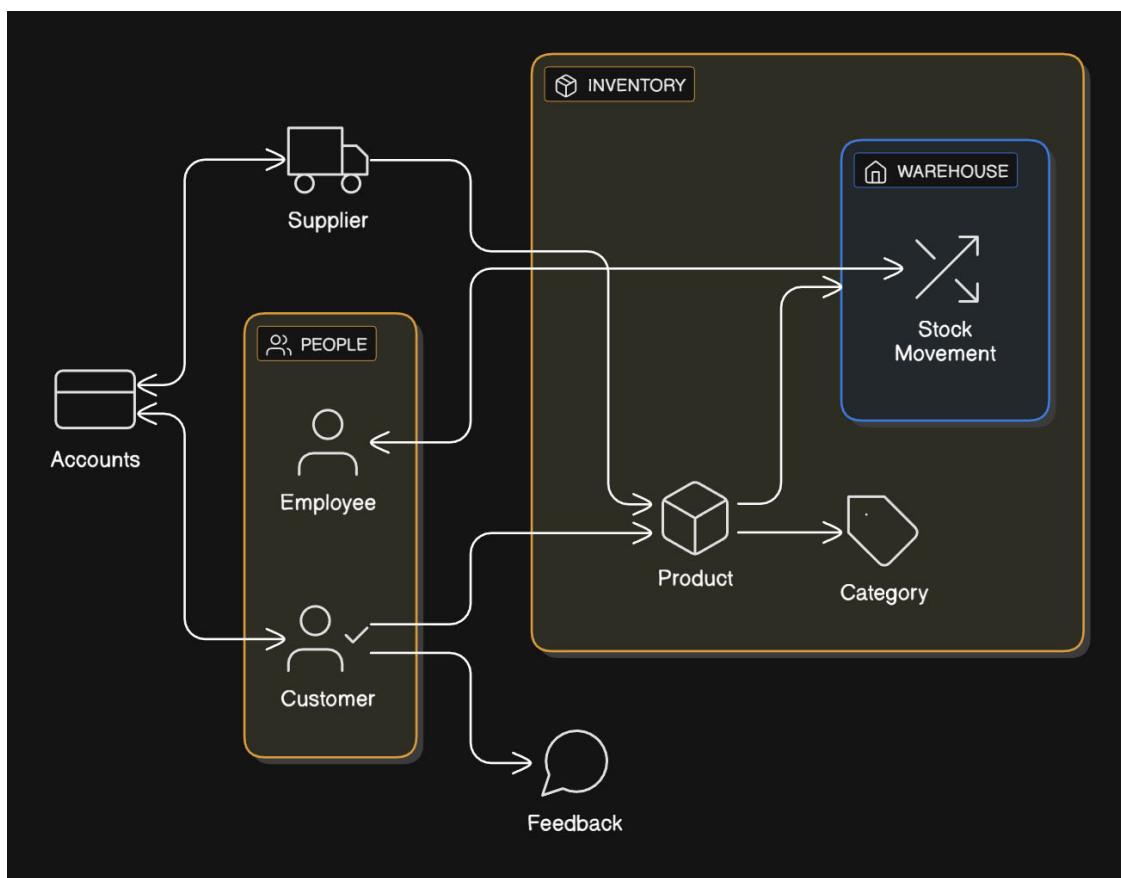
2.3 Project objective

- To overcome scalability issues, the system should be designed with a relational database that can handle large datasets and support multi-branch operations.
- To address single-user limitations, the system can be built as a web-based application with multi-user access and real-time synchronization.
- To improve security, role-based authentication, data encryption, and activity logs should be implemented.
- To provide better decision support, the system should include forecasting, low-stock alerts, and advanced reporting with customizable dashboards.
- To ensure adaptability, the system should be well-documented, actively maintained, and designed with integration capabilities for external platforms.

