CheckInOut – Hostel Management System

CS 432 – Databases (Course Project - Assignment 1)

Module B: Conceptual Design (UML and ER Diagrams)

Semester II (2025–2026)

Date: 15 February 2026

Instructor: Dr. Yogesh K. Meena

Team Members & Contributions

Vipul Sunil Patil – Database schema design, constraints, ER diagram
Sai – Sample data, ER diagram documentation, report writing
Daksh Jain - ER diagram creation, relationship mapping
Daksh Dave - Data validation, testing, documentation
Harshit - Database schema design, constraint verification

GitHub Repository:

https://github.com/Harshitiitgn/CS432_Track1_Group10

Abstract

This report presents the conceptual design of the CheckInOut Hostel Management System developed for CS 432 – Databases (Module B). The system models hostel operations using Entity Relationship (ER) diagrams. The design focuses on room allocation management, asset tracking, complaint handling, visitor logging, and QR-based access auditing. The resulting schema enforces normalization, referential integrity, and realistic operational constraints suitable for institutional deployment.

Keywords

Database Design, ER Diagram, Hostel Management System, Conceptual Modeling

1 Introduction

Hostel operations in educational institutions often rely on fragmented tools or manual record keeping. Such approaches frequently lead to inconsistencies in room allocation, incomplete inventory tracking, delayed complaint resolution, and limited security auditing.

The CheckInOut Hostel Management System is designed as a centralized relational database to address these operational gaps. The objective of this project is to construct a structured conceptual design using ER principles that supports:

- Student check-in and check-out workflows
- Room allocation and capacity management
- Furniture inventory tracking
- Complaint and maintenance handling
- Visitor logging for security
- QR code–based access verification and logging

The conceptual model serves as the foundation for relational implementation and future system expansion.

2 System Overview and Design Objectives

The system is designed around three primary goals:

2.1 Operational Consistency

All hostel data must be stored in a structured relational form to prevent duplication and maintain referential integrity.

2.2 Traceability and Auditability

Key processes such as room allocation history, QR scans, complaints, and visitor entries must be fully logged to enable auditing and security analysis.

2.3 Scalability and Extensibility

The schema must allow future additions such as triggers, views, analytics modules, and user interfaces without structural redesign.

,
2026.

3 Entity Relationship (ER) Model

3.1 Complete ER Diagram

The ER diagram represents the finalized conceptual database design with all entities, relationships, cardinalities, and constraints clearly defined.

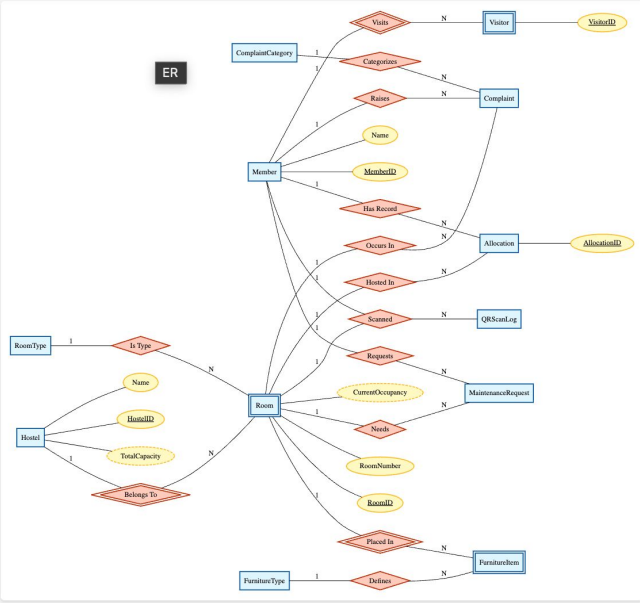

Figure 1: Complete Entity Relationship Diagram

ER Diagram Components:

The diagram includes the following 12 entities (exceeding the minimum requirement of 10 tables):

