INVENTORY MANAGEMENT SYSTEM

A PROJECT REPORT
Submitted by

SANYOG DANI [RA2211031010087]

ARUSH SIROTIYA [RA2211031010092]

NIKHIL KUMAR [RA2211031010097]

Under the guidance of

Dayana. D. S

Assistant Professor, Networking and Communications

In partial fulfilment of the requirements of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING W/S IN INFORMATION TECHNOLOGY

DEPARTMENT OF NETWORKING AND COMMUNICATIONS
COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR - 603203

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no. RA2211031010087, RA2211031010092, RA2211031010097 Certified to be the bonafide work done by **Sanyog Dani, Arush Sirotiya, Nikhil Kumar** of II Year / IV Sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date:03.05.2024

Faculty in Charge:
Dayana. D. S,
Assistant Professor,
Department of Networking and Communications,
SRMIST-KTR.

Head of the Department:
Dr.AnnapuraniPanaiyappan K, Professor and Head
Department of Networking and Communications,
SRMIST-KTR.

ABSTRACT

The Inventory Management System (IMS) is a pivotal tool for organizations seeking efficient management of their inventory operations. IMS offers a comprehensive solution designed to streamline processes, maximize productivity, and enhance overall operational efficiency. At its core, IMS provides a centralized platform encompassing inventory control, procurement, and monitoring functionalities.

One of the key features of IMS is its robust inventory tracking capabilities. By offering real-time visibility into stock levels, locations, and movement, IMS enables organizations to maintain accurate and up-to-date inventory records. This visibility not only facilitates efficient inventory management but also helps in minimizing stockouts and overstock situations, thereby optimizing inventory levels.

IMS allows users to swiftly record incoming and outgoing inventory transactions. This feature not only minimizes errors but also enhances accuracy in inventory management processes. By automating transaction recording, IMS reduces manual effort, mitigates the risk of human error, and ensures data integrity throughout the inventory lifecycle.

In addition to transaction recording, IMS provides advanced reporting and analytics functionalities. Stakeholders can leverage these features to gain valuable insights into inventory performance, identify trends, and make data-driven decisions. From analyzing inventory turnover rates to evaluating supplier performance metrics, IMS empowers organizations to optimize inventory strategies and reduce carrying costs.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Inventory management is a critical aspect of operations for businesses across diverse industries. An Inventory Management System (IMS) serves as a pivotal tool in facilitating the efficient handling, monitoring, and optimization of inventory levels within an organization.

The primary objective of an IMS is to ensure that businesses maintain optimal stock levels while minimizing costs associated with excess inventory or stockouts. By providing real-time visibility into inventory data, such as stock quantities, locations, and movement, an IMS enables businesses to make informed decisions regarding procurement, storage, and distribution.

Traditionally, inventory management involved manual processes, such as spreadsheet-based tracking or physical counts. However, the complexity and scale of modern supply chains necessitate more sophisticated solutions.

Key features of an Inventory Management System typically include:

Inventory Tracking: Real-time monitoring of stock levels, allowing businesses to accurately assess inventory positions and respond to fluctuations in demand.

Warehouse Management: Efficient management of warehouse operations, including inventory receipt, storage, picking, packing, and shipping, to streamline logistics processes and minimize errors.

Reporting and Analytics: Comprehensive reporting and analytics capabilities to track key performance indicators analyze inventory trends, and identify opportunities for improvement.

In today's competitive business landscape, effective inventory management is essential for meeting customer demand, optimizing cash flow, and maintaining a competitive edge. An Inventory Management System plays a pivotal role in helping businesses achieve these objectives by providing visibility, control, and efficiency throughout the inventory lifecycle.

Overall, the adoption of an Inventory Management System empowers businesses to enhance operational efficiency, minimize costs, and improve customer satisfaction, ultimately driving sustainable growth and success.

PROBLEM STATEMENT

In today's dynamic business environment, effective inventory management is crucial for businesses to meet customer demand, minimize costs, and maintain a competitive edge. However, many organizations face challenges in efficiently tracking, managing, and optimizing their inventory operations. Manual processes, outdated systems, and lack of real-time visibility often result in inventory inaccuracies, stockouts, excess inventory, and inefficiencies in procurement and logistics.

The problem statement revolves around the need for a comprehensive Inventory Management System (IMS) that addresses the following challenges:

- Inventory Tracking and Visibility
- Inventory Optimization
- Procurement Efficiency
- Warehouse Management
- Reporting and Analytics

LITERATURE SURVEY

1. Introduction

- Define the significance of inventory management in business operations.
- Introduce the purpose of the literature survey, focusing on exploring existing research and methodologies related to implementing inventory management systems using MySQL and PHP.

2. Historical Overview

- Trace the historical development of inventory management systems and the evolution of database technologies like MySQL.
- Highlight key milestones and advancements in PHP development relevant to inventory management.

3. Theoretical Framework

- Discuss theoretical concepts in inventory management that can be implemented using MySQL and PHP, such as database normalization for efficient data storage and retrieval.
- Explore how PHP scripting language can be used to develop dynamic inventory management interfaces and automate inventory-related tasks.

4. Types of Inventory Management Systems

- Review different types of inventory management systems and how they can be implemented using MySQL databases and PHP scripts.
- Compare various inventory management methodologies, such as FIFO, LIFO, and JIT, in the context of MySQL and PHP implementation.

5. Technological Advances

- Explore technological advancements in MySQL and PHP that enhance inventory management capabilities, such as transaction handling, stored procedures, and data encryption for security.
- Discuss the integration of web-based interfaces and mobile applications using PHP frameworks like Laravel or CodeIgniter for real-time inventory tracking and management.

6. Supply Chain Integration

- Investigate methods for integrating inventory management systems with other components of the supply chain using MySQL and PHP.
- Discuss the use of APIs and web services to facilitate data exchange between inventory systems, suppliers, and distributors.

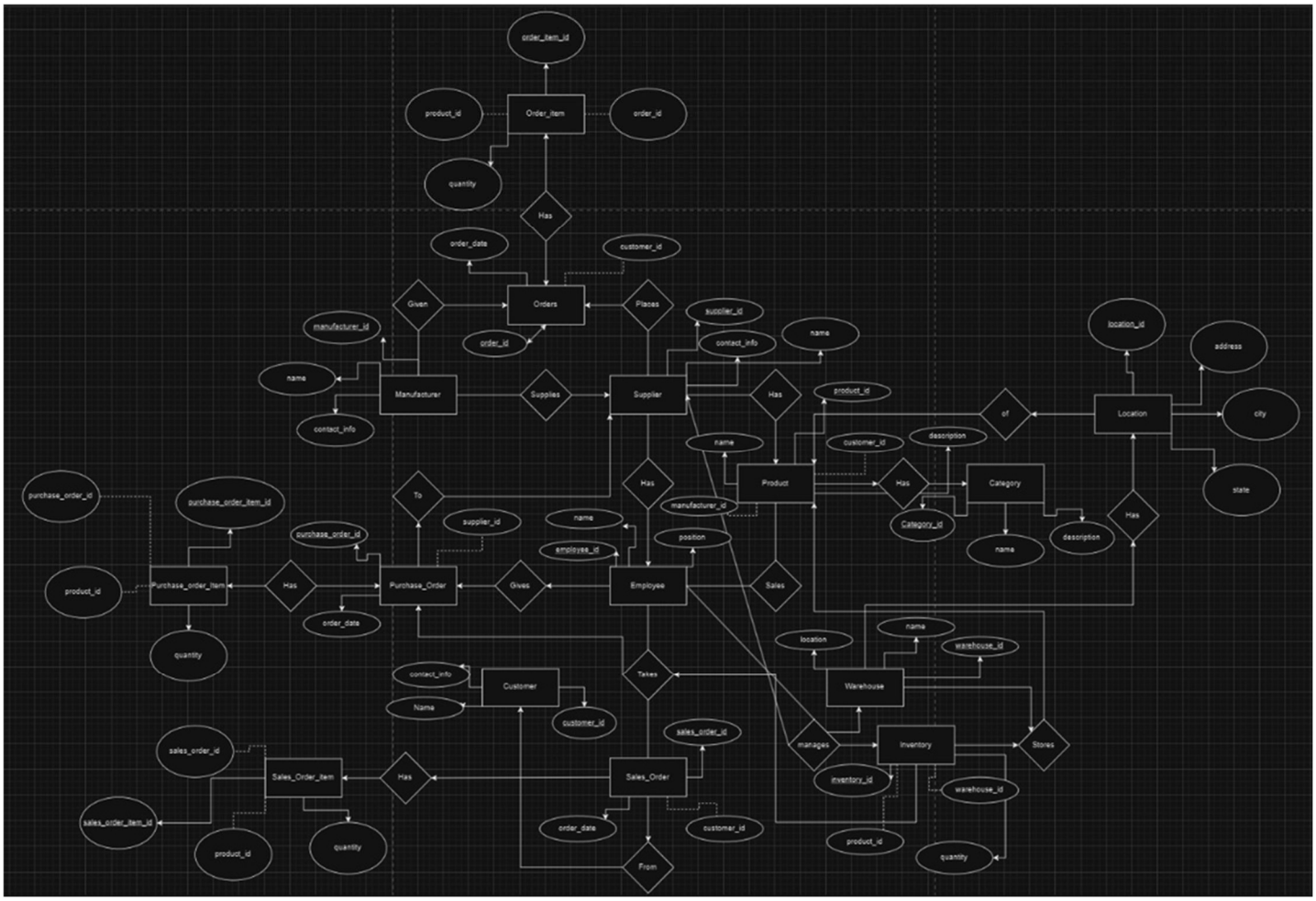
7. Challenges and Solutions

- Identify common challenges in implementing inventory management systems using MySQL and PHP, such as scalability, data consistency, and security.
- Discuss best practices and solutions to overcome these challenges, including database optimization techniques and PHP security measures.

8. Case Studies and Empirical Research

- Review case studies and empirical research on the implementation of inventory management systems using MySQL and PHP.
- Analyze the effectiveness of different approaches and methodologies in real-world scenarios.