Chapter 3

Database Design

3.1 Block Diagram

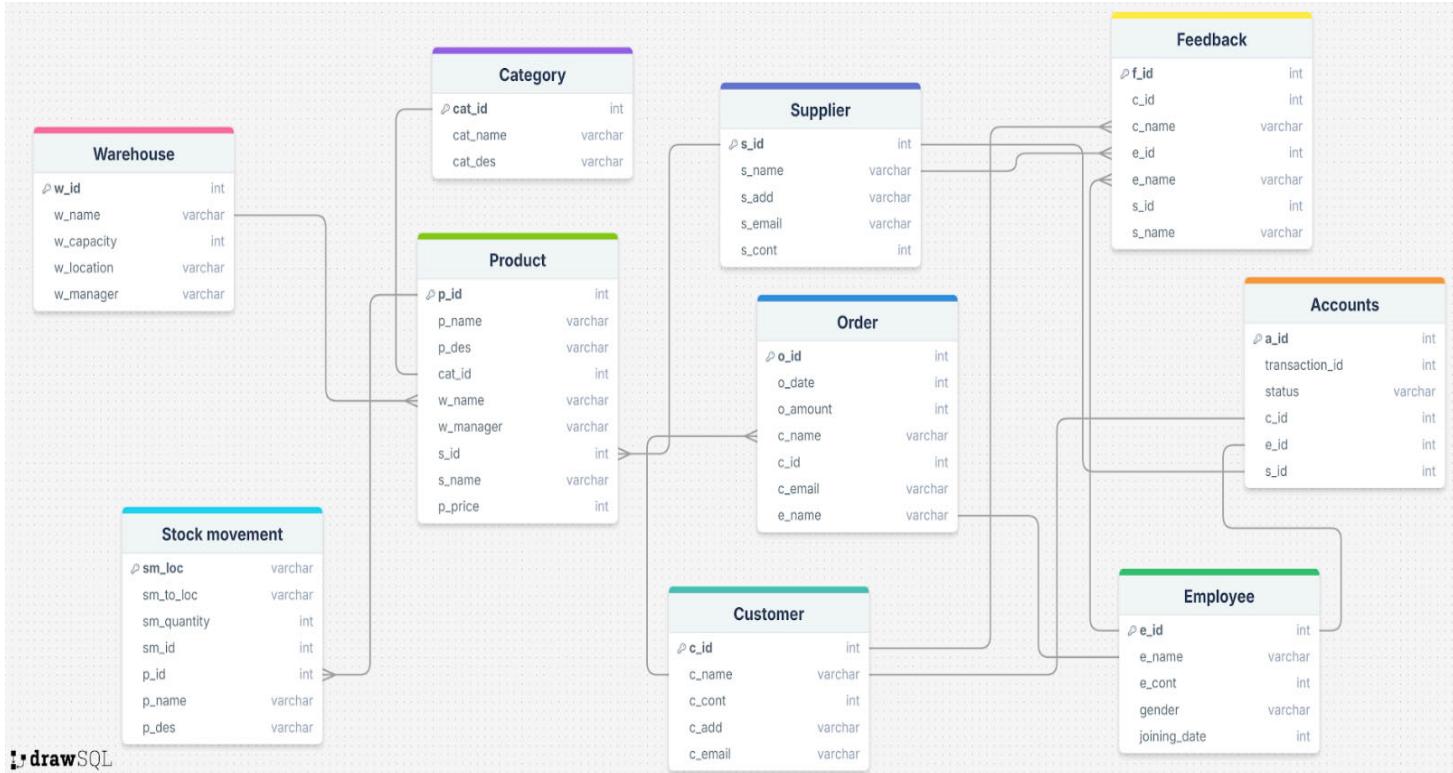


3.2 Block Diagram with their description

The diagram illustrates the flow of information and materials between the main components involved in inventory management:

1. **Supplier** → Provides raw materials or products to the system.
2. **Product** → Represents the items managed within the system.
3. **Stock Movement** → Tracks the transfer of goods between warehouses, employees, and customers.
4. **Category** → Helps organize products for better management and identification.
5. **Employee** → Handles inventory operations, including stock updates, categorization, and customer service.
6. **Customer** → Purchases the products and provides **feedback**.
7. **Feedback** → Returns to improve service quality and supplier decisions.
8. **Accounts** → Manages financial transactions related to purchases, sales, and supplier payments.

3.3 Schema Diagram



3.4 Schema Diagram with their description

1. Warehouse

- **Purpose:** Keeps information about each warehouse that stores products.
- **Attributes:**
 - w_id (warehouse ID)
 - w_name, w_capacity, w_location, w_manager
- **Relation:** Connected to **Product** and **Stock Movement** (to track where goods move).

2. Category

- **Purpose:** Classifies products into types (e.g., electronics, clothing, etc.).
- **Attributes:**
 - cat_id, cat_name, cat_des
- **Relation:** Linked with **Product** — each product belongs to one category.

3. Supplier

Purpose: Contains details of suppliers providing goods to the company.

Attributes:

- o s_id, s_name, s_add, s_email, s_cont

Relation: Connected to **Product**, **Accounts**, **Feedback**, and **Order** — suppliers are part of the supply and financial flow.

4. Product

- **Purpose:** The central table containing all product details.
- **Attributes:**
 - o p_id, p_name, p_des, cat_id, w_manager, s_id, p_price
- **Relation:**
 - o Linked with **Category** (product type)
 - o **Supplier** (who provides it)
 - o **Warehouse and Stock Movement** (where it's stored and moved)

5. Stock Movement

- **Purpose:** Tracks how products move from one location to another (like from warehouse to customer).
- **Attributes:**
 - o sm_id, sm_from_loc, sm_to_loc, sm_quantity, p_id
- **Relation:** Connected to **Product** and **Warehouse** — shows where products go and come from.

6. Order

- **Purpose:** Keeps records of all customer orders.
- **Attributes:**
 - o o_id, o_date, o_amount, c_id, e_id
- **Relation:** Linked with **Customer** and **Employee** — customers make orders handled by employees.

7. Customer

- **Purpose:** Stores customer details.
- **Attributes:**
 - c_id, c_name, c_cont, c_add, c_email
- **Relation:** Connected with **Order**, **Feedback**, and **Accounts** — customers place orders, pay bills, and give feedback.

8. Employee

- **Purpose:** Represents employees who manage warehouse, sales, and customer service.
- **Attributes:**
 - e_id, e_name, e_cont, gender, joining_date
- **Relation:** Connected with **Order**, **Accounts**, and **Feedback**.