(1) **MEMBER** – Core entity storing student/resident information
(2) **HOSTEL** – Building-level information and capacity tracking
(3) **ROOMTYPE** – Lookup table for room classifications
(4) **ROOM** – Individual rooms with occupancy tracking
(5) **ALLOCATION** – Bridge entity linking members to rooms
(6) **FURNITURETYPE** – Lookup table for furniture categories
(7) **FURNITUREITEM** – Individual furniture assets
(8) **COMPLAINTCATEGORY** – Lookup table for complaint types
(9) **COMPLAINT** – User-reported issues
(10) **VISITOR** – Guest entry/exit logging
(11) **QRSCANLOG** – Security audit trail
(12) **MAINTENANCEREQUEST** – Work order tracking

Note on Weak Entities: Room, Visitor, and FurnitureItem are modeled as weak entities (double-border rectangles) in the ER diagram to reflect their logical dependencies on parent entities. However, the SQL implementation uses surrogate primary keys for technical efficiency while preserving these dependencies through foreign key constraints.

, ,

3.2 Key Relationships and Cardinalities

One-to-Many Relationships:

- `MEMBER (1) — (0..*) ALLOCATION`: One member can have multiple historical allocations
- `ROOM (1) — (0..*) ALLOCATION`: One room can be allocated to multiple members over time
- `HOSTEL (1) — (0..*) ROOM`: One hostel contains multiple rooms
- `ROOMTYPE (1) — (0..*) ROOM`: One room type defines multiple rooms
- `MEMBER (1) — (0..*) COMPLAINT`: One member can raise multiple complaints
- `ROOM (1) — (0..*) COMPLAINT`: One room can have multiple complaints
- `COMPLAINTCATEGORY (1) — (0..*) COMPLAINT`: One category includes multiple complaints
- `MEMBER (1) — (0..*) VISITOR`: One member can host multiple visitors
- `FURNITURETYPE (1) — (0..*) FURNITUREITEM`: One type defines multiple items
- `ROOM (1) — (0..*) FURNITUREITEM`: One room contains multiple furniture items
- `MEMBER (1) — (0..*) MAINTENANCEREQUEST`: One member can create multiple maintenance requests
- `ROOM (1) — (0..*) MAINTENANCEREQUEST`: One room can have multiple maintenance requests
- `MEMBER/ROOM (0..1) — (0..*) QRSCANLOG`: A scan log entry optionally references a specific Member or Room.

3.3 Relationship Examples from Sample Data

To illustrate the relationships, we provide concrete examples from the populated database:

Example 1: Historical Allocation Tracking

Member Rahul Sharma (ID: 1) demonstrates the `MEMBER-ALLOCATION` relationship:

- Current allocation: Room 101 (Active since 2025-07-15)
- Past allocation: Room 201 (Completed: 2024-07-10 to 2025-05-15)

This shows how one member can have multiple allocations over time, preserving complete occupancy history.

Example 2: Room-Based Complaint Tracking

Room 102 demonstrates the `ROOM-COMPLAINT` relationship:

- Complaint from Priya Patel: "Ceiling fan making noise"
- Complaint from Rohan Mehta: "Study table wobbling"

Multiple residents can raise different complaints about the same room, enabling comprehensive maintenance tracking.

Example 3: Furniture Inventory Management

Room 301 (Quad occupancy) contains multiple furniture items:

- 4 beds (BED-ABH301-001 to BED-ABH301-004)
- Each tracked individually with condition status
- One bed marked "Fair" with remark: "Mattress needs replacement soon"

This demonstrates individual asset tracking for maintenance planning.

Example 4: Hostel Structure

Aryabhatta Hostel (ID: 1) contains 7 rooms in the sample data:

- Rooms across multiple floors (G, 1, 2, 3)
- Different room types (Single, Double, Triple, Quad)
- Total capacity: 120 rooms (270 students) in full database

Example 5: Security Auditing via QR Scans

The QRScanLog shows access tracking:

- Member scan: MEM-QR-001 at Main Gate (2025-09-05 14:15)
- Room scan: ROOM-ABH-101-QR001 by Warden Kumar

This enables security analysis and access pattern monitoring.

