PES UNIVERSITY, Bengaluru
Department of Computer Science and Engineering
B. Tech (CSE) – 5th Semester – Aug-Dec 2024

UE22CS341A – Software Engineering

PROJECT REPORT
on

Project Title

Submitted by: Team # <SRN1-SRN2>

PES1UG22CS321	Mahanth Kumar
PES1UG22CS321	Manish Meghashyam Naik

Class of Prof. Raghu B. A.

5th Sem. F Sec.

Table of Contents

1. Project Statement / Synopsis

The project, *Store Management System*, aims to provide a comprehensive database solution for managing branches, items, customers, employees, and suppliers in a retail chain. It includes CRUD operations, transaction tracking, stock management, and advanced database features like triggers, stored procedures, and views. A front-end using Streamlit enhances usability by enabling interactive access.

Scope:

This mini-project models a real-time store's operations:

1. **Stock Management: Tracks items across 4 branches, including supplier details and stock updates.**
2. **Transaction Monitoring: Manages customer purchases, including dynamic updates to bills and orders.**
3. **Employee Records: Tracks employee details and links them with specific transactions and branches.**
4. **Automation: Uses triggers and procedures to ensure data consistency and automated backups.**

---

2. Software Requirements Specification (SRS) with RTM (Initial Version)

- **Purpose:**
  The SRS outlines functional and non-functional requirements for developing the Store Management System, targeting developers, managers, and end users.
- **Features:**
    1. **Inventory Management:**

- **Add, update, view, and delete items, tracking quantities per branch.**

2. **Customer and Employee Management:**
    - **Maintain detailed records, including personal and transactional data.**
3. **Order Processing:**
    - **Handle order creation, updates, and payment tracking.**
4. **Dynamic Reporting:**
    - **Generate sales, inventory, and transaction reports dynamically using SQL queries and views.**

• Sl.	Require-ment	Brief	Architect	Design	Code	Test	System Test

No.	ID	Description of Requirement	ure Refer-ence	Reference	File Reference	Case ID	Case ID
1	REQ-001	Customer Registration	ERD CUSTOMERS Entity	CustDesign001	CustCode001	TC_001	STC_001
2	REQ-002	Item Management (CRUD Operations)	ERD ITEMS Entity	ItemDesign002	ItemCode002	TC_002	STC_002
3	REQ-003	Bill Generation and Tracking	ERD BILL Entity	BillDesign003	BillCode003	TC_003	STC_003

4	REQ-004	Supplier and Shipment Management	ERD SUPPLIERS & SHIPMENT	SupplyDesign004	SupplyCode004	TC_004	STC_004
5	REQ-005	Employee Management and Super-vision	ERD EMPLOYEE Entity	EmpDesign005	EmpCode005	TC_005	STC_005

3. Project Plan with Gantt Chart (Baseline)

Life-cycle followed

The project will follow the Agile model. Agile is chosen because of its iterative nature, allowing for continuous feedback and improvements. Since the project has several independent modules (like database design, CRUD operations, triggers, and frontend development), Agile is a good fit as it allows for concurrent development and continuous validation with stakeholders.

Justification:

- **Agile allows for frequent iterations, which helps in refining the product as feedback is received.**
- **Each functionality (like database schema, CRUD operations, or frontend) can be developed in sprints, allowing progress even if some parts are not fully defined at the beginning.**

Work Breakdown Structure

- **Phase 1**: Planning & Requirements Analysis (~0.5 months)
    - Defining scope and requirements, finalizing the list of entities.
- **Phase 2**: Database Design (~0.5 months)
    - ER diagram creation, defining relationships between entities, and building relational schema.
- **Phase 3**: Database Setup (~1 month)
    - Writing DDL for tables, constraints, and indexes.
    - Populating sample data.
- **Phase 4**: Backend Development (~1 month)
    - Writing triggers, procedures, and functions.
- **Phase 5**: Frontend Development (~1 month)
    - Developing the Streamlit interface for CRUD operations.
- **Phase 6**: Testing (~0.5 months)
    - Testing the database with SQL queries and validating the frontend functionality.
- **Phase 7**: Final Deployment & Debugging (~0.5 months)
    - Fixing bugs, final integration, and deploying the project.