ER DIAGRAM


order_item_id
product_id
Order_item
order_id
quantity
Has
order_date
customer_id
Given
Orders
Places
manufacturer_id
order_id
supplier_id
contact_info
name
name
Manufacturer
Supplies
Supplier
Has
product_id
location_id
address
of
Location
city
contact_info
name
customer_id
description
state
purchase_order_id
purchase_order_item_id
To
Has
Product
Has
Category
manufacturer_id
supplier_id
name
Has
purchase_order_id
employee_id
position
Category_id
name
description
product_id
Purchase_order_Item
Has
Purchase_Order
Gives
Employee
Sales
order_date
quantity
location
name
warehouse_id
Takes
contact_info
Customer
Warehouse
Name
customer_id
sales_order_id
manages
Inventory
Stores
sales_order_id
Sales_Order_Item
Has
Sales_Order
inventory_id
warehouse_id
sales_order_item_id
order_date
customer_id
product_id
quantity
product_id
quantity
From

CHAPTER 2

RELATIONAL SCHEMA

1. ims_brand

id (Primary Key) ,categoryid (ForeignKey referencing ims_ category.categoryid) , bname , status

2. ims_category

categoryid (Primary Key), name , status

3. ims_customer

id (Primary Key) , name , address , mobile , balance

4. ims_order

order_id (Primary Key) , product_id (Foreign Key referencing ims_product.pid) , total_shipped , customer_id (Foreign Key referencing ims_customer.id) , order_date

5. ims_product

pid (Primary Key) category , id (Foreign Key referencing ims_category.category) , id brand , id (Foreign Key referencing ims_brand.id), pname model , base_price , tax , minimum_order , supplier , status , date

6. ims_purchase

purchase_id (Primary Key) , supplier_id (Foreign Key referencing ims_supplier.supplier_id) , product_id (Foreign Key referencing ims_product.pid), quantity purchase_date

7. ims_supplier,

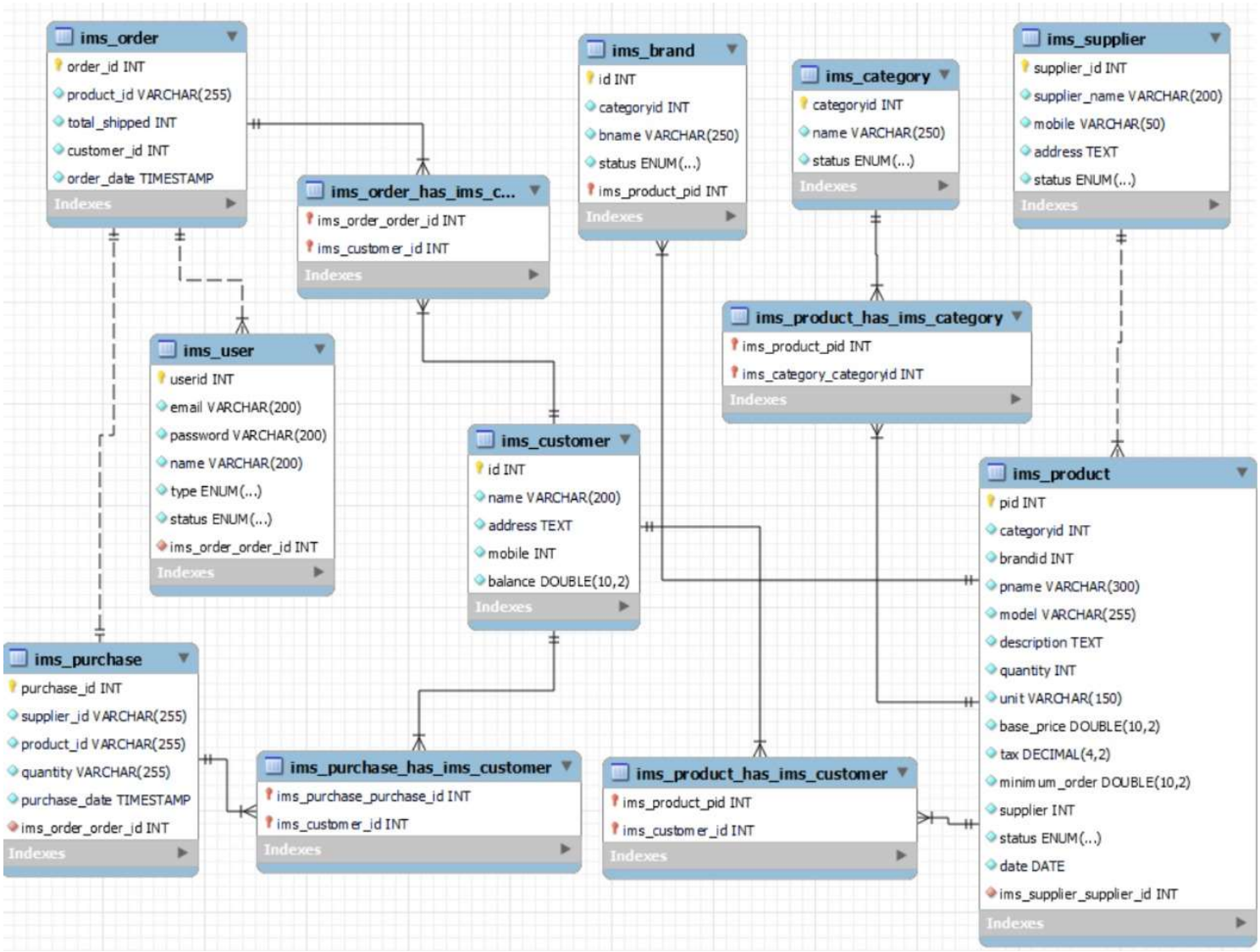
supplier_id (Primary Key), supplier_name, mobile, address, status

8. ims_user

userid (Primary Key), email, password, name, type, status

Foreign keys in tables `ims_brand`, `ims_order`, `ims_product`, `ims_purchase` reference primary keys in other tables, establishing relationships between them.

Each table represents a specific entity and its attributes. Relationships between entities are established through foreign key constraints, ensuring data integrity and consistency.


ims_order
order_id INT
product_id VARCHAR(255)
total_shipped INT
customer_id INT
order_date TIMESTAMP
Indexes
ims_order_has_ims_c...
ims_order_order_id INT
ims_customer_id INT
Indexes
ims_brand
id INT
categoryid INT
bname VARCHAR(250)
status ENUM(...)
ims_product_pid INT
Indexes
ims_category
categoryid INT
name VARCHAR(250)
status ENUM(...)
Indexes
ims_supplier
supplier_id INT
supplier_name VARCHAR(200)
mobile VARCHAR(50)
address TEXT
status ENUM(...)
Indexes
ims_product_has_ims_category
ims_product_pid INT
ims_category_categoryid INT
Indexes
ims_user
userid INT
email VARCHAR(200)
password VARCHAR(200)
name VARCHAR(200)
type ENUM(...)
status ENUM(...)
ims_order_order_id INT
Indexes
ims_customer
id INT
name VARCHAR(200)
address TEXT
mobile INT
balance DOUBLE(10,2)
Indexes
ims_product
pid INT
categoryid INT
brandid INT
pname VARCHAR(300)
model VARCHAR(255)
description TEXT
quantity INT
unit VARCHAR(150)
base_price DOUBLE(10,2)
tax DECIMAL(4,2)
minimum_order DOUBLE(10,2)
supplier INT
status ENUM(...)
date DATE
ims_supplier_supplier_id INT
Indexes
ims_purchase
purchase_id INT
supplier_id VARCHAR(255)
product_id VARCHAR(255)
quantity VARCHAR(255)
purchase_date TIMESTAMP
ims_order_order_id INT
Indexes
ims_purchase_has_ims_customer
ims_purchase_purchase_id INT
ims_customer_id INT
Indexes
ims_product_has_ims_customer
ims_product_pid INT
ims_customer_id INT
Indexes

CHAPTER 3
TABLES

- Database

```
+------------------+
| Tables_in_ims_db |
+------------------+
| employee         |
| ims_brand        |
| ims_category     |
| ims_customer     |
| ims_order        |
| ims_product      |
| ims_purchase     |
| ims_supplier     |
| ims_user         |
| inventory        |
| location         |
| order_item       |
| product_details  |
| purchase_order   |
| warehouse        |
+------------------+
15 rows in set (0.00 sec)
```

