FreshWash Laundry Management System

CS 432 Assignment 1 Report

https://github.com/Vedant-VB07/Databases_Assignment_1

February 15, 2026

Abstract

This report details the design and implementation of **FreshWash**, a comprehensive database-driven software system aimed at modernizing laundry service operations. Addressing the inefficiencies of manual record-keeping—such as misplaced orders and revenue leakage—we propose a robust solution grounded in strict database theory. The project follows a systematic **"UML-first"** approach, transitioning from high-level conceptual modeling to a normalized Entity-Relationship (ER) diagram, and finally to a physical MySQL schema. The resulting system features **13 normalized tables**, enforcing strict referential integrity, real-time order tracking, and dynamic pricing logic, bridging the gap between theoretical database concepts and real-world application.

Contents

1 Introduction

1.1 Problem Statement

The local and medium-scale laundry service industry continues to struggle with the inherent inefficiencies of manual record-keeping. Reliance on physical registers, sporadic phone communication, and paper-based billing frequently results in operational failures, including misplaced orders, billing inaccuracies, and missed pickup schedules.

From a customer perspective, these manual processes create a lack of transparency and reliability. Issues such as lost garments, ambiguous payment details, the absence of feedback mechanisms, and significant service delays contribute to a widening "service quality gap" that prevents modern laundry businesses from scaling effectively.

1.2 Proposed Solution

We propose a comprehensive, database-driven management system designed to eliminate manual inefficiencies by automating the entire lifecycle of laundry operations. The system is built upon a robust relational schema that ensures scalability, data integrity, and real-time operational visibility. To address the identified service quality gaps, **FreshWash** incorporates the following core features:

- **Digital Order Tracking:** Replaces manual logs with a real-time status monitoring system that tracks every order from initial pickup to final delivery.
- **Item-Wise Laundry Processing:** Enables detailed logging of individual garments within a single order to prevent misplaced items and ensure adherence to specific care instructions.
- **Automated Service & Pricing Engine:** Provides a structured menu of services (e.g., Wash, Dry, Iron) with pre-defined pricing to ensure billing accuracy and transparency.
- **Centralized Member Management:** Utilizes a dedicated table to store secure profiles for both customers and staff, including contact details and identification.
- **Payment & Transaction Management:** Monitors payment statuses and maintains a digital trail of all financial transactions to prevent revenue leakage and simplify auditing.
- **Operational Integrity & Validation:** Employs strict relational constraints to ensure that every record—from a staff assignment to a delivery timestamp—is logically sound and verifiable.

1.3 Technologies Used

The design and implementation of the system rely on the following technical stack:

SQL (Structured Query Language): The primary language used for defining the relational schema, enforcing integrity constraints, and performing data manipulation.

MySQL DBMS: Chosen as the core relational engine for its strict enforcement of referential integrity and efficient management of the 13 defined tables.

Modeling Tools (draw.io, TikZ/LaTeX): Employed to generate UML Class Diagrams and formal ER representations, facilitating a systematic "UML-first" design approach.

Development Environment: MySQL Workbench for visual modeling and script execution, alongside VS Code for project specification.

1.4 Key Outcomes

- **Robust Relational Schema:** Implementation of 13 normalized tables, exceeding the minimum requirement to ensure a scalable, redundancy-free structure.
- **Enforced Data Integrity:** Establishment of strict referential integrity via Primary/Foreign Key relationships to maintain consistency across the order lifecycle.
- **Logical Validation:** Utilization of `CHECK` constraints to ensure operational correctness (e.g., preventing delivery dates from preceding pickup times).
- **Operational Visibility:** A digital order status log provides real-time tracking, effectively resolving the "lost garment" issue inherent in manual systems.

2 UML Class Diagram

2.1 Purpose of the UML Class Diagram

The UML Class Diagram was developed as the foundational step in our conceptual modeling process. The primary objective was to identify core system objects, attributes, and relationships prior to translation into a relational database schema. Using a **UML-first** approach allowed us to:

- Clearly define system boundaries and identify entities as real-world objects.
- Capture conceptual relationships before enforcing rigid database constraints.
- Detect and resolve many-to-many associations early in the design phase.
- Ensure a modular and scalable schema design.

Note: The complete, high-resolution source file for the UML Class Diagram is available in the associated GitHub repository detailed in the report header.

2.2 Identification of Classes

We identified the following core classes based on system requirements:

Member: Represents registered users (customers and optional staff). Captures identity attributes: `member_id, name, age, email, contact_number`.