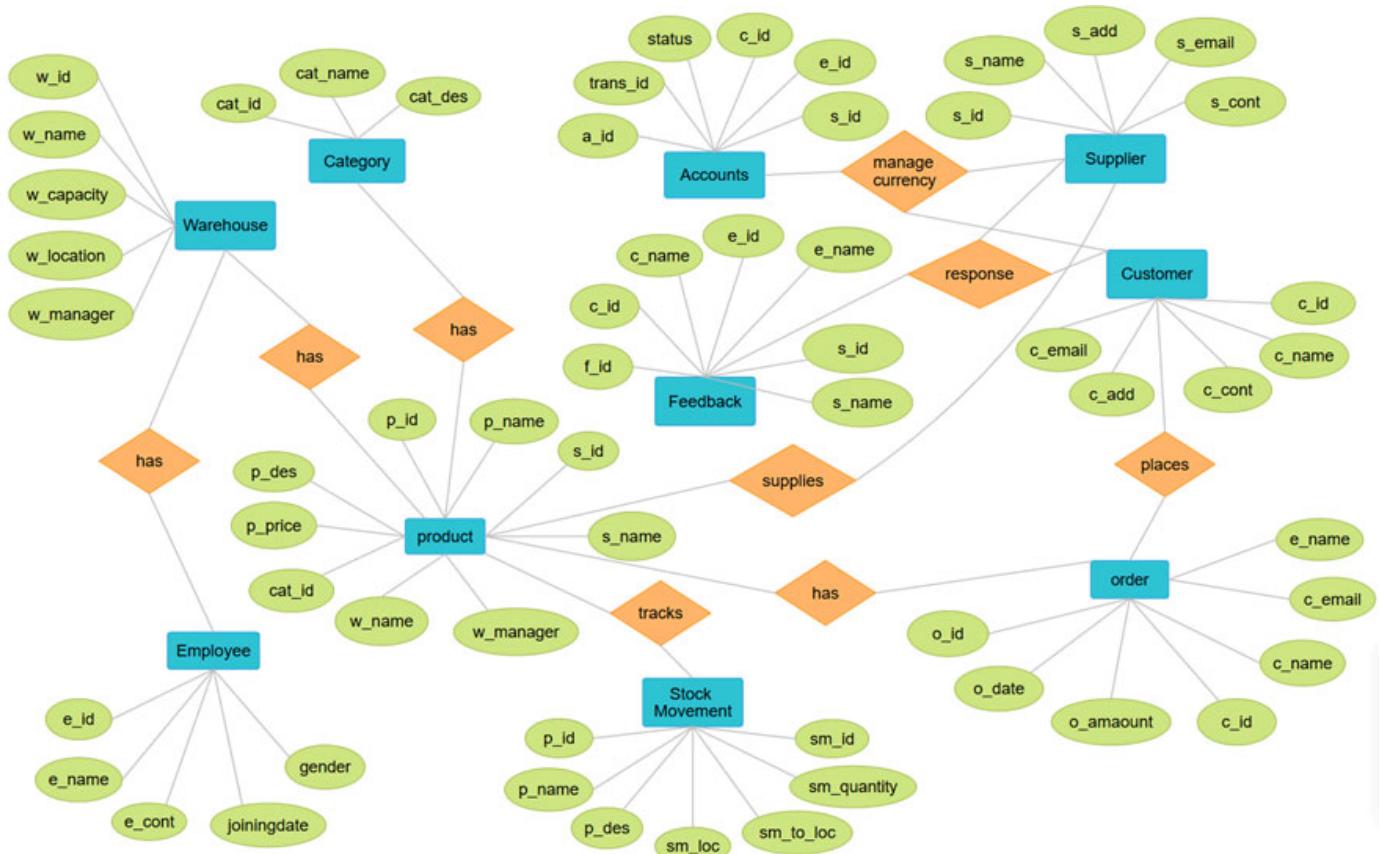
9. Feedback

- **Purpose:** Stores feedback from customers or employees about products or suppliers.
- **Attributes:**
 - f_id, c_id, e_id, s_id, f_name
- **Relation:** Linked to **Customer**, **Employee**, and **Supplier**.

10. Accounts

- **Purpose:** Manages all financial transactions — payments to suppliers, customer payments, etc.
- **Attributes:**
 - a_id, transaction_id, status, c_id, e_id, s_id
- **Relation:** Connected to **Customer**, **Employee**, and **Supplier** — tracking who's involved in each transaction.

3.5 ER Diagram



3.6 ER Diagram with their description

- The diagram represents a conceptual Entity Relationship Diagram (ERD) for an Inventory Management System (IMS).
- It shows how different entities (like Product, Warehouse, Supplier, Order, Customer, etc.) are connected and how data flows between them.
- **Product** is the central entity, linked to Warehouse, Supplier, Stock Movement, and Category — showing that all activities revolve around the product.
- The **Warehouse** stores information about where products are kept, including warehouse name, capacity, and manager.
- **Category** groups products into types, making it easier to organize and search for them.
- **Supplier** provides products to the system and is linked to both Accounts (for payments) and Feedback (for performance reviews).
- **Stock Movement** tracks the transfer of products between warehouses or to customers, including quantity and location.
- **Order** records all customer purchases and connects with both Customer and Employee, showing who made and handled the order.
- **Customer** holds details about buyers and links to Orders, Feedback, and Accounts.
- **Employees** store staff information and connect to Accounts and Feedback, showing who manages operations and customer relations.
- **Feedback** captures customer and supplier responses, helping improve service and product quality.
- **Accounts** manage all financial transactions, including supplier payments and customer invoices, connecting to several entities for complete tracking.

Overall, the diagram shows how all departments—warehouse, sales, accounts, and feedback—work together to ensure efficient inventory control and smooth business operations.

Chapter 4

Implementation

4.1 Create Table

Create Database inventory management

```
create table product6(
    pid number(10) primary key,
    pname varchar(100) not null,
    pdestination varchar(100),
    pprice NUMBER(10),
    catid number(10),
    wname varchar(100),
    wmanager varchar(100),
    sid number(10),
    sname varchar(100)
);

create table Employee7(
    eid number(10) primary key,
    ename varchar(100) not null,
    econt number (10),
    gender varchar(100),
    joiningdate varchar(100)

);

create table Category(
    catid number(10) primary key,
    catname varchar(100) not null,
    catdes varchar(100)

);

create table Supplier(
    sid number(10) primary key,
    sname varchar(100) not null,
    sadd varchar(100),
    semail varchar(100),
    scont number(10)
);
```

```

create table Accounts(
    aid number(10) primary key,
    transactionid number(10) not null,
    status varchar(100),
    cid number(10),
    eid number(10),
    sid number(10)
);

create table Customer1(
    cid number(10) primary key,
    cname varchar(100),
    ccont number(10),
    cadd varchar(100),
    cemail varchar(100)
);

create table Feedback(
    fid number(10) primary key,
    cid number(10),
    cname varchar(100),
    eid number(10),
    ename varchar(100),
    sid number(10),
    sname varchar(100)
);

create table Warehouse(
    wid number(10) primary key,
    wname varchar(100),
    wcapacity varchar(100),
    wlocation varchar(100),
    wmanager varchar(100)
);

create table Stock_Movement1(
    smloc varchar(100),
    smtoloc varchar(100),
    smquantity varchar(100),
    smid number(10) primary key,
    pid number(10),
    pname varchar(100),
    pdestination varchar(100)
);