Task	Month-1	Month-2	Month-3	Month-4
Planning & Requirements				
Database Design				
Database Setup				
Backend Development				
Frontend Development				
Testing				
Final Deployment				

4. Architecture & Design Choices and Diagrams

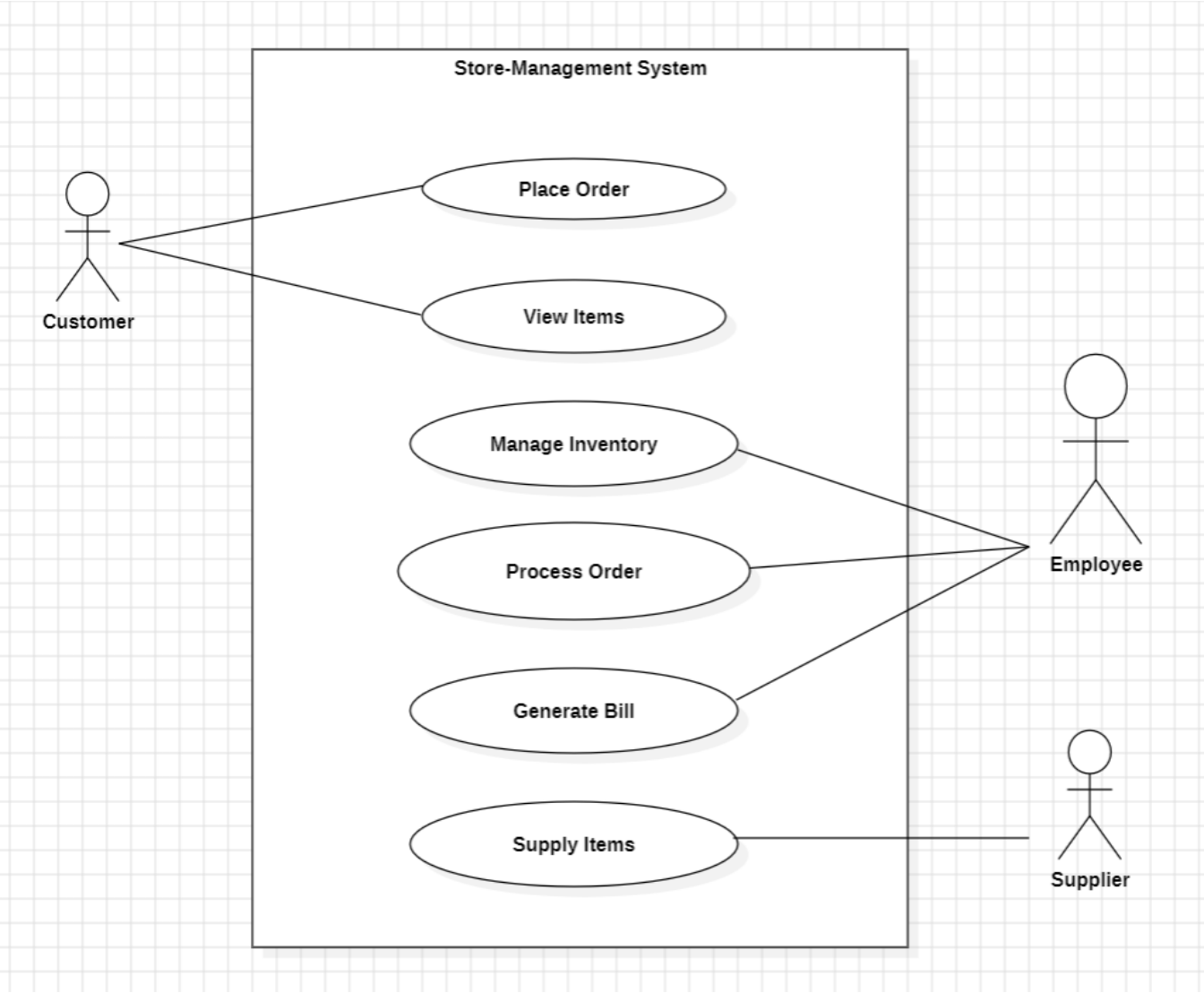
Architectural Design:

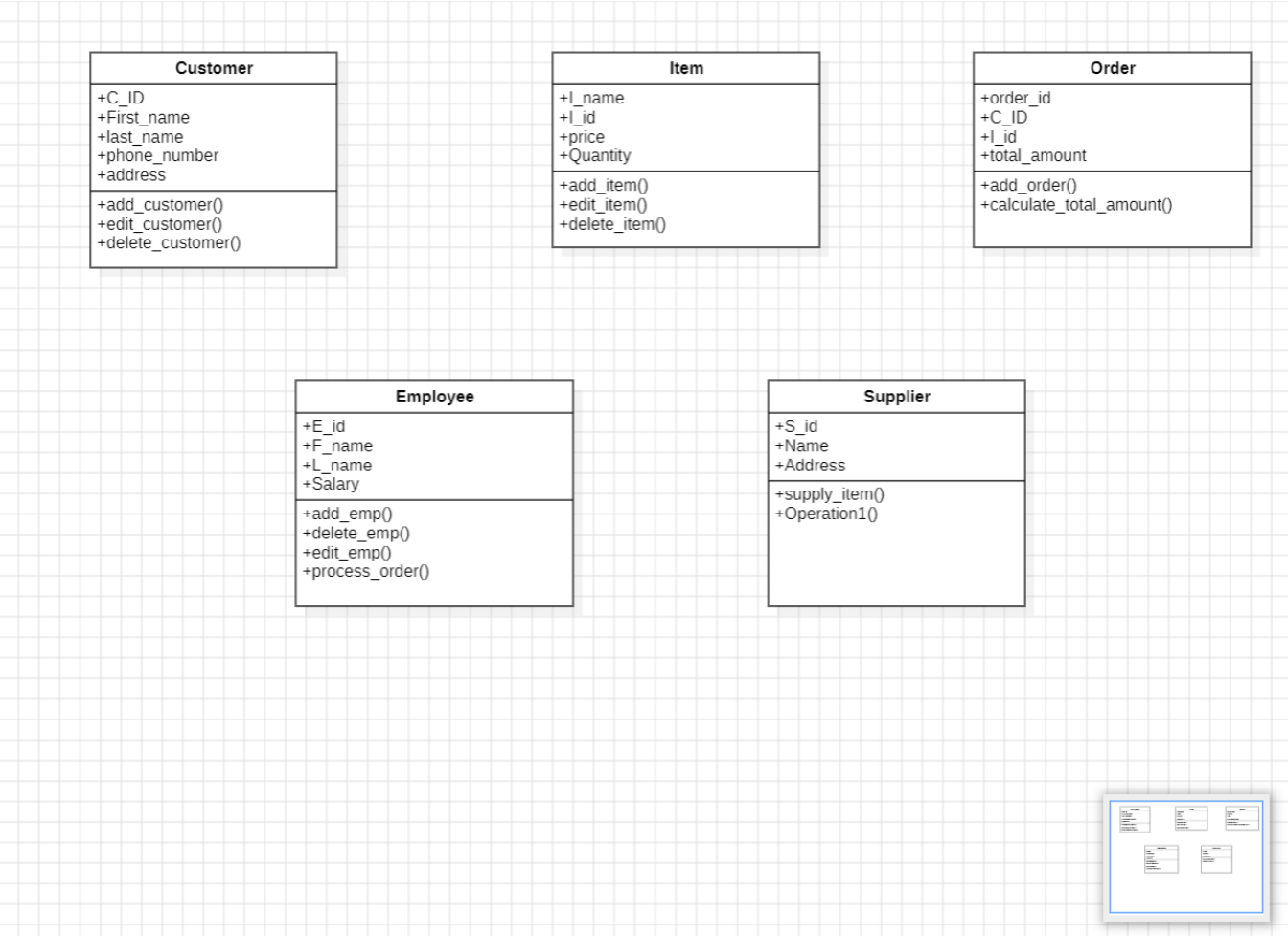
*The architectural design of the system follows a **Three-Tier Architecture**:*

- ***Presentation Layer**: The frontend built using **Streamlit** interacts with the user. Users can perform CRUD operations on customers, employees, orders, and items.*
- ***Application Layer**: This layer includes Python scripts that handle the business logic of the system, such as processing orders, calculating total bill amounts, and managing inventory.*
- ***Data Layer**: The **MySQL Database** stores all the data related to customers, employees, items, orders, and suppliers.*

***Modules**:*

- ***Customer Module**: Manages CRUD operations for customers.*
- ***Inventory Module**: Handles item management.*
- ***Order Processing Module**: Validates and processes orders, interacts with the inventory to update stock.*
- ***Supplier Module**: Manages suppliers and shipment information.*
- ***Billing Module**: Generates and tracks invoices for customer orders.*


