Established – 1961 Subject:_____

SEVA SADAN'S

R. K. TALREJA COLLEGE

OF

ARTS, SCIENCE & COMMERCE ULHASNAGAR – 421003

CERTIFICATE

This is to certify that Mr./Ms. _____ of S.Y. Information Technology (SYIT) Roll No. _____ has satisfactorily completed the Open Source DataBase Management System Mini Project entitled

_____

during the academic year 2025 – 2026, as a part of the practical requirement. The project work is found to be satisfactory and is approved for submission.

PROF. INCHARGE **HEAD OF DEPT**

_____ _____

INDEX

1.INTRODUCTION

In modern business organizations, vendors and suppliers play a crucial role in maintaining smooth procurement and supply chain operations. Every organization depends on suppliers for raw materials, products, or services required for daily business activities. Managing vendor and supplier records efficiently is essential to ensure timely purchases, accurate payments, and financial transparency.

In many small and medium-scale organizations, vendor and supplier data is still maintained using manual methods such as paper registers or spreadsheets. These traditional methods are time-consuming and prone to errors such as duplicate entries, missing records, incorrect payment calculations, and difficulty in tracking transaction history. As the volume of transactions increases, managing such data manually becomes inefficient and unreliable.

To overcome these challenges, the Vendor & Supplier Management Database Management System has been developed. The system is designed using a relational database approach, where data is organized into structured tables with clearly defined relationships. The database stores vendor details, supplier information, product records, purchase transactions, and payment details in an organized manner.

The system is implemented using MySQL, an open-source Relational Database Management System (RDBMS). MySQL provides powerful features such as SQL query processing, constraint enforcement, transaction management, concurrency control, and security mechanisms. By using SQL commands such as CREATE, INSERT, UPDATE, DELETE, JOIN, GROUP BY, and TRANSACTION control statements, the system ensures efficient data management and integrity.

This project provides a practical implementation of Database Management System concepts including normalization, primary and foreign keys, transaction handling, concurrency control, and data security. The system improves operational efficiency, ensures payment consistency, and provides reliable reporting capabilities.

2. PROBLEM DEFINITION

Organizations that manage vendor and supplier data manually face several operational and financial challenges.

One major issue is data redundancy and duplication. When vendor and supplier records are stored in multiple registers or spreadsheets, it becomes difficult to maintain consistency. Duplicate vendor entries and repeated product records increase confusion and reduce accuracy.

Another significant problem is payment inconsistency. Without proper transaction control, purchase records may be entered without corresponding payment updates, or payments may be recorded incorrectly. This leads to financial discrepancies and audit difficulties.

Searching and updating records manually is time-consuming. Retrieving supplier transaction history or calculating total purchases requires extensive manual effort, increasing the chances of errors.

Concurrency problems also arise when multiple users update records simultaneously. Without proper isolation and locking mechanisms, data conflicts may occur.

Lack of normalization results in unnecessary data repetition, increasing storage requirements and maintenance complexity.

Therefore, there is a strong need for a centralized relational database system that can:

- Maintain vendor and supplier data systematically
- Ensure payment consistency using transactions
- Handle concurrent access safely
- Enforce normalization principles
- Provide accurate and timely reports

3. OBJECTIVES OF THE PROJECT

The primary objective of the Vendor & Supplier Management Database Management System is to design and implement an efficient database solution using MySQL.

The specific objectives are:

- To design a structured relational database using MySQL
- To store vendor, supplier, product, purchase, and payment data efficiently
- To implement primary and foreign key constraints for referential integrity
- To maintain data consistency using transaction management
- To handle partial payments safely
- To manage concurrent transactions using isolation levels
- To apply normalization up to Third Normal Form (3NF)
- To generate reports using SQL queries such as JOIN, GROUP BY, and HAVING
- To implement security using role-based access control

4. SCOPE OF THE PROJECT

Users of the System
The **NGO Donation and Fund Utilization Database Management System** is designed to be used by different categories of authorized users within the NGO. Each user has specific roles and responsibilities to ensure smooth operation, data accuracy, and security.

- **NGO Administrator**
  The NGO Administrator has the highest level of access and control over the system. This user is responsible for managing donor records, creating and updating project details, monitoring overall donation inflow, and reviewing fund utilization across various projects. The administrator can also generate comprehensive reports required for decision-making, financial planning, and compliance purposes.
- **Staff Members**
  Staff members are responsible for the day-to-day operations of the NGO database. Their role includes recording new donations, updating donation details, entering fund utilization data for ongoing projects, and assisting in maintaining accurate and up-to-date records. Staff members play a key role in ensuring that donation and fund usage data is entered correctly and in a timely manner.
- **Auditors**
  Auditors are provided with restricted, read-only access to the system. Their primary responsibility is to verify donation records, review project-wise fund utilization, and ensure that funds are used in accordance with organizational and regulatory guidelines. This access helps maintain transparency, accountability, and trust among donors and regulatory authorities.