```

4.2 Insert Data In Table

Product

```
insert into product values
(01, 'Laptop', 'UK', 1500, 01, 'Gulsan', 'Kamal', 01, 'Alif');
(02, 'Phone', 'UK', 800, 02, 'Mirpur', 'Masud', 02, 'Rahim');
(03, 'Tablet', 'USA', 600, 03, 'Agargaon', 'Asif', 03, 'Monsur');
(04, 'Monitor', 'UAE', 400, 04, 'Uttara', 'Siam', 04, 'Marjan');
(05, 'Headphones', 'Brazil', 75, 05, 'Farmgate', 'Fahim', 05, 'Jui');
(06, 'Keyboard', 'Maxico', 100, 06, 'Faridpur', 'Omar', 06, 'Saif');
(07, 'Mouse', 'Maxico', 15, 07, 'Bhatiyari', 'Ayon', 07, 'Faruk');
(08, 'Webcam', 'Uganda', 120, 08, 'Pollobi', 'Jubayer', 08, 'Leo');
```

Employee

```
insert into Employee7 values
(01, 'Tom Hardy', 1234567890, 'Male', '02/07/2022');
(02, 'Tom Cruse', 6734567890, 'Male', '07/03/2022');
(03, 'Tom Holland', 9934567890, 'Female', '09/03/2022');
(04, 'Chris Gaile', 9934567890, 'Female', '01/01/2022');
(06, 'Henry Cavil', 8634567890, 'Female', '09/03/2022');
```

Category

```
insert into Category values
(01, 'Plastic', 'Low');
(02, 'Ceramic', 'Average');
(03, 'Wood', 'Good');
(04, 'Concrete', 'Average');
(05, 'Aluminium', 'Excellent');
(06, 'Titanium', 'Excellent');
(07, 'Glass', 'Good');
(08, 'Metal', 'Average');
```

Supplier

```
insert into Supplier values
(01, 'Alif', 'Mirpur', 'asif@example.com', 1234567890);
(02, 'Rahim', 'Pallabi', 'rahim@example.com', 2234567890);
(03, 'Monsur', 'Kajipara', 'monsur@example.com', 1134567890);
(04, 'Marjan', 'Farmgate', 'marjan@example.com', 3334567890);
(05, 'Jui', 'Uttara', 'jui@example.com', 9234567890);
(06, 'Saif', 'Sahbag', 'saif@example.com', 8834567890);
(07, 'Faruk', 'Motizil', 'faruk@example.com', 6634567890);
(08, 'Leo', 'Faridpur', 'leo@example.com', 7734567890);
```

Accounts

```
insert into Accounts values
(101, 5758, 'Pending', 01, 01, 01);
(102, 5778, 'Completed', 02, 02, 02);
(103, 7758, 'Overdue', 03, 03, 03);
(104, 5752, 'Refunded', 04, 04, 04);
(105, 5558, 'Completed', 05, 05, 05);
(106, 8858, 'Pending', 06, 06, 06);
(107, 9958, 'Overdue', 07, 07, 07);
(108, 5228, 'Completed', 08, 08, 08);
```

Customer

```
insert into Customer1 values
(01, 'Lyra', 1234567890, 'Pallabi', 'lyra@example.com');
(02, 'Orion', 2234567890, 'Kajipara', 'orion@example.com');
(03, 'Kairo', 1334567890, 'Farmgate', 'kairo@example.com');
(04, 'Nova', 1234637890, 'Sahbag', 'nova@example.com');
(05, 'Soren', 1239967890, 'Motizil', 'soren@example.com');
(06, 'Elara', 1234567990, 'Faridpur', 'elara@example.com');
(07, 'Talon', 1234560890, 'Mirpur', 'talon@example.com');
(08, 'Mira', 2234567890, 'Uttara', 'mira@example.com');
```

Feedback

```
insert into Feedback values
(501, 01, 'Lyra', 01, 'Tom Hardy', 01, 'Alif');
(502, 02, 'Orion', 02, 'Tom Cruse', 02, 'Rahim');
(503, 03, 'Kairo', 03, 'Tom Holland', 03, 'Monsur');
(504, 04, 'Nova', 04, 'Chris Gaile', 04, 'Marjan');
(505, 05, 'Soren', 05, 'Chris Hemsworth', 05, 'Jui');
(506, 06, 'Elara', 06, 'Henry Cavil', 06, 'Saif');
(507, 07, 'Talon', 07, 'Brad Pitt', 07, 'Faruk');
(508, 08, 'Mira', 08, 'Chris Evans', 08, 'Leo');
```

Warehouse

```
insert into Warehouse values
(701, 'Gulsan', '5000 m2', 'America', 'Kamal');
(702, 'Mirpur', '3000 m2', 'Jamica', 'Masud');
(703, 'Agargaon', '8000 m2', 'Canada', 'Asif');
(704, 'Uttara', '4000 m2', 'Australia', 'Siam');
(705, 'Farmgate', '9000 m2', 'Nijaria', 'Fahim');
(706, 'Faridpur', '7000 m2', 'Egypt', 'Omar');
(707, 'Bhatiyari', '10000 m2', 'Russia', 'Ayon');
(708, 'Pollobi', '12000 m2', 'Brazil', 'Jubayer');
```