- Tables

```
MariaDB [ims_db]> select * from ims_brand;
+----+------------+-------------+--------+
| id | categoryid | bname       | status |
+----+------------+-------------+--------+
|  1 |          2 | Oneplus     | active |
|  2 |          2 | Tony        | active |
|  3 |          2 | Panasonic   | active |
|  4 |          1 | Nothing     | active |
|  5 |          1 | Ramsung     | active |
|  6 |          1 | Nokia       | active |
|  7 |          5 | Asus        | active |
|  8 |          5 | HP          | active |
|  9 |          3 | Skull Candy | active |
| 10 |          3 | JBL         | active |
| 11 |          6 | HP          | active |
| 12 |          4 | ALLWYN      | active |
| 13 |          4 | CASIO       | active |
| 14 |          7 | LG          | active |
| 15 |          7 | GODREJ      | active |
| 16 |          8 | SAMSUNG     | active |
+----+------------+-------------+--------+
16 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_category;
+------------+-----------------+--------+
| categoryid | name            | status |
+------------+-----------------+--------+
|          1 | Smartphone      | active |
|          2 | TV              | active |
|          3 | Speaker         | active |
|          4 | Watch           | active |
|          5 | Laptop          | active |
|          6 | Printer         | active |
|          7 | Washing Machine | active |
|          8 | Refrigerator    | active |
+------------+-----------------+--------+
8 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_order;
+----------+------------+---------------+-------------+---------------------+
| order_id | product_id | total_shipped | customer_id | order_date          |
+----------+------------+---------------+-------------+---------------------+
|        1 | 1          |             5 |           1 | 2022-06-20 13:50:40 |
|        2 | 2          |             2 |           2 | 2022-06-20 13:50:48 |
|        3 | 1          |             1 |           3 | 2024-05-03 13:54:39 |
|        4 | 3          |             4 |           5 | 2024-05-03 13:54:47 |
|        5 | 5          |             1 |           3 | 2024-05-03 13:54:59 |
+----------+------------+---------------+-------------+---------------------+
5 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_customer;
+----+---------------+---------------------------+------------+----------+
| id | name          | address                   | mobile     | balance  |
+----+---------------+---------------------------+------------+----------+
|  2 | Nikhil Kumar  | Vill+Post Deoghar Jharkhand | 2147483647 | 35000.00 |
|  3 | Sanyog Dani   | Twins City Durg-Bhilai CG   | 2147483647 |  9669.00 |
|  5 | Arush Sirotiya | Nagpur MH                  |  357698798 | 23456.00 |
+----+---------------+---------------------------+------------+----------+
3 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_product;
+-----+------------+---------+-------------+--------+-------------------------+----------+------+------------+------+-------------+----------+--------+------------+
| pid | categoryid | brandid | pname       | model  | description             | quantity | unit | base_price | tax  | minimum_order | supplier | status | date       |
+-----+------------+---------+-------------+--------+-------------------------+----------+------+------------+------+-------------+----------+--------+------------+
|   1 |          5 |       7 | ROG STRIX   | G15    | GAMING BEAST RGB        |        4 | Nos  |  114000.00 | 18.00 |        1.00 |        1 | active | 0000-00-00 |
|   2 |          1 |       5 | Ramsung n54 | 6157   | GOOD PHONE VERY GOOD    |       17 | Nos  |   50000.00 | 18.00 |        1.00 |        1 | active | 0000-00-00 |
|   3 |          4 |      12 | WRIST WATCH | P-8767 | TELLS TIME VINTAGE WATCH |       20 | Nos  |   10000.00 | 18.00 |        1.00 |        2 | active | 0000-00-00 |
|   4 |          2 |       2 | Tony Tv Q98 | 13456  | 65 INCH TV              |        6 | Nos  |   88000.00 | 18.00 |        1.00 |        4 | active | 0000-00-00 |
|   5 |          3 |      10 | JBL Lords 66 | P98756 | VERY LOUD               |       13 | Nos  |   12000.00 | 18.00 |        1.00 |        3 | active | 0000-00-00 |
+-----+------------+---------+-------------+--------+-------------------------+----------+------+------------+------+-------------+----------+--------+------------+
5 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_supplier;
+-------------+-----------------+--------------+----------------------+--------+
| supplier_id | supplier_name   | mobile       | address              | status |
+-------------+-----------------+--------------+----------------------+--------+
|           1 | JETHALAL        | 09645987123  | GOKULDHAM SOCIETY    | active |
|           2 | CHHEDA          | 094568791252 | POWDER GALI MUMBAI
 | active |
|           3 | SUKESH          | 978989787    | SAROJNI NAGAR DELHI  | active |
|           4 | SUKHVINDAR SINGH | 98764667889 | BRAMPTON CANADA      | active |
+-------------+-----------------+--------------+----------------------+--------+
4 rows in set (0.000 sec)
```

```
MariaDB [ims_db]> select * from ims_purchase;
+-------------+-------------+------------+----------+---------------------+
| purchase_id | supplier_id | product_id | quantity | purchase_date       |
+-------------+-------------+------------+----------+---------------------+
|           1 | 1           | 1          | 25       | 2022-06-20 13:50:07 |
|           2 | 2           | 2          | 35       | 2022-06-20 13:50:14 |
|           3 | 3           | 3          | 10       | 2022-06-20 13:50:29 |
+-------------+-------------+------------+----------+---------------------+
3 rows in set (0.001 sec)
```

```
MariaDB [ims_db]> select * from ims_user;
+--------+----------------+----------------------------------+---------------+-------+--------+
| userid | email          | password                         | name          | type  | status |
+--------+----------------+----------------------------------+---------------+-------+--------+
|      1 | admin@mail.com | 0192023a7bbd73250516f069df18b500 | Administrator | admin | Active |
+--------+----------------+----------------------------------+---------------+-------+--------+
1 row in set (0.001 sec)
```

warehouse_id	name	location
1	Warehouse A	City X, State Y
2	Warehouse B	City Y, State Z
3	Warehouse C	City Z, State X
4	Warehouse D	City A, State B
5	Warehouse E	City B, State C
6	Warehouse F	City C, State A
7	Warehouse G	City D, State E
8	Warehouse H	City E, State F
9	Warehouse I	City F, State D
10	Warehouse J	City G, State H

10 rows in set (0.00 sec)

purchase_order_id	order_date	supplier_id
1	2024-04-01	1
2	2024-04-02	2
3	2024-04-03	3
4	2024-04-04	4
5	2024-04-05	5
6	2024-04-06	6
7	2024-04-07	7
8	2024-04-08	8
9	2024-04-09	9
10	2024-04-10	10

10 rows in set (0.00 sec)

employee_id	name	position
1	Alice Johnson	Manager
2	Bob Smith	Sales Associate
3	Catherine Brown	Warehouse Supervisor
4	Daniel Davis	Customer Service Representative
5	Emma Wilson	Accountant
6	Frank Harris	Delivery Driver
7	Grace Taylor	Human Resources Manager
8	Henry Clark	IT Specialist
9	Isabella Young	Marketing Coordinator
10	Jack Martinez	Inventory Manager

10 rows in set (0.00 sec)

inventory_id	product_id	warehouse_id	quantity
1	1	1	80
2	2	2	190
3	3	3	142
4	4	4	77
5	5	5	114
6	6	6	86
7	7	7	285
8	8	8	238
9	9	9	68
10	10	10	173

10 rows in set (0.00 sec)

location_id	address	city	state
1	123 Main Street	City X	State Y
2	456 Center Avenue	City Y	State Z
3	789 Regional Road	City Z	State X
4	101 Central Street	City A	State B
5	321 Metro Avenue	City B	State C
6	543 Urban Lane	City C	State A
7	765 Rural Road	City D	State E
8	987 Suburban Street	City E	State F
9	234 Coastal Avenue	City F	State D
10	876 Inland Lane	City G	State H
11	123 Main Street	City X	State Y
12	456 Center Avenue	City Y	State Z
13	789 Regional Road	City Z	State X
14	101 Central Street	City A	State B
15	321 Metro Avenue	City B	State C
16	543 Urban Lane	City C	State A
17	765 Rural Road	City D	State E
18	987 Suburban Street	City E	State F
19	234 Coastal Avenue	City F	State D
20	876 Inland Lane	City G	State H

20 rows in set (0.00 sec)

```
mysql> select *from product_details;
```

product_id	product_name	description	category_name
1	Smartphone	Smartphone with latest features	Electronics
2	T-shirt	Cotton T-shirt for everyday wear	Clothing
3	Novel	Bestselling novel by famous author	Books
4	Refrigerator	Energy-efficient refrigerator	Appliances
5	Soccer Ball	Official size and weight soccer ball	Sports
6	Office Chair	Ergonomic office chair with lumbar support	Furniture
7	Action Figure	Popular action figure toy for kids	Toys
8	Shampoo	Moisturizing shampoo for all hair types	Beauty
9	Car Battery	Long-lasting battery for various vehicles	Automotive
10	Drill	Cordless drill with multiple drill bits	Tools

10 rows in set (0.00 sec)

order_item_id	order_id	product_id	quantity
1	1	1	5
2	2	2	10
3	3	3	8
4	4	4	3
5	5	5	6
6	6	6	4
7	7	7	15
8	8	8	12
9	9	9	2
10	10	10	7

10 rows in set (0.00 sec)

CONSTRAINTS

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The constraints used in the IMS project is:

- **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
- **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as foreign key.
- **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

```
CREATE TABLE `ims_brand` (
  `id` int(11) NOT NULL,
  `categoryid` int(11) NOT NULL,
  `bname` varchar(250) NOT NULL,
  `status` enum('active','inactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

QUERIES

SQL query is used to query or retrieve information from the databases. These queries can be used to create a new database and insert data into the database, to retrieve (or fetch) data from the database, to modify or update the existing data in the database, to delete or drop the data or table from the database, to set permissions for the tables, views and procedures.

SUB QUERIES

Subqueries (also known as inner queries or nested queries) are a tool for performing operations in multiple steps. An SQL Subquery, is a SELECT query within another query. It is also known as Inner query or Nested query and the query containing it is the outer query.

The outer query can contain the SELECT, INSERT, UPDATE, and DELETE statements. We can use the subquery as a column expression, as a condition in SQL clauses, and with operators like =, >, <, >=, <=, IN, BETWEEN, etc.

JOINS

A **join** clause in the Structured Query Language (SQL) combines columns from one or more tables into a new table. The operation corresponds to a join operation in relational algebra. Informally, a join stitches two tables and puts on the same row records with matching fields. There are 5 types of joins, namely:

```
CREATE VIEW product_details AS
SELECT p.product_id, p.name AS product_name, p.description, c.name AS category_name, m.name AS manufacturer_name
FROM product p
LEFT JOIN category c ON p.category_id = c.category_id
LEFT JOIN manufacturer m ON p.manufacturer_id = m.manufacturer_id;
```

VIEWS

Views in SQL are a kind of virtual table. A view also has rows and columns like tables, but a view doesn't store data on the disk like a table. View defines a customized query that retrieves data from one or more tables, and represents the data as if it was coming from a single source. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain conditions.

```
CREATE VIEW product_details AS
SELECT p.product_id, p.name AS product_name, p.description, c.name AS category_name, m.name AS manufacturer_name
FROM product p
LEFT JOIN category c ON p.category_id = c.category_id
LEFT JOIN manufacturer m ON p.manufacturer_id = m.manufacturer_id;
```

TRIGGERS

A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. In simple words, a trigger is a collection of SQL statements with particular names that are stored in system memory. It belongs to a specific class of stored procedures that are automatically invoked in response to database server events. Every trigger has a table attached to it.

```
DELIMITER $$
CREATE TRIGGER update_inventory_after_sales_order
AFTER INSERT ON sales_order_item
FOR EACH ROW
BEGIN
  UPDATE inventory
  SET quantity = quantity - NEW.quantity
  WHERE product_id = NEW.product_id;
END$$
DELIMITER ;
```

CURSORS

A cursor in SQL Server is a database object that allows us to retrieve each row at a time and manipulate its data**.** A cursor is nothing more than a pointer to a row. It's always used in conjunction with a SELECT statement. It is usually a collection of SQL logic that loops through a predetermined number of rows one by one.