Application Areas
The NGO Donation and Fund Utilization Database Management System can be effectively applied in various operational and administrative areas of an NGO:

- **Donation Management**
  The system helps in maintaining structured and accurate records of donors and their contributions. It allows the NGO to track donation amounts, payment modes, and donation history, making donor management more efficient.

- **Project Funding Tracking**
  The database enables project-wise tracking of funds allocated and utilized. This helps the NGO monitor project expenses, identify remaining balances, and ensure that funds are used for their intended purposes.
- **Financial Transparency**
  By maintaining clear and organized records of donations and fund utilization, the system promotes transparency in financial operations. This increases donor confidence and strengthens the credibility of the NGO.
- **Audit and Compliance Support**
  The system simplifies auditing processes by providing accurate and easily accessible financial reports. Auditors and management can quickly verify records, check fund utilization, and ensure compliance with internal policies and external regulations.

Limitations of the Project

Although the NGO Donation and Fund Utilization Database Management System provides an efficient solution for managing NGO financial data, it has certain limitations:

- The project focuses only on **database-level implementation** and does not include a graphical user interface or front-end application.
- It does not support **web-based or mobile access**, limiting its usage to direct database interaction.
- **Real-time online payment gateway integration** is not included, and donation entries must be recorded manually.
- Advanced features such as automated notifications, analytics dashboards, and integration with external financial systems are beyond the scope of this project.

5. REQUIREMENT SPECIFICATION

5.1 Hardware Requirements

- Computer or Laptop
- Minimum 4 GB RAM
- Minimum 10 GB free storage

5.2 Software Requirements

Software	**Purpose**
MySQL Server	Database management
MySQL Workbench	Query execution and administration
Windows / Linux OS	Platform
SQL	Query language

.

6. SYSTEM DESIGN

System Architecture

- The system follows a database-centric architecture. Authorized users execute SQL queries through MySQL Workbench or VS Code. The MySQL Server processes queries and stores data in relational tables.

- Primary keys uniquely identify each record. Foreign keys establish relationships between tables and maintain referential integrity.

- Transaction management ensures that purchase and payment entries are processed atomically. If any error occurs during transaction execution, rollback operation restores the database to its previous consistent state.

- Concurrency control mechanisms such as isolation levels and row-level locking prevent conflicts when multiple users access the system simultaneously.

- The architecture ensures accuracy, reliability, security, and efficient data processing

- Query results are displayed in a clear tabular format, making data easy to understand and analyze.

- The architecture ensures fast processing, high security, data reliability, and easy system maintenance.

7. DATABASE DESIGN

7.1 Entities

The system consists of the following main entities:

1. Vendor

The Vendor entity stores information about vendors from whom goods or services are purchased.
It includes details such as vendor name, contact number, email, and address.

2. Supplier

The Supplier entity maintains records of suppliers who provide products to the organization.
Each supplier is uniquely identified and linked to products through a foreign key relationship.

3. Product

The Product entity stores details of products supplied by suppliers.
It contains attributes such as product name, price, and supplier reference.

4. Purchase

The Purchase entity records transaction details whenever products are bought from suppliers.
It includes quantity, total amount, purchase date, and references to supplier and product.

5. Payment

The Payment entity stores payment details related to purchases.
It records amount paid, payment date, payment mode, and links to the corresponding purchase.

7.2 Table Structure

1.Vendor Table

This table stores vendor details.

Attribute	Data Type	Description
vendor_id	INT	Primary key, uniquely identifies each vendor
vendor_name	VARCHAR(100)	Name of the vendor
contact_no	VARCHAR(15)	Contact number of vendor
email	VARCHAR(100)	Unique email address
address	TEXT	Vendor address

2. Supplier Table

This table stores supplier information.

Attribute	Data Type	Description
supplier_id	INT	Primary key, uniquely identifies each supplier
supplier_name	VARCHAR(100)	Name of supplier

Attribute	Data Type	Description
contact_no	VARCHAR(15)	Contact number
email	VARCHAR(100)	Unique email
address	TEXT	Supplier address

3. Product Table

This table stores product details supplied by suppliers.

Attribute	Data Type	Description
product_id	INT	Primary key
product_name	VARCHAR(100)	Name of product
price	DECIMAL(10,2)	Price of product
supplier_id	INT	Foreign key referencing Supplier
payment_mode	ENUM	Mode of payment (Cash / Online / Cheque)

4. Purchase Table

This table records purchase transactions.

Attribute	Data Type	Description
purchase_id	INT	Primary key
supplier_id	INT	Foreign key referencing Supplier
product_id	INT	Foreign key referencing Product
quantity	INT	Quantity purchased
total_amount	DECIMAL(10,2)	Total purchase amount
purchase_date	DATE	Date of purchase

5. Payment Table

This table stores payment details for purchases.