Order: The central transactional entity representing a laundry request. Attributes: `order_id, pickup_time, delivery_time, total_amount`.

Order_Status: Tracks lifecycle stages (e.g., Placed, Washing, Delivered) to enable historical tracking rather than overwriting a single status field.

Service & Clothing_Type: Represents the service catalog (Wash, Dry Clean) and item categories (Shirt, Saree).

Pricing: Defines dynamic pricing rules based on the intersection of Service and Clothing Type.

Payment & Payment_Status: Manages financial transactions and their progress (Initiated, Success, Failed).

Employee: Represents operational staff (Drivers, Washers) handling orders.

Associative Classes: `Order_Service` and `Order_Assignment` resolve many-to-many relationships between Orders, Services, and Employees.

Exceptions: `Feedback` captures user satisfaction, while `Lost_Item` records damaged or missing inventory.

2.3 Relationship Analysis

The diagram incorporates various relationship types to model the domain accurately:

- **One-to-Many (1:M):** Standard parent-child relationships, such as *Member* $\rightarrow$ *Order* and *Order* $\rightarrow$ *Order_Status*.
- **One-to-One (1:1):** Specifically between *Order* and *Payment*, ensuring each order generates exactly one payment record.
- **Many-to-Many (M:N):** Identified between *Order* $\leftrightarrow$ *Service* and *Order* $\leftrightarrow$ *Employee*. These were resolved via associative classes (*Order_Service*, *Order_Assignment*) to ensure proper relational implementation.

2.4 Design Justification

This structured approach ensured clear domain modeling, proper modular grouping (Order Management vs. Staff Management), and a clean transition to the Entity-Relationship diagram.

3 Transition: From UML to ER Model

3.1 Overview of the Transformation

Following the UML design, we systematically transformed the conceptual model into a Chen-style Entity-Relationship (ER) model. The objective was to convert object-oriented abstractions into a relationally implementable database structure while preserving structural constraints and multiplicities.

3.2 Mapping UML Classes to ER Entities

Each UML class was mapped directly to an ER entity. The entities were grouped into logical modules as shown in Table 1.

Table 1: Mapping UML Classes to ER Entities

UML Class	**ER Entity Module**
Member, Employee	Stakeholders
Order, Assignment, Order_Service	Transactional Operations
Service, Clothing_Type, Pricing	Master Data
Payment, Pay_Status, Feedback, Lost_Item	Finance & Exceptions

3.3 Mapping Associations to Relationships

UML associations were converted into diamond-shaped relationships in the ER model:

- **Member – Order:** Mapped as *Member* – $\diamond$ *places* $\diamond$ – *ORDER* with (1:N) cardinality, indicating one member places multiple orders.

- **Order – Service:** The M:N relationship was resolved via the *Order_Service* entity (*ORDER* – ◇ *contains* ◇ – *Order_Service*).
- **Order – Employee:** Resolved via *Assignment*, allowing one employee to handle many assignments and one order to have multiple staff assigned.
- **Order – Payment:** Mapped as a 1:1 relationship where *ORDER* – ◇ *pays* ◇ – *Payment*.

3.4 Translation of Multiplicity

UML multiplicities (e.g., 1, ∗, 1..∗) were translated into ER cardinalities (1, N). Explicit cardinality markers in the ER diagram ensure accurate foreign key placement during schema implementation.

4 Entity Relationship (ER) Diagram

4.1 ER Diagram of the System

The Entity-Relationship Diagram, generated using TikZ/LaTeX to ensure high fidelity and adherence to standard Chen notation, is presented below.