```
DELIMITER $$
CREATE PROCEDURE total_sales_by_employee (IN employee_id INT)
BEGIN
  SELECT SUM(oi.quantity)
  FROM orders o
  JOIN order_item oi ON o.order_id = oi.order_id
  WHERE o.employee_id = employee_id;
END$$
DELIMITER ;
```

CHAPTER 4

ANALYZING PITFALLS

PITFALLS:

- Security Vulnerabilities: Inadequate access controls, encryption, or other security measures can lead to unauthorized access, data breaches, or malicious activities. Example: Weak password policies, lack of encryption for sensitive data like passwords or personal information, and insufficient protection against SQL injection attacks.

- Scalability Issues: The database may struggle to handle increased load, data volume, or user concurrency as the application grows. Example: Poorly optimized queries, lack of indexing, and inefficient database schema design can result in performance degradation and scalability limitations.

- Insufficient Maintenance and Updates: Neglecting regular maintenance tasks such as database backups, software updates, and performance optimizations can lead to system instability, data loss, or security vulnerabilities. Example: Failure to regularly update database software and security patches, infrequent backups, and inadequate monitoring of database performance and health.

- Lack of Analytics and Insights: Without proper analytics capabilities, the organization may miss opportunities for data-driven decision-making, performance optimization, and identification of trends or patterns. Example: Absence of tools or processes for analyzing database usage, query performance, user behavior, or business metrics, leading to missed insights and potential inefficiencies.

FUNCTIONAL DEPENDENCIES

- In the "Brand" table, attributes like "brand_name" and "status" are functionally dependent on the "brand_id". This means each brand ID uniquely determines its corresponding name and status, ensuring consistency and integrity in brand records.

- Similarly, in the "Category" table, "category_name" and "status" depend functionally on the "category_id". Each category ID uniquely determines its name and status, maintaining coherence in category information.

- Moving to the "Customer" table, attributes such as "customer_name", "address", "mobile", and "balance" depend on the "customer_id". This dependency ensures each customer ID uniquely corresponds to its name, address, mobile number, and balance, maintaining accurate customer records.

- In the "Order" table, "customer_id" and "order_date" attributes are functionally dependent on the "order_id". Each order ID uniquely determines the customer ID and order date, facilitating efficient order tracking and management.

- In the "Product" table, attributes like "product_name", "quantity", "unit", "base_price", and "supplier_id" depend on the "product_id". Each product ID uniquely determines these attributes, ensuring accurate representation and identification of products within the system.

- Overall, functional dependencies ensure data integrity and consistency in the inventory management system by establishing clear relationships between primary keys and other attributes, enhancing system reliability and effectiveness.

NORMALIZATION

Victim table before normalization:

```
mysql> select *from location;
+-------------+----------------------+---------+----------+------------------+
| location_id | address              | city    | state    | orders_order_id  |
+-------------+----------------------+---------+----------+------------------+
|           1 | 123 Main Street      | City X  | State Y  |                1 |
|           2 | 456 Center Avenue    | City Y  | State Z  |                2 |
|           3 | 789 Regional Road    | City Z  | State X  |                3 |
|           4 | 101 Central Street   | City A  | State B  |                4 |
|           5 | 321 Metro Avenue     | City B  | State C  |                5 |
|           6 | 543 Urban Lane       | City C  | State A  |                6 |
|           7 | 765 Rural Road       | City D  | State E  |                7 |
|           8 | 987 Suburban Street  | City E  | State F  |                8 |
|           9 | 234 Coastal Avenue   | City F  | State D  |                9 |
|          10 | 876 Inland Lane      | City G  | State H  |               10 |
|          11 | 123 Main Street      | City X  | State Y  |                1 |
|          12 | 456 Center Avenue    | City Y  | State Z  |                2 |
|          13 | 789 Regional Road    | City Z  | State X  |                3 |
|          14 | 101 Central Street   | City A  | State B  |                4 |
|          15 | 321 Metro Avenue     | City B  | State C  |                5 |
|          16 | 543 Urban Lane       | City C  | State A  |                6 |
|          17 | 765 Rural Road       | City D  | State E  |                7 |
|          18 | 987 Suburban Street  | City E  | State F  |                8 |
|          19 | 234 Coastal Avenue   | City F  | State D  |                9 |
|          20 | 876 Inland Lane      | City G  | State H  |               10 |
+-------------+----------------------+---------+----------+------------------+
20 rows in set (0.00 sec)
```

After applying normalization:

```
mysql> select *from locations;
+-------------+------------+----------+
| location_id | address_id | order_id |
+-------------+------------+----------+
|           1 |          1 |        1 |
|           2 |          2 |        2 |
|           3 |          3 |        3 |
|           4 |          4 |        4 |
|           5 |          5 |        5 |
|           6 |          6 |        6 |
|           7 |          7 |        7 |
|           8 |          8 |        8 |
|           9 |          9 |        9 |
|          10 |         10 |       10 |
|          11 |          1 |        1 |
|          12 |          2 |        2 |
|          13 |          3 |        3 |
|          14 |          4 |        4 |
|          15 |          5 |        5 |
|          16 |          6 |        6 |
|          17 |          7 |        7 |
|          18 |          8 |        8 |
|          19 |          9 |        9 |
|          20 |         10 |       10 |
+-------------+------------+----------+
20 rows in set (0.00 sec)
```

```
mysql> select *from address_details;
+------------+----------------------+---------+----------+
| address_id | address              | city    | state    |
+------------+----------------------+---------+----------+
|          1 | 123 Main Street      | City X  | State Y  |
|          2 | 456 Center Avenue    | City Y  | State Z  |
|          3 | 789 Regional Road    | City Z  | State X  |
|          4 | 101 Central Street   | City A  | State B  |
|          5 | 321 Metro Avenue     | City B  | State C  |
|          6 | 543 Urban Lane       | City C  | State A  |
|          7 | 765 Rural Road       | City D  | State E  |
|          8 | 987 Suburban Street  | City E  | State F  |
|          9 | 234 Coastal Avenue   | City F  | State D  |
|         10 | 876 Inland Lane      | City G  | State H  |
+------------+----------------------+---------+----------+
10 rows in set (0.00 sec)
```

