VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590018

A Database Management System Mini Project Report on

"Inventory Management System"

Submitted in Partial fulfillment of the Requirements for the IV Semester of the Degree of

Bachelor of Engineering in

Computer Science & Engineering

By

Sohan B (1CR23CS190)

Swaraj Kumar Sahu(1CR23CS195)

Under the Guidance of,

Dr Sanchari Saha, Asst Professor, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A++" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

2024-25

CMR INSTITUTE OF TECHNOLOGY

Affiliated to VTU, Approved by AICTE, Accredited by NBA and NAAC with "A++" Grade

ITPL MAIN ROAD, BROOKFIELD, BENGALURU-560037, KARNATAKA, INDIA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Database Management System Project work entitled **"Inventory Management System"** has been carried out by **Sohan B (1CR23CS190) and Swaraj Kumar Sahu (1CR23CS195),** Bonafide students of CMR Institute of Technology, Bengaluru in partial fulfillment for the award of the Degree of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belagavi during the year **2024-2025**. It is certified that all corrections/suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the departmental library. This Database Management System Project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Degree.

Signature of Guide	**Signature of HOD**
(Dr Sanchari Saha)	**(Dr. Kesavamoorthy R)**
(Asst Professor)	**Professor & HoD**
Dept. of CSE, CMRIT	**Dept. of CSE, CMRIT**

DECLARATION

We, the students of V semester from Department of Computer Science and Engineering, CMR Institute of Technology, Bangalore declare that the project work entitled "**Inventory Management System**" has been successfully completed under the guidance of Dr Sanchari Saha, Asst Professor, Dept. of Computer Science and Engineering, CMR Institute of technology, Bengaluru. This project work is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering during the academic year 2024-2025. The matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place: Bangalore

Date:28/05/25

Team members:

Sohan B (1CR23CS190)	
Swaraj Kumar Sahu (1CR23CS195)	

ABSTRACT

The Inventory Management System is a full-stack web application developed to streamline the operations of retail and wholesale businesses by managing essential components like customers, products, employees, suppliers, and transactions. The primary aim of this system is to offer a centralized and efficient solution for tracking stock levels, recording sales, monitoring supplier deliveries, and generating real-time insights—all within a user-friendly interface.

The frontend of the application is built using React with Vite, delivering a fast and responsive user experience. Key features include dynamic dashboards, low stock alerts, recent transaction tracking, and CRUD operations for all major entities. The backend is powered by Flask (Python), exposing RESTful APIs that connect seamlessly to a MySQL database. This architecture ensures data consistency, rapid communication, and ease of deployment across platforms.

In addition to real-time operations, the system supports advanced SQL functionalities like views and triggers. For instance, a trigger automatically updates a membership table based on customer transaction frequency, enabling basic loyalty program tracking. Views are used to generate pre-defined reports like customer spending summaries, low-stock product lists, and membership tiers, simplifying data access for analytics and decision-making.

Overall, this project demonstrates the practical implementation of modern web technologies combined with robust database practices. It offers a scalable solution for small and medium-sized enterprises to manage inventory effectively, reduce manual errors, and make data-driven business decisions. The modular design and REST-based architecture also allow for easy expansion, such as integrating billing, invoicing, or analytics modules in the future.

ACKNOWLEDGEMENT

I take this opportunity to express my sincere gratitude and respect to **CMR Institute of Technology, Bengaluru** for providing me a platform to pursue my studies and carry out the Database Management System Project.

It gives me an immense pleasure to express my deep sense of gratitude to **Dr. Sanjay Jain,** Principal, CMRIT, Bengaluru, for his constant encouragement.

I would like to extend my sincere gratitude to **Dr. Kesavamoorthy R,** HOD, Department of Computer Science and Engineering, CMRIT, Bengaluru, who has been a constant support and encouragement throughout the course of this project.

I would like to thank my guide **Dr Sanchari Saha, Asst Professor,** Department of Computer Science and Engineering, for the valuable guidance throughout the tenure of the project work.

I would also like to thank all the faculty members of Department of Computer Science and Engineering who directly or indirectly encouraged me.

Finally, I thank my parents and friends for all the moral support they have given me during the completion of this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Introduction

Inventory management is a critical component of every product-based business, directly affecting profitability, operational efficiency, and customer satisfaction. Traditional inventory handling methods—often reliant on manual recordkeeping or disparate software tools—are prone to human error, stock mismatches, and inefficiencies in tracking transactions or supplier deliveries. To overcome these limitations, we developed a comprehensive Inventory Management System that automates and centralizes all aspects of inventory control through a robust, full-stack application.