Attribute	Data Type	Description
payment_id	INT	Primary key
purchase_id	INT	Foreign key referencing Purchase
amount_paid	DECIMAL(10,2)	Amount paid
payment_date	DATE	Date of payment
payment_mode	ENUM('Cash','Online','Cheque')	Mode of payment
purchase_date	DATE	Date of purchase

1. Primary Key Constraint

• The PRIMARY KEY constraint is used to uniquely identify each record in a table.
• It ensures that no duplicate or NULL values are allowed in the primary key column.
• Examples:
o vendor_id in the Vendor table
o supplier_id in the Supplier table
o product_id in the Product table
o purchase_id in the Purchase table
o payment_id in the Payment table

2. Foreign Key Constraint

• The FOREIGN KEY constraint is used to establish relationships between related tables.
• It ensures referential integrity, meaning that related records must exist in the parent table.
• Examples:
o supplier_id in the Product table references supplier_id in the Supplier table
o product_id in the Purchase table references product_id in the Product table
o purchase_id in the Payment table references purchase_id in the Purchase table

3. NOT NULL Constraint

• The NOT NULL constraint ensures that mandatory fields are not left empty.
• It guarantees that essential information is always provided.
• Examples:
o Vendor name and Supplier name
o Product name and price
o Purchase quantity and total amount

4. UNIQUE Constraint

• The UNIQUE constraint prevents duplicate entries in columns where unique values are required.
• It helps maintain data accuracy and avoids redundancy.
• Examples:
o Email field in the Vendor table
o Email field in the Supplier table

5. ENUM Constraint

• The ENUM constraint restricts a column to accept only a predefined set of values.
• It improves data consistency by preventing invalid entries.
• Examples:
o payment_mode in the Payment table (Cash, Online, Cheque)

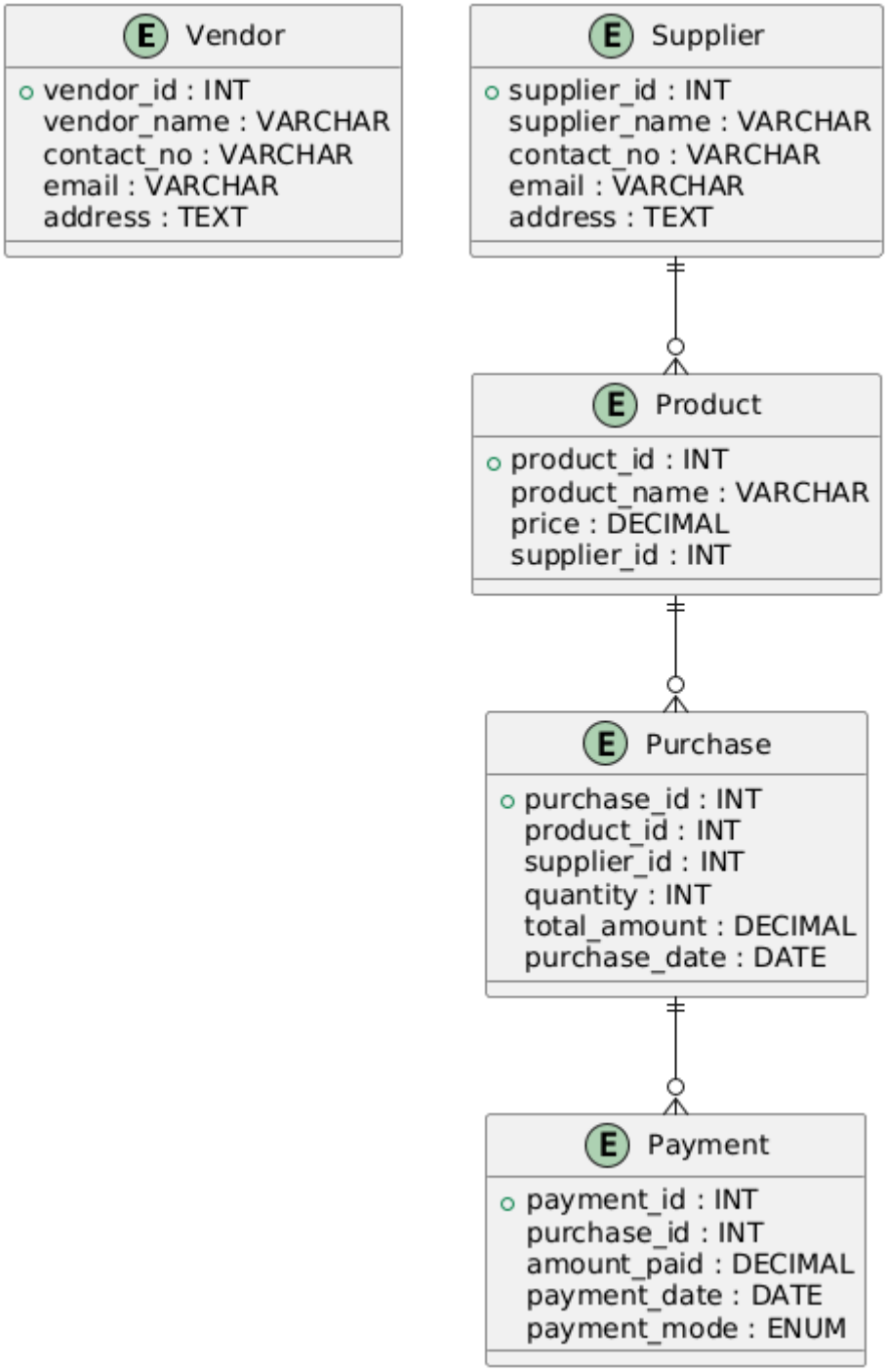
6. Data Type Constraints

• Appropriate data types such as INT, VARCHAR, DATE, DECIMAL, and TEXT are used.
• This ensures correct storage of data and improves database performance.