```
mysql> select *from orders1;
+----------+------------+-------------+
| order_id | order_date | customer_id |
+----------+------------+-------------+
|        1 | 2024-04-17 |         101 |
|        2 | 2024-04-18 |         102 |
|        3 | 2024-04-19 |         103 |
|        4 | 2024-04-20 |         104 |
|        5 | 2024-04-21 |         105 |
|        6 | 2024-04-22 |         106 |
|        7 | 2024-04-23 |         107 |
|        8 | 2024-04-24 |         108 |
|        9 | 2024-04-25 |         109 |
|       10 | 2024-04-26 |         110 |
+----------+------------+-------------+
10 rows in set (0.00 sec)
```

CHAPTER 5

Implementation of concurrency control and recovery mechanisms

MODULES USED:

1. Database Setup:

The Inventory Management System (IMS) is implemented using PHP and MySQL. To begin with, a MySQL database named "ims_db" is created using phpMyAdmin, which is part of the XAMPP stack. XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends. It consists mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in PHP .

2. XAMPP Control Panel:

XAMPP provides a user-friendly control panel that allows easy management of the Apache server, MySQL database. It can be accessed by starting XAMPP and navigating to http://localhost/phpmyadmin/ to access phpMyAdmin for database management.

3. Table Structure: The database structure consists of several tables to store various aspects of inventory management:

- Home Table: This table stores the home page content of the system, including banners, featured products, and promotions.
- Customer Table: Information about customers is stored in this table, including customer ID, name, contact information, and address.
- Category Table: Categories for products are stored in this table, including category ID and name.
- Brand Table: Brands for products are stored in this table, including brand ID and name.
- Product Table: This table stores information about the products in the inventory, including product ID, name, description, quantity, price, category ID, brand ID, and supplier ID.
- Supplier Table: Information about suppliers is stored in this table, including supplier ID, name, contact information, and address.
- Purchase Table: All purchase transactions are recorded in this table, including purchase ID, product ID, quantity, purchase date, and supplier ID.
- Order Table: All sales orders are recorded in this table, including order ID, customer ID, product ID, quantity, and order date.

4. PHP Implementation:

The Inventory Management System is implemented using PHP for server-side scripting. PHP scripts interact with the MySQL database to perform various functions such as adding, updating, and deleting products, managing suppliers, recording transactions, managing orders, and user authentication.

6. Command Prompt Usage:

- Open XAMPP Control Panel.
- Start the Apache and MySQL services.
- Navigate to http://localhost/phpmyadmin/ to access phpMyAdmin for database management.
- Open a web browser and navigate to the directory where the PHP files are located.
- Access the Inventory Management System by typing http://localhost/ in the browser.

5. MySQL Database Connectivity:

PHP scripts establish a connection to the MySQL database using the MySQL or PDO extension. The connection details such as server name, username, password are specified in the PHP scripts.

7. Concurrency Module:

To handle concurrency issues such as simultaneous access to the database by multiple users, the IMS incorporates a concurrency control module. This module ensures data consistency and prevents anomalies such as lost updates, uncommitted data, and inconsistent reads.

The Inventory Management System implemented using PHP and MySQL, with the support of XAMPP, provides a solution for efficient inventory management. By leveraging PHP for server-side scripting, MySQL for database management, and XAMPP for local server environment setup, the system ensures reliable performance, scalability, and security. With XAMPP's user-friendly control panel, managing the server environment and database becomes seamless, making it an ideal solution for local development and testing. The concurrency module ensures data consistency and prevents anomalies, allowing multiple users to access the system simultaneously without risking data integrity. Together, these technologies enable organizations to streamline inventory processes, maximize productivity, and enhance overall operational efficiency.

CHAPTER 6
CODE OF THE PROJECT

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

--
-- Database: `ims_db`
--

-- --------------------------------------------------------

--
-- Table structure for table `ims_brand`
--

CREATE TABLE `ims_brand` (
  `id` int(11) NOT NULL,
  `categoryid` int(11) NOT NULL,
  `bname` varchar(250) NOT NULL,
  `status` enum('active','inactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_brand`
--

INSERT INTO `ims_brand` (`id`, `categoryid`, `bname`, `status`) VALUES
(1, 2, 'Brand 1', 'active'),
(2, 2, 'Brand 2', 'active'),
(3, 2, 'Brand 3', 'active'),
(4, 1, 'Brand 201', 'active'),
(5, 1, 'Brand 202', 'active'),
(6, 1, 'Brand 203', 'active'),
(7, 3, 'Brand 301', 'active'),
(8, 3, 'Brand 302', 'active'),
(9, 3, 'Brand 303', 'active');

-- --------------------------------------------------------

--
```

```sql
-- Table structure for table `ims_category`
--

CREATE TABLE `ims_category` (
  `categoryid` int(11) NOT NULL,
  `name` varchar(250) NOT NULL,
  `status` enum('active','inactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_category`
--

INSERT INTO `ims_category` (`categoryid`, `name`, `status`) VALUES
(1, 'Smartphone', 'active'),
(2, 'Random Item', 'active'),
(3, 'Speaker', 'active');

-- --------------------------------------------------------

--
-- Table structure for table `ims_customer`
--

CREATE TABLE `ims_customer` (
  `id` int(11) NOT NULL,
  `name` varchar(200) NOT NULL,
  `address` text NOT NULL,
  `mobile` int(50) NOT NULL,
  `balance` double(10,2) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_customer`
--

INSERT INTO `ims_customer` (`id`, `name`, `address`, `mobile`, `balance`) VALUES
(1, 'Mark Cooper', 'Sample Address', 2147483647, 25000.00),
(2, 'George Wilson', '2306 St, Here There', 2147483647, 35000.00);

-- --------------------------------------------------------

--
-- Table structure for table `ims_order`
--
```

```sql
CREATE TABLE `ims_order` (
  `order_id` int(11) NOT NULL,
  `product_id` varchar(255) NOT NULL,
  `total_shipped` int(11) NOT NULL,
  `customer_id` int(11) NOT NULL,
  `order_date` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_order`
--

INSERT INTO `ims_order` (`order_id`, `product_id`, `total_shipped`, `customer_id`,
`order_date`) VALUES
(1, '1', 5, 1, '2022-06-20 08:20:40'),
(2, '2', 3, 2, '2022-06-20 08:20:48');

-- --------------------------------------------------------

--
-- Table structure for table `ims_product`
--

CREATE TABLE `ims_product` (
  `pid` int(11) NOT NULL,
  `categoryid` int(11) NOT NULL,
  `brandid` int(11) NOT NULL,
  `pname` varchar(300) NOT NULL,
  `model` varchar(255) NOT NULL,
  `description` text NOT NULL,
  `quantity` int(11) NOT NULL,
  `unit` varchar(150) NOT NULL,
  `base_price` double(10,2) NOT NULL,
  `tax` decimal(4,2) NOT NULL,
  `minimum_order` double(10,2) NOT NULL,
  `supplier` int(11) NOT NULL,
  `status` enum('active','inactive') NOT NULL,
  `date` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_product`
--
```

```sql
INSERT INTO `ims_product` (`pid`, `categoryid`, `brandid`, `pname`, `model`, `description`,
`quantity`, `unit`, `base_price`, `tax`, `minimum_order`, `supplier`, `status`, `date`)
VALUES
(1, 2, 1, 'Product 101', 'P-1001', 'usce auctor faucibus efficitur.', 10, 'Bottles', 500.00,
'12.00', 1.00, 1, 'active', '0000-00-00'),
(2, 1, 4, 'Product 102', 'P-1002', 'Proin vehicula mi pulvinar ipsum ornare tincidunt.', 15,
'Box', 7500.00, '12.00', 1.00, 2, 'active', '0000-00-00'),
(3, 3, 7, 'Product 103', 'P-1003', 'Integer interdum, odio eget mattis venenatis', 20,
'Bags', 350.00, '12.00', 1.00, 3, 'active', '0000-00-00');

-- --------------------------------------------------------

--
-- Table structure for table `ims_purchase`
--

CREATE TABLE `ims_purchase` (
  `purchase_id` int(11) NOT NULL,
  `supplier_id` varchar(255) NOT NULL,
  `product_id` varchar(255) NOT NULL,
  `quantity` varchar(255) NOT NULL,
  `purchase_date` timestamp NOT NULL DEFAULT current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_purchase`
--

INSERT INTO `ims_purchase` (`purchase_id`, `supplier_id`, `product_id`, `quantity`,
`purchase_date`) VALUES
(1, '1', '1', '25', '2022-06-20 08:20:07'),
(2, '2', '2', '35', '2022-06-20 08:20:14'),
(3, '3', '3', '10', '2022-06-20 08:20:29');

-- --------------------------------------------------------
--
-- Table structure for table `ims_supplier`
--
CREATE TABLE `ims_supplier` (
  `supplier_id` int(11) NOT NULL,
  `supplier_name` varchar(200) NOT NULL,
  `mobile` varchar(50) NOT NULL,
  `address` text NOT NULL,
  `status` enum('active','inactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

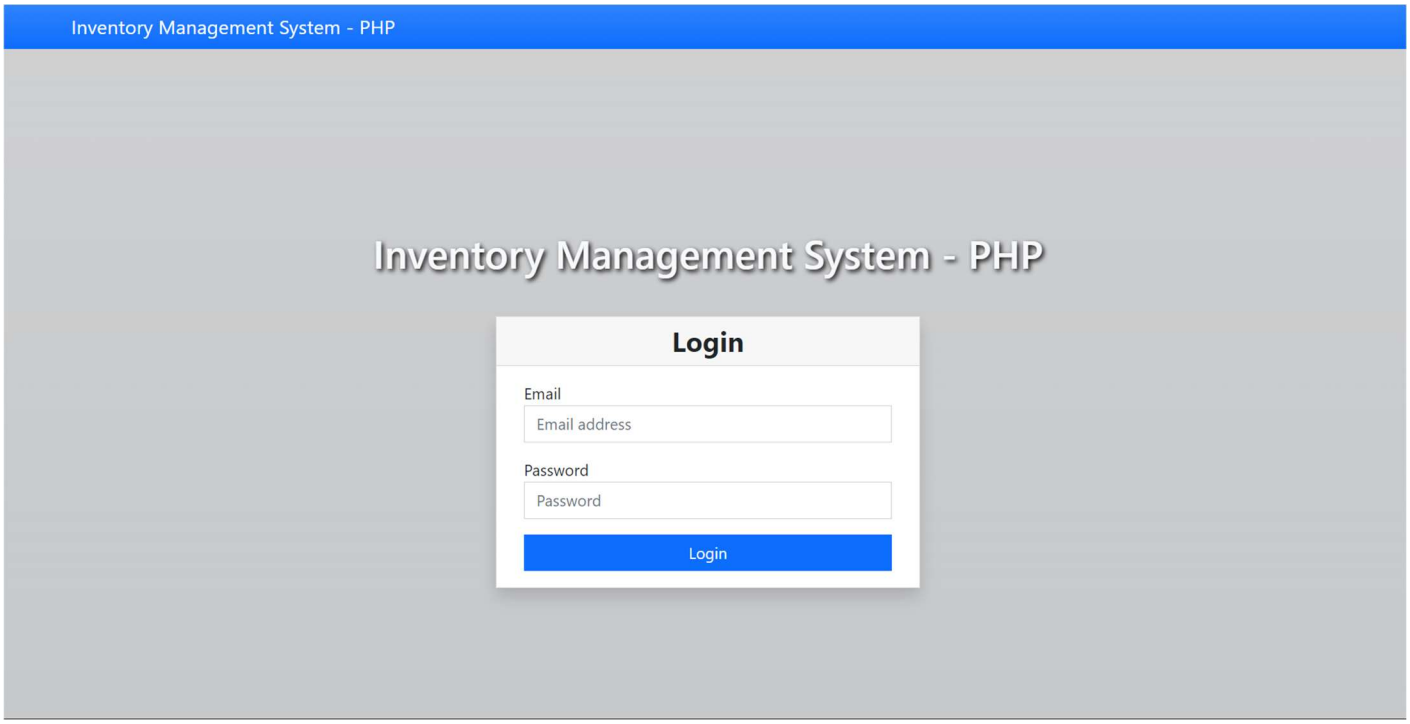
```sql
--
-- Dumping data for table `ims_supplier`
--

INSERT INTO `ims_supplier` (`supplier_id`, `supplier_name`, `mobile`, `address`, `status`)
VALUES
(1, 'Supplier 101', '09645987123', 'Over Here', 'active'),
(2, 'Supplier 102', '094568791252', 'Over There', 'active'),
(3, 'Supplier 103', '09789897879', 'Anywhere There', 'active');

-- --------------------------------------------------------
--
-- Table structure for table `ims_user`
--
CREATE TABLE `ims_user` (
  `userid` int(11) NOT NULL,
  `email` varchar(200) NOT NULL,
  `password` varchar(200) NOT NULL,
  `name` varchar(200) NOT NULL,
  `type` enum('admin','member') NOT NULL,
  `status` enum('Active','Inactive') NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `ims_user`
--

INSERT INTO `ims_user` (`userid`, `email`, `password`, `name`, `type`, `status`) VALUES
(1, 'admin@mail.com', '0192023a7bbd73250516f069df18b500', 'Administrator', 'admin',
'Active');

--
-- Indexes for dumped tables
--
--
-- Indexes for table `ims_brand`
--
ALTER TABLE `ims_brand`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `ims_category`
--
ALTER TABLE `ims_category`
```

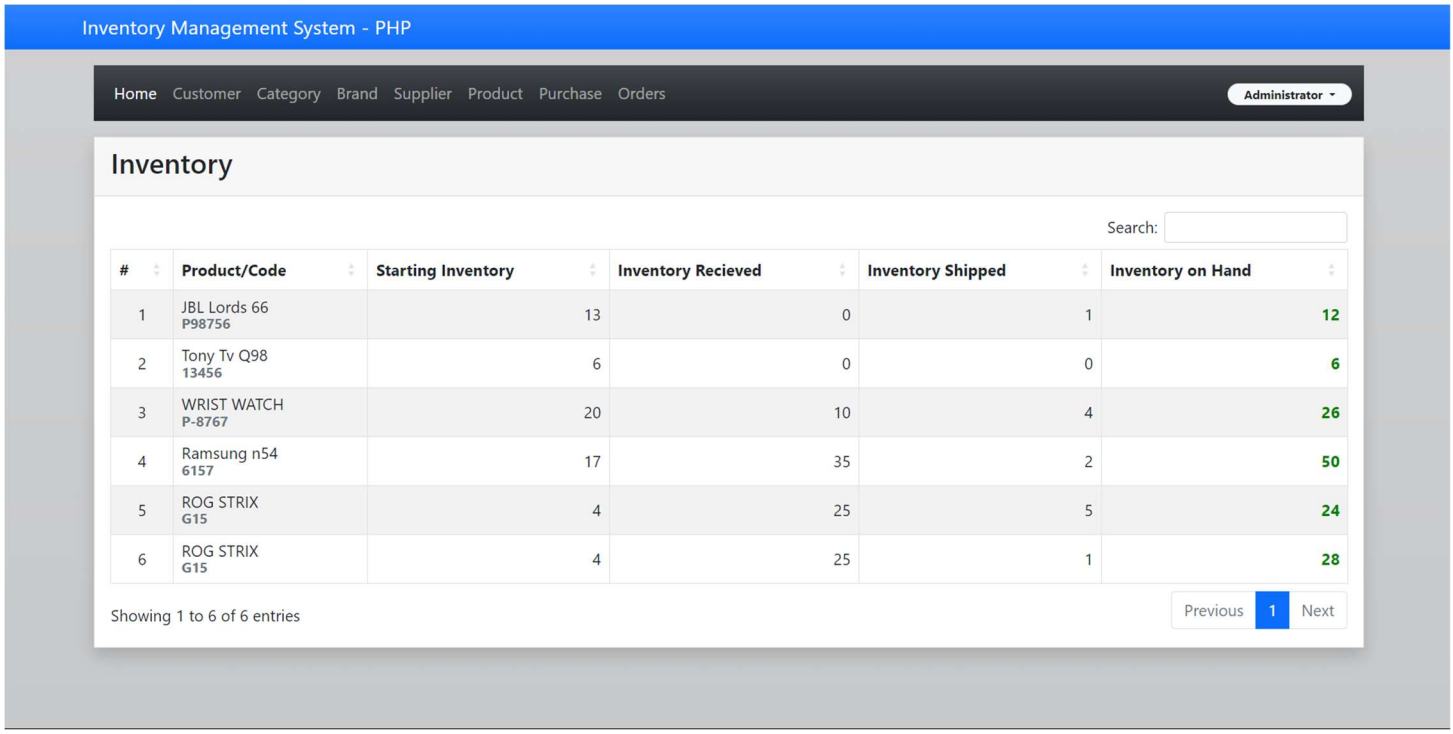
```
  ADD PRIMARY KEY (`categoryid`);
--
-- Indexes for table `ims_customer`
--
ALTER TABLE `ims_customer`
  ADD PRIMARY KEY (`id`);
--
-- Indexes for table `ims_order`
--
ALTER TABLE `ims_order`
  ADD PRIMARY KEY (`order_id`);