3.4 ER Diagram Constraints and Integrity

The ER model enforces the following constraints:

- **Primary Keys (PK)**: Every entity has a unique identifier (underlined in diagrams)
- **Foreign Keys (FK)**: All relationships use foreign keys for referential integrity
- **Unique Keys (UK)**: Email, QRCode, IdentificationNumber, RoomNumber combinations
- **ENUM Types**: Status fields, gender, purpose of stay use controlled vocabularies
- **CHECK Constraints**:
  - CurrentOccupancy ≤ MaxCapacity (Room)
  - Age > 0 (Member)
  - CheckOutDate > CheckInDate (Allocation)
  - OutDateTime > InDateTime (Visitor)
- **NOT NULL**: At least 3 mandatory columns per table
- **ON DELETE Policies**: RESTRICT for critical relationships, CASCADE where appropriate

4 Justification of Key Design Decisions

4.1 Allocation as a Historical Bridge Entity

Decision: Model allocation as a separate entity rather than a simple foreign key in Member or Room.

Justification:

- Preserves complete check-in/check-out history
- Enables occupancy analytics over time
- Supports billing calculations based on duration
- Allows auditing of past room assignments
- Prevents data loss when students change rooms

Alternative Rejected: Adding `CurrentRoomID` to Member table would lose historical data and violate third normal form.

4.2 QRScanLog for Security Auditing

Decision: Create a dedicated log table for all QR code scans.

Justification:

- Provides complete audit trail for security purposes
- Records who scanned, what was scanned, when, and where
- Enables pattern analysis (e.g., unusual access times)
- Supports compliance requirements
- Independent of Member/Room tables for data integrity

Real-world Example: If a security incident occurs, administrators can query all scans in a specific time window and location to identify who had access.

4.3 FurnitureItem for Asset-Level Tracking

Decision: Track individual furniture pieces rather than just counts.

Justification:

- Enables damage attribution to specific items
- Supports maintenance scheduling and history
- Allows lifecycle tracking (purchase, warranty, disposal)
- Facilitates inventory audits at room level
- Serial number tracking for valuable items

Constraint Example: When a complaint is raised about furniture damage, we can link it to the specific `ItemID` and track repair history.

4.4 Complaint and Maintenance Separation

Decision: Separate Complaint and MaintenanceRequest into distinct entities.

Justification:

- Different workflows (user-initiated vs. administrative)
- Different priority schemes
- Different approval processes
- Allows complaints to exist without maintenance requests
- Enables scheduled maintenance independent of complaints

Relationship: One complaint may generate zero or one maintenance request, but maintenance requests can also exist independently (e.g., scheduled painting).

4.5 Visitor Logging for Safety Compliance

Decision: Mandatory visitor registration with entry/exit timestamps.

Justification:

- Security and safety compliance
- Emergency evacuation support
- Visitor pattern analysis
- Liability tracking
- Integration with gate pass systems

Constraint: `OutDateTime` must be greater than `InDateTime`, and both are linked to the host `MemberID`.

4.6 Normalization to Third Normal Form (3NF)

Decision: Normalize lookup tables (RoomType, FurnitureType, ComplaintCategory).

Justification:

- Eliminates data redundancy
- Ensures update consistency
- Reduces storage requirements
- Simplifies maintenance of controlled vocabularies
- Supports referential integrity

Example: Instead of storing "Single, Non-AC, 1-person" repeatedly in Room table, we reference `RoomTypeID` which points to the normalized `RoomType` table.