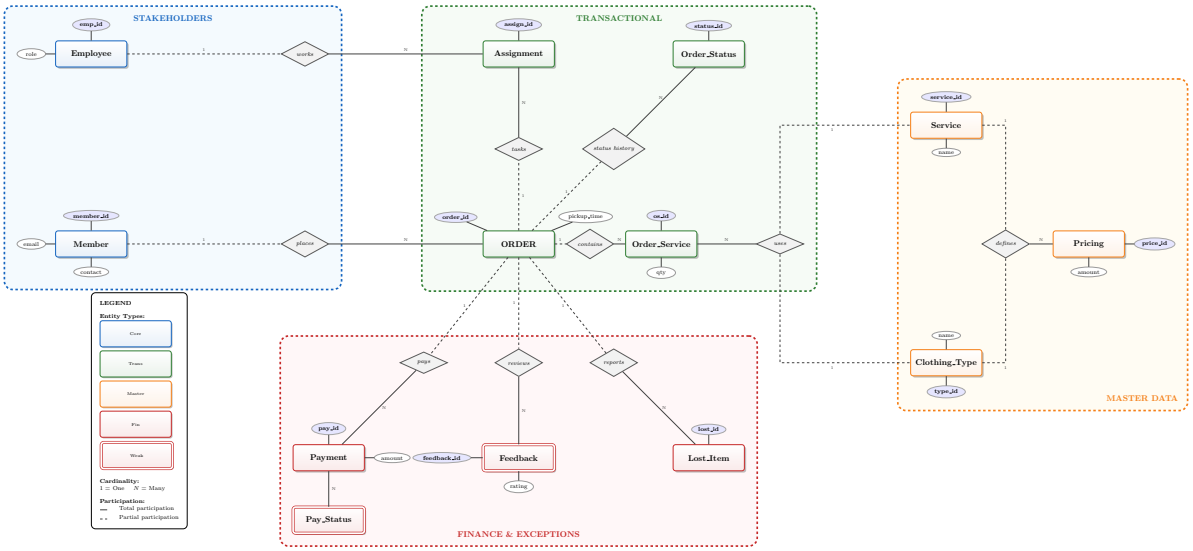

Figure 1: Entity Relationship Diagram for FreshWash System

4.2 Entities and Attributes

4.2.1 Stakeholders

- **Member (Strong Entity):** Represents the customer. Attributes: MemberID, Name, Email, Contact_Number, Address.
- **Employee (Strong Entity):** Represents staff. Attributes: EmployeeID, Name, Role, Phone, Joining_Date.

4.2.2 Master Data (Service Catalog)

- **Service:** Defines processing types (e.g., Dry Cleaning). Attributes: ServiceID, Service_Name, Base_Price.

- **Clothing_Type:** Categorizes items (e.g., Shirt, Saree). Attributes: TypeID, Type_Name, Wash_Instruction.
- **Pricing (Associative):** Determines cost based on Service and Clothing Type. Attributes: PriceID, Amount.

4.2.3 Transactional Operations

- **Order:** The central record. Attributes: OrderID, Order_Date, Delivery_Time, Total_Amount.
- **Order_Service (Weak):** Individual line items. Attributes: OrderServiceID, Quantity, Applied_Price.
- **Order_Status (Weak):** Lifecycle history. Attributes: StatusID, Status_Name, Timestamp.

4.3 Key Constraints & Assumptions

- **Pricing Logic:** A price cannot exist without referencing both a valid Service and a valid Clothing_Type.
- **Order Integrity:** An Order cannot be created without a valid MemberID.
- **Status History:** Both Orders and Payments maintain a historical log (1:N relationships) rather than overwriting a single status field, ensuring auditability.

5 ER Justification and Constraints

5.1 Conceptual Design Justification

- **Decomposition of Many-to-Many Relationships:** A direct M:N relationship between Service and Clothing Type would not allow for granular pricing. By introducing the *Pricing* associative entity, the system can define unique rates for specific combinations (e.g., the cost to "Dry Clean" a "Suit" differs from "Washing" a "Suit").
- **Usage of Weak Entities for History:** Instead of overwriting a single status field, *Order_Status* and *Payment_Status* are modeled as weak entities. This supports the requirement for an audit trail, tracking the exact timestamp of every lifecycle transition.
- **Order Detail Separation:** Specific items are separated from the main Order header into *Order_Service*. This normalization keeps the Order table compact while allowing variable numbers of line items per order.

5.2 Logical & Business Constraints

5.2.1 Domain Constraints

- **Pricing Validity:** The Amount in the Pricing entity must be strictly greater than zero (> 0).
- **Rating Range:** Feedback ratings are restricted to integers between 1 and 5.
- **Payment Consistency:** The recorded payment amount must not exceed the total amount of the associated order.

5.2.2 Temporal Constraints

- **Delivery Logic:** The `Expected_Delivery_Date` must strictly be later than the `Order_Date`.
- **Status Chronology:** Timestamps must appear in chronological order corresponding to valid lifecycle transitions.

6 Database Schema Implementation

6.1 Overview

The conceptual design phase culminated in the physical implementation of the FreshWash Laundry Management System using MySQL. The final schema consists of **13 normalized tables**, exceeding the assignment requirement. The database logic is enforced using strict data typing, primary/foreign key constraints, and check constraints.