--
-- Indexes for table `ims_product`
--
ALTER TABLE `ims_product`
  ADD PRIMARY KEY (`pid`);

--
-- Indexes for table `ims_purchase`
--
ALTER TABLE `ims_purchase`
  ADD PRIMARY KEY (`purchase_id`);

--
-- Indexes for table `ims_supplier`
--
ALTER TABLE `ims_supplier`
  ADD PRIMARY KEY (`supplier_id`);

--
-- Indexes for table `ims_user`
--
ALTER TABLE `ims_user`
  ADD PRIMARY KEY (`userid`);

--
-- AUTO_INCREMENT for dumped tables
--


--
-- AUTO_INCREMENT for table `ims_brand`
--
ALTER TABLE `ims_brand`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=10;
```

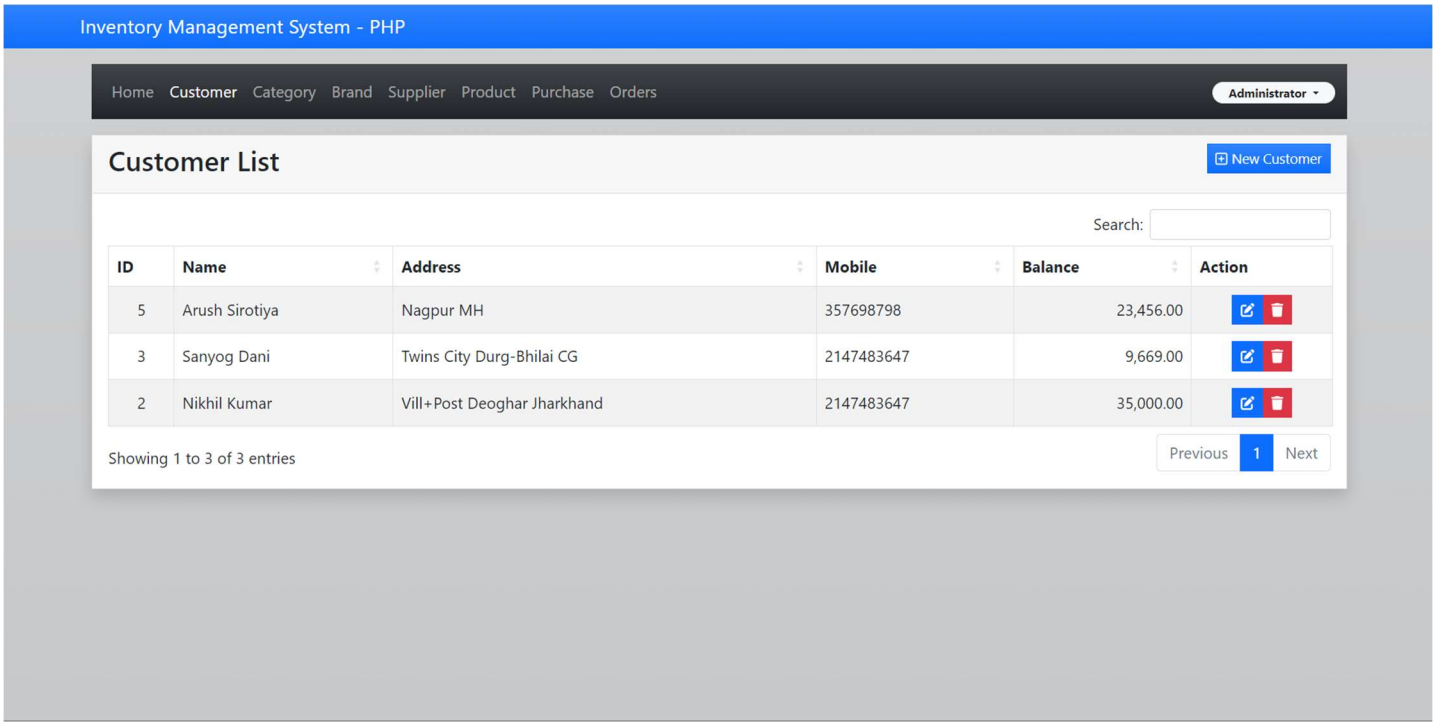
```sql
--
-- AUTO_INCREMENT for table `ims_category`
--
ALTER TABLE `ims_category`
  MODIFY `categoryid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

-- AUTO_INCREMENT for table `ims_customer`
--
ALTER TABLE `ims_customer`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `ims_order`
--
ALTER TABLE `ims_order`
  MODIFY `order_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `ims_product`
--
ALTER TABLE `ims_product`
  MODIFY `pid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT for table `ims_purchase`
--
ALTER TABLE `ims_purchase`
  MODIFY `purchase_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT for table `ims_supplier`
--
ALTER TABLE `ims_supplier`
  MODIFY `supplier_id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;