Store-Management System
Place Order
View Items
Manage Inventory
Process Order
Generate Bill
Supply Items
Customer
Employee
Supplier



5. Development - Code Files [https://github.com/Mahanthb/STORE-MANAGEMENT-SYSTEM-DBMS]

- **GitHub Repository:**
  - **SQL scripts for database schema, triggers, and stored procedures.**
  - **Python files for Streamlit application (e.g., app.py, create.py, update.py).**

6. Test Plans

- **Objectives:**
  Validate functionality for:
  - **CRUD operations.**
  - **Triggers (e.g., quantity constraints).**
  - **Stored procedures and views.**
- **Types of Tests:**
  - **Unit Testing: Validating individual SQL queries and Python modules.**
  - **Integration Testing: Ensuring front-end and database communication works correctly.**

---

7. Test Cases and Test Results Matrix

Test Cases with Results:

Test Case ID	Name of Module	Test Case Description	Pre-conditions	Test Steps	Test Data	Expected Results	Actual Results	Test Result
UT-01	Customer Registration	Verify customer addition functionality	Database connection	1. Open the form 2. Enter customer details 3. Click 'Add'	ID: 2001, Name: John, Phone: 123456	Customer added with ID 2001 and data visible in DB	Customer added successfully and data is visible in DB	Pass

UT-02	Order Management	Validate adding a new order	Customer exists	1. Open order form 2. Select customer ID 3. Add items to the order 4. Submit	ID: 2001, Item: Shirt, Qty: 3	Order created, total calculated, and visible in order table	Order created successfully, visible in order table	Pass
UT-03	Billing Module	Calculate total bill for customer	Order exists	1. Fetch all orders by customer ID 2. Sum item prices * quantity 3. Generate bill entry	Customer ID: 2001	Bill generated with total amount shown	Bill generated accurately with correct total	Pass
IT-01	Order & Customer	Link order creation with existing customer	Customer in system	1. Initiate an order 2. Enter valid customer ID 3. Verify customer data in	Customer ID: 2001, Item: 4001	Order linked to customer with details intact	Customer and order linked successfully	Pass

				order view				
IT-02	Billing & Order	Validate bill calculation on new order	Existing orders	1. Add new order 2. Generate bill 3. Verify updated total in bill	Customer ID: 2001	Bill updated to reflect new order amount	Bill updated accurately with new order amount	Pass
IT-03	Inventory Management	Validate inventory decrease on order	Item stock available	1. Place an order for an item 2. Check stock level of item after order completion	Item: 4001, Qty before: 10, Qty order: 2	Inventory decreases by ordered quantity	Stock level decreased by ordered amount	Pass

ST-01	Customer & Order Module	Verify end-to-end flow for order placing	New customer creation	1. Add new customer 2. Add order for customer 3. Generate bill 4. Validate all stages	Customer ID: 2021, Items: 4001, 4002	Seamless flow from registration to order to bill	Successful flow, all modules integrated	Pass
ST-02	Billing & Inventory	Validate total sales calculation in bill	Orders exist in system	1. Generate a report of total sales per month 2. Validate with individual bills	Month: September	Total matches sum of individual bills	Total sales calculated accurately	Pass
ST-03	Customer Deletion	Validate backup creation on deletion	Customer exists	1. Delete a customer 2. Check backup log 3. Verify deletion from main database, preservat	Customer ID: 2001	Customer removed from main table, logged in backup	Customer removed, backup created as expected	Pass

				ion in backup table				
ST-04	**User Permissions**	**Ensure restricted access to database**	**Admin credentials**	**1. Try to access DB without login 2. Attempt to modify tables without admin role**	**Unauthorized user**	**Access denied, permission error displayed**	**Access restricted, permission error confirmed**	**Pass**