6.2 Relational Schema Notation

In the schema definitions below, underlined attributes denote **Primary Keys**, while *italicized* attributes denote **Foreign Keys**.

1. **Member** (member_id, name, age, email, contact_number, address, created_at)
2. **Employee** (employee_id, employee_name, role, contact_number, joining_date)
3. **Service** (service_id, service_name, service_description, base_price)
4. **Clothing_Type** (type_id, type_name, wash_instruction)
5. **Price** (price_id, *service_id*, *type_id*, price)
6. **Laundry_Order** (order_id, *member_id*, order_date, pickup_time, expected_delivery_time, total_amount, current_status)
7. **Order_Service** (order_service_id, *order_id*, *service_id*, quantity, applied_price)
8. **Order_Assignment** (assignment_id, *order_id*, *employee_id*, assigned_role, assigned_date)
9. **Order_Status_Log** (status_id, *order_id*, status_name, status_timestamp)
10. **Payment** (payment_id, *order_id*, payment_mode, payment_amount, payment_date)
11. **Payment_Status** (payment_status_id, *payment_id*, status_name, status_timestamp)
12. **Feedback** (feedback_id, *member_id*, *order_id*, rating, comments, feedback_date)
13. **Lost_Item** (lost_id, *order_id*, item_description, reported_date, compensation_amount)