--
-- AUTO_INCREMENT for table `ims_user`
--
ALTER TABLE `ims_user`
  MODIFY `userid` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;
COMMIT;
```

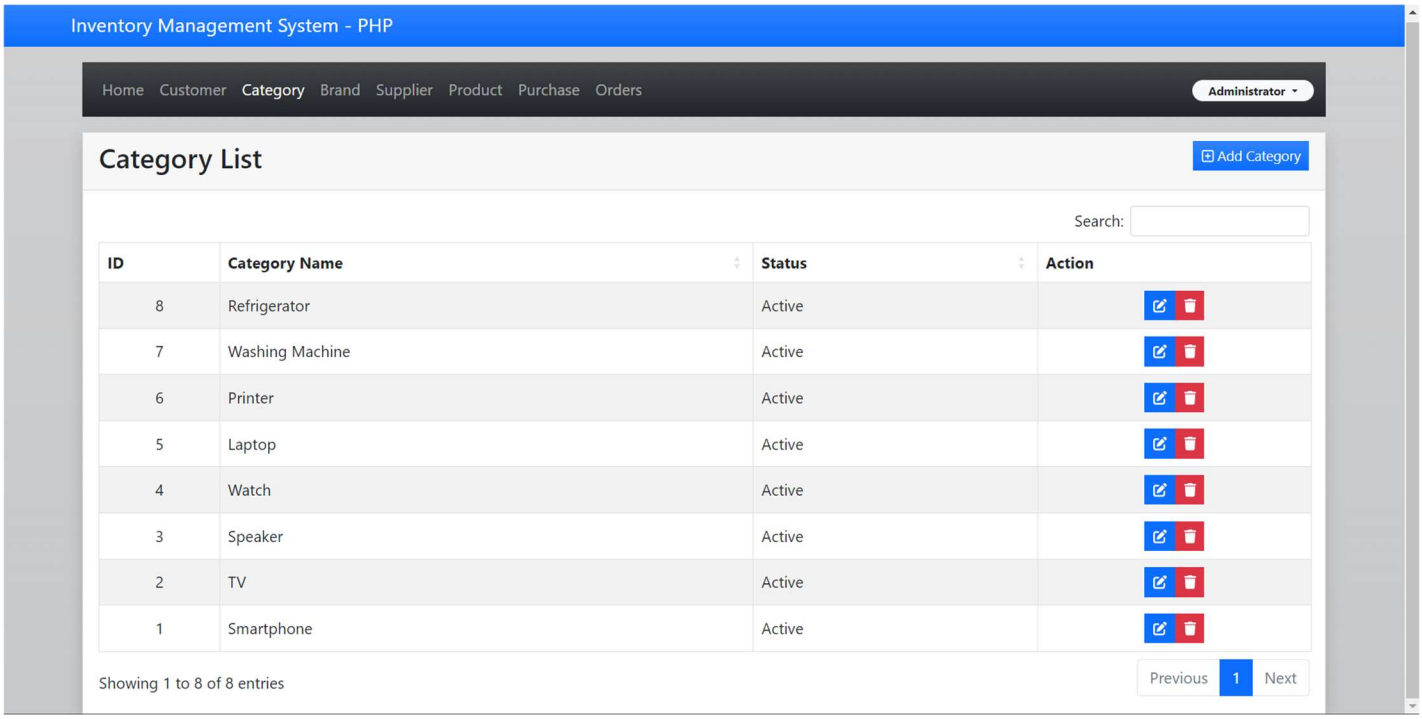
CHAPTER 7

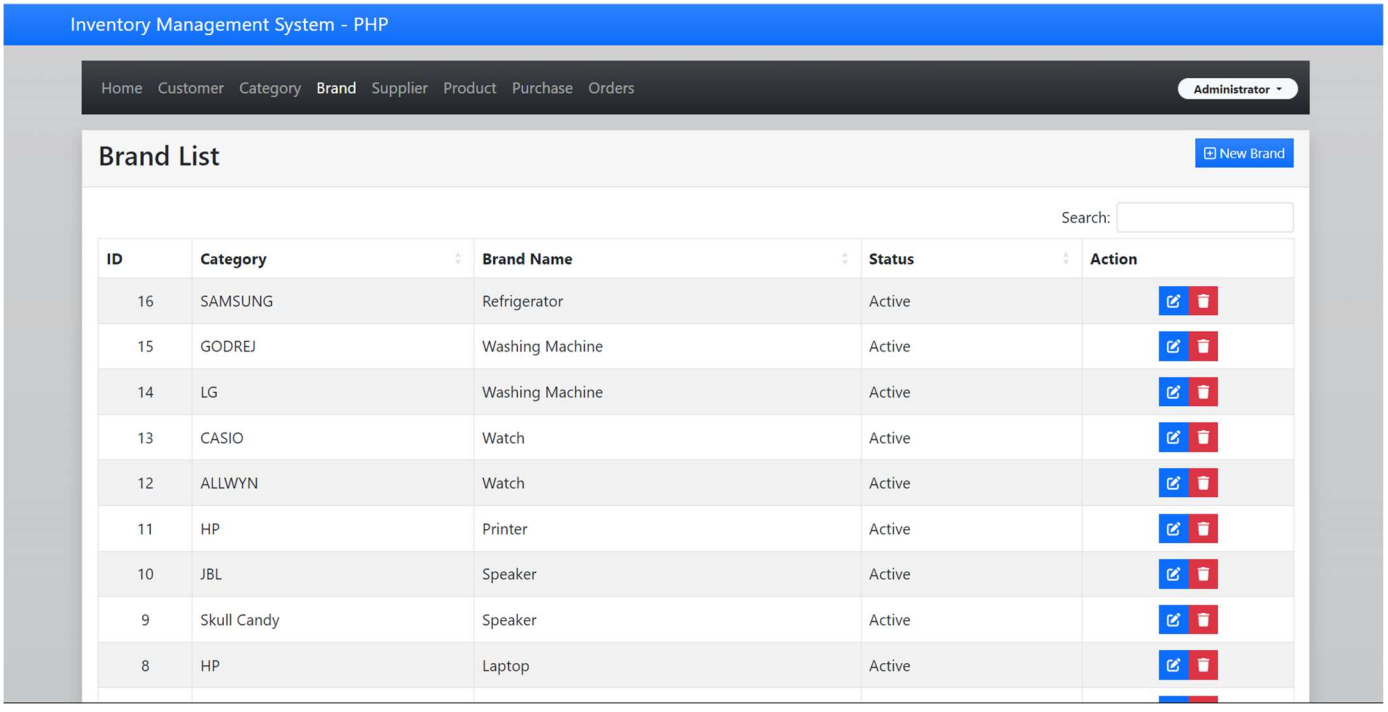
RESULTS & SCREENSHOTS

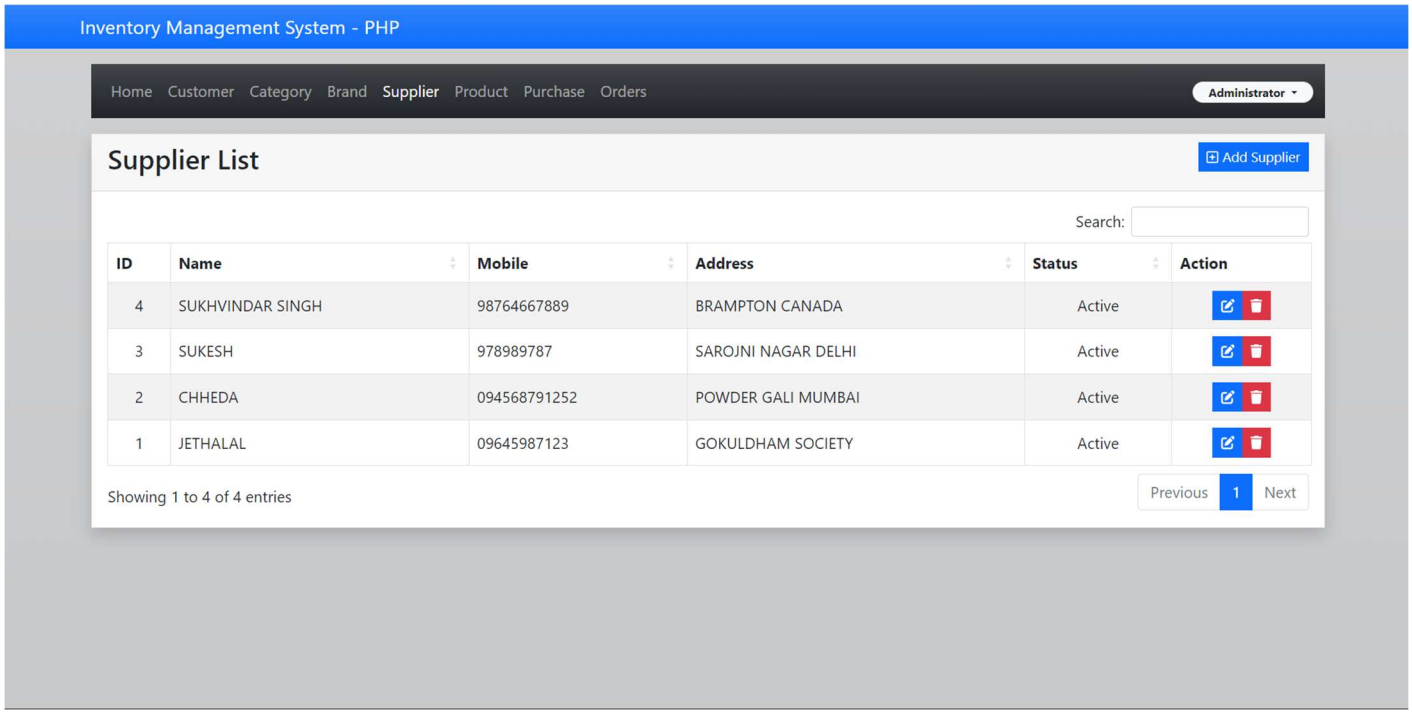


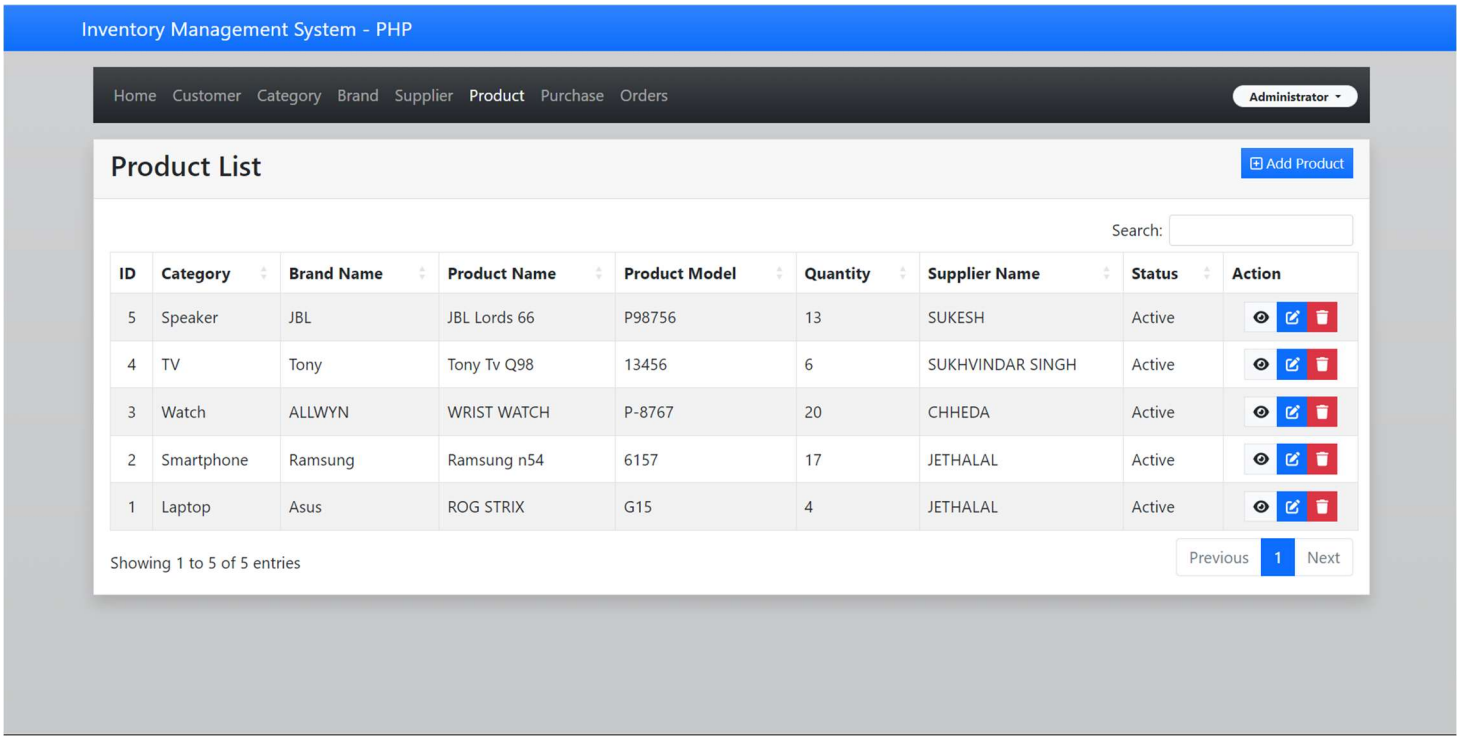

Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Inventory
Search:
Product/Code Starting Inventory Inventory Recieved Inventory Shipped Inventory on Hand
1 JBL Lords 66 P98756 13 0 1 12
2 Tony Tv Q98 13456 6 0 0 6
3 WRIST WATCH P-8767 20 10 4 26
4 Ramsung n54 6157 17 35 2 50
5 ROG STRIX G15 4 25 5 24
6 ROG STRIX G15 4 25 1 28
Showing 1 to 6 of 6 entries
Previous 1 Next

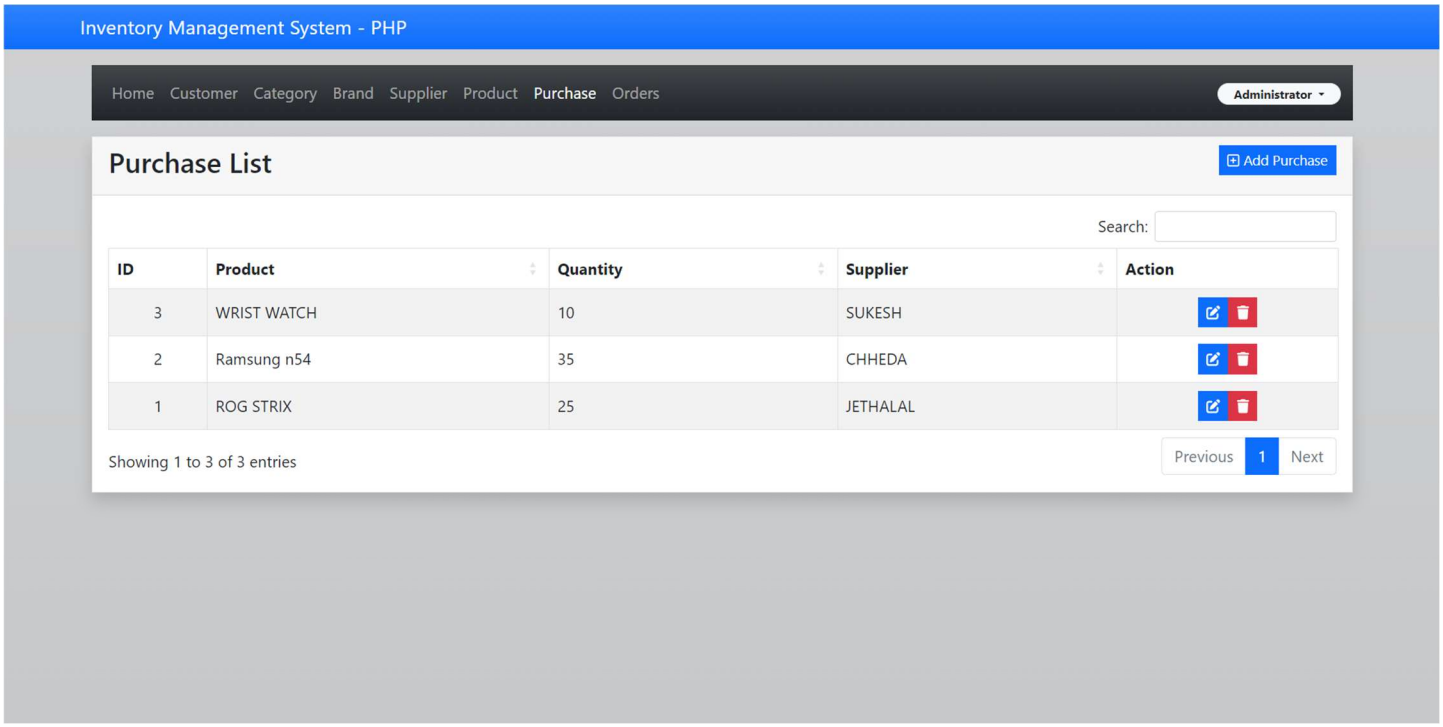

Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Customer List
New Customer
Search:
ID Name Address Mobile Balance Action
5 Arush Sirotiya Nagpur MH 357698798 23,456.00
3 Sanyog Dani Twins City Durg-Bhilai CG 2147483647 9,669.00
2 Nikhil Kumar Vill+Post Deoghar Jharkhand 2147483647 35,000.00
Showing 1 to 3 of 3 entries
Previous 1 Next

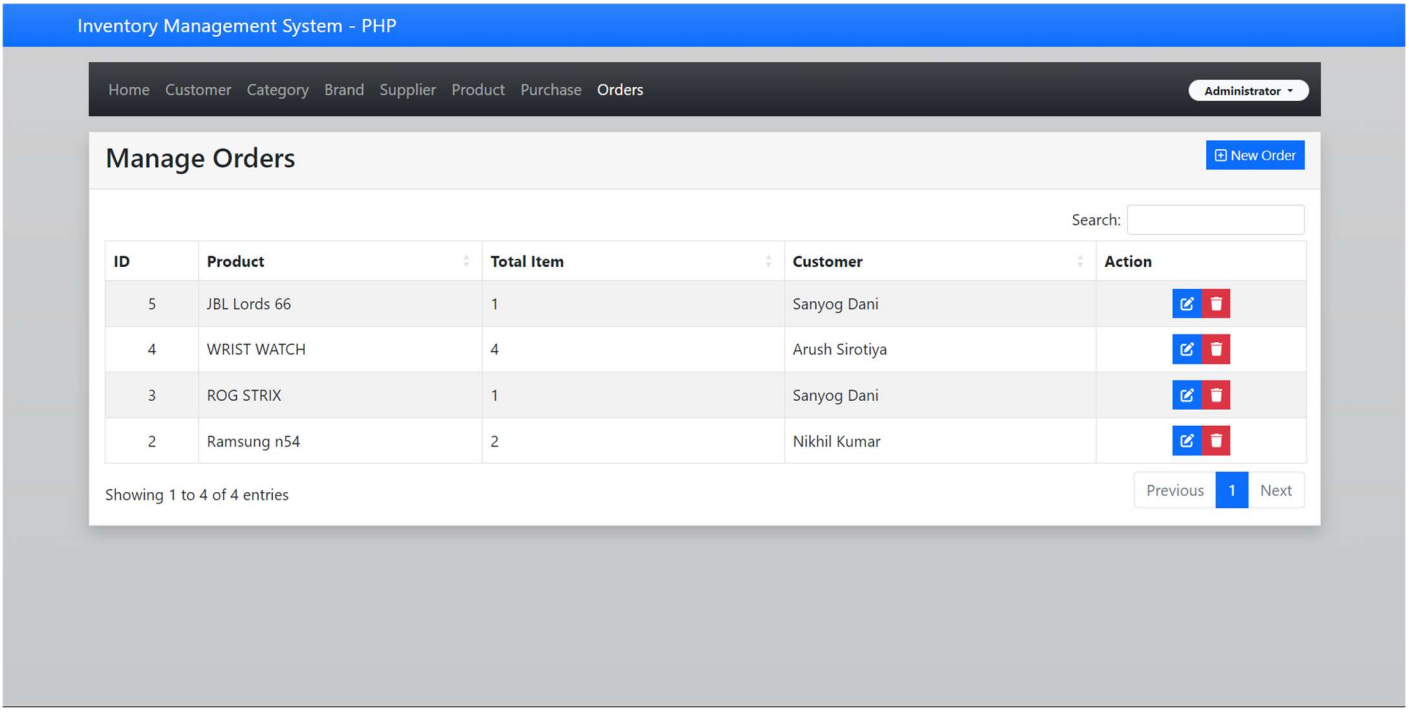

Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Category List
Add Category
Search:
ID
Category Name
Status
Action
8 Refrigerator Active
7 Washing Machine Active
6 Printer Active
5 Laptop Active
4 Watch Active
3 Speaker Active
2 TV Active
1 Smartphone Active
Showing 1 to 8 of 8 entries
Previous 1 Next


Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Brand List
New Brand
Search:
ID
Category
Brand Name
Status
Action
16 SAMSUNG Refrigerator Active
15 GODREJ Washing Machine Active
14 LG Washing Machine Active
13 CASIO Watch Active
12 ALLWYN Watch Active
11 HP Printer Active
10 JBL Speaker Active
9 Skull Candy Speaker Active
8 HP Laptop Active


Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Supplier List
Add Supplier
Search:
ID Name Mobile Address Status Action
4 SUKHVINDAR SINGH 98764667889 BRAMPTON CANADA Active
3 SUKESH 978989787 SAROJNI NAGAR DELHI Active
2 CHHEDA 094568791252 POWDER GALI MUMBAI Active
1 JETHALAL 09645987123 GOKULDHAM SOCIETY Active
Showing 1 to 4 of 4 entries
Previous 1 Next


Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Product List
Add Product
Search:
ID Category Brand Name Product Name Product Model Quantity Supplier Name Status Action
5 Speaker JBL JBL Lords 66 P98756 13 SUKESH Active
4 TV Tony Tony Tv Q98 13456 6 SUKHVINDAR SINGH Active
3 Watch ALLWYN WRIST WATCH P-8767 20 CHHEDA Active
2 Smartphone Ramsung Ramsung n54 6157 17 JETHALAL Active
1 Laptop Asus ROG STRIX G15 4 JETHALAL Active
Showing 1 to 5 of 5 entries
Previous 1 Next


Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Purchase List
Add Purchase
Search:
ID Product Quantity Supplier Action
3 WRIST WATCH 10 SUKESH
2 Ramsung n54 35 CHHEDA
1 ROG STRIX 25 JETHALAL
Showing 1 to 3 of 3 entries
Previous 1 Next


Inventory Management System - PHP
Home Customer Category Brand Supplier Product Purchase Orders
Administrator
Manage Orders
New Order