8. UML DIAGRAMS

8.1 ER Diagram

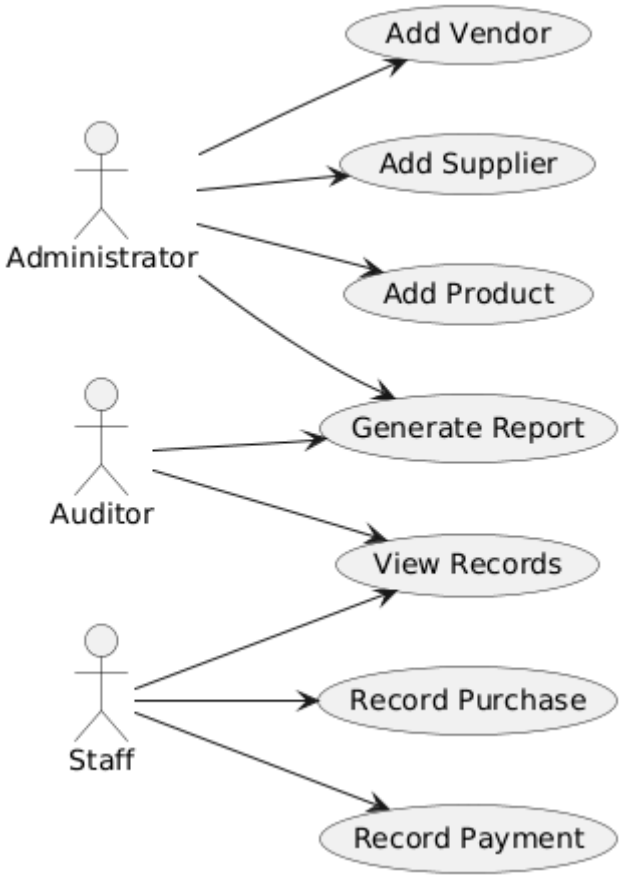
Vender and Supplier Management ER Diagram