Stock Movement

```
insert into Stock_Movement1
('Washington', 'Los Angeles', '500000', 301, 01, 'Laptop', 'UK');
('Seoul', 'New Delhi', '250000', 302, 02, 'Phone', 'UK');
('Dhaka', 'Kolkata', '350000', 303, 03, 'Tablet', 'USA');
('Rajsahai', 'Kasmir', '700000', 304, 04, 'Monitor', 'UAE');
('Chittagong', 'Katmundu', '830000', 305, 05, 'Headphones', 'Brazil');
('Sylhet', 'Manchester', '200000', 306, 06, 'Keyboard', 'Mexico');
('Barisal', 'Milan', '970000', 307, 07, 'Mouse', 'Mexico');
('Chadpur', 'Franch', '550000', 308, 08, 'Webcam', 'Uganda');
```

4.3 SQL QUERY

--Find all warehouses with their locations, capacities and their manager--

Select * from Warehouse;

WID	WNAME	WCAPACITY	WLOCATION	WMANAGER
702	Mirpur	3000 m ²	Jamica	Masud
703	Agargaon	8000 m ²	Canada	Asif
704	Uttara	4000 m ²	Australia	Siam
701	Gulsan	5000 m ²	America	Kamal
705	Farmgate	9000 m ²	Nijaria	Fahim
706	Faridpur	7000 m ²	Egypt	Omar
707	Bhatiyari	10000 m ²	Russia	Ayon
708	Pollobi	12000 m ²	Brazil	Jubayer

--Find the accounts where the status is exactly 'pending--

SELECT aid, transactionid, cid, eid, sid FROM Accounts WHERE status = 'Pending';

	AID	TRANSACTIONID	CID	EID	SID	
1	101	5758	1	1	1	
2	106	8858	6	6	6	

--List all the employees whose joining date is after march 8, 202--

SELECT eid, ename,econt,gender, joiningdate FROM Employee7 WHERE joiningdate > '08/03/2022';

	EID	ENAME	ECONT	GENDER	JOININGDATE
1	6	Henry Cavil	8634567890	Female	09/03/2022
2	3	Tom Holland	9934567890	Female	09/03/2022
3	8	Chris Evans	1174567890	Male	09/07/2022

--Find the stock and product details where stock quantity is greater them 20000--

SELECT smloc, smtoloc, smid, pid, pname, pdestination FROM Stock_Movement1 WHERE TO_NUMBER(smquantity) > 200000;

	SMLOC	SMTOLOC	SMID	PID	PNAME	PDESTINATION
1	Barisal	Milan	307	7	Mouse	Maxico
2	Washington	Los Angeles	301	1	Laptop	UK
3	Seoul	New Delhi	302	2	Phone	UK
4	Dhaka	Kolkata	303	3	Tablet	USA
5	Rajsahi	Kasmir	304	4	Monitor	UAE
6	Chittagong	Katmundu	305	5	Headphones	Brazil
7	Chadpur	Franch	308	8	Webcam	Uganda

--Get feedbacks from customer,employee,supplier where fid is between 502 to 51--

SELECT fid,cid, cname, eid, ename, sid, sname FROM Feedback WHERE fid BETWEEN 502 AND 508;

	FID	CID	CNAME	EID	ENAME	SID	SNAME
1	503	3	Kairo	3	Tom Holland	3	Monsur
2	504	4	Nova	4	Chris Gaile	4	Marjan
3	502	2	Orion	2	Tom Cruse	2	Rahim
4	505	5	Soren	5	Chris Hemsworth	5	Jui
5	506	6	Elara	6	Henry Cavil	6	Saif
6	508	8	Mira	8	Chris Evans	8	Leo
7	507	7	Talon	7	Brad Pitt	7	Faruk

--List of customer information--

SELECT cid, cname, ccont, cadd, cemail FROM Customer1;

CID	CNAME	CCONT	CADD	CEMAIL
1	Lyra	1234567890	Pallabi	lyra@example.com
2	Orion	2234567890	Kajipara	orion@example.com
5	Soren	1239967890	Motizil	soren@example.com
3	Kairo	1334567890	Farmgate	kairo@example.com
4	Nova	1234637890	Sahbag	nova@example.com
7	Talon	1234560890	Mirpur	talon@example.com
6	Elara	1234567990	Faridpur	elara@example.com
8	Mira	2234567890	Uttara	mira@example.com

--List of category of products where category description is both average and excellent--

SELECT catid, catname, catdes FROM Category WHERE catdes IN ('Average', 'Excellent');

	CATID	CATNAME	CATDES
1	2	Ceramic	Average
2	5	Aluminium	Excellent
3	4	Concrete	Average
4	8	Metal	Average
5	6	Titanium	Excellent

--Give pname, pprice, wname, sname from product6 where pid is between 1 to 7--

SELECT pname, pprice, wname, sname FROM product6 WHERE pid BETWEEN 1 AND 7;

	PNAME	PPRICE	WNAME	SNAME
1	Tablet	600	Agargaon	Monsur
2	Keyboard	100	Faridpur	Saif
3	Phone	800	Mirpur	Rahim
4	Mouse	15	Bhatiyari	Faruk
5	Monitor	400	Uttara	Marjan
6	Headphones	75	Farmgate	Jui
7	Laptop	1500	Gulsan	Alif