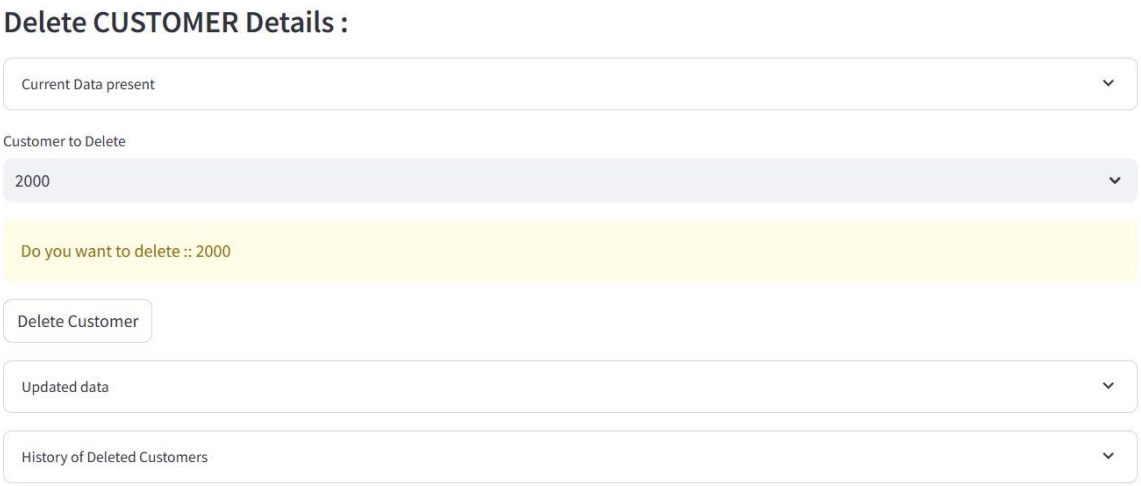
Count of Passed Test Cases = _____10_____

Count of Failed Test Cases = _____0_____

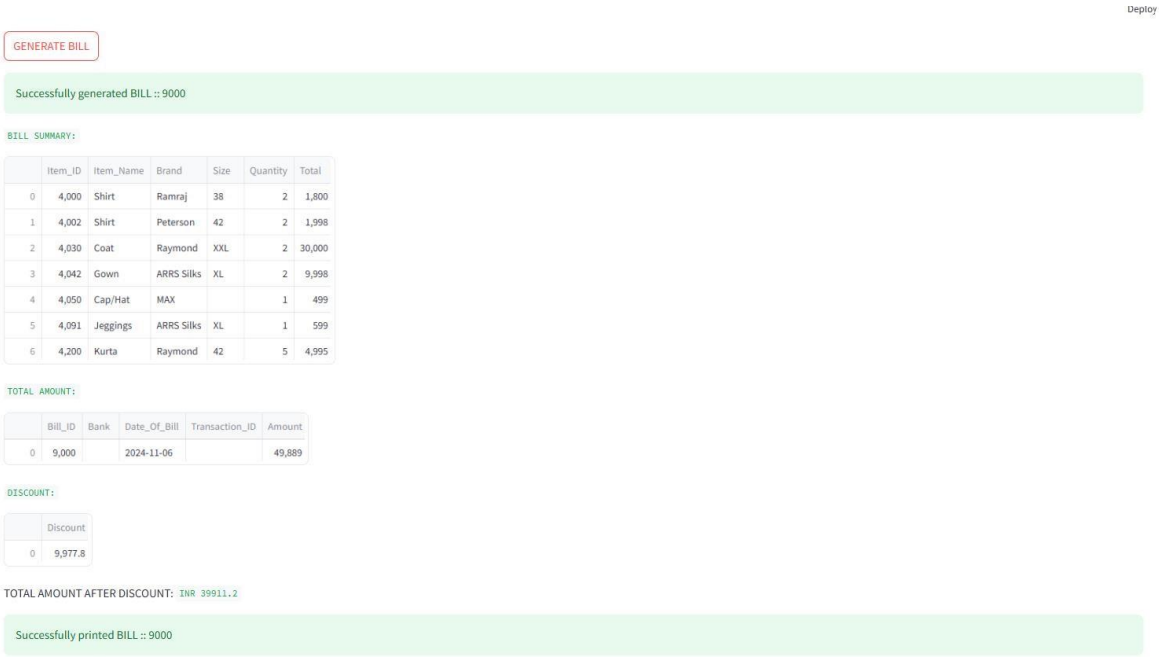
Total Number of Test Cases = _____10_____

SOME IMPORTANT TEST CASES SCREENSHOTS:

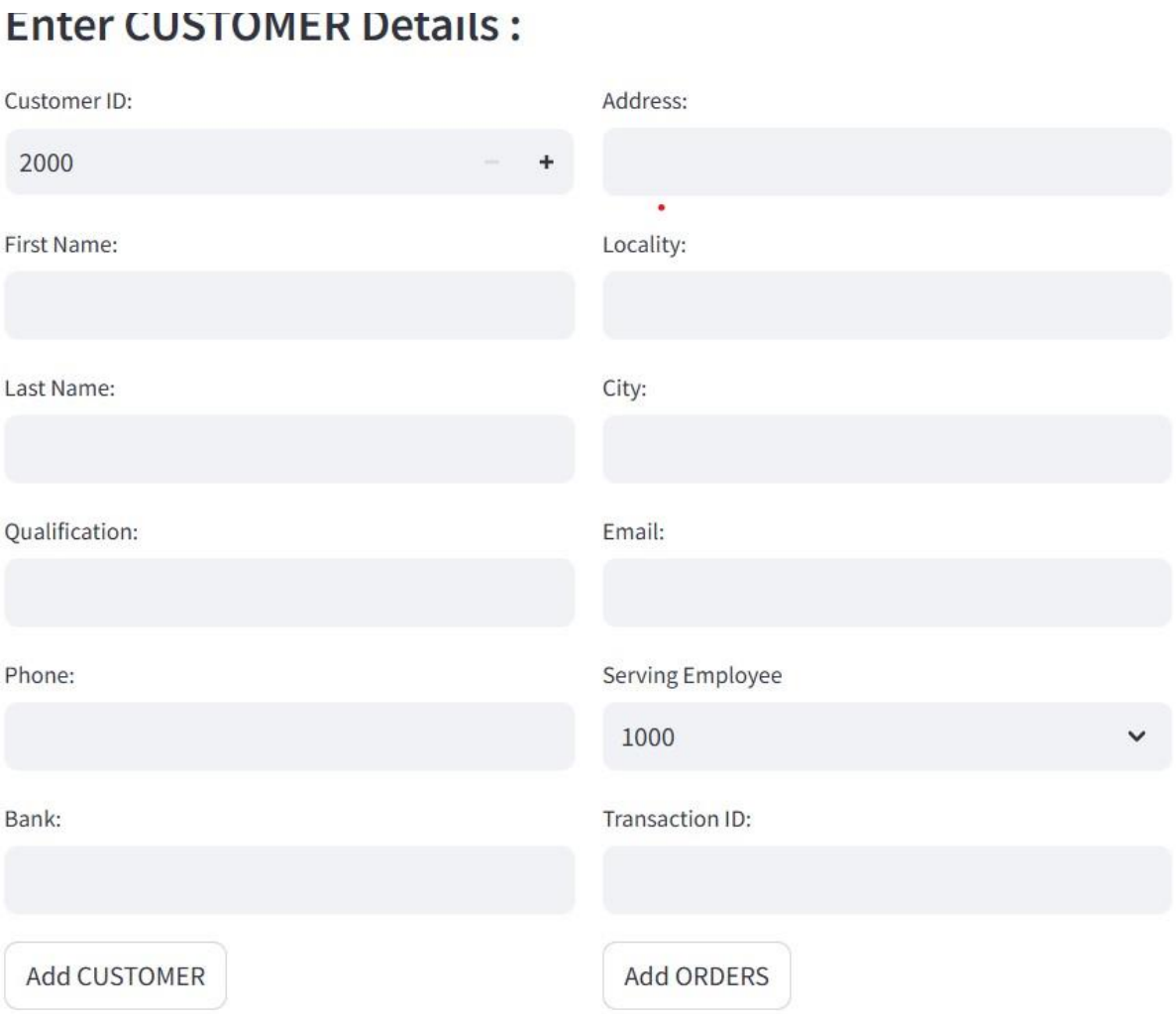
ST-3:


Delete CUSTOMER Details :
Current Data present
Customer to Delete
2000
Do you want to delete :: 2000
Delete Customer
Updated data
History of Deleted Customers

UT-3:


Deploy
GENERATE BILL
Successfully generated BILL :: 9000
BILL SUMMARY:
Item_ID Item_Name Brand Size Quantity Total
0 4,000 Shirt Ramraj 38 2 1,800
1 4,002 Shirt Peterson 42 2 1,998
2 4,030 Coat Raymond XXL 2 30,000
3 4,042 Gown ARRS Silks XL 2 9,998
4 4,050 Cap/Hat MAX 1 499
5 4,091 Jeggings ARRS Silks XL 1 599
6 4,200 Kurta Raymond 42 5 4,995
TOTAL AMOUNT:
Bill_ID Bank Date_Of_Bill Transaction_ID Amount
0 9,000 2024-11-06 49,889
DISCOUNT:
Discount
0 9,977.8
TOTAL AMOUNT AFTER DISCOUNT: INR 39911.2
Successfully printed BILL :: 9000