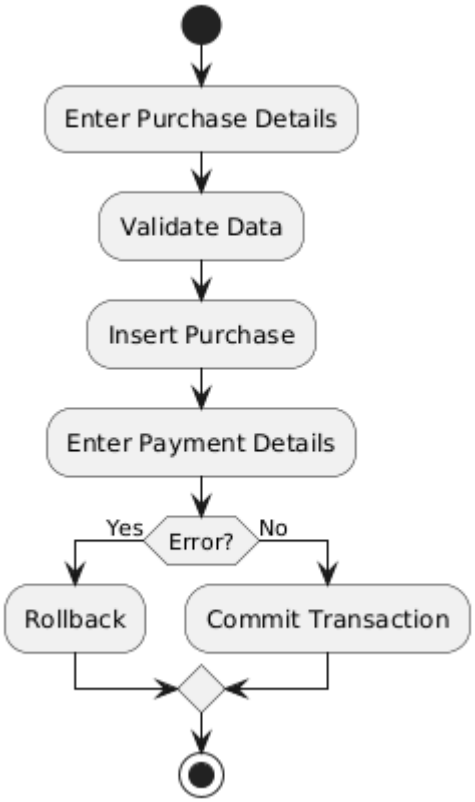
8.1 ER Diagram

8.2 Use Case Diagram

USE CASE DIAGRAM-Vender and Supplier Management



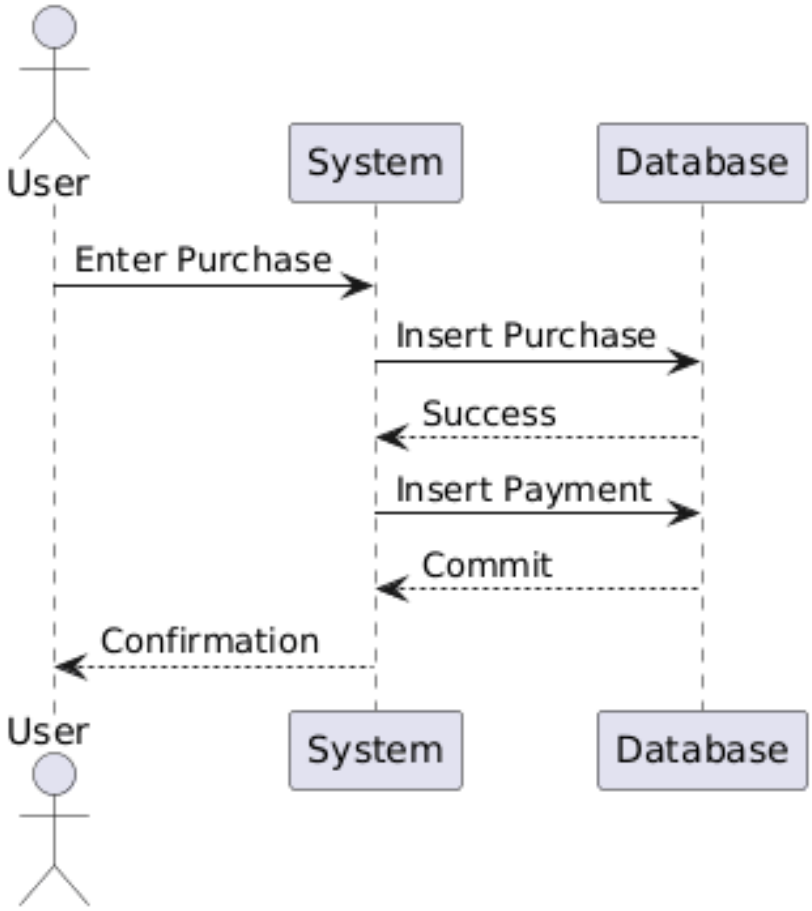
8.3 Activity Diagram



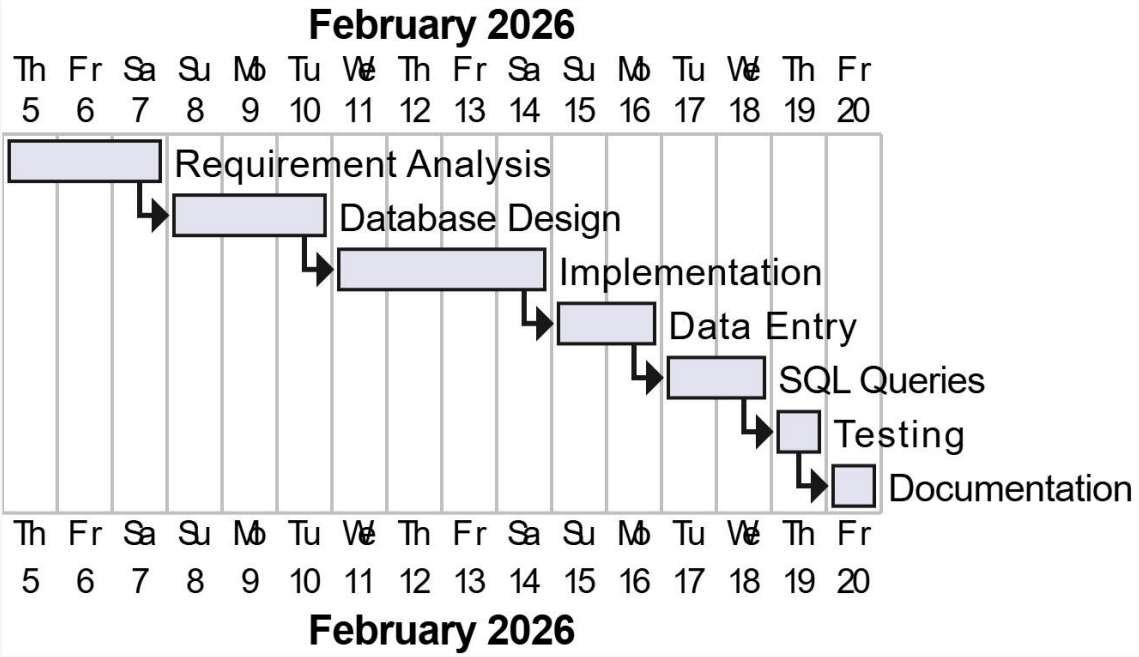
8.3 Activity Diagram

8.4 Sequence Diagram

Sequence Diagram-Vender and Supplier Management



8.4 Sequence Diagram



8.5 Gantt chart

9.SQL Implementation

9.1 Database Creation

```
CREATE DATABASE VendorSupplierDB;
USE VendorSupplierDB;
```

9.2 Table Creation

Vendor Table

```
CREATE TABLE Vendor (
    vendor_id INT AUTO_INCREMENT PRIMARY KEY,
    vendor_name VARCHAR(100) NOT NULL,
    contact_no VARCHAR(15),
    email VARCHAR(100) UNIQUE,
    address TEXT
);
```

Supplier Table

```
CREATE TABLE Supplier (
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,
    supplier_name VARCHAR(100) NOT NULL,
    contact_no VARCHAR(15),
    email VARCHAR(100) UNIQUE,
    address TEXT
);
```

Product Table

```
CREATE TABLE Product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    product_name VARCHAR(100) NOT NULL,
    price DECIMAL(15,2) NOT NULL CHECK (price > 0),
    supplier_id INT,
    FOREIGN KEY (supplier_id) REFERENCES Supplier(supplier_id)
);
```