Search:
ID Product Total Item Customer Action
5 JBL Lords 66 1 Sanyog Dani
4 WRIST WATCH 4 Arush Sirotiya
3 ROG STRIX 1 Sanyog Dani
2 Ramsung n54 2 Nikhil Kumar
Showing 1 to 4 of 4 entries
Previous 1 Next

CONCLUSION

In conclusion, the Inventory Management System (IMS) stands as a pivotal solution for organizations seeking to overcome the challenges associated with inventory management in today's dynamic business environment. IMS offers a comprehensive platform designed to streamline processes, maximize productivity, and enhance overall operational efficiency.

With its robust inventory tracking capabilities, IMS provides real-time visibility into stock levels, locations, and movement, enabling organizations to maintain accurate and up-to-date inventory records. This visibility not only facilitates efficient inventory management but also helps minimize stockouts and overstock situations, thereby optimizing inventory levels.

IMS allows users to swiftly record incoming and outgoing inventory transactions, minimizing errors and enhancing accuracy in inventory management processes. By automating transaction recording, IMS reduces manual effort, mitigates the risk of human error, and ensures data integrity throughout the inventory lifecycle.

Furthermore, IMS provides advanced reporting, empowering stakeholders to gain valuable insights into inventory performance, identify trends, and make data-driven decisions. From analyzing inventory turnover rates to evaluating supplier performance metrics, IMS enables organizations to optimize all inventory strategies.

In today's competitive business landscape, effective inventory management is essential for meeting customer demand, optimizing cash flow, and maintaining a competitive edge. An Inventory Management System plays a crucial role in helping businesses achieve these objectives by providing visibility, control, and efficiency throughout the inventory lifecycle.

By adopting an Inventory Management System, businesses can enhance operational efficiency, minimize costs, and improve customer satisfaction, ultimately driving sustainable growth and success. IMS addresses the challenges of inventory tracking and visibility, inventory optimization, procurement efficiency, warehouse management, reporting, and analytics, ensuring that organizations can effectively manage their inventory operations and stay ahead in the marketplace.

CHAPTER 8

ONLINE COURSE CERTIFICATION

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Sanyog Dani

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 09 February 2024

Anshuman Singh
Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE BY SCALER

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

ARUSH SIROTIYA (RA2211031010092)

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 03 May 2024

Anshuman Singh

Co-founder SCALER

CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

NIKHIL KUMAR

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges 01 March 2024

Anshuman Singh

Co-founder SCALER