UT-1:



8. Final Gantt Chart

Task	Month-1	Month-2	Month-3	Month-4
Planning & Requirements				
Database Design				
Database Setup				

Backend Development				
Frontend Development				
Testing				
Final Deployment				

9. Conclusions

The *Store Management System* project represents a comprehensive effort to simulate and automate the operations of a real-world store chain using advanced database techniques and a user-friendly interface. The successful completion of this project has not only demonstrated the effectiveness of relational database management systems (RDBMS) in handling complex data structures but also showcased the practical application of front-end development tools like Streamlit to make these systems accessible and interactive.

The primary achievement of this project lies in its ability to seamlessly integrate various aspects of store operations, including inventory management, customer transactions, employee tracking, and supplier relationships. The database is equipped with robust functionality, featuring relational schemas, constraints, triggers, and procedures that ensure data consistency and integrity. For example, triggers have been implemented to enforce business rules, such as limiting item stock quantities, while stored procedures automate repetitive tasks like calculating customer loyalty levels based on their purchase history. These features significantly reduce manual effort and the likelihood of errors, making the system reliable and efficient.

Another noteworthy accomplishment is the design and deployment of a dynamic front-end using Streamlit. The interface allows users to perform CRUD operations on key entities such as customers, employees, and items, as well as execute custom queries for detailed insights.

The inclusion of interactive visualizations and dynamic forms has greatly enhanced usability, catering to non-technical users and improving overall accessibility. For instance, a store manager can effortlessly generate bills, update stock records, or analyze sales performance through a few clicks, without needing extensive technical expertise.

Despite these achievements, the project encountered several challenges during its development. One significant challenge was debugging the triggers and procedures to ensure they operated correctly across all edge cases. For instance, the trigger designed to prevent overstocking items required multiple iterations to handle various insertion scenarios. Similarly, designing complex queries for reports, such as listing customers who purchased specific combinations of items, tested the team's understanding of SQL operations like joins, unions, and intersections.

The integration of the database with Streamlit presented another challenge, especially in ensuring the real-time synchronization of the front-end with the back-end. Handling errors gracefully, such as invalid inputs or database connectivity issues, required careful exception handling and user feedback mechanisms. These challenges provided valuable learning experiences, enhancing problem-solving skills and a deeper understanding of database systems and Python-based web development.