The system is designed with a strong emphasis on usability, maintainability, and performance. The frontend, built with React and Vite, provides an intuitive user interface that allows administrators to monitor sales, manage products, and view real-time alerts like low stock warnings and recent transactions. The backend, implemented using Flask, acts as a lightweight and scalable API server, interfacing with a MySQL database to ensure data integrity and fast CRUD operations across multiple relational tables such as customer, product, employee, supplier, and transaction.

Key challenges addressed by this system include synchronizing data between modules, supporting relational integrity via foreign keys, and enabling extensibility for future modules like billing or analytics. The backend also incorporates database-level features like SQL views for report generation and triggers to automate processes—for example, maintaining a membership table that tracks customer purchase frequency and updates loyalty status in real time.

By offering features like real-time dashboard updates, clean data separation, and automated backend logic, this Inventory Management System serves as a modern, scalable alternative to legacy tools. It is well-suited for small to medium enterprises aiming to digitize and streamline their inventory workflows with minimal technical overhead and maximum adaptability.

Requirements

This section outlines the hardware and software components necessary for the successful deployment and operation of the Inventory Management System.

2.1 Hardware Requirements

Processor (CPU): Dual-Core 2.0 GHz Quad-Core 2.5+ GHz

RAM : 4 GB 8 GB or higher

Storage: 2 GB free disk space SSD with 10+ GB free space

Display: 1024x768 resolution Full HD (1920x1080) or higher

Internet: Required for installation & API calls Stable broadband connection

2.2 Software Requirements

Operating System: Windows 10/11, Linux (Ubuntu), or macOS

Frontend Tools: Node.js (v18+), Vite, React, TypeScript

Backend Framework: Python 3.10+, Flask

Database: MySQL 9.x

Package Manager: npm (for frontend), pip (for backend)

Browser: Google Chrome, Firefox, or Edge

Code Editor: Visual Studio Code or any modern IDE

Other Tools: Postman (API testing), Git (version control)