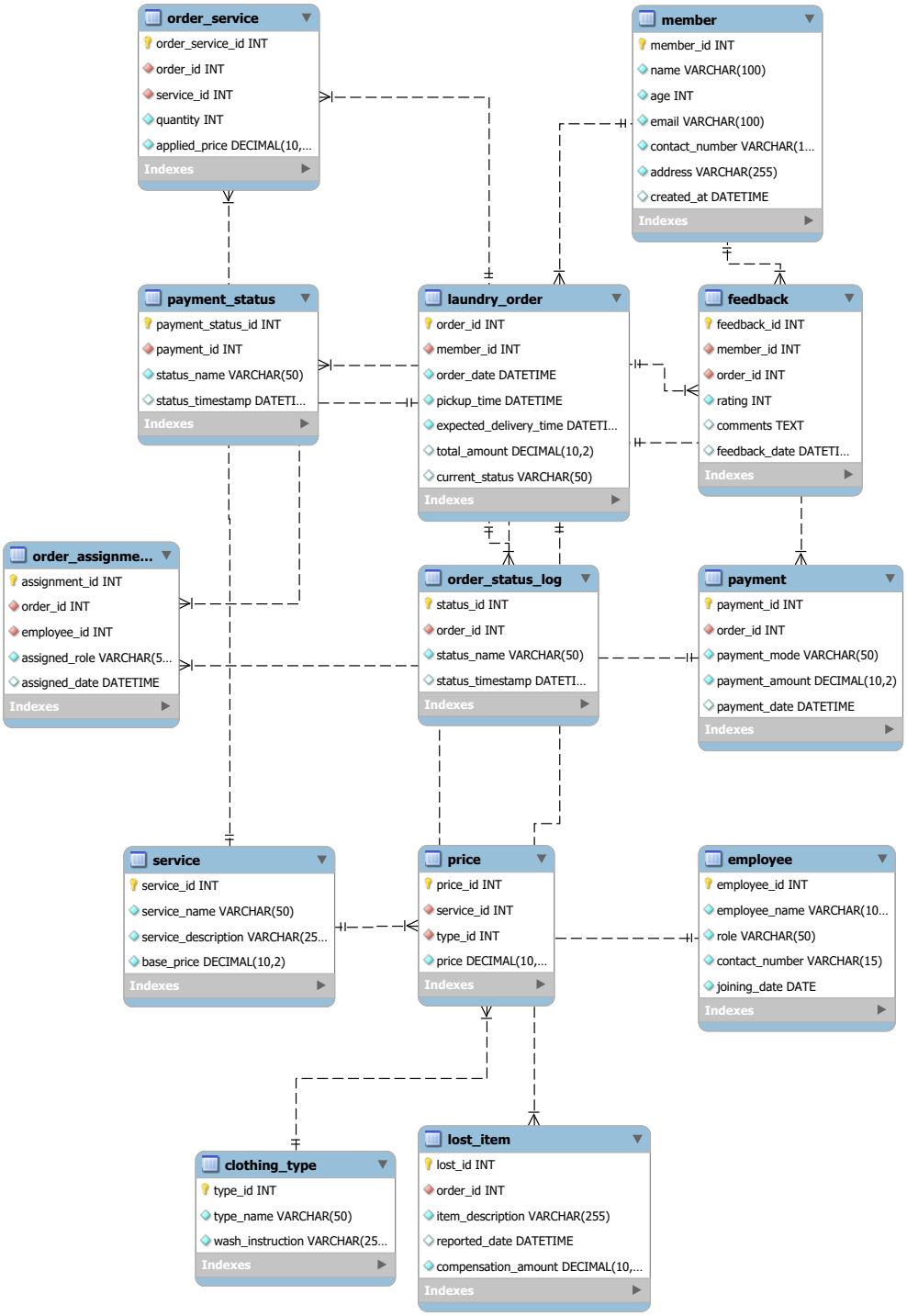

Figure 2: Physical Database Schema (Reverse Engineered from MySQL)

6.3 Data Dictionary and Constraints

- **Stakeholder Management:**

  - `age`: Enforced via `CHECK (age >= 18)` to ensure only adults register.
  - `email`: Enforced via `UNIQUE` constraints to prevent duplicate profiles.

- **Master Data:**
  - `UNIQUE(service_id, type_id)`: Ensures only one price exists per Service/Type combination.
  - `CHECK (price > 0)`: Prevents negative pricing errors.

- **Order Processing:**
  - `CHECK (expected_delivery_time > pickup_time)`: Prevents logical time-travel errors.
  - `CHECK (quantity > 0)`: Ensures line items represent physical goods.

All tables enforce a minimum of three NOT NULL attributes, including primary keys and mandatory business fields, ensuring no incomplete transactional records exist.

6.4 Sample Data Population

To verify the correctness of the schema and constraints, the database was populated with extensive sample data. This dataset includes realistic records for members, services, and orders, ensuring that all foreign key constraints and business rules (e.g., pricing logic) function as expected. The corresponding SQL data insertion scripts, which demonstrate successful order cycles and exception handling, are available in the project's GitHub repository.

6.5 Normalization and Referential Integrity

The schema is optimized to **Third Normal Form (3NF)**. Partial dependencies are removed by isolating pricing logic, and transitive dependencies are handled by separating status logs. Referential integrity is maintained via `ON DELETE CASCADE` for transactional data, ensuring that deleting an order automatically cleans up related payment and service logs, while protecting master data from accidental deletion.

7 Conclusion

7.1 Summary of Achievements

This project successfully designed and implemented a comprehensive backend for the FreshWash Laundry Management System. By adhering to a systematic "UML-first" design philosophy, we translated abstract business requirements into a robust Entity-Relationship model and subsequently into a fully normalized MySQL schema. Key achievements include:

- **Complex Schema Design:** Implementation of 13 normalized tables with complex relationships, exceeding the assignment requirements.
- **Data Integrity:** Enforcement of real-world business rules using SQL `CHECK` constraints and `UNIQUE` indices.
- **Auditability:** Creation of distinct audit trails for Order Status and Payment history to ensure transparency.
- **Scalability:** Verification of data integrity through rigorous testing of Primary and Foreign keys, ensuring the system can handle increased load.

8 Team Contributions

The FreshWash Laundry Management System was developed collaboratively, with each team member contributing to different phases of conceptual modeling, schema design, and implementation. The responsibilities were distributed as follows:

Kaushal

- Led the design of the Entity-Relationship (ER) Diagram using Chen notation.
- Contributed significantly to relational schema structuring and normalization.
- Ensured logical consistency between entities, relationships, and cardinalities.

Vedant

- Designed and documented the UML Class Diagram following a systematic UML-first approach.
- Contributed to relational schema structuring and ensured alignment between conceptual and logical models.
- Played a key role in defining relations between entities.

Anurag

- Responsible for database implementation using MySQL.
- Developed SQL scripts for table creation, constraints, and referential integrity enforcement.
- Verified schema correctness through test data insertion and validation queries.

Dhruv

- Led the preparation and structuring of the final technical report.
- Contributed to schema ideation and assisted in refining the conceptual design.
- Ensured proper documentation formatting and academic alignment with assignment requirements.

Pratik

- Contributed to database implementation and SQL scripting.
- Assisted in enforcing integrity constraints and validating multi-table relationships.
- Participated in testing and debugging schema implementation.

Honor Code Declaration

All team members affirm that this project was completed in accordance with the IITGN Honor Code. No unauthorized assistance or external solutions were used in the preparation of this assignment.