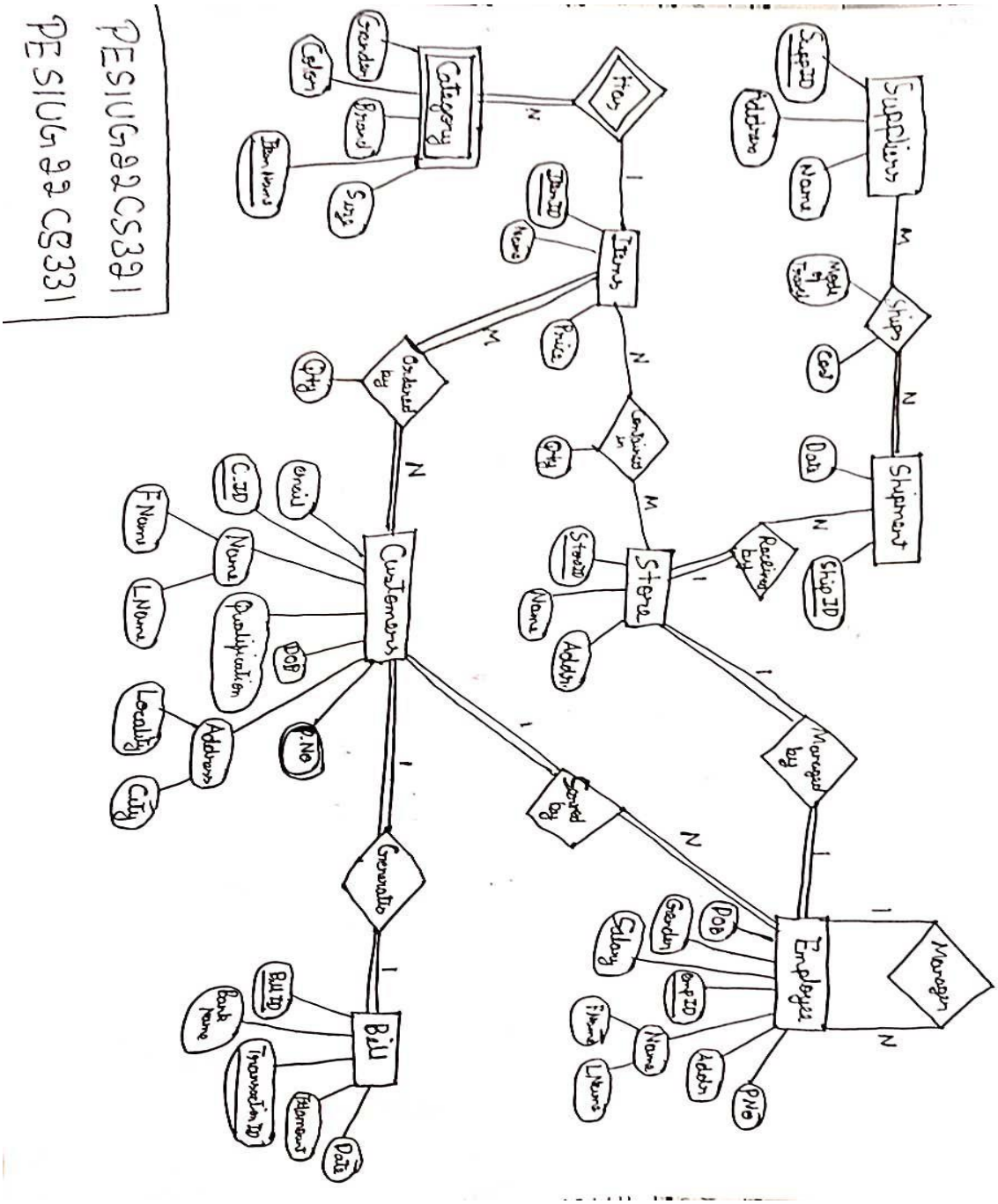
Looking ahead, this project has substantial scope for expansion and improvement. The system can be scaled to support larger chains with multiple branches, possibly integrating real-time analytics using advanced tools like Power BI or Tableau. Additionally, incorporating machine learning algorithms could enable predictive analytics, such as forecasting inventory needs or identifying customer purchasing patterns. Another potential enhancement is the integration of mobile applications to provide on-the-go access for managers and customers alike.

In conclusion, the *Store Management System* stands as a testament to the power of database-driven solutions in solving real-world problems. It not only meets the project's original objectives but also lays a strong foundation for future enhancements. The knowledge and skills gained through this project are invaluable, providing practical insights into database management, front-end development, and the challenges of creating end-to-end solutions.

This project serves as a stepping stone for more ambitious endeavors in the realm of database systems and enterprise application development.

10. Appendix A: Glossary of Abbreviations and Acronyms

- **CRUD: Create, Read, Update, Delete.**
- **RTM: Requirements Traceability Matrix.**
- **E-R Diagram: Entity-Relationship Diagram.**



11. Appendix B: RTM (Final Version)

Sl.	Requirement	Brief	Architect	Design	Code	Test	System Test

No.	ID	Description of Requirement	ure Refer-ence	Reference	File Reference	Case ID	Case ID
1	REQ-001	Customer Registration	ERD CUSTOMERS Entity	CustDesign001	CustCode001	TC_001	STC_001
2	REQ-002	Item Management (CRUD Operations)	ERD ITEMS Entity	ItemDesign002	ItemCode002	TC_002	STC_002
3	REQ-003	Bill Generation and Tracking	ERD BILL Entity	BillDesign003	BillCode003	TC_003	STC_003
4	REQ-004	Supplier and Shipment Management	ERD SUPPLIERS & SHIPMENT	SupplyDesign004	SupplyCode004	TC_004	STC_004
5	REQ-005	Employee Management and Super-vision	ERD EMPLOYEE Entity	EmpDesign005	EmpCode005	TC_005	STC_005

12. Appendix C: Technology Stack and References

- **Technology Stack:**

- **Backend: MySQL relational database.**
- **Frontend: Streamlit for web interface.**
- **Language: Python for application logic.**

- **References:**

1. **MySQL Documentation**

2. **Streamlit Documentation**