Design

Schema Diagram

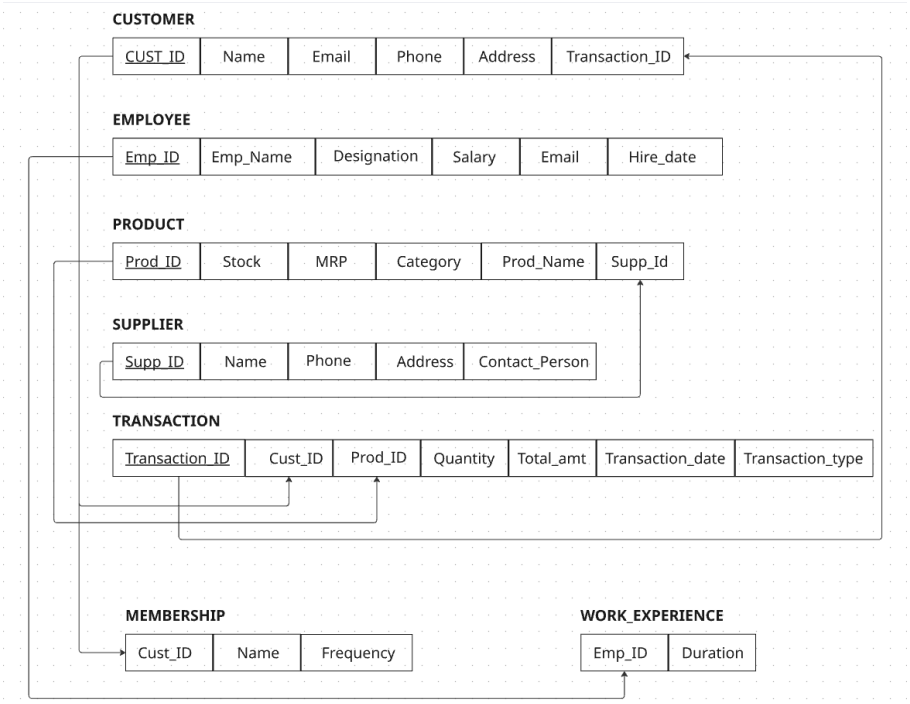

Figure 3.1 – Schema

ER Diagram

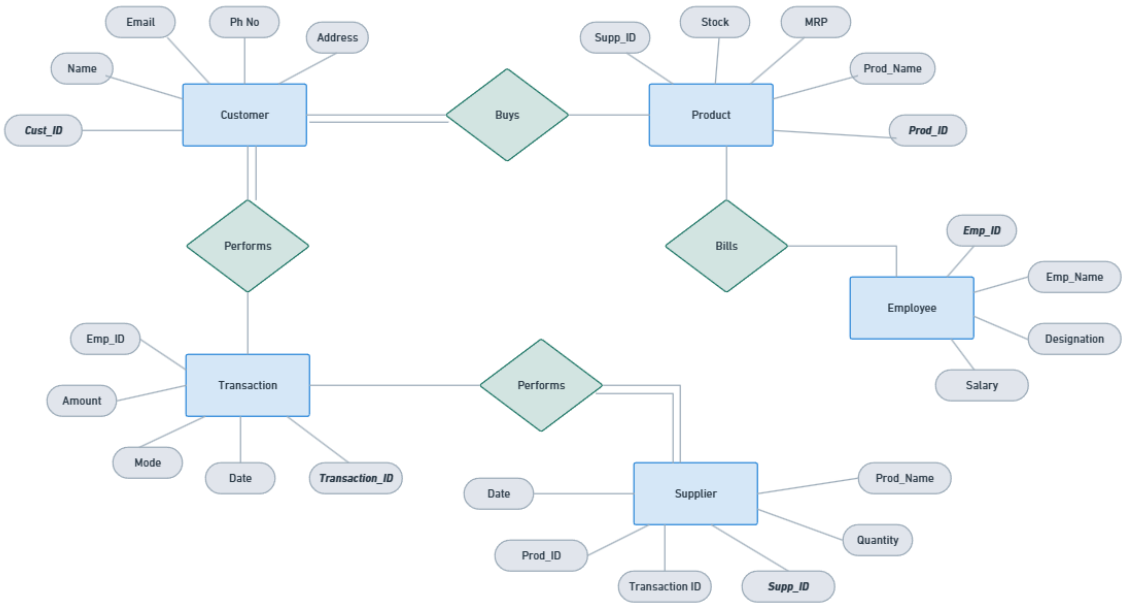

Figure 3.2 – ER Diagram

Implementation (Code)

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from db import get_connection

from flask import Flask, jsonify
from flask_cors import CORS
import mysql.connector

app = Flask(__name__)
CORS(app)

def get_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="9448",
        database="shop"
    )

@app.route('/api/recent-transactions', methods=['GET'])
def get_recent_transactions():
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT transaction_id, total_amount, transaction_date
        FROM transaction
        ORDER BY transaction_date DESC
        LIMIT 3
    """)
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    return jsonify(data)

@app.route('/api/low-stock', methods=['GET'])
def get_low_stock():
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT prod_name, stock
        FROM product
```

```
        WHERE stock < 10
        ORDER BY stock ASC
        LIMIT 3
    """)
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    return jsonify(data)

@app.route('/')
def home():
    return "Backend is running!"

@app.route('/customers', methods=['GET'])
def get_customers():
    try:
        conn = get_connection()
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM customer")
        data = cursor.fetchall()
        cursor.close()
        conn.close()
        return jsonify(data)
    except Exception as e:
        return jsonify({"error": str(e)})

@app.route('/customers', methods=['POST'])
def add_customer():
    try:
        data = request.get_json()
        conn = get_connection()
        cursor = conn.cursor()
        cursor.execute("""
            INSERT INTO customer (cust_id, name, email, address, ph_no, transaction_id)
            VALUES (%s, %s, %s, %s, %s, %s)
        """, (data['cust_id'], data['name'], data['email'], data['address'], data['ph_no'], data['transaction_id']))
        conn.commit()
        cursor.close()
        conn.close()
        return jsonify({"message": "Customer added successfully"})
    except Exception as e:
        return jsonify({"error": str(e)})

@app.route('/customers/<int:cust_id>', methods=['PUT'])
def update_customer(cust_id):
    data = request.get_json()
```

```
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("""
        UPDATE customer
        SET name=%s, email=%s, address=%s, ph_no=%s, transaction_id=%s
        WHERE cust_id=%s
    """, (
        data['name'],
        data['email'],
        data['address'],
        data['ph_no'],
        data.get('transaction_id'),
        cust_id
    ))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": "Customer updated"})