Purchase Table

```
CREATE TABLE Purchase (
    purchase_id INT AUTO_INCREMENT PRIMARY KEY,
    vendor_id INT NOT NULL,
    supplier_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL CHECK (quantity > 0),
    total_amount DECIMAL(15,2) NOT NULL,
```

```
    purchase_date DATE NOT NULL,
    FOREIGN KEY (vendor_id)
    REFERENCES Vendor(vendor_id),
    FOREIGN KEY (supplier_id)
    REFERENCES Supplier(supplier_id),
    FOREIGN KEY (product_id)
    REFERENCES Product(product_id)
);
```

Payment Table

```
CREATE TABLE Payment (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    purchase_id INT NOT NULL,
    amount_paid DECIMAL(15,2) NOT NULL CHECK (amount_paid >= 0),
    payment_date DATE NOT NULL,
    payment_mode ENUM('Cash','Online','Cheque') NOT NULL,
    FOREIGN KEY (purchase_id)
    REFERENCES Purchase(purchase_id)
);
```

9.3 Data Insertion

```
INSERT INTO Vendor (vendor_name, contact_no, email, address)
VALUES ('Rohan Enterprises', '9876543210', 'rohan@gmail.com', 'Mumbai');

INSERT INTO Supplier (supplier_name, contact_no, email, address)
VALUES ('ABC Traders', '9123456789', 'abc@gmail.com', 'Thane');

INSERT INTO Product (product_name, price, supplier_id)
VALUES ('Laptop', 50000.00, 1);
```

9.4 Transaction Handling

```
START TRANSACTION;

-- Step 1: Insert Purchase
INSERT INTO Purchase
(vendor_id, supplier_id, product_id, quantity, total_amount, purchase_date)
VALUES
(1, 1, 1, 2, 100000.00, CURDATE());

-- Step 2: Insert Payment
INSERT INTO Payment
(purchase_id, amount_paid, payment_date, payment_mode)
VALUES
```

```
(LAST_INSERT_ID(), 100000.00, CURDATE(), 'Online');

-- Step 3: Commit Transaction
COMMIT;
```

9.5 Payment Consistency

```
INSERT INTO Payment
(purchase_id, amount_paid, payment_date, payment_mode)
VALUES
(1, 50000.00, CURDATE(), 'Cash');
```

9.6 Remaining Balance Calculation

```
SELECT
   p.purchase_id,
   p.total_amount,
   IFNULL(SUM(pay.amount_paid),0) AS paid_amount,
   (p.total_amount - IFNULL(SUM(pay.amount_paid),0)) AS
remaining_balance
FROM Purchase p
LEFT JOIN Payment pay
ON p.purchase_id = pay.purchase_id
GROUP BY p.purchase_id;
```

9.7 Concurrency Control

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
```

9.8 View Creation

```
CREATE VIEW Purchase_Summary AS
SELECT v.vendor_name,
     s.supplier_name,
     SUM(p.total_amount) AS total_purchase
FROM Purchase p
JOIN Vendor v ON p.vendor_id = v.vendor_id
JOIN Supplier s ON p.supplier_id = s.supplier_id
GROUP BY v.vendor_name, s.supplier_name;
```

Output:

Vendor Table Output

The Vendor table displays all inserted vendor records with their details.
This confirms that the table was created and data was inserted successfully.

```
mysql> select * from Vendor;
+-----------+--------------------+------------+------------------+-------------+
| vendor_id | vendor_name        | contact_no | email            | address     |
+-----------+--------------------+------------+------------------+-------------+
|         1 | Rohan Enterprises  | 9876543210 | rohan@gmail.com  | Mumbai      |
|         2 | ABC Traders        | 9123456789 | abc@gmail.com    | Thane       |
|         3 | Star Supplies      | 9012345678 | star@gmail.com   | Pune        |
|         4 | Global Tech        | 8899776655 | global@gmail.com | Delhi       |
|         5 | Metro Distributors | 9988776655 | metro@gmail.com  | Navi Mumbai |
+-----------+--------------------+------------+------------------+-------------+
5 rows in set (0.00 sec)
```

Supplier Table Output

The Supplier table shows all supplier records stored in the database.
It verifies successful table creation and data insertion.

```
mysql> select*from Supplier;
+-------------+---------------+------------+--------------------+---------+
| supplier_id | supplier_name | contact_no | email              | address |
+-------------+---------------+------------+--------------------+---------+
|           1 | Tech Source   | 9111111111 | tech@gmail.com     | Mumbai  |
|           2 | IT World      | 9222222222 | itworld@gmail.com  | Pune    |
|           3 | Digital Hub   | 9333333333 | digital@gmail.com  | Delhi   |
+-------------+---------------+------------+--------------------+---------+
3 rows in set (0.00 sec)
```

Purchase Table Output

The Purchase table shows one purchase transaction with quantity and total amount.
This confirms that the transaction was inserted correctly.