5 Constraints and Integrity Rules

The system enforces comprehensive constraints to maintain data quality and business logic:

5.1 Domain Constraints

- ENUM types for status fields (RoomStatus, AllocationStatus, ComplaintStatus)
- CHECK constraints for positive values (Age, Capacity, Floor)
- CHECK constraints for date validity (CheckOutDate > CheckInDate)
- CHECK constraints for capacity (CurrentOccupancy ≤ MaxCapacity)

5.2 Key Constraints

- Primary keys for unique identification
- Unique constraints on QRCode, Email, IdentificationNumber
- Composite unique keys (HostelID, RoomNumber)
- Foreign keys with appropriate cascading behavior

5.3 Referential Integrity

- ON DELETE RESTRICT for critical relationships (prevent orphaned records)
- ON UPDATE CASCADE for key modifications
- NOT NULL constraints for mandatory relationships

5.4 Complete Constraint Specifications

The following table lists all CHECK and UNIQUE constraints enforced in the SQL implementation:

Table 1: SQL Constraint Specifications

Entity	Constraint
MEMBER	Age > 0 YearOfStudy BETWEEN 1 AND 10
HOSTEL	All room counts 0 TotalRooms = Sum of room types TotalCapacity matches calculation TotalRooms, TotalCapacity 0
ROOMTYPE	UNIQUE(TypeName, BaseCapacity)
ROOM	MaxCapacity BETWEEN 1 AND 4 CurrentOccupancy MaxCapacity UNIQUE(HostelID, RoomNumber)
ALLOCATION	CheckOutDate CheckInDate UNIQUE(MemberID, AllocationStatus)
VISITOR	OutDateTime InDateTime

Key Constraint Types:

- **CHECK**: Validates data ranges and logical relationships
- **UNIQUE**: Prevents duplicate combinations
- **Composite UNIQUE**: Enforces business rules (e.g., single active allocation per member)

5.5 Business Logic Constraints

- Only one active allocation per member at a time
- Total rooms must equal sum of room type counts
- Total capacity must match calculated capacity
- Visitor exit time must be after entry time

6 Schema Statistics and Requirements Compliance

6.1 Assignment Requirements

Table 2: Assignment Requirements Compliance

Requirement	Required	Implemented
Minimum Tables	10	12
Minimum Entities	5	12
Minimum Functionalities	5	8+
Member Table with Attributes	Yes	Yes
Primary Keys in All Tables	Yes	Yes
Foreign Keys	Yes	Yes
NOT NULL Columns (min 3/table)	3+	3-8 per table
Unique Row Identification	Yes	Yes
Referential Integrity	Yes	Yes
Sample Data (10-20 rows)	Yes	Yes

6.2 Core Functionalities Implemented

(1) **Member Registration and Management** – Complete demographic data
(2) **Room Allocation and Check-in/Check-out** – Historical tracking
(3) **Furniture Inventory Management** – Individual item tracking
(4) **Complaint Handling System** – Category-based tracking with severity
(5) **Maintenance Request Workflow** – Assignment and status tracking
(6) **Visitor Registration and Logging** – Entry/exit timestamps
(7) **QR Code Security System** – Comprehensive scan logging
(8) **Hostel and Room Status Management** – Availability tracking

7 Conclusion

The CheckInOut Hostel Management System demonstrates a comprehensive and well-justified conceptual design that exceeds assignment requirements while maintaining practical applicability.

Key Achievements:

- 12 normalized tables with complete relationships
- Comprehensive ER diagram with all constraints
- Well-justified design decisions with real-world examples
- Complete referential integrity and constraint enforcement

By separating operational, infrastructural, and support components, the system achieves:

- Clear conceptual organization
- Third normal form (3NF) normalization
- Complete auditability through logging
- Scalability for future enhancements

The ER model ensures theoretical correctness, while features like QR logging, allocation history, and individual furniture tracking reflect real-world operational requirements. This design provides a robust foundation for:

- SQL implementation (Module A)
- Trigger and view development
- Analytics and reporting modules
- Web or mobile user interfaces
- Integration with external systems (gate access, billing)

The conceptual design is ready for deployment in institutional hostel environments with appropriate security, scalability, and maintainability characteristics.