@app.route('/customers/<int:cust_id>', methods=['DELETE'])
def delete_customer(cust_id):
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM customer WHERE cust_id = %s", (cust_id,))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": "Customer deleted"})


# -------------------------
# PRODUCT ROUTES
# -------------------------

@app.route('/products', methods=['GET'])
def get_products():
    conn = get_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("SELECT * FROM product")
    data = cursor.fetchall()
    cursor.close()
    conn.close()
    return jsonify(data)

@app.route('/products', methods=['POST'])
def add_product():
```

```
    data = request.get_json()
    print("🛠️ Product data received from frontend:", data)

    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO product (prod_id, prod_name, mrp, stock, category, supplier_id)
        VALUES (%s, %s, %s, %s, %s, %s)
    """, (
        data['prod_id'],
        data['prod_name'],
        data['mrp'],
        data['stock'],
        data.get('category', ''),     # <- safe!
        data['supplier_id']
    ))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": "Product added"})

@app.route('/products/<int:prod_id>', methods=['PUT'])
def update_product(prod_id):
    data = request.get_json()
    conn = get_connection()
    cursor = conn.cursor()
    cursor.execute("""
        UPDATE product
        SET prod_name=%s, mrp=%s, stock=%s, category=%s, supplier_id=%s
        WHERE prod_id=%s
    """, (
        data['prod_name'],
        data['mrp'],
        data['stock'],
        data['category'],
        data['supplier_id'],
        prod_id
    ))
    conn.commit()
    cursor.close()
    conn.close()
    return jsonify({"message": "Product updated"})

@app.route('/products/<int:prod_id>', methods=['DELETE'])
def delete_product(prod_id):
    conn = get_connection()
```

```
cursor = conn.cursor()
cursor.execute("DELETE FROM product WHERE prod_id = %s", (prod_id,))
conn.commit()
cursor.close()
conn.close()
return jsonify({"message": "Product deleted"})
```

Interpretation of Results

MySQL Output:

```
+----------------+
| Tables_in_shop |
+----------------+
| customer       |
| employee       |
| product        |
| supplier       |
| transaction    |
+----------------+
```

Table 1.1 – List of tables

```
+-----------+---------------+------+-----+---------+--------+
| Field     | Type          | Null | Key | Default | Extra  |
+-----------+---------------+------+-----+---------+--------+
| cust_id   | int           | NO   | PRI | NULL    |        |
| name      | varchar(100)  | YES  |     | NULL    |        |
| frequency | int           | YES  |     | 0       |        |
+-----------+---------------+------+-----+---------+--------+
```

Table 1.2 – Membership table

```
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| cust_id        | int         | NO   | PRI | NULL    |       |
| name           | varchar(50) | YES  |     | NULL    |       |
| email          | varchar(100)| YES  |     | NULL    |       |
| address        | varchar(255)| YES  |     | NULL    |       |
| ph_no          | varchar(15) | YES  |     | NULL    |       |
| transaction_id | int         | YES  | MUL | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

Table 1.3 – Customer table

```
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| emp_id    | int           | NO   | PRI | NULL    |       |
| name      | varchar(100)  | YES  |     | NULL    |       |
| position  | varchar(100)  | YES  |     | NULL    |       |
| salary    | decimal(10,2) | YES  |     | NULL    |       |
| email     | varchar(100)  | YES  |     | NULL    |       |
| hire_date | date          | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
```