--Find product and warehouse details where pdestination is uk and maxico--

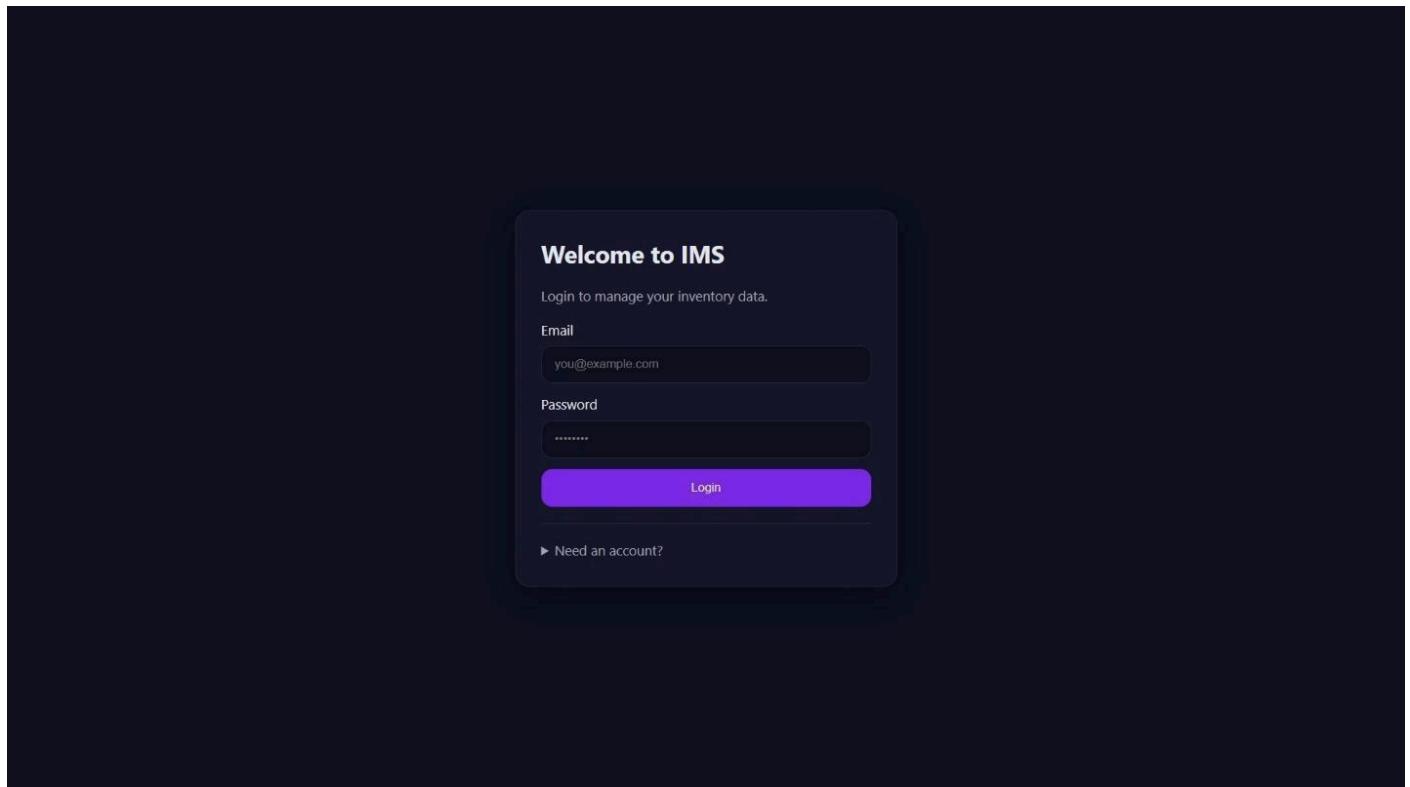
SELECT pid, pname, pdestination, pprice, wname, wmanager FROM product6 WHERE pdestination IN ('UK', 'Maxico');

	PID	PNAME	PDESTINATION	PPRICE	WNAME	WMANAGER
1	6	Keyboard	Maxico	100	Faridpur	Omar
2	2	Phone	UK	800	Mirpur	Masud
3	7	Mouse	Maxico	15	Bhatiyari	Ayon
4	1	Laptop	UK	1500	Gulsan	Kamal