```
mysql> select*from purchase;
+-------------+-----------+-------------+------------+----------+--------------+---------------+
| purchase_id | vendor_id | supplier_id | product_id | quantity | total_amount | purchase_date |
+-------------+-----------+-------------+------------+----------+--------------+---------------+
|           1 |         1 |           1 |          1 |        2 |    100000.00 | 2026-02-22    |
+-------------+-----------+-------------+------------+----------+--------------+---------------+
1 row in set (0.00 sec)
```

View Output (Purchase_Summary)

The view displays vendor name, supplier name, and total purchase amount.
This confirms that JOIN and aggregation are working correctly.

```
mysql> CREATE OR REPLACE VIEW Purchase_Summary AS
    -> SELECT
    ->     v.vendor_name,
    ->     s.supplier_name,
    ->     SUM(p.total_amount) AS total_purchase_amount
    -> FROM Purchase p
    -> JOIN Vendor v ON p.vendor_id = v.vendor_id
    -> JOIN Supplier s ON p.supplier_id = s.supplier_id
    -> GROUP BY v.vendor_name, s.supplier_name;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> SELECT * FROM Purchase_Summary;
+-------------------+---------------+-----------------------+
| vendor_name       | supplier_name | total_purchase_amount |
+-------------------+---------------+-----------------------+
| Rohan Enterprises | Tech Source   |             100000.00 |
+-------------------+---------------+-----------------------+
1 row in set (0.01 sec)
```

Payment Table Output

The Payment table shows two payments for the same purchase.
This proves that the system supports multiple (partial) payments.

```
mysql> select * from payment;
+------------+-------------+-------------+--------------+--------------+
| payment_id | purchase_id | amount_paid | payment_date | payment_mode |
+------------+-------------+-------------+--------------+--------------+
|          1 |           1 |   100000.00 | 2026-02-22   | Online       |
|          2 |           1 |    50000.00 | 2026-02-22   | Cash         |
+------------+-------------+-------------+--------------+--------------+
2 rows in set (0.00 sec)
```

Remaining Balance Output

The query calculates total paid amount and remaining balance.
The result shows the correct balance calculation based on payments.

```
mysql> SELECT
    ->     p.purchase_id,
    ->     p.total_amount,
    ->     IFNULL(SUM(py.amount_paid), 0) AS paid_amount,
    ->     (p.total_amount - IFNULL(SUM(py.amount_paid), 0)) AS remaining_balance
    -> FROM Purchase p
    -> LEFT JOIN Payment py
    ->     ON p.purchase_id = py.purchase_id
    -> GROUP BY p.purchase_id, p.total_amount;
+-------------+--------------+-------------+-------------------+
| purchase_id | total_amount | paid_amount | remaining_balance |
+-------------+--------------+-------------+-------------------+
|           1 |    100000.00 |   150000.00 |         -50000.00 |
+-------------+--------------+-------------+-------------------+
1 row in set (0.01 sec)
```

10. SYSTEM TESTING AND RESULT

System testing was conducted to verify the correctness, reliability, and performance of the Vendor & Supplier Management Database Management System. Each database component was tested using various SQL queries to ensure that the system functions properly and meets the project requirements.

SYSTEM TESTING

The following testing activities were performed during the implementation phase:

- All Data Definition Language (DDL) commands such as CREATE DATABASE and CREATE TABLE were executed successfully to ensure proper database and table creation.
- Data Manipulation Language (DML) operations including INSERT, SELECT, and JOIN were tested with valid data.
- Primary key constraints were tested to ensure that duplicate records are not allowed.
- Foreign key constraints were tested to verify that purchase and payment records cannot exist without valid vendor, supplier, and product references.
- Transaction control statements (START TRANSACTION and COMMIT) were tested to ensure atomicity and data consistency.
- Partial payment functionality was tested to confirm that multiple payments can be recorded for a single purchase.
- Aggregate functions such as SUM and GROUP BY were tested to calculate total payments and remaining balance.
- The view (Purchase_Summary) was tested to verify correct report generation.
- Concurrency control using transaction isolation level was verified to prevent simultaneous data conflicts.

RESULTS

The results of system testing confirmed that the database system operates efficiently and reliably:

- All tables were created successfully with correct constraints.
- Foreign key relationships maintained referential integrity.
- Transactions executed properly and maintained consistency.
- Partial payment handling worked accurately.
- Remaining balance calculation was correct.
- View generated summarized purchase reports successfully.
- No data inconsistency or integrity issues were observed.

CONCLUSION OF TESTING

The system testing results demonstrate that the Vendor & Supplier Management Database Management System operates efficiently and reliably. The database ensures structured data storage, secure transaction handling, and accurate report generation. The system is suitable for small-scale vendor and supplier management applications at the database level.

11. SECURITY, BACKUP AND RECOVERY

Security, backup, and recovery are essential components of any database system. Since this project manages vendor and financial transaction data, proper protection mechanisms are implemented to ensure data confidentiality and integrity.

SECURITY

Database security is implemented using **role-based access control** in MySQL. This ensures that users can access only the data and operations relevant to their roles.

Key security measures include:

- Different users can be assigned specific privileges.
- The GRANT command is used to provide access rights.
- The REVOKE command is used to remove unnecessary privileges.
- Sensitive operations such as table deletion are restricted to administrators.

Example of granting privileges:

```
GRANT SELECT, INSERT, UPDATE
ON VendorSupplierDB.*
TO 'staff_user'@'localhost';
```

These measures help prevent unauthorized data modification and ensure controlled database access.

BACKUP

Database backup is performed using the mysqldump utility. This creates a complete SQL backup file containing both structure and data.

Example backup command:

```
mysqldump -u root -p VendorSupplierDB > VendorSupplierDB_Backup.sql
```

TRegular backups help protect data against accidental deletion, hardware failure, or corruption.

RECOVERY

Recovery is performed by restoring the backup file using MySQL.

Example recovery command:

```
mysql -u root -p VendorSupplierDB < VendorSupplierDB_Backup.sql
```

This restores the database to its previous state and ensures business continuity.

SUMMARY

- Role-based access control ensures secure database usage.
- GRANT and REVOKE commands prevent unauthorized operations.
- mysqldump provides reliable database backups.
- SQL backup files allow easy and fast recovery.
- Together, these mechanisms ensure data security, reliability, and integrity.

12. FUTURE SCOPE AND CONCLUSION

FUTURE SCOPE

Although the Vendor & Supplier Management Database Management System fulfills its core

objectives, several improvements can be made in the future:

Possible future enhancements include:

- **Development of a web-based user interface for easier interaction.**
- **Integration with online payment systems.**
- **Implementation of graphical dashboards and analytical reports.**
- **Addition of inventory management features.**
- **Implementation of advanced security mechanisms such as password encryption.**
- **Multi-user and role-based login system.**

CONCLUSION
The Vendor & Supplier Management Database Management System was successfully designed and implemented using MySQL. The project demonstrates the practical application of relational database concepts such as primary keys, foreign keys, normalization, transaction management, and concurrency control.

The system reduces data redundancy, ensures payment consistency, and provides accurate financial tracking. It improves efficiency in managing vendor and supplier information and supports structured data handling.

Overall, this project provides a strong understanding of database design and implementation using MySQL and fulfills the academic objectives of the Open Source Database Management System course.

13.REFERENCES

• MySQL Official Documentation:

MySQL Documentation, Oracle Corporation. Used as the primary reference for understanding MySQL commands, database creation, table design, queries, security, backup, and recovery concepts.

• Prescribed Textbooks:

Standard textbooks on Database Management Systems recommended in the academic syllabus. These books were referred to for understanding basic DBMS concepts such as relational databases, SQL queries, constraints, and normalization.

• Online Learning Sources:

Educational websites, tutorials, and online learning platforms related to SQL and MySQL. These sources helped in learning practical query execution, examples, and best practices for database implementation.

14: GLOSSARY

The glossary section provides simple and clear definitions of important technical terms used throughout the Vendor & Supplier Management Database Management System project. These terms are mainly related to database management systems, SQL concepts, transaction handling, and data integrity mechanisms, which are essential for understanding how the system operates.

The definitions are written in easy and understandable language so that readers with basic knowledge of database systems can easily comprehend them. This section serves as a quick reference guide and helps eliminate confusion regarding technical terminology used in the project documentation.

DBMS (Database Management System):
A Database Management System is software that allows users to store, organize, manage, and retrieve data efficiently. It provides tools for handling large volumes of data securely while maintaining data integrity and consistency.

RDBMS (Relational Database Management System):

A Relational Database Management System is a type of DBMS that stores data in structured tables with relationships between them. MySQL is an example of an RDBMS.

SQL (Structured Query Language):
SQL is a standard programming language used to interact with relational databases. It is used to create database structures, insert and update data, retrieve records, and control access to the database.

Primary Key:
A primary key is a unique identifier for each record in a database table. It ensures that every record can be uniquely identified and prevents duplicate entries in the table.

Foreign Key:
A foreign key is a field in one table that refers to the primary key of another table. It is used to establish relationships between tables and helps maintain referential integrity within the database.

MySQL:
MySQL is an open-source relational database management system used to create and manage databases. It is widely used due to its reliability, ease of use, strong security features, and support for standard SQL operations.

Constraint:
A constraint is a rule applied to a table column to restrict the type of data that can be stored. Constraints help ensure data accuracy and integrity within the database.

Transaction:
A transaction is a sequence of one or more SQL operations executed as a single unit of work. Transactions ensure that database operations are completed fully or not executed at all, maintaining consistency using commit and rollback operations.

Normalization:
Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing data into multiple related tables.

Concurrency:
Concurrency refers to the ability of the database to allow multiple users to access and modify data simultaneously without causing inconsistencies.

View:
A view is a virtual table created using a SQL query. It displays data from one or more tables and is mainly used for reporting and simplifying complex queries.