Table 1.4 – Employee table

```
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| emp_id   | int         | NO   | PRI | NULL    |       |
| duration | varchar(100)| YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

Table 1.5 – Work_experience table

Field	Type	Null	Key	Default	Extra
prod_id	int	NO	PRI	NULL	
prod_name	varchar(50)	YES		NULL	
mrp	decimal(10,2)	YES		NULL	
stock	int	YES		NULL	
category	varchar(100)	YES		NULL	
supplier_id	int	YES		NULL	

Table 1.6 – Product table

Field	Type	Null	Key	Default	Extra
supplier_id	int	NO	PRI	NULL	
name	varchar(100)	YES		NULL	
contact_person	varchar(100)	YES		NULL	
email	varchar(100)	YES		NULL	
phone	varchar(20)	YES		NULL	
address	text	YES		NULL	

Table 1.7 – Supplier table

Field	Type	Null	Key	Default	Extra
transaction_id	int	NO	PRI	NULL	
cust_id	int	YES	MUL	NULL	
prod_id	int	YES	MUL	NULL	
quantity	int	YES		NULL	
total_amount	decimal(10,2)	YES		NULL	
transaction_date	date	YES		NULL	
transaction_type	varchar(50)	YES		NULL	

Table 1.8 – Transaction table

Interface Output:

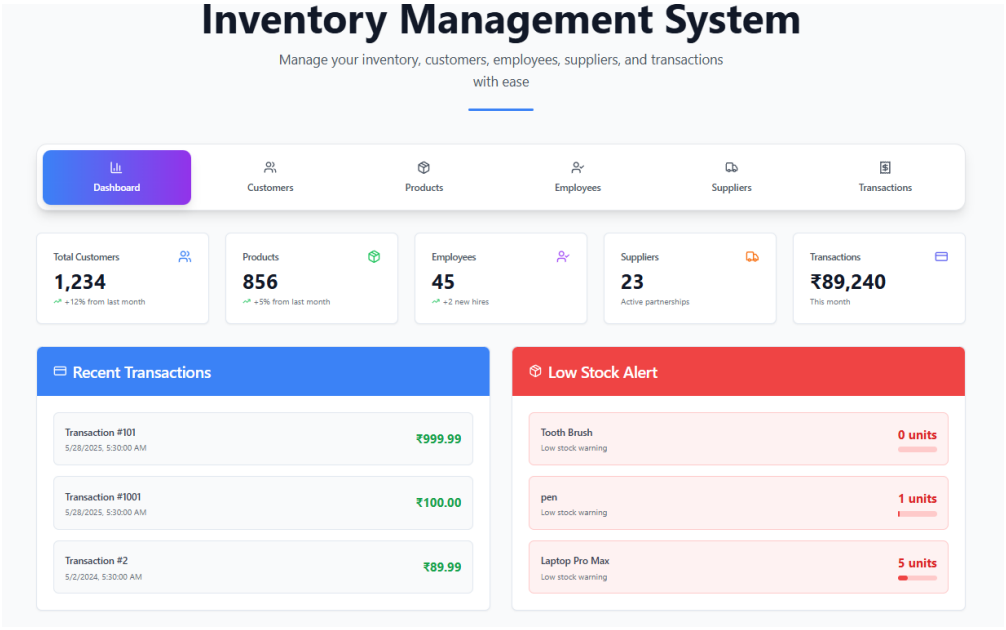

Figure 5.1 – Dashboard

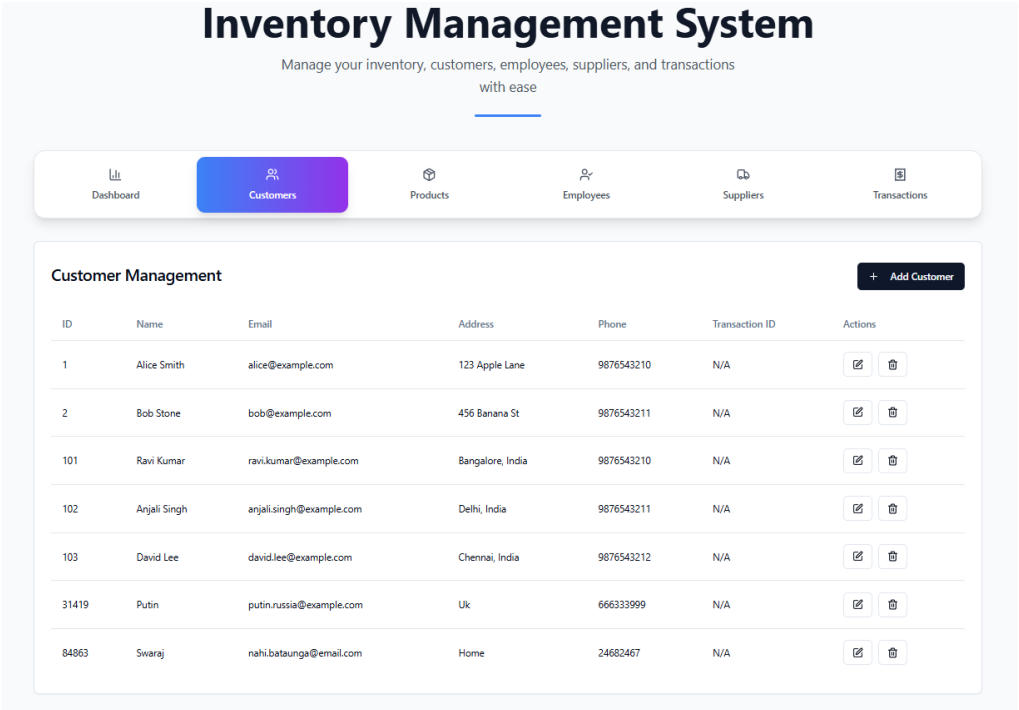