Chapter 5

Analysis & Discussion

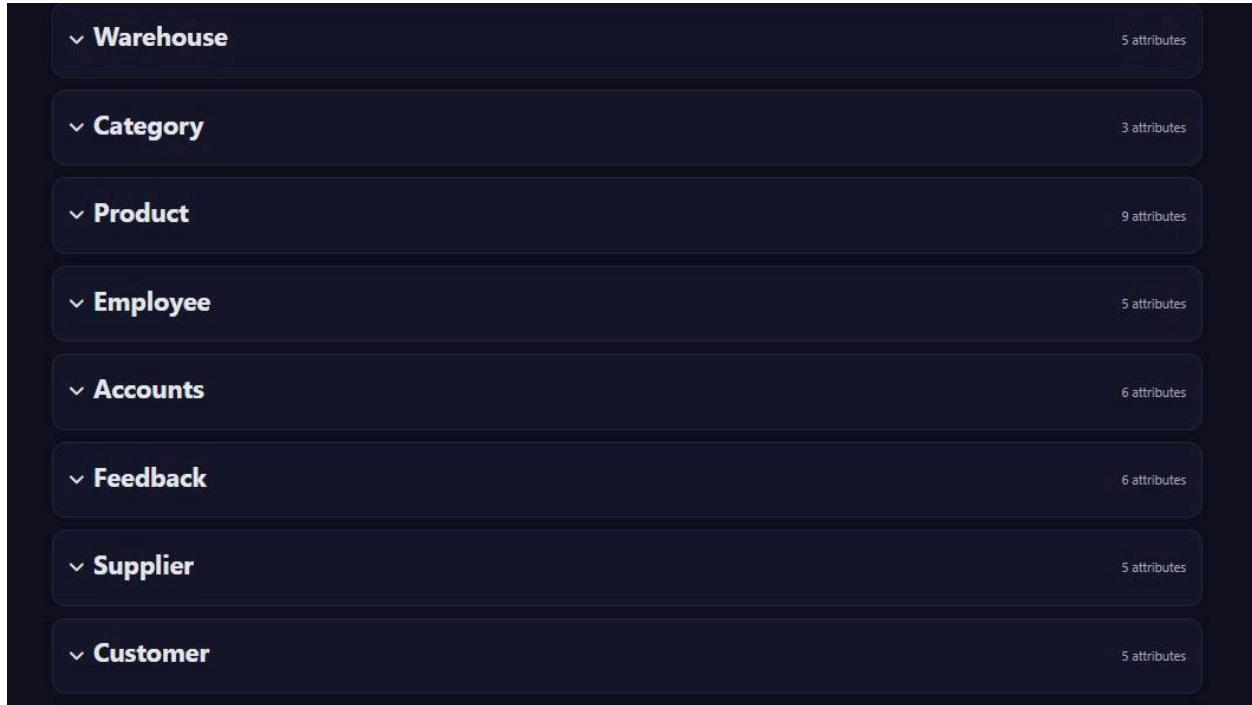
5.1 Website login page screenshot



Picture 1 – Login Page

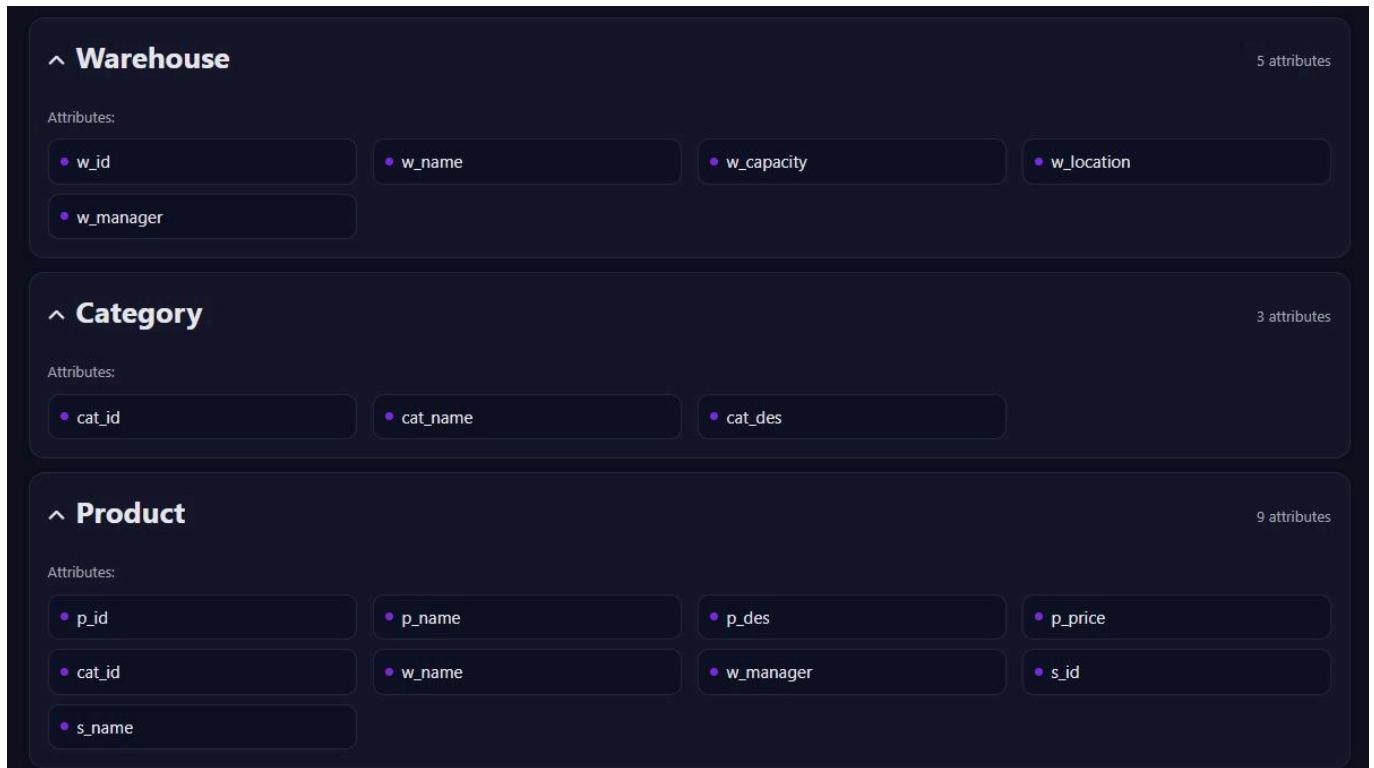
- **Description:** This is the **login interface** of the Inventory Management System (IMS).
- **Purpose:** It allows users (like admins or employees) to **log in securely** using their email and password to access the system. It also includes an option to create a new account if the user doesn't have one.
- **Function:** Provides **authentication and access control** to protect the system from unauthorized use.

5.2 Entity Picture



Picture 2 – Entities Overview

- **Description:** This page displays all the **main entities** (like Warehouse, Category, Product, Employee, Accounts, etc.) in the system.
- **Purpose:** It serves as the **main dashboard** where users can view and select entities to see their stored attributes.
- **Function:** Helps users **navigate and manage** the structure of the database by showing how many attributes each entity contains.