Figure 5.2 – Customer Details

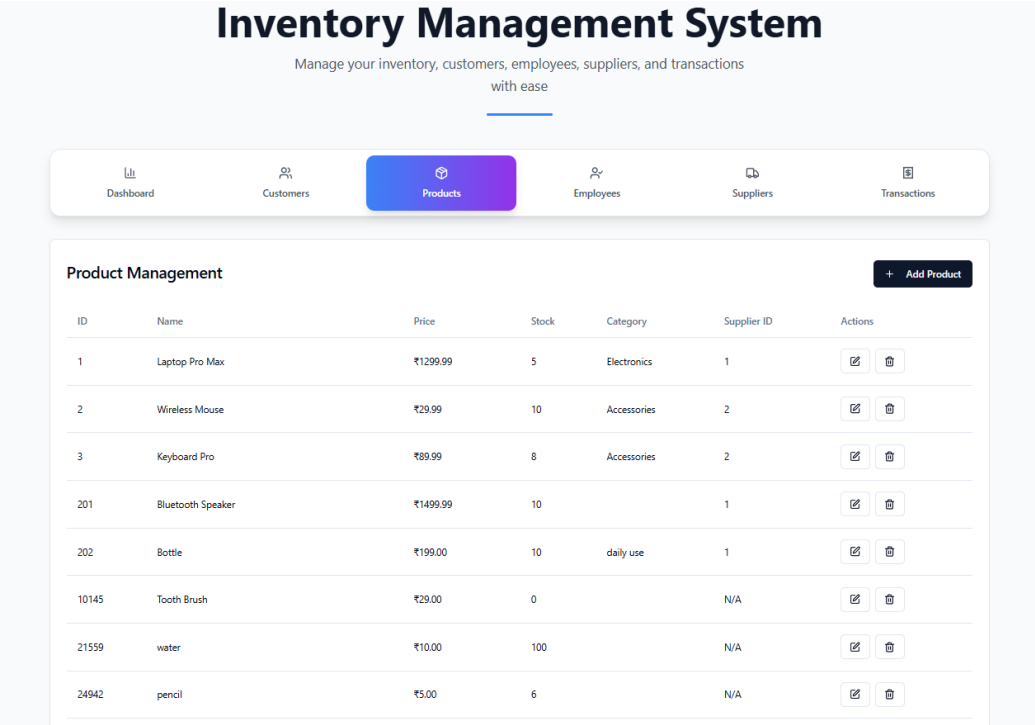

Figure 5.3 – Product Details

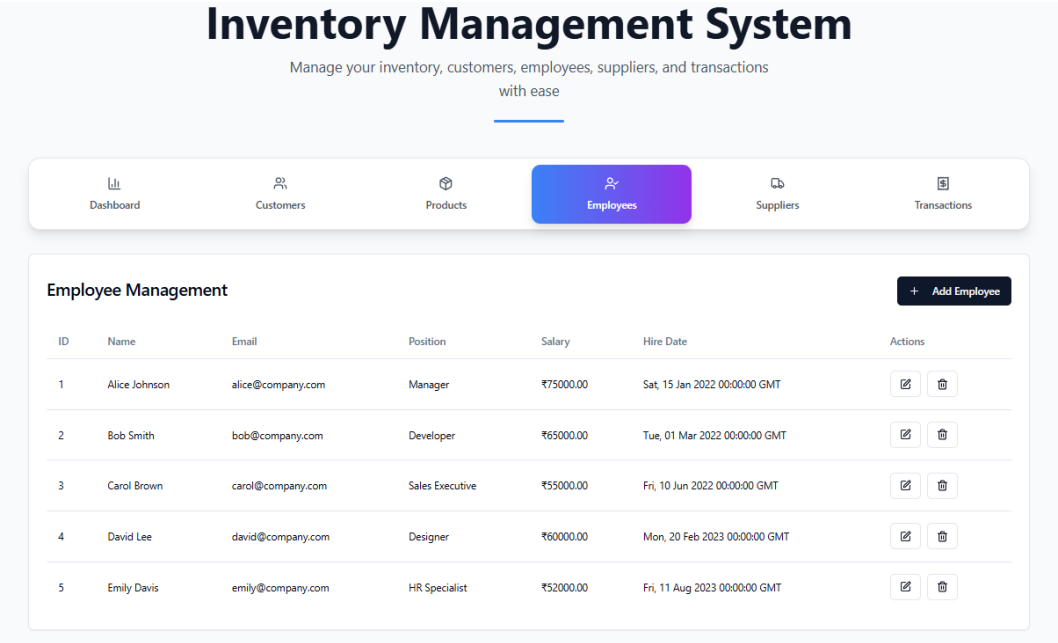

Figure 5.4 – Employee Details

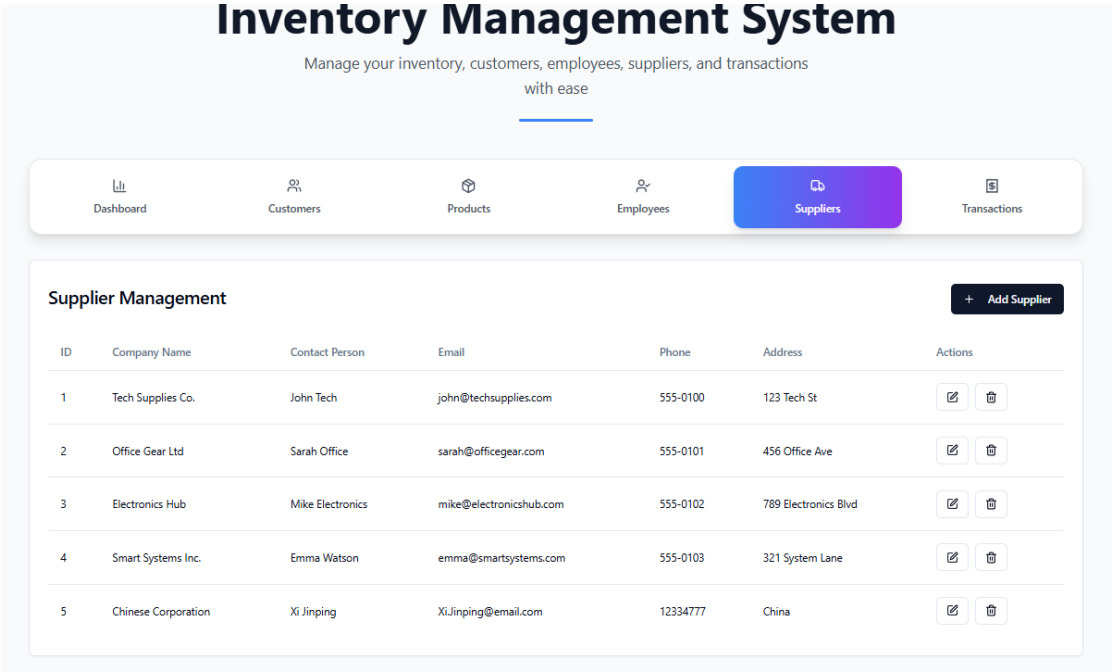

Figure 5.5 – Supplier Details

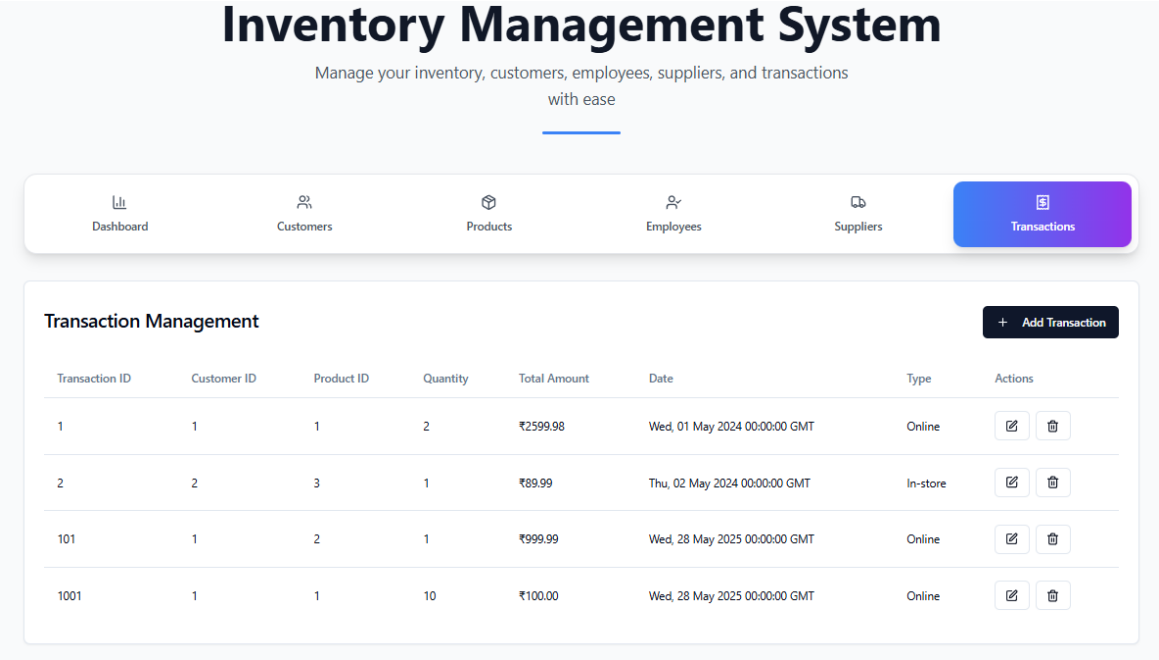

Figure 5.6 – Transaction Details

Conclusion and Future Scope

Conclusion:

The Inventory Management System developed in this project offers a comprehensive and streamlined solution for handling key business operations such as customer management, product tracking, employee oversight, supplier coordination, and transaction monitoring. By integrating a Flask-based backend with a React and Vite-powered frontend, the application ensures a responsive and user-friendly interface along with robust backend functionality. The use of MySQL as the database enhances data integrity and performance, making the system scalable and reliable for real-world usage.

This system allows real-time tracking of stock levels, generates alerts for low inventory, and maintains a complete log of transactions—facilitating efficient decision-making and resource management. With role-based modules for customers, employees, and suppliers, the platform fosters transparency and operational control. The project has successfully demonstrated how a full-stack application can optimize inventory workflows in small to medium-scale businesses.

Future Scope:

1. **Advanced Analytics & Reporting:**
   Integration of analytics dashboards using tools like Chart.js or Power BI could provide deeper insights into sales trends, inventory turnover, and customer behaviour.
2. **Barcode & QR Code Integration:**
   Automating product identification using barcode or QR code scanners would further enhance stock management and checkout speed.
3. **Multi-user Role-Based Access Control (RBAC):**
   Implementing secure login and differentiated access based on user roles (admin, manager, employee) would improve data privacy and control.
4. **Mobile Application Support:**
   A mobile version of the system can be developed using React Native or Flutter to enable on-the-go inventory monitoring and updates.
5. **Cloud Deployment & CI/CD Integration:**
   Hosting the system on platforms like AWS or Azure with continuous deployment pipelines can allow for real-time collaboration, backups, and better scalability.
6. **AI-Powered Demand Forecasting:**
   Incorporating machine learning models to predict product demand based on past data can help in proactive restocking and minimizing losses.

References

1. MySQL Documentation. MySQL 8.0 Reference Manual. Available at: https://dev.mysql.com/doc/
2. Flask Documentation. Flask API Reference. Available at: https://flask.palletsprojects.com/
3. React Documentation. React Official Docs. Available at: https://react.dev/
4. Vite Documentation. Vite Guide and Config Reference. Available at: https://vitejs.dev/
5. W3Schools. React JS Tutorial & MySQL Integration. Available at: https://www.w3schools.com/
6. Mozilla Developer Network (MDN). HTML, CSS, JavaScript Resources. Available at: https://developer.mozilla.org/
7. GeeksforGeeks. Creating RESTful APIs with Flask and MySQL. Available at: https://www.geeksforgeeks.org/
8. Lucide Icons Documentation. Lucide Icon Set for React. Available at: https://lucide.dev/
9. Axios Documentation. Axios HTTP Client Reference. Available at: https://axios-http.com/