Picture 3 – Attribute Details

- **Description:** This screen shows the **detailed attributes** of a selected entity (for example, Product).
- **Purpose:** It allows users to **view and verify** specific fields stored in that entity such as p_id, p_name, p_price, w_name, s_id, etc.
- **Function:** Provides a **data inspection interface**, enabling the user to understand how each entity's data is organized and stored in the database.

Chapter 6

Complex Engineering Problem

6.1 CEP Mapping

Our project is a solution to a complex engineering problem because it can't be resolved without in depth engineering knowledge. There is no obvious solution to and requires some amount of abstract thinking depending on the Database model. It also involves a diverse group of stakeholders with widely varying needs.

6.2 K's, P's Attribute

K's Attribute	How this IMS addresses the attribute	Course Outcome (CO)	Program Output (PO)
K3 - Engineering fundamentals	The project requires systematic application of DBMS fundamentals: relational modelling, normalization, DDL/DML (table creation, inserts, joins) and transaction-aware updates (stock adjustments). The IMS write-up includes schema, ER diagrams and SQL scripts.	CO1, CO2	PO1, PO2
K5 - Engineering design	We produced an ER diagram and schema diagrams to capture entities (Product, Supplier, Warehouse, Stock_Movement, Customer, Accounts, Feedback) and relationships; this guided table definitions and constraints. Diagrams and schema are included in the IMS report.	CO3, CO4	PO3, PO2
K6 - Engineering practice	The design is implemented using SQL (CREATE TABLE, INSERTs, SELECT queries). The project shows real queries (reports, aggregation, joins, subqueries) and sample data to validate behavior.	CO1, CO2, CO5	PO1
P1 - Depth of knowledge required	Requires solid undergraduate-level DBMS theory (data modeling, queries) and practical SQL skills to implement the IMS.	CO3, CO4, CO5	PO1, PO2
P3 - Depth of analysis required	Must analyze different inventory scenarios (stock movement, reorder, multi-warehouse availability), so some abstract reasoning and algorithmic thinking is needed to design queries and reports.	CO3, CO5	PO3
P7 - Interdependence	The system has interdependent modules (orders ↔ products ↔ warehouses ↔ stock movements ↔ accounts). Implementation requires coordinating schemas and transactional logic across these subproblems.	CO3, CO4, CO5	PO2, PO7

6.3 A's attribute

A's Attribute	How this IMS addresses the attribute	Course Outcome (CO)	Program Output (PO)
A1 - Range of resources	Implementation requires a DBMS (e.g., MySQL / Oracle / MSSQL), development workstation, and sample datasets. The IMS includes DDL/DML and sample insertions demonstrating required resources.	CO1	PO5
A5 - Familiarity with domain & tools	Project members needed to learn inventory /business concepts (stock movement, warehouses, suppliers) and DBMS tooling (SQL, schema design). The background and limitations sections show this learning.	CO8	PO11

Chapter 7

Finalization & Improvement

7.1 Future Work

While the current IMS performs basic inventory control and data management efficiently, there are several areas where it can be further enhanced in future versions:

1. Integration with Cloud Databases:

Future versions could migrate from local storage to cloud-based databases such as Firebase, AWS RDS, or Google Cloud SQL to enable multi-user, real-time data access from any location.

2. Automation and Notifications:

Implementing automatic low-stock alerts, purchase order generation, and email/SMS notifications would make the system more proactive and user-friendly.

3. AI-Based Demand Forecasting:

Machine learning algorithms could be added to predict future product demand, optimize stock levels, and reduce wastage or shortages.

4. Advanced Security Features:

Role-based access control, data encryption, and activity logging could be integrated to improve data protection and prevent unauthorized access.

5. Data Visualization Dashboards:

A graphical reporting dashboard with interactive charts and analytics could be added to help managers better understand trends and performance metrics.

6. Integration with External Systems:

The IMS could be linked with accounting software, supplier APIs, or e-commerce platforms (like Shopify or Amazon) for seamless synchronization of data.

7. Mobile Application Development:

Developing a mobile version of the IMS would allow users to manage inventory and monitor stock movements conveniently on their smartphones.

8. Barcode and RFID Implementation:

Incorporating barcode scanning or RFID tracking would help automate product identification and make stock movement faster and more accurate.

7.2 Conclusion

In conclusion, this project successfully demonstrates the design and implementation of a Database-Driven Inventory Management System (IMS) that streamlines and automates inventory operations. By applying the principles of Database Management Systems (DBMS), the project provides an efficient structure for storing, managing, and retrieving large volumes of inventory data related to products, suppliers, warehouses, customers, and transactions. It minimizes manual work, reduces errors, and enhances business decision-making through accurate and real-time data handling.

The system's entity-relationship model and SQL-based implementation ensure data consistency, reliability, and integrity, supporting critical functions such as stock updates, order tracking, and report generation. It also highlights how digital inventory solutions can improve operational efficiency, reduce costs, and enhance customer satisfaction.

Chapter 8

Reference

8.1 linking

Munro, O. (2024). *What is an Inventory Management System? [Complete Guide]*. [online]

Unleashedsoftware.com. Available at:

<https://www.unleashedsoftware.com/inventory-management-guide/inventory-management-systems/>.

Hayes, A. (2024). *Inventory Management Defined, Plus Methods and Techniques*. [online]

Investopedia. Available at:

<https://www.investopedia.com/terms/i/inventory-management.asp>.

Inventory Management System (2025). *Inventory Management System*. [online] Inventory Management System. Available at:

<https://marketplace.microsoft.com/en-us/product/office/wa200006100?tab=overview>

[Accessed 11 Nov. 2025].

The End